

## 목차

변환 언어 참조-저작권.....	5
서문.....	9
Informatica 리소스.....	9
변환 언어.....	11
변환 언어 개요.....	11
식 구문.....	12
식에 설명 추가.....	15
예약어.....	15
상수.....	16
DD_DELETE.....	16
DD_INSERT.....	17
DD_REJECT.....	17
DD_UPDATE.....	18
FALSE.....	19
NULL.....	19
TRUE.....	20
연산자.....	21
연산자 우선 순위.....	21
산술 연산자.....	22
문자열 연산자.....	22
비교 연산자.....	23
논리 연산자.....	24
변수.....	24
기본 제공 변수.....	24
트랜잭션 제어 변수.....	29
로컬 변수.....	30
날짜.....	30
날짜 개요.....	30
날짜 형식 문자열.....	34
TO_CHAR 형식 문자열.....	35

TO_DATE 및 IS_DATE 형식 문자열.....	38
날짜 산술 이해.....	42
함수.....	42
함수 범주.....	42
ABORT.....	51
ABS.....	51
ADD_TO_DATE.....	52
AES_DECRYPT.....	55
AES_ENCRYPT.....	56
ASCII.....	57
AVG.....	58
BINARY_COMPARE.....	59
BINARY_CONCAT.....	60
BINARY_LENGTH.....	61
BINARY_SECTION.....	62
CEIL.....	64
CHOOSE.....	65
CHR.....	66
CHRCODE.....	67
COMPRESS.....	68
CONCAT.....	68
CONVERT_BASE.....	70
COS.....	71
COSH.....	72
COUNT.....	73
CRC32.....	75
CUME.....	76
DATE_COMPARE.....	77
DATE_DIFF.....	78
DEC_BASE64.....	81
DEC_HEX.....	82
DECODE.....	82
DECOMPRESS.....	85
EBCDIC_ISO88591.....	85
ENC_BASE64.....	86
ENC_HEX.....	87

오류.....	87
EXP.....	88
FIRST.....	89
FLOOR.....	91
FV.....	92
GET_DATE_PART.....	92
GREATEST.....	94
IIF.....	95
IN.....	98
INDEXOF.....	99
INITCAP.....	100
INSTR.....	101
ISNULL.....	104
IS_DATE.....	105
IS_NUMBER.....	107
IS_SPACES.....	109
LAG.....	110
LAST.....	112
LAST_DAY.....	113
LEAD.....	114
LEAST.....	115
LENGTH.....	116
LN.....	117
LOG.....	118
LOOKUP.....	119
LOWER.....	121
LPAD.....	122
LTRIM.....	123
MAKE_DATE_TIME.....	124
MAX(날짜).....	125
MAX(숫자).....	126
MAX(문자열).....	128
MD5.....	129
MEDIAN.....	130
METAPHONE.....	131
MIN(날짜).....	135

MIN(숫자).....	136
MIN(문자열).....	138
MOD.....	139
MOVINGAVG.....	140
MOVINGSUM.....	142
NPER.....	143
PERCENTILE.....	144
PMT.....	146
POWER.....	146
PV.....	147
RAND.....	148
RATE.....	149
REG_EXTRACT.....	149
REG_MATCH.....	152
REG_REPLACE.....	153
REPLACECHR.....	154
REPLACESTR.....	157
REVERSE.....	160
ROUND(날짜).....	161
ROUND(숫자).....	165
RPAD.....	168
RTRIM.....	169
SETCOUNTVARIABLE.....	170
SET_DATE_PART.....	172
SETMAXVARIABLE.....	175
SETMINVARIABLE.....	176
SETVARIABLE.....	178
SHA256.....	180
SIGN.....	181
SIN.....	182
SINH.....	183
SOUNDEX.....	184
SQRT.....	185
STDDEV.....	186
SUBSTR.....	188
SUM.....	191

SYSTIMESTAMP. ....	192
TAN. ....	193
TANH. ....	194
TIME_RANGE. ....	195
TO_BIGINT. ....	196
TO_CHAR(날짜). ....	198
TO_CHAR(숫자). ....	203
TO_DATE. ....	205
TO_DECIMAL. ....	208
TO_FLOAT. ....	209
TO_INTEGER. ....	210
TRUNC(날짜). ....	211
TRUNC(숫자). ....	214
UPPER. ....	216
VARIANCE. ....	217
사용자 지정 함수 작성. ....	218
사용자 지정 함수 작성 개요. ....	218
1단계. 리포지토리 ID 특성 가져오기. ....	219
2단계. 헤더 파일 작성. ....	220
3단계. 구현 파일 작성. ....	222
4단계. 모듈 작성. ....	233
5단계. 리포지토리 플러그 인 파일 작성. ....	235
6단계. 사용자 지정 함수 테스트. ....	238
사용자 지정 함수 설치. ....	239
사용자 지정 함수를 포함하는 식 작성. ....	240
사용자 지정 함수 API 참조. ....	240
사용자 지정 함수 API 참조 개요. ....	240
공통 API. ....	240
런타임 API. ....	244

## 변환 언어 참조-저작권

이 소프트웨어와 설명서는 사용 및 공개에 대한 제한 사항이 포함되어 있는 별도의 사용권 계약에 따라서만 제공됩니다. 본 문서의 어떤 부분도 Informatica LLC의 사전 통지 없이 어떠한 형태나 수단(전자적, 사진 복사, 녹음 등)으로 복제되거나 전송될 수 없습니다.

Informatica, Informatica 로고 및 PowerCenter는 미국과 전 세계 여러 관할 국가에서 Informatica LLC의 상표 또는 등록 상표입니다. Informatica 상표의 현재 목록은 <https://www.informatica.com/>

trademarks.html에서 확인할 수 있습니다. 다른 회사 및 제품명은 해당 소유자의 상표 또는 등록 상표일 수 있습니다.

미국 정부 권한. 미국 정부 고객에게 제공되는 프로그램, 소프트웨어, 데이터베이스, 관련 문서 및 기술 데이터는 해당하는 연방 입수 규정 및 기관별 보안 규정에 따라 "상용 컴퓨터 소프트웨어" 또는 "상용 기술 데이터"입니다. 따라서 사용, 복제, 공개, 수정 및 조정은 해당하는 정부 계약에 규정된 제한 사항 및 라이선스 조건을 따르며, 정부 계약 조건에 의해 적용 가능한 한도 내에서, FAR 52.227-19, 상용 소프트웨어 라이선스에 규정된 추가 권한이 적용됩니다.

이 소프트웨어 및/또는 설명서 중 일부는 타사 저작권의 적용을 받으며, 이에 국한되지 않습니다. 저작권 DataDirect Technologies. 모든 권리 보유. 저작권 (c) Sun Microsystems. 모든 권리 보유. 저작권 (c) RSA Security Inc. 모든 권리 보유. 저작권 (c) Ordinal Technology Corp. 모든 권리 보유. 저작권 (c) Aandacht c.v. 모든 권리 보유. 저작권 Genivia, Inc. 모든 권리 보유. 저작권 Isomorphic Software. 모든 권리 보유. 저작권 (c) Meta Integration Technology, Inc. 모든 권리 보유. 저작권 (c) Intalio. 모든 권리 보유. 저작권 (c) Oracle. 모든 권리 보유. 저작권 (c) Adobe Systems Incorporated. 모든 권리 보유. 저작권 (c) DataArt, Inc. 모든 권리 보유. 저작권 (c) ComponentSource. 모든 권리 보유. 저작권 (c) Microsoft Corporation. 모든 권리 보유. 저작권 (c) Rogue Wave Software, Inc. 모든 권리 보유. 저작권 (c) Teradata Corporation. 모든 권리 보유. 저작권 (c) Yahoo! Inc. 모든 권리 보유. 저작권 (c) Glyph & Cog, LLC. 모든 권리 보유. 저작권 (c) Thinkmap, Inc. 모든 권리 보유. 저작권 (c) Clearpace Software Limited. 모든 권리 보유. 저작권 (c) Information Builders, Inc. 모든 권리 보유. 저작권 (c) OSS Nokalva, Inc. 모든 권리 보유. 저작권 Edifecs, Inc. 모든 권리 보유. 저작권 Cleo Communications, Inc. 모든 권리 보유. 저작권 (c) International Organization for Standardization 1986. 모든 권리 보유. 저작권 (c) ej-technologies GmbH. 모든 권리 보유. 저작권 (c) Jaspersoft Corporation. 모든 권리 보유. 저작권 (c) International Business Machines Corporation. 모든 권리 보유. 저작권 (c) yWorks GmbH. 모든 권리 보유. 저작권 (c) Lucent Technologies. 모든 권리 보유. 저작권 (c) University of Toronto. 모든 권리 보유. 저작권 (c) Daniel Veillard. 모든 권리 보유. 저작권 (c) Unicode, Inc. 저작권 IBM Corp. 모든 권리 보유. 저작권 (c) MicroQuill Software Publishing, Inc. 모든 권리 보유. 저작권 (c) PassMark Software Pty Ltd. 모든 권리 보유. 저작권 (c) LogiXML, Inc. 모든 권리 보유. 저작권 (c) 2003-2010 Lorenzi Davide, 모든 권리 보유. 저작권 (c) Red Hat, Inc. 모든 권리 보유. 저작권 (c) The Board of Trustees of the Leland Stanford Junior University. 모든 권리 보유. 저작권 (c) EMC Corporation. 모든 권리 보유. 저작권 (c) Flexera Software. 모든 권리 보유. 저작권 (c) Jinfonet Software. 모든 권리 보유. 저작권 (c) Apple Inc. 모든 권리 보유. 저작권 (c) Telerik Inc. 모든 권리 보유. 저작권 (c) BEA Systems. 모든 권리 보유. 저작권 (c) PDFlib GmbH. 모든 권리 보유. 저작권 (c) Orientation in Objects GmbH. 모든 권리 보유. 저작권 (c) Tanuki Software, Ltd. 모든 권리 보유. 저작권 (c) Ricebridge. 모든 권리 보유. 저작권 (c) Sencha, Inc. 모든 권리 보유. 저작권 (c) Scalable Systems, Inc. 모든 권리 보유. 저작권 (c) jQWidgets. 모든 권리 보유. 저작권 (c) Tableau Software, Inc. 모든 권리 보유. 저작권 (c) MaxMind, Inc. 모든 권리 보유. 저작권 (c) TMate Software s.r.o. 모든 권리 보유. 저작권 (c) MapR Technologies Inc. 모든 권리 보유. 저작권 (c) Amazon Corporate LLC. 모든 권리 보유. 저작권 (c) Highsoft. 모든 권리 보유. 저작권 (c) Python Software Foundation. 모든 권리 보유. 저작권 (c) BeOpen.com. 모든 권리 보유. 저작권 (c) CNRI. 모든 권리 보유.

이 제품에는 Apache Software Foundation(<http://www.apache.org/>)에서 개발한 소프트웨어 및/또는 Apache License의 다양한 버전("라이선스")에 따라 사용이 허가된 기타 소프트웨어가 포함되어 있습니다. <http://www.apache.org/licenses/>에서 이러한 라이선스의 복사본을 얻을 수 있습니다. 관련 법규 또는 서면 동의에 명시되어 있지 않은 경우, 이러한 라이선스에 따라 배포되는 소프트웨어는 어떠한 종류의 명시적 이거나 묵시적인 보증 또는 조건 없이 "있는 그대로" 배포됩니다. 사용 권한에 대한 특정 언어별 라이선스 및 해당 라이선스에 따른 제한 사항을 참조하십시오.

이 제품에는 Mozilla(<http://www.mozilla.org/>)에서 개발한 소프트웨어, JBoss Group, LLC(저작권 JBoss Group, LLC, 모든 권리 보유.)가 저작권을 소유한 소프트웨어, Bruno Lowagie and Paulo Soares(저작권 (c) 1999-2006 by Bruno Lowagie and Paulo Soares)가 저작권을 소유한 소프트웨어 및 GNU Lesser General Public License Agreement(<http://www.gnu.org/licenses/lgpl.html>)의 다양한 버전에 따라 라이선스가 부여된 기타 소프트웨어가 포함되어 있습니다. 해당 정보는 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 이에 국한되지 않는 어떠한 종류의 명시적이거나 묵시적인 보증 없이 "있는 그대로" 제공되며, (c) Informatica는 어떠한 책임도 지지 않습니다.

이 제품에는 Douglas C. Schmidt와 Washington University, University of California, Irvine, Vanderbilt University의 연구팀(저작권 ((c)) 1993-2006, 모든 권리 보유.)이 저작권을 소유한 ACE(TM) 및 TAO(TM) 소프트웨어가 포함되어 있습니다.

이 제품에는 OpenSSL Toolkit(저작권 The OpenSSL Project. 모든 권리 보유.)에서 사용할 수 있도록 OpenSSL Project에서 개발한 소프트웨어가 포함되어 있으며 이 소프트웨어의 재배포는 <http://www.openssl.org> 및 <http://www.openssl.org/source/license.html>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Curl 소프트웨어(저작권 1996-2013, Daniel Stenberg, <daniel@haxx.se>. 모든 권리 보유.)가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한 사항은 <http://curl.haxx.se/docs/copyright.html>에 명시된 조항에 따라 변경될 수 있습니다. 위와 같은 저작권 고지 및 이러한 허가 고지가 모든 제품에 표시되어 있는 경우 목적 및 사용료 유무에 관계없이 이 소프트웨어를 사용, 복사, 수정 및 배포할 수 있는 사용 권한이 부여됩니다.

이 제품에는 MetaStuff, Ltd(저작권 2001-2005 ((C)) MetaStuff, Ltd. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.dom4j.org/license.html>의 조항에 따라 변경될 수 있습니다.

이 제품에는 The Dojo Foundation(저작권 (c) 2004-2007, The Dojo Foundation. 모든 권리 보유.)이 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://dojotoolkit.org/license>의 조항에 따라 변경될 수 있습니다.

이 제품에는 International Business Machines Corporation 등(저작권 International Business Machines Corporation and others. 모든 권리 보유.)이 저작권을 소유한 ICU 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://source.icu-project.org/repos/icu/icu/trunk/license.html>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Per Bothner(저작권 (c) 1996-2006 Per Bothner. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이러한 정보를 사용할 수 있는 권리는 <http://www.gnu.org/software/kawa/Software-License.html>의 라이선스에 설명되어 있습니다.

이 제품에는 OSSP UUID 소프트웨어(저작권 (c) 2002 Ralf S. Engelschall, 저작권 (c) 2002 The OSSP Project 저작권 (c) 2002 Cable & Wireless Deutschland)가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.opensource.org/licenses/mit-license.php>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Boost(<http://www.boost.org/>)에서 개발하거나 Boost 소프트웨어 라이선스에 따라 개발된 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 [http://www.boost.org/LICENSE\\_1\\_0.txt](http://www.boost.org/LICENSE_1_0.txt)의 조항에 따라 변경될 수 있습니다.

이 제품에는 University of Cambridge(저작권 (c) 1997-2007 University of Cambridge)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.pcre.org/license.txt>의 조항에 따라 변경될 수 있습니다.

이 제품에는 Eclipse Foundation(저작권 (c) 2007 The Eclipse Foundation. 모든 권리 보유.)이 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://www.eclipse.org/org/documents/epl-v10.php> 및 <http://www.eclipse.org/org/documents/edl-v10.php> 의 조항에 따라 변경될 수 있습니다.

이 제품에는 <http://www.tcl.tk/software/tcltk/license.html>, <http://www.bosrup.com/web/overlib/?License>, <http://www.stlport.org/doc/license.html>, <http://asm.ow2.org/license.html>, <http://www.cryptix.org/LICENSE.TXT>, <http://hsqldb.org/web/hsqldbLicense.html>, <http://httpunit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, [http://www.gzip.org/zlib/zlib\\_license.html](http://www.gzip.org/zlib/zlib_license.html), <http://www.openldap.org/software/release/license.html>, <http://www.libssh2.org>, <http://slf4j.org/license.html>, <http://www.sente.ch/software/OpenSourceLicense.html>, <http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-license-agreement>; <http://antlr.org/license.html>; <http://aopalliance.sourceforge.net/>; <http://www.bouncycastle.org/licence.html>; <http://www.jgraph.com/jgraphdownload.html>; <http://www.jcraft.com/jsch/LICENSE.txt>; [http://jotm.objectweb.org/bsd\\_license.html](http://jotm.objectweb.org/bsd_license.html); <http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231>, <http://www.slf4j.org/license.html>, <http://nanoxml.sourceforge.net/orig/copyright.html>, <http://www.json.org/license.html>, <http://forge.ow2.org/projects/javaservice/>, <http://www.postgresql.org/about/licence.html>, <http://www.sqlite.org/copyright.html>, <http://www.tcl.tk/software/tcltk/license.html>, <http://www.jaxen.org/faq.html>, <http://www.jdom.org/docs/faq.html>, <http://www.slf4j.org/license.html>, <http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License>, <http://www.keplerproject.org/md5/license.html>, <http://www.toedter.com/en/jcalendar/license.html>, <http://www.edankert.com/bounce/index.html>, <http://www.net-snmp.org/about/license.html>, <http://www.openmdx.org/#FAQ>, [http://www.php.net/license/3\\_01.txt](http://www.php.net/license/3_01.txt), <http://srp.stanford.edu/license.txt>, <http://www.schneier.com/blowfish.html>, <http://www.jmock.org/license.html>, <http://xsom.java.net>, <http://benalman.com/about/license/>, <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>, <http://www.h2database.com/html/license.html#summary>, <http://jsoncpp.sourceforge.net/LICENSE>, <http://jdbc.postgresql.org/license.html>, <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>, <https://github.com/rantav/hector/blob/master/LICENSE>, <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>, <http://jibx.sourceforge.net/jibx-license.html>, <https://github.com/lyokato/libgeohash/blob/master/LICENSE>, <https://github.com/hjiang/jsonxx/blob/master/LICENSE>, <https://code.google.com/p/lz4/>, <https://github.com/jedisct1/libsodium/blob/master/LICENSE>, <http://one-jar.sourceforge.net/index.php?page=documents&file=license>, <https://github.com/EsotericSoftware/kryo/blob/master/license.txt>, <http://www.scala-lang.org/license.html>, <https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt>, <http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>, <https://aws.amazon.com/asl/>, <https://github.com/twbs/bootstrap/blob/master/LICENSE> 및 <https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt>.

이 제품에는 Academic Free License(<http://www.opensource.org/licenses/afl-3.0.php>), Common Development and Distribution License(<http://www.opensource.org/licenses/cddl1.php>), Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), Sun Binary Code License Agreement Supplemental License Terms, BSD License(<http://www.opensource.org/licenses/bsd-license.php>), 새 BSD License(<http://opensource.org/licenses/BSD-3-Clause>), MIT License(<http://www.opensource.org/licenses/mit-license.php>), Artistic License(<http://www.opensource.org/licenses/artistic-license-1.0>) 및 Initial Developer's Public License 버전 1.0(<http://>



[www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/](http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/))에 따라 라이선스가 부여된 소프트웨어가 포함되어 있습니다.

이 제품에는 Joe Walnes와 XStream Committers(저작권 (c) 2003-2006 Joe Walnes, 2006-2007 XStream Committers. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한은 <http://xstream.codehaus.org/license.html>의 조항에 따라 변경될 수 있습니다. 이 제품에는 Indiana University Extreme! Lab에서 개발한 소프트웨어가 포함되어 있습니다. 자세한 내용을 확인하려면 <http://www.extreme.indiana.edu/>를 방문하십시오.

이 제품에는 Frank Balluffi 및 Markus Moeller(저작권 (c) 2013 Frank Balluffi and Markus Moeller. 모든 권리 보유.)가 저작권을 소유한 소프트웨어가 포함되어 있습니다. 이 소프트웨어와 관련된 사용 권한 및 제한 사항은 MIT license에 명시된 조항에 따라 변경될 수 있습니다.

#### 고지 사항

이 Informatica 제품(이하 "소프트웨어")에는 Progress Software Corporation(이하 "DataDirect")의 운영 회사인 DataDirect Technologies의 특정 드라이버(이하 "DataDirect Drivers")가 포함되어 있습니다. 이러한 드라이버에는 다음 조건이 적용됩니다.

1. DataDirect Drivers는 상품성, 특정 목적에의 적합성 및 비침해에 대한 묵시적 보증을 포함하여 이에 국한되지 않는 어떠한 종류의 명시적이거나 묵시적인 보증 없이 "있는 그대로" 제공됩니다.
2. DataDirect 또는 그 타사 공급자는 손해의 발생 가능성을 사전에 알고 있었는지 여부에 관계없이 ODBC 드라이버의 사용으로 발생하는 직접, 간접, 부수적, 특별, 결과적 또는 기타 손해에 대해 어떠한 경우에도 최종 사용자에게 책임을 지지 않습니다. 이러한 제한 사항은 계약 위반, 보증 불이행, 과실, 무과실 책임, 허위 진술 및 기타 불법 행위를 포함하여 이에 국한되지 않는 모든 소송 사유에 적용됩니다.

이 설명서의 정보는 예고 없이 변경될 수 있습니다. 이 문서에서 문제가 발견되는 경우 [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com)으로 보고해 주십시오.

Informatica 제품은 제품이 제공될 당시의 계약 조건에 따라 보증됩니다. Informatica는 상품성과 특정 목적에의 적합성에 대한 보증 그리고 비침해에 대한 보증 또는 조건을 포함하여 어떠한 종류의 명시적이거나 묵시적인 보증 없이 이 문서의 정보를 "있는 그대로" 제공합니다.

## 서문

*PowerCenter® 변환 언어 참조*에서는 PowerCenter의 변환 언어에 대해 설명합니다. 제약, 연산자, 변수, 날짜 및 함수를 사용하여 소스 데이터를 변환하는 방법을 알아볼 수 있습니다.

## Informatica 리소스

Informatica는 Informatica Network 및 기타 온라인 포털을 통해 다양한 범위의 제품 리소스를 제공합니다. 리소스를 통해 Informatica 제품 및 솔루션을 최대한 활용하고 다른 Informatica 사용자 및 주제별 전문가로부터 배울 수 있습니다.

### Informatica 네트워크

Informatica Network는 Informatica 기술 자료, Informatica 글로벌 고객 지원 센터 등 여러 리소스로 연결되는 관문입니다. Informatica Network를 시작하려면 <https://network.informatica.com>을 방문하십시오.

Informatica Network 멤버인 경우 다음 옵션이 가능합니다.

- 기술 자료에서 제품 리소스를 검색할 수 있습니다.
- 제품 사용 가능 여부에 대한 정보를 봅니다.

- 지원 사례를 생성하고 검토할 수 있습니다.
- 거주 지역의 Informatica 사용자 그룹 네트워크를 검색하고 동료와 협업 관계 유지

## Informatica 기술 자료

Informatica 기술 자료를 사용하여 사용 방법 문서, 모범 사례, 비디오 자습서, 자주 묻는 질문에 대한 답변 등 제품 리소스를 확인할 수 있습니다.

기술 자료를 검색하려면 <https://search.informatica.com>을 방문하십시오. 기술 자료에 대한 질문, 의견 또는 아이디어가 있는 경우 [KB\\_Feedback@informatica.com](mailto:KB_Feedback@informatica.com)을 통해 Informatica 기술 자료 팀에 문의해 주시기 바랍니다.

## Informatica 설명서

Informatica 설명서 포털에서 확장된 설명서 라이브러리를 탐색하여 현재 및 최근 제품 릴리스를 확인할 수 있습니다. 설명서 포털을 탐색하려면 <https://docs.informatica.com>을 방문하십시오.

제품 설명서에 대한 질문, 의견 또는 아이디어가 있는 경우 [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com)에서 Informatica 설명서 팀에 문의해 주시기 바랍니다.

## Informatica Product Availability Matrix

PAM(Product Availability Matrix)은 제품 릴리스에서 지원하는 운영 체제 버전, 데이터베이스 및 데이터 소스 유형과 대상을 나타냅니다.

<https://network.informatica.com/community/informatica-network/product-availability-matrices>에서 Informatica PAM을 찾을 수 있습니다.

## Informatica Velocity

Informatica Velocity는 수백 가지 데이터 관리 프로젝트의 실제 경험을 토대로 Informatica 전문 서비스업에서 개발한 팁과 모범 사례 모음입니다. Informatica Velocity는 전 세계의 조직과 협력하여 성공적인 데이터 관리 솔루션을 계획, 개발, 배포 및 유지 관리하는 Informatica 컨설턴트의 포괄적인 지식을 보여줍니다.

Informatica Velocity 리소스는 <http://velocity.informatica.com>에서 확인할 수 있습니다. Informatica Velocity에 대한 질문, 주석 또는 아이디어가 있으시면 Informatica 전문 서비스업([ips@informatica.com](mailto:ips@informatica.com))에 문의하십시오.

## Informatica Marketplace

Informatica Marketplace는 Informatica 구현을 확대 및 개선하기 위한 솔루션을 찾을 수 있는 포럼입니다. Marketplace에서 Informatica 개발자와 파트너가 제공하는 수백 개의 솔루션을 활용하여 생산성을 향상시키고 프로젝트의 구현에 걸리는 시간을 줄일 수 있습니다. <https://marketplace.informatica.com>에서 Informatica Marketplace를 찾을 수 있습니다.

## Informatica 글로벌 고객 지원 센터

전화 또는 Informatica 네트워크를 통해 글로벌 지원 센터에 문의할 수 있습니다.

해당 지역의 Informatica 글로벌 고객 지원 전화 번호는 Informatica 웹 사이트(<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>)를 방문하여 찾을 수 있습니다.

Informatica 네트워크에 대한 온라인 지원 리소스를 찾으려면 <https://network.informatica.com>으로 이동하고 eSupport 옵션을 선택하십시오.

## 변환 언어

### 변환 언어 개요

PowerCenter®는 소스 데이터 변환을 위한 SQL과 유사한 함수를 포함하는 변환 언어를 제공합니다. 이 함수를 사용하여 식을 작성하고 사용자 정의 함수라고 하는 함수를 작성할 수 있습니다.

사용자 정의 함수는 식 논리를 재사용하여 복합 식을 작성합니다. 사용자 정의 함수를 다른 사용자 정의 함수 또는 식에 포함할 수 있습니다. 사용자 정의 함수는 식의 지침과 동일한 지침을 따릅니다. 사용자 정의 함수와 식은 동일한 구문을 사용하며 동일한 변환 언어 구성 요소를 사용할 수 있습니다.

식은 데이터를 수정하거나 데이터가 조건과 일치하는지 여부를 테스트합니다. 예를 들어 AVG 함수를 사용하여 모든 직원의 평균 급여를 계산하거나 SUM 함수를 사용하여 특정 지점의 전체 매출을 계산할 수 있습니다.

ORDERS와 같은 포트만 포함하거나 10과 같은 숫자 리터럴만 포함하는 단순 식을 작성할 수 있습니다. 또한 함수 안에 함수를 중첩하거나 변환 언어 연산자를 사용하여 여러 개의 포트를 결합하는 복합 식을 작성할 수도 있습니다.

### 변환 언어 구성 요소

변환 언어에는 단순하거나 복잡한 변환 식을 작성하는 다음 구성 요소가 포함됩니다.

- **함수.** 100개 이상의 SQL 유사 함수를 사용하여 매핑의 데이터를 변경할 수 있습니다.
- **연산자.** 변환 연산자를 사용하여 수학적 계산, 데이터 결합 또는 데이터 비교를 수행하는 변환 식을 작성할 수 있습니다.
- **상수.** 기본 제공 상수를 사용하여 상수로 유지되는 값(예: TRUE)을 참조할 수 있습니다.
- **매핑 매개 변수 및 변수.** 매핑 또는 맵렛에 사용할 매핑 매개 변수를 작성하여 세션 전체에서 상수로 유지되는 값(예: 판매세율)을 참조할 수 있습니다. 맵렛 또는 매핑에 매핑 변수를 작성하여 세션 간에 변경되는 값을 참조하는 식을 작성할 수 있습니다.
- **워크플로우 변수.** 워크플로우에 사용할 워크플로우 변수를 작성하여 워크플로우 간에 변경되는 값을 참조하는 식을 작성할 수 있습니다.
- **기본 제공 변수 및 로컬 변수.** 기본 제공 변수를 사용하여 변하는 값(예: 시스템 날짜)을 참조하는 식을 쓸 수 있습니다. 로컬 변수를 변환에 작성할 수도 있습니다.
- **반환 값.** 반환 값 조회 변환을 포함하는 식을 작성할 수도 있습니다.

### 국제화 및 변환 언어

변환 언어 함수는 ASCII 또는 유니코드 데이터 이동 모드의 문자 데이터를 처리할 수 있습니다. 유니코드 모드에서는 다중 바이트 문자 데이터를 처리합니다. 다음 함수 및 변환의 반환 값은 PowerCenter 통합 서비스의 코드 페이지와 데이터 이동 모드에 따라 달라집니다.

- INITCAP
- LOWER

- UPPER
- MIN(날짜)
- MIN(숫자)
- MIN(문자열)
- MAX(날짜)
- MAX(숫자)
- MAX(문자열)
- IIF 및 DECODE 등 조건부 문을 사용하여 문자열을 비교하는 모든 함수

MIN 및 MAX는 PowerCenter 통합 서비스 코드 페이지와 관련된 정렬 순서에 따라 값을 반환하기도 합니다.

식 편집기에서 잘못된 식의 유효성을 검사하면 대화 상자에 오류 표시기인 “>>>>”와 함께 식이 표시됩니다. 이 표시기는 오류를 포함하는 식의 왼쪽에 표시되며 해당하는 식의 일부를 가리킵니다. 예를 들어  $a = b + c$  라는 식에서  $c$ 에 오류가 있는 경우 오류 메시지는 다음과 같이 표시됩니다.

$a = b + >>>> c$

문자 데이터를 평가하는 변환 언어 함수는 바이트 중심이 아니라 문자 중심입니다. 예를 들어 LENGTH 함수는 문자열의 바이트 수가 아니라 문자 수를 반환합니다. LOWER 함수는 PowerCenter 통합 서비스의 코드 페이지에 따라 문자열을 소문자로 반환합니다.

## 식 구문

변환 언어는 표준 SQL에 기반하지만 두 언어 사이에는 차이점이 있습니다. 예를 들어 SQL은 집계 함수에 대해 키워드 ALL 및 DISTINCT를 지원하지만 변환 언어는 아닙니다. 반대로 변환 언어는 집계 함수에 대해 선택적 필터 조건을 지원하지만 SQL은 아닙니다.

포트(예: ORDERS), 미리 정의된 워크플로우 변수(예: \$Start.Status) 또는 숫자 리터럴(예: 10)과 같이 단순한 식을 작성할 수 있습니다. 또한 변환 언어 연산자를 사용하여 함수 안에 함수가 중첩된 식 또는 여러 열을 결합한 식과 같이 복합 식을 작성할 수도 있습니다.

## 식 구성 요소

식은 다음 구성 요소의 모든 조합으로 구성될 수 있습니다.

- 포트(입력, 입력/출력, 변수)
- 문자열 리터럴, 숫자 리터럴
- 상수
- 함수
- 기본 제공 변수 및 로컬 변수
- 매핑 매개 변수 및 매핑 변수
- 미리 정의된 워크플로우 변수
- 사용자 정의 워크플로우 변수
- 연산자

- 반환 값

#### 포트 및 반환 값

연결되지 않은 변환의 포트 또는 반환 값을 포함하는 식을 작성하는 경우 다음 테이블의 참조 한정자를 사용합니다.

참조 한정자	설명
:EXT	외부 프로시저 변환의 반환 값을 포함하는 식을 작성하는 경우 필요합니다. 일반 구문은 다음과 같습니다. :EXT.external_procedure_transformation(argument1, argument2, ...)
:LKP	연결되지 않은 조회 변환의 반환 값을 포함하는 식을 작성하는 경우 필요합니다. 일반 구문은 다음과 같습니다. :LKP.lookup_transformation(argument1, argument2, ...) 인수는 조회 조건에 사용된 로컬 포트입니다. 순서는 변환의 포트 순서와 일치해야 합니다. 로컬 포트의 데이터 유형은 조회 조건에 사용된 조회 포트의 데이터 유형과 일치해야 합니다.
:SD	선택 사항입니다(PowerMart 3.5 식에만 해당). 식의 소스 테이블 포트를 한정합니다. 일반 구문은 다음과 같습니다. :SD.source_table.column_name
:SEQ	시퀀스 생성기 변환의 포트를 포함하는 식을 작성하는 경우 필요합니다. 일반 구문은 다음과 같습니다. :SEQ.sequence_generator_transformation.CURRVAL
:SP	연결되지 않은 저장 프로시저 변환의 반환 값을 포함하는 식을 작성하는 경우 필요합니다. 일반 구문은 다음과 같습니다. :SP.stored_procedure_transformation( argument1, argument2, [, PROC_RESULT]) 인수는 연결되지 않은 저장 프로시저 변환의 인수와 일치해야 합니다.
:TD	PowerMart 3.5 LOOKUP 함수의 대상 테이블을 참조하는 경우 필요합니다. 일반 구문은 다음과 같습니다. LOOKUP(:TD.SALES.ITEM_NAME, :TD.SALES.ITEM_ID, 10, :TD.SALES.PRICE, 15.99)

#### 문자열 및 숫자 리터럴

숫자 또는 문자열 리터럴을 포함할 수 있습니다.

문자열 리터럴은 작은따옴표로 묶어야 합니다. 예:

'Alice Davis'

문자열 리터럴은 대/소문자를 구분하며 작은따옴표를 제외한 모든 문자를 포함할 수 있습니다. 예를 들어 다음 문자열은 허용되지 않습니다.

'Joan's car'

작은따옴표를 포함하는 문자열을 반환하려면 다음과 같이 CHR 함수를 사용합니다.

'Joan' || CHR(39) || 's car'

숫자 리터럴에는 작은따옴표를 사용하지 마십시오. 포함할 숫자만 입력합니다. 예:

.05

또는

\$\$Sales\_Tax

## 식 구문에 대한 규칙 및 지침

식을 작성할 때는 다음 규칙 및 지침을 사용합니다.

- 집계 변환에는 단일 수준 집계 함수와 중첩된 집계 함수를 동시에 포함할 수 없습니다.
- 단일 수준 함수와 중첩된 함수를 모두 작성해야 하는 경우 개별 집계 변환을 작성해야 합니다.
- 숫자 식에는 문자열을 사용할 수 없습니다.  
예를 들어 식  $1 + '1'$ 은 올바르지 않습니다. 덧셈은 숫자 데이터 유형에만 수행할 수 있기 때문입니다. 정수와 문자열은 더할 수 없습니다.
- 문자열을 숫자 매개 변수로 사용할 수 없습니다.  
예를 들어 식 `SUBSTR(TEXT_VAL, '1', 10)`은 올바르지 않습니다. `SUBSTR` 함수의 시작 위치에는 문자열이 아닌 정수 값이 필요하기 때문입니다.
- 비교 연산자를 사용하는 경우 데이터 유형을 혼합할 수 없습니다.  
예를 들어 `10`진수 값을 문자열과 비교하는 식인 `123.4 = '123.4'`는 올바르지 않습니다.
- 포트, 리터럴 문자열 또는 숫자, 변수, 조회 변환, 저장 프로시저 변환, 외부 프로시저 변환 또는 다른 식의 결과 값을 전달할 수 있습니다.
- 포트 이름을 식에 입력하려면 식 편집기의 포트 탭을 사용합니다. 연결된 변환의 포트 이름을 변경하면 디자이너가 이름 변경 사항을 변환의 식에 전달합니다.
- 함수의 각 인수는 쉼표로 구분합니다.
- 변환 언어는 리터럴을 제외하고 대/소문자를 구분하지 않습니다.
- 디자이너 및 PowerCenter 통합 서비스는 리터럴을 제외하고 공백을 무시합니다.
- 콜론(:), 쉼표(,) 및 마침표(.)에는 특별한 의미가 있으므로 구문을 지정할 때만 사용해야 합니다.
- PowerCenter 통합 서비스는 대시(-)를 빼기 연산자로 처리합니다.
- 리터럴 값을 함수에 전달하는 경우 리터럴 문자열을 작은따옴표를 묶습니다. 리터럴 숫자에는 따옴표를 사용하지 마십시오. PowerCenter 통합 서비스는 작은따옴표로 묶은 모든 문자열 값을 문자열로 처리합니다.
- 매핑 매개 변수 또는 매핑 변수 또는 워크플로우 변수를 식 안의 함수에 전달하는 경우 따옴표를 사용하여 매핑 매개 변수 또는 매핑 변수 또는 워크플로우 변수를 지정하지 마십시오.
- 따옴표를 사용하여 포트를 지정하지 마십시오.
- 하나의 중첩된 집계 함수만 허용하는 집계 함수를 제외하고 식 안에 여러 함수를 중첩할 수 있습니다. PowerCenter 통합 서비스는 식을 평가할 때 가장 안쪽의 함수부터 평가합니다.

## 식에 설명 추가

변환 언어의 경우 두 개의 설명 지정자를 사용하여 식에 설명을 삽입할 수 있습니다.

- 다음과 같은 대시 2개:

```
-- These are comments
```

- 다음과 같은 슬래시 2개:

```
// These are comments
```

PowerCenter 통합 서비스는 이 두 개의 설명 지정자 뒤에 오는 행의 모든 텍스트를 무시합니다. 예를 들어 문자열 2개를 연결하려는 경우 설명을 포함하는 다음 식을 식의 중간에 입력할 수 있습니다.

```
-- This expression concatenates first and last names for customers:
FIRST_NAME -- First names from the CUST table
|| // Concat symbol
LAST_NAME // Last names from the CUST table
// Joe Smith Aug 18 1998
```

PowerCenter 통합 서비스는 설명을 무시하고 다음과 같이 식을 평가합니다.

```
FIRST_NAME || LAST_NAME
```

설명을 새 행으로 계속할 수는 없습니다.

```
-- This expression concatenates first and last names for customers:
FIRST_NAME -- First names from the CUST table
|| // Concat symbol
LAST_NAME // Last names from the CUST table
Joe Smith Aug 18 1998
```

이 경우 마지막 행이 올바른 식이 아니므로 디자이너 및 워크플로우 관리자가 식의 유효성을 검사하지 않습니다.

설명을 식에 포함하지 않으려는 경우 식 편집기에서 설명을 클릭하여 설명을 추가할 수 있습니다.

## 예약어

상수, 연산자 및 기본 제공 변수와 같은 변환 언어의 일부 키워드는 특정 함수를 위해 예약됩니다. 여기에는 다음이 포함됩니다.

- :EXT
- :INFA
- :LKP
- :MCR
- :SD
- :SEQ
- :SP
- :TD
- 
- AND
- DD\_DELETE

- DD\_INSERT
- DD\_REJECT
- DD\_UPDATE
- FALSE
- NOT
- NULL
- OR
- PROC\_RESULT
- SESSSTARTTIME
- SPOUTPUT
- SYSDATE
- TRUE
- WORKFLOWSTARTTIME

다음 단어는 워크플로우 식을 위해 예약됩니다.

- ABORTED
- DISABLED
- FAILED
- NOTSTARTED
- STARTED
- STOPPED
- SUCCEEDED

**참고:** 예약어는 포트 또는 로컬 변수의 이름으로 사용할 수 없습니다. 예약된 단어는 변환 및 워크플로우 식 안에서만 사용할 수 있습니다. 예약된 단어에는 식에서 미리 정의된 의미가 있습니다.

## 상수

### DD\_DELETE

업데이트 전략 식에서 레코드에 삭제 플래그를 지정합니다. DD\_DELETE는 정수 리터럴 2에 해당됩니다.

**참고:** DD\_DELETE 상수는 업데이트 전략 변환에만 사용됩니다. 정수 리터럴 2 대신 DD\_DELETE를 사용하면 복잡한 숫자 식의 문제를 쉽게 해결할 수 있습니다.

워크플로우를 실행할 때 이 플래그를 사용하여 대상에서 레코드를 삭제하려면 데이터 기반 업데이트 전략을 선택합니다.



## 예

다음 식은 ID 번호가 1001인 항목에 삭제를 표시하고 다른 모든 항목에는 삽입을 표시합니다.

```
IIF( ITEM_ID = 1001, DD_DELETE, DD_INSERT )
```

이 업데이트 전략 식은 숫자 리터럴을 사용하여 동일한 결과를 생성합니다.

```
IIF( ITEM_ID = 1001, 2, 0 )
```

**참고:** 상수를 사용하는 식이 숫자 리터럴을 사용하는 식보다 읽기가 쉽습니다.

## DD\_INSERT

업데이트 전략 식에서 레코드에 삽입 플래그를 지정합니다. DD\_INSERT는 정수 리터럴 0에 해당됩니다.

**참고:** DD\_INSERT 상수는 업데이트 전략 변환에만 사용됩니다. 정수 리터럴 0 대신 DD\_INSERT를 사용하면 복잡한 숫자 식의 문제를 쉽게 해결할 수 있습니다.

워크플로우를 실행할 때 이 플래그를 사용하여 대상에 레코드를 쓰려면 데이터 기반 업데이트 전략을 선택합니다.

## 예

다음 예에서는 영업 직원별 월 매출을 계산하는 매핑을 영업 직원 한 명의 매출을 조사할 수 있도록 수정합니다.

다음 업데이트 전략 식은 직원의 매출에 삽입 플래그를 지정하고 다른 모든 항목은 거부합니다.

```
IIF( EMPLOYEENAME = 'Alex', DD_INSERT, DD_REJECT )
```

이 업데이트 전략 식은 숫자 리터럴을 사용하여 동일한 결과를 생성합니다.

```
IIF( EMPLOYEENAME = 'Alex', 0, 3 )
```

**팁:** 상수를 사용하는 식이 숫자 리터럴을 사용하는 식보다 읽기가 쉽습니다.

다음 업데이트 전략 식은 SESSSTARTTIME을 사용하여 지난 2일 동안 출고된 주문을 찾고 이러한 주문에 삽입 플래그를 지정합니다. 이 식은 DATE\_DIFF를 사용하여 시스템 날짜에서 DATE\_SHIPPED를 빼고 이 두 날짜 간의 차이를 반환합니다. DATE\_DIFF가 배정밀도 값을 반환하므로 이 식은 TRUNC를 사용하여 차이를 잘라냅니다. 그런 다음 결과를 정수 리터럴 2와 비교합니다. 결과가 2보다 클 경우 해당 레코드에 거부 플래그를 지정합니다. 결과가 2 이하일 경우 삽입 플래그를 지정합니다.

```
IIF( TRUNC( DATE_DIFF( SESSSTARTTIME, ORDERS_DATE_SHIPPED, 'DD' ), 0 ) > 2, DD_REJECT, DD_INSERT )
```

## DD\_REJECT

업데이트 전략 식에서 레코드에 거부 플래그를 지정합니다. DD\_REJECT는 정수 리터럴 3에 해당됩니다.

**참고:** DD\_REJECT 상수는 업데이트 전략 변환에만 사용됩니다. 정수 리터럴 3 대신 DD\_REJECT를 사용하면 복잡한 숫자 식의 문제를 쉽게 해결할 수 있습니다.

워크플로우를 실행할 때 이 플래그를 사용하여 대상에서 레코드를 거부하려면 데이터 기반 업데이트 전략을 선택합니다.

DD\_REJECT를 사용하여 데이터를 필터링하거나 유효성을 검사할 수 있습니다. 레코드에 거부 플래그를 지정하면 PowerCenter 통합 서비스가 레코드를 건너뛰고 해당 레코드를 세션 거부 파일에 씁니다.

## 예제

다음 예에서는 현재 월의 매출을 계산하는 매핑을 양수 값만 포함하도록 수정합니다.

이 업데이트 전략 식은 0 미만의 레코드에 거부 플래그를 지정하고 다른 모든 레코드에는 삽입 플래그를 지정합니다.

```
IIF( SALES > 0, DD_INSERT, DD_REJECT )
```

이 식은 숫자 리터럴을 사용하여 동일한 결과를 생성합니다.

```
IIF( SALES > 0, 0, 3 )
```

상수를 사용하는 식이 숫자 리터럴을 사용하는 식보다 읽기가 쉽습니다.

다음 데이터 기반 예에서는 DD\_REJECT 및 IS\_SPACES를 사용하여 대상 테이블의 문자 열에 공백을 쓸 수 없도록 합니다. 이 식은 공백으로만 구성되는 레코드에 거부 플래그를 지정하고 다른 모든 레코드에 삽입 플래그를 지정합니다.

```
IIF( IS_SPACES( CUST_NAMES ), DD_REJECT, DD_INSERT )
```

## DD\_UPDATE

업데이트 전략 식에서 레코드에 업데이트 플래그를 지정합니다. DD\_UPDATE는 정수 리터럴 1에 해당됩니다.

**참고:** DD\_UPDATE 상수는 업데이트 전략 변환에만 사용합니다. 정수 리터럴 1 대신 DD\_UPDATE를 사용하면 복잡한 숫자 식의 문제를 쉽게 해결할 수 있습니다.

워크플로우를 실행할 때 이 플래그를 사용하여 대상에 레코드를 쓰려면 데이터 기반 업데이트 전략을 선택합니다.

## 예

다음 예에서는 현재 월의 매출을 계산하는 매핑을 수정합니다. 이 매핑은 직원 한 명의 매출을 로드합니다.

이 식은 Alex의 레코드에 업데이트 플래그를 지정하고 다른 모든 항목에는 거부 플래그를 지정합니다.

```
IIF( EMPLOYEENAME = 'Alex', DD_UPDATE, DD_REJECT )
```

이 식은 숫자 리터럴을 사용하여 Alex의 매출에 업데이트(1) 플래그를 지정하고 다른 모든 매출 레코드에 거부(3) 플래그를 지정하여 동일한 결과를 생성합니다.

```
IIF( EMPLOYEENAME = 'Alex', 1, 3 )
```

상수를 사용하는 식이 숫자 리터럴을 사용하는 식보다 읽기가 쉽습니다.

다음 업데이트 전략 식은 SYSDATE를 사용하여 지난 2일 동안 출고된 주문을 찾고 이러한 주문에 삽입 플래그를 지정합니다. 이 식은 DATE\_DIFF를 사용하여 시스템 날짜에서 DATE\_SHIPPED를 빼고 이 두 날짜 간의 차이를 반환합니다. DATE\_DIFF가 배정밀도 값을 반환하므로 이 식은 TRUNC를 사용하여 차이를 잘라냅니다. 그런 다음 결과를 정수 리터럴 2와 비교합니다. 결과가 2보다 클 경우 해당 레코드에 거부 플래그를 지정합니다. 결과가 2 이하일 경우 해당 레코드에 업데이트 플래그를 지정합니다. 그렇지 않은 레코드에는 거부 플래그를 지정합니다.

```
IIF( TRUNC( DATE_DIFF( SYSDATE, ORDERS_DATE_SHIPPED, 'DD' ) ) > 2, DD_REJECT, DD_UPDATE )
```

## FALSE

조건부 식을 표시합니다. FALSE는 정수 0에 해당됩니다.

### 예

다음 예에서는 DECODE 식에 FALSE를 사용하여 비교 결과를 바탕으로 값을 반환합니다. 단일 검색 값을 사용하여 여러 검색을 수행하는 경우 유용합니다.

```
DECODE( FALSE,  
  Var1 = 22, 'Variable 1 was 22!',  
  Var2 = 49, 'Variable 2 was 49!',  
  Var1 < 23, 'Variable 1 was less than 23.',  
  Var2 > 30, 'Variable 2 was more than 30.',  
  'Variables were out of desired ranges.')
```

## NULL

값을 알 수 없거나 정의되지 않음을 나타냅니다. NULL은 비어 있는 문자열(문자 열) 또는 0(숫자 열)과 같지 않습니다.

Null을 반환하는 식을 작성할 수는 있지만 NOT NULL 또는 PRIMARY KEY 제약 조건이 있는 모든 열은 Null을 허용하지 않습니다. 따라서 PowerCenter 통합 서비스가 이러한 제약 조건 중 하나를 포함하는 열에 Null 값을 쓰려고 하면 데이터베이스가 해당 행을 거부하고 PowerCenter 통합 서비스는 Null 값을 거부 파일에 쓰게 됩니다. 그러므로 변환을 작성할 때는 Null을 고려하십시오.

함수는 Null을 다르게 처리할 수 있습니다. Null 값을 전달받은 함수는 0 또는 NULL을 반환하거나 Null 값을 무시할 수 있습니다.

#### 관련 항목:

- [“함수” 페이지 42](#)

## 부울 식의 Null 값 작업

Null 값과 부울 식을 결합하는 식은 ANSI와 호환되는 결과를 생성합니다. 예를 들어 PowerCenter 통합 서비스는 다음 결과를 생성합니다.

- NULL AND TRUE = NULL
- NULL AND FALSE = FALSE

## 비교 식의 Null 값

비교 연산자를 포함하는 식에 Null 값을 사용하는 경우 PowerCenter 통합 서비스는 Null 값을 생성합니다. 그러나 비교 연산에서 Null 값을 큰 값 또는 작은 값으로 처리하도록 PowerCenter 통합 서비스를 구성할 수도 있습니다.

비교 연산자의 Null 값을 다음으로 처리 속성을 사용하면 PowerCenter 통합 서비스가 비교 식에서 Null 값을 처리하는 방식을 구성할 수 있습니다.

이 PowerCenter 통합 서비스 구성 속성은 식에서 다음 비교 연산자의 동작에 영향을 미칩니다.

=, !=, ^=, <>, >, >=, <, <=

예를 들어 다음 식을 고려하십시오.

```
NULL > 1
NULL = NULL
```

다음 테이블에는 PowerCenter 통합 서비스가 식을 평가하는 방식이 설명되어 있습니다.

식	비교 연산자의 Null 값을 다음으로 처리		
	NULL	HIGH	LOW
NULL > 1	NULL	TRUE	FALSE
NULL = NULL	NULL	TRUE	TRUE

## 집계 함수의 Null 값

PowerCenter 통합 서비스는 집계 함수의 Null 값을 Null로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. 집계 함수의 Null 값을 0으로 처리하거나 NULL로 처리하도록 PowerCenter 통합 서비스를 구성할 수 있습니다.

## 필터 조건의 Null 값

필터 조건이 NULL로 평가되는 경우 집계 함수는 레코드를 선택하지 않습니다. 필터 조건이 선택한 포트의 모든 레코드에 대해 NULL로 평가되는 경우 집계 함수는 NULL을 반환합니다(0을 반환하는 COUNT는 제외). 필터 조건을 집계 함수와 함께 사용하고 CUME, MOVINGAVG 및 MOVINGSUM 함수와 함께 사용할 수 있습니다.

## Null과 연산자

연산자(문자열 연산자 || 제외)를 사용하고 Null 값을 포함하는 모든 식은 항상 NULL로 평가됩니다. 예를 들어 다음 식은 NULL로 평가됩니다.

```
8 * 10 - NULL
```

Null 값이 있는지 테스트하려면 ISNULL 함수를 사용합니다.

## TRUE

비교 결과에 따라 값을 반환합니다. TRUE는 정수 1에 해당됩니다.

## 예

다음 예에서는 DECODE 식에 TRUE를 사용하여 비교 결과를 바탕으로 값을 반환합니다. 단일 검색 값을 사용하여 여러 검색을 수행하는 경우 유용합니다.

```
DECODE( TRUE,
  Var1 = 22, 'Variable 1 was 22!',
  Var2 = 49, 'Variable 2 was 49!',
  Var1 < 23, 'Variable 1 was less than 23.',
  Var2 > 30, 'Variable 2 was more than 30.',
  'Variables were out of desired ranges.')
```

# 연산자

## 연산자 우선 순위

변환 언어는 다양한 연산자의 사용 및 중첩된 식에서의 연산자 사용을 지원합니다.

여러 연산자를 포함하는 식을 작성하는 경우 PowerCenter 통합 서비스가 식을 평가하는 순서는 다음과 같습니다.

- 1.
2. 산술 연산자
3. 문자열 연산자
4. 비교 연산자
5. 논리 연산자

PowerCenter 통합 서비스는 다음 테이블에 표시되는 순서로 연산자를 평가합니다. 모든 연산자에 대해 동일하게 왼쪽부터 오른쪽의 우선 순위로 식의 연산자를 평가합니다.

다음 테이블에는 모든 변환 언어 연산자의 우선 순위가 나열되어 있습니다.

연산자	의미
( )	괄호.
+, -, NOT	단항 더하기 및 빼기와 논리적 NOT 연산자.
*, /, %	곱하기, 나누기, 모듈러스.
+, -	더하기, 빼기.
	연결.
<, <=, >, >=	보다 작음, 작거나 같음, 보다 큼, 크거나 같음.
=, <>, !=, ^=	같음, 같지 않음, 같지 않음, 같지 않음.
AND	논리적 AND 연산자, 조건을 지정할 때 사용.
또는	논리적 OR 연산자, 조건을 지정할 때 사용.

변환 언어는 중첩된 식에서의 연산자 사용도 지원합니다. 식에 괄호가 포함되는 경우 PowerCenter 통합 서비스는 괄호 안의 연산자를 먼저 평가한 다음 괄호 밖의 연산자를 평가합니다. 가장 안쪽 괄호 안의 연산자가 가장 먼저 평가됩니다.

예를 들어 수식  $8 + 5 - 2 * 8$ 은 연산을 중첩한 방식에 따라 다른 값을 반환합니다.

수식	반환 값
$8 + 5 - 2 * 8$	-3
$8 + (5 - 2) * 8$	32

## 산술 연산자

산술 연산자는 숫자 데이터에 대한 수학적 계산을 수행할 때 사용합니다.

다음 테이블에는 산술 연산자가 변환 언어에서의 우선 순위에 따라 나열되어 있습니다.

연산자	의미
+, -	단항 더하기 및 빼기. 단항 더하기는 양수 값을 나타냅니다. 단항 빼기는 음수 값을 나타냅니다.
*, /, %	곱하기, 나누기, 모듈러스. 모듈러스는 정수 2개를 나눈 나머지입니다. 예를 들어 $13 \% 2 = 1$ 입니다. 13을 2로 나누면 6이고 나머지는 1이기 때문입니다.
+, -	더하기, 빼기. 더하기 연산자(+)는 문자열을 연결하지 않습니다. 문자열을 연결하려면 문자열 연산자   을 사용합니다. 날짜 값에 산술 계산을 수행하려면 날짜 함수를 사용합니다.

Null 값에 산술 계산을 수행하면 함수가 NULL을 반환합니다.

식에서 산술 연산자를 사용할 때 식의 모든 피연산자는 숫자여야 합니다. 예를 들어 식  $1 + '1'$ 은 정수를 문자열에 더하는 것이므로 올바르지 않습니다. 식  $1.23 + 4 / 2$ 는 모든 피연산자가 숫자이므로 유효합니다.

**참고:** 변환 언어는 날짜/시간 값에 대한 산술 계산을 수행할 수 있는 기본 제공 날짜 함수를 제공합니다.

관련 항목:

- [“날짜 산술 이해” 페이지 42](#)

## 문자열 연산자

|| 문자열 연산자를 사용하면 문자열 2개를 연결할 수 있습니다. || 연산자는 모든 데이터 유형(이진 제외)의 피연산자를 문자열 데이터 유형으로 변환한 후에 연결합니다.

입력 값	반환 값
'alpha'    'betical'	alphabetical
'alpha'    2	alpha2
'alpha'    NULL	alpha

|| 연산자에는 선행 및 후행 공백이 포함됩니다. LTRIM 및 RTRIM 함수를 사용하면 선행 및 후행 공백을 잘라낸 후 문자열 2개가 연결됩니다.

## Null

|| 연산자는 Null 값을 무시합니다. 그러나 두 값이 모두 NULL인 경우 || 연산자는 NULL을 반환합니다.

## 예

다음 예는 두 열에 있는 직원의 이름과 성을 연결하는 식을 보여 줍니다. 이 식은 이름의 끝과 성의 시작에 있는 공백을 제거하고 각 이름의 끝에 공백을 연결한 다음 성을 연결합니다.

```
LTRIM( RTRIM( EMP_FIRST ) || ' ' || LTRIM( EMP_LAST ))
```

EMP_FIRST	EMP_LAST	RETURN VALUE
' Alfred'	' Rice '	Alfred Rice
' Bernice'	' Kersins'	Bernice Kersins
NULL	' Proud'	Proud
' Curt'	NULL	Curt
NULL	NULL	NULL

**참고:** CONCAT 함수를 사용하여 문자열 값 2개를 연결할 수도 있습니다. 그러나 || 연산자를 사용하는 것이 시간이 절약됩니다.

## 비교 연산자

비교 연산자는 문자 또는 숫자 문자열을 비교하고 데이터를 조작하고 TRUE(1) 또는 FALSE(0) 값을 반환합니다.

다음 테이블에는 변환 언어의 비교 연산자가 나열되어 있습니다.

연산자	의미
=	같음
>	보다 큼
<	보다 작음
>=	크거나 같음
<=	작거나 같음
<>	같지 않음
!=	같지 않음
^=	같지 않음

보다 큼(>) 및 보다 작음(<) 연산자는 특정 포트의 기본 키 정렬 순서에 따라 숫자 값을 비교하거나 행의 범위를 반환할 때 사용합니다.

비교 연산자를 식에 사용하는 경우 피연산자의 데이터 유형이 동일해야 합니다. 예를 들어 식 123.4 > '123'은 10진수와 문자열을 비교하는 식이므로 올바르지 않습니다. 식 123.4 > 123과 'a' != 'b'는 피연산자의 데이터 유형이 같으므로 유효합니다.

값을 Null 값과 비교하는 경우 결과는 NULL입니다.

필터 조건이 NULL로 평가되는 경우 통합 서비스는 NULL을 반환합니다.

## 논리 연산자

논리 연산자는 숫자 데이터를 조작할 때 사용합니다. 숫자 값을 반환하는 식은 0이 아닌 값의 경우 TRUE, 0인 경우 FALSE 및 NULL인 경우 NULL로 평가됩니다.

다음 테이블에는 변환 언어의 논리 연산자가 나열되어 있습니다.

연산자	의미
NOT	식의 결과를 부정합니다. 예를 들어 식이 TRUE로 평가되는 경우 연산자 NOT은 FALSE를 반환합니다. 식이 FALSE로 평가되는 경우 NOT은 TRUE를 반환합니다.
AND	두 조건 모두가 TRUE로 평가되는 경우 두 조건을 조인하고 TRUE를 반환합니다. 한 조건이 TRUE가 아닌 경우 FALSE를 반환합니다.
OR	한 조건이 TRUE로 평가되는 경우 두 조건을 연결하고 TRUE를 반환합니다. 두 조건이 모두 TRUE가 아닌 경우 FALSE를 반환합니다.

## Null

Null 값과 부울 식을 결합하는 식은 ANSI와 호환되는 결과를 생성합니다. 예를 들어 PowerCenter 통합 서비스는 다음 결과를 생성합니다.

- NULL AND TRUE = NULL
- NULL AND FALSE = FALSE

## 변수

### 기본 제공 변수

변환 언어는 기본 제공 변수를 제공합니다. 기본 제공 변수는 런타임 또는 시스템 정보를 반환합니다. 런타임 변수는 소스 및 대상 테이블 이름, 폴더 이름, 세션 실행 모드 및 워크플로우 실행 인스턴스 이름과 같은 정보를 반환합니다. 시스템 변수는 세션 시작 시간, 시스템 날짜 및 워크플로우 시작 시간을 반환합니다.

기본 제공 변수는 디자이너 또는 워크플로우 관리자에서 식에 사용할 수 있습니다. 예를 들어 DATE\_DIFF 함수에 시스템 변수인 SYSDATE를 사용할 수 있습니다. 런타임 변수는 식과 매핑 또는 워크플로우 변수를 허용하는 입력 필드에 사용할 수 있습니다. 예를 들어 런타임 변수 \$PMWorkflowRunInstanceName을 대상 출력 파일 이름의 일부로 사용할 수 있습니다. PowerCenter 통합 서비스에서는 기본 제공 변수의 값이 설정됩니다. 사용자는 워크플로우 또는 세션 매개 변수 파일에서 기본 제공 변수의 값을 정의할 수 없습니다.

기본 제공 변수는 식에 사용할 수 있습니다. 예를 들어 DATE\_DIFF 함수에 시스템 변수인 SYSDATE를 사용할 수 있습니다.



다음 기본 제공 변수는 런타임 정보를 제공합니다.

- \$PM<SourceName>@TableName, \$PM<TargetName>@TableName
- \$PMFolderName
- \$PMIntegrationServiceName
- \$PMMappingName
- \$PMRepositoryServiceName
- \$PMRepositoryUserName
- \$PMSessionName
- \$PMSessionRunMode
- \$PMWorkflowName
- \$PMWorkflowRunId
- \$PMWorkflowRunInstanceName

다음 기본 제공 변수는 시스템 정보를 제공합니다.

- \$\$\$SessStartTime
- SESSSTARTTIME
- SYSDATE
- WORKFLOWSTARTTIME

다음 테이블에는 디자이너 및 워크플로우 관리자에서 기본 제공 변수를 사용하는 위치가 설명되어 있습니다.

변수 이름	디자이너	워크플로우 관리자
\$PM<SourceName>@TableName, \$PM<TargetName>@TableName,	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> </ul>	<ul style="list-style-type: none"> <li>- 매핑 변수를 허용하는 입력 필드</li> </ul>
\$PMFolderName	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> <li>- 워크플로우 변수를 허용하는 입력 필드</li> </ul>	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> <li>- 워크플로우 변수를 허용하는 입력 필드</li> </ul>
\$PMIntegrationServiceName	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> <li>- 워크플로우 변수를 허용하는 입력 필드</li> </ul>	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> <li>- 워크플로우 변수를 허용하는 입력 필드</li> </ul>
\$PMMappingName	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> </ul>	<ul style="list-style-type: none"> <li>- 매핑 변수를 허용하는 입력 필드</li> </ul>
\$PMRepositoryServiceName	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> <li>- 워크플로우 변수를 허용하는 입력 필드</li> </ul>	<ul style="list-style-type: none"> <li>- 식</li> <li>- 매핑 변수를 허용하는 입력 필드</li> <li>- 워크플로우 변수를 허용하는 입력 필드</li> </ul>

변수 이름	디자이너	워크플로우 관리자
\$PMRepositoryUserName	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드
\$PMSessionName	- 식 - 매핑 변수를 허용하는 입력 필드	- 매핑 변수를 허용하는 입력 필드
\$PMSessionRunMode	- 식 - 매핑 변수를 허용하는 입력 필드	- 매핑 변수를 허용하는 입력 필드
\$PMWorkflowName	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드
\$PMWorkflowRunId	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드
\$PMWorkflowRunInstanceName	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드	- 식 - 매핑 변수를 허용하는 입력 필드 - 워크플로우 변수를 허용하는 입력 필드
\$\$\$SessStartTime	- 매핑 또는 맵렛 필터 조건 - 사용자 정의 조인 - SQL 재정의	- 매핑 또는 맵렛 필터 조건 - 사용자 정의 조인 - SQL 재정의
SESSSTARTTIME	- 식	해당 없음
SYSDATE	- 식	- 식
WORKFLOWSTARTTIME	해당 없음	- 식

## \$PM<SourceName>@TableName, \$PM<TargetName>@TableName

\$PM<SourceName>@TableName 및 \$PM<TargetName>@TableName은 관계형 소스 및 대상 인스턴스에 대한 대상 테이블 이름을 문자열 값으로 반환합니다. 이러한 변수는 문자열 데이터 유형을 허용하는 모든 함수와 함께 사용할 수 있습니다.

변수 이름은 소스 또는 대상 인스턴스 이름에 따라 다릅니다. 예를 들어 소스 인스턴스의 이름이 “Customers”인 경우 기본 제공 변수 이름은 \$PMCustomers@TableName입니다. 관계형 소스 또는 대상이 매핑 내 맵렛의 일부인 경우 기본 제공 변수 이름에 맵렛 이름이 포함됩니다.

- \$PM<MapletName>.<SourceName>@TableName
- \$PM<MapletName>.<TargetName>@TableName

매핑 또는 맵렛에 \$PM<SourceName>@TableName 및 \$PM<TargetName>@TableName을 사용합니다. 예를 들어 여러 관계형 소스를 포함하는 매핑에서 식 변환의 출력 포트에

\$PM<SourceName>@TableName을 사용하여 각 행의 소스 테이블 이름을 대상에 작성할 수 있습니다. 이러한 변수를 매핑 변수를 허용하는 입력 필드에 사용할 수도 있습니다.

## **\$PMFolderName**

**\$PMFolderName**은 리포지토리 폴더의 이름을 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 **\$PMFolderName**을 사용할 수 있습니다.

**\$PMFolderName**을 매핑, 맵렛, 워크플로우 링크 또는 워크플로우 태스크(예: 할당 또는 결정 태스크)에 사용할 수 있습니다. 매핑 또는 워크플로우 변수를 허용하는 입력 필드에도 **\$PMFolderName**을 사용할 수 있습니다.

## **\$PMIntegrationServiceName**

**\$PMIntegrationServiceName**은 세션을 실행하는 PowerCenter 통합 서비스의 이름을 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 **\$PMIntegrationServiceName**을 사용할 수 있습니다.

**\$PMIntegrationServiceName**은 PowerCenter 통합 서비스 이름을 문자열 값으로 반환합니다.

**\$PMIntegrationServiceName**을 매핑, 맵렛, 워크플로우 링크 또는 워크플로우 태스크(예: 할당 또는 결정 태스크)에 사용할 수 있습니다. 매핑 또는 워크플로우 변수를 허용하는 입력 필드에도 **\$PMIntegrationServiceName**을 사용할 수 있습니다.

## **\$PMMappingName**

**\$PMMappingName**은 매핑 이름을 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 **\$PMMappingName**을 사용할 수 있습니다.

**\$PMMappingName**을 매핑 또는 맵렛에 사용할 수 있습니다. **\$PMMappingName**을 매핑 변수를 허용하는 입력 필드에 사용할 수도 있습니다.

## **\$PMRepositoryServiceName**

**\$PMRepositoryServiceName**은 PowerCenter 리포지토리 서비스의 이름을 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 **\$PMRepositoryServiceName**을 사용할 수 있습니다.

**\$PMRepositoryServiceName**을 매핑, 맵렛, 워크플로우 링크 또는 워크플로우 태스크(예: 할당 또는 결정 태스크)에 사용할 수 있습니다. 매핑 또는 워크플로우 변수를 허용하는 입력 필드에도 **\$PMRepositoryServiceName**을 사용할 수 있습니다.

## **\$PMRepositoryUserName**

**\$PMRepositoryUserName**은 세션을 실행하는 리포지토리 사용자의 이름을 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 **\$PMRepositoryUserName**을 사용할 수 있습니다. **\$PMRepositoryUserName**은 리포지토리 사용자 이름을 문자열 값으로 반환합니다.

**\$PMRepositoryUserName**을 매핑, 맵렛, 워크플로우 링크 또는 워크플로우 태스크(예: 할당 또는 결정 태스크)에 사용할 수 있습니다. 매핑 또는 워크플로우 변수를 허용하는 입력 필드에도 **\$PMRepositoryUserName**을 사용할 수 있습니다.

## **\$PMSessionName**

**\$PMSessionName**은 세션 이름을 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 **\$PMSessionName**을 사용할 수 있습니다.

**\$PMSessionName**을 매핑 또는 맵렛에 사용할 수 있습니다. **\$PMSessionName**을 매핑 변수를 허용하는 입력 필드에 사용할 수도 있습니다.

## \$PMSessionRunMode

\$PMSessionRunMode는 세션 실행 모드(일반 또는 복귀)를 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 \$PMSessionRunMode를 사용할 수 있습니다.

\$PMSessionRunMode를 매핑 또는 맵릿에 사용할 수 있습니다. \$PMSessionRunMode를 매핑 변수를 허용하는 입력 필드에 사용할 수도 있습니다.

## \$PMWorkflowName

\$PMWorkflowName은 워크플로우의 이름을 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 \$PMWorkflowName을 사용할 수 있습니다.

\$PMWorkflowName을 매핑, 맵릿, 워크플로우 링크 또는 워크플로우 태스크(예: 할당 또는 결정 태스크)에 사용할 수 있습니다. 매핑 또는 워크플로우 변수를 허용하는 입력 필드에도 \$PMWorkflowName을 사용할 수 있습니다.

## \$PMWorkflowRunId

각 워크플로우 실행에는 고유한 실행 ID가 있습니다. \$PMWorkflowRunId는 워크플로우 실행 ID를 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 \$PMWorkflowRunId를 사용할 수 있습니다.

\$PMWorkflowRunId를 매핑, 맵릿, 워크플로우 링크 또는 워크플로우 태스크(예: 할당 또는 결정 태스크)에 사용할 수 있습니다. 매핑 또는 워크플로우 변수를 허용하는 입력 필드에도 \$PMWorkflowRunId를 사용할 수 있습니다. 예를 들어 동일한 인스턴스 이름으로 동시에 실행되도록 워크플로우를 구성하고 타사 응용 프로그램을 사용하여 각 워크플로우의 상태를 추적하려는 경우 \$PMWorkflowRunId를 세션 후 셸 명령에 사용하면 실행 ID를 응용 프로그램에 전달할 수 있습니다.

## \$PMWorkflowRunInstanceName

\$PMWorkflowRunInstanceName은 워크플로우 실행 인스턴스 이름을 문자열 값으로 반환합니다. 문자열 데이터 유형을 허용하는 모든 함수에 \$PMWorkflowRunInstanceName을 사용할 수 있습니다.

\$PMWorkflowRunInstanceName을 매핑, 맵릿, 워크플로우 링크 또는 워크플로우 태스크(예: 할당 또는 결정 태스크)에 사용할 수 있습니다. 매핑 또는 워크플로우 변수를 허용하는 입력 필드에도

\$PMWorkflowRunInstanceName을 사용할 수 있습니다. 예를 들어 고유한 인스턴스 이름이 있는 워크플로우의 경우 세션 속성의 대상 출력 파일 이름을 "OutFile\_\$PMWorkflowRunInstanceName.txt"로 설정하면 각 실행 인스턴스에 대해 고유한 대상 파일을 작성할 수 있습니다.

또는 세션 후 셸 명령을 사용하여 미리 정의된 이벤트 대기 태스크에 사용할 표시기 파일을 작성할 수 있습니다. 표시기 파일을 생성하는 셸 명령에서 \$PMWorkflowRunInstanceName을 표시기 파일 이름으로 사용하면 워크플로우 실행 인스턴스가 인스턴스를 실행하는 다른 워크플로우에 필요한 표시기 파일을 삭제하지 않습니다.

## SESSSTARTTIME

SESSSTARTTIME은 통합 서비스가 세션을 시작한 후 세션을 실행하는 노드의 현재 날짜 및 시간 값을 반환합니다. 변환 날짜/시간 데이터 유형을 허용하는 모든 함수에 SESSSTARTTIME을 사용할 수 있습니다.

SESSSTARTTIME은 변환 날짜/시간 데이터 유형 값으로 저장됩니다.

SESSSTARTTIME을 매핑 또는 맵릿에 사용할 수 있습니다. SESSSTARTTIME은 식 언어 안에서만 참조할 수 있습니다.

## 예

다음 식은 \$\$\$SessStartTime을 소스 한정자의 소스 필터 조건에 사용하여 증분 추출을 수행합니다. 이 식은 PowerCenter 통합 서비스가 세션을 시작하기 전에 모든 요일의 날짜 범위를 지정합니다. 이 식은 함수 DATE\_DIFF를 사용하여 값 ORDER\_DATE와 \$\$\$SessStartTime 간의 일 수 차이를 찾습니다. 두 날짜 간의 차이가 7일 이하일 경우 PowerCenter 통합 서비스가 소스에서 해당 행을 추출합니다.

```
DATE_DIFF(DAY, ORDER_DATE, '$$$SessStartTime') <= 7
```

## SYSDATE

SYSDATE는 변환을 통해 전달되는 각 행에 대한 세션을 실행하는 노드의 현재 날짜 및 시간을 초 단위까지 반환합니다. SYSDATE는 변환 날짜/시간 데이터 유형 값으로 저장됩니다.

고정 시스템 날짜를 캡처하려면 SYSDATE 대신 SESSSTARTTIME 변수를 사용합니다.

## 예

다음 식은 SYSDATE를 사용하여 지난 2일 동안 출고된 주문을 찾고 해당 주문에 삽입 플래그를 지정합니다. PowerCenter 통합 서비스는 DATE\_DIFF를 사용하여 시스템 날짜에서 DATE\_SHIPPED를 빼고 두 날짜 간의 차이를 반환합니다. DATE\_DIFF가 배정밀도 값을 반환하므로 이 식은 차이를 잘라냅니다. 그런 다음 결과를 정수 리터럴 2와 비교합니다. 결과가 2보다 클 경우 해당 행에 거부 플래그를 지정합니다. 결과가 2 이하일 경우 삽입 플래그를 지정합니다.

```
IIF( TRUNC( DATE_DIFF( SYSDATE, DATE_SHIPPED, 'DD' ),  
0 ) > 2, DD_REJECT, DD_INSERT
```

## WORKFLOWSTARTTIME

WORKFLOWSTARTTIME은 PowerCenter 통합 서비스가 워크플로우를 시작할 때 통합 서비스를 호스트하는 노드의 현재 날짜 및 시간을 반환합니다. 변환 날짜/시간 데이터 유형을 허용하는 모든 함수에 WORKFLOWSTARTTIME을 사용할 수 있습니다. WORKFLOWSTARTTIME은 변환 날짜/시간 데이터 유형 값으로 저장됩니다.

WORKFLOWSTARTTIME을 워크플로우 링크 및 태스크(예: 할당 및 결정 태스크)에 사용할 수 있습니다. WORKFLOWSTARTTIME은 식 언어 안에서만 참조할 수 있습니다.

## 예

다음 식은 WORKFLOWSTARTTIME을 사용하여 워크플로우 시작 시간과 워크플로우의 태스크 시작 시간 간의 차이(분)를 표시합니다. PowerCenter 통합 서비스는 SQL 함수 DATE\_DIFF를 사용하여 WORKFLOWSTARTTIME에서 태스크 시작 시간을 빼고 결과를 일 수로 반환합니다.

```
DATE_DIFF(WORKFLOWSTARTTIME, $s_EmployeeData.StartTime, 'MI')
```

## 트랜잭션 제어 변수

트랜잭션 제어 변수는 데이터베이스 행을 처리하는 동안 트랜잭션을 커밋하거나 롤백하는 조건을 정의합니다. 이러한 변수는 식 편집기에서 빌드한 트랜잭션 제어 식에 사용됩니다. 트랜잭션 제어 식은 IIF 함수를 사용하여 조건을 기준으로 각 행을 테스트합니다. PowerCenter 통합 서비스는 조건의 반환 값에 따라 행을 커밋 또는 롤백하거나 해당 행의 트랜잭션을 변경하지 않습니다.

다음 예에서는 트랜잭션 제어 변수를 사용하여 행 처리를 결정합니다.

IIF (NEWTRAN=1, TC\_COMMIT\_BEFORE, TC\_CONTINUE\_TRANSACTION)

NEWTRAN=1일 경우 TC\_COMMIT\_BEFORE 변수가 현재 행을 처리하기 전에 커밋이 수행되도록 합니다. 그렇지 않을 경우 TC\_CONTINUE\_TRANSACTION 변수가 현재 트랜잭션에서 행을 처리하도록 합니다.

트랜잭션 제어 식을 작성할 때 식 편집기에서 다음 변수를 사용할 수 있습니다.

- **TC\_CONTINUE\_TRANSACTION.** PowerCenter 통합 서비스가 현재 행에 대한 트랜잭션 변경을 수행하지 않습니다. 트랜잭션 제어 변수의 기본값입니다.
- **TC\_COMMIT\_BEFORE.** PowerCenter 통합 서비스가 트랜잭션을 커밋한 다음 새 트랜잭션을 시작하고 현재 행을 대상에 씁니다. 현재 행은 새 트랜잭션에 있습니다.
- **TC\_COMMIT\_AFTER.** PowerCenter 통합 서비스가 현재 행을 대상에 쓴 다음 트랜잭션을 커밋하고 새 트랜잭션을 시작합니다. 현재 행은 커밋된 트랜잭션에 있습니다.
- **TC\_ROLLBACK\_BEFORE.** PowerCenter 통합 서비스가 현재 트랜잭션을 롤백한 다음 새 트랜잭션을 시작하고 현재 행을 대상에 씁니다. 현재 행은 새 트랜잭션에 있습니다.

## 로컬 변수

로컬 변수를 매핑에 사용할 때는 해당 로컬 변수를 매핑의 변환 식에 사용합니다. 예를 들어 매핑 전체에서 복잡한 세금 계산을 사용하는 경우 이 식을 한 번만 작성하고 변수로 지정할 수 있습니다. 이렇게 하면 PowerCenter 통합 서비스가 계산을 한 번만 수행하면 되므로 성능이 개선됩니다.

로컬 변수는 저장 프로시저 식과 함께 사용하여 여러 반환 값을 캡처할 때 유용합니다.

## 날짜

### 날짜 개요

변환 언어는 날짜 변환을 수행하는 데 필요한 날짜 함수와 기본 제공 날짜 변수 집합을 제공합니다. 날짜 함수를 사용하면 날짜를 반올림하거나 잘라내거나 비교하거나 날짜의 한 부분을 추출하거나 날짜에 산술 계산을 수행할 수 있습니다. 날짜 함수에는 날짜 데이터 유형의 모든 값을 전달할 수 있습니다.

날짜 변수를 사용하면 PowerCenter 통합 서비스를 호스트하는 노드의 현재 날짜 또는 세션 시작 시간을 캡처할 수 있습니다.

변환 언어는 다음과 같은 형식 문자열 집합도 제공합니다.

- **날짜 형식 문자열.** 날짜 함수와 함께 사용하여 날짜의 부분을 지정합니다.
- **TO\_CHAR 형식 문자열.** 반환 문자열의 형식을 지정할 때 사용합니다.
- **TO\_DATE 및 IS\_DATE 형식 문자열.** 날짜로 변환하거나 테스트할 문자열의 형식을 지정할 때 사용합니다.

### 날짜/시간 데이터 유형

Informatica는 일반 데이터 유형을 사용하여 서로 다른 소스의 데이터를 변환합니다. 이러한 변환 데이터 유형에는 날짜/시간 값을 나노초 단위까지 지원하는 날짜/시간 데이터 유형이 포함됩니다. Informatica는 날짜를 이진 형식으로 내부에 저장합니다.

날짜 함수는 날짜/시간 값만 허용합니다. 문자열을 날짜 함수에 전달하려면 먼저 **TO\_DATE**를 사용하여 날짜/시간 값으로 변환합니다. 예를 들어 다음 식은 문자열 포트를 날짜/시간 값으로 변환한 다음 각 날짜에 1 개월을 더합니다.

```
ADD_TO_DATE( TO_DATE( STRING_PORT, 'MM/DD/RR'), 'MM', 1 )
```

일반 달력 시스템에서 서기 1년부터 9999년까지의 날짜를 사용할 수 있습니다.

## 율리우스일, 수정된 율리우스일 및 일반 달력

일반 달력 시스템의 날짜만 사용할 수 있습니다. 율리우스력의 날짜는 율리우스 적일이라고 하며 Informatica에서 지원되지 않습니다. 이 단어를 율리우스 일 또는 수정된 율리우스일과 혼동하지 마십시오.

J 형식 문자열을 사용하면 **MJD**(수정된 율리우스일) 형식을 조작할 수 있습니다. 특정 날짜의 **MJD**는 기원전 4713년 1월 1일 00:00:00(자정)부터 해당 날짜까지의 일 수입니다. 정의상 **MJD**에는 24시간의 일부를 나타내는 시간 구성 요소가 포함되며 10진수로 표시됩니다. J 형식 문자열은 이 시간 구성 요소를 변환하지 않습니다.

예를 들어 다음 **TO\_DATE** 식은 **SHIP\_DATE\_MJD\_STRING** 포트의 문자열을 기본 날짜 형식의 날짜 값으로 변환합니다.

```
TO_DATE (SHIP_DATE_MJD_STR, 'J')
```

SHIP_DATE_MJD_STR	RETURN_VALUE
2451544	Dec 31 1999 00:00:00.000000000
2415021	Jan 1 1900 00:00:00.000000000

SHIP_DATE_MJD_STR	RETURN_VALUE
2451544	Dec 31 1999 00:00:00.000000000
2415021	Jan 1 1900 00:00:00.000000000

J 형식 문자열은 날짜의 시간 부분을 포함하지 않으므로 반환 값의 시간이 00:00:00.000000000으로 설정됩니다.

J 형식 문자열을 **TO\_CHAR** 식에서 사용할 수도 있습니다. 예를 들어 J 형식 문자열을 **TO\_CHAR** 식에 사용하면 날짜 값이 문자열로 표시되는 **MJD** 값으로 변환됩니다. 예:

```
TO_CHAR(SHIP_DATE, 'J')
```

SHIP_DATE	RETURN_VALUE
Dec 31 1999 23:59:59	2451544
Jan 1 1900 01:02:03	2415021

**참고:** PowerCenter 통합 서비스는 **TO\_CHAR** 식에서 날짜의 시간 부분을 무시합니다.



## 2000년의 날짜

모든 변환 언어 날짜 함수는 2000년을 지원합니다. PowerCenter는 서기 1년부터 9999년까지의 날짜를 지원합니다.

### RR 형식 문자열

변환 언어는 2자리 연도를 포함하는 문자열을 날짜로 변환하는 RR 형식 문자열을 제공합니다. TO\_DATE 및 RR 형식 문자열을 사용하면 형식 MM/DD/RR의 문자열을 날짜로 변환할 수 있습니다. RR 형식 문자열은 현재 연도에 따라 데이터를 다르게 변환합니다.

- **현재 연도가 0과 49 사이인 경우.** 현재 연도가 0과 49 사이(예: 2003)이고 소스 문자열 연도가 0과 49 사이인 경우 PowerCenter 통합 서비스가 현재 세기와 소스 문자열의 2자리 연도를 반환합니다. 소스 문자열 연도가 50과 99 사이인 경우 통합 서비스는 이전 세기와 소스 문자열의 2자리 연도를 반환합니다.
- **현재 연도가 50과 99 사이인 경우.** 현재 연도가 50과 99 사이(예: 1998)이고 소스 문자열 연도가 0과 49 사이인 경우 PowerCenter 통합 서비스는 다음 세기와 소스 문자열의 2자리 연도를 반환합니다. 소스 문자열 연도가 50과 99 사이인 경우 PowerCenter 통합 서비스는 현재 세기와 지정된 2자리 연도를 반환합니다.

다음 테이블에는 RR 형식 문자열이 날짜로 변환되는 방식이 요약되어 있습니다.

현재 연도	소스 연도	RR 형식 문자열 반환
0-49	0-49	현재 세기
0-49	50-99	이전 세기
50-99	0-49	다음 세기
50-99	50-99	현재 세기

### 예

다음 식은 1950년과 2049년 사이인 모든 현재 연도에 대해 동일한 반환 값을 생성합니다.

```
TO_DATE( ORDER_DATE, 'MM/DD/RR' )
```

ORDER_DATE	RETURN_VALUE
'04/12/98'	04/12/1998 00:00:00.000000000
'11/09/01'	11/09/2001 00:00:00.000000000

### YY 형식 문자열과 RR 형식 문자열 간의 차이

PowerCenter는 YY 형식 문자열도 제공합니다. RR 및 YY 형식 문자열은 2자리 연도를 지정합니다. YY 및 RR 형식 문자열을 TO\_DATE를 제외한 모든 날짜 함수에 사용할 경우 동일한 결과를 생성합니다. TO\_DATE 식에서는 RR과 YY가 서로 다른 결과를 생성합니다.



다음 테이블은 각 형식 문자열이 반환하는 서로 다른 결과를 보여 줍니다.

문자열	현재 연도	TO_DATE(문자열, 'MM/DD/RR')	TO_DATE(문자열, 'MM/DD/YY')
04/12/98	1998	04/12/1998 00:00:00.000000000	04/12/1998 00:00:00.000000000
11/09/01	1998	11/09/2001 00:00:00.000000000	11/09/1901 00:00:00.000000000
04/12/98	2003	04/12/1998 00:00:00.000000000	04/12/2098 00:00:00.000000000
11/09/01	2003	11/09/2001 00:00:00.000000000	11/09/2001 00:00:00.000000000

2000년 및 그 이상의 날짜의 경우 YY 형식 문자열은 RR 형식 문자열보다 의미가 덜한 결과를 생성합니다. 21세기의 날짜에는 RR 형식 문자열을 사용합니다.

## 관계형 데이터베이스의 날짜

일반적으로 날짜는 날짜 및 시간 값을 포함하는 관계형 데이터베이스에 저장됩니다. 날짜에는 월, 일 및 연도가 포함되며 시간에는 시, 분, 초 및 초 미만 단위가 포함될 수 있습니다. 날짜/시간 데이터를 모든 날짜 함수에 전달할 수 있습니다.

## 플랫 파일의 날짜

TO\_DATE 함수를 사용하면 문자열을 날짜/시간 값으로 변환할 수 있습니다. 또한 IS\_DATE를 사용하여 문자열이 유효한 날짜인지 확인한 후에 TO\_DATE를 사용하여 변환할 수도 있습니다. 변환 언어 날짜 함수는 날짜 값만 허용합니다. 문자열을 날짜 함수에 전달하려면 먼저 TO\_DATE 함수를 사용하여 문자열을 변환 날짜/시간 데이터 유형으로 변환해야 합니다.

## 기본 날짜 형식

PowerCenter 통합 서비스는 기본 날짜 형식을 사용하여 날짜를 나타내는 문자열을 저장하고 조작합니다. 기본 날짜 형식을 지정하려면 세션 또는 세션 구성 개체에 대한 개체 구성 탭의 날짜/시간 형식 문자열 특성에 날짜 형식을 입력합니다. 기본적으로 날짜 형식은 MM/DD/YYYY HH24:MI:SS.US입니다.

Informatica는 날짜를 이진 형식으로 저장하므로 PowerCenter 통합 서비스에서 기본 날짜 형식을 사용하면 다음 작업을 수행해야 합니다.

- **날짜/시간 포트를 문자열 포트에 연결하여 날짜를 문자열로 변환합니다.** PowerCenter 통합 서비스가 세션 구성 개체에 정의된 날짜 형식의 문자열로 날짜를 변환합니다.
- **문자열 포트를 날짜/시간 포트에 연결하여 문자열을 날짜로 변환합니다.** PowerCenter 통합 서비스는 해당 문자열 값이 세션 구성 개체에 정의된 날짜 형식일 것으로 예상합니다. 입력 값이 이 형식과 일치하지 않거나 올바르지 않은 날짜인 경우 PowerCenter 통합 서비스가 행을 건너뛵니다. 문자열이 이 형식과 일치하는 경우 PowerCenter 통합 서비스가 문자열을 날짜 값으로 변환합니다.
- **날짜를 문자열로 변환하려면 TO\_CHAR(날짜, [형식\_string])을 사용합니다.** 형식 문자열을 생략한 경우 PowerCenter 통합 서비스가 세션 속성에 정의된 날짜 형식으로 문자열을 반환합니다. 형식 문자열을 지정한 경우 PowerCenter 통합 서비스가 지정된 형식으로 문자열을 반환합니다.
- **TO\_DATE(날짜, [형식\_string])를 사용하여 문자열을 날짜로 변환합니다.** 형식 문자열을 생략한 경우 PowerCenter 통합 서비스는 문자열이 세션 속성에 정의된 날짜 형식일 것으로 예상합니다. 형식 문자열을 지정한 경우 PowerCenter 통합 서비스는 문자열이 지정된 형식일 것으로 예상합니다.

기본 날짜 형식은 MM/DD/YYYY HH24:MI:SS.US이며 다음으로 구성됩니다.

- 월(1월 = 01, 9월 = 09)
- 일(월의 일)
- 연도(1998과 같이 4자리로 표시)
- 시(24시간 형식, 예: 12:00:00AM = 0, 1:00:00AM = 1, 12:00:00PM = 12, 11:00:00PM = 23)
- 분
- 초
- 마이크로초

## 날짜 형식 문자열

형식 문자열과 날짜 함수의 조합을 사용하여 입력 날짜를 평가할 수 있습니다. 날짜 형식 문자열은 국제화되지 않으므로 다음 테이블에 나열된 것과 같이 미리 정의된 형식으로 입력해야 합니다.

다음 테이블에는 날짜의 부분을 지정하는 형식 문자열이 요약되어 있습니다.

형식 문자열	설명
D, DD, DDD, DAY, DY, J	일(01-31). 날짜의 전체 일 부분을 지정할 때 사용하는 형식 문자열입니다. 예를 들어 12-APR-1997을 날짜 함수에 전달하는 경우 이 형식 중 하나를 사용하여 12를 지정합니다.
HH, HH12, HH24	시간(0-23)으로 0은 오전 12시(자정)를 나타냅니다. 날짜의 전체 시간 부분을 지정할 때 사용하는 형식입니다. 예를 들어 날짜 12-APR-1997 2:01:32 PM을 전달하는 경우 HH, HH12 또는 HH24를 사용하여 날짜의 시간 부분을 지정합니다.
MI	분(0-59).
MM, MON, MONTH	월(01-12). 날짜의 전체 월 부분을 지정할 때 사용하는 형식 문자열입니다. 예를 들어 12-APR-1997을 날짜 함수에 전달하는 경우 MM, MON 또는 MONTH를 사용하여 APR을 지정합니다.
MS	밀리초(0-999).
NS	나노초(0-999999999).
SS, SSSS	초(0-59).
KR	마이크로초(0-999999).
Y, YY, YYY, YYYY, RR	날짜의 연도 부분(0001-9999). 날짜의 전체 연도 부분을 지정할 때 사용하는 형식 문자열입니다. 예를 들어 12-APR-1997을 날짜 함수에 전달하는 경우 Y, YY, YYY 또는 YYYY를 사용하여 1997을 지정합니다.

**참고:** 형식 문자열은 대/소문자를 구분하지 않습니다. 형식 문자열은 작은따옴표로 묶어야 합니다.

다음 테이블에는 날짜 형식 문자열을 사용하여 입력 날짜를 평가하는 날짜 함수가 설명되어 있습니다.

함수	설명
ADD_TO_DATE	날짜에서 변경할 부분입니다.
DATE_DIFF	날짜에서 두 날짜 간의 차이를 계산하는 데 사용할 부분입니다.
GET_DATE_PART	날짜에서 반환할 부분입니다. 이 함수는 기본 날짜 형식을 바탕으로 정수 값을 반환합니다.
IS_DATE	검사할 날짜입니다.
ROUND	날짜에서 반올림할 부분입니다.
SET_DATE_PART	날짜에서 변경할 부분입니다.
SYSTIMESTAMP	타임스탬프 전체 자릿수입니다.
TO_CHAR(날짜)	문자열입니다.
TO_DATE	문자열입니다.
TRUNC(날짜)	날짜에서 잘라낼 부분입니다.

## TO\_CHAR 형식 문자열

TO\_CHAR 함수는 날짜/시간 데이터 유형을 사용자가 지정한 형식의 문자열로 변환합니다. 전체 날짜 또는 날짜의 일부를 문자열로 변환할 수 있습니다. 보고를 위한 형식으로 변경하기 위해 TO\_CHAR를 사용하여 날짜를 문자열로 변환할 수 있습니다.

TO\_CHAR는 일반적으로 대상이 플랫폼 파일이거나 날짜/시간 데이터 유형을 지원하지 않는 데이터베이스인 경우 사용됩니다.

다음 테이블에는 함수 TO\_CHAR의 날짜에 대한 형식 문자열이 요약되어 있습니다.

형식 문자열	설명
AM, A.M., PM, P.M.	자오선 표시기입니다. 이러한 형식 문자열 중 하나를 사용하여 AM 및 PM 시간을 지정합니다. AM 및 PM은 A.M. 및 P.M.과 동일한 값을 반환합니다.
D	주의 일입니다(1-7). 여기서 일요일은 1입니다.
DAY	요일입니다. 최대 9자까지 포함할 수 있습니다(예: Wednesday)
DD	월의 일입니다(01-31).
DDD	연도의 일입니다(001-366, 윤년 포함).
DY	3자로 축약된 요일 이름입니다(예: Wed).
HH, HH12	일의 시간입니다(01-12).

형식 문자	설명
HH24	일의 시간(00-23)으로 00은 오전 12시(자정)를 나타냅니다.
J	수정된 율리우스 일입니다. 달력 날짜를 수정된 율리우스일 값에 해당하는 문자열로 변환합니다. 율리우스일 값은 기원전 4713년 1월 1일 00:00:00부터 계산됩니다. 날짜의 시간 구성 요소는 무시됩니다. 예를 들어 식 TO_CHAR( SHIP_DATE, 'J' )는 Dec 31 1999 23:59:59를 문자열 2451544로 변환합니다.
MI	분(00-59).
MM	월(01-12).
MONTH	월의 이름입니다. 최대 9자까지 포함할 수 있습니다(예: January).
MON	3자로 축약된 월의 이름입니다(예: Jan).
MS	밀리초(0-999).
NS	나노초(0-999999999).
Q	분기입니다(1-4). 여기서 1-3월은 1에 해당됩니다.
RR	연도의 마지막 2자리입니다. 이 함수는 앞 자릿수를 제거합니다. 예를 들어 'RR'를 사용하고 1997을 전달할 경우 TO_CHAR는 97을 반환합니다. TO_CHAR와 함께 사용하는 경우 'RR'는 'YY'와 동일한 결과를 생성하며 상호 교환이 가능합니다. 그러나 TO_DATE와 함께 사용할 경우 'RR'은 가장 가까운 세기를 계산한 다음 해당 연도의 처음 두 자리를 제공합니다.
SS	초(00-59).
SSSSS	자정 이후의 초 수입니다(00000 - 86399). SSSSS를 TO_CHAR 식에 사용하는 경우 PowerCenter 통합 서비스는 날짜의 시간 부분만 평가합니다. 예를 들어 식 TO_CHAR(SHIP_DATE, 'MM/DD/YYYY SSSSS')는 12/31/1999 01:02:03을 12/31/1999 03723으로 변환합니다.
KR	마이크로초(0-999999).
연도	연도의 마지막 자리입니다. 이 함수는 앞 자릿수를 제거합니다. 예를 들어 'Y'를 사용하고 1997을 전달할 경우 TO_CHAR는 7을 반환합니다.
YY	연도의 마지막 2자리입니다. 이 함수는 앞 자릿수를 제거합니다. 예를 들어 'YY'를 사용하고 1997을 전달할 경우 TO_CHAR는 97을 반환합니다.
YYY	연도의 마지막 3자리입니다. 이 함수는 앞 자릿수를 제거합니다. 예를 들어 'YYY'를 사용하고 1997을 전달할 경우 TO_CHAR는 997을 반환합니다.
YYYY	날짜의 전체 연도 부분입니다. 예를 들어 'YYYY'를 사용하고 1997을 전달할 경우 TO_CHAR는 1997을 반환합니다.
W	월의 주입니다(1-5). 여기서 주 1은 월의 1일에 시작해서 7일에 끝나고 주 2는 8일에 시작해서 14일에 끝납니다. 예를 들어 Feb 1은 2월의 첫 번째 주를 나타냅니다.
WW	연도의 주입니다(01-53). 여기서 주 01은 1월 1일에 시작해서 1월 7일에 끝나고 주 02는 1월 8일에 시작해서 1월 14일에 끝납니다.

형식 문자	설명
- / . ; :	출력에 표시되는 구두점입니다. 이 기호를 날짜 부분을 구분하는 데 사용할 수 있습니다. 예를 들어 마침표로 날짜 부분을 구분하는 다음과 같은 식을 작성할 수 있습니다. TO_CHAR( DATES, 'MM.DD.YYYY' )
"text"	출력에 표시되는 텍스트입니다. 예를 들어 식 TO_CHAR( DATES, 'MM/DD/YYYY "Sales Were Up"' )으로 출력 포트를 생성하고 Apr 1 1997을 날짜로 전달하면 함수가 '04/01/1997 Sales Were Up' 문자열을 반환합니다. 리포지토리 코드 페이지에서 유효한 다중 바이트 문자를 입력할 수 있습니다.
" "	모호한 형식 문자열은 큰따옴표로 구분합니다(예: D" "DDD). 빈 따옴표는 출력에 표시되지 않습니다.

**참고:** 형식 문자열은 대/소문자를 구분하지 않습니다. 형식 문자열은 작은따옴표로 묶어야 합니다.

## 예

다음 예는 J, SSSSS, RR 및 YY 형식 문자열을 보여 줍니다. 자세한 예는 개별 함수를 참조하십시오.

**참고:** PowerCenter 통합 서비스는 TO\_CHAR 식에서 날짜의 시간 부분을 무시합니다.

### J 형식 문자열

J 형식 문자열을 TO\_CHAR 식에 사용하면 날짜 값이 문자열로 표시되는 MJD 값으로 변환됩니다. 예:

```
TO_CHAR(SHIP_DATE, 'J')
```

SHIP_DATE	RETURN_VALUE
Dec 31 1999 23:59:59	2451544
Jan 1 1900 01:02:03	2415021

### SSSSS 형식 문자열

형식 문자열 SSSSS를 TO\_CHAR 식에서 사용할 수도 있습니다. 예를 들어 다음 식은 SHIP\_DATE 포트의 날짜를 자정 이후의 전체 초 수를 나타내는 문자열로 변환합니다.

```
TO_CHAR( SHIP_DATE, 'SSSSS')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03	3723
09/15/1996 23:59:59	86399

### RR 형식 문자열

다음 식은 날짜를 MM/DD/YY 형식의 문자열로 변환합니다.

```
TO_CHAR( SHIP_DATE, 'MM/DD/RR')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03	12/31/99
09/15/1996 23:59:59	09/15/96
05/17/2003 12:13:14	05/17/03

### YY 형식 문자열

TO\_CHAR 식에서 YY 형식 문자열은 RR 형식 문자열과 동일한 결과를 생성합니다. 다음 식은 날짜를 MM/DD/YY 형식의 문자열로 변환합니다.

```
TO_CHAR( SHIP_DATE, 'MM/DD/YY')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03	12/31/99
09/15/1996 23:59:59	09/15/96
05/17/2003 12:13:14	05/17/03

## TO\_DATE 및 IS\_DATE 형식 문자열

TO\_DATE 함수는 사용자가 지정한 형식의 문자열을 날짜/시간 값으로 변환합니다. TO\_DATE는 일반적으로 플랫폼 파일의 문자열을 날짜/시간 값으로 변환할 때 사용합니다. TO\_DATE 형식 문자열은 국제화되지 않으므로 미리 정의된 형식으로 입력해야 합니다.

**참고:** TO\_DATE와 IS\_DATE는 동일한 형식 문자열 집합을 사용합니다.

TO\_DATE 식을 작성하는 경우 소스 문자열 날짜의 각 부분에 대한 형식 문자열을 사용합니다. 소스 문자열 형식과 형식 문자열이 일치해야 합니다. 날짜 구분 기호는 수행할 날짜 유효성 검사에 대해 일치하지 않아도 됩니다. 일치하지 않는 부분이 있을 경우 PowerCenter 통합 서비스가 문자열을 변환하지 않고 행을 건너뛰는 것입니다. 형식 문자열을 생략하는 경우 소스 문자열은 세션에서 지정한 날짜 형식이어야 합니다.

IS\_DATE는 값이 유효한 날짜인지 여부를 나타냅니다. 유효한 날짜는 세션에서 지정한 날짜 형식의 모든 문자열입니다. 테스트하려는 문자열이 지정된 날짜 형식이 아닌 경우 "TO\_DATE 및 IS\_DATE 형식 문자열" 테이블에 나열된 문자열 형식을 사용합니다. 문자열 형식이 지정된 형식과 일치하지 않는 경우 또는 문자열이 유효한 날짜를 나타내지 않는 경우 이 함수는 FALSE(0)를 반환합니다. 문자열 형식이 문자열의 지정된 형식과 일치하고 유효한 날짜인 경우 이 함수는 TRUE(1)를 반환합니다. IS\_DATE 형식 문자열은 국제화되지 않으므로 다음 테이블에 나열된 형식 중 하나로 입력해야 합니다.

다음 테이블에는 함수 TO\_DATE 및 IS\_DATE에 대한 형식 문자열이 나와 있습니다.

**테이블 1. TO\_DATE 및 IS\_DATE 형식 문자열**

형식 문자열	설명
AM, a.m., PM, p.m.	자오선 표시기입니다. 이러한 형식 문자열 중 하나를 사용하여 AM 및 PM 시간을 지정합니다. AM 및 PM은 a.m. 및 p.m.과 동일한 값을 반환합니다.
DAY	요일입니다. 최대 9자까지 포함할 수 있습니다(예: Wednesday). DAY 형식 문자열은 대/소문자를 구분하지 않습니다.
DD	월의 일입니다(1-31).
DDD	연도의 일입니다(001-366, 윤년 포함).
DY	3자로 축약된 요일 이름입니다(예: Wed). DY 형식 문자열은 대/소문자를 구분하지 않습니다.
HH, HH12	일의 시간입니다(1-12).
HH24	일의 시간(0-23)으로 0은 오전 12시(자정)를 나타냅니다.
J	수정된 율리우스 일입니다. MJD 형식의 문자열을 날짜 값으로 변환합니다. 소스 문자열의 시간 구성 요소는 무시되며 모든 날짜에 00:00:00.000000000의 시간이 할당됩니다. 예를 들어 식 TO_DATE('2451544', 'J')는 2451544를 Dec 31 1999 00:00:00.000000000으로 변환합니다.
MI	분(0-59).
MM	월(1-12).
MONTH	월의 이름입니다. 최대 9자까지 포함할 수 있습니다(예: August). 대/소문자를 구분하지 않습니다.
MON	3자로 축약된 월의 이름입니다(예: Aug). 대/소문자를 구분하지 않습니다.
MS	밀리초(0-999).
NS	나노초(0-999999999).
RR	4자리 연도입니다(예: 1998, 2034). 소스 문자열에 2자리 연도가 포함된 경우 사용합니다. TO_DATE와 함께 사용하면 2자리 연도가 4자리 연도로 변환됩니다. <ul style="list-style-type: none"> <li>- 현재 연도가 50과 99 사이인 경우. 현재 연도가 50과 99 사이(예: 1998)이고 소스 문자열의 연도 값이 0과 49 사이인 경우 PowerCenter 통합 서비스가 다음 세기와 소스 문자열의 2자리 연도를 반환합니다. 소스 문자열의 연도 값이 50과 99 사이인 경우 PowerCenter 통합 서비스는 현재 세기와 지정된 2자리 연도를 반환합니다.</li> <li>- 현재 연도가 0과 49 사이인 경우. 현재 연도가 0과 49 사이(예: 2003)이고 소스 문자열 연도가 0과 49 사이인 경우 PowerCenter 통합 서비스는 현재 세기와 소스 문자열의 2자리 연도를 반환합니다. 소스 문자열 연도가 50과 99 사이인 경우 PowerCenter 통합 서비스는 이전 세기와 소스 문자열의 2자리 연도를 반환합니다.</li> </ul>
SS	초(0-59).

형식 문자열	설명
SSSSS	밤 12시 이후의 초입니다. SSSSS를 TO_DATE 식에 사용하면 PowerCenter 통합 서비스가 날짜의 시간 부분만 평가합니다. 예를 들어 식 TO_DATE( DATE_STR, 'MM/DD/YYYY SSSSS')는 12/31/1999 3783을 12/31/1999 01:02:03으로 변환합니다.
KR	마이크로초(0-999999).
연도	PowerCenter 통합 서비스를 실행하는 노드의 현재 연도의 마지막 자리를 문자열 값으로 바꿉니다.
YY	PowerCenter 통합 서비스를 실행하는 노드의 현재 연도의 마지막 2자리를 문자열 값으로 바꿉니다.
YYY	PowerCenter 통합 서비스를 실행하는 노드의 현재 연도의 마지막 3자리를 문자열 값으로 바꿉니다.
YYYY	4자리 연도입니다. 2자리 연도를 전달할 경우 이 형식 문자열을 사용하지 마십시오. 대신 RR 또는 YY 형식 문자열을 사용하십시오.

## 날짜 형식 문자열의 규칙 및 지침

다음은 날짜 형식 문자열 작업에 사용하는 규칙 및 지침입니다.

- TO\_DATE 문자열의 형식은 형식 문자열과 일치해야 합니다. 일치하지 않는 경우 PowerCenter 통합 서비스가 정확하지 않은 값을 반환하거나 행을 건너뛸 수 있습니다. 예를 들어 2020년 5월 12일을 나타내는 문자열 '20200512'를 TO\_DATE에 전달하는 경우 형식 문자열 YYYYMMDD를 포함해야 합니다. 형식 문자열을 포함하지 않은 경우 PowerCenter 통합 서비스는 해당 문자열이 세션에서 지정한 날짜 형식일 것으로 예상합니다. 마찬가지로 형식 문자열과 일치하지 않는 문자열을 전달하는 경우 PowerCenter 통합 서비스는 오류를 반환하고 행을 건너뛸 수 있습니다. 예를 들어 문자열 2020120을 TO\_DATE에 전달하고 형식 문자열 YYYYMMDD를 포함하는 경우 문자열이 형식 문자열과 일치하지 않으므로 PowerCenter 통합 서비스가 오류를 반환하고 행을 건너뛸 수 있습니다.
- 형식 문자열은 작은따옴표로 묶어야 합니다.
- PowerCenter 통합 서비스는 세션에 지정된 기본 날짜/시간 형식을 사용합니다. 기본값은 MM/DD/YYYY HH24:MI:SS.US입니다. 형식 문자열은 대/소문자를 구분하지 않습니다.

## 예

다음 예는 J, RR 및 SSSSS 형식 문자열을 설명합니다. 자세한 예는 개별 함수를 참조하십시오.



### J 형식 문자열

다음 식은 SHIP\_DATE\_MJD\_STRING 포트의 문자열을 기본 날짜 형식의 날짜 값으로 변환합니다.

TO\_DATE (SHIP\_DATE\_MJD\_STR, 'J')

SHIP_DATE_MJD_STR	RETURN_VALUE
2451544	Dec 31 1999 00:00:00.000000000
2415021	Jan 1 1900 00:00:00.000000000

J 형식 문자열은 날짜의 시간 부분을 포함하지 않으므로 반환 값의 시간이 00:00:00.000000000으로 설정됩니다.

### RR 형식 문자열

다음 식은 문자열을 4자리 연도 형식으로 변환합니다. 현재 연도는 1998입니다.

TO\_DATE( DATE\_STR, 'MM/DD/RR')

DATE_STR	RETURN VALUE
04/01/98	04/01/1998 00:00:00.000000000
08/17/05	08/17/2005 00:00:00.000000000

### YY 형식 문자열

다음 식은 문자열을 4자리 연도 형식으로 변환합니다. 현재 연도는 1998입니다.

TO\_DATE( DATE\_STR, 'MM/DD/YY')

DATE_STR	RETURN VALUE
04/01/98	04/01/1998 00:00:00.000000000
08/17/05	08/17/1905 00:00:00.000000000

**참고:** 두 번째 행의 경우 RR은 2005년을 반환하지만 YY는 1905년을 반환합니다.

### SSSSS 형식 문자열

다음 식은 자정 이후의 초 수를 포함하는 문자열을 날짜 값으로 변환합니다.

TO\_DATE( DATE\_STR, 'MM/DD/YYYY SSSSS')

DATE_STR	RETURN_VALUE
12/31/1999 3783	12/31/1999 01:02:03.000000000
09/15/1996 86399	09/15/1996 23:59:59.000000000

## 날짜 산술 이해

변환 언어는 다음과 같이 날짜/시간 값에 대한 산술 계산을 수행할 수 있는 기본 제공 날짜 함수를 제공합니다.

- **ADD\_TO\_DATE.** 날짜의 특정 부분을 더하거나 뺍니다.
- **DATE\_DIFF.** 두 날짜를 뺍니다.
- **SET\_DATE\_PART.** 날짜의 한 부분을 변경합니다.

날짜를 더하거나 뺄 때 숫자 산술 연산자(예: + 또는 -)를 사용할 수 없습니다.

변환 언어는 윤년을 인식하며 서기 0001년 1월 1일 00:00:00.000000000부터 서기 9999년 12월 31일 23:59:59.999999999까지의 날짜를 허용합니다.

**참고:** 변환 언어는 변환 날짜/시간 데이터 유형을 사용하여 날짜 값을 지정합니다. 날짜 함수는 날짜/시간 값에만 사용할 수 있습니다.

## 함수

이 장에는 변환 언어의 함수 지원에 대한 정보가 포함되어 있습니다.

### 함수 범주

변환 언어는 다음 유형의 함수를 제공합니다.

- 집계
- 이진
- 문자
- 변환
- 데이터 정리
- 날짜
- 인코딩
- 재무
- 숫자
- 지수
- 특수
- 문자열
- 테스트
- 변수
- 창

## 집계 함수

집계 함수는 선택한 포트에서 Null이 아닌 값에 대한 요약 값을 반환합니다. 집계 함수를 사용하면 다음을 수행할 수 있습니다.

- 그룹의 모든 행에 대한 단일 값을 계산합니다.
- 집계 변환에서 각 그룹에 대한 단일 값을 반환합니다.
- 필터를 적용하여 선택한 포트의 특정 행에 대한 값을 계산합니다.
- 함수 내 산술 계산을 수행하려면 연산자를 사용합니다.
- 단일 패스의 동일한 소스 열에서 파생된 둘 이상의 집계 값을 계산합니다.

변환 언어에는 다음 집계 함수가 포함됩니다.

- ANY
- AVG
- COUNT
- FIRST
- LAST
- MAX (날짜)
- MAX(숫자)
- MAX(문자열)
- MEDIAN
- MIN(날짜)
- MIN(숫자)
- MIN(문자열)
- PERCENTILE
- STDDEV
- SUM
- VARIANCE

PowerCenter 통합 서비스를 유니코드 모드에서 실행되도록 구성한 경우 MIN 및 MAX는 세션 속성에서 정의한 코드 페이지의 정렬 순서에 따라 값을 반환합니다.

집계 함수를 집계 변환에서 사용할 수 있습니다. 집계 함수 안에는 하나의 집계 함수만 중첩할 수 있습니다. PowerCenter 통합 서비스는 가장 안쪽의 집계 함수 식을 평가한 다음 이 결과를 사용하여 바깥쪽의 집계 함수 식을 평가합니다. ID를 기준으로 그룹화하고 2개의 집계 함수를 중첩하는 다음과 같은 집계 변환을 설정할 수 있습니다.

`SUM( AVG( earnings ) )`

여기서 데이터 집합에는 다음 값이 포함됩니다.

ID	EARNINGS
1	32

ID	EARNINGS
1	45
1	100
2	65
2	75
2	76
3	21
3	45
3	99

반환 값은 186입니다. PowerCenter 통합 서비스는 ID를 기준으로 그룹화하고 AVG 식을 평가하고 3개의 값을 반환합니다. 그런 다음 SUM 함수로 이 값을 더하여 결과를 생성합니다.

또한 집계 함수를 식 변환에서 창 함수로 사용할 수도 있습니다. Spark 엔진에서 매핑을 실행할 때 집계 함수를 창 함수로 사용하려면 창 작업을 사용하도록 변환을 구성해야 합니다. 집계 함수를 창 함수로 사용하는 경우 식 변환이 활성 변환이 됩니다.

## 집계 함수 및 Null

PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. 집계 함수의 Null 값을 NULL 또는 0으로 처리하도록 PowerCenter 통합 서비스를 구성할 수 있습니다.

기본적으로 PowerCenter 통합 서비스 집계 함수에서 null 값을 NULL로 처리합니다. 전체 포트 또는 Null 값 그룹을 COUNT 함수에 전달하는 경우 이 함수는 0을 반환합니다. 전체 포트 또는 Null 값 그룹을 기타 모든 집계 함수에 전달하는 경우 이 함수는 NULL을 반환합니다. 필요한 경우 전체 포트의 Null 값을 집계 함수에 전달할 때 PowerCenter 통합 서비스가 0을 반환하도록 구성할 수 있습니다.

### 필터 조건

필터 조건을 사용하면 검색에서 반환되는 행을 제한할 수 있습니다.

필터는 검색에서 반환되는 행을 제한합니다. 모든 집계 함수와 CUME, MOVINGAVG 및 MOVINGSUM 함수에 필터 조건을 적용할 수 있습니다. 필터 조건은 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 필터 조건이 NULL 또는 FALSE로 평가되면 PowerCenter 통합 서비스가 행을 선택하지 않습니다.

유효한 모든 변환 식을 입력할 수 있습니다. 예를 들어 다음 식은 연봉이 \$50,000달러 이상인 모든 직원의 급여에서 중수를 계산합니다.

```
MEDIAN( SALARY, SALARY > 50000 )
```

다른 숫자 값을 필터 조건으로 사용할 수도 있습니다. 예를 들어 숫자 포트를 포함하여 MEDIAN 함수에 대한 전체 구문으로 다음을 입력할 수 있습니다.

```
MEDIAN( PRICE, QUANTITY > 0 )
```

모든 경우 PowerCenter 통합 서비스는 10진수 값을 정수로 반올림하여 필터 조건을 평가합니다(예: 1.5는 2로, 1.2는 1로, 0.35는 0으로). 반올림한 값이 0인 경우 필터 조건은 FALSE를 반환합니다. 값을 반올림하지 않으려면 TRUNC 함수를 사용하여 값을 정수로 잘라냅니다.

```
MEDIAN( PRICE, TRUNC( QUANTITY ) > 0 )
```

필터 조건을 생략할 경우 함수가 포트의 모든 행을 선택합니다.

## 이진 함수

이진 식에서 이진 함수를 사용할 수 있습니다. 이진 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

변환 언어에는 다음과 같은 이진 함수가 포함됩니다.

- EBCDIC\_ISO88591
- BINARY\_COMPARE
- BINARY\_CONCAT
- BINARY\_LENGTH
- BINARY\_SECTION
- DEC\_HEX
- ENC\_HEX
- SHA256

## 문자 함수

변환 언어에는 다음 문자 함수가 포함됩니다.

- ASCII
- CHR
- CHRCODE
- CONCAT
- INITCAP
- INSTR
- LENGTH
- LOWER
- LPAD
- LTRIM
- METAPHONE
- REPLACECHR
- REPLACESTR
- RPAD
- RTRIM

- SOUNDEX
- SUBSTR
- UPPER

MAX, MIN, LOWER, UPPER 및 INITCAP 문자 함수는 PowerCenter 통합 서비스의 코드 페이지를 사용하여 문제 데이터를 평가합니다.

## 변환 함수

변환 언어에는 다음 변환 함수가 포함됩니다.

- TO\_BIGINT
- TO\_CHAR(숫자)
- TO\_DATE
- TO\_DECIMAL
- TO\_FLOAT
- TO\_INTEGER

## 데이터 정리 함수

변환 언어에는 데이터 오류를 제거하는 함수 그룹이 포함됩니다. 데이터 정리 함수를 사용하여 다음 태스크를 완료할 수 있습니다.

- 입력 값을 테스트합니다.
- 입력 값의 데이터 유형을 변환합니다.
- 문자열 값을 잘라냅니다.
- 문자열의 문자를 바꿉니다.
- 문자열을 인코딩합니다.
- 정규식의 패턴을 일치시킵니다.

변환 언어에는 다음 데이터 정리 함수가 포함됩니다.

- GREATEST
- IN
- INSTR
- IS\_DATE
- IS\_NUMBER
- IS\_SPACES
- ISNULL
- LEAST
- LTRIM
- METAPHONE
- REG\_EXTRACT

- REG\_MATCH
- REG\_REPLACE
- REPLACECHR
- REPLACESTR
- RTRIM
- SQL\_LIKE
- SOUNDEX
- SUBSTR
- TO\_BIGINT
- TO\_CHAR
- TO\_DATE
- TO\_DECIMAL
- TO\_FLOAT
- TO\_INTEGER

## 날짜 함수

변환 언어에는 날짜를 반올림하거나 잘라내거나 비교하거나 날짜의 한 부분을 추출하거나 날짜에 산술 계산을 수행하는 날짜 함수 그룹이 포함됩니다.

날짜 함수에는 날짜 데이터 유형의 모든 값을 전달할 수 있습니다. 그러나 문자열을 날짜 함수에 전달하려면 먼저 **TO\_DATE** 함수를 사용하여 변환 날짜/시간 데이터 유형으로 변환해야 합니다.

변환 언어에는 다음 날짜 함수가 포함됩니다.

- ADD\_TO\_DATE
- DATE\_COMPARE
- DATE\_DIFF
- GET\_DATE\_PART
- IS\_DATE
- LAST\_DAY
- MAKE\_DATE\_TIME
- MAX
- MIN
- ROUND(날짜)
- SET\_DATE\_PART
- SYSTIMESTAMP
- TO\_CHAR(날짜)
- TIME\_RANGE
- TRUNC(날짜)

여러 날짜 함수에는 형식 인수가 포함됩니다. 이 인수에는 변환 언어 형식 문자열 중 하나를 지정해야 합니다. 날짜 형식 문자열은 국제화되지 않습니다.

날짜/시간 변환 데이터 유형은 나노초 단위까지의 전체 자릿수를 포함하는 날짜를 지원합니다.

관련 항목:

- [“날짜 형식 문자열” 페이지 34](#)

## 인코딩 함수

변환 언어에는 데이터 암호화, 압축, 인코딩 및 체크섬을 위한 다음 함수가 포함됩니다.

- AES\_DECRYPT
- AES\_ENCRYPT
- COMPRESS
- CRC32
- DEC\_BASE64
- DECOMPRESS
- ENC\_BASE64
- MD5

## 재무 함수

변환 언어에는 다음 재무 함수가 포함됩니다.

- FV
- NPER
- PMT
- PV
- RATE

## 숫자 함수

변환 언어에는 다음 숫자 함수가 포함됩니다.

- ABS
- CEIL
- CONV
- CUME
- EXP
- FLOOR
- LN
- LOG
- MAX



- MIN
- MOD
- MOVINGAVG
- MOVINGSUM
- POWER
- RAND
- ROUND
- SIGN
- SQRT
- TRUNC

## 지수 함수

변환 언어에는 다음 지수 함수가 포함됩니다.

- COS
- COSH
- SIN
- SINH
- TAN
- TANH

## 특수 함수

변환 언어에는 다음 특수 함수가 포함됩니다.

- ABORT
- DECODE
- 오류
- IIF
- LOOKUP

일반적으로 특수 함수는 식, 필터 및 업데이트 전략 변환에 사용됩니다. 특수 함수 안에 다른 함수를 중첩할 수 있습니다. 특수 함수를 집계 함수 안에 중첩할 수도 있습니다.

## 문자열 함수

변환 언어에는 다음 문자열 함수가 포함됩니다.

- CHOOSE
- INDEXOF
- MAX
- MIN

- REVERSE

## 테스트 함수

변환 언어에는 다음 테스트 함수가 포함됩니다.

- ISNULL
- IS\_DATE
- IS\_NUMBER
- IS\_SPACES

## 변수 함수

변환 언어에는 매핑 변수의 현재 값을 세션 전체에 업데이트하는 변수 함수 그룹이 포함됩니다. 워크플로우를 실행하면 **PowerCenter** 통합 서비스가 마지막 세션 실행의 최종 변수 값을 바탕으로 세션이 시작될 때의 변수의 시작 값과 현재 값을 평가합니다. 다음 변수 함수를 사용합니다.

- SetCountVariable
- SetMaxVariable
- SetMinVariable
- SetVariable

변수에 사용하는 변수 함수는 변수의 집계 유형에 따라 다릅니다.

다중 파티션을 포함하는 세션의 매핑 변수를 사용하는 경우 변수 함수를 사용하여 각 파티션에 대한 변수의 최종 값을 결정합니다. **PowerCenter** 통합 서비스는 세션이 종료될 때 모든 파티션에 집계 함수를 수행하여 리포지토리에 저장할 하나의 최종 값을 결정합니다. 재정의하지 않은 경우 다음에 이 세션을 사용하면 저장된 값이 변수의 시작 값으로 사용됩니다.

예를 들어 **SetMinVariable**을 사용하여 변수를 평가된 최소값으로 설정한 경우 **PowerCenter** 통합 서비스는 각 파티션에 대한 변수의 현재 최소값을 계산합니다. 그런 다음 세션이 종료될 때 모든 파티션에 대한 현재 최소값을 찾아 리포지토리에 저장합니다.

**SetVariable**은 파이프라인의 각 매핑 변수에 대해 한 번씩만 사용합니다. 파이프라인에 여러 파티션을 작성하는 경우 **PowerCenter** 통합 서비스가 여러 개의 스레드를 사용하여 이 파이프라인을 처리합니다. 동일한 변수에 대해 이 함수를 두 번 이상 사용하면 매핑 변수의 현재 값을 예측할 수 없게 될 수 있습니다.

## 창 함수

변환 언어에는 현재 행과 관련된 행 집합에 대한 계산을 수행하는 창 함수 그룹이 포함됩니다. 이 함수는 모든 입력 행에 대해 단일 반환 값을 계산합니다. 창 함수는 **Spark** 엔진에서 실행되는 매핑에 사용할 수 있습니다.

변환 언어에는 다음 창 함수가 포함됩니다.

- LAG
- LEAD

창 함수를 식 변환에서 사용할 수 있습니다. 식 변환에서 창 함수를 사용하면 변환이 활성 변환이 됩니다.

## ABORT

세션을 중지하고 지정된 오류 메시지를 세션 로그 파일에 기록합니다. PowerCenter 통합 서비스가 ABORT 함수를 만나면 해당 행에서 데이터 변환을 중지합니다. 세션 중단 전의 모든 행 읽기는 처리되고 세션에 정의된 소스 또는 대상 기반 커밋 간격 및 버퍼 블록 크기에 따라 로드됩니다. PowerCenter 통합 서비스는 중단된 행까지 대상에 쓴 다음 커밋되지 않은 모든 데이터를 마지막 커밋 지점으로 롤백합니다. 롤백 후에는 세션 복구를 수행할 수 있습니다.

ABORT는 데이터의 유효성을 검사할 때 사용합니다. 일반적으로 IIF 또는 DECODE 함수 안에 ABORT를 사용하여 세션 중단에 대한 규칙을 설정합니다.

ABORT 함수는 입력 및 출력 포트 기본값으로 사용할 수 있습니다. ABORT를 입력 포트에 사용하면 Null 값이 변환으로 전달되는 것을 방지할 수 있습니다. 또한 ABORT를 사용하여 식 안의 ERROR 함수 호출을 포함하는 모든 종류의 변환 오류를 처리할 수 있습니다. 기본값은 식의 ERROR 함수를 재정의합니다. 오류 발생 시 세션이 중지되도록 하려면 ABORT를 기본값으로 할당합니다.

연결되지 않은 포트에 대한 식에서 ABORT를 사용하는 경우 PowerCenter 통합 서비스가 ABORT 함수를 실행하지 않습니다.

**참고:** PowerCenter 통합 서비스는 ABORT 함수를 워크플로우 관리자에서 실행하는 Abort 명령과 다르게 처리합니다.

### 구문

ABORT( *string* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자열. 세션 중지 시 세션 로그 파일에 표시할 메시지입니다. 문자열 길이에는 제한이 없습니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

NULL.

## ABS

숫자 값의 절대값을 반환합니다.

### 구문

ABS( *numeric\_value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 절대값을 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

양수 값. ABS는 인수로 전달된 숫자 값과 동일한 데이터 유형을 반환합니다. 배정밀도를 전달하는 경우 배정밀도가 반환됩니다. 마찬가지로 정수를 전달하면 정수가 반환됩니다.

Null 값을 함수에 전달하는 경우 NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## 예

다음 식은 두 숫자 간의 차이를 큰 수에 관계없이 양수 값으로 반환합니다.

ABS( PRICE - COST )

PRICE	COST	RETURN VALUE
250	150	100
52	48	4
169.95	69.95	100
59.95	NULL	NULL
70	30	40
430	330	100
100	200	100

## ADD\_TO\_DATE

날짜/시간 값의 한 부분에 지정된 값을 추가한 다음, 함수로 전달하는 날짜와 같은 형식으로 반환합니다.

ADD\_TO\_DATE에는 양의 정수 및 음의 정수 값을 사용할 수 있습니다. ADD\_TO\_DATE를 사용하여 날짜의 다음 부분을 변경할 수 있습니다.

- **연도.** 수량 인수에 양 또는 음의 정수를 입력합니다. 다음 연도 형식 문자열 중 하나를 사용합니다. Y, YY, YYY 또는 YYYY. 다음 식은 SHIP\_DATE 포트의 모든 날짜에 10년을 더합니다.

ADD\_TO\_DATE ( SHIP\_DATE, 'YY', 10 )

- **월.** 수량 인수에 양 또는 음의 정수를 입력합니다. 다음 월 형식 문자열 중 하나를 사용합니다. MM, MON, MONTH. 다음 식은 SHIP\_DATE 포트의 각 날짜에서 10개월을 뺍니다.

ADD\_TO\_DATE( SHIP\_DATE, 'MONTH', -10 )

- **일.** 수량 인수에 양 또는 음의 정수를 입력합니다. 다음 일 형식 문자열 중 하나를 사용합니다. D, DD, DDD, DY 및 DAY. 다음 식은 SHIP\_DATE 포트의 각 날짜에 10일을 더합니다.

ADD\_TO\_DATE( SHIP\_DATE, 'DD', 10 )

- **시간.** 수량 인수에 양 또는 음의 정수를 입력합니다. 다음 시간 형식 문자열 중 하나를 사용합니다. HH, HH12, HH24. 다음 식은 SHIP\_DATE 포트의 각 날짜에 14시간을 더합니다.

ADD\_TO\_DATE( SHIP\_DATE, 'HH', 14 )

- **분.** 수량인수에 양 또는 음의 정수를 입력합니다. **MI** 형식 문자열을 사용하여 분을 설정합니다. 다음 식은 **SHIP\_DATE** 포트의 각 날짜에 25분을 더합니다.  
`ADD_TO_DATE( SHIP_DATE, 'MI', 25 )`
- **초.** 수량인수에 양 또는 음의 정수를 입력합니다. **SS** 형식 문자열을 사용하여 초를 설정합니다. 다음 식은 **SHIP\_DATE** 포트의 각 날짜에 59초를 더합니다.  
`ADD_TO_DATE( SHIP_DATE, 'SS', 59 )`
- **밀리초.** 수량인수에 양 또는 음의 정수를 입력합니다. **MS** 형식 문자열을 사용하여 밀리초를 설정합니다. 다음 식은 **SHIP\_DATE** 포트의 각 날짜에 125밀리초를 더합니다.  
`ADD_TO_DATE( SHIP_DATE, 'MS', 125 )`
- **마이크로초.** 수량인수에 양 또는 음의 정수를 입력합니다. **US** 형식 문자열을 사용하여 밀리초를 설정합니다. 다음 식은 **SHIP\_DATE** 포트의 각 날짜에 2,000밀리초를 더합니다.  
`ADD_TO_DATE( SHIP_DATE, 'US', 2000 )`
- **나노초.** 수량인수에 양 또는 음의 정수를 입력합니다. **NS** 형식 문자열을 사용하여 나노초를 설정합니다. 다음 식은 **SHIP\_DATE** 포트의 각 날짜에 3,000,000나노초를 더합니다.  
`ADD_TO_DATE( SHIP_DATE, 'NS', 3000000 )`

#### 구문

`ADD_TO_DATE( date, format, amount )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>날짜</i>	필수	날짜/시간 데이터 유형. 변경할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>형식</i>	필수	변경할 날짜 값 부분을 지정하는 형식 문자열입니다. 형식 문자열은 작은따옴표로 묶습니다(예: 'mm'). 형식 문자열은 대/소문자를 구분하지 않습니다.
<i>수량</i>	필수	변경할 날짜 값의 기준으로 사용할 연도, 월, 일, 시간 등의 기간을 지정하는 정수 값입니다. 정수로 평가되는 유효한 모든 변환 식을 입력할 수 있습니다.

#### 반환 값

이 함수에 전달하는 날짜와 동일한 형식의 날짜가 반환됩니다.

Null 값을 함수의 인수로 전달한 경우 NULL이 반환됩니다.

#### 예

다음 식은 **DATE\_SHIPPED** 포트의 각 날짜에 1개월을 더합니다. 특정 월에 존재하지 않는 날짜를 작성하는 값을 전달하면 **PowerCenter** 통합 서비스가 해당 월의 마지막 날을 반환합니다. 예를 들어 **Jan 31 1998**에 1개월을 더할 경우 **PowerCenter** 통합 서비스는 **Feb 28 1998**을 반환합니다.

또한 ADD\_TO\_DATE는 윤년을 인식하여 Jan 29 2000에 1개월을 더합니다.

```
ADD_TO_DATE( DATE_SHIPPED, 'MM', 1 )
ADD_TO_DATE( DATE_SHIPPED, 'MON', 1 )
ADD_TO_DATE( DATE_SHIPPED, 'MONTH', 1 )
```

DATE_SHIPPED	RETURN VALUE
Jan 12 1998 12:00:30AM	Feb 12 1998 12:00:30AM
Jan 31 1998 6:24:45PM	Feb 28 1998 6:24:45PM
Jan 29 2000 5:32:12AM	Feb 29 2000 5:32:12AM ( <i>Leap Year</i> )
Oct 9 1998 2:30:12PM	Nov 9 1998 2:30:12PM
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 10일을 뺍니다.

```
ADD_TO_DATE( DATE_SHIPPED, 'D', -10 )
ADD_TO_DATE( DATE_SHIPPED, 'DD', -10 )
ADD_TO_DATE( DATE_SHIPPED, 'DDD', -10 )
ADD_TO_DATE( DATE_SHIPPED, 'DY', -10 )
ADD_TO_DATE( DATE_SHIPPED, 'DAY', -10 )
```

DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:30AM	Dec 22 1996 12:00AM
Jan 31 1997 6:24:45PM	Jan 21 1997 6:24:45PM
Mar 9 1996 5:32:12AM	Feb 29 1996 5:32:12AM ( <i>Leap Year</i> )
Oct 9 1997 2:30:12PM	Sep 30 1997 2:30:12PM
Mar 3 1996 5:12:20AM	Feb 22 1996 5:12:20AM
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 15시간을 뺍니다.

```
ADD_TO_DATE( DATE_SHIPPED, 'HH', -15 )
ADD_TO_DATE( DATE_SHIPPED, 'HH12', -15 )
ADD_TO_DATE( DATE_SHIPPED, 'HH24', -15 )
```

DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:30AM	Dec 31 1996 9:00:30AM
Jan 31 1997 6:24:45PM	Jan 31 1997 3:24:45AM
Oct 9 1997 2:30:12PM	Oct 8 1997 11:30:12PM

DATE_SHIPPED	RETURN VALUE
Mar 3 1996 5:12:20AM	Mar 2 1996 2:12:20PM
Mar 1 1996 5:32:12AM	Feb 29 1996 2:32:12PM ( <i>Leap Year</i> )
NULL	NULL

## 날짜 작업

ADD\_TO\_DATE 작업 시에는 다음 팁을 따릅니다.

- 형식 문자열을 지정하고 수량 인수에 양 또는 음의 정수를 입력하여 날짜의 원하는 부분을 더하거나 뺄 수 있습니다.
- 특정 월에 존재하지 않는 날짜를 작성하는 값을 전달하면 PowerCenter 통합 서비스가 해당 월의 마지막 날을 반환합니다. 예를 들어 Jan 31 1998에 1개월을 더할 경우 PowerCenter 통합 서비스는 Feb 28 1998을 반환합니다.
- TRUNC 및 ROUND를 중첩하여 날짜를 조작할 수 있습니다.
- TO\_DATE를 중첩하여 문자열을 날짜로 변환할 수 있습니다.
- ADD\_TO\_DATE는 날짜에 지정된 한 부분만 변경합니다. 표준 시간에서 일광 절약 시간으로 변경되도록 날짜를 수정하려면 날짜의 시간 부분을 변경해야 합니다.

## AES\_DECRYPT

암호 해독된 데이터가 문자열 형식으로 반환됩니다. PowerCenter 통합 서비스는 128비트 인코딩의 AES(Advanced Encryption Standard) 알고리즘을 사용합니다. AES 알고리즘은 FIPS 승인을 받은 암호화 알고리즘입니다.

### 구문

AES\_DECRYPT ( *value*, *key* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	이진 데이터 유형. 암호 해독할 값입니다.
<i>키</i>	필수	문자열 데이터 유형. 전체 자릿수는 16자 이하입니다. 매핑 변수를 키에 사용할 수 있습니다. 암호화할 때 사용한 키와 동일한 키를 사용하여 값을 암호 해독합니다.

### 반환 값

암호 해독된 이진 값.

입력 값이 Null 값인 경우 NULL이 반환됩니다.

## 예

다음 예는 암호 해독된 미국의 주민등록번호를 반환합니다. 이 예에서 PowerCenter 통합 서비스는 SUBSRT 함수를 사용하여 주민등록번호의 처음 3개 숫자에서 키를 파생합니다.

```
AES_DECRYPT( SSN_ENCRYPT, SUBSTR( SSN,1,3 ))
```

SSN_ENCRYPT	DECRYPTED VALUE
07FB945926849D2B1641E708C85E4390	832-17-1672
9153ACAB89D65A4B81AD2ABF151B099D	832-92-4731
AF6B5E4E39F974B3F3FB0F22320CC60B	832-46-7552
992D6A5D91E7F59D03B940A4B1CBBCBE	832-53-6194
992D6A5D91E7F59D03B940A4B1CBBCBE	832-81-9528

## AES\_ENCRYPT

암호화된 형식의 데이터가 반환됩니다. PowerCenter 통합 서비스는 128비트 인코딩의 AES(Advanced Encryption Standard) 알고리즘을 사용합니다. AES 알고리즘은 FIPS 승인을 받은 암호화 알고리즘입니다.

이 함수를 사용하면 중요한 데이터가 노출되는 것을 방지할 수 있습니다. 예를 들어 주민등록번호/사회보장번호를 데이터 웨어하우스에 저장하려는 경우 AES\_ENCRYPT 함수로 주민등록번호를 암호화하면 기밀성을 유지할 수 있습니다.

## 구문

```
AES_ENCRYPT ( value, key )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	문자열 데이터 유형. 암호화할 값입니다.
키	필수	문자열 데이터 유형. 전체 자릿수는 16자 이하입니다. 매핑 변수를 키에 사용할 수 있습니다.

## 반환 값

암호화된 이진 값.

입력이 Null 값인 경우 NULL이 반환됩니다.



## 예제

다음 예는 미국 주민등록번호의 암호화된 값을 반환합니다. 이 예에서 PowerCenter 통합 서비스는 SUBSTR 함수를 사용하여 주민등록번호의 처음 3개 숫자에서 키를 파생합니다.

```
AES_ENCRYPT( SSN, SUBSTR( SSN,1,3 ))
```

SSN	ENCRYPTED VALUE
832-17-1672	07FB945926849D2B1641E708C85E4390
832-92-4731	9153ACAB89D65A4B81AD2ABF151B099D
832-46-7552	AF6B5E4E39F974B3F3FB0F22320CC60B
832-53-6194	992D6A5D91E7F59D03B940A4B1CBBCBE
832-81-9528	20812B3331676B15A9378000EB900EE3

## 팁

대상이 이진 데이터를 지원하지 않는 경우 AES\_ENCRYPT와 ENC\_BASE64 함수를 함께 사용하면 데이터베이스에 호환되는 형식으로 데이터를 저장할 수 있습니다.

## ASCII

PowerCenter 통합 서비스에서 ASCII 모드를 사용하는 경우 ASCII 함수는 함수에 전달된 문자열의 첫 번째 문자에 대한 숫자 ASCII 값을 반환합니다.

PowerCenter 통합 서비스에서 유니코드 모드를 사용하는 경우 ASCII 함수는 함수에 전달된 문자열의 첫 번째 문자에 대한 숫자 유니코드 값을 반환합니다. 유니코드 값의 범위는 0-65,535입니다.

모든 크기의 문자열을 ASCII에 전달할 수 있지만 이 함수는 문자열의 첫 번째 문자만 평가합니다. 문자열 값을 ASCII에 전달하기 전에 ASCII 또는 유니코드 값으로 변환할 특정 문자를 구문 분석할 수 있습니다. 예를 들어 RTRIM 또는 다른 문자열 조작 함수를 사용할 수 있습니다. 숫자 값을 전달하면 ASCII 함수가 문자열로 변환한 다음 해당 문자열의 첫 번째 문자에 대한 ASCII 또는 유니코드 값을 반환합니다.

이 함수는 CHRCODE 함수와 동일하게 동작합니다. ASCII 함수를 기존 식에 사용하는 경우 문제가 발생하지 않습니다. 그러나 새 식을 작성하는 경우에는 ASCII 함수 대신 CHRCODE 함수를 사용해야 합니다.

## 구문

ASCII ( *string* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
문자열	필수	문자열입니다. ASCII 값으로 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

정수. 문자열의 첫 번째 문자에 대한 ASCII 또는 유니코드 값이 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 ITEMS 포트에 있는 각 값의 첫 번째 문자에 대한 ASCII 또는 유니코드 값을 반환합니다.

ASCII( ITEMS )

ITEMS	RETURN VALUE
Flashlight	70
Compass	67
Safety Knife	83
Depth/Pressure Gauge	68
Regulator System	82

## AVG

행 그룹의 모든 값의 평균을 반환합니다. 필요한 경우 필터를 적용하여 평균을 계산할 때 읽을 행을 제한할 수 있습니다. AVG 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 숫자 데이터 유형을 반환해야 합니다.

## 구문

AVG( *numeric\_value* [, *filter\_condition*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 평균을 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우 NULL입니다. 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우를 예로 들 수 있습니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## Null

값이 NULL인 경우 AVG에서 행이 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

## 그룹 기준

AVG는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 AVG 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

다음 식은 Flashlight의 평균 도매 가격을 반환합니다.

```
AVG( WHOLESALE_COST, ITEM_NAME='Flashlight' )
```

ITEM_NAME	WHOLESALE_COST
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00
Flashlight	31.00

**RETURN VALUE:** 31.66

## 팁

AVG 함수가 평균을 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다. 예:

```
AVG( QTY * PRICE - DISCOUNT )
```

## BINARY\_COMPARE

두 이진 값을 비교한 후 동일한 경우 TRUE(1)를 반환하고 다른 경우 FALSE(0)를 반환합니다.

BINARY\_COMPARE 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

## 구문

```
BINARY_COMPARE( value1 [, value2] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>value1</i>	필수	Binary 데이터 유형
<i>value2</i>	필수	Binary 데이터 유형

### 반환 값

*value1*과 *value2*의 값이 동일한 경우 TRUE(1).

*value1*과 *value 2*의 값이 다른 경우 FALSE(0).

입력이 Null 값인 경우 NULL.

### 예

다음 예는 2개의 이진 값을 비교합니다.

```
BINARY_COMPARE( SYSID1, SYSID2 )
```

SYSID1 (Shown in Hex)	SYSID2 (Shown in Hex)	RETURN VALUE
0x000102030405060708	0x000102030405060708	1
0x000102030405060708	0x0405060708090A0B	0
0x000102030405060708	NULL	NULL
NULL	0x000102030405060708	NULL
NULL	NULL	NULL

## BINARY\_CONCAT

둘 이상의 이진 값을 하나로 연결한 후 연결된 값을 반환합니다.

BINARY\_CONCAT 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

### 구문

```
BINARY_CONCAT( value1, value2 [, value3] ... [, valueN] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>value1</i>	필수	Binary 데이터 유형
<i>value2</i>	필수	Binary 데이터 유형
<i>value3 - valueN</i>	선택 사항	Binary 데이터 유형

## 반환 값

이진.

입력 값이 null인 경우 NULL.

## Null

입력 값의 일부가 null인 경우 BINARY\_CONCAT 함수는 해당하는 입력 값을 무시하고 나머지 입력 값의 연결된 값을 반환합니다.

모든 입력 값이 null인 경우 BINARY\_CONCAT 함수는 null을 반환합니다.

## 예

다음 예는 2개의 이진 값을 연결합니다.

```
BINARY_CONCAT( SYSID1, SYSID2 )
```

SYSID1 (Shown in Hex)	SYSID2 (Shown in Hex)	RETURN VALUE (Shown in Hex)
0x000102030405060708	0x000102030405060708	0x000102030405060708000102030405060708
0x000102030405060708	0x0405060708090A0B	0x0001020304050607080405060708090A0B
0x000102030405060708	NULL	0x000102030405060708
NULL	0x000102030405060708	0x000102030405060708
NULL	NULL	NULL

## BINARY\_LENGTH

이진 값의 길이를 반환합니다.

BINARY\_LENGTH 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

## 구문

```
BINARY_LENGTH( value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	Binary 데이터 유형

### 반환 값

이진 값 길이를 나타내는 정수.

입력이 Null 값인 경우 NULL이 반환됩니다.

### 예제

다음 예는 다양한 이진 값의 길이를 반환합니다.

`BINARY_LENGTH( SYSID )`

SYSID1 (Shown in Hex)	RETURN VALUE
0x000102030405060708	9
0x00010203040506	7
0x00010203040506070809	10
NULL	NULL

## BINARY\_SECTION

이진 값의 일부를 반환합니다.

BINARY\_SECTION 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

### 구문

`BINARY_SECTION( binary, start [, length] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>binary</i>	필수	이진 값을 입력하거나 이진 데이터 유형을 반환하는 변환 식을 입력할 수 있습니다.
<i>start</i>	필수	정수를 입력하거나 정수를 반환하는 변환 식을 입력할 수 있습니다. 계산을 시작할 이진의 위치입니다. 시작 위치가 양수인 경우 BINARY_SECTION 함수는 입력 이진의 처음부터 수를 계산하여 시작 위치를 찾습니다. 시작 위치가 음수인 경우 BINARY_SECTION 함수는 입력 이진의 끝부터 수를 계산하여 시작 위치를 찾습니다. 시작 위치가 0인 경우 BINARY_SECTION 함수는 입력 이진의 첫 번째 바이트에서 검색을 시작합니다.
<i>length</i>	선택 사항	0보다 큰 정수를 입력하거나 양의 정수를 반환하는 변환 식을 입력할 수 있습니다. BINARY_SECTION 함수를 사용하여 반환하려는 바이트 수입니다. length 인수를 생략하는 경우 BINARY_SECTION 함수는 입력 이진의 시작 위치부터 끝까지의 모든 바이트를 반환합니다. 음의 정수 또는 0을 전달하는 경우 함수는 오류를 반환합니다.

## 반환 값

이진.

입력이 Null 값인 경우 NULL이 반환됩니다.

## 예제

다음 식은 이진 값의 처음 3개 바이트를 반환합니다.

```
BINARY_SECTION( SYSID, 0, 3 )
```

SYSID (Shown in Hex)	RETURN VALUE (Shown in Hex)
0x00010203	0x000102
0x0405060708	0x040506
NULL	NULL

다음 식은 이진 값의 4~8바이트를 반환합니다.

```
BINARY_SECTION( SYSID, 4, 5 )
```

SYSID (Shown in Hex)	RETURN VALUE (Shown in Hex)
0x000102030405060708	0x0304050607
0x0405060708090A0B	0x0708090A0B
NULL	NULL

음수 시작 값을 전달할 수도 있습니다. 이 경우에도 식은 *length* 인수의 결과를 반환할 때 왼쪽에서 오른쪽으로 소스 이진을 읽습니다.

```
BINARY_SECTION( SYSID, -6, 5 )
```

SYSID (Shown in Hex)	RETURN VALUE (Shown in Hex)
0x000102030405060708	0x0304050607
0x0405060708090A0B	0x060708090A
NULL	NULL

*length* 인수가 입력 값보다 긴 경우 BINARY\_SECTION 함수는 입력 값의 시작 위치부터 끝까지의 모든 바이트를 반환합니다. 다음 예제를 고려하십시오.

```
BINARY_SECTION( SYSID, 2, 8 )
```

여기서 SYSID는 0x000102030405에 해당하는 이진입니다.

반환 값은 0x0102030405에 해당하는 이진입니다. 이 결과를 다음 예와 비교하십시오.

```
BINARY_SECTION ( SYSID, -2, 8)
```

여기서 SYSID는 0x000102030405에 해당하는 이진입니다.

반환 값은 0x0405에 해당하는 이진입니다.

## CEIL

이 함수로 전달된 숫자 값 이상의 정수 중 가장 작은 정수를 반환합니다. 예를 들어 3.14를 CEIL에 전달하면 4가 반환됩니다. 3.98을 CEIL에 전달해도 4가 반환됩니다. 마찬가지로 -3.17을 CEIL에 전달하면 -3이 반환됩니다.

### 구문

```
CEIL( numeric_value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

숫자 값.

선언된 전체 자릿수가 28보다 큰 숫자 값을 전달하는 경우 배정밀도가 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.



## 예

다음 식은 다음 정수로 반올림된 가격을 반환합니다.

`CEIL( PRICE )`

PRICE	RETURN VALUE
39.79	40
125.12	126
74.24	75
NULL	NULL
-100.99	-100

**팁:** CEIL 함수가 다음 정수 값을 반환하기 전에 CEIL 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다. 예를 들어 숫자 값에 10을 곱한 후에 수정된 값보다 큰 가장 작은 정수를 계산하려는 경우 다음과 같이 함수를 쓸 수 있습니다.

`CEIL( PRICE * 10 )`

## CHOOSE

지정된 위치를 기준으로 문자열 목록에서 문자열을 선택합니다. 위치 및 값을 지정해야 합니다. 값이 위치와 일치하는 경우 PowerCenter 통합 서비스가 값을 반환합니다.

### 구문

`CHOOSE( index, string1 [, string2, ..., stringN] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
인덱스	필수	숫자 데이터 유형. 일치시킬 값의 위치에 대한 숫자를 입력합니다.
문자열	필수	모든 문자 값.

### 반환 값

인덱스 값의 위치와 일치하는 문자열이 반환됩니다.

인덱스 값의 위치와 일치하는 문자열이 없는 경우 NULL이 반환됩니다.

## 예

다음 식에서는 인덱스 값이 2이므로 문자열 'flashlight'가 반환됩니다.

`CHOOSE( 2, 'knife', 'flashlight', 'diving hood' )`

다음 식에서는 인덱스 값이 4이므로 NULL이 반환됩니다.

```
CHOOSE( 4, 'knife', 'flashlight', 'diving hood' )
```

CHOOSE 함수는 이 식에 4번째 인수가 없으므로 NULL을 반환합니다.

## CHR

PowerCenter 통합 서비스에서 ASCII 모드를 사용하는 경우 CHR 함수는 이 함수에 전달된 숫자 값에 해당하는 ASCII 문자를 반환합니다. ASCII 값의 범위는 0-255입니다. 모든 정수를 CHR에 전달할 수 있지만 인쇄할 수 있는 문자는 ASCII 코드 32부터 126까지입니다.

PowerCenter 통합 서비스에서 유니코드 모드를 사용하는 경우 CHR 함수는 이 함수에 전달된 숫자 값에 해당하는 유니코드 문자를 반환합니다. 유니코드 값의 범위는 0-65,535입니다.

### 구문

```
CHR( numeric_value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. ASCII 또는 유니코드 문자로 반환할 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

ASCII 또는 유니코드 문자. 한 문자를 포함하는 문자열이 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

### 예

다음 식은 ITEM\_ID 포트의 각 숫자 값에 대한 ASCII 또는 유니코드 문자를 반환합니다.

```
CHR( ITEM_ID )
```

ITEM_ID	RETURN VALUE
65	A
122	z
NULL	NULL
88	X
100	d
71	G

작은따옴표를 문자열에 연결하려면 **CHR** 함수를 사용합니다. 작은따옴표는 문자열 리터럴 안에 사용할 수 없는 유일한 문자입니다. 다음 예제를 고려하십시오.

```
'Joan' || CHR(39) || 's car'
```

반환 값은 다음과 같습니다.

```
Joan's car
```

## CHRCODE

**PowerCenter** 통합 서비스에서 **ASCII** 모드를 사용하는 경우 **CHRCODE** 함수는 함수에 전달된 문자열의 첫 번째 문자에 대한 숫자 **ASCII** 값을 반환합니다. **ASCII** 값의 범위는 0-255입니다.

**PowerCenter** 통합 서비스에서 유니코드 모드를 사용하는 경우 **CHRCODE** 함수는 함수에 전달된 문자열의 첫 번째 문자에 대한 숫자 유니코드 값을 반환합니다. 유니코드 값의 범위는 0-65,535입니다.

일반적으로 문자열 값을 **CHRCODE**에 전달하기 전에 **ASCII** 또는 유니코드 값으로 변환할 특정 문자를 구문 분석할 수 있습니다. 예를 들어 **RTRIM** 또는 다른 문자열 조작 함수를 사용할 수 있습니다. 숫자 값을 전달하면 **CHRCODE** 함수가 문자열로 변환한 다음 해당 문자열의 첫 번째 문자에 대한 **ASCII** 또는 유니코드 값을 반환합니다.

이 함수는 **ASCII** 함수와 동일하게 동작합니다. 현재 **ASCII** 함수를 식에 사용하는 경우 문제가 발생하지 않습니다. 그러나 새 식을 작성하는 경우에는 **ASCII** 함수 대신 **CHRCODE** 함수를 사용해야 합니다.

### 구문

**CHRCODE** ( *string* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자열입니다. <b>ASCII</b> 또는 유니코드 값으로 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

정수.

함수에 전달된 값이 **NULL**인 경우 **NULL**입니다.

### 예

다음 식은 **ITEMS** 포트에 있는 각 값의 첫 번째 문자에 대한 **ASCII** 또는 유니코드 값을 반환합니다.

**CHRCODE**( **ITEMS** )

<b>ITEMS</b>	<b>RETURN VALUE</b>
Flashlight	70
Compass	67
Safety Knife	83

ITEMS	RETURN VALUE
Depth/Pressure Gauge	68
Regulator System	82

## COMPRESS

zlib 1.2.1 압축 알고리즘을 사용하여 데이터를 압축합니다. COMPRESS 함수는 광역 네트워크를 통해 용량이 큰 데이터를 전송하기 전에 사용합니다.

### 구문

COMPRESS( *value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 데이터 유형. 압축할 데이터입니다.

### 반환 값

입력 값의 압축된 이진 값.

입력이 Null 값인 경우 NULL이 반환됩니다.

### 예

온라인 주문 서비스를 제공하는 조직이 있습니다. 이 조직의 직원은 광역 네트워크를 통해 고객의 주문 데이터를 전송하려고 합니다. 소스에 크기가 10MB인 행이 하나 있습니다. 직원은 이 행의 데이터를 COMPRESS를 사용하여 압축할 수 있습니다. 데이터를 압축하면 PowerCenter 통합 서비스가 네트워크에 써야 하는 데이터의 양이 줄어듭니다. 따라서 성능이 개선됩니다.

## CONCAT

두 문자열을 연관시킵니다. CONCAT 함수는 문자열을 연결하기 전에 모든 데이터를 텍스트로 변환합니다. || 문자열 연산자를 사용하여 문자열을 연결할 수도 있습니다. CONCAT 대신 || 문자열 연산자를 사용하면 PowerCenter 통합 서비스 성능이 개선됩니다.

### 구문

CONCAT( *first\_string*, *second\_string* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>first_string</i>	필수	이진을 제외한 모든 데이터 유형. 연결할 문자열의 첫 번째 부분입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>second_string</i>	필수	이진을 제외한 모든 데이터 유형. 연결할 문자열의 두 번째 부분입니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

문자열.

두 문자열 값이 모두 NULL인 경우 NULL이 반환됩니다.

## Null

문자열 중 하나가 NULL인 경우 CONCAT 함수는 이 문자열을 무시하고 다른 문자열을 반환합니다.

두 문자열이 모두 NULL인 경우 CONCAT 함수는 NULL을 반환합니다.

## 예

다음 식은 FIRST\_NAME과 LAST\_NAME 포트의 이름을 연결합니다.

```
CONCAT( FIRST_NAME, LAST_NAME )
```

FIRST_NAME	LAST_NAME	RETURN VALUE
John	Baer	JohnBaer
NULL	Campbell	Campbell
Bobbi	Apperley	BobbiApperley
Jason	Wood	JasonWood
Dan	Covington	DanCovington
Greg	NULL	Greg
NULL	NULL	NULL
100	200	100200

CONCAT 함수는 공백을 추가하여 문자열을 구분하지 않습니다. 두 문자열 사이에 공백을 추가하려면 중첩된 두 CONCAT 함수를 포함하는 식을 작성해야 합니다. 예를 들어 다음 식은 이름의 끝에 공백을 연결한 다음 성을 연결합니다.

```
CONCAT( CONCAT( FIRST_NAME, ' ' ), LAST_NAME )
```

FIRST_NAME	LAST_NAME	RETURN VALUE
John	Baer	John Baer
NULL	Campbell	Campbell <i>(includes leading blank)</i>
Bobbi	Apperley	Bobbi Apperley
Jason	Wood	Jason Wood
Dan	Covington	Dan Covington
Greg	NULL	Greg
NULL	NULL	NULL

작은따옴표를 문자열에 연결하려면 CHR 및 CONCAT 함수를 사용합니다. 작은따옴표는 문자열 리터럴 안에 사용할 수 없는 유일한 문자입니다. 다음 예제를 고려하십시오.

```
CONCAT( 'Joan', CONCAT( CHR(39), 's car' ) )
```

반환 값은 다음과 같습니다.

```
Joan's car
```

## CONVERT\_BASE

음이 아닌 숫자 문자열을 한 기준 값에서 다른 기준 값으로 변환합니다.

구문

```
CONVERT_BASE( value, source_base, dest_base )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 데이터 유형. 하나의 기준에서 다른 기준으로 변환할 값입니다. 최대값은 9,233,372,036,854,775,806입니다.
<i>source_base</i>	필수	숫자 데이터 유형. 변환할 데이터의 현재 기준값입니다. 최소 기준은 2입니다. 최대 기준은 36입니다.
<i>dest_base</i>	필수	숫자 데이터 유형. 데이터를 변환할 기준값입니다. 최소 기준은 2입니다. 최대 기준은 36입니다.

반환 값

숫자 값.

## 예

다음 예는 2222의 기준값을 10진수 기준값 10에서 이진 기준값 2로 변환합니다.

```
CONVERT_BASE( "2222", 10, 2 )
```

PowerCenter 통합 서비스는 100010101110을 반환합니다.

## COS

라디안으로 표시된 숫자 값의 코사인을 반환합니다.

## 구문

```
COS( numeric_value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 라디안(각도에 파이를 곱하고 180으로 나눈 값)으로 표시된 숫자 데이터. 코사인을 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

배정밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 Degrees 포트의 모든 값에 대한 코사인을 반환합니다.

```
COS( DEGREES * 3.14159265359 / 180 )
```

DEGREES	RETURN VALUE
0	1.0
90	0.0
70	0.342020143325593
30	0.866025403784421
5	0.996194698091745
18	0.951056516295147
89	0.0174524064371813
NULL	NULL

**팁:** COS 함수가 코사인을 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다. 예를 들어 코사인을 계산하기 전에 다음과 같이 포트의 값을 라디안으로 변환할 수 있습니다.

`COS( ARCS * 3.14159265359 / 180 )`

## COSH

라디안으로 표시된 숫자 값의 쌍곡선 코사인을 반환합니다.

### 구문

`COSH( numeric_value )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 라디안(각도에 파이를 곱하고 180으로 나눈 값)으로 표시된 숫자 데이터. 쌍곡선 코사인을 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

배정밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

### 예

다음 식은 Angles 포트의 값에 대한 쌍곡선 코사인을 반환합니다.

`COSH( ANGLES )`

ANGLES	RETURN VALUE
1.0	1.54308063481524
2.897	9.0874465864177
3.66	19.4435376920294
5.45	116.381231106176
0	1.0
0.345	1.06010513656773
NULL	NULL

**팁:** COSH 함수가 쌍곡선 코사인을 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다. 예:

`COSH( MEASURES.ARCS / 360 )`



## COUNT

그룹에서 Null이 아닌 값을 가진 행 수를 반환합니다. 필요에 따라 별표(\*) 인수를 포함하여 변환의 모든 입력 값 개수를 계산할 있습니다. COUNT 안에는 다른 집계 함수 하나만 중첩할 수 있습니다. 행 수를 계산하기 전에 조건을 적용하여 행을 필터링할 수 있습니다.

### 구문

```
COUNT( value [, filter_condition] )
```

또는

```
COUNT( * [, filter_condition] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	이진을 제외한 모든 데이터 유형. 수를 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
*	선택 사항	변환에서 모든 행의 수를 계산할 때 사용합니다.
filter_condition	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

정수.

이 함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우 0입니다(별표 인수를 포함한 경우 제외).

### Null

모든 값이 NULL인 경우 0이 반환됩니다.

별표 인수를 적용하면 행에 Null 값을 포함하는 열이 있더라도 모든 행의 수가 계산됩니다.

값 인수를 적용하면 Null 값을 포함하는 열이 무시됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

### 그룹 기준

COUNT는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다. 그룹 기준 포트가 없는 경우 COUNT 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

예

다음 식은 재고가 5개 미만인 항목의 수를 Null 값을 제외하고 계산합니다.

```
COUNT( ITEM_NAME, IN_STOCK < 5 )
```

ITEM_NAME	IN_STOCK
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5
Safety Knife	8
Halogen Flashlight	1

**RETURN VALUE: 1**

이 예에서 함수는 Halogen Flashlight의 수를 계산하고 NULL 항목의 수는 계산하지 않았습니다. 다음 예에서는 함수가 Null 값을 포함하여 변환의 모든 행 수를 계산합니다.

```
COUNT( *, QTY < 5 )
```

ITEM_NAME	QTY
Flashlight	10
NULL	2
Compass	NULL
Regulator System	5
Safety Knife	8
Halogen Flashlight	1

**RETURN VALUE: 2**

이 예에서 함수는 NULL 항목과 Halogen Flashlight의 수를 계산합니다. 별표 인수를 포함하고 필터를 사용하지 않으면 함수가 변환에 전달되는 모든 행의 수를 계산합니다. 예:

```
COUNT( * )
```

ITEM_NAME	QTY
Flashlight	10
NULL	2

ITEM_NAME	QTY
Compass	NULL
Regulator System	5
Safety Knife	8
Halogen Flashlight	1
RETURN VALUE: 6	

## CRC32

CRC32(32비트 Cyclic Redundancy Check) 값을 반환합니다. CRC32를 사용하면 데이터 변환 오류를 찾을 수 있습니다. 파일에 저장된 데이터가 수정되지 않았는지를 확인하려는 경우에도 CRC32를 사용할 수 있습니다.

CRC32를 사용하여 ASCII 모드 및 유니코드 모드의 데이터에 중복성 검사를 수행하면 PowerCenter 통합 서비스가 동일한 입력 값에 서로 다른 결과를 생성할 수 있습니다. CRC32를 사용하여 서로 다른 운영 체제의 데이터에 중복성 검사를 수행하면 PowerCenter 통합 서비스가 동일한 입력 값에 서로 다른 결과를 생성할 수 있습니다.

**참고:** CRC32는 서로 다른 입력 문자열에 대해 동일한 출력을 반환할 수 있습니다. 매핑의 키를 생성하려는 경우 시퀀스 생성기 변환을 사용해야 합니다. CRC32를 사용하여 매핑의 키를 생성하면 예기치 않은 결과가 나올 수 있습니다.

### 구문

CRC32( *value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 또는 이진 데이터 유형. 중복성 검사를 수행할 값을 전달합니다. 입력 값은 대/소문자를 구분합니다. 입력 값의 대/소문자는 반환 값에 영향을 미칩니다. 예를 들어 CRC32(informatica)와 CRC32(Informatica)는 다른 값을 반환합니다.

### 반환 값

32비트 정수 값.

### 예

광역 네트워크에 있는 소스의 데이터를 읽고 전송 중에 데이터가 수정되지 않았는지를 확인하려는 경우 파일의 데이터에 대한 체크섬을 계산하고 파일과 함께 저장합니다. 소스 데이터를 읽으면 PowerCenter 통합 서비스가 CRC32를 사용하여 체크섬을 계산하고 저장된 값과 비교합니다. 두 값이 동일하면 데이터가 수정되지 않은 것입니다.

## CUME

실행 합계가 반환됩니다. 실행 합계란 **CUME**가 값을 추가할 때마다 합계를 반환하는 것입니다. 실행 합계를 계산하기 전에 행 집합에서 특정 행을 필터링하는 조건을 추가할 수 있습니다.

**CUME** 및 유사 함수(예: **MOVINGAVG** 및 **MOVINGSUM**)를 사용하여 실행 값을 계산하면 보고가 단순해집니다.

### 구문

```
CUME( numeric_value [, filter_condition] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 실행 합계를 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 함수의 결과를 바탕으로 실행 합계를 계산하는 중첩된 식을 작성할 수 있습니다. 이때 함수의 결과는 숫자 값이어야 합니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

### Null

값이 NULL인 경우 **CUME** 함수는 이전 행에 대한 실행 합계를 반환합니다. 그러나 선택한 포트의 모든 값이 NULL인 경우 **CUME** 함수는 NULL을 반환합니다.

### 예

다음 샘플 행 집합은 **CUME** 함수를 사용하여 얻을 수 있는 결과입니다.

```
CUME( PERSONAL_SALES )
```

PERSONAL_SALES	RETURN VALUE
40000	40000
80000	120000
40000	160000
60000	220000

PERSONAL_SALES	RETURN VALUE
NULL	220000
50000	270000

마찬가지로 실행 합계를 계산하기 전에 값을 추가할 수 있습니다.

CUME( CA\_SALES + OR\_SALES )

CA_SALES	OR_SALES	RETURN VALUE
40000	10000	50000
80000	50000	180000
40000	2000	222000
60000	NULL	222000
NULL	NULL	222000
50000	3000	275000

## DATE\_COMPARE

두 날짜 중 더 이른 날짜를 나타내는 정수가 반환됩니다. DATE\_COMPARE는 날짜값이 아닌 정수 값을 반환합니다.

구문

DATE\_COMPARE( *date1*, *date2* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>date1</i>	필수	날짜/시간 데이터 유형. 비교할 첫 번째 날짜. 날짜로 평가되는 모든 유효한 변환식을 입력할 수 있습니다.
<i>date2</i>	필수	날짜/시간 데이터 유형. 비교할 두 번째 날짜. 날짜로 평가되는 모든 유효한 변환식을 입력할 수 있습니다.

반환 값

첫 번째 날짜가 더 이른 경우 -1이 반환됩니다.

두 날짜가 동일한 경우 0이 반환됩니다.

두 번째 날짜가 더 이른 경우 1이 반환됩니다.

날짜 값 중 하나가 NULL인 경우 NULL이 반환됩니다.

## 예

다음 식은 DATE\_PROMISED 및 DATE\_SHIPPED 포트의 각 날짜를 비교하여 더 이른 날짜를 나타내는 정수 값을 반환합니다.

```
DATE_COMPARE( DATE_PROMISED, DATE_SHIPPED )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997	Jan 13 1997	-1
Feb 1 1997	Feb 1 1997	0
Dec 22 1997	Dec 15 1997	1
Feb 29 1996	Apr 12 1996	-1 ( <i>Leap year</i> )
NULL	Jan 6 1997	NULL
Jan 13 1997	NULL	NULL

## DATE\_DIFF

두 날짜 간의 시간 길이를 반환합니다. 연도, 월, 일, 시간, 분, 초, 밀리초, 마이크로초 또는 나노초 형식을 반환하도록 요청할 수 있습니다. PowerCenter 통합 서비스는 첫 번째 날짜에서 두 번째 날짜를 빼고 그 차이를 반환합니다.

PowerCenter 통합 서비스는 일 수가 아닌 월 수를 기준으로 DATE\_DIFF 함수를 계산합니다. 각 월에서 선택된 일을 사용하여 특정 월에 대한 날짜 차이를 계산합니다. 특정 월에 대한 날짜 차이를 계산할 때 PowerCenter 통합 서비스는 해당 월에서 지난 일 수를 더한 다음 선택된 월의 전체 일 수로 이 값을 나눕니다.

PowerCenter 통합 서비스는 윤년 기간과 윤년이 아닌 기간의 샘플 기간에 대해 다른 값을 제공합니다. 이 차이는 DATE\_DIFF 함수에 2월이 포함될 경우 발생합니다. DATE\_DIFF 함수는 윤년 2월의 경우 일 수를 29로 나누고 평년 2월의 경우 28로 나눕니다.

예를 들어 9월 13일부터 2월 19일까지의 월 수를 계산하려는 경우 윤년 기간에서는 DATE\_DIFF 함수가 2월을 19/29개월 또는 0.655개월로 계산합니다. 평년 기간에서는 DATE\_DIFF 함수가 2월을 19/28개월 또는 0.678개월로 계산합니다. PowerCenter 통합 서비스는 남은 월의 날짜 차이도 이와 유사한 방식으로 계산하며 DATE\_DIFF 함수는 지정된 기간에 대한 합계 값을 반환합니다.

**참고:** 일부 데이터베이스는 다른 알고리즘을 사용하여 날짜 차이를 계산할 수 있습니다.

## 구문

```
DATE_DIFF( date1, date2, format )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>date1</i>	필수	날짜/시간 데이터 유형. 비교할 첫 번째 날짜에 대한 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>date2</i>	필수	날짜/시간 데이터 유형. 비교할 두 번째 날짜에 대한 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<b>형식</b>	필수	날짜 또는 시간 측정을 지정하는 형식 문자열. 연도, 월, 일, 시간, 분, 초, 밀리초, 마이크로초 또는 나노초를 지정할 수 있습니다. 'mm'과 같이 날짜의 한 부분만 지정할 수 있습니다. 형식 문자열은 작은따옴표로 묶습니다. 형식 문자열은 대/소문자를 구분하지 않습니다. 예를 들어 형식 문자열 'mm'은 'MM', 'Mm' 또는 'mM'과 동일합니다.

### 반환 값

배정밀도 값. *date1*이 *date2*보다 이후인 경우 반환 값은 양수입니다. *date1*이 *date2*보다 이전인 경우 반환 값은 음수입니다.

날짜가 동일한 경우 0이 반환됩니다.

날짜 값 중 하나(또는 둘 다)가 NULL인 경우 NULL이 반환됩니다.

### 예

다음 식은 DATE\_PROMISED와 DATE\_SHIPPED 포트 간의 시간 수를 반환합니다.

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'HH' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'HH12' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'HH24' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-2100
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	2100
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	6812.8916666667
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-8784

다음 식은 DATE\_PROMISED와 DATE\_SHIPPED 포트 간의 일 수를 반환합니다.

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'D' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DD' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DDD' )
```

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DY' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'DAY' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-87.5
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	87.5
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	283.870486111111
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-366

다음 식은 DATE\_PROMISED와 DATE\_SHIPPED 포트 간의 월 수를 반환합니다.

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MM' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MON' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MONTH' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-2.91935483870968
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	2.91935483870968
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	9.3290162037037
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-12

다음 식은 DATE\_PROMISED와 DATE\_SHIPPED 포트 간의 년 수를 반환합니다.

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'Y' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'YY' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'YYY' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'YYYY' )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-0.24327956989247
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	0.24327956989247
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL



DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	0.77741801697531
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-1

다음 식은 DATE\_PROMISED와 DATE\_SHIPPED 포트 간의 월 수를 반환합니다.

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MM' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MON' )
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED, 'MONTH' )
```

DATE_PROMISED	DATE_SHIPPED	LEAP YEAR VALUE (in Months)	NON-LEAP YEAR VALUE (in Months)
Sept 13	Feb 19	-5.237931034	-5.260714286
NULL	Feb 19	NULL	N/A
Sept 13	NULL	NULL	N/A

## DEC\_BASE64

base 64 인코딩 값을 디코딩한 후 데이터를 이진 데이터로 표현하는 문자열을 반환합니다. ENC\_BASE64를 사용하여 데이터를 인코딩하고 DEC\_BASE64를 사용하여 데이터를 디코딩하는 경우 동일한 데이터 이동 모드에서 디코딩 세션을 실행해야 합니다. 그렇지 않은 경우 디코딩된 데이터의 출력이 원래 데이터와 다를 수 있습니다.

### 구문

DEC\_BASE64( *value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 데이터 유형. 디코딩할 데이터입니다.

### 반환 값

디코딩된 이진 값.

입력이 Null 값인 경우 NULL이 반환됩니다.

반환 값은 세션을 실행하는 모드(유니코드 모드 및 ASCII 모드)에 따라 달라집니다.

### 예

WebSphere MQ 메시지 ID를 인코딩하고 워크플로우 중에 플랫폼 파일에 쓴 경우 WebSphere MQ 메시지 ID를 포함하여 플랫폼 파일 소스에서 데이터를 읽으려면 DEC\_BASE64를 사용하여 ID를 디코딩하고 원래 이진 값으로 변환해야 합니다.

## DEC\_HEX

16진수 인코딩 값을 디코딩한 후 데이터를 이진 데이터로 표현하는 이진 값을 반환합니다.

DEC\_HEX 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

### 구문

DEC\_HEX( *value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 데이터 유형. 이진으로 디코딩하려는 16진수 데이터입니다. 16진수 데이터에는 선행 0x 또는 0X가 포함될 수 있지만 필수는 아닙니다. 입력 데이터의 A~F 문자열은 대문자 또는 소문자일 수 있습니다.

### 반환 값

디코딩된 이진 값.

입력이 Null 값인 경우 NULL이 반환됩니다.

입력 문자열에 0~9 또는 A~F(대문자 또는 소문자) 외의 문자가 포함되는 경우 오류가 반환됩니다. 16진수 콘텐츠를 나타내는 선행 0x 또는 0X를 사용할 수 있으며 오류가 반환되지 않습니다.

### 예

WebSphere MQ 메시지 ID를 16진수 형식으로 인코딩하고 워크플로우 중에 플랫 파일에 쓴 경우가 있을 수 있습니다. WebSphere MQ 메시지 ID를 포함하여 플랫 파일 소스에서 데이터를 읽으려면 DEC\_HEX를 사용하여 ID를 디코딩하고 원래 이진 값으로 변환해야 합니다.

## DECODE

지정한 값에 대해 포트를 검색합니다. 함수에서 값이 검색되면 사용자가 정의한 결과 값이 반환됩니다.

DECODE 함수 내에서 검색을 수에 제한 없이 작성할 수 있습니다.

DECODE를 사용하여 문자열 포트의 값을 검색하는 경우 RTRIM 함수를 사용하여 후행 공백을 잘라내거나 검색 문자열에 공백을 포함할 수 있습니다.

### 구문

DECODE( *value*, *first\_search*, *first\_result* [, *second\_search*, *second\_result*]...[,*default*] )

다음 테이블에는 이 명령의 인수와 설명이 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	이진 데이터를 제외한 모든 데이터 유형. 검색할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>검색</i>	필수	값 인수와 동일한 데이터 유형의 값. 검색할 값을 전달합니다. 검색 값은 값 인수와 일치해야 합니다. 값의 일부를 검색할 수는 없습니다. 또한 검색 값은 대/소문자를 구분합니다. 예를 들어 특정 포트에서 문자열 'Halogen Flashlight'를 검색하려는 경우 'Halogen'이 아니라 'Halogen Flashlight'를 입력해야 합니다. 'Halogen'만 입력하면 검색에서 일치하는 값을 찾지 못합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>결과</i>	필수	이진 데이터를 제외한 모든 데이터 유형. 검색에서 일치하는 값을 찾을 경우 반환할 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>기본값</i>	선택 사항	이진 데이터를 제외한 모든 데이터 유형. 검색에서 일치하는 값을 찾지 못할 경우 반환할 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

검색에서 일치하는 값을 찾을 경우 *First\_result*가 반환됩니다.

검색에서 일치하는 값을 찾지 못한 경우 기본값이 반환됩니다.

기본값 인수를 생략하고 검색에서 일치하는 값을 찾지 못한 경우 NULL이 반환됩니다.

여러 조건이 충족된 경우에도 PowerCenter 통합 서비스는 처음 일치하는 결과를 반환합니다.

데이터에 다중 바이트 문자가 포함되고 DECODE 식이 문자열 데이터를 비교하는 경우 반환 값은 PowerCenter 통합 서비스의 코드 페이지 및 데이터 이동 모드에 따라 다릅니다.

## DECODE 및 데이터 유형

DECODE를 사용하는 경우 반환 값의 데이터 유형은 항상 전체 자릿수가 가장 큰 결과의 데이터 유형과 동일합니다.

예를 들어 다음 식이 있습니다.

```
DECODE ( CONST_NAME
         'Five', 5,
         'Pythagoras', 1.414213562,
         'Archimedes', 3.141592654,
         'Pi', 3.141592654 )
```

이 식의 반환 값은 5, 1.414213562 및 3.141592654입니다. 첫 번째 결과는 정수이고 다른 결과는 10진수입니다. 10진수 데이터 유형은 정수보다 전체 자릿수가 큼니다. 이 식은 항상 결과를 10진수로 씁니다.

많은 전체 자릿수 모드에서 세션을 실행할 때 최소 1개의 결과가 배정될도일 경우 반환 값의 데이터 유형은 배정될도입니다.

문자열과 숫자 반환 값을 모두 포함하는 DECODE 함수를 작성할 수 없습니다.

예를 들어 다음 식은 올바르지 않습니다.

```
DECODE ( CONST_NAME
         'Five', 5,
         'Pythagoras', '1.414213562',
         'Archimedes', '3.141592654',
         'Pi', 3.141592654 )
```

위의 식에 대한 유효성을 검사하면 다음과 같은 오류 메시지가 표시됩니다.

Function cannot resolve operands of ambiguously mismatching datatypes.

예

특정 ITEM\_ID를 검색하고 ITEM\_NAME을 반환하는 식에 DECODE를 사용하면

```
DECODE( ITEM_ID, 10, 'Flashlight',
        14, 'Regulator',
        20, 'Knife',
        40, 'Tank',
        'NONE' )
```

ITEM_ID	RETURN VALUE
10	Flashlight
14	Regulator
17	NONE
20	Knife
25	NONE
NULL	NONE
40	Tank

DECODE가 항목 17과 25에 기본값 NONE을 반환합니다. 검색 값이 ITEM\_ID와 일치하지 않기 때문입니다. 또한 DECODE는 NULL ITEM\_ID에 대해서도 NONE을 반환합니다.

다음 식은 TRUE 또는 FALSE에 대해 위에서 아래의 순서로 평가된 여러 열 및 조건을 테스트합니다.

```
DECODE( TRUE,
        Var1 = 22, 'Variable 1 was 22!',
        Var2 = 49, 'Variable 2 was 49!',
        Var1 < 23, 'Variable 1 was less than 23.',
        Var2 > 30, 'Variable 2 was more than 30.',
        'Variables were out of desired ranges.')
```

Var1	Var2	RETURN VALUE
21	47	Variable 1 was less than 23.
22	49	Variable 1 was 22!
23	49	Variable 2 was 49!

Var1	Var2	RETURN VALUE
24	27	Variables were out of desired ranges.
25	50	Variable 2 was more than 30.

## DECOMPRESS

zlib 1.2.1 압축 알고리즘을 사용하여 데이터 압축을 해제합니다. zlib 1.2.1 알고리즘을 사용하는 압축 도구 또는 COMPRESS 함수를 사용하여 압축된 데이터에 대해 DECOMPRESS 함수를 사용합니다. 데이터 압축을 해제하는 세션의 데이터 이동 모드가 데이터를 압축한 세션의 이동 모드와 다른 경우 압축 해제된 데이터의 출력이 원래 데이터와 다를 수 있습니다.

### 구문

DECOMPRESS( *value*, *precision* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	이진 데이터 유형. 압축 해제할 데이터입니다.
<i>정밀도</i>	선택 사항	정수 데이터 유형.

### 반환 값

입력 값의 압축 해제된 이진 값.

입력이 Null 값인 경우 NULL이 반환됩니다.

### 예

온라인 주문 서비스를 제공하는 조직이 있습니다. 이 조직의 직원은 광역 네트워크를 통해 고객의 주문 데이터를 전송받았습니다. 이 직원은 PowerCenter를 사용하여 데이터를 읽고 데이터 웨어하우스에 데이터를 로드하려고 합니다. 이 경우 DECOMPRESS를 행에 사용하여 데이터의 각 행을 압축 해제할 수 있습니다. PowerCenter 통합 서비스는 압축 해제된 데이터를 대상에 로드할 수 있습니다.

## EBCDIC\_ISO88591

EBCDIC로 인코딩된 이진 값을 ISO-8859-1로 인코딩된 문자열 값으로 변환합니다. 두 번째 인수가 제공되지 않은 경우 함수는 EBCDIC 037 코드 페이지를 사용하여 변환을 수행합니다. 올바른 두 번째 인수 값은 "037" 및 "1047"입니다.

EBCDIC\_ISO88591 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

### 구문

EBCDIC\_ISO88591( *value1* [, *value2*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>value1</i>	필수	EBCDIC 문자를 포함하는 이진 데이터입니다.
<i>value2</i>	선택 사항	"037" 또는 "1047" 중 하나를 포함하는 문자열입니다.

## 반환 값

EBCDIC 인코딩 이진 데이터에 해당하는 ISO-8859-1 문자열.

입력이 Null 값인 경우 NULL이 반환됩니다.

## 예제

다음 예는 ISO-8859-1로 인코딩된 문자열 값을 반환합니다.

```
EBCDIC_ISO88591( BIN_EBCDIC )
```

BIN_EBCDIC (Shown in Hex)	RETURN VALUE
0xC885939396	"Hello"
0xE696999384	"World"
NULL	NULL

## ENC\_BASE64

MIME(Multipurpose Internet Mail Extensions) 인코딩을 사용하여 이진 데이터를 문자열 데이터로 변환하는 방법으로 데이터를 인코딩합니다. 데이터를 인코딩하면 이진 데이터를 허용하지 않는 데이터베이스 또는 파일에 데이터를 저장할 수 있습니다. 또한 데이터를 인코딩하면 변환을 통해 이진 데이터를 문자열 형식으로 전달할 수 있습니다. 인코딩된 데이터는 원래 데이터보다 약 33% 더 깁니다. 무작위 문자 집합으로 표시됩니다.

## 구문

```
ENC_BASE64( value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	이진 또는 문자열 데이터 유형. 인코딩할 데이터입니다.

## 반환 값

인코딩된 값.

입력이 Null 값인 경우 NULL이 반환됩니다.

## 예

WebSphere MQ의 메시지를 읽고 해당 데이터를 플랫폼 파일 대상에 쓰려고 합니다. 이때 WebSphere MQ 메시지의 ID를 대상 데이터의 일부로 포함하려고 합니다. 그러나 MsgID 필드는 이진이고 플랫폼 파일 대상은 이진 데이터를 지원하지 않습니다. 이 경우 PowerCenter 통합 서비스가 데이터를 대상에 쓰기 전에 ENC\_BASE64를 사용하여 MsgID를 인코딩할 수 있습니다.

## ENC\_HEX

16진수 인코딩을 사용하여 이진 데이터를 문자열 데이터로 변환합니다. 데이터를 인코딩하면 이진 데이터를 허용하지 않는 데이터베이스 또는 파일에 데이터를 저장할 수 있습니다. 또한 데이터를 인코딩하면 변환을 통해 이진 데이터를 문자열 형식으로 전달할 수 있습니다. 인코딩된 데이터의 길이는 원래 이진 데이터의 두 배입니다. 0~9 및 A~F의 문자로 표시됩니다.

ENC\_HEX 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

## 구문

ENC\_HEX( *value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	Binary 데이터 유형입니다. 16진수로 인코딩할 데이터입니다.

## 반환 값

16진수 인코딩 문자열 값.

입력이 Null 값인 경우 NULL이 반환됩니다.

## 예제

WebSphere MQ의 메시지를 읽고 해당 데이터를 플랫폼 파일 대상에 쓰려고 합니다. 이때 WebSphere MQ 메시지의 ID를 대상 데이터의 일부로 포함하려고 합니다. 그러나 MsgID 필드는 이진이고 플랫폼 파일 대상은 이진 데이터를 지원하지 않습니다. 이 경우 PowerCenter 통합 서비스가 데이터를 대상에 쓰기 전에 ENC\_HEX를 사용하여 MsgID를 인코딩할 수 있습니다.

## 오류

PowerCenter 통합 서비스가 행을 건너뛰고 사용자가 정의한 오류 메시지를 표시합니다. 오류 메시지는 세션 로그에 표시됩니다. PowerCenter 통합 서비스는 이 건너뛴 행을 세션 거부 파일에 쓰지 않습니다.

ERROR를 식 변환에 사용하여 데이터의 유효성을 검사할 수 있습니다. 일반적으로 IIF 또는 DECODE 함수 안에 ERROR를 사용하여 행 건너뛰에 대한 규칙을 설정합니다.

ERROR 함수는 입력 및 출력 포트 기본값에 사용할 수 있습니다. ERROR를 입력 포트에 사용하면 Null 값이 변환으로 전달되는 것을 방지할 수 있습니다.

ERROR를 출력 포트에 사용하여 식 안의 ERROR 함수 호출을 포함하는 모든 종류의 변환 오류를 처리할 수 있습니다. ERROR 함수를 식 및 출력 포트 기본값에 사용하는 경우 PowerCenter 통합 서비스가 행을 건너

뛰고 식 및 기본값의 오류 메시지를 기록합니다. PowerCenter 통합 서비스가 오류를 생성하는 행을 건너뛰도록 하려면 ERROR를 기본값으로 할당합니다.

ERROR가 아닌 출력 기본값을 사용하는 경우 기본값이 식의 ERROR 함수를 재정의합니다. 예를 들어 ERROR 함수를 식에 사용하고 기본값 '1234'를 출력 포트에 할당합니다. PowerCenter 통합 서비스는 식에서 ERROR 함수가 발견될 때마다 오류를 값 '1234'로 재정의하고 '1234'를 다음 변환에 전달합니다. 행을 건너뛰지 않으며 오류를 세션 로그에 기록하지도 않습니다.

#### 구문

ERROR( *string* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자열 값. 통합 서비스가 ERROR 함수를 포함하는 식에 따라 행을 건너뛴 때 표시할 메시지입니다. 문자열 길이에는 제한이 없습니다.

#### 반환 값

문자열.

#### 예

다음 예에서는 조직의 모든 부서 직원의 평균 급여를 계산하는 매핑을 음수 값을 건너뛰면서 참조합니다. 다음 식은 ERROR 함수를 IIF 함수에 중첩하여 PowerCenter 통합 서비스가 Salary 포트에서 음수 급여를 포함하는 행을 건너뛰고 오류를 표시하도록 합니다.

IIF( SALARY < 0, ERROR ('Error. Negative salary found. Row skipped.'), EMP\_SALARY )

SALARY	RETURN VALUE
10000	10000
-15000	'Error. Negative salary found. Row skipped.'
NULL	NULL
150000	150000
1005	1005

## EXP

e를 지정된 제곱으로 올림한 값을 반환합니다. 여기서 e는 2.71828183입니다. 예를 들어 EXP(2)는 7.38905609893065를 반환합니다. 이 함수를 사용하면 비즈니스 데이터 외에 과학 및 기술 데이터를 분석할 수 있습니다. EXP는 숫자 값의 자연 대수를 반환하는 LN 함수의 역수입니다.

#### 구문

EXP( *exponent* )



다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
지수	필수	숫자 데이터 유형. e를 제공할 값입니다. 수식 $e^{\text{값}}$ 의 지수입니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

배정밀도 값.

함수에 인수로 전달된 값이 NULL인 경우 NULL이 반환됩니다.

## 예

다음 식은 Numbers 포트에 저장된 값을 지수 값으로 사용합니다.

EXP( NUMBERS )

NUMBERS	RETURN VALUE
10	22026.4657948067
-2	0.135335283236613
8.55	5166.754427176
NULL	NULL

## FIRST

포트 또는 그룹 내에서 찾은 첫 번째 값을 반환합니다. 필요한 경우 필터를 적용하여 PowerCenter 통합 서비스가 읽는 행을 제한할 수 있습니다. FIRST 안에는 다른 집계 함수 하나만 중첩할 수 있습니다.

## 구문

FIRST( value [, filter\_condition ] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	이진을 제외한 모든 데이터 유형. 첫 번째 값을 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
filter_condition	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

그룹의 첫 번째 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

## Null

값이 NULL인 경우 FIRST에서 행이 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

## 그룹 기준

FIRST는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다. 그룹 기준 포트가 없는 경우 FIRST 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

다음 식은 ITEM\_NAME 포트에서 가격이 \$10.00보다 큰 첫 번째 값을 반환합니다.

```
FIRST( ITEM_NAME, ITEM_PRICE > 10 )
```

ITEM_NAME	ITEM_PRICE
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00
Flashlight	31.00

**RETURN VALUE:** Flashlight

다음 식은 ITEM\_NAME 포트에서 가격이 \$40.00보다 큰 첫 번째 값을 반환합니다.

```
FIRST( ITEM_NAME, ITEM_PRICE > 40 )
```

ITEM_NAME	ITEM_PRICE
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00

ITEM_NAME	ITEM_PRICE
-----------	------------

Flashlight 31.00

RETURN VALUE: Regulator System

## FLOOR

이 함수로 전달하는 숫자 값 이하의 정수 중 가장 큰 정수를 반환합니다. 예를 들어 3.14를 FLOOR에 전달하면 3이 반환됩니다. 3.98을 FLOOR에 전달해도 3이 반환됩니다. 마찬가지로 -3.17을 FLOOR에 전달하면 -4가 반환됩니다.

### 구문

FLOOR( *numeric\_value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 숫자 데이터로 평가되는 모든 유효한 변환 식을 입력할 수 있습니다.

### 반환 값

선언된 전체 자릿수가 0과 28 사이인 숫자 값을 전달하는 경우 정수가 반환됩니다.

선언된 전체 자릿수가 28보다 큰 숫자 값을 전달하는 경우 배정밀도 값이 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

### 예

다음 식에서는 PRICE 포트의 값보다 작거나 같은 정수 중 가장 큰 정수가 반환됩니다.

FLOOR( PRICE )

PRICE	RETURN VALUE
39.79	39
125.12	125
74.24	74
NULL	NULL
-100.99	-101

**팁:** FLOOR에 전달하는 값에 대한 산술 계산을 수행할 수 있습니다. 예를 들어 숫자 값에 10을 곱한 후에 제곱보다 작은 정수 중 가장 큰 정수를 계산하려는 경우 다음과 같이 함수를 쓸 수 있습니다.

FLOOR( UNIT\_PRICE \* 10 )

## FV

정기적으로 일정 금액을 불입하고 있고 투자 금액에 대한 이율이 일정한 경우 투자의 미래 가치가 반환됩니다.

### 구문

`FV( rate, terms, payment [, present value, type] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>비율</i>	필수	분수. 각 기간별 이율입니다. 10진수로 표시됩니다. 백분율을 100으로 나눈 후 소수로 표시합니다.
<i>용어</i>	필수	분수. 기간 개수 또는 불입 횟수. 0보다 커야 합니다.
<i>불입</i>	필수	분수. 기간별 불입액입니다. 음수여야 합니다.
<i>현재 가치</i>	선택	분수. 투자의 현재 가치입니다. 이 인수를 생략한 경우 FV는 0을 사용합니다.
<i>유형</i>	선택	정수. 불입 시기. 기간 시작일에 불입할 경우 1을 입력합니다. 기간 종료일에 불입할 경우 0을 입력합니다. 기본값은 0입니다. 0 또는 1 이외의 값을 입력하면 PowerCenter 통합 서비스가 값을 1로 처리합니다.

### 반환 값

숫자.

### 예

매월 복리 9%(월리 9%/12, 즉 0.75%)로 \$2,000씩 예금하고 있습니다. 앞으로 12개월 동안 매월 초에 \$250를 예금할 계획입니다. 다음 식에서는 12개월 후의 잔액으로 \$5,337.96가 반환됩니다.

`FV(0.0075, 12, -250, -2000, TRUE)`

### 참고

기간별 이율을 계산하려면 연이율을 연간 불입 횟수로 나눕니다. 불입값과 현재값은 사용자가 불입하는 금액이므로 음수입니다.

## GET\_DATE\_PART

지정된 날짜 부분을 정수 값으로 반환합니다. 따라서 날짜의 월 부분을 반환하는 식을 작성하고 Apr 1 1997 00:00:00과 같은 날짜를 전달하는 경우 GET\_DATE\_PART는 4를 반환합니다.

### 구문

`GET_DATE_PART( date, format )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>날짜</i>	필수	날짜/시간 데이터 유형. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>형식</i>	필수	반환할 날짜 값 부분을 지정하는 형식 문자열. 형식 문자열을 작은따옴표로 묶습니다(예: 'mm'). 형식 문자열은 대/소문자를 구분하지 않습니다. 각 형식 문자열은 세션에 지정된 날짜 형식에 따라 날짜의 전체 부분을 반환합니다. 예를 들어 날짜 Apr 1 1997을 GET_DATE_PART에 전달한 경우 형식 문자열 'Y', 'YY', 'YYY' 또는 'YYYY'는 모두 1997을 반환합니다.

### 반환 값

날짜의 지정된 부분을 나타내는 정수.

함수에 전달된 값이 NULL인 경우 NULL입니다.

### 예

다음 식은 DATE\_SHIPPED 포트의 각 날짜에 대한 시간을 반환합니다. 12:00:00AM의 경우 기본 날짜 형식이 24시간 간격에 기반하므로 0이 반환됩니다.

```
GET_DATE_PART( DATE_SHIPPED, 'HH' )
GET_DATE_PART( DATE_SHIPPED, 'HH12' )
GET_DATE_PART( DATE_SHIPPED, 'HH24' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	0
Sep 2 1997 2:00:01AM	2
Aug 22 1997 12:00:00PM	12
June 3 1997 11:30:44PM	23
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에 대한 일을 반환합니다.

```
GET_DATE_PART( DATE_SHIPPED, 'D' )
GET_DATE_PART( DATE_SHIPPED, 'DD' )
GET_DATE_PART( DATE_SHIPPED, 'DDD' )
GET_DATE_PART( DATE_SHIPPED, 'DY' )
GET_DATE_PART( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	13
June 3 1997 11:30:44PM	3

DATE_SHIPPED	RETURN VALUE
Aug 22 1997 12:00:00PM	22
NULL	NULL

다음 식은 DATE\_SHIPPED 포트에 있는 각 날짜의 월을 반환합니다.

```
GET_DATE_PART( DATE_SHIPPED, 'MM' )
GET_DATE_PART( DATE_SHIPPED, 'MON' )
GET_DATE_PART( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	3
June 3 1997 11:30:44PM	6
NULL	NULL

다음 식은 DATE\_SHIPPED 포트에 있는 각 날짜의 연도를 반환합니다.

```
GET_DATE_PART( DATE_SHIPPED, 'Y' )
GET_DATE_PART( DATE_SHIPPED, 'YY' )
GET_DATE_PART( DATE_SHIPPED, 'YYY' )
GET_DATE_PART( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	1997
June 3 1997 11:30:44PM	1997
NULL	NULL

## GREATEST

입력 값 목록에서 가장 큰 값을 반환합니다. 이 함수를 사용하면 가장 큰 문자열, 날짜 또는 숫자가 반환됩니다. 기본적으로 일치하는 항목을 찾을 때는 대/소문자를 구분합니다.

구문

```
GREATEST( value1, [value2, ..., valueN,])
```

다음 테이블에는 이 명령의 인수에 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	이진수를 제외한 모든 데이터 유형입니다. 데이터 유형은 다른 값과 호환되어야 합니다. 다른 값과 비교할 값입니다. 값 인수를 하나 이상 입력해야 합니다. 값이 숫자이고 다른 입력 값이 숫자인 경우 가장 많은 전체 자릿수가 모든 값에 사용됩니다. 예를 들어 일부 값의 데이터 유형이 정수이고 다른 값의 데이터 유형은 고정밀도인 경우 PowerCenter 통합 서비스는 값을 고정밀도로 변환합니다.
<i>CaseFlag</i>	선택 사항	정수여야 합니다. 입력 값 인수가 문자열 값인 경우 값을 지정합니다. 이 함수의 인수가 대/소문자를 구분하는지를 결정합니다. 유효한 모든 변환 식을 입력할 수 있습니다. CaseFlag가 0 이외의 숫자인 경우 이 함수는 대/소문자를 구분합니다. CaseFlag가 0인 경우 이 함수는 대/소문자를 구분하지 않습니다. 기본값은 대/소문자 구분입니다.

## 반환 값

*value1*(입력 값 중 가장 큰 값인 경우), *value2*(입력 값 중 가장 큰 값인 경우).

인수 중 하나가 Null인 경우 NULL이 반환됩니다.

## 예제

다음 식은 주문된 항목 중 수량이 가장 많은 항목을 반환합니다.

GREATEST( QUANTITY1, QUANTITY2, QUANTITY3 )

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	756
			NULL
5000	97	17	5000
120	1724	965	1724

## IIF

사용자가 지정하는 두 값 중 하나를 조건의 결과에 따라 반환합니다.

## 구문

IIF( *condition*, *value1* [, *value2*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>condition</i>	필수	평가할 조건입니다. TRUE 또는 FALSE로 평가되는 유효한 모든 변환 식을 입력할 수 있습니다.
<i>value1</i>	필수	이진을 제외한 모든 데이터 유형. 조건이 TRUE일 경우 반환할 값입니다. 반환 값은 항상 사용자가 이 인수에 지정한 데이터 유형의 값입니다. 다른 IIF 식을 포함하여 유효한 모든 변환 식을 입력할 수 있습니다.
<i>value2</i>	선택 사항	이진을 제외한 모든 데이터 유형. 조건이 FALSE일 경우 반환할 값입니다. 다른 IIF 식을 포함하여 유효한 모든 변환 식을 입력할 수 있습니다.

일부 시스템의 조건부 함수와 달리 IIF 함수에서는 FALSE(*value2*) 조건이 필수가 아닙니다. *value2*를 생략한 경우 조건이 FALSE이면 다음이 반환됩니다.

- *value1*이 숫자 데이터 유형인 경우 0이 반환됩니다.
- *value1*이 문자열 데이터 유형인 경우 빈 문자열이 반환됩니다.
- *value1*이 날짜/시간 데이터 유형인 경우 NULL이 반환됩니다.

예를 들어 다음 식에는 FALSE 조건이 포함되어 있지 않고 *value1*이 문자열 데이터 유형이므로 PowerCenter 통합 서비스가 FALSE로 평가되는 각 행에 대해 빈 문자열을 반환합니다.

```
IIF( SALES > 100, EMP_NAME )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	' ' (empty string)
120	Sally Green	Sally Green
NULL	Greg Jones	' ' (empty string)

## 반환 값

조건이 TRUE이면 *value1*이 반환됩니다.

조건이 FALSE이면 *value2*가 반환됩니다.

예를 들어 다음 식에는 FALSE 조건 NULL이 포함되므로 PowerCenter 통합 서비스가 FALSE로 평가되는 각 행에 대해 NULL을 반환합니다.

```
IIF( SALES > 100, EMP_NAME, NULL )
```

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	NULL



SALES	EMP_NAME	RETURN VALUE
120	Sally Green	Sally Green
NULL	Greg Jones	NULL

데이터에 다중 바이트 문자가 포함되고 조건 인수가 문자열 데이터를 비교하는 경우 반환 값은 PowerCenter 통합 서비스의 코드 페이지 및 데이터 이동 모드에 따라 다릅니다.

### IIF 및 데이터 유형

IIF를 사용하는 경우 반환 값의 데이터 유형은 전체 자릿수가 가장 큰 결과의 데이터 유형과 동일합니다.

예를 들어 다음 식이 있습니다.

```
IIF( SALES < 100, 1, .3333 )
```

TRUE 결과(1)는 정수이고 FALSE 결과(.3333)는 10진수입니다. 10진수 데이터 유형은 전체 자릿수가 정수보다 많으므로 반환 값의 데이터 유형은 항상 10진수가 됩니다.

많은 전체 자릿수 모드에서 세션을 실행할 때 최소 1개의 결과가 배정될도인 경우 반환 값의 데이터 유형은 배정될도입니다.

### IIF의 특수한 사용

중첩된 IIF 문을 사용하여 여러 조건을 테스트합니다. 다음 예는 다양한 조건을 테스트하고 매출이 0 또는 음수일 경우 0을 반환합니다.

```
IIF( SALES > 0, IIF( SALES < 50, SALARY1, IIF( SALES < 100, SALARY2, IIF( SALES < 200, SALARY3,
BONUS))), 0 )
```

이 논리에 가독성을 높이는 설명을 추가할 수 있습니다.

```
IIF( SALES > 0,
--then test to see if sales is between 1 and 49:
  IIF( SALES < 50,

--then return SALARY1
    SALARY1,

--else test to see if sales is between 50 and 99:
    IIF( SALES < 100,

--then return
      SALARY2,

--else test to see if sales is between 100 and 199:
      IIF( SALES < 200,

--then return
        SALARY3,

--else for sales over 199, return
        BONUS)
    ),
--else for sales less than or equal to zero, return
  0)
```

IIF를 업데이트 전략에 사용합니다. 예:

```
IIF( ISNULL( ITEM_NAME ), DD_REJECT, DD_INSERT)
```

## IIF의 대체 함수

“[DECODE](#)” [페이지 82](#)는 많은 경우에 IIF 대신 사용됩니다. DECODE는 가독성을 개선합니다. 다음은 이전 섹션의 첫 번째 예를 사용하여 IIF 대신 DECODE를 사용하는 방법을 보여 줍니다.

```
DECODE( TRUE,
        SALES > 0 and SALES < 50, SALARY1,
        SALES > 49 AND SALES < 100, SALARY2,
        SALES > 99 AND SALES < 200, SALARY3,
        SALES > 199, BONUS)
```

IIF 대신 필터 변환을 사용하면 세션 성능을 극대화할 수 있습니다.

## IN

입력 데이터를 값 목록에 일치시킵니다. 기본적으로 일치하는 항목을 찾을 때는 대/소문자를 구분합니다.

### 구문

```
IN( valueToSearch, value1, [value2, ..., valueN,] CaseFlag )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>valueToSearch</i>	필수	문자열, 날짜 또는 숫자 값을 입력할 수 있습니다. 쉼표로 구분된 값 목록과 일치시킬 입력 값입니다.
<i>값</i>	필수	<i>valueToSearch</i> 인수에 대해 지정된 유형에 따라 문자열, 날짜 또는 숫자 값일 수 있습니다. 검색할 값의 쉼표로 구분된 목록입니다. 변환의 포트를 값으로 입력할 수 있습니다. 나열할 수 있는 값의 수에는 제한이 없습니다.
<i>CaseFlag</i>	선택 사항	정수 또는 NULL이어야 합니다. 이 함수의 인수가 대/소문자를 구분하는지를 결정합니다. 유효한 모든 변환식을 입력할 수 있습니다. CaseFlag가 0 이외의 숫자인 경우 이 함수는 대/소문자를 구분합니다. CaseFlag가 0인 경우 이 함수는 대/소문자를 구분하지 않습니다. CaseFlag가 null 값인 경우 이 함수는 함수의 인수와 일치하지 않으면 NULL을 반환합니다. 그렇지 않고 함수의 인수와 일치하면 CaseFlag는 1을 반환합니다. 기본값은 대/소문자 구분입니다.

### 반환 값

입력 값이 목록의 값과 일치하는 경우 TRUE(1)가 반환됩니다.

입력 값이 목록의 값과 일치하지 않는 경우 FALSE(0)가 반환됩니다.

입력이 Null 값인 경우 NULL이 반환됩니다.

## 예제

다음 식은 입력 값이 Safety Knife, Chisel Point Knife 또는 Medium Titanium Knife인지 여부를 결정합니다. 입력 값은 쉼표로 구분된 값 목록과 대/소문자가 일치하지 않아도 됩니다.

```
IN( ITEM_NAME, 'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife', 0 )
```

ITEM_NAME	RETURN VALUE
Stabilizing Vest	0 (FALSE)
Safety knife	1 (TRUE)
Medium Titanium knife	1 (TRUE)
	NULL

## INDEXOF

값 목록에서 값 인덱스를 찾습니다. 기본적으로 일치하는 항목을 찾을 때는 대/소문자를 구분합니다.

### 구문

```
INDEXOF( valueToSearch, string1 [, string2, ..., stringN,] [CaseFlag] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>valueToSearch</i>	필수	문자열 데이터 유형. 문자열 목록에서 검색할 값입니다.
<i>문자열</i>	필수	문자열 데이터 유형. 검색 기준이 되는 값의 쉼표로 구분된 목록입니다. 값은 문자열 형식으로 입력할 수 있습니다. 나열할 수 있는 값의 수에는 제한이 없습니다. 이 값은 CaseFlag를 0으로 설정하지 않는 한 대/소문자를 구분합니다.
<i>CaseFlag</i>	선택 사항	정수여야 합니다. valueToSearch 인수가 문자열 값인 경우 값을 지정합니다. 이 함수의 인수가 대/소문자를 구분하는지를 결정합니다. 유효한 모든 변환 식을 입력할 수 있습니다. CaseFlag가 0 이외의 숫자인 경우 이 함수는 대/소문자를 구분합니다. CaseFlag가 0인 경우 이 함수는 대/소문자를 구분하지 않습니다.

### 반환 값

입력 값이 *string1*과 일치하는 경우 1이 반환되고 입력 값이 *string2*와 일치하는 경우 2가 반환됩니다.

입력 값을 찾을 수 없는 경우 0이 반환됩니다.

입력이 Null 값인 경우 NULL이 반환됩니다.

## 예

다음 식은 ITEM\_NAME 포트의 값이 첫 번째, 두 번째 또는 세 번째 문자열과 일치하는지 여부를 결정합니다.

```
INDEXOF( ITEM_NAME, 'diving hood', 'flashlight', 'safety knife')
```

ITEM_NAME	RETURN VALUE
Safety Knife	0
diving hood	1
Compass	0
safety knife	3
flashlight	2

Safety Knife는 입력 값의 대/소문자와 일치하지 않으므로 0 값을 반환합니다.

## INITCAP

문자열에서 각 단어의 첫 글자는 대문자로 표시하고 다른 문자는 모두 소문자로 변환합니다. 단어는 영숫자가 아닌 문자와 공백(공백, 폼피드, 줄 바꿈, 캐리지 리턴, 탭 또는 세로 탭)으로 구분됩니다. 예를 들어 문자열 '...THOMAS'를 전달하면 Thomas가 반환됩니다.

## 구문

```
INITCAP( string )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>string</i>	필수	이진을 제외한 모든 데이터 유형. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

문자열. 데이터에 다중 바이트 문자가 포함되는 경우 반환 값은 PowerCenter 통합 서비스의 코드 페이지 및 데이터 이동 모드에 따라 다릅니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

예

다음 식은 FIRST\_NAME 포트의 모든 이름을 대문자로 변환합니다.

```
INITCAP( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
ramona	Ramona
18-albert	18-Albert
NULL	NULL
?!SAM	?!Sam
THOMAS	Thomas
PierRe	Pierre

## INSTR

문자열에서 왼쪽에서 오른쪽으로 계산된 문자 집합의 위치를 반환합니다.

구문

```
INSTR( string, search_value [,start [,occurrence [,comparison_type ]]] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택	설명
문자열	필수	문자열이어야 합니다. 평가할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 식의 결과는 문자열이어야 합니다. 그렇지 않은 경우 INSTR 함수가 평가 전에 값을 문자열로 변환합니다.
search_value	필수	모든 값. 검색 값은 대/소문자를 구분합니다. 검색할 문자 집합입니다. search_value는 문자열의 일부와 일치해야 합니다. 예를 들어 INSTR('Alfred Pope', 'Alfred Smith')를 쓰면 0이 반환됩니다. 유효한 모든 변환 식을 입력할 수 있습니다. 문자열을 검색하려는 경우 검색할 문자를 작은따옴표로 묶습니다(예: 'abc').
시작	선택 사항	정수 값이어야 합니다. 검색을 시작할 문자열의 위치입니다. 유효한 모든 변환 식을 입력할 수 있습니다. 기본값은 1이며 INSTR 함수가 문자열의 첫 글자에서 검색을 시작하는 것을 의미합니다. 시작 위치가 0인 경우 INSTR 함수는 문자열의 첫 문자부터 검색합니다. 시작 위치가 양수인 경우 INSTR 함수는 문자열의 처음부터 수를 계산하여 시작 위치를 찾습니다. 시작 위치가 음수인 경우 INSTR 함수는 문자열의 끝부터 수를 계산하여 시작 위치를 찾습니다. 이 인수를 생략할 경우 기본값 1이 사용됩니다.

인수	필수/ 선택	설명
<i>발생</i>	선택 사항	0보다 큰 양의 정수. 유효한 모든 변환 식을 입력할 수 있습니다. 검색 값이 문자열에서 두 번 이상 나타나는 경우 어떤 항목에서 검색할지 일치 항목을 지정할 수 있습니다. 예를 들어 시작 위치에서 두 번째 항목을 검색하려면 2를 입력합니다. 이 인수를 생략할 경우 INSTR 함수가 검색 값의 첫 번째 일치 항목을 검색하는 것을 의미하는 기본값 1이 사용됩니다. 10진수를 전달하면 PowerCenter 통합 서비스가 가까운 정수 값으로 반올림합니다. 음의 정수 또는 0을 전달하면 세션이 실패합니다.
<i>comparison_type</i>	선택 사항	PowerCenter 통합 서비스를 유니코드 모드에서 실행할 때의 문자열 비교 유형 (언어 또는 이진)입니다. PowerCenter 통합 서비스를 ASCII 모드에서 실행할 때의 비교 유형은 항상 이진입니다. 언어 비교는 언어별 정렬 규칙을 고려하는 반면 이진 비교는 비트 일치를 수행합니다. 예를 들어 독일어의 샤프 s 문자는 언어 비교에서 문자열 “ss”와 일치하지만 이진 비교에서는 아닙니다. 이진 비교는 언어 비교보다 빠르게 실행됩니다. 0 또는 1의 정수 값이어야 합니다. - 0: INSTR 함수가 언어 문자열 비교를 수행합니다. - 1: INSTR 함수가 이진 문자열 비교를 수행합니다. 기본값은 0입니다. 0을 입력하는 경우 세션 정렬 순서는 이진이 아니어야 합니다.

## 반환 값

검색이 성공한 경우 정수가 반환됩니다. 정수는 *search\_value*의 왼쪽에서 오른쪽으로 수를 계산하여 첫 번째 문자의 위치를 나타냅니다.

검색이 실패한 경우 0이 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 각 회사 이름의 처음부터 시작하여 문자 'a'가 첫 번째로 발생한 위치를 반환합니다.

*search\_value* 인수는 대/소문자를 구분하므로 'Blue Fin Aqua Center'의 'A'를 건너뛰고 'Aqua'의 'a'에 대한 위치를 반환합니다.

```
INSTR( COMPANY, 'a' )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	13
Maco Shark Shop	2
Scuba Gear	5
Frank's Dive Shop	3
VIP Diving Club	0

다음 식은 각 회사 이름의 처음부터 시작하여 문자 'a'가 두 번째로 발생한 위치를 반환합니다.  
*search\_value* 인수는 대/소문자를 구분하므로 'Blue Fin Aqua Center'의 'A'를 건너뛰고 0을 반환합니다.

```
INSTR( COMPANY, 'a', 1, 2 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	8
Scuba Gear	9
Frank's Dive Shop	0
VIP Diving Club	0

다음 식은 각 회사 이름의 마지막 문자부터 시작하여 문자 'a'가 두 번째로 발생한 위치를 반환합니다.  
*search\_value* 인수는 대/소문자를 구분하므로 'Blue Fin Aqua Center'의 'A'를 건너뛰고 0을 반환합니다.

```
INSTR( COMPANY, 'a', -1, 2 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	2
Scuba Gear	5
Frank's Dive Shop	0
VIP Diving Club	0

다음 식은 문자열 'Blue Fin Aqua Center'의 첫 번째 문자 위치를 반환합니다(회사 이름의 마지막 문자부터 시작).

```
INSTR( COMPANY, 'Blue Fin Aqua Center', -1, 1 )
```

COMPANY	RETURN VALUE
Blue Fin Aqua Center	1
Maco Shark Shop	0
Scuba Gear	0
Frank's Dive Shop	0
VIP Diving Club	0

### 중첩된 INSTR 사용

INSTR 함수를 다른 함수 안에 중첩하여 더 복잡한 테스트를 완료할 수 있습니다.

다음 식은 문자열의 끝부터 시작하여 문자열을 평가합니다. 이 식은 문자열에서 마지막(맨 오른쪽) 공백을 찾은 다음 이 공백의 왼쪽에 있는 모든 문자를 반환합니다.

```
SUBSTR( CUST_NAME,1,INSTR( CUST_NAME,' ', -1,1 ))
```

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

다음 식은 문자열에서 문자 '#'를 제거합니다.

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

CUST_ID	RETURN VALUE
ID#33	ID33
#A3577	A3577
SS #712403399	SS 712403399

## ISNULL

값이 NULL인지 여부가 반환됩니다. ISNULL은 빈 문자열을 FALSE로 평가합니다.

**참고:** 빈 문자열이 있는지 테스트하려면 LENGTH를 사용합니다.

### 구문

```
ISNULL( value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>value</i>	필수	이진을 제외한 모든 데이터 유형. 평가할 행을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

값이 NULL인 경우 TRUE(1)가 반환됩니다.

값이 NULL이 아닌 경우 FALSE(0)가 반환됩니다.



## 예

다음 예는 항목 테이블의 Null 값을 검사합니다.

```
ISNULL( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (FALSE)
NULL	1 (TRUE)
Regulator system	0 (FALSE)
' '	0 (FALSE) <i>Empty string is not NULL</i>

## IS\_DATE

문자열 값이 유효한 날짜인지 여부를 반환합니다. 유효한 날짜는 세션에서 지정한 날짜/시간 형식의 날짜 부분에 있는 모든 문자열입니다. 테스트할 문자열이 이 데이터 형식이 아닌 경우 **TO\_DATE** 형식 문자열을 사용하여 날짜 형식을 지정합니다. **IS\_DATE**에 전달된 문자열이 지정된 형식 문자열과 일치하지 않는 경우 **FALSE(0)**가 반환됩니다. 문자열이 형식 문자열과 일치하는 경우 **TRUE(1)**가 반환됩니다.

**IS\_DATE**는 문자열을 평가하고 정수 값을 반환합니다.

**IS\_DATE** 식의 출력 포트는 문자열 또는 숫자 데이터 유형이어야 합니다.

**IS\_DATE**를 사용하여 플랫폼 파일의 데이터를 대상에 쓰기 전에 테스트하거나 필터링할 수 있습니다.

**IS\_DATE**에는 YY 형식 문자열 대신 RR 형식 문자열을 사용합니다. 두 형식 문자열은 대부분의 경우 동일한 값을 반환하지만 YY가 잘못된 결과를 반환하는 특수한 경우가 일부 있습니다. 예를 들어 식

**IS\_DATE('02/29/00', 'YY')**는 내부에서 **IS\_DATE(02/29/1900 00:00:00)**로 계산되어 **FALSE**를 반환합니다. 그러나 PowerCenter 통합 서비스는 식 **IS\_DATE('02/29/00', 'RR')**를 **IS\_DATE(02/29/2000 00:00:00)**로 계산하여 **TRUE**를 반환합니다. 첫 번째 경우에서 1900년은 윤년이 아니므로 2월 29일이 존재하지 않습니다.

**참고:** **IS\_DATE**는 **TO\_DATE**와 동일한 형식 문자열을 사용합니다.

## 구문

```
IS_DATE( value [,format] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	문자열 데이터 유형이어야 합니다. 평가할 행을 전달합니다. 유효한 모든 변환식을 입력할 수 있습니다.
형식	선택 사항	유효한 <b>TO_DATE</b> 형식 문자열을 입력합니다. 형식 문자열은 문자열 인수의 일부와 일치해야 합니다. 예를 들어 문자열 'Mar 15 1997 12:43:10AM'을 전달하는 경우 형식 문자열 'MON DD YYYY HH12:MI:SSAM'을 사용해야 합니다. 형식 문자열을 생략하는 경우 문자열 값은 세션에서 지정한 날짜 형식이어야 합니다.

## 반환 값

행이 유효한 날짜인 경우 **TRUE(1)**가 반환됩니다.

행이 유효한 날짜가 아닌 경우 **FALSE(0)**가 반환됩니다.

식의 값이 **NULL**이거나 형식 문자열이 **NULL**인 경우 **NULL**이 반환됩니다.

**경고:** **IS\_DATE** 문자열의 형식은 날짜 구분 기호를 포함하여 형식 문자열과 일치해야 합니다. 일치하지 않을 경우 **PowerCenter** 통합 서비스가 정확하지 않은 값을 반환하거나 레코드를 건너뛸 수 있습니다.

## 예

다음 식은 **INVOICE\_DATE** 포트를 검사하여 유효한 날짜인지 확인합니다.

```
IS_DATE( INVOICE_DATE )
```

이 식은 다음과 유사한 데이터를 반환합니다.

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	0 (FALSE)
'04/01/98'	0 (FALSE)
'04/01/1998 00:12:15.7008'	1 (TRUE)
'02/31/1998 12:13:55.9204'	0 (FALSE) ( <i>February does not have 31 days</i> )
'John Smith'	0 (FALSE)

다음 **IS\_DATE** 식은 형식 문자열 'YYYY/MM/DD'를 지정합니다.

```
IS_DATE( INVOICE_DATE, 'YYYY/MM/DD' )
```

문자열 값이 이 형식과 일치하지 않는 경우 **IS\_DATE**가 **FALSE**를 반환합니다.

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	0 (FALSE)
'04/01/98'	0 (FALSE)
'1998/01/12'	1 (TRUE)
'1998/11/21 00:00:13'	0 (FALSE)
'1998/02/31'	0 (FALSE) ( <i>February does not have 31 days</i> )
'John Smith'	0 (FALSE)

다음 예는 TO\_DATE를 사용하여 문자열을 날짜로 변환하기 전에 IS\_DATE를 사용하여 데이터를 테스트하는 방법을 보여 줍니다. 이 식은 INVOICE\_DATE 포트의 값을 검사하고 각 유효한 날짜를 날짜 값으로 변환합니다. 값이 유효한 날짜가 아닌 경우 PowerCenter 통합 서비스가 ERROR를 반환하고 행을 건너뜁니다.

이 예는 날짜/시간 값을 반환합니다. 그러므로 식의 출력 포트는 날짜/시간이어야 합니다.

```
IIF( IS_DATE ( INVOICE_DATE, 'YYYY/MM/DD' ), TO_DATE( INVOICE_DATE ), ERROR('Not a valid date' ) )
```

INVOICE_DATE	RETURN VALUE
NULL	NULL
'180'	'Not a valid date'
'04/01/98'	'Not a valid date'
'1998/01/12'	1998/01/12
'1998/11/21 00:00:13'	'Not a valid date'
'1998/02/31'	'Not a valid date'
'John Smith'	'Not a valid date'

## IS\_NUMBER

문자열이 유효한 수인지 여부를 반환합니다. 유효한 숫자는 다음으로 구성됩니다.

- 숫자 앞의 선택적 공백
- 선택적 기호(+/-)
- 한 자릿수 이상(선택적 소수점 포함)
- 선택적 지수 기호(예: 문자 'e' 또는 'E' 및 Windows의 문자 'd' 또는 'D') 다음에 선택적 기호(+/-)가 오고 그 다음에 한 자릿수 이상의 숫자가 옵니다
- 숫자 뒤의 선택적 공백

다음 숫자는 모두 유효합니다.

```
' 100 '
' +100 '
' -100 '
' -3.45e+32 '
' +3.45E-32 '
' +3.45d+32 ' (Windows only)
' +3.45D-32 ' (Windows only)
' .6804 '
```

IS\_NUMBER 식의 출력 포트는 문자열 또는 숫자 데이터 유형이어야 합니다.

IS\_NUMBER를 사용하여 플랫폼 파일의 데이터를 대상에 쓰기 전에 테스트하거나 필터링할 수 있습니다.

### 구문

```
IS_NUMBER( value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 데이터 유형이어야 합니다. 평가할 행을 전달합니다. 유효한 모든 변환식을 입력할 수 있습니다.

#### 반환 값

행이 유효한 숫자인 경우 TRUE(1)가 반환됩니다.

행이 유효한 숫자가 아닌 경우 FALSE(0)가 반환됩니다.

식의 값이 Null인 경우 NULL이 반환됩니다.

#### 예

다음 식은 ITEM\_PRICE 포트를 검사하여 유효한 숫자인지 확인합니다.

```
IS_NUMBER( ITEM_PRICE )
```

ITEM_PRICE	RETURN VALUE
'123.00'	1 (True)
'-3.45e+3'	1 (True)
'-3.45D-3'	1 (True - Windows only)
'-3.45d-3'	0 (False - UNIX only)
'3.45E- '	0 (False) <i>Incomplete number</i>
' '	0 (False) <i>Consists entirely of blanks</i>
''	0 (False) <i>Empty string</i>
' +123abc'	0 (False)
' 123'	1 (True) <i>Leading white blanks</i>
'123 '	1 (True) <i>Trailing white blanks</i>
'ABC'	0 (False)
'-ABC'	0 (False)
NULL	NULL

TO\_FLOAT와 같은 숫자 변환 함수를 사용하기 전에 IS\_NUMBER를 사용하여 데이터를 테스트할 수 있습니다. 예를 들어 다음 식은 ITEM\_PRICE 포트의 값을 검사하고 각 유효한 숫자를 배정밀도 부동 소수점 값으로 변환합니다. 값이 유효한 숫자가 아닌 경우 PowerCenter 통합 서비스가 0.00을 반환합니다.

```
IIF( IS_NUMBER ( ITEM_PRICE ), TO_FLOAT( ITEM_PRICE ), 0.00 )
```

ITEM_PRICE	RETURN VALUE
'123.00'	123
'-3.45e+3'	-3450
'3.45E-3'	0.00345
' '	0.00 <i>Consists entirely of blanks</i>
''	0.00 <i>Empty string</i>
'+123abc'	0.00
'' 123ABC'	0.00
'ABC'	0.00
'-ABC'	0.00
NULL	NULL

## IS\_SPACES

문자열 값이 모두 공백으로 이루어져 있는지 여부를 반환합니다. 이 공백에는 공백, FF(FormFeed), 줄 바꿈, 캐리지 리턴, 탭 또는 세로 탭이 포함됩니다.

빈 문자열의 경우 공백이 없으므로 FALSE로 평가됩니다. 빈 문자열이 있는지 테스트하려면 LENGTH를 사용합니다.

### 구문

```
IS_SPACES( value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	문자열 데이터 유형이어야 합니다. 평가할 행을 전달합니다. 유효한 모든 변환식을 입력할 수 있습니다.

### 반환 값

행이 공백으로만 구성되는 경우 TRUE(1)가 반환됩니다.

행에 데이터가 포함되는 경우 FALSE(0)가 반환됩니다.

식의 값이 Null인 경우 NULL이 반환됩니다.

예

다음 식은 ITEM\_NAME 포트를 검사하여 공백으로만 구성된 행이 있는지 확인합니다.

```
IS_SPACES( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	0 (False)
	1 (True)
Regulator system	0 (False)
NULL	NULL
' '	0 (FALSE) ( <i>Empty string does not contain spaces.</i> )

**팁:** IS\_SPACES를 사용하면 대상 테이블의 문자 열에 공백을 쓰는 것을 방지할 수 있습니다. 예를 들어 대상 테이블의 고정 길이 CHAR(5) 열에 고객 이름을 쓰는 변환에서 공백 대신 '00000'을 써야 하는 경우 다음과 유사한 식을 작성할 수 있습니다.

```
IIF( IS_SPACES( CUST_NAMES ), '00000', CUST_NAMES )
```

## LAG

식 변환에서 현재 행 앞의 행에 대한 오프셋 수인 값을 반환합니다. Hadoop 환경의 Spark 엔진에서 매핑을 실행하는 경우 이 함수를 사용하여 현재 행의 값과 이전 행의 값을 비교할 수 있습니다.

데이터 집합에서 lag 값은 현재 행의 앞에 표시됩니다.

변환에서 LAG 함수를 사용하는 경우 창 작업을 사용하도록 변환을 구성해야 합니다. 창 작업 속성은 데이터의 분할 및 정렬 방법을 정의합니다.

### 구문

```
LAG ( column_name, offset, default )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
column_name	필수	함수가 작동하는 대상 열 또는 식입니다.
offset	필수	정수 데이터 유형. 값을 가져올 현재 행 앞의 행 수입니다.
default	선택 사항	오프셋이 파티션 또는 테이블의 범위 밖에 있는 경우 반환할 기본값입니다. 기본값은 NULL입니다.

### 반환 값

지정된 column\_name의 데이터 유형.

반환 값이 지정된 파티션의 범위 밖에 있는 경우 *Default*.  
*default*가 생략되었거나 NULL로 설정된 경우 NULL.

예

다음 식은 이전 정렬이 배치된 날짜를 반환합니다.

```
LAG ( ORDER_DATE, 1, NULL )
```

ORDER_DATE	ORDER_ID	RETURN VALUE
2017/09/25	1	NULL
2017/09/26	2	2017/09/25
2017/09/27	3	2017/09/26
2017/09/28	4	2017/09/27
2017/09/29	5	2017/09/28
2017/09/30	6	2017/09/29

첫 번째 행의 lag 값이 파티션 밖에 있으므로 함수가 기본값인 NULL을 반환합니다.

다음 예에서는 자동차에서 주행 및 이벤트 ID와 타임스탬프가 포함되는 GPS Ping을 수신합니다. 이제 각 Ping 간의 시간 차이를 계산하려고 합니다.

다음 식은 두 번의 개별 주행에 대해 현재 행과 이전 행의 시간 차이를 계산합니다.

```
DATE_DIFF( EVENT_TIME, LAG ( EVENT_TIME, 1, NULL ), 'ss' )
```

데이터를 주행 ID로 분할하고 이벤트 ID로 정렬합니다.

TRIP_ID	EVENT_ID	EVENT_TIME	RETURN VALUE
101	1	2017-05-03 12:00:00	NULL
101	2	2017-05-03 12:00:34	34
101	3	2017-05-03 12:02:00	86
102	1	2017-05-03 12:00:00	NULL
102	2	2017-05-03 12:01:56	116
102	3	2017-05-03 12:02:00	4

첫 번째 및 네 번째 행의 lag 값이 지정된 파티션 밖에 있으므로 함수가 기본값인 NULL 값 2개를 반환했습니다.

## LAST

선택한 포트의 마지막 행을 반환합니다. 필요한 경우 필터를 적용하여 PowerCenter 통합 서비스가 읽는 행을 제한할 수 있습니다. LAST 안에는 다른 집계 함수 하나만 중첩할 수 있습니다.

### 구문

```
LAST( value [, filter_condition ] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	이진을 제외한 모든 데이터 유형. 마지막 행을 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

포트의 마지막 행.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

### 예

다음 식은 ITEMS\_NAME 포트에서 가격이 \$10.00보다 큰 마지막 행을 반환합니다.

```
LAST( ITEM_NAME, ITEM_PRICE > 10 )
```

ITEM_NAME	ITEM_PRICE
Flashlight	35.00
Navigation Compass	8.05
Regulator System	150.00
Flashlight	29.00
Depth/Pressure Gauge	88.00
Vest	31.00
RETURN VALUE:Vest	



## LAST\_DAY

포트의 각 날짜에 대한 월의 마지막 날 날짜를 반환합니다.

### 구문

`LAST_DAY( date )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
날짜	필수	날짜/시간 데이터 유형. 월의 마지막 날을 반환할 날짜를 전달합니다. 날짜로 평가되는 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

날짜. 이 함수에 전달한 날짜 값에 대한 월의 마지막 날이 반환됩니다.

선택한 포트의 값이 Null인 경우 NULL이 반환됩니다.

### Null

값이 NULL인 경우 LAST\_DAY에서 행이 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

### 그룹 기준

LAST\_DAY는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다. 그룹 기준 포트가 없는 경우 LAST\_DAY 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

### 예

다음 식은 ORDER\_DATE 포트의 각 날짜에 대한 월의 마지막 날을 반환합니다.

`LAST_DAY( ORDER_DATE )`

ORDER_DATE	RETURN VALUE
Apr 1 1998 12:00:00AM	Apr 30 1998 12:00:00AM
Jan 6 1998 12:00:00AM	Jan 31 1998 12:00:00AM
Feb 2 1996 12:00:00AM	Feb 29 1996 12:00:00AM (Leap year)
NULL	NULL
Jul 31 1998 12:00:00AM	Jul 31 1998 12:00:00AM

TO\_DATE를 중첩하여 문자열 값을 날짜로 변환할 수 있습니다. TO\_DATE에는 항상 시간 정보가 포함됩니다. 시간 값이 없는 문자열을 전달할 경우 반환되는 날짜에는 시간 00:00:00이 포함됩니다.

다음 예는 각 주문 날짜에 대한 월의 마지막 날을 문자열과 동일한 형식으로 반환합니다.

```
LAST_DAY( TO_DATE( ORDER_DATE, 'DD-MON-YY' ) )
```

ORDER_DATE	RETURN VALUE
'18-NOV-98'	Nov 30 1998 00:00:00
'28-APR-98'	Apr 30 1998 00:00:00
NULL	NULL
'18-FEB-96'	Feb 29 1996 00:00:00 ( <i>Leap year</i> )

## LEAD

식 변환에서 현재 행 다음의 행에 대한 오프셋 수인 값을 반환합니다. Hadoop 환경의 Spark 엔진에서 매핑을 실행하는 경우 이 함수를 사용하여 현재 행의 값과 이후 행의 값을 비교할 수 있습니다.

데이터 집합에서 lead 값은 현재 행의 뒤에 표시됩니다.

**참고:** 변환에서 LEAD 함수를 사용하는 경우 창 작업을 사용하도록 변환을 구성해야 합니다. 창 작업 속성은 데이터의 분할 및 정렬 방법을 정의합니다.

### 구문

```
LEAD ( column_name, offset, default )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>column_name</i>	필수	함수가 작동하는 대상 열 또는 식입니다.
<i>offset</i>	필수	정수 데이터 유형. 값을 가져올 현재 행 뒤의 행 수입니다.
<i>default</i>	선택 사항	오프셋이 파티션 또는 테이블의 범위 밖에 있는 경우 반환할 기본값입니다. 기본값은 NULL입니다.

### 반환 값

지정된 *column\_name*의 데이터 유형.

반환 값이 지정된 파티션의 범위 밖에 있는 경우 *Default*.

*default*가 생략되었거나 NULL로 설정된 경우 NULL.

## 예

다음 식은 각 직원에 대해 다음 직원이 고용된 날짜를 반환합니다.

LEAD ( HIRE\_DATE, 1, NULL )

EMPLOYEE	HIRE_DATE	RETURN VALUE
Hynes	2012/12/07	2014/05/18
Williams	2014/05/18	2015/07/24
Pritchard	2015/07/24	2015/12/24
Snyder	2015/12/24	2016/11/15
Troy	2016/11/15	2017/08/10
Randolph	2017/08/10	NULL

마지막 행에 사용할 수 있는 lead 값이 없으므로 함수가 기본값인 NULL을 반환했습니다.

다음 식은 2년에 대한 1분기와 3분기 사이의 영업 할당량 값 차이를 반환합니다.

LEAD ( Sales\_Quota, 2, 0 ) - Sales\_Quota

데이터를 연도로 분할하고 분기로 정렬합니다.

YEAR	QUARTER	SALES_QUOTA	QUOTA_DIFF
2016	1	300	7700
2016	2	7000	0
2016	3	8000	0
2017	1	5000	4000
2017	2	400	0
2017	3	9000	0

2분기 및 3분기의 lead 값이 지정된 파티션 밖에 있으므로 함수가 "0" 값을 반환했습니다.

## LEAST

입력 값 목록에서 가장 작은 값이 반환됩니다. 기본적으로 일치하는 항목을 찾을 때는 대/소문자를 구분합니다.

### 구문

LEAST( value1, [value2, ..., valueN,] )

다음 테이블에는 이 명령의 인수에 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	이진수를 제외한 모든 데이터 유형. 데이터 유형은 다른 값과 호환되어야 합니다. 다른 값과 비교할 값입니다. 값 인수를 하나 이상 입력해야 합니다. 해당 값이 숫자이고 다른 입력 값은 다른 숫자 데이터 유형인 경우 가장 많은 전체 자릿수가 모든 값에 사용됩니다. 예를 들어 일부 값의 데이터 유형이 정수이고 다른 값의 데이터 유형은 배정밀도인 경우 PowerCenter 통합 서비스가 값을 배정밀도로 변환합니다.
<i>CaseFlag</i>	선택 사항	정수여야 합니다. 입력 값 인수가 문자열 값인 경우 값을 지정합니다. 이 함수의 인수가 대/소문자를 구분하는지를 결정합니다. 유효한 모든 변환 식을 입력할 수 있습니다. CaseFlag가 0 이외의 숫자인 경우 이 함수는 대/소문자를 구분합니다. CaseFlag가 0인 경우 이 함수는 대/소문자를 구분하지 않습니다. 기본값은 대/소문자 구분입니다.

## 반환 값

*value1*(입력 값 중 가장 작은 값인 경우), *value2*(입력 값 중 가장 작은 값인 경우).

인수 중 하나가 Null인 경우 NULL이 반환됩니다.

## 예제

다음 식은 주문된 항목 중 수량이 가장 작은 항목을 반환합니다.

```
LEAST( QUANTITY1, QUANTITY2, QUANTITY3 )
```

QUANTITY1	QUANTITY2	QUANTITY3	RETURN VALUE
150	756	27	27
			NULL
5000	97	17	17
120	1724	965	120

## LENGTH

문자열 중 후행 공백을 포함하여 모든 문자의 개수를 반환합니다.

## 구문

```
LENGTH( string )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자열 데이터 유형. 평가할 문자열입니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

문자열의 길이를 나타내는 정수.

함수에 전달된 값이 NULL인 경우 NULL입니다.

### 예

다음 식은 각 고객 이름의 길이를 반환합니다.

```
LENGTH( CUSTOMER_NAME )
```

CUSTOMER_NAME	RETURN VALUE
Bernice Davis	13
NULL	NULL
John Baer	9
Greg Brown	10

### LENGTH 함수에 대한 팁

LENGTH 함수는 빈 문자열 조건을 테스트할 때 사용합니다. 고객 이름이 비어 있는 필드를 찾으려는 경우 다음과 같은 식을 사용합니다.

```
IIF( LENGTH( CUSTOMER_NAME ) = 0, 'EMPTY STRING' )
```

Null 필드가 있는지 테스트하려면 ISNULL을 사용합니다. 공백이 있는지 테스트하려면 IS\_SPACES를 사용합니다.

## LN

숫자 값의 자연 대수를 반환합니다. 예를 들어 LN(3)은 1.098612를 반환합니다. 이 함수는 주로 비즈니스 데이터 외에 과학 데이터를 분석하는 데 사용됩니다.

이 함수는 EXP 함수의 역수입니다.

### 구문

```
LN( numeric_value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 0보다 큰 양수여야 합니다. 자연 대수를 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

배정밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 NUMBERS 포트의 모든 값에 대한 자연 대수를 반환합니다.

LN( NUMBERS )

NUMBERS	RETURN VALUE
10	2.302585092994
125	4.828313737302
0.96	-0.04082199452026
NULL	NULL
-90	Error. (The Integration Service does not write row.)
0	Error. (The Integration Service does not write row.)

**참고:** 음수 또는 0을 전달하면 PowerCenter 통합 서비스가 오류를 표시하고 해당 행을 쓰지 않습니다. *numeric\_value*는 0보다 큰 양수여야 합니다.

## LOG

숫자 값의 대수를 반환합니다. 주로 이 함수는 비즈니스 데이터 외에 과학 데이터를 분석하는 데 사용됩니다.

## 구문

LOG( *base*, *exponent* )

다음 테이블에는 이 명령의 인수 설명이 있습니다.

인수	필수/ 선택 사항	설명
기본	필수	대수의 밑수. 0 또는 1이 아닌 양의 숫자 값이어야 합니다. 0 또는 1이 아닌 양수로 평가되는 모든 유효한 변환 식을 입력할 수 있습니다.
지수	필수	대수의 지수. 0보다 큰 양의 숫자 값이어야 합니다. 0보다 큰 양수로 평가되는 모든 유효한 변환 식을 입력할 수 있습니다.

## 반환 값

배정밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 NUMBERS 포트의 모든 값에 대한 대수를 반환합니다.

LOG( BASE, EXPONENT )

BASE	EXPONENT	RETURN VALUE
15	1	0
.09	10	-0.956244644696599
NULL	18	NULL
35.78	NULL	NULL
-9	18	Error. (PowerCenter 통합 서비스 does not write the row.)
0	5	Error. (PowerCenter 통합 서비스 does not write the row.)
10	-2	Error. (PowerCenter 통합 서비스 does not write the row.)

음수, 0, 또는 1을 밑수 값으로 전달하거나 지수에 대한 음수 값을 전달하면 PowerCenter 통합 서비스가 오류를 표시하고 해당 행을 쓰지 않습니다.

## LOOKUP

조회 소스 열의 값을 검색합니다.

LOOKUP 함수는 조회 소스의 데이터를 사용자가 지정한 값과 비교합니다. PowerCenter 통합 서비스가 조회 테이블에서 검색 값을 찾으면 조회 테이블과 동일한 행에서 지정된 열의 값을 반환합니다.

LOOKUP 함수를 사용하는 매핑에 기반하는 세션을 작성하는 경우 세션 속성에서 \$Source 연결 값과 \$Target 연결 값에 대한 데이터베이스 연결을 지정해야 합니다. 식 변환에서 조회 함수의 유효성을 검사하려면 조회 정의가 매핑에 있는지 확인합니다.

**참고:** 이 함수는 맵렛에서 지원되지 않습니다.

## 조희 변환 또는 LOOKUP 함수 사용

PowerCenter 매핑의 값을 조희할 때는 LOOKUP 함수 대신 조희 변환을 사용합니다. LOOKUP 함수를 매핑에 사용하는 경우 세션 속성에서 3.5 호환성에 대한 조희 캐싱 옵션을 실행해야 합니다. 이 옵션은 조희 변환을 작성하지 않고 LOOKUP 함수를 계속해서 사용하려는 PowerMart 3.5 사용자를 위한 것입니다. 자세한 내용은 *PowerCenter 변환 가이드*의 "조희 변환"을 참조하십시오.

LOOKUP 함수 안에 하나의 조희 테이블에 대한 여러 개의 검색을 정의할 수 있습니다. 그러나 조희 값은 각 검색에서 일치하는 값을 찾은 경우에만 반환됩니다.

### 구문

LOOKUP( *result*, *search1*, *value1* [, *search2*, *value2*]... )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>결과</i>	필수	이진을 제외한 모든 데이터 유형. 검색과 동일한 조희 테이블의 출력 포트여야 합니다. 검색에서 일치하는 값을 찾을 경우 반환할 값을 지정합니다. 이 인수 앞에는 항상 참조 한정자 :TD가 있어야 합니다.
<i>search1</i>	필수	<i>value1</i> 과 일치하는 데이터 유형. 결과와 동일한 조희 테이블의 출력 포트여야 합니다. 값과 일치시킬 값을 지정합니다. 이 인수 앞에는 항상 참조 한정자 :TD가 있어야 합니다.
<i>value1</i>	필수	이진을 제외한 모든 데이터 유형. <i>search1</i> 데이터 유형과 일치해야 합니다. <i>search1</i> 에서 지정한 조희 소스 열에서 검색할 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

모든 검색에서 일치하는 값을 찾은 경우 *결과*가 반환됩니다. PowerCenter 통합 서비스가 일치하는 값을 찾은 경우 *search1* 인수와 동일한 행의 결과가 반환됩니다.

검색에서 일치하는 값을 찾지 못한 경우 NULL이 반환됩니다.

검색에서 둘 이상의 일치하는 값을 찾은 경우 오류가 반환됩니다.

### 예

다음 식은 조희 소스 :TD.SALES에서 특정 항목 ID와 가격을 검색하고 두 검색에서 일치점을 찾은 경우 항목 이름을 반환합니다.

LOOKUP( :TD.SALES.ITEM\_NAME, :TD.SALES.ITEM\_ID, 10, :TD.SALES.PRICE, 15.99 )

ITEM_NAME	ITEM_ID	PRICE
Regulator	5	100.00
Flashlight	10	15.99
Halogen Flashlight	15	15.99



ITEM_NAME	ITEM_ID	PRICE
NULL	20	15.99

**RETURN VALUE:** Flashlight

## LOOKUP 함수에 대한 팁

char과 varchar 값을 비교하는 경우 LOOKUP 함수는 두 행이 일치하는 경우에만 결과를 반환합니다. 즉, 각 행의 값과 길이가 모두 일치해야 합니다. 조회 소스가 지정된 길이의 여백이 있는 char 값이고 조회 검색이 varchar 값인 경우 RTRIM 함수를 사용하여 조회 소스에서 후행 공백을 잘라내야 값이 조회 검색과 일치합니다.

```
LOOKUP(:TD.ORDERS.PRICE, :TD.ORDERS.ITEM, RTRIM( ORDERS.ITEM, ' '))
```

LOOKUP 함수의 *결과* 및 *search* 인수에는 :TD 참조 한정자를 사용합니다.

```
LOOKUP(:TD.ORDERS.ITEM, :TD.ORDERS.PRICE, ORDERS.PRICE, :TD.ORDERS.QTY, ORDERS.QTY)
```

## LOWER

문자열의 대문자를 소문자로 변환합니다.

### 구문

```
LOWER( string )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
문자열	필수	모든 문자열 값입니다. 이 인수는 소문자로 반환할 문자열 값을 전달합니다. 문자열로 평가되는 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

소문자 문자열. 데이터에 다중 바이트 문자가 포함되는 경우 반환 값은 통합 서비스의 코드 페이지 및 데이터 이동 모드에 따라 다릅니다.

선택한 포트의 값이 Null인 경우 NULL이 반환됩니다.

### 예

다음 식은 모든 이름을 소문자로 반환합니다.

```
LOWER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
antonia	antonia
NULL	NULL

FIRST_NAME	RETURN VALUE
THOMAS	thomas
PierRe	pierre
BERNICE	bernice

## LPAD

문자열 앞에 공백 또는 문자 집합을 추가하여 해당 문자열을 지정된 길이로 설정합니다.

### 구문

LPAD( *first\_string*, *length* [,*second\_string*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>first_string</i>	필수	문자열을 입력할 수 있습니다. 변경할 문자열입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>length</i>	필수	양의 정수 리터럴이어야 합니다. 이 인수는 각 문자열의 길이를 지정합니다.
<i>second_string</i>	선택 사항	모든 문자열 값을 입력할 수 있습니다. <i>first_string</i> 값의 왼쪽에 추가할 문자입니다. 유효한 모든 변환 식을 입력할 수 있습니다. 특정 문자열 리터럴을 입력할 수 있습니다. 그러나 문자열의 시작에 추가할 문자를 작은따옴표로 묶어야 합니다(예: 'abc'). 이 인수는 대/소문자를 구분합니다. <i>second_string</i> 을 생략할 경우 함수가 첫 번째 문자열의 시작에 공백을 추가합니다.

### 반환 값

지정된 길이의 문자열.

함수에 전달된 값이 NULL이거나 길이가 음수인 경우 NULL이 반환됩니다.

### 예

다음 식은 숫자 앞에 선행 0을 추가하여 숫자를 6자리로 표준화합니다.

LPAD( PART\_NUM, 6, '0' )

PART_NUM	RETURN VALUE
702	000702
1	000001
0553	000553
484834	484834

LPAD는 왼쪽에서 오른쪽으로 길이를 계산합니다. 첫 번째 문자열이 이 길이보다 길면 LPAD가 오른쪽에서 왼쪽으로 문자열을 잘라냅니다. 예를 들어 LPAD('alphabetical', 5, 'x')는 문자열 'alpha'를 반환합니다.

두 번째 문자열이 지정된 길이를 반환하는 데 필요한 전체 문자 수보다 길면 LPAD는 두 번째 문자열의 부분을 사용합니다.

```
LPAD( ITEM_NAME, 16, '*.*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	*.*.*.Flashlight
Compass	*.*.*.*.Compass
Regulator System	Regulator System
Safety Knife	*.*.Safety Knife

## LTRIM

문자열 앞에서 공백이나 문자를 제거합니다. LTRIM을 식 또는 업데이트 전략 변환에서 IIF 또는 DECODE와 함께 사용하면 대상 테이블의 공백을 방지할 수 있습니다.

식에서 *trim\_set* 매개 변수를 지정하지 않은 경우는 다음과 같습니다.

- 유니코드 모드에서 LTRIM은 문자열의 시작에서 싱글바이트 및 더블바이트 공백을 제거합니다.
- ASCII 모드에서 LTRIM은 단일 바이트 공백만 제거합니다.

LTRIM을 사용하여 문자열에서 문자를 제거하는 경우 LTRIM은 *trim\_set*과 문자열 인수의 각 문자를 문자열의 왼쪽부터 한 문자씩 비교합니다. 문자열에서 *trim\_set*의 문자와 일치하는 문자는 제거됩니다. LTRIM은 *trim\_set*에 일치하는 문자가 없을 때까지 계속해서 문자를 비교하고 제거합니다. 그런 다음 일치하는 문자를 포함하지 않는 문자열을 반환합니다.

### 구문

```
LTRIM( string [, trim_set] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	모든 문자열 값입니다. 수정할 문자열을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 문자열의 시작에서 문자를 제거하기 전에 연산자를 사용하여 문자열을 비교하거나 연결합니다.
<i>trim_set</i>	선택 사항	모든 문자열 값입니다. 첫 번째 문자열의 시작에서 제거할 문자를 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 문자열을 입력할 수도 있습니다. 그러나 문자열의 시작에서 제거할 문자를 작은따옴표로 묶어야 합니다(예: 'abc'). 두 번째 문자열을 생략할 경우 문자열의 시작에서 공백이 제거됩니다. LTRIM은 대/소문자를 구분합니다. 예를 들어 문자열 'Alfredo'에서 'A' 문자를 제거하려는 경우 'a'가 아니라 'A'를 입력해야 합니다.

## 반환 값

문자열. 지정한 문자를 포함하는 문자열 값이 *trim\_set* 인수에서 제거됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다. *trim\_set*이 NULL인 경우 NULL이 반환됩니다.

## 예

다음 식은 문자 'S'와 '.'를 LAST\_NAME 포트의 문자열에서 제거합니다.

```
LTRIM( LAST_NAME, 'S.' )
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Osborne	Osborne
NULL	NULL
S. MacDonald	MacDonald
Sawyer	awyer
H. Bender	H. Bender
Steadman	teadman

LTRIM은 S. MacDonald에서 'S.'를 제거하고 Sawyer와 Steadman에서 'S'를 제거하지만 H. Bender에서 마침표는 제거하지 않습니다. LTRIM은 사용자가 *trim\_set* 인수에 지정한 문자 집합을 한 글자씩 검색하기 때문입니다. 문자열의 첫 번째 문자가 *trim\_set*의 첫 번째 문자와 일치할 경우 LTRIM은 이 문자를 제거합니다. 그런 다음 문자열의 두 번째 문자를 확인합니다. *trim\_set*의 두 번째 문자와 일치하면 LTRIM이 이 문자를 제거합니다. 문자열의 첫 번째 문자가 *trim\_set*에서 해당하는 문자와 일치하지 않는 경우 LTRIM은 문자열을 반환하고 다음 행을 평가합니다.

H. Bender의 예에서 H는 *trim\_set* 인수의 어떤 문자와도 일치하지 않으므로 LTRIM은 LAST\_NAME 포트의 문자열을 반환하고 다음 행으로 이동합니다.

## LTRIM 함수에 대한 팁

RTRIM 및 LTRIM 함수를 || 또는 CONCAT와 함께 사용하면 두 문자열을 연결한 후에 선행 및 후행 공백이 제거됩니다.

LTRIM을 중첩하여 여러 문자 집합을 제거할 수도 있습니다. 예를 들어 이름 열에서 선행 공백과 문자 'T'를 제거하려는 경우 다음과 유사한 식을 작성할 수 있습니다.

```
LTRIM( LTRIM( NAMES ), 'T' )
```

## MAKE\_DATE\_TIME

입력 값을 기준으로 날짜 및 시간을 반환합니다.

## 구문

```
MAKE_DATE_TIME( year, month, day, hour, minute, second, nanosecond )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
년도	필수	숫자 데이터 유형. 4자리 양의 정수입니다. 이 함수에 2자리 연도를 전달하면 PowerCenter 통합 서비스가 연도의 처음 두 자리로 "00"을 반환합니다.
월	필수	숫자 데이터 유형. 1과 12 사이의 양의 정수입니다(1월=1 및 12월=12). <b>참고:</b> Spark 엔진은 MAKE_DATE_TIME 함수의 월 인수가 잘못된 값을 전달하면 행에 대해 Null 값을 씁니다. 원시 환경에서 데이터 통합 서비스는 행을 거부하고 대상에 쓰지 않습니다.
일	필수	숫자 데이터 유형. 1과 31 사이의 양의 정수입니다(일 수가 31일보다 적은 월인 2, 4, 6, 9 및 11월은 예외).
시간	선택	숫자 데이터 유형. 0과 24 사이의 양의 정수입니다(0=12AM, 12=12PM 및 24=12AM).
분	선택	숫자 데이터 유형. 0과 59 사이의 양의 정수입니다.
초	선택	숫자 데이터 유형. 0과 59 사이의 양의 정수입니다.
nanosecond	선택	숫자 데이터 유형. 0과 999,999,999 사이의 양의 정수입니다.

## 반환 값

MM/DD/YYYY HH24:MI:SS 형식의 날짜. 함수에 연도, 월 또는 일을 전달하지 않은 경우 Null 값이 반환됩니다.

## 예

다음 식은 입력 포트에서 날짜와 시간을 작성합니다.

```
MAKE_DATE_TIME( SALE_YEAR, SALE_MONTH, SALE_DAY, SALE_HOUR, SALE_MIN, SALE_SEC )
```

SALE_YR	SALE_MTH	SALE_DAY	SALE_HR	SALE_MIN	SALE_SEC	RETURN VALUE
2002	10	27	8	36	22	10/27/2002 08:36:22
2000	6	15	15	17		06/15/2000 15:17:00
2003	1	3		22	45	01/03/2003 00:22:45
04	3	30	12	5	10	03/30/0004 12:05:10
99	12	12	5		16	12/12/0099 05:00:16

## MAX(날짜)

포트 또는 그룹 내에서 찾은 가장 늦은 날짜가 반환됩니다. 필터를 적용하여 검색의 행을 제한할 수 있습니다. MAX 안에는 다른 집계 함수 하나만 중첩할 수 있습니다.

MAX를 사용하여 포트 또는 그룹에서 가장 큰 숫자 값이나 가장 큰 문자열 값을 반환할 수도 있습니다.

## 구문

MAX( date [, filter\_condition] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
날짜	필수	날짜/시간 데이터 유형. 최대 날짜를 반환할 날짜를 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
filter_condition	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

날짜.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

## 예

포트 또는 그룹에 대한 최대 날짜를 반환할 수 있습니다. 다음 식은 Flashlight의 최대 주문 날짜를 반환합니다.

MAX( ORDERDATE, ITEM\_NAME='Flashlight' )

ITEM_NAME	ORDER_DATE
Flashlight	Apr 20 1998
Regulator System	May 15 1998
Flashlight	Sep 21 1998
Diving Hood	Aug 18 1998
Flashlight	NULL

## MAX(숫자)

포트 또는 그룹 내에서 찾은 최대 숫자 값이 반환됩니다. 필터를 적용하여 검색의 행을 제한할 수 있습니다. MAX 안에는 다른 집계 함수 하나만 중첩할 수 있습니다. MAX를 사용하여 포트 또는 그룹에서 가장 늦은 날짜 또는 가장 큰 문자열 값을 반환할 수도 있습니다.

## 구문

MAX( numeric\_value [, filter\_condition] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 최대 숫자 값을 반환할 숫자 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## Null

NULL 값은 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

## 그룹 기준

MAX는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 MAX 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

첫 번째 식은 Flashlight의 최대 가격을 반환합니다.

```
MAX( PRICE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	PRICE
Flashlight	10.00
Regulator System	360.00
Flashlight	55.00
Diving Hood	79.00
Halogen Flashlight	162.00
Flashlight	85.00

ITEM_NAME	PRICE
Flashlight	NULL
RETURN VALUE: 85.00	

## MAX(문자열)

포트 또는 그룹 내에서 찾은 가장 큰 문자열 값이 반환됩니다. 필터를 적용하여 검색의 행을 제한할 수 있습니다. MAX 안에는 다른 집계 함수 하나만 중첩할 수 있습니다.

**참고:** MAX 함수는 분류기 변환과 동일한 정렬 순서를 사용합니다. 그러나 MAX 함수는 대/소문자를 구분하고 분류기 변환은 대/소문자를 구분하지 않습니다.

MAX를 사용하여 포트 또는 그룹에서 가장 늦은 날짜 또는 가장 큰 숫자 값을 반환할 수도 있습니다.

### 구문

MAX( *string* [, *filter\_condition*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자열 데이터 유형. 최대 문자열 값을 반환할 문자열 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

문자열.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

### Null

NULL 값은 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

### 그룹 기준

MAX는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 MAX 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.



예

다음 식은 제조업체 ID 104에 대한 최대 항목 이름을 반환합니다.

```
MAX( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME
101	First Stage Regulator
102	Electronic Console
104	Flashlight
104	Battery (9 volt)
104	Rope (20 ft)
104	60.6 cu ft Tank
107	75.4 cu ft Tank
108	Wristband Thermometer

**RETURN VALUE:** Rope (20 ft)

## MD5

입력 값의 체크섬을 계산합니다. 이 함수는 메시지 다이제스트 알고리즘 5(MD5)를 사용합니다. MD5는 128 비트 해시 값을 포함하는 단방향 암호화 해시 함수입니다. 입력 값의 체크섬이 다를 경우 입력 값이 다르다는 결론을 내릴 수 있습니다. MD5는 데이터 무결성을 확인하는 데 사용됩니다.

구문

```
MD5( value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	문자열 또는 이진 데이터 유형. 체크섬을 계산할 값입니다. 입력 값의 대/소문자는 반환 값에 영향을 미칩니다. 예를 들어 MD5(informatica)와 MD5(Informatica)는 다른 값을 반환합니다.

반환 값

16진수의 자릿수(0-9 및 a-f)의 고유한 32자 문자열.

입력이 Null 값인 경우 NULL이 반환됩니다.

## 예

변경된 데이터를 데이터베이스에 쓰려고 합니다. MD5를 사용하여 소스에서 읽는 데이터 행에 대한 체크섬 값을 생성합니다. 세션을 실행하면 이전에 생성한 체크섬 값과 새 체크섬 값이 비교됩니다. 그런 다음 업데이트된 체크섬 값이 대상에 기록됩니다. 사용자는 업데이트된 체크섬 값을 보고 데이터가 변경된 것을 알 수 있습니다.

## 팁

반환 값을 해시 키로 사용할 수 있습니다.

## MEDIAN

선택한 포트의 모든 값에 대한 중수를 반환합니다.

포트에 짝수 값이 있는 경우 중수는 모든 값을 일직선상에 순서대로 배치했을 때 중간에 있는 값 2개의 평균입니다. 포트에 홀수 값이 있는 경우 중수는 중간에 있는 숫자입니다.

MEDIAN 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 숫자 데이터 유형을 반환해야 합니다.

PowerCenter 통합 서비스는 데이터의 모든 행을 읽고 중수 계산을 수행합니다. 계산을 수행하기 위해 데이터 행을 읽는 프로세스 때문에 성능이 저하될 수 있습니다. 필요한 경우 필터를 적용하여 중수를 계산할 때 읽을 행을 제한할 수 있습니다.

## 구문

```
MEDIAN( numeric_value [, filter_condition ] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 중수를 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우 NULL이 반환됩니다. 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우를 예로 들 수 있습니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## Null

값이 NULL인 경우 MEDIAN에서 행이 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

## 그룹 기준

MEDIAN은 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 MEDIAN 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

모든 부서의 급여에 대한 중수를 계산하려면 다음 식을 지정하는 포트를 사용하여 부서로 그룹화된 집계 변환을 작성합니다.

```
MEDIAN( SALARY )
```

다음 식은 Stabilizing Vest 주문에 대한 중수 값을 반환합니다.

```
MEDIAN( SALES, ITEM = 'Stabilizing Vest' )
```

ITEM	SALES
Flashlight	85
Stabilizing Vest	504
Stabilizing Vest	36
Safety Knife	5
Medium Titanium Knife	150
Tank	NULL
Stabilizing Vest	441
Chisel Point Knife	60
Stabilizing Vest	NULL
Stabilizing Vest	1044
Wrist Band Thermometer	110

**RETURN VALUE:** 472.5

## METAPHONE

문자열 값이 인코딩됩니다. 인코딩할 문자열의 길이를 지정할 수 있습니다.

METAPHONE 함수는 영어 알파벳(A-Z) 문자를 인코딩합니다. 대문자와 소문자를 모두 대문자로 인코딩합니다.

METAPHONE 함수는 다음 규칙 목록에 따라 문자를 인코딩합니다.

- 입력 문자열의 첫 번째 문자가 아닌 모음(A, E, I, O 및 U)은 건너뜁니다. METAPHONE('CAR')는 'KR'를 반환하고 METAPHONE('AAR')는 'AR'를 반환합니다.
- 특수 인코딩 지침을 사용합니다.

다음 테이블에는 METAPHONE 인코딩 지침이 나열되어 있습니다.

입력	반환	조건	예
B	- 해당 없음	- M이 앞에 있는 경우	- METAPHONE ('Lamb')는 LM을 반환합니다.
B	- B	- 다른 모든 경우	- METAPHONE ('Box')는 BKS를 반환합니다.
C	- X	- IA 또는 H가 뒤에 있는 경우	- METAPHONE ('Facial')은 FXL를 반환합니다.
C	- S	- I, E 또는 Y가 뒤에 있는 경우	- METAPHONE ('Fence')는 FNS를 반환합니다.
C	- 해당 없음	- S가 앞에 있고 I, E 또는 Y가 뒤에 있는 경우	- METAPHONE ('Scene')은 SN을 반환합니다.
C	- K	- 다른 모든 경우	- METAPHONE ('Cool')은 KL을 반환합니다.
D	- J	- GE, GY 또는 GI가 뒤에 있는 경우	- METAPHONE ('Dodge')는 TJ를 반환합니다.
D	- T	- 다른 모든 경우	- METAPHONE ('David')는 TFT를 반환합니다.
F	- F	- 모든 경우	- METAPHONE ('FOX')는 FKS를 반환합니다.
G	- F	- H가 뒤에 있고 입력 문자열의 첫 번째 문자가 B, D 또는 H가 아닌 경우	- METAPHONE ('Tough')는 TF를 반환합니다.
G	- 해당 없음	- H가 뒤에 있고 입력 문자열의 첫 번째 문자가 B, D 또는 H인 경우	- METAPHONE ('Hugh')는 HF를 반환합니다.
G	- J	- I, E 또는 Y가 뒤에 있고 반복되지 않는 경우	- METAPHONE ('Magic')은 MJK를 반환합니다.
G	- K	- 다른 모든 경우	- METAPHONE ('GUN')은 KN을 반환합니다.
H	- H	- C, G, P, S 또는 T가 앞에 있지 않고 A, E, I 또는 U가 뒤에 있는 경우	- METAPHONE ('DHAT')는 THT를 반환합니다.
H	- 해당 없음	- 다른 모든 경우	- METAPHONE ('Chain')은 XN을 반환합니다.
J	- J	- 모든 경우	- METAPHONE ('Jen')은 JN을 반환합니다.

입력	반환	조건	예
K	- 해당 없음 - K	- C가 앞에 있는 경우 - 다른 모든 경우	- METAPHONE ('Ckim')은 KM을 반환합니다. - METAPHONE ('Kim')은 KM을 반환합니다.
L	- L	- 모든 경우	- METAPHONE ('Laura')는 LR을 반환합니다.
M	- M	- 모든 경우	- METAPHONE ('Maggi')는 MK를 반환합니다.
N	- N	- 모든 경우	- METAPHONE ('Nancy')는 NNS를 반환합니다.
P	- F	- H가 뒤에 있는 경우	- METAPHONE ('Phone')은 FN을 반환합니다.
P	- P	- 다른 모든 경우	- METAPHONE ('Pip')은 PP를 반환합니다.
Q	- K	- 모든 경우	- METAPHONE ('Queen')은 KN을 반환합니다.
R	- R	- 모든 경우	- METAPHONE ('Ray')는 R를 반환합니다.
S	- X	- H, IO, IA 또는 CHW가 뒤에 있는 경우	- METAPHONE ('Cash')는 KX를 반환합니다.
S	- S	- 다른 모든 경우	- METAPHONE ('Sing')은 SNK를 반환합니다.
T	- X	- IA 또는 IO가 뒤에 있는 경우	- METAPHONE ('Patio')는 PX를 반환합니다.
T	- 0 <sup>1</sup>	- H가 뒤에 있는 경우	- METAPHONE ('Thor')는 0R를 반환합니다.
T	- 해당 없음	- CH가 뒤에 있는 경우	- METAPHONE ('Glitch')는 KLTX를 반환합니다.
T	- 화	- 다른 모든 경우	- METAPHINE ('Tim')은 TM을 반환합니다.
V	- F	- 모든 경우	- METAPHONE ('Vin')은 FN을 반환합니다.
W	- W	- A, E, I, O 또는 U가 뒤에 있는 경우	- METAPHONE ('Wang')은 WNK를 반환합니다.
W	- 해당 없음	- 다른 모든 경우	- METAPHONE ('When')은 HN을 반환합니다.
X	- KS	- 모든 경우	- METAPHONE ('Six')는 SKS를 반환합니다.

입력	반환	조건	예
Y	- 연도	- A, E, I, O 또는 U가 뒤에 있는 경우	- METAPHONE ('Yang')은 YNK를 반환합니다.
Y	- 해당 없음	- 다른 모든 경우	- METAPHONE ('Bobby')는 BB를 반환합니다.
Z	- S	- 모든 경우	- METAPHONE ('Zack')은 SK를 반환합니다.

#### 1. 정수 0.

- 입력 문자열의 처음 두 문자가 다음 값 중 하나인 경우 첫 문자를 건너뛰고 나머지 문자열을 인코딩합니다.
  - **KN.** 예를 들어 METAPHONE('KNOT')는 'NT'를 반환합니다.
  - **GN.** 예를 들어 METAPHONE('GNOB')는 'NB'를 반환합니다.
  - **PN.** 예를 들어 METAPHONE('PNRX')는 'NRKS'를 반환합니다.
  - **AE.** 예를 들어 METAPHONE('AERL')은 'ERL'을 반환합니다.
- "C" 이외의 문자가 입력 문자열에서 두 번 이상 발생하는 경우 첫 일치 항목만 인코딩합니다. 예를 들어 METAPHONE('BBOX')는 'BKS'를 반환하고 METAPHONE('CCOX')는 'KKKS'를 반환합니다.

#### 구문

METAPHONE( *string* [, *length*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>string</i>	필수	문자 문자열이어야 합니다. 인코딩할 값을 전달합니다. 첫 번째 문자는 영어 알파벳(A-Z)의 문자여야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다. <i>string</i> 에서 알파벳 이외의 문자는 건너뛸니다.
<i>length</i>	선택 사항	0보다 큰 정수여야 합니다. 인코딩할 <i>string</i> 의 문자 수를 지정합니다. 유효한 모든 변환 식을 입력할 수 있습니다. <i>length</i> 가 0이거나 <i>string</i> 의 길이보다 큰 값인 경우 전체 입력 문자열이 인코딩됩니다. 기본값은 0입니다.

#### 반환 값

문자열.

다음 조건 중 하나가 참인 경우 NULL이 반환됩니다.

- 함수에 전달된 모든 값이 NULL입니다.
- 문자열에 영어 알파벳 문자가 없습니다.
- 문자열이 비어 있습니다.

## 예

다음 식은 EMPLOYEE\_NAME 포트의 처음 두 문자를 문자열로 인코딩합니다.

```
METAPHONE( EMPLOYEE_NAME, 2 )
```

Employee_Name	Return Value
John	JH
*@#\$	NULL
P\$%%oc&&KMNL	PK

다음 식은 EMPLOYEE\_NAME 포트의 처음 네 문자를 문자열로 인코딩합니다.

```
METAPHONE( EMPLOYEE_NAME, 4 )
```

Employee_Name	Return Value
John	JHN
1ABC	ABK
*@#\$	NULL
P\$%%oc&&KMNL	PKKM

## MIN(날짜)

포트 또는 그룹에서 찾은 가장 이른 날짜를 반환합니다. 필터를 적용하여 검색의 행을 제한할 수 있습니다. MIN 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 날짜 데이터 유형을 반환해야 합니다. MIN을 사용하여 포트 또는 그룹에서 가장 작은 숫자 값이나 가장 작은 문자열 값을 반환할 수도 있습니다.

### 구문

```
MIN( date [, filter_condition] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>날짜</i>	필수	날짜/시간 데이터 유형. 최소값을 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

값 인수가 날짜인 경우 날짜가 반환됩니다.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

## Null

단일 값이 NULL인 경우 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

## 그룹 기준

MIN은 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 MIN 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

다음 식은 Flashlight의 가장 오래된 주문 날짜를 반환합니다.

```
MIN( ORDER_DATE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	ORDER_DATE
Flashlight	Apr 20 1998
Regulator System	May 15 1998
Flashlight	Sep 21 1998
Diving Hood	Aug 18 1998
Halogen Flashlight	Feb 1 1998
Flashlight	Oct 10 1998
Flashlight	NULL

**RETURN VALUE:** Feb 1 1998

## MIN(숫자)

포트 또는 그룹에서 찾은 최소 숫자 값이 반환됩니다. 필터를 적용하여 검색의 행을 제한할 수 있습니다.

MIN 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 숫자 데이터 유형을 반환해야 합니다.

MIN을 사용하여 포트 또는 그룹에서 가장 늦은 날짜 또는 가장 작은 문자열 값을 반환할 수도 있습니다.

## 구문

```
MIN( numeric_value [, filter_condition] )
```



다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 최소값을 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## Null

단일 값이 NULL인 경우 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

## 그룹 기준

MIN은 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 MIN 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

다음 식은 Flashlight의 최소 가격을 반환합니다.

```
MIN ( PRICE, ITEM_NAME='Flashlight' )
```

ITEM_NAME	PRICE
Flashlight	10.00
Regulator System	360.00
Flashlight	55.00
Diving Hood	79.00
Halogen Flashlight	162.00
Flashlight	85.00

ITEM_NAME	PRICE
Flashlight	NULL
RETURN VALUE: 10.00	

## MIN(문자열)

포트 또는 그룹에서 찾은 가장 작은 문자열 값을 반환합니다. 필터를 적용하여 검색의 행을 제한할 수 있습니다. MIN 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 문자열 데이터 유형을 반환해야 합니다.

**참고:** MIN 함수는 분류기 변환과 동일한 정렬 순서를 사용합니다. 그러나 MIN 함수는 대/소문자를 구분하지만 분류기 변환은 대/소문자를 구분하지 않습니다.

MIN을 사용하여 포트 또는 그룹에서 가장 늦은 날짜 또는 최소 숫자 값을 반환할 수도 있습니다.

### 구문

MIN( *string* [, *filter\_condition*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자열 데이터 유형. 최소값을 반환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

문자열 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

### Null

단일 값이 NULL인 경우 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

### 그룹 기준

MIN은 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 MIN 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

다음 식은 제조업체 ID 104에 대한 최소 항목 이름을 반환합니다.

```
MIN ( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME
101	First Stage Regulator
102	Electronic Console
104	Flashlight
104	Battery (9 volt)
104	Rope (20 ft)
104	60.6 cu ft Tank
107	75.4 cu ft Tank
108	Wristband Thermometer

**RETURN VALUE:** 60.6 cu ft Tank

## MOD

나누기 계산의 나머지를 반환합니다. 예를 들어 MOD(8,5)는 3을 반환합니다.

### 구문

```
MOD( numeric_value, divisor )
```

다음 테이블에는 이 명령의 인수와 설명이 포함되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 나눌 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>divisor</i>	필수	나눌 숫자 값입니다. 제수는 0이 될 수 없습니다.

### 반환 값

함수에 전달하는 데이터 유형의 숫자 값이 반환됩니다. 제수로 나눈 숫자 값의 나머지가 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 PRICE 포트의 값을 QTY 포트의 값을 나눈 값의 모듈러스를 반환합니다.

MOD( PRICE, QTY )

PRICE	QTY	RETURN VALUE
10.00	2	0
12.00	5	2
9.00	2	1
15.00	3	0
NULL	3	NULL
20.00	NULL	NULL
25.00	0	Error. Integration Service does not write row.

0으로 나눌 수는 없으므로 마지막 행(25, 0)에서 오류가 생성되었습니다. 수량이 0이 아닌 경우에만 Price를 Quantity로 나눈 모듈러스를 반환하는 다음과 유사한 식을 작성하면 0으로 나누는 것을 방지할 수 있습니다. 수량이 0인 경우 NULL이 반환됩니다.

MOD( PRICE, IIF( QTY = 0, NULL, QTY ) )

PRICE	QTY	RETURN VALUE
10.00	2	0
12.00	5	2
9.00	2	1
15.00	3	0
NULL	3	NULL
20.00	NULL	NULL
25.00	0	NULL

IIF 함수가 NULL을 QTY 포트의 0으로 바꾸므로 마지막 행(25, 0)에서 오류가 아닌 NULL이 생성되었습니다.

## MOVINGAVG

특정 행 집합의 평균(행별)이 반환됩니다. 원하는 경우 이동 평균을 계산하기 전에 행을 필터링할 조건을 적용할 수 있습니다.

## 구문

`MOVINGAVG( numeric_value, rowset [, filter_condition] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 이동 평균을 계산할 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<b>행 집합</b>	필수	0보다 큰 양의 정수 리터럴이어야 합니다. 이동 평균을 계산할 행 집합을 정의합니다. 예를 들어 데이터 열에 대한 이동 평균을 한 번에 5개 행씩 계산하려면 <code>MOVINGAVG(SALES, 5)</code> 와 같은 식을 작성할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## Null

MOVINGAVG는 이동 평균을 계산할 때 Null 값을 무시합니다. 그러나 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

## 예

다음 식은 Stabilizing Vest에 대한 평균 주문을 Sales 포트의 처음 5개 행을 바탕으로 반환한 다음 마지막 5개 행에 대한 평균을 반환합니다.

`MOVINGAVG( SALES, 5 )`

ROW_NO	SALES	RETURN VALUE
1	600	NULL
2	504	NULL
3	36	NULL
4	100	NULL
5	550	358
6	39	245.8
7	490	243

이 함수는 5개 행 집합에 대한 평균을 반환합니다. 따라서 행 1부터 행 5까지의 평균인 358, 행 2부터 행 6까지의 평균인 245.8과 행 3부터 행 7까지의 평균인 243이 반환됩니다.

## MOVINGSUM

지정한 행 집합의 합계(행별)를 반환합니다.

필요한 경우 이동 합계를 계산하기 전에 행을 필터링할 조건을 적용할 수 있습니다.

### 구문

`MOVINGSUM( numeric_value, rowset [, filter_condition] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 이동 합계를 계산할 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<b>행 집합</b>	필수	0보다 큰 양의 정수 리터럴이어야 합니다. 이동 합계를 계산할 행 집합을 정의합니다. 예를 들어 데이터 열에 대한 이동 합계를 한 번에 5개 행씩 계산하려면 <code>MOVINGSUM(SALES, 5)</code> 와 같은 식을 작성할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 함수가 행을 선택하지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

### Null

MOVINGSUM은 이동 합계를 계산할 때 Null 값을 무시합니다. 그러나 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

### 예

다음 식은 Stabilizing Vest에 대한 주문 합계를 Sales 포트의 처음 5개 행을 바탕으로 반환한 다음 마지막 5개 행에 대한 평균을 반환합니다.

`MOVINGSUM( SALES, 5 )`

ROW_NO	SALES	RETURN VALUE
1	600	NULL
2	504	NULL

ROW_NO	SALES	RETURN VALUE
3	36	NULL
4	100	NULL
5	550	1790
6	39	1229
7	490	1215

이 함수는 5개 행 집합에 대한 합계를 반환합니다. 따라서 행 1부터 행 5까지의 평균인 1790, 행 2부터 행 6까지의 평균인 1229과 행 3부터 행 7까지의 평균인 1215이 반환됩니다.

## NPER

이율이 일정하고 정기적으로 일정하게 불입되는 경우에 대해 투자 기간 개수를 반환합니다.

### 구문

**NPER**( *rate*, *present value*, *payment* [, *future value*, *type*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>비율</i>	필수	분수. 각 기간별 이율입니다. 10진수로 표시됩니다. 이율을 100으로 나눈 후 10진수로 표시합니다.
<i>현재 가치</i>	필수	분수. 미래 불입 금액의 가치에 대한 총액입니다.
<i>불입</i>	필수	분수. 기간별 불입액입니다. 음수여야 합니다.
<i>미래 가치</i>	선택	분수. 마지막 불입 이후 지급받고자 하는 현금 잔액입니다. 이 값을 생략하면 NPER에서 0을 사용합니다.
<i>유형</i>	선택	부울. 불입 시기. 기간 시작일에 불입할 경우 1을 입력합니다. 기간 종료일에 불입할 경우 0을 입력합니다. 기본값은 0입니다. 0 또는 1 이외의 값을 입력하면 PowerCenter 통합 서비스가 값을 1로 처리합니다.

### 반환 값

숫자.

### 예

투자의 현재 가치는 \$500입니다. \$2000씩 불입되며 투자의 미래 가치는 \$20,000입니다. 다음 식에서는 불입 기간 개수로 9가 반환됩니다.

**NPER** ( 0.015, -500, -2000, 20000, TRUE )

## 참고

기간별 이율을 계산하려면 연이율을 연간 불입 횟수로 나눕니다. 예를 들어, 연이율이 15%이고 매월 지급되는 경우 비율 인수값은 15%로 12로 나누면 됩니다. 매년 불입되는 경우 비율 인수값은 15%입니다.

불입값과 현재값은 사용자가 불입하는 금액이므로 음수입니다.

## PERCENTILE

숫자 그룹에서 주어진 백분위수에 속하는 값이 계산됩니다. PERCENTILE 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 숫자 데이터 유형을 반환해야 합니다.

PowerCenter 통합 서비스는 데이터의 모든 행을 읽고 백분위수 계산을 수행합니다. 계산을 수행하기 위해 행을 읽는 프로세스 때문에 성능이 저하될 수 있습니다. 필요한 경우 필터를 적용하여 백분위수를 계산할 때 읽을 행을 제한할 수 있습니다.

## 구문

PERCENTILE( *numeric\_value*, *percentile* [, *filter\_condition* ] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 백분위수를 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
백분위수	필수	0과 100 사이(경계값 포함)의 정수. 계산할 백분위수를 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 0-100의 범위를 벗어난 숫자를 전달하면 PowerCenter 통합 서비스가 오류를 표시하고 해당 행을 쓰지 않습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## Null

값이 NULL인 경우 PERCENTILE에서 행이 무시됩니다. 그러나 그룹의 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.



## 그룹 기준

PERCENTILE은 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 PERCENTILE 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

### 예

PowerCenter 통합 서비스는 다음 논리를 사용하여 백분위수를 계산합니다.

$$i = \frac{(x + 1) \times \text{percentile}}{100}$$

이 수식에는 다음 지침을 사용합니다.

- $x$ 는 값 그룹에서 백분위수를 계산할 요소의 수입니다.
- $i$ 가 1보다 작으면 목록에서 첫 번째 요소의 값이 반환됩니다.
- $i$ 가 정수 값이면 목록에서  $i$ 번째 요소의 값이 반환됩니다.
- 그렇지 않은 경우  $n$  값이 반환됩니다.

$$n = [\lfloor i \rfloor \text{th?element} \times (\lceil i \rceil - i)] + [\lceil i \rceil \text{th?element} \times (i - \lfloor i \rfloor)]$$

다음 식은 \$50,000보다 큰 급여 중에서 백분위수가 75번째인 급여를 반환합니다.

PERCENTILE( SALARY, 75, SALARY > 50000 )

#### SALARY

125000.0

27900.0

100000.0

NULL

55000.0

9000.0

85000.0

86000.0

48000.0

99000.0

**RETURN VALUE:** 106250.0

## PMT

불입 금액과 이율이 일정한 경우, 대출 불입 금액을 반환합니다.

### 구문

`PMT( rate, terms, present value[, future value, type] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
비율	필수	분수. 각 기간에 대한 대출 이율입니다. 10진수로 표시됩니다. 이율을 100으로 나눈 후 10진수로 표시합니다.
terms	필수	분수. 기간 개수 또는 불입 횟수. 0보다 커야 합니다.
현재 가치	필수	분수. 대출 원금입니다.
미래 가치	선택	분수. 마지막 불입 이후에 달성하고자 하는 잔액입니다. 이 값을 생략하면 PMT가 0을 사용합니다.
유형	선택	부울. 불입 시기. 기간 시작일에 불입할 경우 1을 입력합니다. 기간 종료일에 불입할 경우 0을 입력합니다. 기본값은 0입니다. 0 또는 1 이외의 값을 입력하면 PowerCenter 통합 서비스가 값을 1로 처리합니다.

### 반환 값

숫자.

### 예

다음 식은 매월 대출 불입 금액으로 -2111.64를 반환합니다.

`PMT( 0.01, 10, 20000 )`

### 참고

기간별 이율을 계산하려면 연이율을 연간 불입 횟수로 나눕니다. 예를 들어 연 15%의 이율로 매월 불입하는 경우 이율은 15%/12입니다. 연 불입의 경우 이율은 15%입니다.

불입 금액은 사용자가 지불하는 금액이므로 음수입니다.

## POWER

함수로 전달하는 지수로 올림한 값을 반환합니다.

### 구문

`POWER( base, exponent )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
기본	필수	숫자 값. 이 인수는 밑수 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다. 기본값이 음수인 경우 지수는 반드시 정수여야 합니다.
지수	필수	숫자 값. 이 인수는 지수 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다. 기본값이 음수인 경우 지수는 반드시 정수여야 합니다. 이러한 경우 가장 근사한 정수까지 모든 소수값을 반올림한 값이 반환됩니다.

## 반환 값

배정 밀도 값.

Null 값을 함수에 전달하는 경우 NULL이 반환됩니다.

## 예

다음 식은 Numbers 포트의 값을 Exponent 포트의 값으로 제공한 값을 반환합니다.

POWER( NUMBERS, EXPONENT )

NUMBERS	EXPONENT	RETURN VALUE
10.0	2.0	100
3.5	6.0	1838.265625
3.5	5.5	982.594307804838
NULL	2.0	NULL
10.0	NULL	NULL
-3.0	-6.0	0.00137174211248285
3.0	-6.0	0.00137174211248285
-3.0	6.0	729.0
-3.0	5.5	729.0

-3.0 값의 6 제곱과 -3.0의 5.5 제곱은 동일한 결과를 반환합니다. 밑수가 음수인 경우 지수는 정수여야 합니다. 그렇지 않은 경우 PowerCenter 통합 서비스가 지수를 가까운 정수 값으로 반올림합니다.

## PV

투자의 현재 가치를 반환합니다.

## 구문

PV( *rate*, *terms*, *payment* [, *future value*, *type*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
rate	필수	분수. 각 기간별 이율입니다. 10진수로 표시됩니다. 이율을 100으로 나눈 후 10진수로 표시합니다.
terms	필수	분수. 기간 개수 또는 불입 횟수입니다. 0보다 커야 합니다.
불입	필수	분수. 기간별 불입액입니다. 음수여야 합니다.
미래 가치	선택	분수. 마지막 불입 이후의 현금 잔액입니다. 이 값을 생략하면 PV에서 0을 사용합니다.
유형	선택	부울. 불입 시기. 기간 시작일에 불입할 경우 1을 입력합니다. 기간 종료일에 불입할 경우 0을 입력합니다. 기본값은 0입니다. 0 또는 1 이외의 값을 입력하면 PowerCenter 통합 서비스가 값을 1로 처리합니다.

## 반환 값

숫자.

## 예

다음 식은 각 기간이 시작될 때 \$500를 예금하는 경우 1년 후의 미래 가치를 \$20,000로 만들기 위해 현재 계좌에 예치해야 하는 금액으로 12,524.43을 반환합니다.

PV( 0.0075, 12, -500, 20000, TRUE )

## RAND

0~1의 난수를 반환합니다. 이 함수는 수익성 시나리오에 유용합니다.

## 구문

RAND( seed )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
시드	선택 사항	숫자. 통합 서비스에서 무작위 숫자를 생성할 때 사용하는 시작 값입니다. 값은 상수여야 합니다. 시드를 입력하지 않으면 PowerCenter 통합 서비스가 현재 시스템 시간을 사용하여 1971년 1월 1일 이후의 초 수를 파생한 다음 이 값을 시드로 사용합니다.

## 반환 값

숫자.

시드가 동일한 경우 PowerCenter 통합 서비스는 동일한 숫자 시퀀스를 생성합니다.

예

다음 식은 0.417022004702574 값을 반환할 수 있습니다.

RAND (1)

## RATE

채권의 기간별 이율을 반환합니다.

구문

RATE( *terms*, *payment*, *present value*[, *future value*, *type*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
용어	필수	숫자. 기간 개수 또는 불입 횟수입니다. 0보다 커야 합니다.
불입	필수	숫자. 기간별 불입액입니다. 음수여야 합니다.
현재 가치	필수	숫자. 미래 불입 금액을 현재 가치로 환산한 총액입니다.
미래 가치	선택 사항	숫자. 마지막 불입 이후에 달성하고자 하는 잔액입니다. 예를 들어 대출금의 미래 가치는 0입니다. 이 인수를 생략한 경우 RATE는 0을 사용합니다.
유형	선택 사항	부울. 불입 시기. 기간 시작일에 불입할 경우 1을 입력합니다. 기간 종료일에 불입할 경우 0을 입력합니다. 기본값은 0입니다. 0 또는 1 이외의 값을 입력하면 PowerCenter 통합 서비스가 값을 1로 처리합니다.

반환 값

숫자.

예

다음 식은 대출금의 월 이율로 0.0077을 반환합니다.

RATE( 48, -500, 20000 )

대출금의 연 이율을 계산하려면 0.0077에 12를 곱합니다. 연 이율은 0.0924 또는 9.24%입니다.

## REG\_EXTRACT

입력 값 내에서 정규식의 하위 패턴을 추출합니다. 예를 들어 전체 이름에 대한 정규식 패턴에서 이름 또는 성을 추출할 수 있습니다.

**참고:** 문자열의 문자 패턴을 다른 문자 패턴으로 바꾸려면 REG\_REPLACE 함수를 사용합니다.

구문

REG\_EXTRACT( *subject*, '*pattern*', *subPatternNum*, *match\_from\_start* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>subject</i>	필수	문자열 데이터 유형. 정규식 패턴과 비교할 값을 전달합니다.
<i>pattern</i>	필수	문자열 데이터 유형. 일치시킬 정규식 패턴입니다. perl 호환 정규식 구문을 사용해야 합니다. 패턴은 작은따옴표로 묶습니다. 각 하위 패턴은 괄호로 묶습니다.
<i>subPatternNum</i>	선택 사항	정수 값. 일치시킬 정규식의 하위 패턴 번호입니다. 하위 패턴 번호는 다음 지침에 따라 결정됩니다. <ul style="list-style-type: none"> <li>- 값 없음 또는 1. 첫 번째 정규식 하위 패턴이 추출됩니다.</li> <li>- 2. 두 번째 정규식 하위 패턴이 추출됩니다.</li> <li>- n. n번째 정규식 하위 패턴이 추출됩니다.</li> </ul> 기본값은 1입니다.
<i>match_from_start</i>	선택 사항	숫자 값. 문자열의 시작에서 일치가 발견된 경우 하위 문자열이 반환됩니다. 시작 값의 일치는 다음 지침에 따라 결정됩니다. <ul style="list-style-type: none"> <li>- 0. 패턴을 시작 인덱스 또는 모든 인덱스의 대상 문자열과 일치시킵니다.</li> <li>- 0이 아님. 패턴을 시작 인덱스의 대상 문자열과 일치시킵니다.</li> </ul>

#### perl 호환 정규식 구문 사용

REG\_EXTRACT, REG\_MATCH 및 REG\_REPLACE 함수에는 perl 호환 정규식 구문을 사용해야 합니다.

다음 테이블에는 perl 호환 정규식 구문 지침이 제공되어 있습니다.

구문	설명
.	(마침표) 문자 하나를 일치시킵니다.
[a-z]	문자의 인스턴스 하나를 소문자로 일치시킵니다. 예를 들어 [a-z]는 ab를 일치시킵니다. 문자를 대문자로 일치시키려면 [A-Z]를 사용합니다.
\d	모든 숫자(0-9)의 인스턴스 하나를 일치시킵니다.
\s	공백 문자를 일치시킵니다.
\w	밑줄(_)을 포함하여 영숫자 하나를 일치시킵니다.
()	식을 그룹화합니다. 예를 들어 (\d-\d-\d\d)에서 괄호는 12-34와 같이 뒤에 하이픈과 2자리 숫자가 오는 모든 2자리 숫자를 찾는 식인 \d-\d-\d\d를 그룹화합니다.
{}	문자 수를 일치시킵니다. 예를 들어 \d{3}는 650 또는 510과 같은 3자리 숫자를 일치시킵니다. 또는 [a-z]{2}는 CA 또는 NY와 같은 2자리 문자를 일치시킵니다.
?	앞에 있는 문자 또는 문자 그룹을 0회 또는 1회 일치시킵니다. 예를 들어 \d{3}(-{d{4}})?는 모든 3자리 숫자를 일치시킵니다. 이 숫자 뒤에는 하이픈과 4자리 숫자가 올 수 있습니다.

구문	설명
*(별표)	별표 뒤에 오는 값의 인스턴스(0 이상)를 일치시킵니다. 예를 들어 *0은 0 앞에 있는 모든 값입니다.
+	더하기 기호 뒤에 오는 값의 인스턴스(1 이상)를 일치시킵니다. 예를 들어 \w+는 영숫자 뒤에 오는 모든 값입니다.

예를 들어 다음 정규식은 5자리 미국 우편 번호(예: 93930)와 9자리 우편 번호(예: 93930-5407)를 찾습니다.

`\d{5}(-\d{4})?`

`\d{5}`는 93930과 같은 모든 5자리 숫자를 나타냅니다. `-\d{4}`를 감싸는 괄호는 식에서 이 세그먼트를 그룹화합니다. 하이픈은 93930-5407에서와 같이 9자리 우편 번호의 하이픈을 나타냅니다. `\d{4}`는 5407과 같은 4자리 숫자를 나타냅니다. 물음표는 하이픈과 마지막 4자리가 선택 사항이거나 한 번 나타날 수 있다는 것을 명시합니다.

#### COBOL 구문을 perl 호환 정규식 구문으로 변환

COBOL 구문에 익숙한 사용자는 다음 정보를 사용하여 perl 호환 정규식을 작성할 수 있습니다.

다음 테이블에는 COBOL 구문 및 해당하는 perl 구문에 대한 예가 표시되어 있습니다.

COBOL 구문	perl 구문	설명
9	<code>\d</code>	모든 숫자(0-9)의 인스턴스 하나를 일치시킵니다.
9999	<code>\d\d\d\d</code> 또는 <code>\d{4}</code>	0-9의 4자리를 일치시킵니다(예: 1234 또는 5936).
x	<code>[a-z]</code>	문자의 인스턴스 하나를 일치시킵니다.
9xx9	<code>\d[a-z][a-z]\d</code>	1ab2에서와 같이 2자리 문자와 다른 숫자가 뒤에 있는 모든 숫자를 일치시킵니다.

#### SQL 구문을 perl 호환 정규식 구문으로 변환

SQL 구문에 익숙한 사용자는 다음 정보를 사용하여 perl 호환 정규식을 작성할 수 있습니다.

다음 테이블에는 SQL 구문 및 해당하는 perl 구문에 대한 예가 표시되어 있습니다.

SQL 구문	perl 구문	설명
%	<code>.</code> *	모든 문자열을 일치시킵니다.
A%	<code>A.</code> *	Area와 같이 앞에 문자 "A"가 있는 모든 문자열을 일치시킵니다.
_	<code>.</code> (마침표)	문자 하나를 일치시킵니다.
A_	<code>A.</code>	AZ와 같이 "A"로 시작하는 모든 2자리 문자를 일치시킵니다.

## 반환 값

입력 값의 일부인  $n$ 번째 하위 패턴의 값이 반환됩니다.  $n$ 번째 하위 패턴은 `subPatternNum`에 지정한 값에 기반합니다.

입력이 Null 값이거나 패턴이 Null인 경우 NULL이 반환됩니다.

## 예

REG\_EXTRACT를 식에 사용하여 이름, 중간 이름 및 성을 일치시키는 정규식에서 중간 이름을 추출할 수 있습니다. 예를 들어 다음 식은 정규식의 중간 이름을 반환합니다.

```
REG_EXTRACT( Employee_Name, '(\w+)\s+(\w+)\s+(\w+)', 2)
```

Employee_Name	Return Value
Stephen Graham Smith	Graham
Juan Carlos Fernando	Carlos

## REG\_MATCH

값이 정규식 패턴과 일치하는지 여부를 반환합니다. 이 함수를 사용하면 ID, 전화 번호, 우편 번호 및 주 이름 등 데이터 패턴의 유효성을 검사할 수 있습니다.

**참고:** 문자열의 문자 패턴을 새 문자 패턴으로 바꾸려면 REG\_REPLACE 함수를 사용합니다.

## 구문

```
REG_MATCH( subject, pattern )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
제목	필수	문자열 데이터 유형. 정규식 패턴과 일치시킬 값을 전달합니다.
패턴	필수	문자열 데이터 유형. 일치시킬 정규식 패턴입니다. perl 호환 정규식 구문을 사용해야 합니다. 패턴은 작은따옴표로 묶습니다. 자세한 내용은 <a href="#">“REG_EXTRACT” 페이지 149</a> 을 참조하십시오.

## 반환 값

데이터가 패턴과 일치하는 경우 TRUE가 반환됩니다.

데이터가 패턴과 일치하지 않는 경우 FALSE가 반환됩니다.

입력이 Null 값이거나 패턴이 NULL인 경우 NULL이 반환됩니다.



## 예

REG\_MATCH를 식에 사용하여 전화 번호의 유효성을 검사할 수 있습니다. 예를 들어 다음 식은 10자리 전화 번호를 패턴에 일치시키고 일치에 따라 부울 값을 반환합니다.

```
REG_MATCH (Phone_Number, '(\d\d\d\d-\d\d\d\d\d\d)' )
```

Phone_Number	Return Value
408-555-1212	TRUE
	NULL
510-555-1212	TRUE
92 555 51212	FALSE
650-555-1212	TRUE
415-555-1212	TRUE
831 555 12123	FALSE

## 팁

REG\_MATCH를 다음 태스크에도 사용할 수 있습니다.

- 값이 패턴과 일치하는지 확인합니다. 이 사용은 SQL LIKE 함수와 유사합니다.
- 값이 문자인지 확인합니다. 이 사용은 SQL IS\_CHAR 함수와 유사합니다.

값이 패턴과 일치하는지 확인하려면 마침표(.)와 별표(\*)를 REG\_MATCH 함수와 함께 식에 사용합니다. 마침표는 모든 1자리 문자를 일치시킵니다. 별표는 별표 뒤에 오는 값의 0 이상의 인스턴스를 일치시킵니다.

예를 들어 다음 식을 사용하면 1835로 시작하는 계좌 번호를 찾을 수 있습니다.

```
REG_MATCH(ACCOUNT_NUMBER, '1835.*')
```

값이 문자인지 확인하려면 REG\_MATCH 함수를 정규식 [a-zA-Z]+와 함께 사용합니다. a-z는 모든 소문자 문자를 일치시킵니다. A-Z는 모든 대문자 문자를 일치시킵니다. 더하기 기호(+)는 문자 수가 1개 이상이어야 함을 나타냅니다.

예를 들어 다음 식을 사용하면 성 목록에 문자만 포함되는지를 확인할 수 있습니다.

```
REG_MATCH(LAST_NAME, '[a-zA-Z]+')
```

## REG\_REPLACE

문자열의 문자를 다른 문자 패턴으로 바꿉니다. 기본적으로 REG\_REPLACE는 입력 문자열을 검색하여 사용자가 지정한 문자를 찾은 다음 모든 일치 항목을 대체 패턴으로 바꿉니다. 문자열에서 바꾸려는 패턴의 발생 횟수를 지정할 수도 있습니다.

## 구문

```
REG_REPLACE( subject, pattern, replace, numReplacements )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>제목</i>	필수	문자열 데이터 유형. 검색할 문자열을 전달합니다.
<i>패턴</i>	필수	문자열 데이터 유형. 바꿀 문자열을 전달합니다. perl 호환 정규식 구문을 사용해야 합니다. 패턴은 작은따옴표로 묶습니다. 자세한 내용은 <a href="#">“REG_EXTRACT” 페이지 149</a> 을 참조하십시오.
<i>바꾸기</i>	필수	문자열 데이터 유형. 새 문자열을 전달합니다.
<i>numReplacement s</i>	선택 사항	숫자 데이터 유형. 바꿀 발생 횟수를 지정합니다. 이 옵션을 생략하면 REG_REPLACE가 문자열의 모든 일치 항목을 바꿉니다.

## 반환 값

문자열

예

다음 식은 Employee\_name 포트의 각 행에 대한 직원 이름 데이터에서 추가 공백을 제거합니다.

```
REG_REPLACE( Employee_Name, '\s+', ' ' )
```

Employee_Name	RETURN VALUE
Adam Smith	Adam Smith
Greg Sanders	Greg Sanders
Sarah Fe	Sarah Fe
Sam Cooper	Sam Cooper

## REPLACECHR

문자열의 문자를 한 문자나 공백으로 바꿉니다. REPLACECHR는 입력 문자열을 검색하여 사용자가 지정한 문자를 찾은 다음 모든 문자의 모든 일치 항목을 사용자가 지정한 새 문자로 바꿉니다.

구문

```
REPLACECHR( CaseFlag, InputString, OldCharSet, NewChar )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>CaseFlag</i>	필수	정수여야 합니다. 이 함수의 인수가 대/소문자를 구분하는지를 결정합니다. 유효한 모든 변환 식을 입력할 수 있습니다. <i>CaseFlag</i> 가 0이 아닌 숫자인 경우 이 함수는 대/소문자를 구분합니다. <i>CaseFlag</i> 가 Null 값 또는 0인 경우 이 함수는 대/소문자를 구분하지 않습니다.
<i>InputString</i>	필수	문자 문자열이어야 합니다. 검색할 문자열을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 숫자 값을 전달하는 경우 함수가 문자열로 변환합니다. <i>InputString</i> 이 NULL인 경우 REPLACECHR는 NULL을 반환합니다.
<i>OldCharSet</i>	필수	문자 문자열이어야 합니다. 바꿀 문자입니다. 하나 이상의 문자를 입력할 수 있습니다. 유효한 모든 변환 식을 입력할 수 있습니다. 텍스트 리터럴을 작은따옴표로 묶어 입력할 수도 있습니다(예: 'abc'). 숫자 값을 전달하는 경우 함수가 문자열로 변환합니다. <i>OldCharSet</i> 가 NULL이거나 비어 있는 경우 REPLACECHR는 <i>InputString</i> 을 반환합니다.
<i>NewChar</i>	필수	문자 문자열이어야 합니다. 한 문자, 빈 문자열 또는 NULL을 입력할 수 있습니다. 유효한 모든 변환 식을 입력할 수 있습니다. <i>NewChar</i> 이 NULL이거나 비어 있는 경우 REPLACECHR는 <i>OldCharSet</i> 의 모든 문자의 모든 일치 항목을 <i>InputString</i> 에서 제거합니다. <i>NewChar</i> 에 2개 이상의 문자가 포함되는 경우 REPLACECHR는 첫 번째 문자를 사용하여 <i>OldCharSet</i> 를 바꿉니다.

## 반환 값

문자열.

REPLACECHR가 *InputString*에서 모든 문자를 제거한 경우 빈 문자열이 반환됩니다.

*InputString*이 NULL인 경우 NULL이 반환됩니다.

*OldCharSet*가 NULL이거나 비어 있는 경우 *InputString*이 반환됩니다.

## 예

다음 식은 WEBLOG 포트의 각 행에 대한 웹 로그 데이터에서 큰따옴표를 제거합니다.

```
REPLACECHR( 0, WEBLOG, '"', NULL )
```

WEBLOG	RETURN VALUE
"GET /news/index.html HTTP/1.1"	GET /news/index.html HTTP/1.1
"GET /companyinfo/index.html HTTP/1.1"	GET /companyinfo/index.html HTTP/1.1
GET /companyinfo/index.html HTTP/1.1	GET /companyinfo/index.html HTTP/1.1
NULL	NULL

다음 식은 WEBLOG 포트의 각 행에 대한 여러 문자를 제거합니다.

REPLACECHR ( 1, WEBLOG, ']'["', NULL )

WEBLOG	RETURN VALUE
[29/Oct/2001:14:13:50 -0700]	29/Oct/2001:14:13:50 -0700
[31/Oct/2000:19:45:46 -0700] "GET /news/index.html HTTP/1.1"	31/Oct/2000:19:45:46 -0700 GET /news/index.html HTTP/1.1
[01/Nov/2000:10:51:31 -0700] "GET /news/index.html HTTP/1.1"	01/Nov/2000:10:51:31 -0700 GET /news/index.html HTTP/1.1
NULL	NULL

다음 식은 CUSTOMER\_CODE 포트의 각 행에 대한 고객 코드의 값 일부를 변경합니다.

REPLACECHR ( 1, CUSTOMER\_CODE, 'A', 'M' )

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	abM
BBC	BBC
ACC	MCC
NULL	NULL

다음 식은 CUSTOMER\_CODE 포트의 각 행에 대한 고객 코드의 값 일부를 변경합니다.

REPLACECHR ( 0, CUSTOMER\_CODE, 'A', 'M' )

CUSTOMER_CODE	RETURN VALUE
ABA	MBM
abA	MbM
BBC	BBC
ACC	MCC

다음 식은 CUSTOMER\_CODE 포트의 각 행에 대한 고객 코드의 값 일부를 변경합니다.

REPLACECHR ( 1, CUSTOMER\_CODE, 'A', NULL )

CUSTOMER_CODE	RETURN VALUE
ABA	B

CUSTOMER_CODE	RETURN VALUE
BBC	BBC
ACC	CC
AAA	<i>[empty string]</i>
aaa	aaa
NULL	NULL

다음 식은 INPUT 포트의 각 행에 대한 여러 숫자를 제거합니다.

```
REPLACECHR ( 1, INPUT, '14', NULL )
```

INPUT	RETURN VALUE
12345	235
4141	NULL
111115	5
NULL	NULL

작은따옴표(')를 *OldCharSet* 또는 *NewChar*에 사용하려는 경우 CHR 함수를 사용해야 합니다. 작은따옴표는 문자열 리터럴 안에 사용할 수 없는 유일한 문자입니다.

다음 식은 INPUT 포트의 각 행에 대한 여러 문자(작은따옴표 포함)를 제거합니다.

```
REPLACECHR (1, INPUT, CHR(39), NULL )
```

INPUT	RETURN VALUE
'Tom Smith' 'Laura Jones'	Tom Smith Laura Jones
Tom's	Toms
NULL	NULL

## REPLACESTR

문자열의 문자를 한 문자, 여러 문자 또는 공백으로 바꿉니다. REPLACESTR 함수는 입력 문자열을 검색하여 사용자가 지정한 모든 문자열을 찾은 다음 사용자가 지정한 새 문자열로 바꿉니다.

### 구문

```
REPLACESTR ( CaseFlag, InputString, OldString1, [OldString2, ... OldStringN,] NewString )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>CaseFlag</i>	필수	정수여야 합니다. 이 함수의 인수가 대/소문자를 구분하는지를 결정합니다. 유효한 모든 변환 식을 입력할 수 있습니다. <i>CaseFlag</i> 가 0이 아닌 숫자인 경우 이 함수는 대/소문자를 구분합니다. <i>CaseFlag</i> 가 Null 값 또는 0인 경우 이 함수는 대/소문자를 구분하지 않습니다.
<i>InputString</i>	필수	문자 문자열이어야 합니다. 검색할 열을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 숫자 값을 전달하는 경우 함수가 문자열로 변환합니다. <i>InputString</i> 이 NULL인 경우 REPLACESTR가 NULL을 반환합니다.
<i>OldString</i>	필수	문자 문자열이어야 합니다. 바꿀 문자열입니다. 하나 이상의 <i>OldString</i> 인수를 입력해야 합니다. <i>OldString</i> 인수당 하나 이상의 문자를 입력할 수 있습니다. 유효한 모든 변환 식을 입력할 수 있습니다. 텍스트 리터럴을 작은따옴표로 묶어 입력할 수도 있습니다(예: 'abc'). 숫자 값을 전달하는 경우 함수가 문자열로 변환합니다. REPLACESTR에 여러 <i>OldString</i> 인수가 포함될 때 하나 이상의 <i>OldString</i> 인수가 NULL이거나 비어 있는 경우 REPLACESTR가 해당 <i>OldString</i> 인수를 무시합니다. 모든 <i>OldString</i> 인수가 NULL이거나 비어 있는 경우 REPLACESTR는 <i>InputString</i> 을 반환합니다. 이 함수는 <i>OldString</i> 인수의 문자를 함수에 나타나는 순서로 바꿉니다. 예를 들어 여러 <i>OldString</i> 인수를 입력한 경우 첫 번째 <i>OldString</i> 인수는 두 번째 <i>OldString</i> 인수보다 우선하며 두 번째 <i>OldString</i> 인수는 세 번째 <i>OldString</i> 인수보다 우선합니다. REPLACESTR는 문자열을 바꿀 때 <i>InputString</i> 에서 바꾼 문자열 뒤에 커서를 배치한 후 다음 일치 검색합니다.
<i>NewString</i>	필수	문자 문자열이어야 합니다. 한 문자, 여러 문자, 빈 문자열 또는 NULL을 입력할 수 있습니다. 유효한 모든 변환 식을 입력할 수 있습니다. <i>NewString</i> 이 NULL이거나 비어 있는 경우 REPLACESTR는 <i>OldString</i> 의 모든 일치 항목을 <i>InputString</i> 에서 제거합니다.

## 반환 값

문자열.

REPLACESTR가 *InputString*에서 모든 문자를 제거한 경우 빈 문자열이 반환됩니다.

*InputString*이 NULL인 경우 NULL이 반환됩니다.

모든 *OldString*이 NULL이거나 비어 있는 경우 *InputString*이 반환됩니다.

## 예

다음 식은 WEBLOG 포트의 각 행에 대한 웹 로그 데이터에서 큰따옴표와 텍스트 문자열 2개를 제거합니다.

```
REPLACESTR( 1, WEBLOG, '"', 'GET ', ' HTTP/1.1', NULL )
```

WEBLOG	RETURN VALUE
"GET /news/index.html HTTP/1.1"	/news/index.html
"GET /companyinfo/index.html HTTP/1.1"	/companyinfo/index.html

WEBLOG	RETURN VALUE
GET /companyinfo/index.html	/companyinfo/index.html
GET	[empty string]
NULL	NULL

다음 식은 TITLE 포트의 각 행에 대한 특정 값의 제목을 변경합니다.

```
REPLACESTR ( 1, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
Miss	Ms.
Mr.	Mr.
MRS.	MRS.

다음 식은 TITLE 포트의 각 행에 대한 특정 값의 제목을 변경합니다.

```
REPLACESTR ( 0, TITLE, 'rs.', 'iss', 's.' )
```

TITLE	RETURN VALUE
Mrs.	Ms.
MRS.	Ms.

다음 식은 REPLACESTR 함수를 사용하여 INPUT 포트의 각 행에 대한 여러 OldString 인수를 바꾸는 방법을 보여 줍니다.

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', '*' )
```

INPUT	RETURN VALUE
abc	*c
abbc	**
abbbbc	*bb*
bc	*

다음 식은 REPLACESTR 함수를 사용하여 INPUT 포트의 각 행에 대한 여러 OldString 인수를 바꾸는 방법을 보여 줍니다.

```
REPLACESTR ( 1, INPUT, 'ab', 'bc', 'b' )
```

INPUT	RETURN VALUE
ab	b
bc	b
abc	bc
abbc	bb
abbcc	bbc

작은따옴표(')를 *OldString* 또는 *NewString*에 사용하려는 경우 CHR 함수를 사용해야 합니다. 작은따옴표를 문자열에 연결하려면 CHR 및 CONCAT 함수를 둘 다 사용합니다. 작은따옴표는 문자열 리터럴 안에 사용할 수 없는 유일한 문자입니다. 다음 예제를 고려하십시오.

```
CONCAT( 'Joan', CONCAT( CHR(39), 's car' ) )
```

반환 값은 다음과 같습니다.

```
Joan's car
```

다음 식은 INPUT 포트의 각 행에 대한 작은따옴표를 포함하는 문자열을 변경합니다.

```
REPLACESTR ( 1, INPUT, CONCAT('it', CONCAT(CHR(39), 's' )), 'its' )
```

INPUT	RETURN VALUE
it's	its
mit's	mits
mits	mits
mits'	mits'

## REVERSE

입력 문자열을 반전합니다.

구문

```
REVERSE( string )
```



다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
문자열	필수	모든 문자 값. 반전할 값입니다.

### 반환 값

문자열. 입력 값의 반전이 반환됩니다.

### 예

다음 식은 고객 코드의 숫자를 반전합니다.

```
REVERSE( CUSTOMER_CODE )
```

CUSTOMER_CODE	RETURN VALUE
0001	1000
0002	2000
0003	3000
0004	4000

## ROUND(날짜)

날짜의 한 부분을 반올림합니다. 숫자를 반올림할 때도 ROUND를 사용할 수 있습니다.

이 함수는 날짜의 다음 부분을 반올림합니다.

### 연도

날짜의 연도 부분을 월을 기준으로 반올림합니다.

### 월

날짜의 월 부분을 해당 월의 일을 기준으로 반올림합니다.

### 일

날짜의 일 부분을 시간을 기준으로 반올림합니다.

### 시간

날짜의 시간 부분을 해당 시간의 분을 기준으로 반올림합니다.

### 분

날짜의 분 부분을 초를 기준으로 반올림합니다.

### 초

날짜의 초 부분을 밀리초를 기준으로 반올림합니다.

## 밀리초

날짜의 밀리초 부분을 마이크로초를 기준으로 반올림합니다.

## 마이크로초

날짜의 마이크로초 부분을 나노초를 기준으로 반올림합니다.

다음 테이블에는 ROUND 식이 사용하는 조건과 반환 값이 표시되어 있습니다.

조건	식	반환 값
월이 1월부터 6월 사이인 경우 같은 연도의 1월 1일이 반환되고 시간은 00:00:00.000000000으로 설정됩니다.	ROUND(TO_DATE('04/16/1998 8:24:19', 'MM/DD/YYYY HH24:MI:SS'), 'YY')	01/01/1998 00:00:00.000000000
월이 7월부터 12월 사이인 경우 다음 연도의 1월 1일이 반환되고 시간은 00:00:00.000000000으로 설정됩니다.	ROUND(TO_DATE('07/30/1998 2:30:55', 'MM/DD/YYYY HH24:MI:SS'), 'YY')	01/01/1999 00:00:00.000000000
일이 1일부터 15일 사이인 경우 날짜로 입력 월의 첫째 날이 반환되고 시간은 00:00:00.000000000으로 설정됩니다.	ROUND(TO_DATE('04/15/1998 8:24:19', 'MM/DD/YYYY HH24:MI:SS'), 'MM')	04/01/1998 00:00:00.000000000
일이 16일부터 해당 월의 마지막 날 사이인 경우 다음 월의 첫째 날이 반환되고 시간은 00:00:00.000000000으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29', 'MM/DD/YYYY HH24:MI:SS'), 'MM')	06/01/1998 00:00:00.000000000
시간이 00:00:00(오전 12시)부터 오전 11:59:59 사이인 경우 현재 날짜가 반환되고 시간은 00:00:00.000000000(오전 12시)으로 설정됩니다.	ROUND(TO_DATE('06/13/1998 2:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/13/1998 00:00:00.000000000
시간이 12:00:00(오후 12시)이거나 그 이후인 경우 날짜가 다음 일로 반올림되고 시간은 00:00:00.000000000(오전 12시)으로 설정됩니다.	ROUND(TO_DATE('06/13/1998 22:30:45', 'MM/DD/YYYY HH24:MI:SS'), 'DD')	06/14/1998 00:00:00.000000000
시간의 분 부분이 0부터 29분 사이인 경우 현재 시간이 반환되고 분, 초, 밀리초 및 나노초가 0으로 설정됩니다.	ROUND(TO_DATE('04/01/1998 11:29:35', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 11:00:00.000000000
시간의 분 부분이 30 이상인 경우 다음 시간이 반환되고 분, 초, 밀리초, 나노초가 0으로 설정됩니다.	ROUND(TO_DATE('04/01/1998 13:39:00', 'MM/DD/YYYY HH24:MI:SS'), 'HH')	04/01/1998 14:00:00.000000000
시간이 0부터 29초 사이인 경우 현재 분이 반환되고 초, 밀리초 및 나노초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:15:00.000000000
시간이 30부터 59초 사이인 경우 다음 분이 반환되고 초, 밀리초 및 나노초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:30', 'MM/DD/YYYY HH24:MI:SS'), 'MI')	05/22/1998 10:16:00.000000000

조건	식	반환 값
시간이 0부터 499밀리초 사이인 경우 현재 초가 반환되고 밀리초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29.499','MM/DD/YYYY HH24:MI:SS.MS'),'SS')	05/22/1998 10:15:29.000000000
시간이 500부터 999밀리초 사이인 경우 다음 초가 반환되고 밀리초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29.500','MM/DD/YYYY HH24:MI:SS.MS'),'SS')	05/22/1998 10:15:30.000000000
시간이 0부터 499마이크로초 사이인 경우 현재 밀리초가 반환되고 마이크로초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29.498125','MM/DD/YYYY HH24:MI:SS.US'),'MS')	05/22/1998 10:15:29.498000000
시간이 500부터 999마이크로초 사이인 경우 다음 밀리초가 반환되고 마이크로초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29.498785','MM/DD/YYYY HH24:MI:SS.US'),'MS')	05/22/1998 10:15:29.499000000
시간이 0부터 499나노초 사이인 경우 현재 마이크로초가 반환되고 나노초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29.498125345','MM/DD/YYYY HH24:MI:SS.NS'),'US')	05/22/1998 10:15:29.498125000
시간이 500부터 999나노초 사이인 경우 다음 마이크로초가 반환되고 나노초가 0으로 설정됩니다.	ROUND(TO_DATE('05/22/1998 10:15:29.498125876','MM/DD/YYYY HH24:MI:SS.NS'),'US')	05/22/1998 10:15:29.498126000

## 구문

ROUND( *date* [,*format*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>date</i>	필수	날짜/시간 데이터 유형. TO_DATE를 중첩하여 반올림 전에 문자열을 날짜로 변환할 수 있습니다.
<i>format</i>	선택 사항	유효한 형식 문자열을 입력합니다. 날짜에서 반올림할 부분입니다. 날짜의 한 부분만 반올림할 수 있습니다. 형식 문자열을 생략하는 경우 날짜가 가까운 날로 반올림됩니다.

## 반환 값

지정한 부분이 반올림된 날짜가 반환됩니다. ROUND는 소스 날짜와 동일한 형식으로 날짜를 반환합니다. 이 함수의 결과를 날짜/시간 데이터 유형의 모든 포트에 연결할 수 있습니다.

Null 값을 함수에 전달하는 경우 NULL이 반환됩니다.

## 예

다음 식은 DATE\_SHIPPED 포트의 날짜에서 연도 부분을 반올림합니다.

```
ROUND( DATE_SHIPPED, 'Y' )
ROUND( DATE_SHIPPED, 'YY' )
```

```
ROUND( DATE_SHIPPED, 'YYY' )
ROUND( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Jan 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 월 부분을 반환합니다.

```
ROUND( DATE_SHIPPED, 'MM' )
ROUND( DATE_SHIPPED, 'MON' )
ROUND( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	May 1 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 일 부분을 반환합니다.

```
ROUND( DATE_SHIPPED, 'D' )
ROUND( DATE_SHIPPED, 'DD' )
ROUND( DATE_SHIPPED, 'DDD' )
ROUND( DATE_SHIPPED, 'DY' )
ROUND( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 12:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 20 1998 12:00:00.000000000AM
Dec 20 1998 3:29:55PM	Dec 21 1998 12:00:00.000000000AM
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 시간 부분을 반올림합니다.

```
ROUND( DATE_SHIPPED, 'HH' )  
ROUND( DATE_SHIPPED, 'HH12' )  
ROUND( DATE_SHIPPED, 'HH24' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:31AM	Jan 15 1998 2:00:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 2:00:00.000000000PM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:00:00.000000000PM
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 분 부분을 반올림합니다.

```
ROUND( DATE_SHIPPED, 'MI' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 2:11:00.000000000AM
Apr 19 1998 1:31:20PM	Apr 19 1998 1:31:00.000000000PM
Dec 20 1998 3:29:55PM	Dec 20 1998 3:30:00.000000000PM
Dec 31 1998 11:59:59PM	Jan 1 1999 12:00:00.000000000AM
NULL	NULL

## ROUND(숫자)

숫자를 지정된 자릿수 또는 소수 자릿수로 반올림합니다. 날짜를 반올림할 때도 ROUND를 사용할 수 있습니다.

### 구문

```
ROUND( numeric_value [, precision] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 유효한 모든 변환 식을 입력할 수 있습니다. 값을 반올림하기 전에 산술 계산을 수행하려면 연산자를 사용합니다.
<b>정밀도</b>	선택 사항	<p>양 또는 음의 정수. <b>정밀도</b>에 양수를 입력하면 이 숫자의 소수 자릿수로 반올림됩니다. 예를 들어 ROUND(12.99, 1)은 13.0을 반환하고 ROUND(15.44, 1)은 15.4를 반환합니다.</p> <p><b>정밀도</b>에 음수를 입력하면 소수점을 기준으로 왼쪽으로 이 숫자의 자릿수가 반올림되고 정수가 반환됩니다. 예를 들어 ROUND(12.99, -1)은 10을 반환하고 ROUND(15.99, -1)은 20을 반환합니다.</p> <p><b>정밀도</b>에 소수를 입력하면 가까운 정수로 반올림한 후 식을 평가합니다. 예를 들어 ROUND(12.99, 0.8)은 0.8을 1로 반올림한 다음 식을 평가하므로 13.0이 반환됩니다.</p> <p><b>정밀도</b> 인수를 생략하면 소수부를 잘라내고 함수가 가까운 정수로 반올림됩니다. 예를 들어 ROUND(12.99)는 13을 반환합니다.</p>

## 반환 값

숫자 값.

인수 중 하나가 NULL인 경우 NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## 예

다음 식은 Price 포트의 값을 소수점 세 자리까지 반올림하여 반환합니다.

```
ROUND( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.956
NULL	NULL

**정밀도** 인수에 음의 정수를 전달하면 소수점을 기준으로 왼쪽으로 자릿수를 반올림할 수 있습니다.

```
ROUND( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13200.0

PRICE	RETURN VALUE
1435.99	1400.0
-108.95	-100.0
NULL	NULL

정밀도 인수에 소수 값을 전달하면 PowerCenter 통합 서비스가 가까운 정수로 반올림한 다음 식을 평가합니다.

`ROUND( PRICE, 0.8 )`

PRICE	RETURN VALUE
12.99	13.0
56.34	56.3
NULL	NULL

정밀도 인수를 생략하면 가까운 정수로 반올림됩니다.

`ROUND( PRICE )`

PRICE	RETURN VALUE
12.99	13.0
-15.99	-16.0
-18.99	-19.0
56.95	57.0
NULL	NULL

## 팁

계산된 값의 전체 자릿수를 명시적으로 설정하여 예상한 결과를 얻으려는 경우에도 **ROUND**를 사용할 수 있습니다. PowerCenter 통합 서비스를 작은 전체 자릿수 모드에서 실행할 때 값의 전체 자릿수가 15자리를 초과하면 통합 서비스가 계산의 결과를 잘라냅니다. 예를 들어 작은 전체 자릿수 모드에서 다음 식을 처리하려고 합니다.

`7/3 * 3 = 7`

이 경우 PowerCenter 통합 서비스가 첫 번째 나누기 연산의 결과를 잘라내므로 식의 왼쪽이 6.999999999999999로 평가됩니다. PowerCenter 통합 서비스는 이 전체 식을 **FALSE**로 평가합니다. 사용자는 이러한 결과를 원하지 않을 수 있습니다.

예상한 결과를 얻으려면 **ROUND**를 사용하여 식의 왼쪽에서 잘린 결과를 예상한 결과로 반올림해야 합니다. PowerCenter 통합 서비스는 다음 식을 **TRUE**로 평가합니다.

`ROUND(7/3 * 3) = 7`

## RPAD

문자열 끝에 공백이나 문자를 추가하여 문자열을 지정된 길이로 변환합니다.

### 구문

`RPAD( first_string, length [,second_string] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>first_string</i>	필수	모든 문자열 값입니다. 변경할 문자열입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>길이</i>	필수	양의 정수 리터럴이어야 합니다. 각 문자열의 원하는 길이를 지정합니다.
<i>second_string</i>	선택 사항	모든 문자열 값입니다. <i>first_string</i> 값의 오른쪽에 추가할 문자열을 전달합니다. 문자열의 끝에 추가할 문자는 작은따옴표로 묶어야 합니다(예: 'abc'). 이 인수는 대/소문자를 구분합니다. 두 번째 문자열을 생략할 경우 함수가 첫 번째 문자열의 끝에 공백을 추가합니다.

### 반환 값

지정된 길이의 문자열.

함수에 전달된 값이 **NULL**이거나 길이가 음수인 경우 **NULL**이 반환됩니다.

### 예

다음 식은 길이가 16자인 각 항목 이름의 끝에 문자열 '.'를 추가하여 항목 이름을 반환합니다.

`RPAD( ITEM_NAME, 16, '.')`

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight.....
Compass	Compass.....
Regulator System	Regulator System
Safety Knife	Safety Knife....

RPAD는 왼쪽에서 오른쪽으로 길이를 계산합니다. 따라서 첫 번째 문자열이 이 길이보다 길면 RPAD가 오른쪽에서 왼쪽으로 문자열을 잘라냅니다. 예를 들어 `RPAD('alphabetical', 5, 'x')`는 문자열 'alpha'를 반환합니다. RPAD는 필요한 경우 *second\_string*의 일부를 사용합니다.



다음 식은 길이가 16자인 각 항목 이름의 끝에 문자열 ‘\*..\*’를 추가하여 항목 이름을 반환합니다.

```
RPAD( ITEM_NAME, 16, '*..*' )
```

ITEM_NAME	RETURN VALUE
Flashlight	Flashlight*..**.
Compass	Compass*..**..**
Regulator System	Regulator System
Safety Knife	Safety Knife*..*

## RTRIM

문자열 끝에서 공백이나 문자를 제거합니다.

식에서 *trim\_set* 매개 변수를 지정하지 않은 경우는 다음과 같습니다.

- 유니코드 모드에서 RTRIM은 문자열의 끝에서 싱글바이트 및 더블바이트 공백을 제거합니다.
- ASCII 모드에서 RTRIM은 싱글바이트 공백만 제거합니다.

RTRIM을 사용하여 문자열에서 문자를 제거하는 경우 RTRIM은 *trim\_set*과 문자열 인수의 각 문자를 문자열의 오른쪽부터 한 문자씩 비교합니다. 문자열에서 *trim\_set*의 문자와 일치하는 문자는 제거됩니다.

RTRIM은 *trim\_set*에 일치하는 문자가 없을 때까지 계속해서 문자를 비교하고 제거합니다. 그런 다음 일치하는 문자가 없는 문자열을 반환합니다.

### 구문

```
RTRIM( string [, trim_set] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	모든 문자열 값입니다. 잘라내려는 모든 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 문자열 끝에서 공백을 제거하기 전에 비교를 수행하거나 문자열을 연결하려면 연산자를 사용하십시오.
<i>trim_set</i>	선택 사항	모든 문자열 값입니다. 문자열의 끝에서 제거할 문자를 전달합니다. 텍스트 리터럴을 입력할 수도 있습니다. 그러나 문자열의 끝에서 제거할 문자를 작은따옴표로 묶어야 합니다(예: 'abc'). 두 번째 문자열을 생략할 경우 함수가 첫 번째 문자열의 끝에서 공백을 제거합니다. RTRIM은 대/소문자를 구분합니다.

### 반환 값

문자열. 지정한 문자를 포함하는 문자열 값이 *trim\_set* 인수에서 제거됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 LAST\_NAME 포트의 문자열에서 문자 're'를 제거합니다.

```
RTRIM( LAST_NAME, 're')
```

LAST_NAME	RETURN VALUE
Nelson	Nelson
Page	Pag
Osborne	Osborn
NULL	NULL
Sawyer	Sawy
H. Bender	H. Bend
Steadman	Steadman

여기에서 *trim\_set*의 첫 번째 문자가 'r'이지만 Page의 'e'가 제거됩니다. RTRIM은 사용자가 *trim\_set* 인수에 지정한 문자 집합을 한 글자씩 검색하기 때문입니다. 문자열의 마지막 문자가 *trim\_set*의 첫 번째 문자와 일치할 경우 RTRIM은 이 문자를 제거합니다. 그러나 문자열의 마지막 문자가 일치하지 않을 경우 RTRIM은 *trim\_set*의 두 번째 문자와 비교합니다. 문자열의 마지막 문자의 두 번째가 *trim\_set*의 두 번째 문자와 일치할 경우 RTRIM은 이 문자를 제거합니다. 문자열의 문자가 *trim\_set*와 일치하지 않는 경우 RTRIM은 문자열을 반환하고 다음 행을 평가합니다.

마지막 예에서 Nelson의 마지막 문자는 *trim\_set* 인수의 어떤 문자와도 일치하지 않으므로 RTRIM은 문자열 'Nelson'을 반환하고 다음 행을 평가합니다.

## RTRIM 함수에 대한 팁

RTRIM 및 LTRIM 함수를 || 또는 CONCAT와 함께 사용하면 두 문자열을 연결한 후에 선행 및 후행 공백이 제거됩니다.

RTRIM을 중첩하여 여러 문자 집합을 제거할 수도 있습니다. 예를 들어 이름 열의 각 문자열의 끝에서 후행 공백과 문자 't'를 제거하려는 경우 다음과 유사한 식을 작성할 수 있습니다.

```
RTRIM( RTRIM( NAMES ), 't' )
```

## SETCOUNTVARIABLE

함수가 평가한 행 수를 계산하고 계산된 수에 따라 매핑 변수의 현재 값을 증가시킵니다. 삽입하도록 표시된 각 행에 대해 현재 값이 1씩 증가됩니다. 삭제하도록 표시된 각 행에 대해 현재 값이 1씩 감소됩니다. 업데이트 또는 거부하도록 표시된 각 행에 대해 현재 값이 동일하게 유지됩니다. 새 현재 값이 반환됩니다.

세션이 성공적으로 완료되면 PowerCenter 통합 서비스가 마지막 현재 값을 리포지토리에 저장합니다. 여러 파티션을 포함하는 세션을 사용하는 경우 PowerCenter 통합 서비스가 각 파티션에 대한 여러 현재 값을 생성합니다. 그런 다음 세션이 종료될 때 모든 파티션에 대한 합계를 결정하고 리포지토리에 저장합니다. 재정의하지 않은 경우 다음에 이 세션을 사용하면 저장된 값이 변수의 초기 값으로 사용됩니다.

SETCOUNTVARIABLE 함수는 파이프라인의 각 매핑 변수에 대해 한 번씩만 사용해야 합니다. PowerCenter 통합 서비스는 변수 함수를 매핑에서 발견되는 대로 처리합니다. PowerCenter 통합 서비스가 매핑에서 변수 함수를 발견하는 순서는 세션 실행에 따라 다를 수 있습니다. 따라서 매핑에서 동일한 변수 함수를 여러 번 사용할 경우 일관되지 않은 결과가 생성될 수 있습니다.

SETCOUNTVARIABLE은 카운트 집계 유형의 매핑 변수에 사용됩니다. SETCOUNTVARIABLE은 다음 변환에 사용됩니다.

- 식
- 필터
- 라우터
- 업데이트 전략

다음 중 하나가 참일 경우 PowerCenter 통합 서비스가 매핑 변수의 최종 값을 리포지토리에 저장하지 않습니다.

- 세션 완료가 실패했습니다.
- 세션이 테스트 로드를 수행하도록 구성되었습니다.
- 세션이 디버그 세션입니다.
- 세션이 디버그 모드에서 실행되고 세션 출력을 무시하도록 구성되었습니다.

## 구문

SETCOUNTVARIABLE( \$\$Variable )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
\$\$변수	필수	설정할 매핑 변수의 이름입니다. 카운트 집계 유형의 매핑 변수를 사용합니다.

## 반환 값

변수의 현재 값이 반환됩니다.

## 예

유통업체 정보를 포함하고 느리게 변경되는 차원 테이블을 업데이트하는 매핑이 있습니다. 다음 식은 매핑 변수 \$\$CurrentDistributors를 사용하여 현재 유통업체의 수를 계산하고 현재 값을 CUR\_DIST 포트에 반환합니다. 각 행이 삽입되면 수를 1씩 늘리고 각 행이 삭제되면 수를 1씩 줄이며 업데이트되거나 거부된 모든 행에 대해서는 수를 동일하게 유지합니다. 이전 세션 실행의 \$\$CurrentDistributors 초기 값은 23입니다.

SETCOUNTVARIABLE ( \$\$CurrentDistributors)

(row marked for...)	DIST_ID	DISTRIBUTOR	CUR_DIST
(update)	000015	MSD Inc.	23
(insert)	000024	Darkroom Co.	24
(insert)	000025	Howard's Supply	25

(row marked for...)	DIST_ID	DISTRIBUTOR	CUR_DIST
(update)	000003	JNR Ltd.	25
(delete)	000024	Darkroom Co.	24
(insert)	000026	Supply.com	25

세션이 종료되면 PowerCenter 통합 서비스가 '25'를 \$\$CurrentDistributors의 현재 값으로 리포지토리에 저장합니다. 다음에 세션을 실행하면 통합 서비스가 \$\$CurrentDistributors의 초기 값을 '25'로 평가합니다.

PowerCenter 통합 서비스는 여러 파티션을 포함하는 세션에 대해 단일 파티션을 포함하는 세션과 동일한 \$\$CurrentDistributors 값을 리포지토리에 저장합니다.

## SET\_DATE\_PART

날짜/시간 값의 일부를 지정하는 값으로 설정합니다. SET\_DATE\_PART를 사용하여 날짜의 다음 부분을 변경할 수 있습니다.

- **연도.** 양의 정수를 값 인수에 입력하여 연도를 변경합니다. 다음 연도 형식 문자열 중 하나를 사용합니다. 연도를 설정하려면 Y, YY, YYYY 또는 YYYYY를 사용합니다. 예를 들어 다음 식은 SHIP\_DATE 포트에 있는 모든 날짜의 연도를 2001로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'YY', 2001 )
```

- **월.** 1과 12 사이의 양의 정수(1월=1 및 12월=12)를 값 인수에 입력하여 월을 변경합니다. 다음 월 형식 문자열 중 하나를 사용합니다. 월을 설정하려면 MM, MON, MONTH를 사용합니다. 예를 들어 다음 식은 SHIP\_DATE 포트에 있는 모든 날짜의 월을 10월로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'MONTH', 10 )
```

- **일.** 1과 31 사이의 양의 정수(일 수가 31일보다 적은 월인 2, 4, 6, 9 및 11월은 예외)를 값 인수에 입력하여 일을 변경합니다. 일을 설정하려면 월 형식 문자열(D, DD, DDD, DY 및 DAY)을 사용합니다. 예를 들어 다음 식은 SHIP\_DATE 포트에 있는 모든 날짜의 일을 10으로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'DD', 10 )
```

- **시간.** 0과 24 사이의 양의 정수(여기서 0=12AM, 12=12PM 및 24=12AM)를 값 인수에 입력하여 시간을 변경합니다. 시간을 설정하려면 시간 형식 문자열(HH, HH12, HH24)을 사용합니다. 예를 들어 다음 식은 SHIP\_DATE 포트에 있는 모든 날짜의 시간을 14:00:00(또는 2:00:00PM)으로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'HH', 14 )
```

- **분.** 0과 59 사이의 양의 정수를 값 인수에 입력하여 분을 변경합니다. MI 형식 문자열을 사용하여 분을 설정합니다. 예를 들어 다음 식은 SHIP\_DATE 포트에 있는 모든 날짜의 분을 25로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'MI', 25 )
```

- **초.** 0과 59 사이의 양의 정수를 값 인수에 입력하여 초를 변경합니다. SS 형식 문자열을 사용하여 초를 설정합니다. 예를 들어 다음 식은 SHIP\_DATE 포트에 있는 모든 날짜의 초를 59로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'SS', 59 )
```

- **밀리초.** 0과 999 사이의 양의 정수를 *값* 인수에 입력하여 밀리초를 변경합니다. **MS** 형식 문자열을 사용하여 밀리초를 설정합니다. 예를 들어 다음 식은 **SHIP\_DATE** 포트에 있는 모든 날짜의 밀리초를 125로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'MS', 125 )
```

- **마이크로초.** 1000과 999999 사이의 양의 정수를 *값* 인수에 입력하여 마이크로초를 변경합니다. **US** 형식 문자열을 사용하여 밀리초를 설정합니다. 예를 들어 다음 식은 **SHIP\_DATE** 포트에 있는 모든 날짜의 마이크로초를 12555로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'US', 12555 )
```

- **나노초.** 1000000과 999999999 사이의 양의 정수를 *값* 인수에 입력하여 나노초를 변경합니다. **NS** 형식 문자열을 사용하여 나노초를 설정합니다. 예를 들어 다음 식은 **SHIP\_DATE** 포트에 있는 모든 날짜의 나노초를 12555555로 변경합니다.

```
SET_DATE_PART( SHIP_DATE, 'NS', 12555555 )
```

## 구문

```
SET_DATE_PART( date, format, value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>날짜</i>	필수	날짜/시간 데이터 유형. 수정할 날짜입니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>형식</i>	필수	변경할 날짜 부분을 지정하는 형식 문자열입니다. 형식 문자열은 대/소문자를 구분하지 않습니다.
<i>값</i>	필수	지정된 날짜 부분에 할당된 양의 정수 값입니다. 이 정수는 변경할 날짜의 부분에 유효한 값이어야 합니다. February 30과 같이 부적절한 값을 입력하면 세션이 실패합니다.

## 반환 값

지정된 부분이 변경된 날짜가 소스 날짜와 동일한 형식으로 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 **DATE\_PROMISED** 포트에 있는 각 날짜의 시간을 4PM으로 변경합니다.

```
SET_DATE_PART( DATE_PROMISED, 'HH', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH12', 16 )
SET_DATE_PART( DATE_PROMISED, 'HH24', 16 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jan 1 1997 4:15:56PM
Feb 13 1997 2:30:01AM	Feb 13 1997 4:30:01PM
Mar 31 1997 5:10:15PM	Mar 31 1997 4:10:15PM

DATE_PROMISED	RETURN VALUE
Dec 12 1997 8:07:33AM	Dec 12 1997 4:07:33PM
NULL	NULL

다음 식은 DATE\_PROMISED 포트에 있는 날짜의 월을 6월로 변경합니다. 3월 31일을 6월 31일로 변경하는 등 존재하지 않는 날짜를 작성하려고 하면 PowerCenter 통합 서비스가 오류를 표시합니다.

```
SET_DATE_PART( DATE_PROMISED, 'MM', 6 )
SET_DATE_PART( DATE_PROMISED, 'MON', 6 )
SET_DATE_PART( DATE_PROMISED, 'MONTH', 6 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jun 1 1997 12:15:56AM
Feb 13 1997 2:30:01AM	Jun 13 1997 2:30:01AM
Mar 31 1997 5:10:15PM	<i>Error. Integration Service doesn't write row.</i>
Dec 12 1997 8:07:33AM	Jun 12 1997 8:07:33AM
NULL	NULL

다음 식은 DATE\_PROMISED 포트에 있는 날짜의 연도를 2000으로 변경합니다.

```
SET_DATE_PART( DATE_PROMISED, 'Y', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YY', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YYY', 2000 )
SET_DATE_PART( DATE_PROMISED, 'YYYY', 2000 )
```

DATE_PROMISED	RETURN VALUE
Jan 1 1997 12:15:56AM	Jan 1 2000 12:15:56AM
Feb 13 1997 2:30:01AM	Feb 13 2000 2:30:01AM
Mar 31 1997 5:10:15PM	Mar 31 2000 5:10:15PM
Dec 12 1997 8:07:33AM	Dec 12 2000 4:07:33PM
NULL	NULL

## 팁

날짜의 여러 부분을 한 번에 변경하려는 경우 날짜 인수 안에 여러 SET\_DATE\_PART 함수를 중첩할 수 있습니다. 예를 들어 DATE\_ENTERED 포트의 모든 날짜를 1998년 7월 1일로 변경하려는 경우 다음과 같은 식을 작성할 수 있습니다.

```
SET_DATE_PART( SET_DATE_PART( SET_DATE_PART( DATE_ENTERED, 'YYYY', 1998), 'MM', 7), 'DD', 1)
```

## SETMAXVARIABLE

매핑 변수의 현재 값을 두 값, 즉 변수의 현재 값 또는 사용자가 지정한 값 중 더 큰 값으로 설정합니다. 새 현재 값이 반환됩니다. 이 함수는 삽입으로 표시된 행에만 실행됩니다. 다른 모든 행 유형은 무시되며 현재 값은 변경되지 않은 상태로 유지됩니다.

세션이 성공적으로 완료되면 PowerCenter 통합 서비스가 최종 현재 값을 리포지토리에 저장합니다. 여러 파티션을 포함하는 세션을 사용하는 경우 PowerCenter 통합 서비스가 각 파티션에 대한 여러 현재 값을 생성합니다. 그런 다음 세션이 종료될 때 모든 파티션에서 가장 큰 현재 값을 리포지토리에 저장합니다. 재정의하지 않은 경우 다음에 이 세션을 실행하면 저장된 값이 변수의 초기 값으로 사용됩니다.

SETMAXVARIABLE을 문자열 매핑 변수와 함께 사용하면 세션에서 선택한 정렬 순서를 기준으로 더 큰 문자열이 반환됩니다.

SETMAXVARIABLE 함수는 파이프라인의 각 매핑 변수에 대해 한 번씩만 사용해야 합니다. PowerCenter 통합 서비스는 변수 함수를 매핑에서 발견되는 대로 처리합니다. PowerCenter 통합 서비스가 매핑에서 변수 함수를 발견하는 순서는 세션 실행에 따라 다를 수 있습니다. 따라서 매핑에서 동일한 변수 함수를 여러 번 사용할 경우 일관되지 않은 결과가 생성될 수 있습니다.

SETMAXVARIABLE은 최대 집계 유형의 매핑 변수에 사용됩니다. SETMAXVARIABLE은 다음 변환에 사용됩니다.

- 식
- 필터
- 라우터
- 업데이트 전략

다음 조건 중 하나가 참일 경우 PowerCenter 통합 서비스가 매핑 변수의 최종 값을 리포지토리에 저장하지 않습니다.

- 세션 완료가 실패했습니다.
- 세션이 테스트 로드를 수행하도록 구성되었습니다.
- 세션이 디버그 세션입니다.
- 세션이 디버그 모드에서 실행되고 세션 출력을 무시하도록 구성되었습니다.

### 구문

SETMAXVARIABLE( \$\$Variable, value )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
\$\$변수	필수	설정할 매핑 변수의 이름입니다. 최대 집계 유형의 매핑 변수를 사용합니다.
값	필수	PowerCenter 통합 서비스에서 변수의 현재 값과 비교할 값입니다. 변수의 데이터 유형과 호환되는 데이터 유형으로 평가되는 모든 유효한 변환 식을 입력할 수 있습니다.

## 반환 값

다음 두 값 중에서 더 큰 값이 반환됩니다. 변수의 현재 값 또는 사용자가 지정한 값. 반환 값은 변수의 새 현재 값입니다.

값이 NULL인 경우 PowerCenter 통합 서비스가 *\$\$Variable*의 현재 값을 반환합니다.

## 예

다음 식은 각 트랜잭션에서 구매한 항목의 수를 매핑 변수 *\$\$MaxItems*와 비교합니다. 이 식은 *\$\$MaxItems*를 두 값 중 더 큰 값으로 설정하고 지금까지 단일 트랜잭션에서 구매한 항목 수 중 가장 큰 수를 MAX\_ITEMS 포트에 반환합니다. 이전 세션 실행의 *\$\$MaxItems* 초기 값은 22입니다.

SETMAXVARIABLE (\$\$MAXITEMS, ITEMS)

TRANSACTION	ITEMS	MAX_ITEMS
0100002	12	22
0100003	5	22
0100004	18	22
0100005	35	35
0100006	5	35
0100007	14	35

세션이 종료되면 PowerCenter 통합 서비스가 '35'를 *\$\$MaxItems*의 최대 현재 값으로 리포지토리에 저장합니다. 다음에 세션을 실행하면 PowerCenter 통합 서비스가 *\$\$MaxItems*에 대한 초기 값을 '35'로 평가합니다.

3개의 파티션을 포함하는 동일한 세션이 있는 경우 PowerCenter 통합 서비스는 *\$\$MaxItems*를 각 파티션에 대해 평가합니다. 그런 다음 가장 큰 값을 리포지토리에 저장합니다. 예를 들어 각 파티션에서 마지막으로 평가된 *\$\$MaxItems* 값은 다음과 같습니다.

Partition	Final Current Value for \$\$MaxItems
Partition 1	35
Partition 2	23
Partition 3	22

## SETMINVARIABLE

매핑 변수의 현재 값이 두 값, 즉 변수의 현재 값 또는 사용자가 지정한 값 중 더 작은 값으로 설정됩니다. 새 현재 값이 반환됩니다. SETMINVARIABLE 함수는 삽입으로 표시된 행에만 실행됩니다. 다른 모든 행 유형은 무시되며 현재 값은 변경되지 않은 상태로 유지됩니다.



세션이 성공적으로 완료되면 **PowerCenter** 통합 서비스가 최종 현재 값을 리포지토리에 저장합니다. 여러 파티션을 포함하는 세션을 사용하는 경우 **PowerCenter** 통합 서비스가 각 파티션에 대한 여러 현재 값을 생성합니다. 그런 다음 세션이 종료될 때 모든 파티션에서 가장 작은 현재 값을 리포지토리에 저장합니다. 재정의하지 않은 경우 다음에 이 세션을 실행하면 저장된 값이 변수의 초기 값으로 사용됩니다.

**SETMINVARIABLE**을 문자열 매핑 변수와 함께 사용하면 세션에서 선택한 정렬 순서를 기준으로 더 작은 문자열이 반환됩니다.

**SETMINVARIABLE** 함수는 파이프라인의 각 매핑 변수에 대해 한 번씩만 사용해야 합니다. **PowerCenter** 통합 서비스는 변수 함수를 매핑에서 발견되는 대로 처리합니다. **PowerCenter** 통합 서비스가 매핑에서 변수 함수를 발견하는 순서는 세션 실행에 따라 다를 수 있습니다. 따라서 매핑에서 동일한 변수 함수를 여러 번 사용할 경우 일관되지 않은 결과가 생성될 수 있습니다.

**SETMINVARIABLE**은 최소 집계 유형의 매핑 변수에 사용됩니다. **SETMINVARIABLE**은 다음 변환에 사용됩니다.

- 식
- 필터
- 라우터
- 업데이트 전략

다음 조건 중 하나가 참일 경우 **PowerCenter** 통합 서비스가 매핑 변수의 최종 값을 리포지토리에 저장하지 않습니다.

- 세션 완료가 실패했습니다.
- 세션이 테스트 로드를 수행하도록 구성되었습니다.
- 세션이 디버그 세션입니다.
- 세션이 디버그 모드에서 실행되고 세션 출력을 무시하도록 구성되었습니다.

## 구문

**SETMINVARIABLE( \$\$Variable, value )**

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<b>\$\$변수</b>	필수	설정할 매핑 변수의 이름입니다. 최소 집계 유형의 매핑 변수에 사용됩니다.
<b>값</b>	필수	<b>PowerCenter</b> 통합 서비스에서 변수의 현재 값과 비교할 값입니다. 변수의 데이터 유형과 호환되는 데이터 유형으로 평가되는 모든 유효한 변환 식을 입력할 수 있습니다.

## 반환 값

다음 두 값 중에서 더 작은 값이 반환됩니다. 변수의 현재 값 또는 사용자가 지정한 값. 반환 값은 변수의 새 현재 값입니다.

값이 **NULL**인 경우 **PowerCenter** 통합 서비스가 **\$\$Variable**의 현재 값을 반환합니다.

## 예

다음 식은 항목의 가격을 매핑 변수 `$$MinPrice`와 비교합니다. 이 식은 `$$MinPrice`를 두 값 중 작은 값으로 설정하고 지금까지의 항목 가격 중 가장 작은 값을 `MIN_PRICE` 포트에 반환합니다. 이전 세션 실행의 `$MinPrice` 초기 값은 22.50입니다.

```
SETMINVARIABLE ($$MinPrice, PRICE)
```

DATE	PRICE	MIN_PRICE
05/01/2000 09:00:00	23.50	22.50
05/01/2000 10:00:00	27.00	22.50
05/01/2000 11:00:00	26.75	22.50
05/01/2000 12:00:00	25.25	22.50
05/01/2000 13:00:00	22.00	22.00
05/01/2000 14:00:00	22.75	22.00
05/01/2000 15:00:00	23.00	22.00
05/01/2000 16:00:00	24.25	22.00
05/01/2000 17:00:00	24.00	22.00

세션이 종료되면 PowerCenter 통합 서비스가 22.00을 `$$MinPrice`의 최소 현재 값으로 리포지토리에 저장합니다. 다음에 세션을 실행하면 PowerCenter 통합 서비스가 `$$MinPrice`에 대한 초기 값을 22.00으로 평가합니다.

3개의 파티션을 포함하는 동일한 세션이 있는 경우 PowerCenter 통합 서비스는 `$$MinPrice`를 각 파티션에 대해 평가합니다. 그런 다음 가장 작은 값을 리포지토리에 저장합니다. 예를 들어 각 파티션에서 마지막으로 평가된 `$$MinPrice` 값은 다음과 같습니다.

Partition	Final Current Value for \$\$MinPrice
Partition 1	22.00
Partition 2	22.50
Partition 3	22.50

## SETVARIABLE

매핑 변수의 현재 값이 사용자가 지정한 값으로 설정됩니다. 지정한 값이 반환됩니다. `SETVARIABLE` 함수는 행이 삽입 또는 업데이트하도록 표시된 경우에만 실행됩니다. `SETVARIABLE`은 기타 모든 행 유형을 무시하며, 현재 값은 변경되지 않고 유지됩니다.

세션이 성공적으로 완료되면 **PowerCenter** 통합 서비스가 변수의 최종 현재 값을 변수의 시작 값과 비교합니다. 그런 다음 변수의 집계 유형에 따라 최종 현재 값을 리포지토리에 저장합니다. 재정의하지 않은 경우 다음에 이 세션을 실행하면 저장된 값이 변수의 초기 값으로 사용됩니다.

**SETVARIABLE** 함수는 파이프라인의 각 매핑 변수에 대해 한 번씩만 사용해야 합니다. **PowerCenter** 통합 서비스는 변수 함수를 매핑에서 발견되는 대로 처리합니다. **PowerCenter** 통합 서비스가 매핑에서 변수 함수를 발견하는 순서는 세션 실행에 따라 다를 수 있습니다. 따라서 매핑에서 동일한 변수 함수를 여러 번 사용할 경우 일관되지 않은 결과가 생성될 수 있습니다.

**SETVARIABLE**은 다음 변환에 사용됩니다.

- 식
- 필터
- 라우터
- 업데이트 전략

다음 조건 중 하나가 참일 경우 **PowerCenter** 통합 서비스가 매핑 변수의 최종 값을 리포지토리에 저장하지 않습니다.

- 세션 완료가 실패했습니다.
- 세션이 테스트 로드를 수행하도록 구성되었습니다.
- 세션이 디버그 세션입니다.
- 세션이 디버그 모드에서 실행되고 세션 출력을 무시하도록 구성되었습니다.

## 구문

**SETVARIABLE( \$\$Variable, value )**

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<b>\$\$변수</b>	필수	설정할 매핑 변수의 이름입니다. 최대/최소 집계 유형의 매핑 변수에 사용합니다.
<b>값</b>	필수	변수의 현재 값으로 설정할 값입니다. 변수의 데이터 유형과 호환되는 데이터 유형으로 평가되는 모든 유효한 변환 식을 입력할 수 있습니다.

## 반환 값

변수의 현재 값입니다.

값이 **NULL**인 경우 **PowerCenter** 통합 서비스가 **\$\$Variable**의 현재 값을 반환합니다.

## 예

다음 식은 매핑 변수 \$\$Time을 PowerCenter 통합 서비스가 행을 평가할 때의 시스템 날짜로 설정하고 해당 시스템 날짜를 SET\_\$\$TIME 포트에 반환합니다.

```
SETVARIABLE ($$Time, SYSDATE)
```

TRANSACTION	TOTAL	SET_\$\$TIME
0100002	534.23	10/10/2000 01:34:33
0100003	699.01	10/10/2000 01:34:34
0100004	97.50	10/10/2000 01:34:35
0100005	116.43	10/10/2000 01:34:36
0100006	323.95	10/10/2000 01:34:37

세션이 종료되면 PowerCenter 통합 서비스가 10/10/2000 01:34:37을 \$\$Time의 마지막으로 평가된 값으로 리포지토리에 저장합니다. 다음에 세션을 실행하면 PowerCenter 통합 서비스가 \$\$Time에 대한 모든 참조를 10/10/2000 01:34:37로 평가합니다.

다음 식은 매핑 변수 \$\$Timestamp를 행에 연결된 타임스탬프로 설정하고 해당 타임스탬프를 SET\_\$\$TIMESTAMP 포트에 반환합니다.

```
SETVARIABLE ($$Time, TIMESTAMP)
```

TRANSACTION	TIMESTAMP	TOTAL	SET_\$\$TIMESTAMP
0100002	10/01/2000 12:01:01	534.23	10/01/2000 12:01:01
0100003	10/01/2000 12:10:22	699.01	10/01/2000 12:10:22
0100004	10/01/2000 12:16:45	97.50	10/01/2000 12:16:45
0100005	10/01/2000 12:23:10	116.43	10/01/2000 12:23:10
0100006	10/01/2000 12:40:31	323.95	10/01/2000 12:40:31

세션이 종료되면 PowerCenter 통합 서비스가 10/01/2000 12:40:31을 \$\$Timestamp의 마지막으로 평가된 값으로 리포지토리에 저장합니다.

다음에 세션을 실행하면 PowerCenter 통합 서비스가 \$\$Timestamp에 대한 초기 값을 10/01/2000 12:40:31로 평가합니다.

## SHA256

입력 값의 SHA-256 다이제스트를 계산합니다. 이 함수는 SHA-2(Secure Hash Algorithm 2)를 사용하고 256비트 다이제스트를 반환합니다. SHA-256은 256비트 해시 값을 포함하는 단방향 암호화 해시 함수입니다. 입력 값의 SHA-256이 다를 경우 입력 값이 다르다는 결론을 내릴 수 있습니다. SHA256을 사용하여 데이터 무결성을 확인하거나 고유 키를 생성할 수 있습니다.

PowerCenter 통합 서비스가 ASCII 및 유니코드 모드에서 실행되는 경우 SHA-256 함수는 다른 값을 반환합니다.

SHA256 함수를 사용하려면 INFA\_ENABLE\_BINARY\_FUNCTIONS 환경 변수를 True 또는 Yes로 설정합니다.

#### 구문

SHA256( *value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 또는 이진 데이터 유형입니다. 다이제스트를 계산하려는 대/소문자 구분 값입니다.

#### 반환 값

고유한 32바이트 이진.

입력이 Null 값인 경우 NULL이 반환됩니다.

#### 예제

변경된 데이터를 데이터베이스에 쓰려고 합니다. SHA256을 사용하여 소스에서 읽는 데이터 행에 대한 다이제스트 값을 생성합니다. 세션을 실행하면 이전에 생성한 체크섬 값과 새 체크섬 값이 비교됩니다. 그런 다음 업데이트된 체크섬 값이 대상에 기록됩니다. 사용자는 업데이트된 체크섬 값을 보고 데이터가 변경된 것을 알 수 있습니다.

**참고:** 대부분의 경우 SHA256을 사용하여 기본 키를 생성할 수 있습니다.

## SIGN

숫자 값을 양수, 음수, 0 중 하나로 반환합니다.

#### 구문

SIGN( *numeric\_value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 값. 평가할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

#### 반환 값

음수 값의 경우 -1이 반환됩니다.

0의 경우 0이 반환됩니다.

양수 값의 경우 1이 반환됩니다.

NULL의 경우 NULL이 반환됩니다.

## 예

다음 식은 **SALES** 포트에 음수 값이 포함되는지를 결정합니다.

**SIGN( SALES )**

SALES	RETURN VALUE
100	1
-25.99	-1
0	0
NULL	NULL

## SIN

라디안으로 표시된 숫자 값의 사인을 반환합니다.

## 구문

**SIN( *numeric\_value* )**

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 라디안(각도에 파이를 곱하고 180으로 나눈 값)으로 표시된 숫자 데이터. 사인을 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 연산자를 사용하여 숫자 값을 라디안으로 변환하거나 SIN 계산 안에서 산술 계산을 수행할 수 있습니다.

## 반환 값

배정밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 **Degrees** 포트의 값을 라디안으로 변환한 다음 각 라디안에 대한 사인을 계산합니다.

**SIN( DEGREES \* 3.14159265359 / 180 )**

DEGREES	RETURN VALUE
0	0
90	1
70	0.939692620785936
30	0.500000000000003

DEGREES	RETURN VALUE
5	0.0871557427476639
89	0.999847695156393
NULL	NULL

SIN 함수가 사인을 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다. 예:

`SIN( ARCS * 3.14159265359 / 180 )`

## SINH

숫자 값의 쌍곡선 사인을 반환합니다.

### 구문

`SINH( numeric_value )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 라디안(각도에 파이를 곱하고 180으로 나눈 값)으로 표시된 숫자 데이터. 쌍곡선 사인을 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

배정밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

### 예

다음 식은 **Angles** 포트의 값에 대한 쌍곡선 사인을 반환합니다.

`SINH( ANGLES )`

ANGLES	RETURN VALUE
1.0	1.1752011936438
2.897	9.03225804884884
3.66	19.4178051793031
5.45	116.376934801486
0	0.0

ANGLES	RETURN VALUE
0.345	0.35188478309993
NULL	NULL

#### 팁

SINH 함수가 쌍곡선 사인을 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다. 예:

`SINH( MEASURES.ARCS / 180 )`

## SOUNDEX

문자열 값을 4자 문자열로 인코딩합니다.

SOUNDEX는 영어 알파벳(A-Z) 문자에 작용합니다. 이 함수는 입력 문자열의 첫 번째 문자를 반환 값의 첫 번째 문자로 사용하고 나머지 3개의 고유한 자음을 숫자로 인코딩합니다.

SOUNDEX는 다음 규칙 목록에 따라 문자를 인코딩합니다.

- 문자열의 첫 번째 문자를 반환 값의 첫 번째 문자로 사용하고 대문자로 인코딩합니다. 예를 들어 SOUNDEX('John')과 SOUNDEX('john')은 모두 'J500'을 반환합니다.
- 문자열의 첫 번째 문자 다음에 처음 3개의 고유 자음을 인코딩하고 나머지는 무시합니다. 예를 들어 SOUNDEX('JohnRB')와 SOUNDEX('JohnRBCD')는 모두 'J561'을 반환합니다.
- 소리가 비슷한 자음에 하나의 코드를 할당합니다.

다음 테이블에는 자음에 대한 SOUNDEX 인코딩 지침이 나열되어 있습니다.

**테이블 2. SOUNDEX 자음 인코딩 지침**

코드	자음
1	B, P, F, V
2	C, S, G, J, K, Q, X, Z
3	D, T
4	L
5	M, N
6	R

- 문자열의 첫 번째 문자가 아닌 A, E, I, O, U, H 및 W 문자는 건너뜁니다. 예를 들어 SOUNDEX('A123')은 'A000'을 반환하고 SOUNDEX('MAeiouhWC')는 'M200'을 반환합니다.
- 문자열이 생성하는 문자가 4자 미만인 경우 결과 문자열에 0이 채워집니다. 예를 들어 SOUNDEX('J')는 'J000'을 반환합니다.



- 문자열에 [“SOUNDEX” 페이지 184](#)에 나열된 동일한 샘플 코드를 사용하는 연속 자음 집합이 포함되는 경우 SOUNDEX는 첫 번째 일치 항목을 인코딩하고 집합의 나머지 일치 항목은 건너뛵니다. 예를 들어 SOUNDEX(‘AbbpdMN’)은 ‘A135’를 반환합니다.
- 문자열의 숫자는 건너뛵니다. 예를 들어 SOUNDEX(‘Joh12n’)과 SOUNDEX(‘1John’)은 모두 ‘J500’을 반환합니다.
- 문자열이 NULL이거나 문자열의 모든 문자가 영어 알파벳이 아닌 경우 NULL이 반환됩니다.

## 구문

SOUNDEX( *string* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>string</i>	필수	문자열입니다. 인코딩할 문자열 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

문자열.

다음 조건 중 하나가 참인 경우 NULL이 반환됩니다.

- 함수에 전달된 값이 NULL입니다.
- 문자열에 영어 알파벳 문자가 없습니다.
- 문자열이 비어 있습니다.

## 예

다음 식은 EMPLOYEE\_NAME 포트의 값을 인코딩합니다.

SOUNDEX( EMPLOYEE\_NAME )

EMPLOYEE_NAME	RETURN VALUE
John	J500
William	W450
jane	J500
joh12n	J500
1abc	A120
NULL	NULL

## SQRT

음수가 아닌 숫자 값의 제곱근을 반환합니다.

## 구문

`SQRT( numeric_value )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<code>numeric_value</code>	필수	양수 값. 제곱근을 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

배정 밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 Numbers 포트의 값에 대한 제곱근을 반환합니다.

`SQRT( NUMBERS )`

NUMBERS	RETURN VALUE
100	10
-100	<i>Error. PowerCenter 통합 서비스 does not write row.</i>
NULL	NULL
60.54	7.78074546557076

값 -100은 오류를 야기합니다. 함수 **SQRT**는 양수 값만 평가하기 때문입니다. 음수 또는 문자 값을 전달하면 **PowerCenter** 통합 서비스가 변환 평가 오류를 표시하고 해당 행을 쓰지 않습니다.

**SQRT** 함수가 제곱근을 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다.

## STDDEV

이 함수에 전달한 숫자 값의 표준 편차를 반환합니다. **STDDEV**는 통계 데이터를 분석하는 데 사용됩니다. **STDDEV** 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 숫자 데이터 유형을 반환해야 합니다.

## 구문

`STDDEV( numeric_value [, filter_condition] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 이 함수는 표준 편차를 계산할 값 또는 함수의 결과를 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 연산자를 사용하여 다른 포트의 값에 대한 평균을 구할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

## Null

단일 값이 NULL인 경우 무시됩니다. 그러나 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

## 그룹 기준

STDDEV는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 STDDEV 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

다음 식은 TOTAL\_SALES 포트에서 \$2000.00보다 큰 모든 행의 표준 편차를 계산합니다.

```
STDDEV( SALES, SALES > 2000.00 )
```

## SALES

2198.0

1010.90

2256.0

153.88

3001.0

## SALES

NULL

8953.0

**RETURN VALUE:** 3254.60361129688

값 1010.90과 153.88은 함수 계산에 포함하지 않습니다. *filter\_condition*에 \$2,000보다 큰 매출이 지정되어 있기 때문입니다.

다음 식은 **SALES** 포트에 있는 모든 행의 표준 편차를 계산합니다.

STDDEV(SALES)

## SALES

2198.0

2198.0

2198.0

2198.0

**RETURN VALUE:** 0

각 행에 같은 숫자가 포함되어 있기 때문에 반환 값은 0입니다(표준 편차가 없음). 표준 편차가 없는 경우 반환 값은 0입니다.

## SUBSTR

문자열의 일부를 반환합니다. **SUBSTR**에서는 공백 등 모든 문자 개수가 문자열의 처음부터 계산됩니다.

구문

SUBSTR( *string*, *start* [, *length*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자 문자열이어야 합니다. 검색할 열을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 숫자 값을 전달하는 경우 함수가 문자열로 변환합니다.
<i>시작</i>	필수	정수여야 합니다. 세기 시작할 문자열의 위치입니다. 유효한 모든 변환 식을 입력할 수 있습니다. 시작 위치가 양수인 경우 SUBSTR 함수는 문자열의 처음부터 수를 계산하여 시작 위치를 찾습니다. 시작 위치가 음수인 경우 SUBSTR 함수는 문자열의 끝부터 수를 계산하여 시작 위치를 찾습니다. 시작 위치가 0인 경우 SUBSTR 함수는 문자열의 첫 문자부터 검색합니다.
<i>길이</i>	선택 사항	0보다 큰 정수여야 합니다. SUBSTR 함수를 통해 반환할 문자 수를 입력합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 길이 인수를 생략할 경우 문자열의 시작 위치부터 끝까지 모든 문자가 반환됩니다. 음의 정수 또는 0을 전달하는 경우 함수는 빈 문자열을 반환합니다. 소수를 전달하는 경우 함수는 근사한 정수 값으로 반올림합니다.

## 반환 값

문자열.

음수 또는 길이가 0인 값을 전달하는 경우 빈 문자열입니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 Phone 포트에 있는 각 행의 지역 번호를 반환합니다.

SUBSTR( PHONE, 0, 3 )

PHONE	RETURN VALUE
809-555-0269	809
357-687-6708	357
NULL	NULL

SUBSTR( PHONE, 1, 3 )

PHONE	RETURN VALUE
809-555-3915	809
357-687-6708	357
NULL	NULL

다음 식은 **Phone** 포트에 있는 각 행의 지역 번호를 제외한 전화 번호를 반환합니다.

```
SUBSTR( PHONE, 5, 8 )
```

PHONE	RETURN VALUE
808-555-0269	555-0269
809-555-3915	555-3915
357-687-6708	687-6708
NULL	NULL

음수 시작 값을 전달하여 **Phone** 포트에 있는 각 행의 전화 번호를 반환할 수도 있습니다. 이 식도 *length* 인수의 결과를 반환할 때 왼쪽에서 오른쪽으로 소스 문자열을 읽습니다.

```
SUBSTR( PHONE, -8, 3 )
```

PHONE	RETURN VALUE
808-555-0269	555
809-555-3915	555
357-687-6708	687
NULL	NULL

**INSTR**를 *start* 또는 *length* 인수에 중첩하여 특정 문자열을 검색하고 위치를 반환할 수 있습니다.

다음 식은 문자열의 끝부터 시작하여 문자열을 평가합니다. 이 식은 문자열에서 마지막(맨 오른쪽) 공백을 찾은 다음 이 공백의 앞에 있는 모든 문자를 반환합니다.

```
SUBSTR( CUST_NAME,1,INSTR( CUST_NAME,' ', -1,1 ) - 1 )
```

CUST_NAME	RETURN VALUE
PATRICIA JONES	PATRICIA
MARY ELLEN SHAH	MARY ELLEN

다음 식은 문자열에서 문자 '#'를 제거합니다.

```
SUBSTR( CUST_ID, 1, INSTR(CUST_ID, '#')-1 ) || SUBSTR( CUST_ID, INSTR(CUST_ID, '#')+1 )
```

*length* 인수가 문자열보다 긴 경우 문자열의 시작 위치부터 끝까지 모든 문자가 반환됩니다. 다음 예제를 고려하십시오.

```
SUBSTR('abcd', 2, 8)
```

반환 값은 'bcd'입니다. 이 결과를 다음 예와 비교하십시오.

```
SUBSTR('abcd', -2, 8)
```

반환 값은 'cd'입니다.

## SUM

선택한 포트에 있는 모든 값의 합계를 반환합니다. 필요한 경우 필터를 적용하여 합계를 계산할 때 읽을 행을 제한할 수 있습니다. SUM 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 숫자 데이터 유형을 반환해야 합니다.

### 구문

```
SUM( numeric_value [, filter_condition ] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 더할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다. 연산자를 사용하여 다른 포트의 값을 더할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

숫자 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: 필터 조건이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 높은 정밀도를 활성화하여 10진수 전체 자릿수를 최대 38자리까지 보장할 수 있습니다.

### Null

단일 값이 NULL인 경우 무시됩니다. 그러나 포트에서 전달된 모든 값이 NULL인 경우에는 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

### 그룹 기준

SUM은 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 SUM 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

### 예

다음 식은 Sales 포트에서 2000보다 큰 모든 값의 합계를 반환합니다.

```
SUM( SALES, SALES > 2000 )
```

### SALES

2500.0

## SALES

1900.0

1200.0

NULL

3458.0

4519.0

**RETURN VALUE:** 10477.0

### 팁

SUM 함수가 합계를 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다. 예:

`SUM( QTY * PRICE - DISCOUNT )`

## SYSTIMESTAMP

PowerCenter 통합 서비스를 호스트하는 노드의 현재 날짜 및 시간을 나노초 단위의 전체 자릿수로 반환합니다. 날짜 및 시간을 표시할 전체 자릿수는 플랫폼에 따라 다릅니다.

이 함수의 반환 값은 구성된 인수에 따라 다릅니다.

- SYSTIMESTAMP의 인수를 변수로 구성하면 PowerCenter 통합 서비스가 변환의 각 행에 대해 함수를 평가합니다.
- SYSTIMESTAMP의 인수를 상수로 구성하면 PowerCenter 통합 서비스가 함수를 한 번 평가하고 변환의 각 행에 대해 이 값을 유지합니다.

### 구문

`SYSTIMESTAMP( [format] )`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
형식	선택 사항	타임스탬프를 검색할 전체 자릿수입니다. 전체 자릿수를 초(SS), 밀리초(MS), 마이크로초(US) 또는 나노초(NS)까지 지정할 수 있습니다. 형식 문자열은 작은따옴표로 묶습니다. 형식 문자열은 대/소문자를 구분하지 않습니다. 예를 들어 밀리초의 전체 자릿수로 날짜 및 시간을 표시하려면 다음 구문을 사용합니다. SYSTIMESTAMP('MS'). 기본 전체 자릿수는 마이크로초(US)입니다.

### 반환 값

타임스탬프. 지정한 전체 자릿수의 날짜 및 시간이 반환됩니다.



## 예

온라인 주문 서비스를 제공하고 실시간 데이터를 처리하는 조직이 있습니다. 이 조직의 직원은 **SYSTIMESTAMP** 함수를 사용하여 대상 데이터베이스에 있는 각 트랜잭션에 대한 기본 키를 생성할 수 있습니다.

다음 포트 및 값을 포함하는 식 변환을 작성합니다.

Port Name	Port Type	Expression
Customer_Name	Input	n/a
Order_Qty	Input	n/a
Time_Counter	Variable	'US'
Transaction_Id	Output	SYSTIMESTAMP ( Time_Counter )

런타임 시 **PowerCenter** 통합 서비스는 각 행에 대한 시스템 시간을 마이크로초의 전체 자릿수로 생성합니다.

Customer_Name	Order_Qty	Transaction_Id
Vani Deed	14	07/06/2007 18:00:30.701015000
Kalia Crop	3	07/06/2007 18:00:30.701029000
Vani Deed	6	07/06/2007 18:00:30.701039000
Harry Spoon	32	07/06/2007 18:00:30.701048000

## TAN

라디안으로 표시된 숫자 값의 탄젠트를 반환합니다.

### 구문

**TAN**( *numeric\_value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 라디안(각도에 파이를 곱하고 180으로 나눈 값)으로 표시된 숫자 데이터. 탄젠트를 계산할 숫자 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

배정밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 Degrees 포트의 모든 값에 대한 탄젠트를 반환합니다.

TAN( DEGREES \* 3.14159 / 180 )

DEGREES	RETURN VALUE
70	2.74747741945531
50	1.19175359259435
30	0.577350269189672
5	0.0874886635259298
18	0.324919696232929
89	57.2899616310952
NULL	NULL

## TANH

이 함수에 전달된 숫자 값의 쌍곡선 탄젠트를 반환합니다.

## 구문

TANH( *numeric\_value* )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 라디안(각도에 파이를 곱하고 180으로 나눈 값)으로 표시된 숫자 데이터. 쌍곡선 탄젠트를 계산할 숫자 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

배정 밀도 값.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 Angles 포트의 값에 대한 쌍곡선 탄젠트를 반환합니다.

TANH( ANGLES )

ANGLES	RETURN VALUE
1.0	0.761594155955765

ANGLES	RETURN VALUE
2.897	0.993926947790665
3.66	0.998676551914886
5.45	0.999963084213409
0	0.0
0.345	0.331933853503641
NULL	NULL

#### 팁

TANH 함수가 쌍곡선 탄젠트를 계산하기 전에 함수에 전달된 값에 대한 산술 계산을 수행할 수 있습니다.  
예:

TANH( ARCS / 360 )

## TIME\_RANGE

조인할 스트리밍 이벤트의 시간 범위를 결정합니다.

TIME\_RANGE 함수는 스트리밍 매핑의 조이너 변환에만 적용할 수 있습니다.

#### 구문

TIME\_RANGE(EventTime1,EventTime2,Format,Interval)

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
EventTime1	필수	날짜 데이터 유형. 조이너 변환의 마스터 포트에서 스트리밍 이벤트가 생성된 시간입니다.
EventTime2	필수	날짜 데이터 유형. 조이너 변환의 세부 정보 포트에서 스트리밍 이벤트가 생성된 시간입니다.

인수	필수/선택	설명
형식	필수	<p>변경할 이벤트 시간 값 부분을 지정하는 형식 문자열입니다. 형식 문자열은 작은따옴표로 묶습니다. 예를 들어 'Seconds'와 같이 묶으면 됩니다. format 문자열은 대/소문자를 구분하지 않습니다.</p> <p>format 인수는 다음 값을 허용합니다.</p> <ul style="list-style-type: none"> <li>- 연도</li> <li>- 월</li> <li>- 주</li> <li>- 일</li> <li>- 시간</li> <li>- 분</li> <li>- 초</li> <li>- 밀리초</li> <li>- 마이크로초</li> </ul>
간격	필수	형식을 기반으로 이벤트 시간 값을 변경하려는 정수 값입니다.

### 반환 값

Null 값을 함수에 전달하는 경우 NULL이 반환됩니다.

### 예

다음 예에서는 조이너 변환의 시간 범위 식을 반환합니다.

```
TIME_RANGE(EventTime1,EventTime2,'Second',4)
```

반환 값:

```
(EventTime1.<=(EventTime2).&&(EventTime2.<=(EventTime1.+(expr("INTERVAL 4 SECONDS")))))
```

## TO\_BIGINT

문자열 또는 숫자 값을 **bigint** 값으로 변환합니다. TO\_BIGINT 구문에는 숫자를 가장 가까운 정수로 반올림하거나 소수부를 자르는 데 사용할 수 있는 선택적 인수가 있습니다. TO\_BIGINT는 선행 공백을 무시합니다.

### 구문

```
TO_BIGINT( value [, flag] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>값</i>	필수	문자열 또는 숫자 데이터 유형. bigint 값으로 변환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>플래그</i>	선택 사항	소수부를 자를지, 아니면 반올림할지를 지정합니다. 플래그는 정수 리터럴이거나 상수 TRUE 또는 FALSE여야 합니다. 플래그가 TRUE이거나 0이 아닌 숫자인 경우 소수부가 잘립니다. 플래그가 FALSE 또는 0이거나 사용자가 이 인수를 생략한 경우 값이 가까운 정수로 반올림됩니다. 플래그는 기본적으로 설정되지 않습니다.

## 반환 값

Bigint.

함수에 전달된 값이 NULL인 경우 NULL이 반환됩니다.

함수에 전달된 값에 영숫자 문자가 포함되는 경우 0이 반환됩니다.

## 예

다음 식은 포트 IN\_TAX의 값을 사용합니다.

```
TO_BIGINT( IN_TAX, TRUE )
```

IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123435
'7245176201123435.2'	7245176201123435
'7245176201123435.2.48'	7245176201123435
NULL	NULL
'A12.3Grove'	0
'176201123435.87'	176201123435
'-7245176201123435.2'	-7245176201123435
'-7245176201123435.23'	-7245176201123435

IN_TAX	RETURN VALUE
-9223372036854775806.9	-9223372036854775806
9223372036854775806.9	9223372036854775806
TO_BIGINT( IN_TAX )	

IN_TAX	RETURN VALUE
'7245176201123435.6789'	7245176201123436
'7245176201123435.2'	7245176201123435
'7245176201123435.348'	7245176201123435
NULL	NULL
'A12.3Grove'	0
' 176201123435.87'	176201123436
'-7245176201123435.6789'	-7245176201123436
'-7245176201123435.23'	-7245176201123435
-9223372036854775806.9	-9223372036854775807
9223372036854775806.9	9223372036854775807

## TO\_CHAR(날짜)

날짜를 문자열로 변환합니다. TO\_CHAR는 숫자 값도 문자열로 변환합니다. TO\_CHAR 형식 문자열을 사용하여 날짜를 모든 형식으로 변환할 수 있습니다.

TO\_CHAR (date [,format])는 데이터 유형 또는 날짜, 타임스탬프, 시간대가 지정된 타임스탬프 또는 로컬 시간대가 지정된 타임스탬프 데이터 유형의 내부 값을 형식 문자열에 의해 지정된 문자열 데이터 유형의 값으로 변환합니다.

### 구문

TO\_CHAR( date [,format] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
날짜	필수	날짜/시간 데이터 유형. 문자열로 변환할 날짜 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
형식	선택 사항	유효한 TO_CHAR 형식 문자열을 입력합니다. 형식 문자열은 날짜 인수의 값에 대한 형식을 정의하는 것이 아니라 반환 값의 형식을 정의합니다. 형식 문자열을 생략할 경우 세션에 지정된 날짜 형식에 따라 문자열이 반환됩니다.

#### 반환 값

문자열.

함수에 전달된 값이 NULL인 경우 NULL입니다.

#### 예

다음 식은 DATE\_PROMISED 포트의 날짜를 MON DD YYYY 형식의 텍스트로 변환합니다.

```
TO_CHAR( DATE_PROMISED, 'MON DD YYYY' )
```

DATE_PROMISED	RETURN VALUE
Apr 1 1998 12:00:10AM	'Apr 01 1998'
Feb 22 1998 01:31:10PM	'Feb 22 1998'
Oct 24 1998 02:12:30PM	'Oct 24 1998'
NULL	NULL

형식 인수를 생략할 경우 TO\_CHAR는 세션에 지정된 날짜 형식(기본적으로 MM/DD/YYYY HH24:MI:SS.US)으로 문자열을 반환합니다.

```
TO_CHAR( DATE_PROMISED )
```

DATE_PROMISED	RETURN VALUE
Apr 1 1998 12:00:10AM	'04/01/1998 00:00:10.000000'
Feb 22 1998 01:31:10PM	'02/22/1998 13:31:10.000000'
Oct 24 1998 02:12:30PM	'10/24/1998 14:12:30.000000'
NULL	NULL

다음 식은 포트의 각 날짜에 대한 주의 일을 반환합니다.

TO\_CHAR( DATE\_PROMISED, 'D' )

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'3'
02-22-1997 01:31:10PM	'7'
10-24-1997 02:12:30PM	'6'
NULL	NULL

TO\_CHAR( DATE\_PROMISED, 'DAY' )

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'Tuesday'
02-22-1997 01:31:10PM	'Saturday'
10-24-1997 02:12:30PM	'Friday'
NULL	NULL

다음 식은 포트의 각 날짜에 대한 월의 일을 반환합니다.

TO\_CHAR( DATE\_PROMISED, 'DD' )

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'01'
02-22-1997 01:31:10PM	'22'
10-24-1997 02:12:30PM	'24'
NULL	NULL

다음 식은 포트의 각 날짜에 대한 연도의 일을 반환합니다.

TO\_CHAR( DATE\_PROMISED, 'DDD' )

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'091'
02-22-1997 01:31:10PM	'053'



DATE_PROMISED	RETURN VALUE
10-24-1997 02:12:30PM	'297'
NULL	NULL

다음 식은 포트의 각 날짜에 대한 일의 시간을 반환합니다.

```
TO_CHAR( DATE_PROMISED, 'HH' )
TO_CHAR( DATE_PROMISED, 'HH12' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'12'
02-22-1997 01:31:10PM	'01'
10-24-1997 02:12:30PM	'02'
NULL	NULL

```
TO_CHAR( DATE_PROMISED, 'HH24' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'00'
02-22-1997 01:31:10PM	'13'
10-24-1997 11:12:30PM	'23'
NULL	NULL

다음 식은 날짜 값을 문자열로 표시되는 MJD 값으로 변환합니다.

```
TO_CHAR( SHIP_DATE, 'J' )
```

SHIP_DATE	RETURN_VALUE
Dec 31 1999 03:59:59PM	2451544
Jan 1 1900 01:02:03AM	2415021

다음 식은 날짜를 MM/DD/YY 형식의 문자열로 변환합니다.

```
TO_CHAR( SHIP_DATE, 'MM/DD/RR' )
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	12/31/99

SHIP_DATE	RETURN_VALUE
09/15/1996 03:59:59PM	09/15/96
05/17/2003 12:13:14AM	05/17/03

형식 문자열 SSSSS를 TO\_CHAR 식에서 사용할 수도 있습니다. 예를 들어 다음 식은 SHIP\_DATE 포트의 날짜를 자정 이후의 전체 초 수를 나타내는 문자열로 변환합니다.

```
TO_CHAR( SHIP_DATE, 'SSSSS')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	3783
09/15/1996 03:59:59PM	86399

TO\_CHAR 식에서 YY 형식 문자열은 RR 형식 문자열과 동일한 결과를 생성합니다.

다음 식은 날짜를 MM/DD/YY 형식의 문자열로 변환합니다.

```
TO_CHAR( SHIP_DATE, 'MM/DD/YY')
```

SHIP_DATE	RETURN_VALUE
12/31/1999 01:02:03AM	12/31/99
09/15/1996 03:59:59PM	09/15/96
05/17/2003 12:13:14AM	05/17/03

다음 식은 포트의 각 날짜에 대한 월의 주를 반환합니다.

```
TO_CHAR( DATE_PROMISED, 'W' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10AM	'01'
02-22-1997 01:31:10AM	'04'
10-24-1997 02:12:30PM	'04'
NULL	NULL

다음 식은 포트의 각 날짜에 대한 연도의 주를 반환합니다.

```
TO_CHAR( DATE_PROMISED, 'WW' )
```

DATE_PROMISED	RETURN VALUE
04-01-1997 12:00:10PM	'18'
02-22-1997 01:31:10AM	'08'
10-24-1997 02:12:30AM	'43'
NULL	NULL

## 팁

다음과 같이 TO\_CHAR과 TO\_DATE를 함께 사용하면 월의 숫자 값을 텍스트 값으로 변환할 수 있습니다.

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

## TO\_CHAR(숫자)

숫자 값을 텍스트 문자열로 변환합니다. TO\_CHAR는 날짜도 문자열로 변환합니다.

## 구문

```
TO_CHAR( numeric_value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 문자열로 변환할 숫자 값입니다. 유효한 모든 변환 식을 입력할 수 있습니다.

TO\_CHAR는 다음과 같이 배정밀도 값을 텍스트 문자열로 변환합니다.

- 배정밀도 값을 최대 16자리의 문자열로 변환하고 최대 15자리의 정확도를 제공합니다. 15자리를 초과하는 숫자를 전달하는 경우 TO\_CHAR는 16자리수를 기반으로 숫자를 반올림하고 지수 표기법에서 숫자의 문자열 표현을 반환합니다. 예를 들어 1234567890123456 배정밀도 값을 '1.23456789012346e+015' 문자열 값으로 변환합니다.
- 범위(-1e16,-1e-16) 및 [1e-16, 1e16)의 숫자에 대해 10진수 표기법을 반환합니다. 이 범위 밖의 숫자에 대해서는 지수 표기법을 반환합니다. 예를 들어 10842764968208837340 배정밀도 값을 '1.08427649682088e+019' 문자열 값으로 변환합니다.

TO\_CHAR는 다음과 같이 10진수 값을 텍스트 문자열로 변환합니다.

- 높은 정밀도 모드에서, TO\_CHAR는 최대 28자리의 10진수 값을 문자열로 반환합니다. 28자리를 초과하는 10진수 값을 전달하는 경우 TO\_CHAR는 28자리보다 큰 숫자에 대해서는 지수 표기법을 반환합니다.
- 낮은 정밀도 모드에서, TO\_CHAR는 10진수 값을 배정밀도 값으로 처리합니다.

- 10진수 포트를 TO\_CHAR 함수에 전달했을 때 입력 값의 자릿수가 부족하여 10진수 포트의 소수 자릿수와 일치하지 않는 경우 TO\_CHAR 함수가 값에 0을 추가합니다.

예를 들어 10진수 포트의 소수 자릿수가 5이고 행의 값이 7.6901인 경우 TO\_CHAR 함수는 입력 값을 7.69010으로 처리하고 반환 값은 '7.69010'입니다.

## 반환 값

문자열.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 배정밀도 변환 예

다음 식은 SALES 포트의 배정밀도 값을 문자열로 변환합니다.

TO\_CHAR( SALES )

SALES	RETURN VALUE
1010.99	'1010.99'
-15.62567	'-15.62567'
10842764968208837340	'1.08427649682088e+019' (rounded based on the 16th digit and returns the value in scientific notation)
236789034569723	'236789034569723'
0	'0'
33.15	'33.15'
NULL	NULL

## 10진수 변환 예

다음 식은 SALES 포트의 10진수 값을 높은 정밀도 모드에서 문자열로 변환합니다.

TO\_CHAR( SALES )

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
1234567890123456789012345679	'1234567890123456789012345679'
1.234578945469649345876123456	'1.234578945469649345876123456'
0.999999999999999999999999999999	'0.999999999999999999999999999999'
12345678901234567890123456799 (28보다 큼)	'1.23456789012346e+028'

## TO\_DATE

문자열을 날짜/시간 데이터 유형으로 변환합니다. TO\_DATE 형식 문자열을 사용하여 소스 문자열의 형식을 지정해야 합니다.

출력 포트는 TO\_DATE 식에 대한 날짜/시간이어야 합니다.

TO\_DATE를 사용하여 2자리 연도를 변환하는 경우 RR 또는 YY 형식 문자열을 사용합니다. YYYY 형식 문자열을 사용하지 마십시오.

### 구문

TO\_DATE( *string* [, *format*] )

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
문자열	필수	문자열 데이터 유형이어야 합니다. 날짜로 변환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
형식	선택 사항	유효한 TO_DATE 형식 문자열을 입력합니다. 형식 문자열은 문자열 인수의 일부와 일치해야 합니다. 예를 들어 문자열 'Mar 15 1998 12:43:10AM'을 전달하는 경우 형식 문자열 'MON DD YYYY HH12:MI:SSAM'을 사용해야 합니다. 형식 문자열을 생략하는 경우 문자열 값은 세션에서 지정한 날짜 형식이어야 합니다.

### 반환 값

날짜.

TO\_DATE는 항상 날짜와 시간을 반환합니다. 시간 값이 없는 문자열을 전달할 경우 반환되는 날짜에는 항상 시간 00:00:00.000000000이 포함됩니다. 이 함수의 결과를 데이터 유형이 날짜/시간인 모든 대상 열에 매핑할 수 있습니다. 대상 열의 전체 자릿수가 나노초보다 작은 경우 PowerCenter 통합 서비스는 대상 열의 전체 자릿수와 일치하도록 날짜/시간 값을 자른 다음 날짜/시간 값을 대상에 씁니다.

Null 값을 이 함수에 전달하는 경우 NULL이 반환됩니다.

**경고:** TO\_DATE 문자열의 형식은 날짜 구분 기호를 포함하여 형식 문자열과 일치해야 합니다. 일치하지 않을 경우 PowerCenter 통합 서비스가 정확하지 않은 값을 반환하거나 레코드를 건너뛸 수 있습니다.

### 예

다음 식은 DATE\_PROMISED 포트의 문자열에 대한 날짜 값을 반환합니다. TO\_DATE는 항상 날짜와 시간을 반환합니다. 시간 값이 없는 문자열을 전달할 경우 반환되는 날짜에는 항상 시간 00:00:00.000000000이 포함됩니다. 20세기에 세션을 실행하는 경우 세기는 19가 됩니다. 이 예에서 PowerCenter 통합 서비스를 실행하는 노드의 현재 연도는 1998입니다. 대상 열의 날짜/시간 형식이 MON DD YY HH24:MI SS이므로 PowerCenter 통합 서비스는 초로 자른 날짜/시간 값을 대상에 씁니다.

TO\_DATE( DATE\_PROMISED, 'MM/DD/YY' )

DATE\_PROMISED

RETURN VALUE

'01/22/98'

Jan 22 1998 00:00:00

DATE_PROMISED	RETURN VALUE
'05/03/98'	May 3 1998 00:00:00
'11/10/98'	Nov 10 1998 00:00:00
'10/19/98'	Oct 19 1998 00:00:00
NULL	NULL

다음 식은 DATE\_PROMISED 포트의 문자열에 대한 날짜 및 시간 값을 반환합니다. 시간 값이 없는 문자열을 전달하면 PowerCenter 통합 서비스가 오류를 반환합니다. 20세기에 세션을 실행하는 경우 세기는 19가 됩니다. PowerCenter 통합 서비스를 실행하는 노드의 현재 연도는 1998입니다.

TO\_DATE( DATE\_PROMISED, 'MON DD YYYY HH12:MI:SSAM' )

DATE_PROMISED	RETURN VALUE
'Jan 22 1998 02:14:56PM'	Jan 22 1998 02:14:56PM
'Mar 15 1998 11:11:11AM'	Mar 15 1998 11:11:11AM
'Jun 18 1998 10:10:10PM'	Jun 18 1998 10:10:10PM
'October 19 1998'	<i>Error. Integration Service skips this row.</i>
NULL	NULL

다음 식은 SHIP\_DATE\_MJD\_STRING 포트의 문자열을 날짜 값으로 변환합니다.

TO\_DATE (SHIP\_DATE\_MJD\_STR, 'J')

SHIP_DATE_MJD_STR	RETURN VALUE
'2451544'	Dec 31 1999 00:00:00.000000000
'2415021'	Jan 1 1900 00:00:00.000000000

J 형식 문자열은 날짜의 시간 부분을 포함하지 않으므로 반환 값의 시간이 00:00:00.000000000으로 설정됩니다.

다음 식은 문자열을 4자리 연도 형식으로 변환합니다. 현재 연도는 1998입니다.

TO\_DATE( DATE\_STR, 'MM/DD/RR')

DATE_STR	RETURN VALUE
'04/01/98'	04/01/1998 00:00:00.000000000
'08/17/05'	08/17/2005 00:00:00.000000000

다음 식은 문자열을 4자리 연도 형식으로 변환합니다. 현재 연도는 1998입니다.

```
TO_DATE( DATE_STR, 'MM/DD/YY')
```

DATE_STR	RETURN VALUE
'04/01/98'	04/01/1998 00:00:00.000000000
'08/17/05'	08/17/1905 00:00:00.000000000

**참고:** 두 번째 행의 경우 RR은 2005년을 반환하고 YY는 1905년을 반환합니다.

다음 식은 문자열을 4자리 연도 형식으로 변환합니다. 현재 연도는 1998입니다.

```
TO_DATE( DATE_STR, 'MM/DD/Y')
```

DATE_STR	RETURN VALUE
'04/01/8'	04/01/1998 00:00:00.000000000
'08/17/5'	08/17/1995 00:00:00.000000000

다음 식은 문자열을 4자리 연도 형식으로 변환합니다. 현재 연도는 1998입니다.

```
TO_DATE( DATE_STR, 'MM/DD/YYYY')
```

DATE_STR	RETURN VALUE
'04/01/998'	04/01/1998 00:00:00.000000000
'08/17/995'	08/17/1995 00:00:00.000000000

다음 식은 자정 이후의 초 수를 포함하는 문자열을 날짜 값으로 변환합니다.

```
TO_DATE( DATE_STR, 'MM/DD/YYYY SSSS')
```

DATE_STR	RETURN VALUE
'12/31/1999 3783'	12/31/1999 01:02:03
'09/15/1996 86399'	09/15/1996 23:59:59

대상이 서로 다른 날짜 형식을 허용하는 경우 TO\_DATE와 IS\_DATE를 DECODE 함수와 함께 사용하여 허용 가능한 형식을 테스트할 수 있습니다. 예:

```
DECODE( TRUE,
  --test first format
  IS_DATE( CLOSE_DATE, 'MM/DD/YYYY HH24:MI:SS' ),
  --if true, convert to date
  TO_DATE( CLOSE_DATE, 'MM/DD/YYYY HH24:MI:SS' ),
  --test second format; if true, convert to date
  IS_DATE( CLOSE_DATE, 'MM/DD/YYYY' ), TO_DATE( CLOSE_DATE, 'MM/DD/YYYY' ),
```

```
--test third format; if true, convert to date
IS_DATE( CLOSE_DATE, 'MON DD YYYY'), TO_DATE( CLOSE_DATE, 'MON DD YYYY'),

--if none of the above
ERROR( 'NOT A VALID DATE' ) )
```

다음과 같이 TO\_CHAR과 TO\_DATE를 함께 사용하면 월의 숫자 값을 텍스트 값으로 변환할 수 있습니다.

```
TO_CHAR( TO_DATE( numeric_month, 'MM' ), 'MONTH' )
```

관련 항목:

- [“날짜 형식 문자열의 규칙 및 지침” 페이지 40](#)

## TO\_DECIMAL

문자열 또는 숫자 값을 10진수 값으로 변환합니다. TO\_DECIMAL은 선행 공백을 무시합니다.

구문

```
TO_DECIMAL( value [, scale] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	문자열 또는 숫자 데이터 유형이어야 합니다. 10진수 값으로 변환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
소수 자릿수	선택 사항	0과 28 사이(경계값 모두 포함)의 정수 리터럴이어야 합니다. 소수점 뒤에 허용되는 자릿수를 지정합니다. 이 인수를 생략할 경우 입력 값과 동일한 배율의 값이 반환됩니다.  10진수를 TO_DECIMAL 함수에 전달하여 10진수를 다른 소수 자릿수의 10진수로 캐스트하는 경우 소수 자릿수 인수는 최대 값이며 소수 자릿수를 확장할 수 없습니다. 예를 들어 소수 자릿수 인수가 5이고 입력 값의 소수 자릿수가 6이면 소수 자릿수가 잘립니다. 입력 값의 소수 자릿수가 4인 경우 소수 자릿수는 동일하게 유지됩니다.

반환 값

0과 28 사이(경계값 모두 포함)의 전체 자릿수 및 배율의 10진수가 반환됩니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

문자열에 숫자가 아닌 문자가 포함되는 경우 문자열의 숫자 부분이 숫자가 아닌 첫 번째 문자까지 변환됩니다.

첫 번째 숫자 문자가 숫자가 아닌 경우 0이 반환됩니다.

**참고:** 반환 값이 전체 자릿수가 15보다 큰 10진수인 경우 많은 전체 자릿수를 활성화하여 10진수 전체 자릿수를 최대 28자리까지 보장할 수 있습니다.



## 예

이 식은 포트 IN\_TAX의 값을 사용합니다. IN\_TAX는 전체 자릿수가 44자리인 문자열 데이터 유형입니다. RETURN VALUE는 전체 자릿수가 28자리이고 배율이 3인 10진수 값입니다.

```
TO_DECIMAL( IN_TAX, 3 )
```

IN_TAX	RETURN VALUE
'15.6789'	15.679
'60.2'	60.200
'118.348'	118.348
NULL	NULL
'A12.3Grove'	0
'711A1'	711
'1234567890.123'	1234567890.123
'123456789012345678901234567890.123'	Error. Integration Service skips this row.
'1234567890123456789012345678901234567890.123'	Error. Integration Service skips this row.

## TO\_FLOAT

문자열 또는 숫자 값을 배정밀도 부동 소수점 숫자(배정밀도 데이터 유형)로 변환합니다. TO\_FLOAT는 선행 공백을 무시합니다.

### 구문

```
TO_FLOAT( value )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	문자열 또는 숫자 데이터 유형이어야 합니다. 배정밀도 값으로 변환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

### 반환 값

배정밀도 값.

이 함수에 전달된 값이 NULL인 경우 NULL이 반환됩니다.

포트의 값이 비어 있거나 숫자가 아닌 문자인 경우 0이 반환됩니다.

예

이 식은 포트 IN\_TAX의 값을 사용합니다.

```
TO_FLOAT( IN_TAX )
```

IN_TAX	RETURN VALUE
'15.6789'	15.6789
'60.2'	60.2
'118.348'	118.348
NULL	NULL
'A12.3Grove'	0

## TO\_INTEGER

문자열 또는 숫자 값을 정수로 변환합니다. TO\_INTEGER 구문에는 숫자를 가장 가까운 정수로 반올림하거나 소수부를 자르는 데 사용할 수 있는 선택적 인수가 있습니다. TO\_INTEGER는 선행 공백을 무시합니다.

구문

```
TO_INTEGER( value [, flag] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
값	필수	문자열 또는 숫자 데이터 유형. 정수로 변환할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
플래그	선택 사항	소수부를 자를지, 아니면 반올림할지를 지정합니다. 플래그는 정수 리터럴이거나 상수 TRUE 또는 FALSE여야 합니다. 플래그가 TRUE이거나 0이 아닌 숫자인 경우 소수부가 잘립니다. 플래그가 FALSE 또는 0이거나 사용자가 이 인수를 생략한 경우 값이 가까운 정수로 반올림됩니다.

반환 값

정수.

함수에 전달된 값이 NULL인 경우 NULL이 반환됩니다.

함수에 전달된 값에 영숫자 문자가 포함되는 경우 0이 반환됩니다.

## 예

다음 식은 포트 IN\_TAX의 값을 사용합니다. 변환으로 인해 숫자 오버플로우가 발생할 경우 PowerCenter 통합 서비스가 오류를 표시합니다.

TO\_INTEGER( IN\_TAX, TRUE )

IN_TAX	RETURN VALUE
'15.6789'	15
'60.2'	60
'118.348'	118
'5,000,000,000'	<i>Error. Integration Service skips this row.</i>
NULL	NULL
'A12.3Grove'	0
' 123.87'	123
'-15.6789'	-15
'-15.23'	-15

TO\_INTEGER( IN\_TAX, FALSE)

IN_TAX	RETURN VALUE
'15.6789'	16
'60.2'	60
'118.348'	118
'5,000,000,000'	<i>Error. Integration Service skips this row.</i>
NULL	NULL
'A12.3Grove'	0
' 123.87'	124
'-15.6789'	-16
'-15.23'	-15

## TRUNC(날짜)

날짜를 특정 연도, 월, 일, 시간, 분, 초, 밀리초 또는 마이크로초까지 자릅니다. TRUNC를 사용하여 숫자를 자를 수도 있습니다.

다음 날짜 부분을 잘라낼 수 있습니다.

- **연도.** 날짜의 연도 부분을 잘라내는 경우 입력 연도의 1월 1일이 00:00:00.000000000으로 설정된 시간과 함께 반환됩니다. 예를 들어 다음 식은 1/1/1997 00:00:00.000000000을 반환합니다.

```
TRUNC(12/1/1997 3:10:15, 'YY')
```

- **월.** 날짜의 월 부분을 잘라내는 경우 해당 월의 1일이 00:00:00.000000000으로 설정된 시간과 함께 반환됩니다. 예를 들어 다음 식은 4/1/1997 00:00:00.000000000을 반환합니다.

```
TRUNC(4/15/1997 12:15:00, 'MM')
```

- **일.** 날짜의 일 부분을 잘라내는 경우 날짜가 00:00:00.000000000으로 설정된 시간과 함께 반환됩니다. 예를 들어 다음 식은 6/13/1997 00:00:00.000000000을 반환합니다.

```
TRUNC(6/13/1997 2:30:45, 'DD')
```

- **시간.** 날짜의 시간 부분을 잘라내는 경우 분, 초 및 하위 초가 0으로 설정된 날짜가 반환됩니다. 예를 들어 다음 식은 4/1/1997 11:00:00.000000000을 반환합니다.

```
TRUNC(4/1/1997 11:29:35, 'HH')
```

- **분.** 날짜의 분 부분을 잘라내는 경우 초 및 하위 초가 0으로 설정된 날짜가 반환됩니다. 예를 들어 다음 식은 5/22/1997 10:15:00.000000000을 반환합니다.

```
TRUNC(5/22/1997 10:15:29, 'MI')
```

- **초.** 날짜의 초 부분을 잘라내는 경우 밀리초가 0으로 설정된 날짜가 반환됩니다. 예를 들어 다음 식은 5/22/1997 10:15:29.000000000을 반환합니다.

```
TRUNC(5/22/1997 10:15:29.135, 'SS')
```

- **밀리초.** 날짜의 밀리초 부분을 잘라내는 경우 마이크로초가 0으로 설정된 날짜가 반환됩니다. 예를 들어 다음 식은 5/22/1997 10:15:30.135000000을 반환합니다.

```
TRUNC(5/22/1997 10:15:30.135235, 'MS')
```

- **마이크로초.** 날짜의 마이크로초 부분을 잘라내는 경우 나노초가 0으로 설정된 날짜가 반환됩니다. 예를 들어 다음 식은 5/22/1997 10:15:30.135235000을 반환합니다.

```
TRUNC(5/22/1997 10:15:29.135235478, 'US')
```

## 구문

```
TRUNC( date [,format] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
날짜	필수	날짜/시간 데이터 유형. 잘라낼 날짜 값입니다. 날짜로 평가되는 유효한 모든 변환 식을 입력할 수 있습니다.
형식	선택 사항	유효한 형식 문자열을 입력합니다. 형식 문자열은 대/소문자를 구분하지 않습니다. 형식 문자열을 생략할 경우 함수가 날짜의 시간 부분을 잘라내고 00:00:00.000000000으로 설정합니다.

## 반환 값

날짜.

함수에 전달된 값이 NULL인 경우 NULL입니다.

예

다음 식은 DATE\_SHIPPED 포트의 날짜에서 연도 부분을 잘라냅니다.

```
TRUNC( DATE_SHIPPED, 'Y' )  
TRUNC( DATE_SHIPPED, 'YY' )  
TRUNC( DATE_SHIPPED, 'YYY' )  
TRUNC( DATE_SHIPPED, 'YYYY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 00:00:00.000000000
Apr 19 1998 1:31:20PM	Jan 1 1998 00:00:00.000000000
Jun 20 1998 3:50:04AM	Jan 1 1998 00:00:00.000000000
Dec 20 1998 3:29:55PM	Jan 1 1998 00:00:00.000000000
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 월 부분을 잘라냅니다.

```
TRUNC( DATE_SHIPPED, 'MM' )  
TRUNC( DATE_SHIPPED, 'MON' )  
TRUNC( DATE_SHIPPED, 'MONTH' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 00:00:00.000000000
Apr 19 1998 1:31:20PM	Apr 1 1998 00:00:00.000000000
Jun 20 1998 3:50:04AM	Jun 1 1998 00:00:00.000000000
Dec 20 1998 3:29:55PM	Dec 1 1998 00:00:00.000000000
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 일 부분을 잘라냅니다.

```
TRUNC( DATE_SHIPPED, 'D' )  
TRUNC( DATE_SHIPPED, 'DD' )  
TRUNC( DATE_SHIPPED, 'DDD' )  
TRUNC( DATE_SHIPPED, 'DY' )  
TRUNC( DATE_SHIPPED, 'DAY' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 00:00:00.000000000
Apr 19 1998 1:31:20PM	Apr 19 1998 00:00:00.000000000
Jun 20 1998 3:50:04AM	Jun 20 1998 00:00:00.000000000

DATE_SHIPPED	RETURN VALUE
Dec 20 1998 3:29:55PM	Dec 20 1998 00:00:00.000000000
Dec 31 1998 11:59:59PM	Dec 31 1998 00:00:00.000000000
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 시간 부분을 잘라냅니다.

```
TRUNC( DATE_SHIPPED, 'HH' )
TRUNC( DATE_SHIPPED, 'HH12' )
TRUNC( DATE_SHIPPED, 'HH24' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:31AM	Jan 15 1998 02:00:00.000000000
Apr 19 1998 1:31:20PM	Apr 19 1998 13:00:00.000000000
Jun 20 1998 3:50:04AM	Jun 20 1998 03:00:00.000000000
Dec 20 1998 3:29:55PM	Dec 20 1998 15:00:00.000000000
Dec 31 1998 11:59:59PM	Dec 31 1998 23:00:00.000000000
NULL	NULL

다음 식은 DATE\_SHIPPED 포트의 각 날짜에서 분 부분을 잘라냅니다.

```
TRUNC( DATE_SHIPPED, 'MI' )
```

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 02:10:00.000000000
Apr 19 1998 1:31:20PM	Apr 19 1998 13:31:00.000000000
Jun 20 1998 3:50:04AM	Jun 20 1998 03:50:00.000000000
Dec 20 1998 3:29:55PM	Dec 20 1998 15:29:00.000000000
Dec 31 1998 11:59:59PM	Dec 31 1998 23:59:00.000000000
NULL	NULL

## TRUNC(숫자)

숫자를 특정 자릿수까지 자릅니다. TRUNC를 사용하여 날짜를 자를 수도 있습니다.

구문

```
TRUNC( numeric_value [, precision] )
```



예

다음 식은 Price 포트의 값을 잘라냅니다.

TRUNC( PRICE, 3 )

PRICE	RETURN VALUE
12.9995	12.999
-18.8652	-18.865
56.9563	56.956
15.9928	15.992
NULL	NULL

TRUNC( PRICE, -1 )

PRICE	RETURN VALUE
12.99	10.0
-187.86	-180.0
56.95	50.0
1235.99	1230.0
NULL	NULL

TRUNC( PRICE )

PRICE	RETURN VALUE
12.99	12.0
-18.99	-18.0
56.95	56.0
15.99	15.0
NULL	NULL

## UPPER

문자열의 소문자를 대문자로 변환합니다.

구문

UPPER( *string* )



다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>문자열</i>	필수	문자열 데이터 유형. 대문자 텍스트로 변경할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

대문자 문자열. 데이터에 다중 바이트 문자가 포함되는 경우 반환 값은 PowerCenter 통합 서비스의 코드 페이지 및 데이터 이동 모드에 따라 다릅니다.

함수에 전달된 값이 NULL인 경우 NULL입니다.

## 예

다음 식은 FIRST\_NAME 포트의 모든 이름을 대문자로 변경합니다.

```
UPPER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
Ramona	RAMONA
NULL	NULL
THOMAS	THOMAS
PierRe	PIERRE
Bernice	BERNICE

## VARIANCE

전달한 값의 분산을 반환합니다. VARIANCE는 통계 데이터를 분석하는 데 사용됩니다. VARIANCE 안에는 다른 집계 함수 하나만 중첩할 수 있으며 중첩된 함수는 숫자 데이터 유형을 반환해야 합니다.

## 구문

```
VARIANCE( numeric_value [, filter_condition ] )
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/ 선택 사항	설명
<i>numeric_value</i>	필수	숫자 데이터 유형. 분산을 계산할 값을 전달합니다. 유효한 모든 변환 식을 입력할 수 있습니다.
<i>filter_condition</i>	선택 사항	검색에서 행을 제한합니다. 필터 조건은 숫자 값이거나 TRUE, FALSE 또는 NULL로 평가되어야 합니다. 유효한 모든 변환 식을 입력할 수 있습니다.

## 반환 값

배정밀도 값.

함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우(예: *filter\_condition*이 모든 행에 대해 FALSE 또는 NULL로 평가된 경우) NULL이 반환됩니다.

## Null

단일 값이 NULL인 경우 무시됩니다. 그러나 함수에 전달된 모든 값이 NULL이거나 행이 선택되지 않은 경우 NULL이 반환됩니다.

**참고:** 기본적으로 PowerCenter 통합 서비스는 집계 함수의 Null 값을 NULL로 처리합니다. 전체 포트 또는 그룹의 Null 값을 전달하는 경우 이 함수는 NULL을 반환합니다. 그러나 PowerCenter 통합 서비스를 구성할 때 집계 함수의 Null 값을 처리하는 방식을 선택할 수 있습니다. Null 값을 집계 함수에서 0으로 처리하거나 NULL로 처리할 수 있습니다.

## 그룹 기준

VARIANCE는 변환에 정의된 그룹 기준 포트에 따라 값을 그룹화하여 각 그룹에 대한 하나의 결과를 반환합니다.

그룹 기준 포트가 없는 경우 VARIANCE 함수는 모든 행을 하나의 그룹으로 처리하고 하나의 값을 반환합니다.

## 예

다음 식은 TOTAL\_SALES 포트에 있는 모든 행의 분산을 계산합니다.

```
VARIANCE( TOTAL_SALES )
```

### TOTAL\_SALES

2198.0

2256.0

3001.0

NULL

8953.0

**RETURN VALUE:** 10592444.6666667

# 사용자 지정 함수 작성

## 사용자 지정 함수 작성 개요

사용자 지정 함수는 PowerCenter 함수의 라이브러리를 확대합니다. 사용자 지정 함수는 변환 및 워크플로우 식에 사용할 목적으로 작성하는 함수로, 사용자 지정 함수 API를 사용하여 PowerCenter 외부에서 작성합니다. 사용자 지정 함수 API를 사용하려면 Informatica Developer 커뮤니티 웹 사이트에서 Informatica 개발 플랫폼을 설치해야 합니다.

사용자 지정 함수 API는 C 프로그래밍 언어를 사용합니다. 사용자 지정 함수를 다른 사용자와 공유할 수 있습니다. 함수를 리포지토리에 추가하여 PowerCenter 변환 언어 함수처럼 사용할 수 있습니다.

이 장에는 푸시다운 최적화를 통해 사용자 지정 함수를 작성하고 사용하는 방법을 보여 주는 샘플 함수가 포함되어 있습니다. 이 장에 포함된 단계는 ECHO 함수를 작성하는 단계입니다. 이 함수는 인수를 입력으로 가져와 입력 값을 사용자에게 반환합니다. ECHO 함수의 샘플 코드는 Informatica 개발 플랫폼 설치의 \CustomFunctionAPI\samples\echo 디렉터리에 있습니다.

복잡한 샘플 사용자 지정 함수도 추가로 볼 수 있습니다. SampleLoanPayment 사용자 지정 함수에는 C를 통해 사용할 수 없는 함수가 포함됩니다. SampleLoanPayment는 Informatica 개발 플랫폼 설치의 \CustomFunctionAPI\samples\loanpayment 디렉터리에 있습니다.

## 사용자 지정 함수 작성 단계

사용자 지정 함수를 작성하려면 다음 단계를 완료합니다.

1. **리포지토리 ID 특성을 가져옵니다.** 리포지토리 ID 특성을 가져와 리포지토리 플러그 인에 포함합니다.
2. **헤더 파일을 작성합니다.** 헤더 파일에 하나 이상의 사용자 지정 함수를 정의합니다.
3. **구현 파일을 작성합니다.** 구현 파일에 하나 이상의 사용자 지정 함수를 정의합니다.
4. **모듈을 작성합니다.** 모듈을 작성하여 DLL 및 공유 라이브러리를 작성합니다.
5. **리포지토리 플러그 인 파일을 작성합니다.** 사용자 지정 함수에 대한 메타데이터를 정의합니다.
6. **사용자 지정 함수를 테스트합니다.** 사용자 지정 함수를 설치하고 매핑 및 워크플로우에 사용하여 확인합니다.

## 사용자 지정 함수 설치

사용자 지정 함수를 사용하려면 해당 함수를 PowerCenter 환경에 추가해야 합니다.

관련 항목:

- [“사용자 지정 함수 설치” 페이지 239](#)

## 1단계. 리포지토리 ID 특성 가져오기

사용자 지정 함수를 개발하기 전에 사용자 지정 함수 리포지토리 플러그 인에 대한 리포지토리 ID 특성을 결정해야 합니다. 플러그 인 메타데이터를 정의할 때 리포지토리 ID 특성을 사용하여 플러그 인을 식별할 수 있습니다.

리포지토리 ID 특성을 가져오려면 다음 태스크 중 하나를 수행합니다.

- 사용자 지정 함수를 조직 외부에 배포하는 경우 Informatica에 문의하십시오. Informatica가 각 플러그 인에 고유한 리포지토리 ID 특성을 할당합니다. 다른 공급업체와 충돌하는 리포지토리 ID 특성은 유효하지 않습니다. 리포지토리 ID 특성을 가져오려면 <https://community.informatica.com/community/marketplace/repositoryidattributes>을(를) 방문하고 **제출**을 클릭합니다.
- 사용자 지정 함수를 조직 안에서만 사용하는 경우 Informatica에 문의하지 않고 리포지토리 ID 특성을 정의할 수 있습니다. 조직에서 PowerCenter의 다른 플러그 인과 함께 사용할 플러그 인을 개발하려면 각 플러그 인에 대한 리포지토리 ID 특성에 고유한 값을 할당해야 합니다.

다음 테이블은 고유한 값으로 플러그 인을 정의해야 하는 XML 특성을 보여 줍니다.

리포지토리 ID 특성	설명
플러그 인 ID	플러그 인의 ID를 식별합니다. 이 값은 PLUGIN 요소의 ID 특성에 해당합니다.
공급업체 ID	플러그 인을 개발한 공급업체를 식별합니다. 이 값은 PLUGIN 요소의 VENDORID 특성에 해당합니다.
함수 그룹 ID	함수 그룹의 ID를 식별합니다. 이 값은 FUNCTION_GROUP 요소의 ID 특성에 해당합니다.
함수 ID	함수의 ID를 식별합니다. 이 값은 FUNCTION 요소의 ID 특성에 해당합니다.

**참고:** 서로 충돌하는 리포지토리 ID 특성은 유효하지 않습니다.

관련 항목:

- [“PLUGIN 요소” 페이지 235](#)
- [“FUNCTION\\_GROUP 요소” 페이지 236](#)
- [“FUNCTION 요소” 페이지 236](#)

## 2단계. 헤더 파일 작성

C를 사용하여 모든 함수를 선언하는 헤더 파일을 작성합니다. 하나의 헤더 파일을 하나 이상의 사용자 지정 함수에 사용합니다.

다음 예는 ECHO 사용자 지정 함수에 대한 **echo.h** 헤더 파일을 보여 줍니다.

```
#ifndef __ECHO_PLUGIN_HPP
#define __ECHO_PLUGIN_HPP

#if defined(WIN32)
    #if defined(SAMPLE_EXPR_EXPORTS)
        #define SAMPLE_EXPR_SPEC __declspec(dllexport)
    #else
        #define SAMPLE_EXPR_SPEC __declspec(dllimport)
    #endif
#else
    #define SAMPLE_EXPR_SPEC
#endif

// method to get description of Echo function
extern "C" SAMPLE_EXPR_SPEC IUNICHAR * getDescriptionEcho(IUNICHAR* ns, IUNICHAR* sFuncName);

// method to get prototype of Echo function
extern "C" SAMPLE_EXPR_SPEC IUNICHAR * getPrototypeEcho(IUNICHAR* ns, IUNICHAR* sFuncName);

// method to validate usage of Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS validateFunctionEcho(IUNICHAR* ns, IUNICHAR* sFuncName,
    IUINT32 numArgs, INFA_EXPR_OPD_METADATA** inputArgList,
    INFA_EXPR_OPD_METADATA* retValue);

//method to generate SQL code for pushdown optimization
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS pushdownFunctionEcho(IUNICHAR* sNameSpace,
    IUNICHAR* sFuncName,
```

```

        IUINT32 numArgs,
        INFA_EXPR_OPD_METADATA** inputArgList,
        EDatabaseType dbType,
        EPushdownMode pushdownMode,
        IUNICHAR** sGenSql);

// method to process row for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_ROWSTATUS processRowEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE
*fnInstance, IUNICHAR **errMsg);

// method to do module level initialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS moduleInitEcho(INFA_EXPR_MODULE_HANDLE *modHandle);

// method to do module level deinitialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS moduleDeinitEcho(INFA_EXPR_MODULE_HANDLE *modHandle);

// method to do function level initialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS functionInitEcho(INFA_EXPR_FUNCTION_HANDLE *funHandle);

// method to do function level deinitialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS functionDeinitEcho(INFA_EXPR_FUNCTION_HANDLE *funHandle);

// method to do function instance level initialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS functionInstInitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE
*funInstHandle);

// method to do function instance level deinitialization for Echo function
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
functionInstDeinitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *funInstHandle);

/**
    These are all plugin callbacks, which have been implemented to get various module,
    function level interfaces
*/
// method to get plugin version
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS INFA_EXPR_GetPluginVersion(INFA_VERSION* sdkVersion,
INFA_VERSION* pluginVersion);

// method to delete the string allocated by this plugin. used for deleting the error
// messages
extern "C" SAMPLE_EXPR_SPEC void INFA_EXPR_DestroyString(IUNICHAR *);

// method to get validation interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS INFA_EXPR_ValidateGetUserInterface( IUNICHAR* ns,
IUNICHAR* sFuncName, INFA_EXPR_VALIDATE_METHODS* functions);

// method to get module interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS INFA_EXPR_ModuleGetUserInterface(INFA_EXPR_LIB_METHODS*
functions);

// method to get function interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS INFA_EXPR_FunctionGetUserInterface(IUNICHAR*
nameSpaceName, IUNICHAR* functionName, INFA_EXPR_FUNCTION_METHODS* functions);

// method to get function instance interfaces
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS INFA_EXPR_FunctionInstanceGetUserInterface(IUNICHAR*
nameSpaceName, IUNICHAR* functionName, INFA_EXPR_FUNCTION_INSTANCE_METHODS* functions);

#endif

```

### 3단계. 구현 파일 작성

구현 파일에는 사용자 지정 함수를 작성할 때 사용하는 함수의 정의가 포함됩니다. C를 사용하여 구현 파일을 작성합니다. 하나의 구현 파일을 하나 이상의 사용자 지정 함수에 사용할 수 있습니다. 하나의 구현 파일을 사용하여 사용자 지정 함수의 유효성 검사 함수와 런타임 함수를 정의할 수도 있습니다.

다음 예는 ECHO 사용자 지정 함수에 대한 `echo.c` 구현 파일을 보여 줍니다.

```
/******  
 * ECHO function Procedure File  
 *  
 * This file contains code that creates the ECHO function, which the  
 * Integration Service calls during a workflow.  
 *****/  
  
/* Informatica ECHO function example developed using the Custom Function  
 * API.  
  
 * Filename: Echo.c  
  
 * An example of a custom function developed using PowerCenter  
 *  
 * The purpose of the ECHO function is to return the input value to the user.  
 *  
 */  
  
/******  
                Includes  
 *****/  
#include <stdio.h>  
#include <string.h>  
#include "sdkexpr/exprsdk.h"  
  
#define SAMPLE_EXPR_EXPORTS  
#include "SampleExprPlugin.hpp"  
  
static IUNICHAR ECHO_STR[80];  
  
/******  
                Functions  
 *****/  
  
/******  
    Function: INFA_EXPR_GetPluginVersion  
  
Description: Defines the version of the plug-in. It must be the same as the  
Custom Function API version. Returns ISUCCESS if the plug-in version  
matches the Custom Function API version. Otherwise, returns IFailure.  
  
Input:  sdkVersion - Current version of the Custom Function API.  
Output: pluginVersion - Set the version of the plug-in.  
Remarks: Custom Function API checks for compatibility between itself and the  
plug-in version.  
 *****/  
  
extern "C" SAMPLE_EXPR_SPEC  
INFA_EXPR_STATUS INFA_EXPR_GetPluginVersion(INFA_VERSION* sdkVersion, INFA_VERSION* pluginVersion)  
{
```

```

    pluginVersion->m_major = 1;
    pluginVersion->m_minor = 0;
    pluginVersion->m_patch = 0;

    INFA_EXPR_STATUS retStatus;
    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

/*****
    Function: INFA_EXPR_DestroyString

Description: Destroys all strings the plug-in returns. For example, it
destroys error messages or the return value of other function calls, such
as getFunctionDescription. Returns no value.

Input: The pointer to the allocated string.
Output: N/A
Remarks: Frees the memory to avoid issues with multiple heaps.
*****/

extern "C" SAMPLE_EXPR_SPEC void INFA_EXPR_DestroyString(IUNICHAR *strToDelete)
{
    delete [] strToDelete;
}

/*****
    Function: INFA_EXPR_ValidateGetUserInterface

Description: Returns function pointers to the validation functions. Returns
ISUCCESS when the plug-in implemented the function. Returns IFailure
when the plug-in did not implement the function or another error occurred.

Input: Namespace and name of function.
Output: Functions. The plug-in needs to set various function pointers.
Remarks: Check the namespace and function name for validity. Set the various
function pointers appropriately.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_ValidateGetUserInterface(IUNICHAR* ns, IUNICHAR* sFuncName,
    INFA_EXPR_VALIDATE_METHODS* functions)
{
    INFA_EXPR_STATUS retStatus;
    retStatus.errMsg = NULL;

    // check function name is not null
    if (!sFuncName)
    {
        retStatus.status = IFailure;
        return retStatus;
    }

    // set the appropriate function pointers
    functions->validateFunction = validateFunctionEcho;
    functions->getFunctionDescription = getDescriptionEcho;
    functions->getFunctionPrototype = getPrototypeEcho;
    functions->pushdownFunction = pushdownFunctionEcho;

```

```

        retStatus.status = ISUCCESS;
        return retStatus;
    }

/*****
    Function: INFA_EXPR_ModuleGetUserInterface

Description: Sets the function pointers for module-level interaction.
Returns ISUCCESS when functions pointers are set appropriately. Otherwise,
returns IFAILURE.

Input: N/A
Output: Functions. The plug-in needs to set various function pointers.
Remarks: Set the module init/deinit function pointers.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_ModuleGetUserInterface(INFA_EXPR_LIB_METHODS* functions)
{
    functions->module_init = moduleInitEcho;
    functions->module_deinit = moduleDeinitEcho;

    INFA_EXPR_STATUS retStatus;
    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

/*****
    Function: INFA_EXPR_FunctionGetUserInterface

Description: Sets the function pointers for function-level interaction.
PowerCenter calls this function for every custom function this library
implements. Returns ISUCCESS when The plugin implements this function and
sets the function pointers correctly. Otherwise, returns IFAILURE.

Input: Namespace and name of function.
Output: Functions. The plug-in needs to set function pointers for function
init/deinit.
Remarks: Set the function init/deinit function pointers.
*****/

extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_FunctionGetUserInterface(IUNICHAR* nameSpaceName,
                                                         IUNICHAR* functionName,
                                                         INFA_EXPR_FUNCTION_METHODS* functions)
{
    functions->function_init = functionInitEcho;
    functions->function_deinit = functionDeinitEcho;

    INFA_EXPR_STATUS retStatus;
    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

/*****
    Function: INFA_EXPR_FunctionInstanceGetUserInterface

```



Description: Sets the function pointers for function instance-level interaction. PowerCenter calls this function for every custom function this library implements. Returns ISUCCESS when The plugin implements this function and sets the function pointers correctly. Otherwise, returns IFailure.

Input: Namespace and name of function.

Output: Functions. The plug-in needs to set function pointers for instance init/deinit/processrow.

Remarks: Set the function instance init/deinit/processrow function pointers.  
 \*\*\*\*\*/

```
extern "C" SAMPLE_EXPR_SPEC
    INFA_EXPR_STATUS INFA_EXPR_FunctionInstanceGetUserInterface(IUNICHAR* nameSpaceName,
                                                                IUNICHAR* functionName,
                                                                INFA_EXPR_FUNCTION_INSTANCE_METHODS*
functions)
{
    functions->fnInstance_init = functionInstInitEcho;
    functions->fnInstance_processRow = processRowEcho;
    functions->fnInstance_deinit = functionInstDeinitEcho;

    INFA_EXPR_STATUS retStatus;
    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}
```

\*\*\*\*\*/  
 Function: INFA\_EXPR\_getDescriptionEcho

Description: Gets the description of the ECHO function. It calls destroyString to delete the arguments from memory when usage is complete. The return value must be a null-terminated string.

Input: Namespace and name of function.

Output: Description of the function.

Remarks: Returns the description of function. The Custom Functions API calls destroy string to free the allocated memory.  
 \*\*\*\*\*/

```
extern "C" SAMPLE_EXPR_SPEC
    IUNICHAR * getDescriptionEcho(IUNICHAR* ns, IUNICHAR* sFuncName)
{
    static IUNICHAR *uniDesc = NULL;
    const char *description = "Echoes the input";

    if (uniDesc)
        return uniDesc;

    int i, len;
    len = strlen(description);
    uniDesc = new IUNICHAR[2*len+2];
    for (i=0; i<len; i++)
    {
        uniDesc[i] = description[i];
    }
    uniDesc[i] = 0;
    return uniDesc;
}
```

```
/******
```

Function: INFA\_EXPR\_getPrototypeEcho

Description: Gets the arguments of the ECHO function in the Expression Editor. It calls destroyString to delete the arguments from memory when usage is complete. The return value must be a null-terminated string. The function returns NULL if there is no value for the arguments.

Input: Namespace and name of the function.

Output: Prototype of the function

Remarks: Returns the prototype of function. The Custom Functions API calls destroy string to free the allocated memory.

```
*****/
```

```
extern "C" SAMPLE_EXPR_SPEC
IUNICHAR * getPrototypeEcho(IUNICHAR* ns, IUNICHAR* sFuncName)
{
    static IUNICHAR *uniProt = NULL;
    const char *prototype = "Echo(x), where x can be any type, returns x";

    if (uniProt)
        return uniProt;

    int i, len;
    len = strlen(prototype);
    uniProt = new IUNICHAR[2*len+2];
    for (i=0; i<len; i++)
    {
        uniProt[i] = prototype[i];
    }
    uniProt[i] = 0;
    return uniProt;
}
```

```
/******
```

Function: validateFunctionEcho

Description: Validates the arguments in the ECHO function. Provides the name, datatype, precision, and scale of the arguments in the ECHO function. Provides the datatype of the return value of the ECHO function. PowerCenter calls this function once for each instance of the ECHO function used in a mapping or workflow. Returns ISUCCESS when function usage is valid as per the syntax.

The ECHO function takes exactly one argument of any datatype. The return datatype is the same as the input datatype, because the function echoes the input. Otherwise, returns IFailure.

Input: Namespace and name of the function, the number of arguments being passed, and the metadata (datatype, scale, precision) of each argument.

Output: retValue. Set the metadata for return type.

Remarks: Called by the Custom Functions API to validate the usage of the function and the input argument metadata to be passed. The plug-in needs to verify the number of arguments for this function, the expected metadata for each argument, etc. The plug-in can optionally change the expected datatype of the input arguments. The plug-in needs to set the return type metadata. The plugin can specify if the return value of this function is constant, depending on whether or not all input arguments are constant.

```

*****/

extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_STATUS validateFunctionEcho(IUNICHAR* ns, IUNICHAR* sFuncName,
                                      IUINT32 numArgs,
                                      INFA_EXPR_OPD_METADATA** inputArgList,
                                      INFA_EXPR_OPD_METADATA* retValue)
{
    INFA_EXPR_STATUS exprStatus;

    // Check number of arguments.
    if (numArgs != 1)
    {
        static const char *err = "Echo function takes one argument.";
        IUNICHAR *errMsg = NULL;

        unsigned int len = strlen(err);
        errMsg = new IUNICHAR[2*len+2];
        unsigned int i;
        for (i=0; i<len; i++)
        {
            errMsg[i] = err[i];
        }
        errMsg[i] = 0;

        exprStatus.status = IFailure;
        exprStatus.errMsg = errMsg;
        return exprStatus;
    }

    // This is an echo function.
    // It returns the input value.
    retValue->datatype = inputArgList[0]->datatype;
    retValue->precision = inputArgList[0]->precision;
    retValue->scale = inputArgList[0]->scale;

    // If the input value is constant,
    // the return value is also constant.
    if (inputArgList[0]->isValueConstant)
        retValue->isValueConstant = ITRUE;
    else
        retValue->isValueConstant = IFALSE;

    exprStatus.status = ISUCCESS;
    return exprStatus;
}

```

```

/*****
Function: processRowEcho

```

Description: Called when an input row is available to an ECHO function instance. The data for the input arguments of the ECHO function is bound and accessed through fnInstance-inputOPDHandles. Set the data, length, and indicator for the output and return ports in fnInstance->retHandle. PowerCenter calls the function-level initialization function before calling this function.

Returns INFA\_ROWSUCCESS when the function successfully processes the row of

data. Returns INFA\_ROWERROR when the function encounters an error for the row of data. The Integration Service increments the internal error count. Only returns this value when the data access mode is row. Returns INFA\_FATALERROR when the function encounters a fatal error for the row of data or the block of data. The Integration Service fails the session.

Input: Function instance handle, which has the input data.

Output: return value

Remarks: The plug-in needs to get various input arguments from the function instance handle, perform calculations, and set the return value.

```

*****/

extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_ROWSTATUS processRowEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *fnInstance, IUNICHAR
**errMsg)
{
    INFA_EXPR_OPD_RUNTIME_HANDLE* arg1 = fnInstance->inputOPDHandles[0];
    INFA_EXPR_OPD_RUNTIME_HANDLE* retHandle = fnInstance->retHandle;

    // Check if the input argument has a null indicator.
    // If yes, the return value is also null.
    if (INFA_EXPR_GetIndicator(arg1) == INFA_EXPR_NULL_DATA)
    {
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_NULL_DATA);
        return INFA_EXPR_SUCCESS;
    }

    short sval;
    long lval;
    int ival;
    char *strval;
    IUNICHAR *ustrval;
    void *rawval;
    float fval;
    double dval;
    INFA_EXPR_DATE *infaDate = NULL;
    int len;

    // Depending on the datatype,
    // get the input argument
    // and set the same value in the return value.
    // Also, set the same indicator.
    switch (arg1->pOPDMetadata->datatype)
    {
        case eCTYPE_SHORT:
            sval = INFA_EXPR_GetShort(arg1);
            INFA_EXPR_SetShort(retHandle, sval);
            INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
            break;

        case eCTYPE_LONG:
        case eCTYPE_LONG64:
            lval = INFA_EXPR_GetLong(arg1);
            INFA_EXPR_SetLong(retHandle, lval);
            INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
            break;

        case eCTYPE_INT32:
            ival = INFA_EXPR_GetInt(arg1);

```

```

        INFA_EXPR_SetInt(retHandle, ival);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_CHAR:
        strval = INFA_EXPR_GetString(arg1);
        len = INFA_EXPR_GetLength(arg1);
        strcpy((char *)retHandle->pUserDefinedPtr, strval);
        INFA_EXPR_SetString(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetLength(retHandle, INFA_EXPR_GetLength(arg1));
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_RAW:
        rawval = INFA_EXPR_GetRaw(arg1);
        len = INFA_EXPR_GetLength(arg1);
        memcpy(retHandle->pUserDefinedPtr, rawval, len);
        INFA_EXPR_SetRaw(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetLength(retHandle, len);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_UNICHAR:
        ustrval = INFA_EXPR_GetUniString(arg1);
        len = INFA_EXPR_GetLength(arg1);
        memcpy(retHandle->pUserDefinedPtr, ustrval, 2*(len+1));
        INFA_EXPR_SetUniString(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetLength(retHandle, len);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_TIME:
        infaDate = INFA_EXPR_GetDate(arg1);
        *((INFA_EXPR_DATE *)retHandle->pUserDefinedPtr) = *infaDate;
        INFA_EXPR_SetDate(retHandle, retHandle->pUserDefinedPtr);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_FLOAT:
        fval = INFA_EXPR_GetFloat(arg1);
        INFA_EXPR_SetFloat(retHandle, fval);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    case eCTYPE_DOUBLE:
        dval = INFA_EXPR_GetDouble(arg1);
        INFA_EXPR_SetDouble(retHandle, dval);
        INFA_EXPR_SetIndicator(retHandle, INFA_EXPR_GetIndicator(arg1));
        break;

    default:
        return INFA_EXPR_ROWERROR;
        break;
}
return INFA_EXPR_SUCCESS;
}

/*****
Function: moduleInitEcho

```

Description: Called once for each module to initialize any global data structure in the function. Called before calling any function-level functions. Returns ISUCCESS when module initialization is successful. Otherwise, returns IFailure.

Input: module handle

Output: status

Remarks: The plug-in can optionally implement this method for one-time initialization.

\*\*\*\*\*/

```
extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_STATUS moduleInitEcho(INFA_EXPR_MODULE_HANDLE *modHandle)
{
    INFA_EXPR_STATUS exprStatus;

    // initialize the ECHO_STR
    const char *fnName = "Echo";
    int len = strlen(fnName);
    int i;
    for (i=0;i<len;i++)
        ECHO_STR[i] = fnName[i];

    exprStatus.status = ISUCCESS;
    return exprStatus;
}
```

\*\*\*\*\*

Function: moduleDeinitEcho

Description: Called once for each module to deinitialize any data structure in this function. Called after all function-level interactions are complete. Returns ISUCCESS when module deinitialization is successful. Otherwise, returns IFailure.

Input: module handle

Output: status

Remarks: The plug-in can optionally implement this method for one-time deinitialization.

\*\*\*\*\*/

```
extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_STATUS moduleDeinitEcho(INFA_EXPR_MODULE_HANDLE *modHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    return exprStatus;
}
```

\*\*\*\*\*

Function: functionInitEcho

Description: Called once for each custom function to initialize any structure related to the custom function. Module-level initialization function is called before this function. Returns ISUCCESS when function init is successful. Otherwise, returns IFailure.

Input: function handle

Output: status

Remarks: The plug-in can optionally implement this method for one-time function initialization.

```
*****/
extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_STATUS functionInitEcho(INFA_EXPR_FUNCTION_HANDLE *funHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    return exprStatus;
}

/*****
    Function: functionDeinitEcho
```

Description: Called once for each function level to deinitialize any structure related to the ECHO function. Returns ISUCCESS when function deinit is successful. Otherwise, returns IFailure.

Input: function handle

Output: status

Remarks: The plug-in can optionally implement this method for one-time function deinitialization.

```
*****/
extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_STATUS functionDeinitEcho(INFA_EXPR_FUNCTION_HANDLE *funHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    return exprStatus;
}

/*****
    Function: functionInstInitEcho
```

Description: Called once for each custom function instance to initialize any structure related to the an instance of the ECHO function. If there are two instances of ECHO in a mapping or workflow, PowerCenter calls this function twice. PowerCenter calls the module-level initialization function before calling this function. Returns ISUCCESS when function instance initialization is successful. Otherwise, returns IFailure.

Input: function instance handle

Output: status

Remarks: The plug-in can optionally implement this method for one-time function instance initialization.

```
*****/
extern "C" SAMPLE_EXPR_SPEC
INFA_EXPR_STATUS functionInstInitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *funInstHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;

    INFA_EXPR_OPD_RUNTIME_HANDLE *retHandle = funInstHandle->retHandle;

    // Allocate memory depending on the datatype.
```

```

    if (retHandle->pOPDMetadata->datatype == eCTYPE_CHAR)
        retHandle->pUserDefinedPtr = new char[retHandle->pOPDMetadata->precision+1];
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_UNICHAR)
        retHandle->pUserDefinedPtr = new IUNICHAR[retHandle->pOPDMetadata->precision+1];
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_RAW)
        retHandle->pUserDefinedPtr = new unsigned char[retHandle->pOPDMetadata->precision];
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_TIME)
        retHandle->pUserDefinedPtr = new INFA_EXPR_DATE();
    return exprStatus;
}

```

```

/*****
Function: functionInstDeinitEcho

```

Description: Called once for each function level during deinitialization.  
Can deinitialize any structure related to the ECHO function. Returns ISUCCESS  
when deinitialization is successful. Otherwise, returns IFailure.

Input: function instance handle

Output: status

Remarks: The plug-in can optionally implement this method for one-time  
function instance deinitialization.

```

*****/

```

```

extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS
functionInstDeinitEcho(INFA_EXPR_FUNCTION_INSTANCE_HANDLE *funInstHandle)
{
    INFA_EXPR_STATUS exprStatus;
    exprStatus.status = ISUCCESS;
    INFA_EXPR_OPD_RUNTIME_HANDLE *retHandle = funInstHandle->retHandle;

    if (retHandle->pOPDMetadata->datatype == eCTYPE_CHAR)
        delete [] (char *)retHandle->pUserDefinedPtr;
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_UNICHAR)
        delete [] (IUNICHAR *)retHandle->pUserDefinedPtr;
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_RAW)
        delete [] (unsigned char *)retHandle->pUserDefinedPtr;
    else if (retHandle->pOPDMetadata->datatype == eCTYPE_TIME)
        delete (INFA_EXPR_DATE *)retHandle->pUserDefinedPtr;
    return exprStatus;
}

```

```

/*****
Function: pushdownFunctionEcho

```

Description: Method to generate SQL code for pushdown optimization.

Input: Namespace and name of the function, the number of arguments being passed,  
and the metadata (datatype, scale, precision) of each argument, database type,  
Pushdown mode.

Output: Generated SQL.

Remarks: The plug-in can optionally implement this method to enable pushdown  
optimization.

```

*****/

```

```

//method to generate SQL code for pushdown optimization
extern "C" SAMPLE_EXPR_SPEC INFA_EXPR_STATUS pushdownFunctionEcho(IUNICHAR* sNameSpace,
    IUNICHAR* sFuncName,
    IUINT32 numArgs,

```



```

        INFA_EXPR_OPD_METADATA** inputArgList,
        EDatabaseType dbType,
        EPushdownMode pushdownMode,
        IUNICHAR** sGenSql)
{
    INFA_EXPR_STATUS retStatus;
    static const char *sql_str = "{1}";

    // Construct the SQL: "{1}"
    unsigned int len = strlen(sql_str);

    IUNICHAR *pGenSql = new IUNICHAR[len+1];
    unsigned int i;

    for (i=0; i<len; i++)
    {
        pGenSql[i] = sql_str[i];
    }

    pGenSql[len] = 0;

    // Return the generated SQL
    *sGenSql = pGenSql;

    retStatus.status = ISUCCESS;
    retStatus.errMsg = NULL;
    return retStatus;
}

```

## 4단계. 모듈 작성

Windows 또는 UNIX에 모듈을 작성할 수 있습니다. PowerCenter가 실행되는 각 플랫폼에 대한 모듈을 작성합니다. PowerCenter 클라이언트가 Windows에 상주하므로 Windows에 모듈을 작성해야 합니다. 통합 서비스를 호스트하는 노드에 따라 UNIX 또는 Linux에 모듈을 작성해야 할 수도 있습니다.

다음 테이블에는 모듈을 작성할 때 필요한 각 플랫폼의 라이브러리 파일 이름이 나열되어 있습니다.

플랫폼	모듈 파일 이름
Windows	<module_identifier>.dll
AIX	lib<module_identifier>.a
Linux	lib<module_identifier>.so
Solaris	lib<module_identifier>.so

이 모듈은 리포지토리 플러그 인 XML 파일에 선언합니다.

관련 항목:

- [“5단계. 리포지토리 플러그 인 파일 작성” 페이지 235](#)

## Windows 모듈 작성

Windows에서는 Microsoft Visual C++를 사용하여 모듈을 작성할 수 있습니다.

Windows에 모듈을 작성하려면 다음을 수행하십시오.

1. Visual C++를 시작합니다.
2. 파일 > 새로 만들기를 클릭합니다.
3. 새로 만들기 대화 상자에서 프로젝트 탭을 클릭하고 Win32 동적 연결 라이브러리 옵션을 선택합니다.
4. 위치를 입력합니다.

Echo 예에서는 다음 디렉터리를 입력합니다.

<Informatica Development Platform installation directory>\CustomFunctionAPI\samples\echo

5. 프로젝트 이름을 입력합니다.

사용자 지정 함수에 지정된 모듈 이름을 프로젝트 이름으로 사용해야 합니다. Echo 예에서는 EchoDemo를 입력합니다.

6. 확인을 클릭합니다.

Visual C++가 프로젝트 구성 요소를 정의하는 마법사를 작성합니다.

7. 마법사에서 빈 DLL 프로젝트를 선택하고 마침을 클릭합니다. 새 프로젝트 정보 대화 상자에서 확인을 클릭합니다.

Visual C++가 지정한 디렉터리에 프로젝트 파일을 작성합니다.

8. 프로젝트 > 프로젝트에 추가 > 파일을 클릭합니다.

9. 한 수준 위의 디렉터리를 탐색합니다. 작성한 프로시저 파일이 이 디렉터리에 포함됩니다. 모든 .c 파일을 선택하고 확인을 클릭합니다.

Echo 예에서는 Echo.c 파일을 추가합니다.

10. 프로젝트 > 설정을 클릭합니다.

11. C/C++ 탭을 클릭하고 범주 필드에서 전처리를 선택합니다.

12. 추가 포함 디렉터리 필드에 다음 경로를 입력하고 확인을 클릭합니다.

..; <Informatica Development Platform installation directory>\CustomFunctionAPI\samples\echo; ...

13. 빌드 > <module\_name>.dll 빌드를 클릭하거나 F7을 눌러 프로젝트를 작성합니다.

Visual C++가 DLL을 작성하고 프로젝트 디렉터리 아래의 디버그 또는 릴리스 디렉터리에 배치합니다.

## UNIX 모듈 작성

UNIX에서는 C 컴파일러를 사용하여 모듈을 작성할 수 있습니다.

UNIX에 모듈을 작성하려면 다음을 수행하십시오.

1. 환경 변수 INFA\_HOME을 PowerCenter 통합 서비스 설치 디렉터리로 설정합니다.

**참고:** INFA\_HOME 환경 변수에 대해 잘못된 디렉터리 경로를 지정하면 PowerCenter 통합 서비스가 시작되지 않습니다.

노드를 다시 시작하여 변경 사항을 적용합니다.

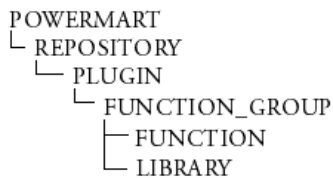
2. 다음 테이블의 명령을 입력하여 프로젝트를 만듭니다.

UNIX 버전	명령
AIX(32비트)	make -f makefile.aix
AIX(64비트)	make -f makefile.aix64
Linux	make -f makefile.linux
Solaris	make -f makefile.sol

## 5단계. 리포지토리 플러그 인 파일 작성

XML 파일을 작성하여 추가 사용자 지정 함수의 함수 메타데이터를 정의합니다. 플러그 인 XML 파일을 작성하거나 수정할 때는 플러그 인 DTD 파일의 구조를 사용합니다. 플러그 인 DTD 파일인 **plugin.dtd**는 **PowerCenter** 클라이언트 디렉터리에 저장됩니다. XML 파일을 작성할 수 있는 도구를 사용합니다. 리포지토리 플러그 인 파일을 작성하는 경우 파일의 새 이름을 입력합니다.

다음 그림은 **plugin.dtd**의 구조를 보여 줍니다.



## PLUGIN 요소

PLUGIN 요소는 작성할 플러그 인에 대한 메타데이터를 정의할 때 사용합니다. PLUGIN 요소에는 플러그 인 메타데이터를 고유하게 식별하는 특성이 있습니다.

다음 테이블은 PLUGIN 요소의 특성을 보여 줍니다.

특성	필수/선택 사항	설명
이름	필수	플러그 인의 이름입니다. 플러그 인 이름은 PowerCenter 리포지토리 서비스의 플러그 인 탭에 표시됩니다.
ID	필수	플러그 인의 ID입니다. 동일한 VENDORID로 개발된 플러그 인을 구분하는데 사용됩니다.
VENDORNAME	필수	공급업체의 이름입니다. 공급업체 이름은 PowerCenter 리포지토리 서비스의 플러그 인 탭에 표시됩니다.
VENDORID	필수	공급업체 ID입니다. 조직 외부에 배포할 사용자 지정 함수를 개발하는 경우 Informatica에 공급업체 ID를 문의해야 합니다. 자세한 내용은 <a href="#">“1단계. 리포지토리 ID 특성 가져오기” 페이지 219</a> 를 참조하십시오.

특성	필수/ 선택 사항	설명
DESCRIPTION	선택 사항	플러그 인의 설명입니다. 플러그 인 설명은 PowerCenter 리포지토리 서비스의 플러그 인 탭에 표시됩니다.
VERSION	필수	플러그 인의 버전입니다. 플러그 인 메타데이터에 대한 업데이트를 추적하는 데 사용됩니다.

## FUNCTION\_GROUP 요소

FUNCTION\_GROUP 요소는 사용자 지정 함수가 속하는 그룹을 정의할 때 사용합니다.

다음 테이블은 FUNCTION\_GROUP 요소의 특성을 보여 줍니다.

특성	필수/ 선택 사항	설명
이름	필수	정의할 사용자 지정 함수 그룹의 이름입니다. 함수 그룹 이름은 PowerCenter 리포지토리 서비스의 플러그 인 탭에 표시됩니다.
ID	필수	함수 그룹의 ID입니다. 조직 외부에 배포할 사용자 지정 함수를 개발하는 경우 Informatica에 함수 그룹 ID를 문의해야 합니다. 자세한 내용은 <a href="#">“1단계. 리포지토리 ID 특성 가져오기” 페이지 219</a> 를 참조하십시오. 함수 그룹 ID는 PowerCenter 리포지토리 서비스의 플러그 인 탭에 표시됩니다.
COMPONENTVERSION	필수	함수 그룹의 버전 번호입니다. 이 특성은 FUNCTION_GROUP 요소에 대한 업데이트를 추적합니다.
DESCRIPTION	선택 사항	함수 그룹의 설명입니다. 함수 그룹 설명은 PowerCenter 리포지토리 서비스의 플러그 인 탭에 표시됩니다.
NAMESPACE	필수	함수 그룹의 네임스페이스입니다. 사용자 지정 함수 및 네임스페이스는 식 편집기의 함수 탭에 개별 폴더로 표시됩니다. 네임스페이스는 대/소문자를 구분하지 않습니다. 네임스페이스 “infa”는 사용할 수 없습니다. 예약된 네임스페이스입니다. 또한 네임스페이스를 비워 둘 수도 없습니다.

## 네임스페이스 결정

작성하는 모든 함수에 하나의 네임스페이스를 선택할 수 있습니다. 그러나 다른 공급업체가 개발한 사용자 지정 함수의 네임스페이스와 충돌하는 네임스페이스는 사용할 수 없습니다. 그러므로 고유한 네임스페이스를 선택해야 합니다. 예를 들어 회사의 주식 기호와 같이 회사 이름과 관련된 네임스페이스를 선택할 수 있습니다.

## FUNCTION 요소

FUNCTION 요소는 사용자 지정 함수의 속성을 정의할 때 사용합니다.

다음 테이블은 FUNCTION 요소의 특성을 보여 줍니다.

특성	필수/ 선택 사항	설명
이름	필수	정의할 타사 함수의 이름입니다.
ID	필수	FUNCTION 요소의 ID입니다. 함수를 식별합니다. 조직 외부에 배포할 사용자 지정 함수를 개발하는 경우 Informatica에 함수 ID를 문의해야 합니다. 자세한 내용은 <a href="#">“1단계. 리포지토리 ID 특성 가져오기” 페이지 219</a> 를 참조하십시오.
FUNCTION_CATEGORY	선택 사항	정의할 함수의 범주입니다. 다음 범주 중 하나를 사용합니다. <ul style="list-style-type: none"> <li>- 문자</li> <li>- 변환</li> <li>- 데이터 정리</li> <li>- 날짜</li> <li>- 숫자</li> <li>- 지수</li> <li>- 특수</li> <li>- 테스트</li> </ul> 사용자 지정 함수는 식 편집기에서 이 범주 아래에 표시됩니다.

## LIBRARY 요소

LIBRARY 요소는 사용자 지정 함수의 컴파일된 공유 라이브러리를 지정할 때 사용합니다.

다음 테이블은 LIBRARY 요소의 특성을 보여 줍니다.

특성	필수/ 선택 사항	설명
이름	필수	컴파일된 공유 라이브러리의 수입입니다.
OSTYPE	필수	공유 라이브러리를 컴파일할 운영 체제입니다.
TYPE	필수	공유 라이브러리의 유형입니다. 다음 유형 중 하나를 지정합니다. <ul style="list-style-type: none"> <li>- VALIDATION. PowerCenter 클라이언트가 사용자 지정 함수 설명을 검색하고 함수 호출(예: 반환 유형 및 인수 수)의 유효성을 검사할 때 사용하는 라이브러리입니다.</li> <li>- SERVER. PowerCenter 통합 서비스가 함수 호출을 실행할 때 사용하는 라이브러리입니다.</li> </ul>

## 샘플 플러그 인 XML 파일

다음 예는 ECHO 사용자 지정 함수를 정의하는 리포지토리 플러그 인 파일을 보여 줍니다.

```
<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE POWERMART SYSTEM "plugin.dtd">
<POWERMART>
  <REPOSITORY CODEPAGE="us-ascii">
    <PLUGIN NAME="Echo" ID="506001" VENDORNAME="Informatica"
      VENDORID="1">
```

```

DESCRIPTION="Plugin for Expressions from Informatica">
<FUNCTION_GROUP ID="506002" NAME="INFA Function Group1"
  COMPONENTVERSION="1.0.0"
  DESCRIPTION="The functions group for my own Echo function"
  NAMESPACE="">
  <FUNCTION ID="506004" NAME="ECHO" FUNCTION_CATEGORY="Data Cleansing"/>
  <LIBRARY NAME="pmecho.dll" OSTYPE="NT" TYPE="VALIDATION"/>
  <LIBRARY NAME="libpmecho.sl" OSTYPE="HPUX" TYPE="VALIDATION"/>
  <LIBRARY NAME="libpmecho.so" OSTYPE="SOLARIS" TYPE="VALIDATION"/>
  <LIBRARY NAME="libpmecho.so" OSTYPE="LINUX" TYPE="VALIDATION"/>
  <LIBRARY NAME="libpmecho.a" OSTYPE="AIX" TYPE="VALIDATION"/>
  <LIBRARY NAME="pmecho.dll" OSTYPE="NT" TYPE="SERVER"/>
  <LIBRARY NAME="libpmecho.sl" OSTYPE="HPUX" TYPE="SERVER"/>
  <LIBRARY NAME="libpmecho.so" OSTYPE="SOLARIS" TYPE="SERVER"/>
  <LIBRARY NAME="libpmecho.so" OSTYPE="LINUX" TYPE="SERVER"/>
  <LIBRARY NAME="libpmecho.a" OSTYPE="AIX" TYPE="SERVER"/>
</FUNCTION_GROUP>
</PLUGIN>
</REPOSITORY>
</POWERMART>

```

## 6단계. 사용자 지정 함수 테스트

사용자 지정 함수를 개발하는 동안 해당 함수를 테스트할 수 있습니다. **PowerCenter**에서 사용자 지정 함수를 테스트하려면 다음 작업을 완료합니다.

- 리포지토리 플러그인 XML 파일의 유효성을 검사합니다.
- 식의 사용자 지정 함수가 정확한 데이터를 생성하는지 확인합니다.

사용자 지정 함수를 테스트하려면 **PowerCenter** 환경에 사용자 지정 함수를 설치해야 합니다.

### 리포지토리 플러그인 파일 유효성 검사

**PowerCenter** 리포지토리에 리포지토리 플러그인 파일을 등록하여 유효성을 검사할 수 있습니다. 플러그인 파일을 등록하면 연결된 DTD 파일인 **plugin.dtd**가 파일 구조의 유효성을 검사합니다. 플러그인 파일은 연결된 **plugin.dtd**의 구조를 따라야 합니다. **plugin.dtd**는 **PowerCenter** 클라이언트 디렉터리에 있습니다.

사용자 지정 함수를 개발할 때 함수의 헤더 및 구현 파일 작성을 마치기 전에 리포지토리 플러그인 파일을 작성하고 등록할 수 있습니다. 파일을 등록할 때는 플러그인 ID, 네임스페이스 및 함수 이름과 같은 사용자 지정 함수 메타데이터를 추가합니다. 이렇게 하면 이 정보가 리포지토리에 예약됩니다.

리포지토리 플러그인 파일을 등록한 후에 계속해서 사용자 지정 함수를 개발할 수 있습니다. 함수 개발을 마치면 리포지토리 플러그인 파일을 다시 등록하여 리포지토리의 사용자 지정 함수 메타데이터를 업데이트합니다.

리포지토리 플러그인을 등록한 후에 플러그인 메타데이터를 볼 수 있습니다.

## 플러그 인 메타데이터 세부 정보 보기

다음 테이블은 Informatica Administrator의 플러그 인 탭에 표시되는 메타데이터를 보여 줍니다.

리포지토리 서비스 특성	XML 요소 및 특성
이름	PLUGIN NAME
공급업체 이름	PLUGIN VENDORNAME
설명	PLUGIN DESCRIPTION
그룹 이름	FUNCTION_GROUP NAME
그룹 ID	FUNCTION_GROUP ID
그룹 설명	FUNCTION_GROUP DESCRIPTION

## 함수 정확도 확인

사용자 지정 함수의 정확도를 확인하려면 함수를 포함하는 식을 작성하고 매핑 및 워크플로우에 식을 포함합니다. 사용자 지정 함수의 정확도를 확인하려면 다음 단계를 완료합니다.

1. 테스트 데이터를 작성합니다.
2. 매핑을 생성하십시오.
3. 사용자 지정 함수를 매핑의 식에 추가합니다.
4. 매핑을 생성하십시오.
5. 디버거를 실행합니다(선택 사항). 또는 매핑에 대한 세션 및 워크플로우를 작성합니다.
6. 워크플로우를 실행합니다.
7. 결과를 봅니다.

## 사용자 지정 함수 설치

사용자 지정 함수를 설치하려면 다음 단계를 완료합니다.

1. 사용자 지정 함수 라이브러리를 **PowerCenter** 환경에 복사합니다.
2. 리포지토리 플러그 인을 등록합니다.

사용자 지정 함수를 설치한 후에 해당 함수를 변환 및 워크플로우 식에 사용합니다.

### 1단계. 사용자 지정 함수 라이브러리를 PowerCenter에 복사

사용자 지정 함수 라이브러리 및 리포지토리 플러그 인 XML 파일을 사용자 지정 함수 개발자 지침에 따라 **PowerCenter** 클라이언트 및 통합 서비스 디렉터리에 복사합니다.

그리드에 고가용성 또는 실행 세션이 있는 경우 라이브러리를 단일 위치에 배치하고 해당 위치를 필수 리소스로 정의합니다.

### 2단계. 플러그 인 등록

Administrator 도구에서 리포지토리 플러그 인 XML 파일을 등록합니다.

## 사용자 지정 함수를 포함하는 식 작성

사용자 지정 함수를 식에 추가할 수 있습니다. 식을 수동으로 작성할 때 사용자 지정 함수를 입력하려면 사용자 지정 함수 개발자가 제공하는 네임스페이스를 사용자 정의 함수의 접두사로 지정해야 합니다. 식 편집기를 사용하여 식을 작성하는 경우 모든 함수 및 함수 유형 목록에 사용자 지정 함수가 표시됩니다. 사용자 지정 함수를 다른 함수처럼 사용할 수 있습니다.

식의 유효성을 검사할 때 디자이너 또는 워크플로우 관리자는 사용자 지정 함수의 유효성을 검사하지 않습니다. 식의 유효성만 검사합니다. 플러그 인은 사용자 지정 함수의 유효성을 검사합니다.

## 사용자 지정 함수 API 참조

### 사용자 지정 함수 API 참조 개요

사용자 지정 함수 API는 변환 또는 워크플로우 식에 포함할 사용자 지정 함수를 개발하는 데 사용됩니다. 사용자 지정 함수 API는 사용자 지정 함수를 작성하는 프레임워크입니다. 여기에는 공통 API 및 런타임 API가 포함됩니다. API는 PowerCenter가 사용자 지정 함수를 포함하는 식의 유효성을 검사하고 이러한 식을 워크플로우에 사용할 수 있도록 합니다.

먼저 API를 헤더 및 구현 파일에 사용하여 사용자 지정 함수를 개발합니다. 그런 다음 헤더 및 구현 파일을 포함하는 공유 라이브러리를 작성합니다. PowerCenter에 등록된 리포지토리 플러그 인 파일에 공유 라이브러리를 지정합니다. 또한 공유 라이브러리를 PowerCenter 환경에 복사합니다.

### 공통 API

PowerCenter 클라이언트, 통합 서비스 및 리포지토리 서비스는 공통 API를 호출하여 식의 유효성을 검사하고 사용된 함수 반환, 함수 설명 및 프로토타입을 메모리에서 삭제합니다.

공통 API에는 다음 구조가 포함됩니다.

```
INFA_EXPR_VALIDATE_METHODS
├── INFA_EXPR_ValidateGetUserInterface()
│   ├── validateFunction
│   ├── getFunctionDescription
│   └── getFunctionPrototype
INFA_EXPR_OPD_METADATA
INFA_EXPR_GetPluginVersion
INFA_EXPR_DestroyString
```

### 유효성 검사 핸들

INFA\_EXPR\_VALIDATE\_METHODS 핸들은 유효성 검사 핸들입니다. PowerCenter는 INFA\_EXPR\_ValidateGetUserInterface를 호출하여 이 유효성 검사 핸들의 함수 포인터를 가져옵니다.

### 사용자 인터페이스 유효성 검사 함수

PowerCenter가 INFA\_EXPR\_ValidateGetUserInterface를 호출하면 플러그 인이 유효성 검사 함수에 대한 함수 포인터를 반환합니다.



다음 구문을 사용합니다.

```
INFA_EXPR_STATUS INFA_EXPR_ValidateGetUserInterface( IUNICHAR* sNamespace, IUNICHAR* sFuncName,
INFA_EXPR_VALIDATE_METHODS* functions);
```

인수	데이터 유형	입력/출력	설명
sNamespace	IUNICHAR	입력	사용자 지정 함수의 네임스페이스입니다.
sFuncName	IUNICHAR	입력	사용자 지정 함수의 이름입니다.
함수	INFA_EXPR_VALIDATE_METHODS	출력	유효성 검사 및 보고 중에 호출된 여러 함수에 대한 포인터입니다.

반환 데이터 유형은 `INFA_EXPR_STATUS`입니다. `ISUCCESS` 및 `IFAILURE`를 반환 값으로 사용합니다. 함수가 `IFAILURE`를 반환하면 플러그 인이 함수를 구현하지 못했거나 다른 오류가 발생한 것을 나타냅니다.

`INFA_EXPR_ValidateGetUserInterface`는 다음 함수를 반환합니다.

- **validateFunction.** 사용자 지정 함수의 유효성을 검사합니다.
- **getFunctionDescription.** 사용자 지정 함수를 설명합니다.
- **getFunctionPrototype.** 사용자 지정 함수에 대한 프로토타입을 제공합니다.
- **pushdownFunction.** 푸시다운 최적화를 위한 SQL 코드를 생성합니다.

#### 사용자 지정 함수 유효성 검사 함수

`PowerCenter`는 `validateFunction`을 호출하여 사용자 지정 함수의 인수에 대한 유효성을 검사합니다. 이 함수를 사용하여 사용자 지정 함수의 인수에 대한 이름, 데이터 유형, 전체 자릿수 및 배율을 제공합니다. 사용자 지정 함수의 반환 값에 대한 데이터 유형을 제공할 때도 이 함수를 사용합니다.

`PowerCenter`는 매핑 또는 워크플로우에 사용된 사용자 지정 함수의 각 인스턴스에 대해 이 함수를 한 번씩 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS *(validateFunction)(IUNICHAR* sNamespace, IUNICHAR* sFuncName, IUINT32 numArgs,
INFA_EXPR_OPD_METADATA** inputArgList, INFA_EXPR_OPD_METADATA* retValue);
```

인수	데이터 유형	입력/출력	설명
sNamespace	IUNICHAR	입력	함수의 네임스페이스입니다.
sFuncName	IUNICHAR	입력	유효성을 검사할 사용자 지정 함수의 이름입니다.
numArgs	IUINT32	입력	사용자 지정 함수에 있는 인수의 수입니다.

인수	데이터 유형	입력/출력	설명
inputArgList	INFA_EXPR_OPD_METADATA	입력	사용자 지정 함수의 입력 인수입니다.
retValue	INFA_EXPR_OPD_METADATA	출력	사용자 지정 함수의 반환 포트에 대한 메타데이터입니다. 반환 값의 데이터 유형, 전체 자릿수 및 배율을 이 인수에 설정합니다.

반환 데이터 유형은 `INFA_EXPR_STATUS`입니다. `ISUCCESS` 및 `IFAILURE`를 반환 값으로 사용합니다. 함수가 `IFAILURE`를 반환하면 PowerCenter가 오류 메시지를 표시합니다.

#### 사용자 지정 함수 설명 함수

PowerCenter는 `getFunctionDescription`을 호출하여 사용자 지정 함수의 설명을 가져옵니다. 또한 `destroyString`을 호출하여 사용을 마친 설명을 메모리에서 삭제합니다.

다음 구문을 사용합니다.

```
IUNICHAR* *(getFunctionDescription) (IUNICHAR* sNamespace, IUNICHAR* sFuncName);
```

인수	데이터 유형	입력/출력	설명
sNamespace	IUNICHAR	입력	함수의 네임스페이스입니다.
sFuncName	IUNICHAR	입력	플러그 인이 설명할 사용자 지정 함수의 이름입니다.

반환 데이터 유형은 `IUNICHAR`입니다. 반환 값은 `Null`로 종료되는 문자열이어야 합니다.

#### 사용자 지정 함수 프로토타입 함수

PowerCenter는 `getFunctionPrototype`을 호출하여 사용자 지정 함수의 인수를 식 편집기에 가져옵니다. 또한 `destroyString`을 호출하여 사용을 마친 인수를 메모리에서 삭제합니다.

다음 구문을 사용합니다.

```
IUNICHAR* *(getFunctionPrototype) (IUNICHAR* sNamespace, IUNICHAR* sFuncName);
```

인수	데이터 유형	입력/출력	설명
sNamespace	IUNICHAR	입력	함수의 네임스페이스입니다.
sFuncName	IUNICHAR	입력	플러그 인이 설명할 사용자 지정 함수의 이름입니다.

반환 데이터 유형은 `IUNICHAR`입니다. 반환 값은 `Null`로 종료되는 문자열이어야 합니다. 인수에 대한 값이 없는 경우 `NULL`을 반환합니다.

## 사용자 지정 함수 푸시다운 함수

PowerCenter는 pushdownFunction을 호출하여 푸시다운 최적화를 위한 SQL 코드를 생성합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS pushdownFunctionEcho(IUNICHAR* sNameSpace,
                                       IUNICHAR* sFuncName,
                                       IUINT32 numArgs,
                                       INFA_EXPR_OPD_METADATA** inputArgList,
                                       EDatabaseType dbType,
                                       EPushdownMode pushdownMode,
                                       IUNICHAR** sGenSql)
```

인수	데이터 유형	입력/출력	설명
sNameSpace	IUNICHAR	입력	함수의 네임스페이스입니다.
sFuncName	IUNICHAR	입력	유효성을 검사할 사용자 지정 함수의 이름입니다.
numArgs	IUNINT32	입력	사용자 지정 함수에 있는 인수의 수입니다.
inputArgList	INFA_EXPR_OPD_METADATA	입력	사용자 지정 함수의 입력 인수입니다.
dbType	EDatabaseType	입력	데이터베이스 유형입니다.
pushdownMode	EPushdownMode	입력	푸시다운 최적화의 유형입니다.
sGenSql	IUNICHAR	출력	사용자 지정 함수가 생성한 SQL입니다.

반환 데이터 유형은 INFA\_EXPR\_STATUS입니다. ISUCCESS 및 IFailure를 반환 값으로 사용합니다. 함수가 IFailure를 반환하면 PowerCenter가 오류 메시지를 표시합니다.

## INFA\_EXPR\_OPD\_METADATA 구조

이 구조는 함수에 전달된 인수 및 반환 유형 등 식 피연산자의 메타데이터를 정의합니다.

이 구조에는 다음 메타데이터가 포함됩니다.

- **데이터 유형.** 인수의 데이터 유형입니다.
- **전체 자릿수.** 인수의 전체 자릿수입니다.
- **배율.** 인수의 배율입니다.
- **isValueConstant.** 인수가 상수인지 여부를 나타냅니다. 상수인 경우 프레임워크가 각 함수 호출에 대해 한 번씩 인수를 평가합니다. 이 프레임워크는 isValueConstant를 사용하여 성능을 최적화합니다. 입력 인수가 상수인 경우 플러그 인은 함수 인스턴스 초기화 중에 인수 값을 가져와 성능을 최적화합니다. 출력 값의 경우 플러그 인은 isValueConstant를 TRUE로 설정합니다.

## 플러그 인 버전 함수 가져오기

이 함수는 플러그 인의 버전을 정의합니다. 이 함수는 사용자 지정 함수 API 버전인 1.0.0과 동일해야 합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS INFA_EXPR_GetPluginVersion(INFA_VERSION *sdkVersion, INFA_VERSION *pluginVersion);
```

인수	데이터 유형	입력/출력	설명
sdkVersion	INFA_VERSION	입력	사용자 지정 함수 API의 버전입니다. 1.0.0을 사용합니다.
pluginVersion	INFA_VERSION	출력	작성할 플러그 인의 버전입니다.

반환 데이터 유형은 **INFA\_EXPR\_STATUS**입니다. **ISUCCESS** 및 **IFAILURE**를 반환 값으로 사용합니다. 이 함수가 **IFAILURE**를 반환할 경우 세션 또는 워크플로우가 실패합니다.

### 문자열 함수 제거

이 함수는 플러그 인이 반환하는 모든 문자열을 제거합니다. 예를 들어 이 함수는 **getFunctionDescription**과 같은 다른 함수 호출의 오류 메시지 또는 반환 값을 제거합니다.

다음 구문을 사용합니다.

```
void *(DestroyString)(IUNICHAR* str);
```

인수	데이터 유형	입력/출력	설명
str	IUNICHAR	입력	이 함수가 삭제할 입력 문자열입니다.

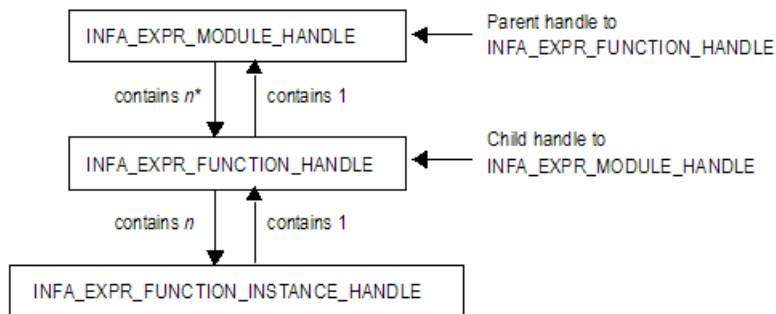
이 함수는 값을 반환하지 않습니다.

## 런타임 API

**PowerCenter** 통합 서비스는 세션 중에 런타임 API를 호출하여 사용자 지정 함수를 포함하는 식을 평가합니다. 통합 서비스는 플러그 인을 모듈, 함수 및 함수 인스턴스 수준에서 초기화합니다.

각 수준에는 함수 집합이 포함됩니다. 이러한 함수는 **INFA\_EXPR\_MODULE\_HANDLE**과 같은 핸들과 연결됩니다. 이러한 함수의 첫 번째 매개 변수는 함수의 영향을 받는 핸들입니다. 사용자 지정 함수 API 핸들은 서로 계층적 관계에 있습니다. 상위 핸들은 하위 핸들과 **1:n** 관계를 가집니다.

다음 그림은 사용자 지정 함수 API 핸들을 보여 줍니다.



다음 테이블에는 런타임 핸들이 설명되어 있습니다.

핸들 이름	설명
INFA_EXPR_MODULE_HANDLE	공유 라이브러리 또는 DLL을 나타냅니다. 플러그 인은 자체 공유 라이브러리 또는 DLL에 있는 모듈 핸들에만 액세스할 수 있습니다. 다른 모든 공유 라이브러리 또는 DLL의 모듈 핸들에는 액세스할 수 없습니다.
INFA_EXPR_FUNCTION_HANDLE	공유 라이브러리 또는 DLL 안의 사용자 지정 함수를 나타냅니다.
INFA_EXPR_FUNCTION_INSTANCE_HANDLE	특정 사용자 지정 함수 인스턴스를 나타냅니다.

## 모듈 수준 함수

PowerCenter는 각 공유 라이브러리 또는 DLL에 대해 모듈 수준 함수를 한 번씩 호출합니다.

### 사용자 인터페이스 모듈 수준 함수 가져오기

이 함수는 모듈 수준 상호 작용에 대한 함수 포인터를 설정합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS INFA_EXPR_ModuleGetUserInterface(INFA_EXPR_LIB_METHODS* functions);
```

인수	데이터 유형	입력/출력	설명
함수	INFA_EXPR_LIB_METHODS	출력	모듈이 사용자 인터페이스 함수를 가져옵니다.

반환 데이터 유형은 INFA\_EXPR\_STATUS입니다. ISUCCESS 및 IFailure를 반환 값으로 사용합니다. 이 함수가 IFailure를 반환할 경우 세션 또는 워크플로우가 실패합니다.

이 함수는 다음 함수를 반환합니다.

- **function\_init.** 함수를 초기화합니다.
- **function\_deinit.** 함수 초기화를 취소합니다.

### 모듈 수준 초기화 함수

PowerCenter는 각 모듈에 대해 이 `module_init`를 한 번씩 호출하여 함수의 모든 글로벌 데이터 구조를 초기화합니다. PowerCenter는 함수 수준 함수를 호출하기 전에 이 함수를 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS (*module_init) (INFA_EXPR_MODULE_HANDLE module);
```

인수	데이터 유형	입력/출력	설명
모듈	INFA_EXPR_MODULE_HANDLE	입력	함수 수준에서 검색할 수 있는 데이터를 저장합니다.

반환 데이터 유형은 `INFA_EXPR_STATUS`입니다. `ISUCCESS` 및 `IFAILURE`를 반환 값으로 사용합니다. 이 함수가 `IFAILURE`를 반환할 경우 세션 또는 워크플로우가 실패합니다.

#### 모듈 수준 초기화 취소 함수

`PowerCenter`는 각 모듈에 대해 `module_deinit`를 한 번씩 호출하여 이 함수의 모든 데이터 구조의 초기화를 취소합니다. `PowerCenter`는 모든 함수 수준 통합이 완료된 후에 이 함수를 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS (*module_deinit) (INFA_EXPR_MODULE_HANDLE module);
```

인수	데이터 유형	입력/출력	설명
모듈	<code>INFA_EXPR_MODULE_HANDLE</code>	입력	모듈 초기화 함수가 호출되었을 때 프레임워크가 플러그 인에 전달하는 모듈 수준 핸들입니다.

반환 데이터 유형은 `INFA_EXPR_STATUS`입니다. `ISUCCESS` 및 `IFAILURE`를 반환 값으로 사용합니다. 이 함수가 `IFAILURE`를 반환할 경우 세션 또는 워크플로우가 실패합니다.

### 함수 수준 함수

`PowerCenter`는 각 사용자 지정 함수와 사용자 지정 함수에 대한 매개 변수를 제공하는 각 공유 라이브러리 또는 DLL에 대해 함수 수준 함수를 한 번씩 호출합니다.

#### 사용자 인터페이스 함수 수준 함수 가져오기

이 함수는 함수 수준 상호 작용에 대한 함수 포인터를 설정합니다. `PowerCenter`는 이 라이브러리가 구현하는 모든 사용자 지정 함수에 대해 이 함수를 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS INFA_EXPR_FunctionGetUserInterface (IUNICHAR* nameSpaceName, IUNICHAR* functionName, INFA_EXPR_FUNCTION_METHODS* functions);
```

인수	데이터 유형	입력/출력	설명
<code>nameSpaceName</code>	<code>IUNICHAR</code>	입력	함수의 네임스페이스입니다.
<code>functionName</code>	<code>IUNICHAR</code>	입력	플러그 인이 설명할 사용자 지정 함수의 이름입니다.
함수	<code>INFA_EXPR_FUNCTION_METHODS</code>	입력	함수 인스턴스 수준에서 호출할 함수 포인터에 대한 자리 표시자입니다.

반환 데이터 유형은 `INFA_EXPR_STATUS`입니다. `ISUCCESS` 및 `IFAILURE`를 반환 값으로 사용합니다. 이 함수가 `IFAILURE`를 반환할 경우 세션 또는 워크플로우가 실패합니다.

이 함수는 다음 함수를 반환합니다.

- **function\_init.** 함수를 초기화합니다.

- **function\_deinit.** 함수 초기화를 취소합니다.

#### 함수 수준 초기화 함수

PowerCenter는 각 사용자 지정 함수에 대해 **function\_init**를 한 번씩 호출하여 사용자 지정 함수에 관련된 모든 구조를 초기화합니다. PowerCenter는 모듈 수준 초기화 함수를 호출한 다음 이 함수를 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS (*function_init) (INFA_EXPR_FUNCTION_HANDLE fnInstance);
```

인수	데이터 유형	입력/출력	설명
fnInstance	INFA_EXPR_FUNCTION_HANDLE	입력	<p>다음 태스크를 수행합니다.</p> <ul style="list-style-type: none"> <li>- 런타임 또는 초기화 취소 중에 검색할 프레임워크에 대한 사용자 정의 포인터를 저장합니다.</li> <li>- 함수 인스턴스 수준에 대한 데이터 구조를 초기화합니다.</li> <li>- 입력 인수가 상수인 경우 플러그 인이 이 상수 값을 검색하고 필요한 모든 전처리를 수행합니다.</li> </ul>

반환 데이터 유형은 **INFA\_EXPR\_STATUS**입니다. **ISUCCESS** 및 **IFAILURE**를 반환 값으로 사용합니다. 이 함수가 **IFAILURE**를 반환할 경우 세션 또는 워크플로우가 실패합니다.

#### 함수 수준 초기화 취소 함수

PowerCenter는 각 함수 수준에 대해 이 함수를 한 번씩 호출하여 사용자 지정 함수에 관련된 모든 구조의 초기화를 취소합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS (*function_deinit) (INFA_EXPR_FUNCTION_HANDLE function);
```

인수	데이터 유형	입력/출력	설명
fnInstance	INFA_EXPR_FUNCTION_HANDLE	입력	함수 인스턴스 수준 초기화 함수가 호출되었을 때 프레임워크가 플러그 인에 전달하는 함수 수준 핸들입니다.

반환 데이터 유형은 **INFA\_EXPR\_STATUS**입니다. **ISUCCESS** 및 **IFAILURE**를 반환 값으로 사용합니다. 이 함수가 **IFAILURE**를 반환할 경우 세션 또는 워크플로우가 실패합니다.

#### 함수 인스턴스 수준 함수

PowerCenter는 사용자 지정 함수가 매핑 또는 워크플로우에 사용될 때마다 이 함수를 호출합니다.

### 사용자 인터페이스 함수 수준 함수 가져오기

이 함수는 함수 수준 상호 작용에 대한 함수 포인터를 설정합니다. PowerCenter는 이 라이브러리가 구현하는 모든 사용자 지정 함수에 대해 이 함수를 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS INFA_EXPR_FunctionInstanceGetUserInterface(IUNICHAR* functionName,
INFA_EXPR_FUNCTION_INSTANCE_METHODS* functions)
```

인수	데이터 유형	입력/출력	설명
functionName	IUNICHAR	입력	함수의 네임스페이스입니다.
함수	INFA_EXPR_FUNCTION_INSTANCE_METHODS	입력	함수 인스턴스 수준에서 호출할 함수 포인터에 대한 자리 표시자입니다.

이 함수는 다음 함수를 반환합니다.

- **fnInstance\_init.** 사용자 지정 함수의 인스턴스를 초기화합니다.
- **fnInstance\_processRow.** 사용자 지정 함수의 인스턴스에 대한 데이터를 처리합니다.
- **fnInstance\_deinit.** 사용자 지정 함수의 인스턴스 초기화를 취소합니다.

### 함수 인스턴스 수준 초기화 함수

PowerCenter는 각 사용자 지정 함수 인스턴스에 대해 **fnInstance\_init**를 한 번씩 호출하여 사용자 지정 함수 인스턴스에 관련된 모든 구조를 초기화합니다. 매핑 또는 워크플로우에 사용자 지정 함수의 인스턴스가 2개 있는 경우 PowerCenter는 이 함수를 두 번 호출합니다. PowerCenter는 모듈 수준 초기화 함수를 호출한 다음 이 함수를 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS (*fnInstance_init)(INFA_EXPR_FUNCTION_INSTANCE_HANDLE fnInstance);
```

인수	데이터 유형	입력/출력	설명
fnInstance	INFA_EXPR_FUNCTION_HANDLE	입력	다음 태스크를 수행합니다. <ul style="list-style-type: none"> <li>- 런타임 또는 초기화 취소 중에 검색할 프레임워크에 대한 사용자 정의 포인터를 저장합니다.</li> <li>- 함수 인스턴스 수준에 대한 데이터 구조를 초기화합니다.</li> <li>- 입력 인수가 상수인 경우 플러그 인이 이 상수 값을 검색하고 필요한 모든 전처리를 수행합니다.</li> </ul>

반환 데이터 유형은 **INFA\_EXPR\_STATUS**입니다. **ISUCCESS** 및 **IFAILURE**를 반환 값으로 사용합니다. 이 함수가 **IFAILURE**를 반환할 경우 세션 또는 워크플로우가 실패합니다.



### 함수 인스턴스 행 처리 함수

PowerCenter는 입력 행이 사용자 지정 함수 인스턴스에 제공되는 경우 `fnInstance_processRow`를 호출합니다. 사용자 지정 함수의 입력 인수에 대한 데이터는 `fnInstance-inputOPDHandles`를 통해 바인딩되고 액세스됩니다. 출력 및 반환 포트에 대한 데이터, 길이 및 표시기는 `fnInstance->retHandle`에서 설정합니다. PowerCenter는 함수 수준 초기화 함수를 호출한 다음 이 함수를 호출합니다.

다음 구문을 사용합니다.

```
INFA_EXPR_ROWSTATUS (*fnInstance_processRow) (INFA_EXPR_FUNCTION_INSTANCE_HANDLE fnInstance);
```

인수	데이터 유형	입력/출력	설명
fnInstance	INFA_EXPR_FUNCTION_HANDLE	입력	데이터를 사용할 수 있는 함수 수준 핸들입니다.

반환 값의 데이터 유형은 `INFA_EXPR_ROWSTATUS`입니다. 다음 값을 반환 값으로 사용합니다.

- **INFA\_ROWSUCCESS.** 함수가 데이터 행을 성공적으로 처리했음을 나타냅니다.
- **INFA\_ROWERROR.** 함수가 데이터 행을 처리하는 동안 오류가 발생했음을 나타냅니다. PowerCenter 통합 서비스가 내부 오류 수를 증분 처리합니다. 이 값은 데이터 액세스 모드가 행인 경우에만 반환됩니다.
- **INFA\_FATALERROR.** 함수가 데이터 행 또는 데이터 블록을 처리하는 동안 치명적인 오류가 발생했음을 나타냅니다. PowerCenter 통합 서비스의 세션이 실패합니다.

### 함수 인스턴스 수준 초기화 취소 함수

PowerCenter는 초기화를 취소하는 동안 각 함수 수준에 대해 `fnInstance_deinit`를 한 번씩 호출합니다. PowerCenter는 이 함수를 호출하여 사용자 지정 함수에 관련된 모든 구조의 초기화를 취소할 수 있습니다.

다음 구문을 사용합니다.

```
INFA_EXPR_STATUS (*fnInstance_deinit)(INFA_EXPR_FUNCTION_INSTANCE_HANDLE fnInstance);
```

인수	데이터 유형	입력/출력	설명
fnInstance	INFA_EXPR_FUNCTION_INSTANCE_HANDLE	입력	함수 인스턴스 수준 초기화 함수가 호출되었을 때 프레임워크가 플러그 인에 전달하는 함수 수준 핸들입니다.

반환 데이터 유형은 `INFA_EXPR_STATUS`입니다. `ISUCCESS` 및 `IFAILURE`를 반환 값으로 사용합니다. 이 함수가 `IFAILURE`를 반환할 경우 세션 또는 워크플로우가 실패합니다.