ऑन लाइन पाठ्य सामग्री

1PGDCA3
DATABASE USING MYSQL
(ELECTIVE–I)

इकाई – तीन

प्रियंका अस्थाना
गेस्ट फैकल्टी, कम्प्यूटर विज्ञान एवं अनुप्रयोग
माखनलाल चतुर्वेदी राष्ट्रीय पत्रकारिता एवं संचार विश्वविद्यालय, भोपाल

माखनलाल चतुर्वेदी राष्ट्रीय पत्रकारिता एवं संचार विश्वविद्यालय
बी-38, विकास भवन, एम.पी. नगर, झोन – I, भोपाल 462011

इकाई-3

3.1 DATABASE QUERY COMMAND:

3.1.1. DDL-

डेटा डेफिनिशन लैंग्वेज (DDL) स्टेटमेंट्स का उपयोग डेटाबेस स्ट्रक्चर या Schema को परिभाषित करने के लिए किया जाता है। डेटा डेफिनिशन लैंग्वेज, डेटाबेस Schema के साथ समझकर यह बताता है कि डेटा को डेटाबेस में कैसे होना चाहिए, इसलिए भाषा विवरण जैसे CREATE TABLE या ALTER TABLE DDL से संबंधित है। DDL "मेटाडेटा" के बारे में है।

DDL में CREATE, ALTER और DROP स्टेटमेंट जैसे कमांड शामिल हैं। DDL का उपयोग डेटाबेस ऑब्जेक्ट्स (Table, Views, Users) को बनाने, बदलने या बनाने के लिए किया जाता है।

डेटा डेफिनिशन लैंग्वेज (DDL) में विभिन्न कथनों का इस्तेमाल किया गया है:

• CREATE - डेटाबेस में ऑब्जेक्ट बनाने के लिए

• ALTER - डेटाबेस की संरचना को बदल देता है

• DROP - डेटाबेस से ऑब्जेक्ट हटाएं

• TRUNCATE - एक टेबल से सभी रिकॉर्ड को हटा देता है , जिसमें रिकॉर्ड के लिए □वंटित सभी रिक्त स्थान शामिल हैं

• COMMENT - डेटा शब्दकोश में टिप्पणी जोड़ने के लिए

• RENAME - एक ऑब्जेक्ट का नाम बदलने के लिए

3.1.2. DML-

डेटा मैनीपुलेशन लैंग्वेज (DML) स्टेटमेंट्स को Schema ऑब्जेक्ट्स के भीतर डेटा के प्रबंधन के लिए उपयोग किया जाता है। DML डेटा हेरफेर (मैनीपुलेशन) से संबंधित है, और इसलिए इसमें अधिकांश सामान्य SQL कथन शामिल हैं जैसे: SELECT, INSERT □दि DML स्वयं डेटा को add / modify / delete करने की अनुमति देता है।

डेटाबेस ऑब्जेक्ट्स में मौजूदा डेटा के साथ हेरफेर (मैनीपुलेशन) करने के लिए DML का उपयोग किया जाता है (insert, select, update, delete).

DML COMMANDS**:**

1. INSERT: Tables में डेटा डालने के लिए।
2. SELECT: Tables से डेटा पुनर्प्राप्त करने के लिए।
3. UPDATE: Tables में डेटा को संशोधित करने के लिए।
4. DELETE: Table से डेटा को delete/clear करने के लिए।

3.1.3. DCL-

DCL डेटा कंट्रोल लैंग्वेज का सार है। डेटा कंट्रोल लैंग्वेज में GRANT जैसी कमांड्स शामिल हैं, और अधिकारों, अनुमतियों और डेटाबेस सिस्टम के अन्य नियंत्रणों से संबंधित हैं। DCL का उपयोग डेटाबेस और उनकी सामग्री पर अनुमति देने / रद्द करने के लिए किया जाता है। DCL सरल है, लेकिन MySQL अनुमतियां थोड़ी जटिल हैं। DCL सुरक्षा के बारे में है। DCL का उपयोग डेटाबेस ट्रांजेक्शन को नियंत्रित करने के लिए किया जाता है। DCL स्टेटमेंट □पको यह नियंत्रित करने की अनुमति देता है कि □पके डेटाबेस में कौन-सी विशिष्ट वस्तु है। दो तरह के DCL कमांड निम्नलिखित हैं:

1. GRANT
2. REVOKE

1. GRANT:

यह डेटाबेस तक उपयोगकर्ता की पहुँच विशेषाधिकार प्रदान करता है। MySQL डेटाबेस में कंट्रोलर विकल्पों के लिए बहुत हद तक प्रशासक और उपयोगकर्ता दोनों प्रदान करता है। प्रक्रिया के व्यवस्थापन पक्ष में व्यवस्थापकों के लिए MySQL सर्वर पर कुछ उपयोगकर्ता विशेषाधिकारों को नियंत्रित करने की संभावना शामिल होती है, ताकि वे एक संपूर्ण डेटाबेस तक अपनी पहुंच को सीमित कर सकें या किसी विशिष्ट तालिका के लिए अनुमतियों को सीमित कर सकें। यह सुरक्षा प्रणाली में एक प्रविष्टि बनाता है जो वर्तमान डेटाबेस में उपयोगकर्ता को डेटा के साथ काम करने या विशिष्ट विवरणों को निष्पादित करने की अनुमति देता है।

Syntax :

Statement permissions:

GRANT { ALL | statement [,...n] }

TO security_account [,...n]

□म तौर पर, एक डेटाबेस व्यवस्थापक खाता बनाने के लिए पहले CREATE USER का उपयोग करता है, फिर अपने विशेषाधिकारों और विशेषताओं को परिभाषित करने के लिए GRANT का|

2. REVOKE:

REVOKE कथन सिस्टम प्रशासकों को सक्षम बनाता है और MySQL खातों से विशेषाधिकार रद्द करता है।

Syntax : REVOKE

priv_type [(column_list)]

[, priv_type [(column_list)]] ...

ON [object_type] priv_level

FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION

FROM user [, user] ...

3.2 DATA TYPES

एक डेटाबेस Table में विशिष्ट डेटा प्रकार जैसे कि संख्यात्मक या स्ट्रिंग के साथ कई कॉलम होते हैं। MySQL सिर्फ संख्यात्मक या स्ट्रिंग के अलावा अन्य डेटा प्रकार प्रदान करता है। MySQL के प्रत्येक डेटा प्रकार को निम्नलिखित विशेषताओं द्वारा निर्धारित किया जा सकता है:

• यह किस प्रकार के मूल्यों (values) का प्रतिनिधित्व करता है।

• वह स्थान जो ऊपर ले जाता है और बताता है कि मान निश्चित लंबाई है या परिवर्तनशील लम्बाई है।

• डेटा प्रकार के मूल्यों (values) को अनुक्रमित (index) किया जा सकता है या नहीं।

• MySQL एक विशिष्ट डेटा प्रकार के मूल्यों (values) की तुलना कैसे करता है

MySQL में तीन मुख्य डेटा प्रकार हैं: स्ट्रिंग, न्यूमेरिक, और दिनांक और समय। (string, numeric, date and time)

3.2.1. स्ट्रिंग डेटा प्रकार (String data type):

डेटा प्रकार	विवरण
CHAR (Size)	एक FIXED लंबाई की स्ट्रिंग (इसमें अक्षर, संख्या और विशेष वर्ण हो सकते हैं)। Size पैरामीटर निर्दिष्ट करता है कि वर्णों में स्तंभ की लंबाई 0 से 255 तक हो सकती है। डिफ़ॉल्ट 1 है
VARCHAR (Size)	एक VARIABLE लंबाई स्ट्रिंग (अक्षर, संख्या और विशेष वर्ण हो सकते हैं)। Size पैरामीटर वर्णों में अधिकतम स्तंभ लंबाई निर्दिष्ट करता है - 0 से 65535 तक हो सकता है
BINARY (Size)	CHAR () के बराबर है, लेकिन बाइनरी बाइट स्ट्रिंग्स को संग्रहीत करता है। Size पैरामीटर बाइट्स में स्तंभ की लंबाई निर्दिष्ट करता है। डिफ़ॉल्ट 1 है
VARBINARY (size)	वर्चसर () के बराबर, लेकिन बाइनरी बाइट स्ट्रिंग्स को संग्रहीत करता है। Size पैरामीटर बाइट्स में अधिकतम स्तंभ कि लंबाई निर्दिष्ट करता है।
TINYBLOB	BLOBs (Binary Large Objects) के लिए। अधिकतम लंबाई: 255 बाइट्स
TINYTEXT	255 अक्षरों की अधिकतम लंबाई के साथ एक स्ट्रिंग रखता है
TEXT (Size)	65,535 बाइट्स की अधिकतम लंबाई के साथ एक स्ट्रिंग धारण करता है
BLOB (Size)	BLOB (Binary Large Objects) के लिए । डेटा के 65,535 बाइट्स तक को धारण रखता है
MEDIUMTEXT	16,777,215 वर्णों की अधिकतम लंबाई के साथ एक स्ट्रिंग बनता है

MEDIUMBLOB	BLOBs के लिए (Binary Large Objects)। डेटा के 16,777,215 बाइट्स तक को धारणरखता है
LONGTEXT	4,294,967,295 वर्णों की अधिकतम लंबाई के साथ एक स्ट्रिंग को धारण करता है
LONGBLOB	BLOBs के लिए (Binary Large Objects)। 4,294,967,295 बाइट्स का डेटा धारण करता है
ENUM	एक स्ट्रिंग ऑब्जेक्ट जिसका केवल एक मान हो सकता है,जो कि
(val1, val2, val3, ...)	संभावित मानों की सूची से चुना गया है। □प एक ENUM सूची में 65535 मान तक सूचीबद्ध कर सकते हैं। यदि कोई मान सम्मिलित किया गया है जो सूची में नहीं है, तो कोई रिक्त मान सम्मिलित किया जाता है। □पके द्वारा दर्ज किए गए क्रम में मानों को क्रमबद्ध किया गया है
SET	एक स्ट्रिंग ऑब्जेक्ट जिसमें 0 या अधिक मान हो सकते हैं, जो कि
(val1, val2, val3, ...)	संभावित मानों की सूची से चुना गया है। □प एक SET सूची में 64 मान तक सूचीबद्ध कर सकते हैं

3.2.2. संख्यात्मक डेटा प्रकार (Numeric data types):

डेटा प्रकार	विवरण
BIT (size)	एक बिट-मूल्य प्रकार। प्रति मूल्य कि बिट्स की संख्या Size में निर्दिष्ट है। Size पैरामीटर 1 से 64 तक मान रख सकता है। Size का डिफ़ॉल्ट मान 1 है।
TINYINT (size)	एक बहुत छोटा पूर्णांक। Signed range -128 से 127 है। Unsigned range 0 से 255 तक है। Size पैरामीटर अधिकतम प्रदर्शन चौड़ाई निर्दिष्ट करता है (जो 255 है)
BOOL	शून्य को असत्य(false) माना जाता है, गैर-शून्य मानों को सत्य (true) माना जाता है।
BOOLEAN	BOOL के बराबर

SMALLINT (size)	एक छोटा पूर्णांक। Signed range -32768 से 32767 तक है। Unsigned range 0 से 65535 तक है। Size पैरामीटर अधिकतम प्रदर्शन चौड़ाई (जो 255 है) निर्दिष्ट करता है
MEDIUMINT (size)	एक मध्यम पूर्णांक। Signed range -8388608 से 8388607 है। Unsigned range 0 से 16777215 तक है। Size पैरामीटर अधिकतम प्रदर्शन चौड़ाई निर्दिष्ट करता है (जो 255 है)
INT (Size)	एक मध्यम पूर्णांक। Signed range -2147483648 से 2147483647 तक है। Unsigned range 0 से 4294967295 तक है। Size पैरामीटर अधिकतम डिस्प्ले चौड़ाई निर्दिष्ट करता है (जो 255 है)
INTEGER (Size)	INT (Size) के बराबर
BIGINT (size)	एक बड़ा पूर्णांक। Signed range -9223372036854775808 से 9223372036854775807 है। Unsigned range 0 से 18446744073709551615 है। Size पैरामीटर अधिकतम डिस्प्ले चौड़ाई (जो 255 है) निर्दिष्ट करता है
FLOAT (Size, d)	एक फ्लोटिंग पॉइंट नंबर। अंकों की कुल संख्या Size में निर्दिष्ट है। दशमलव बिंदु के बाद अंकों की संख्या d पैरामीटर में निर्दिष्ट है। इस सिंटैक्स को MySQL 8.0.17 में पदावनत (deprecated) किया गया है, और इसे भविष्य के MySQL versions में हटा दिया जाएगा
FLOAT (p)	एक फ्लोटिंग पॉइंट नंबर। MySQL, p मान का उपयोग यह निर्धारित करने के लिए करता है कि परिणामी डेटा प्रकार के लिए FLOAT या DOUBLE का उपयोग करना है या नहीं। यदि p, 0 से 24 तक है, तो डेटा प्रकार FLOAT () हो जाता है। यदि p, 25 से 53 तक है, तो डेटा प्रकार DOUBLE () बन जाता है
DOUBLE (size, d)	एक सामान्य Size का फ्लोटिंग पॉइंट नंबर। अंकों की कुल संख्या Size में निर्दिष्ट है। दशमलव बिंदु के बाद अंकों की संख्या d पैरामीटर में निर्दिष्ट है
DOUBLE PRECISION (size, d)	

DECIMAL (Size, d)	एक सटीक निश्चित बिंदु संख्या। अंकों की कुल संख्या Size में निर्दिष्ट है। दशमलव बिंदु के बाद अंकों की संख्या d पैरामीटर में निर्दिष्ट है। Size के लिए अधिकतम संख्या 65 है। d के लिए अधिकतम संख्या 30 है। Size के लिए डिफ़ॉल्ट मान 10 है। d के लिए डिफ़ॉल्ट मान 0 है।
DEC (Size, d)	DECIMAL के बराबर (Size, d)

3.2.3. दिनांक और समय डेटा प्रकार (Date and Time data types):

डेटा प्रकार	विवरण
DATE	डेट प्रारूप: YYYY-MM-DD| समर्थित सीमा '1000-01-01' से '9999-12-31' तक है
DATETIME (fsp)	एक दिनांक और समय संयोजन। स्वरूप: YYYY-MM-DD hh: mm: ss। समर्थित रेंज '1000-01-01 00:00:00' से '9999-12-31 23:59:59' तक है। Automatic initialization और updating प्राप्त करने के लिए column definition में DEFAULT and ON UPDATE जोड़ें
TIMESTAMP (fsp)	एक टाइमस्टैम्प। TIMESTAMP मान यूनिक्स युग ('1970-01-01 00:00:00' UTC) के बाद से सेकंड की संख्या के रूप में संग्रहीत हैं। स्वरूप: YYYY-MM- DD hh: mm: ss। समर्थित सीमा '1970-01-01 00:00:01' UTC से '2038-01-09 03:14:07 'UTC से है। कॉलम की परिभाषा में DEFAULT CURRENT_TIMESTAMP और ON UPDATE CURRENT_TIMESTAMP का उपयोग करके वर्तमान दिनांक और समय पर Automatic initialization and updating किया जा सकता है
TIME (fsp)	एक समय। स्वरूप: hh: mm: ss। समर्थित सीमा '-838: 59: 59' से '838: 59: 59' तक है
YEAR	चार अंकों के प्रारूप में एक वर्ष। चार अंकों के प्रारूप में अनुमत मान: 1901 से 2155, और 0000। MySQL 8.0 दो अंकों के प्रारूप में वर्ष का समर्थन नहीं करता है।

3.2.4. MySQL spatial data types

MySQL कई स्थानिक डेटा प्रकारों का समर्थन करता है जिसमें विभिन्न प्रकार के ज्यामितीय और भौगोलिक मूल्य शामिल हैं जैसा कि निम्नलिखित Table में दिखाया गया है:

स्थानिक डेटा प्रकार	विवरण
GEOMETRY:	किसी भी प्रकार का एक स्थानिक मान
POINT:	एक बिंदु (X-Y coordinates की एक जोड़ी)
LINESTRING:	एक वक्र (एक या अधिक बिंदु मान)
POLYGON:	एक बहुभुज
GEOMETRYCOLLECTION:	GEOMETRY values का संग्रह
MULTILINESTRING:	LINESTRING values का संग्रह
MULTIPOINT:	POINT values का संग्रह
MULTIPOLYGON:	POLYGON values का संग्रह

3.2.5. JSON डेटा प्रकार

MySQL ने version 5.7.8 के बाद से एक देशी JSON डेटा प्रकार का समर्थन किया जो □पको JSON दस्तावेज़ों को अधिक कुशलता से संग्रहीत और प्रबंधित करने की अनुमति देता है। देशी JSON डेटा प्रकार JSON दस्तावेजों और optimal storage प्रारूप का स्वत: सत्यापन प्रदान करता है।

3.3 SQL कमांड का उपयोग करते हुए TABLE का निर्माण:

Table निर्माण कमांड के साथ शुरू करने के लिए निम्नलिखित विवरण की □वश्यकता होती है -

- टेबल का नाम (Name of the table)
- फ़ील्ड्स का नाम (Name of the fields)
- प्रत्येक फील्ड कि परिभाषा (Definitions for each field)

Syntax:

यहाँ एक सामान्य SQL सिंटैक्स एक MySQL Table बनाने के लिए है -

Create table table_name (fieldname1 datatype(), fieldname2 datatype()...);

Ex: CREATE TABLE Persons (PersonID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255));

3.4 APPLYING CONSTRAINTS ON TABLES:

MySQL CONSTRAINT का उपयोग नियमों को परिभाषित करने या उन मूल्यों को प्रतिबंधित करने के लिए किया जाता है जो स्तंभों में संग्रहीत किए जा सकते हैं। बाध्यताओ को प्रेरित करने का उद्देश्य एक डेटाबेस की अखंडता(integrity) को लागू करना है।

MySQL CONSTRAINTS का उपयोग डेटा के प्रकार को सीमित करने के लिए किया जाता है जिसे एक Table में डाला जा सकता है।

MySQL CONSTRAINTS को दो प्रकारों में वर्गीकृत किया जा सकता है - column level और table level ।

स्तंभ स्तर की बाध्यताएं केवल एक स्तंभ पर लागू हो सकती हैं जहाँ Table स्तर की बाध्यताएं पूरी Table पर लागू होती हैं।

Table बनाते समय MySQL CONSTRAINT घोषित किया जाता है।

MySQL CONSTRAINTs हैं:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY

- C̈ ECK
- DEFAULT

CONSTRAINT	DESCRIPTION
NOT NULL	MySQL में NOT NULL बाध्यताएं यह निर्दिष्ट करने की अनुमति देती है कि किसी स्तंभ में कोई पूर्ण (null) मान नहीं हो सकता है। MySQL NOT NULL को एक टेबल बनाने और टेबल में बदलाव के लिए इस्तेमाल किया जा सकता है।
UNIQUE	MySQL में UNIQUE बाध्यताएं (constraint) एक कॉलम में डुप्लिकेट मान सम्मिलित करने की अनुमति नहीं देती है। UNIQUE बाध्यताएं एक तालिका में एक स्तंभ की विशिष्टता को बनाए रखती है। एक तालिका में एक से अधिक UNIQUE कॉलम का उपयोग किया जा सकता है।
PRIMARY KEY	एक तालिका के लिए एक प्राथमिक कुंजी बाध्यता विशिष्ट कॉलम के लिए अद्वितीय (unique) डेटा को स्वीकार करने के लिए तालिका को लागू करती है और यह constraint तालिका को तेजी से एक्सेस करने के लिए एक अद्वितीय सूचकांक बनाती है।
FOREIGN KEY	MySQL में एक FOREIGN KEY दोनों तालिकाओं के एक विशिष्ट कॉलम द्वारा दो तालिकाओं के बीच एक लिंक बनाती है। एक तालिका में निर्दिष्ट कॉलम एक प्राथमिक कुंजी होना चाहिए और एक अन्य तालिका के स्तंभ द्वारा संदर्भित किया जाना चाहिए जिसे FOREIGN KEY कहा जाता है।
C̈ ECK	एक चेक बाधा संबंधित कॉलम में मूल्यों को नियंत्रित करती है। C̈ ECK बाधा निर्धारित करती है कि मान तार्किक अभिव्यक्ति से मान्य है या नहीं।
DEFAULT	MySQL तालिका में, प्रत्येक कॉलम में एक मान (एक NULL सहित) होना चाहिए। किसी तालिका में डेटा सम्मिलित करते समय, यदि स्तंभ में कोई मूल्य नहीं दिया जाता है, तो स्तंभ को मान को DEFAULT के रूप में सेट किया जाता है।

General Syntax :

```
CREATE  TABLE [table name]

([column name] [data type]([size]) [column constraint]….

[table constraint] ([[column name]……])……);
```

3.4.1 NULL CONSTRAINT के साथ MySQL CREATE TABLE

MySQL Table बनाते समय, डिफ़ॉल्ट मान NOT NULL का उपयोग करें, यह लागू किया जा सकता है कि Table में एक स्तंभ को NULL मान संग्रहीत करने की अनुमति नहीं है।

उदाहरण

यदि □प एक table बनाना चाहते हैं 'newauthor' जहां किसी भी कॉलम को NULL VALUES संग्रहीत करने की अनुमति नहीं है, निम्नलिखित कथन का उपयोग किया जा सकता है।

```
CREATE TABLE IF NOT EXISTS newauthor

(aut_id varchar(8) NOT NULL,

aut_name varchar(50) NOT NULL,

country varchar(25) NOT NULL,

home_city varchar(25) NOT NULL );
```

यहाँ उपरोक्त कथन में NULL Value को बाहर करने के लिए बाध्यता 'NOT NULL' का उपयोग किया गया है।

निम्न चित्र से पता चलता है कि कॉलम NULL मान को स्वीकार नहीं करेंगे।

	Field	Type	Collation	Attributes	Null	Default	Extra
☐	aut_id	varchar(8)	latin1_swedish_ci		No	*None*	
☐	aut_name	varchar(50)	latin1_swedish_ci		No	*None*	
☐	country	varchar(25)	latin1_swedish_ci		No	*None*	
☐	home_city	varchar(25)	latin1_swedish_ci		No	*None*	

3.4.2 CREATE TABLE to check values with CHECK CONSTRAINT

किसी Table के स्तंभ पर एक C ECK CONSTRAINT जोड़कर, □प उस स्तंभ में संग्रहीत मानों की सीमा को सीमित कर सकते हैं।

उदाहरण

यदि □प 'book_id' कॉलम पर एक प्राथमिक कुंजी के साथ एक Table 'newbook_mast' बनाना चाहते हैं, तो 'isbn_no' कॉलम पर एक अद्वितीय बाध्यता और इसमें no_page सेट करें, ताकि यह शून्य से अधिक मानों को धारण कर सके, निम्न कथन इस्तेमाल किया जा सकता है।

```
CREATE TABLE IF NOT EXISTS
newbook_mast (book_id varchar(15) NOT NULL UNIQUE,
book_name varchar(50)  ,
isbn_no varchar(15)  NOT NULL UNIQUE  ,
cate_id varchar(8)  ,
aut_id varchar(8) ,
pub_id varchar(8) ,
dt_of_pub date ,
pub_lang varchar(15) ,
no_page decimal(5,0)
C ECK(no_page>0) ,
book_price decimal(8,2) ,
PRIMARY KEY (book_id)
);
```

यहाँ ऊपर MySQL स्टेटमेंट में एक Table 'newbook_mast' बनाई जाएगी जिसमें 'Book _id' कॉलम पर एक PRIMARY KEY होगी, 'isbn_no' कॉलम पर अद्वितीय बाध्यता

(unique constraint) और C̈ ECK (no_page> 0) को जोड़ने से, no_page को इस तरह सेट किया जाएगा, कि यह केवल शून्य से अधिक मूल्य दे।

3.4.3 MySQL UNIQUE CONSTRAINT

UNIQUE बाध्यता इस तरह से एक सूचकांक (इंडेक्स) बनाता है जिससे सूचकांक कॉलम में सभी मान अद्वितीय होने चाहिए। त्रुटि तब होती है जब कोई भी एक महत्वपूर्ण मान के साथ एक नई पंक्ति जोड़ने का प्रयास करता है जो उस पंक्ति में पहले से मौजूद है।

उदाहरण

नीचे दिया गया MySQL कथन एक स्तंभ 'aut_id' के साथ एक Table 'newauthor' बनाएगा जो अद्वितीय मूल्यों को संग्रहीत करेगा क्यूकि UNIQUE (aut_id) का उपयोग किया गया है।

```
CREATE TABLE IF NOT EXISTS
newauthor(aut_id varchar(8) NOT NULL ,
aut_name varchar(50) NOT NULL,
country varchar(25) NOT NULL,
home_city varchar(25) NOT NULL,
UNIQUE (aut_id));
```

नीचे दी गई तस्वीर Table की संरचना को दिखाती है।

	Field	Type	Collation	Attributes	Null	Default	Extra	Acti
☐	aut_id	varchar(8)	latin1_swedish_ci		No	None		
☐	aut_name	varchar(50)	latin1_swedish_ci		No	None		
☐	country	varchar(25)	latin1_swedish_ci		No	None		
☐	home_city	varchar(25)	latin1_swedish_ci		No	None		

Indexes:

Action	Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
	aut_id	BTREE	Yes	No	aut_id	0	A		

3.4.4 MySQL CREATE TABLE with DEFAULT CONSTRAINT

एक Table बनाते समय, MySQL □पको कॉलमों को DEFAULT CONSTRAINTS असाइन करने की अनुमति देता है। DEFAULT का उपयोग कॉलम के लिए डिफ़ॉल्ट मान सेट करने के लिए किया जाता है और इसे DEFAULT default_value का उपयोग करके लागू किया जाता है; जहाँ default_value कॉलम के लिए default मान है।

उदाहरण

नीचे दिया गया MySQL स्टेटमेंट 'pub_id' कॉलम पर PRIMARY KEY के साथ एक TABLE 'newpublisher' बनाएगा, जो country एवं pub_city पर एक C ECK constraint तार्किक ऑपरेटरों के साथ , और pub_id, pub_name, pub_city और country कॉलम के लिए एक डिफ़ॉल्ट मान होगा।

MySQL स्टेटमेंट में pub_id, pub_name, pub_city कॉलम के लिए डिफ़ॉल्ट मान white space के साथ सेट करेगा और country कॉलम के लिए डिफ़ॉल्ट मान के रूप में 'India' भी निर्धारित होता है।

यहाँ नीचे statement दिया गया है।

```
CREATE TABLE IF NOT EXISTS newpublisher

(pub_id varchar(8) NOT NULL UNIQUE DEFAULT " ,

pub_name varchar(50) NOT NULL  DEFAULT " ,

pub_city varchar(25) NOT NULL  DEFAULT " ,

country varchar(25) NOT NULL DEFAULT 'India',

country_office varchar(25) ,

no_of_branch int(3),

estd date

C ECK ((country='India' AND pub_city='Mumbai')

OR (country='India' AND pub_city='New Delhi')) ,

PRIMARY KEY (pub_id));
```

3.4.5 MySQL प्राथमिक कुंजी अनुरूप (PRIMARY KEY CONSTRAINT)

□मतौर पर, एक TABLE में कॉलम या कॉलम का संयोजन होता है जिसमें TABLE में प्रत्येक पंक्ति को विशिष्ट रूप से पहचानने के लिए उपयोग किए जाने वाले मान होते हैं। यह कॉलम या कॉलम का संयोजन PRIMARY KEY कहलाता है और TABLE बनाते समय एक PRIMARY KEY CONSTRAINT को परिभाषित करके बनाया जा सकता है। एक Table में केवल एक प्राथमिक कुंजी हो सकती है। एक प्राथमिक कुंजी स्तंभ में NULL मान नहीं हो सकते।

उदाहरण

नीचे दिया गया MySQL स्टेटमेंट एक TABLE 'न्यूओथोर' बनाएगा जिसमें PRIMARY KEY कॉलम aut_id पर सेट होगी।

```
CREATE TABLE IF NOT EXISTS
newauthor(aut_id varchar(8) NOT NULL ,
aut_name varchar(50) NOT NULL,
country varchar(25) NOT NULL,
home_city varchar(25) NOT NULL,
PRIMARY KEY (aut_id));
```

3.4.6 MySQL एक कॉलम पर TABLE प्राथमिक कुंजी का निर्माण करते हैं: (CREATE TABLE PRIMARY KEY CONSTRAINT on single column)

इस विषय में, हमने चर्चा की है कि TABLE के एक कॉलम पर एक PRIMARY KEY CONSTRAINT कैसे सेट करें।

उदाहरण

नीचे दिया गया MySQL स्टेटमेंट एक TABLE 'newauthor' बनाएगा, जिसमें PRIMARY KEY aut_id कॉलम पर सेट होगी। ध्यान दें कि यहाँ PRIMARY KEY कीवर्ड का उपयोग कॉलम परिभाषा के भीतर किया गया है।

```
CREATE TABLE IF NOT EXISTS
newauthor(aut_id varchar(8) NOT NULL  PRIMARY KEY,
aut_name varchar(50) NOT NULL,
country varchar(25) NOT NULL,
home_city varchar(25) NOT NULL);
```

3.4.7. MY SQL TABLE PRIMARY KEY on multiple columns

MySQL □पको किसी Table के कई स्तंभों पर प्राथमिक कुंजी सेट करने की अनुमति देता है। ऐसा करने से □पको कई कॉलम पर काम करने कि अनुमति मिलती है एकल इकाई की तरह सेट करने में टेबल के लिए |

उदाहरण

नीचे दिया गया MySQL स्टेटमेंट एक TABLE ' newauthor ' बनाएगा जिसमें PRIMARY KEY को aut_id और home_city कॉलम के संयोजन के साथ सेट किया गया है।

```
CREATE TABLE IF NOT EXISTS
newauthor(aut_id varchar(8) NOT NULL ,
aut_name varchar(50) NOT NULL,
country varchar(25) NOT NULL,
home_city varchar(25) NOT NULL,
PRIMARY KEY (aut_id, home_city));
```

3.4.8 MySQL FOREIGN KEY CONSTRAINT के साथ Table का निर्माण

MySQL Table बनाते समय (या संशोधित) करते हुए, □प Table के एक स्तंभ के लिए एक FOREIGN KEY (विदेशी कुंजी) CONSTRAINT सेट कर सकते हैं। एक विदेशी कुंजी एक स्तंभ या कई स्तंभों का संयोजन है जिसका उपयोग दो Table में डेटा के बीच एक लिंक सेट करने के लिए किया जा सकता है। किसी Table का प्राथमिक कुंजी डेटा अखंडता को बढ़ाने के लिए किसी अन्य Table के FOREIGN कुंजी से जुडी हुई होती है।

Syntax :

FOREIGN KEY [column list] REFERENCES [primary key table] ([column list]);

Arguments

Name	Description
column list	A list of the columns on which FOREIGN KEY is to be set.
REFERENCES	Keyword.
primary key table	Table name which contains the PRIMARY KEY.

Example:

यदि □प निचे दिये गये कार्य करना कहते हैं :

एक नयी टेबल 'newbook_mast' का सृजन करना |

PRIMARY KEY 'newbook_mast' नमक टेबल की 'book_id' होगी |

FOREIGN KEY 'newbook_mast' नमक टेबल की 'aut_id' होगी |

'aut_id' जो है , वह PRIMARY KEY है टेबल 'newauthor' की |

'aut_id' जो कि FOREIGN KEY है टेबल 'newbook_mast' की, वह 'newauthor' टेबल कि PRIMARY KEY 'aut_id' को पॉइंट करेगी |

इसका मतलब 'aut_id' जो 'newauthor' टेबल में मौजूद है , केवल वही ऑथर्स 'newbook_mast' में होगे |

उपरोक्त कार्यों के लिए नीचे MySQL स्टेटमेंट दिया गया है:

```
CREATE TABLE IF NOT EXISTS newbook_mast
(book_id varchar(15) NOT NULL PRIMARY KEY,
book_name varchar(50)  ,
isbn_no varchar(15)  NOT NULL  ,
cate_id varchar(8)  ,
aut_id varchar(8) ,
pub_id varchar(8) ,
dt_of_pub date ,
pub_lang varchar(15) ,
no_page decimal(5,0) ,
book_price decimal(8,2) ,
FOREIGN KEY (aut_id) REFERENCES newauthor(aut_id));
```

3.5 Table का अद्यतन करना (UPDATING THE TABLE):

Table की संरचना को अद्यतन (Updating) करने के लिए 'ALTER' कमांड का उपयोग किया जाता है। किसी Table को जोड़ने, स्तंभ को संशोधित करने, स्तंभ को छोड़ने, स्तंभ का नाम बदलने या किसी Table का नाम बदलने के लिए ALTER TABLE कथन का उपयोग किया जाता है।

3.5.1 Table में कॉलम जोड़ें (Add column in table)

Syntax:

MySQL में Table में एक स्तंभ जोड़ने के लिए सिंटेक्स (वैकल्पिक Table विवरण का उपयोग करके) है:

ALTER TABLE table_name

ADD new_column_name column_definition

[FIRST | AFTER column_name];

Arguments:

Name	Description
table_name	The name of the table to modify. टेबल का नाम जिसे modify करना है.
new_column_name	The name of the new column to add to the table. नए कॉलम का नाम जिसे टेबल में जोड़ना है.
column_definition	The datatype and definition of the column (NULL or NOT NULL, etc). कॉलम का डाटाटाइप तथा परिभाषा ((NULL या NOT NULL, etc).
FIRST | AFTER column_name	वैकल्पिक। यह MySQL को बताता है कि Table में स्तंभ कहाँ बनाना है। यदि यह पैरामीटर निर्दिष्ट नहीं है, तो Table के अंत में नया कॉलम जोड़ दिया जाता है।

उदाहरण

□इए एक उदाहरण देखें जो दिखाता है कि कैसे एक MySQL Table में ALTER TABLE स्टेटमेंट का उपयोग करके एक कॉलम जोड़ा जाए।

उदाहरण के लिए:

ALTER TABLE contacts

ADD last_name varchar(40) NOT NULL

AFTER contact_id;

यह MySQL ALTER TABLE उदाहरण, contacts Table में last_name नामक एक कॉलम जोड़ देगा। यह एक NOT NULL कॉलम के रूप में बनाया जाएगा और Table में contact_id फ़ील्ड के बाद दिखाई देगा।

3.5.2 Table में कई कॉलम जोड़ें (Add multiple columns in table):

syntax:

MySQL में एक TABLE में कई कॉलम जोड़ने का सिंटैक्स (ALTER TABLE स्टेटमेंट का उपयोग करके) है:

```
ALTER TABLE table_name
  ADD new_column_name column_definition
    [ FIRST | AFTER column_name ],
  ADD new_column_name column_definition
    [ FIRST | AFTER column_name ],  ...
;
```

Arguments:

Name	Description
table_name	The name of the table to modify. टेबल का नाम जिसे modify करना है.
new_column_name	The name of the new column to add to the table. नए कॉलम का नाम जिसे टेबल में जोड़ना है.
column_definition	The datatype and definition of the column (NULL or NOT NULL, etc). कॉलम का डाटाटाइप तथा परिभाषा ((NULL या NOT NULL, etc).

FIRST | AFTER column_name	वैकल्पिक। यह MySQL बताता है कि स्तंभ बनाने के लिए Table में कहाँ है। यदि यह पैरामीटर निर्दिष्ट नहीं है, तो Table के अंत में नया कॉलम जोड़ा जाएगा।

उदाहरण

□इए एक उदाहरण देखें जो दिखाता है कि कैसे ALTER TABLE स्टेटमेंट का उपयोग करके MySQL TABLE में कई कॉलम जोड़े जा सकते हैं।

उदाहरण के लिए:

```
ALTER TABLE contacts
  ADD last_name varchar(40) NOT NULL
    AFTER contact_id,
  ADD first_name varchar(35) NULL
    AFTER last_name;
```

यह वैकल्पिक Table उदाहरण संपर्क Table में दो कॉलम जोड़ देगा - last_name और first_name।

Last_name फ़ील्ड को varchar (40) NOT NULL कॉलम के रूप में बनाया जाएगा और Table में contact_id कॉलम के बाद दिखाई देगा। First_name कॉलम को varchar (35) NULL कॉलम के रूप में बनाया जाएगा और Table में last_name कॉलम के बाद दिखाई देगा।

3.5.3 Table में स्तंभ संशोधित करें (Modify column in table):

Syntax:

MySQL में एक Table में एक स्तंभ को संशोधित करने के लिए सिंटेक्स (वैकल्पिक Table विवरण का उपयोग करके) है:

```
ALTER TABLE table_name
  MODIFY column_name column_definition
    [ FIRST | AFTER column_name ];
```

Arguments:

Name	Description
table_name	The name of the table to modify. टेबल का नाम जिसे modify करना है.
new_column_name	The name of the new column to add to the table. नए कॉलम का नाम जिसे टेबल में जोड़ना है.
column_definition	The datatype and definition of the column (NULL or NOT NULL, etc). कॉलम का डाटाटाइप तथा परिभाषा ((NULL या NOT NULL, etc).
FIRST | AFTER column_name	वैकल्पिक। यह MySQL को बताता है कि स्तंभ की स्थिति के लिए Table में, यदि □प इसकी स्थिति बदलना चाहते हैं।

उदाहरण

□इए एक उदाहरण देखें जो दिखाता है कि कैसे एक MySQL Table में एक स्तंभ को संशोधित किया जाता है ALTER TABLE स्टेटमेंट का उपयोग करके।

उदाहरण के लिए:

```
ALTER TABLE contacts
  MODIFY last_name varchar(50) NULL;
```

यह ALTER TABLE उदाहरण last_name नामक स्तंभ को varchar (50) के डेटा प्रकार के रूप में संशोधित करेगा और स्तंभ को NULL मानों की अनुमति देने के लिए बाध्य करेगा।

3.5.4 Table में कई कॉलम संशोधित करें (Modify Multiple columns in table):

Syntax:

MySQL में एक TABLE में कई कॉलम को संशोधित करने का सिंटैक्स है (ALTER TABLE स्टेटमेंट का उपयोग करके):

ALTER TABLE table_name

MODIFY column_name column_definition

[FIRST | AFTER column_name],

MODIFY column_name column_definition

[FIRST | AFTER column_name], ... ;

Arguments:

Name	Description
table_name	The name of the table to modify. टेबल का नाम जिसे modify करना है.
new_column_name	The name of the new column to add to the table. नए कॉलम का नाम जिसे टेबल में जोड़ना है.
column_definition	The datatype and definition of the column (NULL or NOT NULL, etc). कॉलम का डाटाटाइप तथा परिभाषा ((NULL या NOT NULL, etc).
FIRST | AFTER column_name	वैकल्पिक। यह MySQL को बताता है कि स्तंभ की स्थिति के लिए Table में, यदि □प इसकी स्थिति बदलना चाहते हैं।

उदाहरण

□इए एक उदाहरण देखें जो यह बताता है कि ALTER TABLE स्टेटमेंट का उपयोग करके MySQL TABLE में कई कॉलम को कैसे संशोधित किया जाए।

उदाहरण के लिए:

ALTER TABLE contacts

MODIFY last_name varchar(55) NULL

AFTER contact_type,

MODIFY first_name varchar(30) NOT NULL;

यह वैकल्पिक Table उदाहरण, contacts Table में दो कॉलम - last_name और first_name को संशोधित करेगा।

Last_name फ़ील्ड को varchar (55) NULL कॉलम में बदल दिया जाएगा और Table में contact_type कॉलम के बाद दिखाई देगा। First_name कॉलम को एक varchar (30) NOT NULL कॉलम में बदला जाएगा (और contacts Table परिभाषा में स्थिति नहीं बदलेगी, क्योंकि कोई FIRST नहीं है। AFTER निर्दिष्ट)।

3.5.5 Table में स्तंभ ड्रॉप करें

Syntax:

MySQL में एक Table में एक कॉलम को छोड़ने का सिंटैक्स (ALTER TABLE स्टेटमेंट का उपयोग करके) है:

ALTER TABLE table_name

DROP COLUMN column_name;

Arguments:

Name	Description
table_name	The name of the table to modify. टेबल का नाम जिसे modify करना है.
column_name	The name of the column to delete from the table. कॉलम का नाम जिसे टेबल से हटाना है.

उदाहरण

□इए एक उदाहरण देखें जो यह बताता है कि माई एसक्यूएल Table में एक स्तंभ को कैसे छोड़ना/ हटाना है।

उदाहरण के लिए:

ALTER TABLE contacts

DROP COLUMN contact_type;

इस Table के उदाहरण में contacts नामक Table से contact_type नामक कॉलम ड्रॉप हो जाएगा।

3.5.6 Table में नाम बदलें (Rename column in table):

Syntax:

MySQL में एक Table में एक स्तंभ का नाम बदलने के लिए सिंटेक्स (वैकल्पिक Table विवरण का उपयोग करके) है:

ALTER TABLE table_name

C ANGE COLUMN old_name new_name

column_definition

[FIRST | AFTER column_name]

Arguments:

Name	Description
table_name	The name of the table to modify. टेबल का नाम जिसे modify करना है.
old_name	The column to rename. कॉलम का नाम जिसे बदलना है.
new_name	The new name for the column. कॉलम का नया नाम.
column_definition	कॉलम का डाटाटाइप तथा परिभाषा ((NULL या NOT NULL, etc). □पको कॉलम की परिभाषा निर्दिष्ट करनी होगी जब □प कॉलम को

	rename करेंगे , भले ही वह बदले नहीं।
FIRST | AFTER column_name	वैकल्पिक। यह MySQL को बताता है कि स्तंभ की स्थिति के लिए Table में, यदि □प इसकी स्थिति बदलना चाहते हैं।

उदाहरण

□इए एक उदाहरण देखें जो दिखाता है कि कैसे एक MySQL Table में ALTER TABLE स्टेटमेंट का उपयोग करके एक कॉलम का नाम बदला जाता है।

उदाहरण के लिए:

ALTER TABLE contacts

C̈ ANGE COLUMN contact_type ctype

varchar(20) NOT NULL;

यह MySQL ALTER TABLE का उदाहरण, के contact_type नामक कॉलम को ctype नाम में बदलेगा। कॉलम को एक varchar (20) NOT NULL कॉलम के रूप में परिभाषित किया जाएगा।

3.5.7 Table का नाम बदलें (Rename table):

Syntax:

MySQL में Table का नाम बदलने का सिंटैक्स है:

ALTER TABLE table_name

RENAME TO new_table_name;

Arguments:

Name	Description
table_name	The table to rename.

	टेबल जिसका नाम बदलना है.
new_table_name	The new table name to use. टेबल न नया नाम जो उपयोग करना है.

उदाहरण

□इए एक उदाहरण देखें जो दिखाता है कि कैसे MySQL में एक Table का नाम बदलने के लिए ALTER TABLE स्टेटमेंट का उपयोग किया जाता है।

उदाहरण के लिए:

```
ALTER TABLE contacts
  RENAME TO people;
```

यह "ALTER TABLE" उदाहरण contacts table का नाम बदलकर people table कर देगा

यह ALTER TABLE उदाहरण, contacts Table का नाम बदल कर 'people' कर देगा।

3.5.8 ड्रॉप TABLE स्टेटमेंट (DROP TABLE Statement):

यह बताता है कि, सिंटैक्स और उदाहरणों के साथ MySQL DROP TABLE स्टेटमेंट का उपयोग कैसे करें।

विवरण (Discription):

MySQL DROP TABLE स्टेटमेंट □पको MySQL डेटाबेस से एक TABLE को हटाने की अनुमति देता है।

syntax:

अपने सरलतम रूप में, MySQL में DROP TABLE स्टेटमेंट के लिए सिंटैक्स है:

```
DROP TABLE table_name;
```

हालाँकि, MySQL DROP TABLE स्टेटमेंट के लिए पूरा सिंटैक्स है:

DROP [TEMPORARY] TABLE [IF EXISTS]

table_name1, table_name2, ...

[RESTRICT | CASCADE];

Parameters or Arguments

Name	Description
TEMPORARY	वैकल्पिक। यह निर्दिष्ट करता है कि केवल अस्थायी Table को DROP TABLE विवरण द्वारा हटा दिया जाना चाहिए।
Table name	डेटाबेस से हटाने के लिए Table का नाम।
table_name1, table_name2	DROP TABLE स्टेटमेंट में एक से अधिक TABLE को डेटाबेस से हटाना है।
IF EXISTS	वैकल्पिक। यदि निर्दिष्ट किया जाता है, यदि TABLE में से कोई एक TABLE मौजूद नहीं है तो DROP TABLE स्टेटमेंट में error नहीं होगी |
RESTRICT	वैकल्पिक। इसका DROP TABLE स्टेटमेंट पर कोई प्रभाव नहीं है, लेकिन टेबल्स को अलग-अलग डेटाबेस में पोर्ट करना □सान बनाने के लिए सिंटैक्स में शामिल है।
CASCADE	वैकल्पिक। इसका DROP TABLE स्टेटमेंट पर कोई प्रभाव नहीं है, लेकिन टेबल्स को अलग-अलग डेटाबेस में पोर्ट करना □सान बनाने के लिए सिंटैक्स में शामिल है।

3.6 डाटा इन्सर्ट करना (INSERTING DATA):

डेटाबेस सिस्टम का मुख्य लक्ष्य tables में डेटा स्टोर करना है। डेटा को □मतौर पर डेटाबेस के शीर्ष पर चलने वाले एप्लिकेशन प्रोग्राम द्वारा □पूर्ति की जाती है। उस छोर की ओर, SQL में INSERT कमांड है जिसका उपयोग डेटा को तालिका में संग्रहीत करने के लिए किया जाता है। INSERT कमांड डेटा स्टोर करने के लिए तालिका में एक नई पंक्ति (row) बनाता है।

मूल syntax:

नीचे दिखाए गए SQL INSERT कमांड के मूल सिंटैक्स को देखें।

INSERT INTO `table_name` (column_1, column_2, ...) VALUES (value_1, value_2, ...);

यहाँ

• INSERT INTO `table_name` कमांड है जो MySQL सर्वर को` table_name` नामक तालिका में नई पंक्ति जोड़ने के लिए कहता है।

• (column_1, column_2, ...) नई पंक्ति में अद्यतन किए जाने वाले कॉलम निर्दिष्ट करता है

• मान (value_1, value_2, ...) नई पंक्ति में जोड़े जाने वाले मानों को निर्दिष्ट करता है

नई तालिका में सम्मिलित किए जाने वाले डेटा मानों की □पूर्ति करते समय, विभिन्न डेटा प्रकारों के साथ काम करते समय निम्नलिखित पर विचार किया जाना चाहिए।

• स्ट्रिंग डेटा प्रकार (**String data types**) - सभी स्ट्रिंग मान एकल उद्धरणों में संलग्न होने चाहिए।

• संख्यात्मक डेटा प्रकार (**Numeric data types**) - सभी संख्यात्मक मूल्यों को एकल या दोहरे उद्धरणों में संलग्न किए बिना सीधे □पूर्ति की जानी चाहिए।

• दिनांक डेटा प्रकार (**Date data types**) - 'YYYY-MM-DD' प्रारूप में एकल उद्धरण में तारीख मान संलग्न करें।

उदाहरण:

मान लीजिए कि हमारे पास नए पुस्तकालय सदस्यों की निम्नलिखित सूची है जिन्हें डेटाबेस में जोड़ने की □वश्यकता है।

Full names	**Date of Birth**	**gender**	**Physical address**	**Postal address**	**Contact number**	**Email Address**
Ravi		Male	New bazar		0845738767	
Rajesh		Male	South avenu		0976736763	
Karim		Male	Lalita nagar		0938867763	

Shivam	14/02/1984	Male	Kamla park		0987636553	
Yash	24/08/1981	Male	Shahi gali	P.O.Box 4234	0987786553	lwolowitz@email.me

एक-एक करके डेटा को INSERT करें। हम Ravi के साथ शुरु□त करेंगे। हम Contact number को एक संख्यात्मक डेटा प्रकार के रूप में मानेंगे और एकल उद्धरणों में संख्या को संलग्न नहीं करेंगे।

INSERT INTO `members` (`full_names`, `gender`, `physical_address`, `contact_number`) VALUES ('Ravi', 'Male', 'New bazar', 0845738767);

उपरोक्त स्क्रिप्ट को निष्पादित करने पर Ravi के Contact number से 0 को छोड़ देता है। इसका कारण यह है कि मान को एक संख्यात्मक मान के रूप में माना जाएगा और शुरु□त में शून्य (0) गिरा दिया जाता है क्योंकि यह महत्वपूर्ण नहीं है।

ऐसी समस्याओं से बचने के लिए, मान को नीचे दिखाए गए अनुसार एकल उद्धरणों में संलग्न किया जाना चाहिए –

INSERT INTO `members` (`full_names`, `gender`, `physical_address`, `contact_number`) VALUES (' Rajesh ', 'Male', ' South avenu ', '0976736763');

कॉलम के क्रम को बदलने से INSERT क्वेरी पर कोई प्रभाव नहीं पड़ता है क्योंकि सही मानों को सही कॉलम में मैप किया गया है।

नीचे दिखाया गया क्वेरी उपरोक्त बिंदु प्रदर्शित करता है।

INSERT INTO `members` (`contact_number`, `gender`, `full_names`, `physical_address`) VALUES ('0938867763','Male',' Karim ',' Lalita nagar ');

नोट: उपरोक्त क्वेरी जन्म तिथि को छोड़ देती है, डिफ़ॉल्ट रूप से MySQL INSERT क्वेरी में छोड़े गए स्तंभों में NULL मान सम्मिलित करेगा।

□इए अब शिवम के लिए रिकॉर्ड डालें जिसमें जन्म की तारीख है। दिनांक मान को 'YYYY-MM-DD' प्रारूप का उपयोग करके एकल उद्धरणों में संलग्न किया जाना चाहिए।

INSERT INTO `members` (`full_names`, `date_of_birth`, `gender`, `physical_address`, `contact_number`) VALUES ('Shivam','1984-02-14', 'Male', 'Kamla park', '0987636553');

उपरोक्त सभी query ने कॉलम निर्दिष्ट किए और उन्हें सम्मिलित विवरण में मानों के लिए मैप किया। यदि हम तालिका में सभी स्तंभों के लिए मान प्रदान कर रहे हैं, तो हम सम्मिलित क्वेरी से कॉलम को छोड़ सकते हैं।

उदाहरण:-

INSERT INTO `members` VALUES ('Yash','Male','1981-08-24','Shahi gali', 'P.O.Box 4234', '0987786553', 'lwolowitz@email.me');

सदस्य तालिका में सभी पंक्तियों को देखने के लिए SELECT स्टेटमेंट का उपयोग करते हैं।

SELECT * FROM `members`;

Full names	Date of Birth	Gender	Physical address	Postal address	Contact number	Email Address
Ravi		Male	New bazar		845738767	
Rajesh		Male	South avenu		0976736763	
Karim		Male	Lalita nagar		0938867763	
Shivam	14/02/1984	Male	Kamla park		0987636553	
Yash	24/08/1981	Male	Shahi gali	P.O.Box 4234	0987786553	lwolowitz@email.me

ध्यान दें कि Ravi के लिए Contact number से शून्य (0) contact number से हटा है। अन्य संपर्क नंबरों ने शुरु□त में शून्य (0) को हटाया नहीं है।

3.7 एक MYSQL DATABASE से डेटा प्राप्त करना (RETRIEVING DATA FROM A MYSQL DATABASE):

पहला SQL कमांड जो सीखेंगे, और जिसे सबसे अधिक बार उपयोग करेंगे, वह है 'SELECT'.

एक SELECT कथन SELECT कीवर्ड से शुरू होता है और इसका उपयोग MySQL डेटाबेस टेबल से जानकारी प्राप्त करने के लिए किया जाता है। उपयोगकर्ता को उस तालिका का नाम बताना पड़ता है जिससे से डेटा प्राप्त करना है - FROM keyword का प्रयोग करके - और एक या एक से अधिक स्तंभ, जो उपयोगकर्ता उस तालिका से प्राप्त करना चाहते हैं।

3.7.1 व्यक्तिगत कॉलम पुनः प्राप्त करना (Retrieving Individual Columns)

यदि हम mysql का उपयोग करते हुए निम्नलिखित एसक्यूएल स्टेटमेंट को निष्पादित करते हैं, तो उत्पादित □उटपुट ऐसे दिखाया जाएगा:

```
mysql> SELECT name
    -> FROM customers;
+----------------------------------+
| name                             |
+----------------------------------+
| Presidents Incorporated          |
| Science Corporation              |
| Musicians of America             |
+----------------------------------+
3 rows in set (0.02 sec)
```

3.7.2 एकाधिक कॉलमों को पुनः प्राप्त करना (Retrieving Multiple Columns)

अब हम एक और सरल SELECT कथन की कोशिश करेंगे, इस बार products की table पर। हम 'SELECT' कीवर्ड के बाद कॉलम की एक सूची निर्दिष्ट करके, उन्हें अल्पविराम से अलग करके एक ही क्वेरी में दो कॉलम से मान प्राप्त कर सकते हैं।

```
mysql> SELECT name, price
```

-> FROM products;

```
+-----------------------------+--------+
| name                        | price  |
+-----------------------------+--------+
| Small product               | 5.99   |
| Medium product              | 9.99   |
| Large product               | 15.99  |
+-----------------------------+--------+
```

3 rows in set (0.01 sec)

□उटपुट में कॉलम क्वेरी में दिए गए क्रम में दिखाई देते हैं। पुनर्प्राप्त किए गए डेटा में weight कॉलम जोड़ने के लिए, इसे चयनित कॉलम की सूची के अंत में इस प्रकार जोड़ें:

mysql> SELECT name, price, weight

-> FROM products;

```
+----------------------+-------------+-----------+
| name                 | price       | weight    |
+----------------------+-------------+-----------+
| Small product        | 5.99        |   1.50    |
| Medium produc t      | 9.99        |   4.50    |
| Large product        | 15.99       |   8.00    |
+----------------------+-------------+-----------+
```

3 rows in set (0.00 sec)

3.7.3 सभी स्तम्भों को पुनः प्राप्त करना (Retrieving All Columns)

यदि हम तालिका के प्रत्येक कॉलम से डेटा को पुनः प्राप्त करना चाहते हैं, तो SELECT कीवर्ड के बाद प्रत्येक कॉलम का नाम निर्दिष्ट करने की □वश्यकता नहीं है। MySQL को निर्दिष्ट तालिका से प्रत्येक कॉलम को वापस करने का निर्देश देने के लिए एक सेलेक्ट स्टेटमेंट में कॉलम सूची के स्थान पर तारांकन वर्ण (*) का उपयोग करें।

निम्नलिखित क्वेरी उत्पाद तालिका से प्रत्येक कॉलम और पंक्ति को पुनः प्राप्त करती है:

mysql> SELECT * FROM products;

```
+----------+---------------------+-----------+---------+
| code     | name                | weight    | price   |
```

```
+----------+--------------------+-----------+---------+
| MINI     | Small product      |  1.50     | 5.99    |
| MIDI     | Medium product     |  4.50     | 9.99    |
| MAXI     | Large product      |  8.00     | 15.99   |
+----------+--------------------+-----------+---------+
```

3 rows in set (0.00 sec)

उत्पादित □उटपुट बिल्कुल वैसा ही है, जैसे कि हमने क्वेरी में प्रत्येक कॉलम को नाम से निर्दिष्ट किया था, जैसे:

```
mysql> SELECT code, name, weight, price
    -> FROM products;
```

```
+----------+------------------------+-------------+--------+
| code     | name                   | weight      | price  |
+----------+------------------------+-------------+--------+
| MINI     | Small product          |  1.50       | 5.99   |
| MIDI     | Medium product         |  4.50       | 9.99   |
| MAXI     | Large product          |  8.00       | 15.99  |
+----------+------------------------+-------------+--------+
```

3 rows in set (0.00 sec)

जब SELECT * का उपयोग करते हैं, तो कॉलम उस क्रम में प्रदर्शित होते हैं, जो वे डेटाबेस टेबल में होते हैं - जिस क्रम में कॉलम बनाया गया था, उस क्रम को निर्दिष्ट किया गया था।

3.7.4 परिणामों की सीमित संख्या (Restricting Number of Results)

तालिका से डेटा पुनर्प्राप्त करते समय परिणाम की संख्या को सीमित करना संभव है जो LIMIT कीवर्ड का उपयोग करके सेलेक्ट स्टेटमेंट द्वारा वापस किया जाता है। इस कीवर्ड को एक संख्या के बाद दर्शाया जाता है कि कितनी पंक्तियों को पुनर्प्राप्त किया जाना है:

```
SELECT * FROM product LIMIT 10;
```

उपरोक्त □देश डेटाबेस तालिका से केवल 10 पंक्तियों को पुनः प्राप्त करेगा। □रंभ और अंत दोनों पंक्तियों को एक तालिका में □गे पढ़ने के लिए निर्दिष्ट किया जा सकता है। इसलिए, निम्न उदाहरण तालिका के 15 के माध्यम से पंक्तियों को निकालता है:

SELECT * FROM product LIMIT 10, 15;

3.7.5 परिणामों से डुप्लिकेट मानों को समाप्त करना (Eliminating Duplicate Values from Results)

एक स्तंभ मान, संयोजन के लिए कई पंक्तियों में डुप्लिकेट होने के लिए असामान्य नहीं है। इसका मतलब यह है कि डेटा प्राप्त करते समय, और विशेष रूप से किसी तालिका में एकल कॉलम के लिए डेटा प्राप्त करते समय, परिणाम में डुप्लिकेट मान दिखाई दे सकते हैं। उदाहरण के लिए:

SELECT prod_desc FROM product;

```
+----------------------------+
| prod_desc                  |
+----------------------------+
| CD Writer                  |
| SATA Disk Drive            |
| Cordless Mouse             |
| SATA Disk Drive            |
| Ergonmoc Keyboard          |
| CD Writer                  |
+----------------------------+
```

जैसा कि हम उपरोक्त □उटपुट में देख सकते हैं, हमारे काल्पनिक ऑन-लाइन स्टोर एक से अधिक प्रकार के Disk Drive और CD Writer बेचते हैं। जबकि उत्पाद कोड और नाम अद्वितीय (unique) होने की संभावना है, विवरण में दोहराव है। यदि हम डुप्लिकेट से रहित उत्पाद विवरणों की सूची प्राप्त करना चाहते हैं तो हम DISTINCT कीवर्ड का उपयोग करेंगे:

SELECT DISTINCT prod_desc FROM product;

इसका परिणाम निम्न □उटपुट में होगा:

```
+----------------------------+
| prod_desc                  |
+----------------------------+
| CD Writer                  |
| SATA Disk Drive            |
| Cordless Mouse             |
| Ergonmoc Keyboard          |
+----------------------------+
```

3.8 MySQL डाटाबेस से लिये गये डाटा कि Sorting करना:

3.8.1 एक सेलेक्ट स्टेटमेंट से डेटा सॉर्ट करना

एक सेलेक्ट स्टेटमेंट उस टेबल से डेटा को उसी क्रम में निकालेगा जिस क्रम में उसे जोड़ा गया था। यह, हालाँकि, डेटाबेस तालिका अद्यतन(update) और विलोपन(delete) के अधीन किया गया है, तो बदल सकता है। काफी सरलता से, उस सटीक क्रम की भविष्यवाणी करने का कोई तरीका नहीं है जिसमें डेटा पुनर्प्राप्त किया जाएगा। इस कारण से SQL ORDER BY कीवर्ड प्रदान करता है जो हमें डेटा को सॉर्ट करने में सक्षम बनाता है क्योंकि यह SELECT स्टेटमेंट द्वारा प्राप्त होता है।

चलो एक नमूना डेटाबेस पर एक पुनर्प्राप्ति प्रदर्शन करके शुरू करते हैं:

```
USE MySampleDB;

SELECT * FROM product;
```

जब इस कमांड को निष्पादित किया जाता है, तो हम तालिका में संग्रहीत सभी कॉलम को पुनः प्राप्त करते हैं। हम देखते हैं कि परिणाम p_code कॉलम के क्रम में सूचीबद्ध थे। शायद ही □श्चर्य की बात है कि यह एक ऑटो-इन्क्रीमेंट मूल्य है और उस क्रम को दर्शाता है जिसमें डेटा जोड़ा गया था:

p_code	prod_name	prod_desc
1	CD-RW Model 4543	CD Writer
2	EasyTech Mouse 7632	Cordless Mouse
3	WildTech 250Gb 1700	SATA Disk Drive
4	Microsoft 10-20 Keyboard	Ergonmoc Keyboard
5	Apple iPhone 8Gb	Smart Phone
6	Moto Razr	Mobile Phone
7	Dell XPS 400	Desktop PC
8	Apple iPod Touch	Portable Music Player
9	Kensington Ci20	Optical Mouse
10	Buffalo AirStation Turbo G	Wireless Ethernet Bridge

मान लीजिए कि □प उत्पाद विवरण वर्णमाला क्रम में क्रमबद्ध करना चाहते थे। यहीं पर ORDER BY कीवर्ड प्ले में □ते हैं। ORDER BY का उपयोग करने का सिंटैक्स इस प्रकार है:

SELECT column(s) FROM table name ORDER By column name

हमारे उदाहरण के साथ जारी रखने के लिए, इसलिए, हम उत्पाद विवरण को निम्नानुसार क्रमबद्ध कर सकते हैं:

SELECT prod_desc FROM product ORDER BY prod_desc;

इस स्थिति में, परिणाम prod_desc कॉलम के अनुसार हल किए जाएंगे:

```
+-----------------------------------+
| prod_desc                         |
+-----------------------------------+
| CD Writer                         |
| Cordless Mouse                    |
| Desktop PC                        |
| Ergonmoc Keyboard                 |
| Mobile Phone                      |
| Optical Mouse                     |
| Portable Music Player             |
| SATA Disk Drive                   |
| Smart Phone                       |
| Wireless Ethernet Bridge          |
+-----------------------------------+
```

डेटा पुनर्प्राप्त करते समय चयनित कॉलम पर सॉर्ट करना □वश्यक नहीं है। यह किसी तालिका में किसी भी स्तंभ पर छाँटने के लिए पूरी तरह से मान्य है, भले ही वह एक सेलेक्ट स्टेटमेंट में संदर्भित कॉलम में से एक हो।

3.8.2 एकाधिक स्तंभों पर छंटनी (Sorting on Multiple Columns)

ORDER BY कई कॉलम पर सॉर्ट करने का एक तरीका भी प्रदान करता है। सॉर्ट के लिए उपयोग किए जाने वाले कॉलम ORDER BY कीवर्ड्स के बाद निर्दिष्ट किए जाते हैं और कॉमा द्वारा अलग किए जाते हैं। उदाहरण के लिए, हमारी तालिका में product name और product description के संयोजन पर छाँटने के लिए हम निम्नलिखित एसक्यूएल कथन को निष्पादित करेंगे:

SELECT * FROM product ORDER BY prod_name, prod_desc;

यह निम्नलिखित डेटा ऑर्डर करने में परिणाम देता है:

p_code	prod_name	prod_desc
5	Apple iPhone 8Gb	Smart Phone
8	Apple iPod Touch	Portable Music Player
10	Buffalo AirStation Turbo G	Wireless Ethernet Bridge
1	CD-RW Model 4543	CD Writer
7	Dell XPS 400	Desktop PC
2	EasyTech Mouse 7632	Cordless Mouse
9	Kensington Ci20	Optical Mouse
4	Microsoft 10-20 Keyboard	Ergonmoc Keyboard
6	Moto Razr	Mobile Phone
3	WildTech 250Gb 1700	SATA Disk Drive

10 rows in set (0.00 sec)

3.8.3 अवरोही क्रम में डेटा छाँटना (Sorting Data in Descending Order)

अब तक, हमने □रोही क्रम में हमारे सभी MySQL डेटाबेस प्रश्नों को बढ़ते क्रम में हल कर दिया है। मान लीजिए, किसी को सबसे महंगी से शुरू होने वाली वस्तुओं की सूची के लिए कीमतों को देखने की जरूरत है। स्पष्ट रूप से मानक ORDER BY विनिर्देश इस □वश्यकता को संबोधित नहीं करेगा। सौभाग्य से SQL में DESC कीवर्ड शामिल है जो यह बताता है कि एक सेलेक्ट स्टेटमेंट के परिणाम अवरोही क्रम(Descending order) में सॉर्ट किए जाने हैं। उदाहरण के लिए, अवरोही क्रम में उत्पाद नामों को क्रमबद्ध करने के लिए:

mysql> SELECT prod_name FROM product ORDER BY prod_name DESC;

```
+-----------------------------------+
| prod_name                         |
+-----------------------------------+
| WildTech 250Gb 1700               |
| Moto Razr                         |
| Microsoft 10-20 Keyboard          |
| Kensington Ci20                   |
| EasyTech Mouse 7632               |
| Dell XPS 400                      |
| CD-RW Model 4543                  |
| Buffalo AirStation Turbo G        |
| Apple iPod Touch                  |
```

| Apple iPhone 8Gb |
+----------------------------------+
10 rows in set (0.00 sec)

3.9 MySQL डेटा को फ़िल्टर करने के लिए 'WHERE' का उपयोग करना

डेटा पुनर्प्राप्त करते समय सबसे महत्वपूर्ण □वश्यकताओं में से एक उस डेटा को फ़िल्टर करने में सक्षम होना है ताकि वापस लौटी एकमात्र पंक्तियाँ वो मिले जो निर्दिष्ट खोज मानदंडों से मेल खाती हैं। यह MySQL में सिलेक्ट स्टेटमेंट के साथ W ERE क्लॉज का उपयोग करके हासिल किया गया जा सकता है।

W ERE क्लॉज का सिंटैक्स इस प्रकार है:

```
SELECT column(s) FROM   table WHERE column = value  ;
```

कथन का पहला भाग किसी भी नियमित चयन कथन की तरह दिखता है। हालांकि क्लॉज चीजों को बदलता है, लेकिन W ERE के बाद हमें उस कॉलम को निर्दिष्ट करने की □वश्यकता है जिस पर खोज मानदंड □धारित हैं, उसके बाद एक ऑपरेटर जो प्रदर्शन करता है तुलना के प्रकार को निर्दिष्ट करता है (इस मामले में हम समानता की तलाश कर रहे हैं) और अंत में वह मान जिस पर कॉलम मेल खाते हुए होना चाहिये।

मान लें कि हमारे पास एक डेटाबेस तालिका है जिसमें prod_code, prod_name, prod_desc और prod_price कॉलम हैं, एक मानक चयन कथन निम्न □उटपुट उत्पन्न कर सकता है:

```
mysql> SELECT * FROM products;
```

p_code	prod_name	prod_desc	prod_price
1	WildTech 250Gb 1700	SATA Disk Drive	120
2	Moto Razr	Mobile Phone	200
3	Microsoft 10-20 Keyboard	Ergonomic Keyboard	49
4	EasyTech Mouse 7632	Cordless Mouse	49
5	Dell XPS 400	Desktop PC	999
6	Buffalo AirStation Turbo G	Wireless Ethernet Bridge	60

7	Apple iPod Touch	Portable Music Player	199
8	Apple iPhone 8Gb	Smart Phone	399

8 rows in set (0.00 sec)

यह ठीक है अगर हम वास्तव में अपनी तालिका की प्रत्येक पंक्ति की सूची चाहते हैं। हालांकि, यह अधिक संभावना है कि हम वास्तव में तालिका में डेटा का सबसेट चाहते हैं। उदाहरण के लिए, यदि हम केवल उन पंक्तियों को पुनः प्राप्त करना चाहते थे, जहां मान एक निश्चित मान से मेल खाता है, तो हम निम्नानुसार SQL कथन का निर्माण कर सकते हैं:

SELECT * FROM products WHERE prod_price = 49;

इस तरह के एक बयान हमारे नमूना डेटाबेस पर निष्पादित होने पर निम्नलिखित □उटपुट में परिणाम होगा:

p_code	prod_name	prod_desc	prod_price
3	Microsoft 10-20 Keyboard	Ergonomic Keyboard	49
4	EasyTech Mouse 7632	Cordless Mouse	49

2 rows in set (0.00 sec)

अब जब हमने समानता (equality) ऑपरेटर (=) को देखा है तो अब हम अन्य W ERE क्लॉज से ऑपरेटरों का पता लगा सकते हैं।

3.9.1. तुलना संचालक

अब तक हमने केवल W ERE क्लॉज इक्वैलिटी ऑपरेटर (=) की समीक्षा की है। यह केवल कई ऑपरेटरों में से एक है जो डेटा पुनर्प्राप्ति को परिष्कृत करने के लिए उपयोग किया जा सकता है:

Operator	Description	
=	Equal to	के बराबर
!=	Not equal to	बराबर नही है
<>	Not equal to	बराबर नही है

>	Greater than	से अधिक
<	Less than	से कम
<=	Less than or equal to	से कम या बराबर
>=	Greater than or equal to	इससे बड़ा या इसके बराबर
BETWEEN x AND y	Falls between the two values x and y	दो मूल्यों x और y के बीच का फॉल्स

हमने संख्यात्मक समानता की जांच करने के लिए पहले से ही समानता (=) ऑपरेटर का उपयोग किया है। हम इस ऑपरेटर का उपयोग अन्य डेटा प्रकारों का मूल्यांकन करने के लिए भी कर सकते हैं, उदाहरण के लिए हम स्ट्रिंग्स की समानता की जांच कर सकते हैं। मान लीजिए कि हमें 8Gb iPhone की कीमत और उत्पाद का विवरण ढूंढना है:

```
SELECT prod_desc, prod_price FROM products WHERE prod_name =
'Apple iPhone 8Gb';
```

उपरोक्त कथन निष्पादन पर निम्नलिखित □उटपुट उत्पन्न करेगा:

prod_desc	prod_price
Smart Phone	399

1 row in set (0.00 sec)

इसी प्रकार (<) ऑपरेटर से कम का उपयोग करके निर्दिष्ट मूल्य सीमा से नीचे के सभी उत्पादों को सूचीबद्ध करना संभव है:

```
mysql> SELECT * from products WHERE prod_price < 200;
```

p_code	prod_name	prod_desc	prod_price
1	WildTech 250Gb 1700	SATA Disk Drive	120
3	Microsoft 10-20 Keyboard	Ergonomic Keyboard	49
4	EasyTech Mouse 7632	Cordless Mouse	49
6	Buffalo AirStation TurboG	Wireless Ethernet Bridge	60
7	Apple iPod Touch	Portable Music Player	199

5 rows in set (0.00 sec)

हम उन डेटा पंक्तियों को भी पुनः प्राप्त कर सकते हैं जहाँ एक मान कुछ मानदंडों से मेल नहीं खाता है। निम्नलिखित उदाहरण उन सभी पंक्तियों को पुनः प्राप्त करेगा जहां मूल्य (! =) 49 के बराबर नहीं है:

SELECT * from products WHERE prod_price != 49;

3.9.2. NULL मान की जाँच करना :

किसी स्तंभ के मान की जांच करने में सक्षम होने के अलावा, उन स्तंभों की जाँच करना भी संभव है जिनमें कोई मूल्य नहीं है। जिन कॉलमों का कोई मूल्य नहीं है, उनमें NULL मान सम्मिलित हैं। □श्चर्य की बात नहीं है, इसलिए, हम NULL के साथ समानता की तलाश करके एक खाली कॉलम की जांच कर सकते हैं (हालांकि इस तथ्य पर ध्यान दें कि हमें अब = चिन्ह के स्थान पर IS का उपयोग करना चाहिए):

SELECT * FROM products WHERE prod_name IS NULL;

जिन पंक्तियों में उत्पाद का नाम नहीं है, उन्हें उपरोक्त चयन विवरण द्वारा पुनः प्राप्त किया जाएगा।

3.9.3. रेंज वैल्यूज़ के भीतर सर्च करना (Searching within Range Values)

ऊपर तालिका में 'BETWEEN ... AND' ऑपरेटर सूचीबद्ध था। इस ऑपरेटर का उद्देश्य उन पंक्तियों का चयन करना है जो निर्दिष्ट ऊपरी और निचली श्रेणियों के बीच □ती हैं। उदाहरण के लिए, हम सभी उत्पादों को सूचीबद्ध कर सकते हैं जिनकी कीमत 100 और 200 के बीच है।

mysql> SELECT * from products WHERE prod_price BETWEEN 100 AND 200;

p_code	prod_name	prod_desc	prod_price
1	WildTech 250Gb 1700	SATA Disk Drive	120
2	Moto Razr	Mobile Phone	200
7	Apple iPod Touch	Portable Music Player	199

3 rows in set (0.02 sec)

3.10. MYSQL FUNCTIONS AND OPERATORS:

3.10.1. The MySQL Aggregate Functions

MySQL निम्नलिखित aggregate functions को supports करता है :

Function	Description
AVG()	Returns the average of the values in the selected column चयनित कॉलम में मानों का औसत लौटाता है
COUNT()	Returns the number of rows returned for a selection चयन के लिए लौटी पंक्तियों की संख्या देता है
MAX()	Returns the maximum value for a column किसी स्तंभ के लिए अधिकतम मान देता है
MIN()	Returns the minimum value of a column किसी स्तंभ का न्यूनतम मान लौटाता है
SUM()	Returns the sum of the values in a specified column किसी निर्दिष्ट स्तंभ में मानों का योग देता है

I) Aggregate functions का उपयोग करना: (Using the Aggregate Functions):

कुल Aggregate Functions को प्रदर्शित करने के प्रयोजनों के लिए हम निम्नलिखित डेटा के साथ एक तालिका का उपयोग करेंगे:

p_code	prod_name	prod_desc	prod_price
1	WildTech 250Gb 1700	SATA Disk Drive	120
2	Moto Razr	Mobile Phone	200
3	Microsoft 10-20 Keyboard	Ergonmoc Keyboard	49
4	EasyTech Mouse 7632	Cordless Mouse	49
5	Dell XPS 400	Desktop PC	999
6	Buffalo AirStation Turbo G	I Wireless Ethernet Bridge	60
7	Apple iPod Touch	Portable Music Player	199
8	Apple iPhone 8Gb	Smart Phone	399

1. MySQL AVG () फ़ंक्शन का उपयोग करना

AVG () फ़ंक्शन एक सेलेक्ट स्टेटमेंट में निर्दिष्ट कॉलम के लिए सभी मानों को एक साथ जोड़ता है और औसत मूल्य पर □ने वाली पंक्तियों की संख्या से इसे विभाजित करता है। परिणाम तब AS उपवाक्य(clause का उपयोग करते हुए किसी अन्य को सौंपा जा सकता है। उदाहरण के लिए, हमारे डेटाबेस में उत्पादों की औसत कीमत का पता लगाने के लिए, और price_avg नामक एक उपनाम के परिणाम को असाइन करें:

mysql> SELECT AVG(prod_price) AS price_ag FROM products;

price_avg
259.375

1 row in set (0.00 sec)

हम W ERE क्लॉज का उपयोग करके औसत गणना में प्रयुक्त पंक्तियों के बारे में भी चयनात्मक हो सकते हैं:

mysql> SELECT AVG(prod_price) AS price_avg FROM products
WHERE prod_price BETWEEN 10 and 199;

price_avg
95.4

1 row in set (0.00 sec)

2. MySQL COUNT () फ़ंक्शन का उपयोग करना

MySQL COUNT () फ़ंक्शन उन पंक्तियों की गणना करता है जो एक चयन कथन में निर्दिष्ट फ़िल्टर मानदंडों से मेल खाते हैं। उदाहरण के लिए, हमारे नमूना तालिका में मूल्य के साथ पंक्तियों की संख्या की गणना करने के लिए:

mysql> SELECT COUNT(*) FROM products;

price_ag
8

1 row in set (0.00 sec)

इसी प्रकार, हम एक विशिष्ट मूल्य सीमा के नीचे उत्पादों की संख्या को सूचीबद्ध करने के लिए अपने मानदंड को सीमित कर सकते हैं:

mysql> SELECT COUNT(prod_price) AS low_price_items FROM products WHERE prod_price < 200;

low_price_items
5

1 row in set (0.00 sec)

3. MySQL MAX () फ़ंक्शन का उपयोग करना

MAX () फ़ंक्शन पंक्ति से डेटा लौटाता है जिसमें निर्दिष्ट कॉलम में उच्चतम मूल्य होता है। उदाहरण के लिए हम अपने डेटाबेस तालिका में सबसे महंगा उत्पाद पा सकते हैं:

mysql> SELECT MAX(prod_price) AS max_price FROM products;

Max_price
999

1 row in set (0.00 sec)

4. MySQL MIN () फ़ंक्शन का उपयोग करना

MIN () फ़ंक्शन MAX () फ़ंक्शन के विपरीत कार्य करता है जिसमें यह निर्दिष्ट कॉलम में सबसे कम मूल्य वाली पंक्ति से डेटा लौटाता है। उदाहरण के लिए, हमारी तालिका में कम से कम महंगी वस्तु खोजने के लिए:

mysql> SELECT MIN(prod_price) AS min_price FROM products;

Max_price
49

1 row in set (0.00 sec)

5. SUM () फ़ंक्शन का उपयोग करना

SUM () फ़ंक्शन एक निर्दिष्ट कॉलम में सभी मानों का योग(Total) रिटर्न करता है। इसलिए, तालिका में प्रत्येक □इटम का कुल मूल्य प्राप्त करने के लिए:

mysql> SELECT SUM(prod_price) AS total_price FROM products;

Max_price
2075

1 row in set (0.00 sec)

II) एक साथ कई Aggregate Functions का उपयोग करना: (Using Multiple Aggregate Functions):

चयनित विवरण एक एकल फ़ंक्शन तक ही सीमित नहीं हैं। उदाहरण के लिए, कई कार्यों में कॉल शामिल करना पूरी तरह से मान्य है:

mysql> SELECT MAX(prod_price) AS max_price, MIN(prod_price) AS max_price FROM products;

Max_price	Max_price
999	49

1 row in set (0.00 sec)

3.10.2 MySQL कंट्रोल फ़्लो फ़ंक्शंस

1. CASE ऑपरेटर

MySQL में, CASE स्टेटमेंट का उपयोग किसी संग्रहीत प्रोग्राम में एक जटिल सशर्त निर्माण को लागू करने के लिए किया जाता है।

Syntax:

```
CASE value WHEN [compare_value] THEN
result

[WHEN [compare_value] THEN result ...]

[ELSE result]

END
```

OR

```
CASE WHEN [condition] THEN result

[WHEN [condition]

THEN result ...]

[ELSE result]

END
```

- पहला सिंटैक्स परिणाम देता है जहाँ Value = Compar_value होती है

• दूसरा सिंटैक्स पहली शर्त के लिए परिणाम देता है जो true है।

• संबंधित SQL कथनों की सूची तब निष्पादित होगी जब कोई खोज स्थिति सत्य का मूल्यांकन करती है।

• ELSE भाग में स्टेटमेंट लिस्ट तब निष्पादित होगी जब कोई खोज स्थिति मेल नहीं खाती।

• यदि ELSE भाग में कोई मिलान मूल्य नहीं मिला है, तो NULL वापस कर दिया जाएगा।

• प्रत्येक स्टेटमेंट लिस्ट में एक या अधिक स्टेटमेंट हो सकते हैं और किसी भी खाली स्टेटमेंट की अनुमति नहीं है।

उदाहरण: MySQL CASE ऑपरेटर

निम्नलिखित कथन में, CASE 1 है, इसलिए "this is case one" लौटा दिया गया है।

कोड:

```
SELECT CASE 1 WHEN 1 THEN 'this is case one'

WHEN 2 THEN 'this is case two'
```

ELSE 'this is not in the case'

END as 'how to execute case statement';

Sample Output:

```
mysql> SELECT CASE 1 WHEN 1 THEN 'this is case one'
    -> WHEN 2 THEN 'this is case two'
    -> ELSE 'this is not in the case'
    -> END as 'how to execute case statement';
+-------------------------------------------+
| how to execute case statement             |
+-------------------------------------------+
| this is case one                          |
+-------------------------------------------+
1 row in set (0.00 sec)
```

2. IF () फ़ंक्शन

MySQL IF () तीन एक्सप्रेशन लेता है और यदि पहला एक्सप्रेशन सत्य है, शून्य नहीं है और NULL नहीं है, तो यह दूसरा एक्सप्रेशन लौटाता है। अन्यथा, यह तीसरी अभिव्यक्ति लौटाता है।

जिस संदर्भ में इसका उपयोग किया जाता है, उसके □धार पर यह या तो संख्यात्मक या स्ट्रिंग मान देता है।

Syntax:

IF(expression ,expr_true, expr_false);

Parameters

Name	Description	Return Type
expression	An expression.	
expr_true	Returns when the condition is TRUE.	एक स्ट्रिंग जब expr_true एक स्ट्रिंग है, एक अस्थायी बिंदु मान जब expr _true एक अस्थायी

		बिंदु मूल्य है और एक पूर्णांक जब expr _true एक पूर्णांक होता है।
expr_false	Returns when the condition is FALSE.	एक स्ट्रिंग जब expr_false एक स्ट्रिंग है , एक अस्थायी-बिंदु मान जब expr _false एक अस्थायी-बिंदु मान होता है और एक पूर्णांक जब expr _false एक पूर्णांक होता है।

Example : MySQL IF() function

निम्नलिखित कथन में चूंकि 1, 3 से कम है, इसलिए IF () तीसरी अभिव्यक्ति लौटाता है, अर्थात गलत।

i.e. false.

Code:

```
SELECT IF(1>3,'true','false');
```

Sample Output:

```
mysql> SELECT IF(1>3,'true','false');

+----------------------------+
|  IF(1>3,'true','false') |
+----------------------------+
|  false                     |
+----------------------------+
1 row in set (0.00 sec)
```

Example : IF() function with CASE

निम्नलिखित उदाहरण में MySQL स्टेटमेंट तीसरी अभिव्यक्ति को ' false ' लौटाता है क्योंकि पहला एक्सप्रेशन सही नहीं है।

Code:

```
SELECT IF((SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false'
END),'true','false');
```

Sample Output:

mysql> SELECT IF((SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END),'true','false');

```
+-------------------------------------------------------------------------------------------+
| IF((SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END),'true','false')|
+-------------------------------------------------------------------------------------------+
| false                                                                                     |
+-------------------------------------------------------------------------------------------+
```

1 row in set, 1 warning (0.02 sec)

3. IFNULL () फ़ंक्शन

MySQL IFNULL () दो एक्सप्रेशन लेता है और यदि पहला एक्सप्रेशन NULL नहीं है, तो यह पहला एक्सप्रेशन देता है। अन्यथा, यह दूसरी अभिव्यक्ति लौटाता है।

जिस संदर्भ में इसका उपयोग किया जाता है, उसके □धार पर यह या तो संख्यात्मक या स्ट्रिंग मान देता है।

Syntax:

IFNULL(expression1, expression2);

Arguments

Name	Description
expression1	An expression.
expression2	An expression.

Example: MySQL IFNULL() function

निम्न MySQL स्टेटमेंट पहला एक्सप्रेशन देता है, यानी 0, क्योंकि पहला एक्सप्रेशन NULL नहीं है।

Code:

SELECT IFNULL(0,2);

Sample Output:

mysql> SELECT IFNULL(0,2);

```
+----------------------------+
| IFNULL(0,2)                |
+----------------------------+
|          0                 |
+----------------------------+
1 row in set (0.03 sec)
```

Example: IFNULL() function with non zero 1st argument

निम्न MySQL स्टेटमेंट पहला एक्सप्रेशन देता है, यानी 1, क्योंकि पहला एक्सप्रेशन NULL नहीं है।

Code:

```
SELECT IFNULL(1,2);
```

Sample Output:

mysql> SELECT IFNULL(1,2);

```
+----------------------------+
| IFNULL(1,2)                |
+----------------------------+
|          1                 |
+----------------------------+
1 row in set (0.00 sec)
```

Example: IFNULL() function NULL

निम्न MySQL स्टेटमेंट दूसरी एक्सप्रेशन, अर्थात 2, को लौटाता है, क्योंकि पहला एक्सप्रेशन NULL है।

Code:

```
SELECT IFNULL(NULL,2);
```

Sample Output:

mysql> SELECT IFNULL(NULL,2);

```
+----------------------------+
| IFNULL(NULL,2)       |
```

```
+----------------------------+
|            2               |
+----------------------------+
1 row in set (0.00 sec)
```

4. NULLIF () फ़ंक्शन

MySQL NULLIF () NULL लौटाता है जब पहला दूसरे एक्सप्रेशन के बराबर होता है, अन्यथा, यह पहला एक्सप्रेशन देता है।

Syntax:

NULLIF(expression1, expression2);

Arguments

Name	Description
expression1	An expression.
expression2	An expression.

Example: MySQL NULLIF() function

निम्नलिखित के बाद से MySQL स्टेटमेंट में भाव समान हैं, यह NULL देता है।

Code:

SELECT NULLIF(2,2);

Sample Output:

mysql> SELECT NULLIF(2,2);

```
+--------------------------+
| NULLIF(2,2)              |
+--------------------------+
|       NULL               |
+--------------------------+
1 row in set (0.03 sec)
```

Example : MySQL NULLIF() function with unequal arguments

निम्नलिखित MySQL कथन में चूंकि अभिव्यक्तियाँ समान नहीं हैं, यह पहली अभिव्यक्ति, अर्थात 2 देता है।

Code:

```
SELECT NULLIF(2,3);
```

Sample Output:

```
 mysql> SELECT NULLIF(2,3);
+-------------------------+
| NULLIF(2,3)             |
+-------------------------+
|          2              |
+-------------------------+
1 row in set (0.00 sec)
```

3.10.3 MYSQL STRING FUNCTIONS:

MySQL पाठ मानों में परिवर्तन करने के लिए SQL कथन के दौरान कहे जाने वाले फ़ंक्शन का विस्तृत चयन प्रदान करता है। निम्न तालिका इस श्रेणी में सबसे अधिक बार उपयोग किए जाने वाले कार्यों को सूचीबद्ध करती है। इन कार्यों का उपयोग कैसे करें के कुछ उदाहरण तालिका के बाद शामिल किए गए हैं।

Name	Description
CONCAT	Concatenate two or more strings into a single string एक स्ट्रिंग में दो या दो से अधिक स्ट्रिंग्स को मिलाती है |
INSTR	Return the position of the first occurrence of a substring in a string स्ट्रिंग में एक सबस्ट्रिंग की पहली घटना की स्थिति लौटाती है |
LENGTH	Get the length of a string in bytes and in characters बाइट्स और वर्णों में एक स्ट्रिंग की लंबाई प्राप्त कराती है |
LEFT	Get a specified number of leftmost characters from a string एक स्ट्रिंग से सबसे बाईं ओर के वर्णों की एक निश्चित संख्या प्राप्त कराती है |
LOWER	Convert a string to lowercase एक स्ट्रिंग को लोअरकेस में बदलें
LTRIM	Remove all leading spaces from a string एक स्ट्रिंग से सभी प्रमुख रिक्त स्थान निकालें
REPLACE	Search and replace a substring in a string स्ट्रिंग कि एक सब स्ट्रिंग में एक विकल्प को खोजें और बदलें

RIGHT	Get a specified number of rightmost characters from a string एक स्ट्रिंग से सही वर्णों की एक निर्दिष्ट संख्या दाहिनी ओर से प्राप्त करें
RTRIM	Remove all trailing spaces from a string एक स्ट्रिंग से सभी अनुगामी रिक्त स्थान निकालें
SUBSTRING	Extract a substring starting from a position with a specific length. एक विशिष्ट लंबाई के साथ एक स्थिति से शुरू होने वाले एक सबस्ट्रिंग को निकालें।
SUBSTRING_INDEX	Return a substring from a string before a specified number of occurrences of a delimiter एक सीमांकक की घटनाओं की एक निर्दिष्ट संख्या से पहले एक स्ट्रिंग से एक प्रतिस्थापन लौटें
TRIM	Remove unwanted characters from a string. एक स्ट्रिंग से अवांछित वर्ण निकालें।
FIND_IN_SET	Find a string within a comma-separated list of strings अल्पविराम से बनी सूची के भीतर एक स्ट्रिंग का पता लगाएं
FORMAT	Format a number with a specific locale, rounded to the number of decimals एक विशिष्ट लोकेल के साथ एक संख्या को प्रारूपित करें, दशमलव की संख्या तक गोल
UPPER	Convert a string to uppercase एक स्ट्रिंग को अपरकेस में बदलें

1. MySQL CONCAT फ़ंक्शन

MySQL CONCAT फ़ंक्शन एक या अधिक स्ट्रिंग तर्क (arguments) लेता है और उन्हें एक ही स्ट्रिंग में समेटता है। CONCAT फ़ंक्शन को न्यूनतम एक पैरामीटर की □वश्यकता होती है अन्यथा यह एक त्रुटि बताता है।

निम्नलिखित CONCAT फ़ंक्शन के सिंटैक्स को दिखाता है।

```
CONCAT(string1,string2, ... );
```

CONCAT फ़ंक्शन समसामयिकी से पहले सभी तर्कों को स्ट्रिंग प्रकार में परिवर्तित करता है। यदि कोई तर्क NULL है, तो CONCAT फ़ंक्शन NULL मान लौटाता है।

निम्न कथन दो उद्धृत स्ट्रिंग को जोड़ता है: MySQL और CONCAT

```
SELECT CONCAT('MySQL','CONCAT');
```

	CONCAT('MySQL','CONCAT')
▶	MySQLCONCAT

Another example:

यदि किसी तालिका में दो कॉलम 'first name' और 'last name' होते हैं। संपर्कों के पूर्ण नाम प्राप्त करने के लिए, □प CONCAT फ़ंक्शन का उपयोग पहले first name, space, last name को निम्नलिखित के रूप में लिखते हैं:

```
SELECT
   concat(contactFirstName,' ',contactLastName) Fullname
FROM    customers;
```

Output:

	Fullname
▶	Carine Schmitt
	Jean King
	Peter Ferguson
	Janine Labrune
	Jonas Bergulfsen
	Susan Nelson
	Zbyszek Piestrzeniewicz

2. INSTR () फ़ंक्शन

MySQL INSTR () तर्क के रूप में एक स्ट्रिंग और एक सबस्ट्रिंग लेता है, और एक पूर्णांक देता है जो स्ट्रिंग के भीतर सबस्ट्रिंग की पहली घटना की स्थिति को इंगित करता है।

Syntax:

INSTR (ori_str, sub_str)

Arguments

Name	Description
ori_str	The string to be searched. स्ट्रिंग जो सर्च करना है |
sub_str	The string to be searched for within the ori_str. स्ट्रिंग जो सर्च करना है ori_str के अंदर |

Example : MySQL INSTR() function

निम्न MySQL स्टेटमेंट 5 स्थिति में 'myteststring' में 'st' की पहली घटना पाता है, यह 5 रिटर्न करता है।

Code:

```
SELECT INSTR('myteststring','st');
```

Sample Output:

```
mysql> SELECT INSTR('myteststring','st');

+----------------------------------+
| INSTR('myteststring','st')  |
+----------------------------------+
|                        5            |
+----------------------------------+
1 row in set (0.03 sec)
```

3. LOWER () फ़ंक्शन

MySQL LOWER () सभी वर्णों को एक स्ट्रिंग में निचले वर्णों में परिवर्तित करता है।

Syntax:

```
LOWER(str)
```

Example: MySQL LOWER() function

निम्न MySQL स्टेटमेंट अपने सभी वर्णों को लोअरकेस में परिवर्तित करने के बाद दिए गए स्ट्रिंग को लौटाता है।

Code:

```
SELECT LOWER('MYTESTSTRING');
```

Sample Output:

```
mysql> SELECT LOWER('MYTESTSTRING');

+-------------------------------------------+
| LOWER('MYTESTSTRING')      |
+-------------------------------------------+
| myteststring                             |
```

```
+-----------------------------------------+
1 row in set (0.01 sec)
```

4. LTRIM () फ़ंक्शन

MySQL LTRIM () एक तर्क के रूप में पारित स्ट्रिंग के प्रमुख अंतरिक्ष वर्णों को हटा देता है।

Syntax:

LTRIM(str)

Example of MySQL LTRIM() function

निम्नलिखित MySQL स्टेटमेंट प्रमुख स्पेस वर्णों को हटाने के बाद स्ट्रिंग 'हेलो' लौटाता है। □उटपुट में, पहला कॉलम प्रमुख अंतरिक्ष वर्णों के साथ स्ट्रिंग दिखाता है और दूसरा कॉलम LTRIM करने के बाद स्ट्रिंग दिखाता है।

Code:

```
SELECT ' Hellow', LTRIM(' Hellow');
```

Sample Output:

```
mysql> SELECT ' Hellow', LTRIM(' Hellow');

+-----------------+-------------------------+
| Hellow        | LTRIM(' Hellow')    |
+-----------------+-------------------------+
|  Hellow       | Hellow                  |
+-----------------+-------------------------+
1 row in set (0.02 sec)
```

5. SUBSTRING () फ़ंक्शन

MySQL SUBSTRING () दिए गए स्ट्रिंग की एक विशेष स्थिति से वर्णों की एक निर्दिष्ट संख्या देता है।

Syntax:

SUBSTRING(str, pos, len)

OR

SUBSTRING(str FROM pos FOR len)

Arguments

Name	Description
Str	A string.
Pos	Starting position.
Len	Length in characters.

Example : MySQL SUBSTRING() function

निम्नलिखित MySQL स्टेटमेंट स्ट्रिंग 'w3resource' के 4 वें स्थान से वर्णों के 3 नंबर देता है।

Code:

```
SELECT SUBSTRING('w3resource',4,3);
```

Sample Output:

```
mysql> SELECT SUBSTRING('w3resource',4,3);

+-----------------------------------------+
| SUBSTRING('w3resource',4,3)   |
+-----------------------------------------+
| eso                                     |
+-----------------------------------------+
1 row in set (0.00 sec)
```

6. TRIM () फ़ंक्शन

MySQL TRIM () फ़ंक्शन दिए गए स्ट्रिंग से सभी उपसर्गों या प्रत्ययों को हटाने के बाद एक स्ट्रिंग लौटाता है।

Syntax:

```
TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM ] str)
```

Arguments

Name	**Description**

BOTH	दर्शाता है कि बाएं और दाएं दोनों से उपसर्गों(prefixes) को हटाया जाना है।
LEADING	दर्शाता है कि केवल प्रमुख उपसर्गों (leading prefixes)को हटाया जाना है
TRAILING	दर्शाता है कि केवल अनुगामी उपसर्गों (trailing prefixes) को हटाया जाना है।
Remstr	स्ट्रिंग जिसे हटाना है |
FROM	कीवर्ड (Keyword)
Str	वास्तविक स्ट्रिंग जहाँ से remstr को हटाना है |

Example : MySQL TRIM() function

निम्नलिखित MySQL स्टेटमेंट 'स्ट्रिंग' ट्रिम से अग्रणी और अनुगामी रिक्त स्थान को हटाने के बाद स्ट्रिंग लौटाता है।

```
SELECT TRIM(' trim ');
```

Sample Output:

```
mysql> SELECT TRIM(' trim ');

+-------------------------+
| TRIM(' trim ')          |
+-------------------------+
| trim                    |
+-------------------------+
1 row in set (0.00 sec)
```

3.10.4 NUMERICAL FUNCTIONS

MySQL सांख्यिक कार्यों का उपयोग मुख्यतः संख्यात्मक हेरफेर और / या गणितीय गणनाओं के लिए किया जाता है। निम्न तालिका MySQL कार्यान्वयन में उपलब्ध संख्यात्मक कार्यों का विवरण देती है।

Sr.No.	Name	Description
1	ABS()	Returns the absolute value of numeric expression.
2	ACOS()	Returns the arccosine of numeric expression. Returns NULL if the value is not in the range -1 to 1.
3	ASIN()	Returns the arcsine of numeric expression. Returns NULL if value is not in the range -1 to 1
4	ATAN()	Returns the arctangent of numeric expression.

5	ATAN2()	Returns the arctangent of the two variables passed to it.
6	BIT_AND()	Returns the bitwise AND all the bits in expression.
7	BIT_COUNT()	Returns the string representation of the binary value passed to it.
8	BIT_OR()	Returns the bitwise OR of all the bits in the passed expression.
9	CEIL()	Returns the smallest integer value that is not less than passed numeric expression
10	CEILING()	Returns the smallest integer value that is not less than passed numeric expression
11	CONV()	Converts numeric expression from one base to another.
12	COS()	Returns the cosine of passed numeric expression. The numeric expression should be expressed in radians.
13	COT()	Returns the cotangent of passed numeric expression.
14	DEGREES()	Returns numeric expression converted from radians to degrees.
15	EXP()	Returns the base of the natural logarithm (e) raised to the power of passed numeric expression.
16	FLOOR()	Returns the largest integer value that is not greater than passed numeric expression.
17	FORMAT()	Returns a numeric expression rounded to a number of decimal places.
18	GREATEST()	Returns the largest value of the input expressions.
19	INTERVAL()	Takes multiple expressions exp1, exp2 and exp3 so on.. and returns 0 if exp1 is less than exp2, returns 1 if exp1 is less than exp3 and so on.
20	LEAST()	Returns the minimum-valued input when given two or more.
21	LOG()	Returns the natural logarithm of the passed numeric expression.
22	LOG10()	Returns the base-10 logarithm of the passed numeric expression.
23	MOD()	Returns the remainder of one expression by diving by another expression.
24	OCT()	Returns the string representation of the octal value of the passed numeric expression. Returns NULL if passed value is NULL.
25	PI()	Returns the value of pi
26	POW()	Returns the value of one expression raised to the power of another expression
27	POWER()	Returns the value of one expression raised to the power of another expression

28	RADIANS()	Returns the value of passed expression converted from degrees to radians.
29	ROUND()	Returns numeric expression rounded to an integer. Can be used to round an expression to a number of decimal points
30	SIN()	Returns the sine of numeric expression given in radians.
31	SQRT()	Returns the non-negative square root of numeric expression.
32	STD()	Returns the standard deviation of the numeric expression.
33	STDDEV()	Returns the standard deviation of the numeric expression.
34	TAN()	Returns the tangent of numeric expression expressed in radians.
35	TRUNCATE()	Returns numeric exp1 truncated to exp2 decimal places. If exp2 is 0, then the result will have no decimal point.

उपरोक्त कार्यों में से कुछ उदाहरणों का उपयोग करते हुए नीचे वर्णित हैं:

1. ABS(X)

ABS () फ़ंक्शन X का पूर्ण मान लौटाता है। निम्न उदाहरण पर विचार करें –

```
mysql> SELECT ABS(2);
+----------------------------------------------------------+
|                      ABS(2)                              |
+----------------------------------------------------------+
|                        2                                 |
+----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ABS(-2);
+----------------------------------------------------------+
|                      ABS(2)                              |
+----------------------------------------------------------+
|                        2                                 |
+----------------------------------------------------------+
1 row in set (0.00 sec)
```

2. ACOS(X)

यह फ़ंक्शन X का arccosine लौटाता है। X का मान .1 से 1 के बीच होना चाहिए नहीं तो NULL वापस हो जाएगा। निम्नलिखित उदाहरण पर विचार करें –

```
mysql> SELECT ACOS(1);
+-----------------------------------------------------+
|                    ACOS(1)                          |
+-----------------------------------------------------+
|                  0.000000                           |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

ASIN(X)

ASIN () फ़ंक्शन X का arcsine लौटाता है। X का मान .1 से 1 के रेंज में होना चाहिए या फिर NULL लौटाया जाना चाहिए।

```
mysql> SELECT ASIN(1);
+-----------------------------------------------------+
|                    ASIN(1)                          |
+-----------------------------------------------------+
|                1.5707963267949                      |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

ATAN(X)

यह फ़ंक्शन X का arctangent लौटाता है।

```
mysql> SELECT ATAN(1);
+-----------------------------------------------------+
|                    ATAN(1)                          |
+-----------------------------------------------------+
|               0.78539816339745                      |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

3. CEIL(X)

CEILING(X)

यह फ़ंक्शन सबसे छोटा पूर्णांक मान लौटाता है जो X से छोटा नहीं है। निम्न उदाहरण पर विचार करें–

```
mysql> SELECT CEILING(3.46);
+-----------------------------------------------------+
|                CEILING(3.46)                        |
+-----------------------------------------------------+
|                  4                                  |
+-----------------------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT CEIL(-6.43);

+-----------------------------------------------------+
|                CEIL(-6.43)                          |
+-----------------------------------------------------+
|                  -6                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

4. EXP(X)

यह फ़ंक्शन एक्स की घात के लिए उठाए गए ई (प्राकृतिक लॉगरिथम का □धार) का मान लौटाता है।

```
mysql>SELECT EXP(3);
+-----------------------------------------------------+
|                  EXP(3)                             |
+-----------------------------------------------------+
|                20.085537                            |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

5. FLOOR(X)

यह फ़ंक्शन सबसे बड़ा पूर्णांक मान लौटाता है जो X से अधिक नहीं है।

```
mysql>SELECT FLOOR(7.55);
+-----------------------------------------------------+
|                FLOOR(7.55)                          |
+-----------------------------------------------------+
|                  7                                  |
+-----------------------------------------------------+
```

1 row in set (0.00 sec)

6. GREATEST(n1,n2,n3,.....)

GREATEST () फ़ंक्शन इनपुट पैरामीटर (n1, n2, n3, और इसी तरह) के सेट में सबसे बड़ा मान लौटाता है। निम्नलिखित उदाहरण GREATEST () फ़ंक्शन का उपयोग संख्यात्मक मानों के सेट से सबसे बड़ी संख्या को वापस करने के लिए करता है -

```
mysql>SELECT GREATEST(3,5,1,8,33,99,34,55,67,43);
+-----------------------------------------------------------+
|         GREATEST(3,5,1,8,33,99,34,55,67,43)               |
+-----------------------------------------------------------+
|                        99                                 |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

7. LEAST(N1,N2,N3,N4,.....)

LEAST () फ़ंक्शन GREATEST () फ़ंक्शन के विपरीत है। इसका उद्देश्य मूल्य सूची (n1, n2, n3, और इसी तरह) से कम से कम मूल्यवान वस्तु को वापस करना है। निम्न उदाहरण LEAST () फ़ंक्शन के लिए उचित उपयोग और आउटपुट दिखाता है -

```
mysql>SELECT LEAST(3,5,1,8,33,99,34,55,67,43);

+-----------------------------------------------------+
|         LEAST(3,5,1,8,33,99,34,55,67,43)            |
+-----------------------------------------------------+
|                        1                            |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

8. MOD(N,M)

यह फ़ंक्शन M द्वारा विभाजित N के शेष भाग को लौटाता है। निम्न उदाहरण पर विचार करें -

```
mysql>SELECT MOD(29,3);
+-----------------------------------------------------+
|                   MOD(29,3)                         |
+-----------------------------------------------------+
|                        2                            |
```

```
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

9. PI()

यह फ़ंक्शन केवल पाई का मान लौटाता है। MySQL आंतरिक रूप से pi के पूर्ण दोहरे परिशुद्धता (double-precision) मान को संग्रहीत करता है।

```
mysql>SELECT PI();
+-----------------------------------------------------+
|                    PI()                             |
+-----------------------------------------------------+
|                  3.141593                           |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

10. POW(X,Y)

POWER(X,Y)

यह फ़ंक्शन Y की घात के लिए उठाए गए X का मान लौटाता हैं।

```
mysql> SELECT POWER(3,3);
+-----------------------------------------------------+
|                  POWER(3,3)                         |
+-----------------------------------------------------+
|                    27                               |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

11. ROUND(X)

ROUND(X,D)

यह फ़ंक्शन एक्स को निकटतम पूर्णांक तक गोल करता है। यदि एक दूसरा तर्क, D, की आपूर्ति की जाती है, तो फ़ंक्शन X को D दशमलव स्थान तक राउंड करेगा। D धनात्मक (positive value) होना चाहिए या दशमलव बिंदु के दाईं ओर के सभी अंक हटा दिए जाएंगे। निम्नलिखित उदाहरण पर विचार करें -

```
mysql>SELECT ROUND(5.693893);
```

```
+-----------------------------------------------------+
|                ROUND(5.693893)                      |
+-----------------------------------------------------+
|                     6                               |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql>SELECT ROUND(5.693893,2);
+-----------------------------------------------------+
|                ROUND(5.693893,2)                    |
+-----------------------------------------------------+
|                     5.69                            |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

12. SQRT(X)

यह फ़ंक्शन X के गैर-नकारात्मक (non negative) वर्गमूल को लौटाता है। निम्न उदाहरण पर विचार करें -

```
mysql>SELECT SQRT(49);
+-----------------------------------------------------+
|                   SQRT(49)                          |
+-----------------------------------------------------+
|                     7                               |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```