Introduction to MySQL

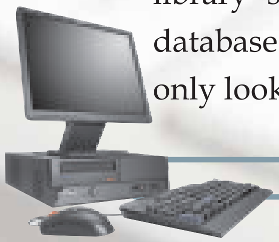
Learning Objectives

After studying this lesson the students will be able to

- State what a database is.
- Express the relationship between a database and table
- Recognize different parts of a table like Row and Column.
- Define DBMS related terms like Primary key, Candidate key, Alternate key etc.
- List the functions of a DBMS.
- Write examples of popular DBMS software.
- State what is MySQL.
- Install MySQL in a computer.

Most of us keep diaries to store details like names, addresses, birthdays of our friends. Teachers keep marks registers to keep track of marks secured by their students. A shopkeeper keeps details of customers who frequently visit his /her shop in a register. These all are examples of paper-based databases. A database is an organized collection of related data. However, generally, when we use the term 'database' we think of a computerized database. In this lesson, let us study more about such databases and numerous tasks that we can do on them.

These days computerized databases can be seen being used almost everywhere. The police force uses various computerized databases to help them track criminals and solve crimes. A library stores details of all their books, in a computerized database. When we want to know if a book is in stock, we cannot only look it up, but can also check when it is due to be returned.



The database also records details of all the borrowers, what books they currently have borrowed and when they are due back.

What is Database Management System(DBMS)?

To create and maintain a database on a computer, we need a database program, called a Database management system, or DBMS. Database Management System is a software that enables users to create and maintain databases. Examples of popular DBMSs are MySQL, PostgreSQL, Microsoft Access, Oracle, Microsoft SQL Server, DB2 and Sybase.

A DBMS gives us tools to:

- store data in a structured way.
- query the database (that is, ask questions about the data)
- sort and manipulate the data in the database
- validate the data entered and check for inconsistencies
- produce flexible reports, both on screen and on paper, that make it easy to comprehend the information stored in the database.

Tables in a Database

Relational Databases store data or information in tables. A table is similar to a spreadsheet where data is stored in rows and columns. A table refers to a two dimensional representation of data using rows and columns. For example, consider the following table named Customer with details about customers:

Table: Customer

Column

Row

Customer_ID	FirstName	LastName	Address	Telephone No
101	Prachi	Mehra	145, Mahatma Avenue, Delhi	9178908767
102	Vinay	Ahlurkar	76-A/32, Adarsh Nagar, Delhi	9278906351
103	Venu	Magalam	C-6, Kanthi Nagar, Delhi	9323764561
104	Neeza	Ali	B-6-B,Fateh Nagar, Meerut	9143347330

The horizontal subset of the Table is known as a Row/Tuple. Each row represents a record, which is a collection of data about a particular person, place or thing. The vertical subset of the Table is known as a Column/Attribute. The term field is also often used for column. Each column has a unique name and the content within it must be of the same type.

Relational Database

In the database named Learner shown below, the data is organized into separate tables. Once the tables have been set up, a relationship can be created to link them together. Such a database that stores data in separate tables that are related through the use of a common column is called a Relational database.

Database : Learner

Student table

StudentName	StudentID
K.S. Lakshmi	84
Ankita Matta	100
Himali Shah	92
Arushi Goel	106

Participant table

StudentID	Activity
84	Swimming
84	Dancing
92	Tennis
100	Golf
100	Cricket
106	Squash

Activity table

Activity	Cost
Swimming	2000.00
Dancing	1500.00
Tennis	900.00
Golf	1500.00
Cricket	2000.00
Squash	2500.00

RDBMS Terminology:

Primary key

When you got admission in the school, you were given an Admission number. The Admission number assigned to you was not assigned to any other student of your school (it is unique). When patients go to a hospital, each patient is given a unique patient number. When you go to open an account in the bank, you are given a unique account number. Admission number, Patient number, Account number are all examples of Primary key. A primary key is a field in a table that is unique for each record. Every database table should have a column or a group of columns designated as the primary key. The value this key holds should be unique for each record in the table.

Some more examples of Primary key are: Accession Number of a Book in the Book table, Employee ID of an employee in the Employee Table, Item Code of an item in the Stock table, Flight Number of a flight in the Flight Master Table, etc.

The purpose of a primary key is to uniquely identify each record in a table.

Candidate key

In a table, there may be more than one field that uniquely identifies a record. All such fields are called candidate keys. A Candidate key is an attribute (or set of attributes) that uniquely identifies a row. A Primary Key is one of the candidate keys. A table may have more than one candidate keys but definitely has one and only one primary key.

Example: Consider the following Table, RollNo and Admission_no both may be used to uniquely identify each row in this Table, so both are candidate keys.

Admission_No	RollNo	Name	Class	Sec	Dues
2301	1	Simran Chadha	11	A	23
1501	2	Ajay Kartik	11	B	15
1678	3	Vanshay Chawla	11	A	20
7003	4	Vibhor Madan	11	C	15

Alternate Key:

Only one of the Candidate keys is selected as the primary key of a table. All other

candidate keys are called Alternate keys. In the above example, if we use one of the candidate keys, say, Admission_No as the Primary Key, the other Candidate Key RollNo is the Alternate Key and vice-versa.

Introduction to MySQL:

The software required to manipulate relational databases is known as Relational Database Management System (RDBMS) . Popular RDBMSs include MySQL, Oracle, Sybase, DB2, MS SQL Server.

MySQL is a relational database management system (RDBMS). It is pronounced as "My Sequel". MySQL was originally founded and developed in Sweden by David Axmark, Allan Larsson and Michael Widenius, who had worked together since the 1980s.

Characteristics of MySQL:

- MySQL is released under an open-source license so it is customizable. It requires no cost or payment for its usage.
- MySQL has superior speed, is easy to use and is reliable.
- MySQL uses a standard form of the well-known ANSI-SQL standards.
- MySQL is a platform independent application which works on many operating systems like Windows, UNIX, LINUX etc. and has compatibility with many languages including JAVA , C++, PHP, PERL, etc.
- MySQL is an easy to install RDBMS and is capable of handling large data sets.

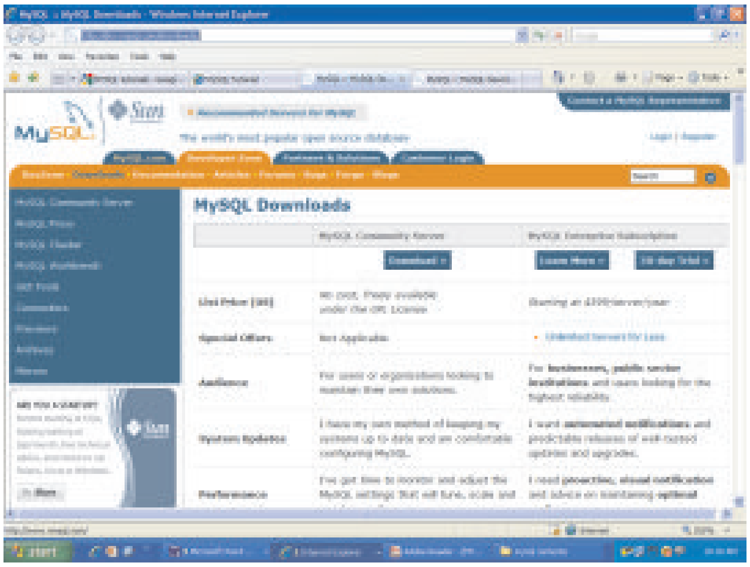
Since MySQL is released under an open-source license, it does not require any cost or payment for its usage. Any one can download this software from specific location on Internet. If you want to download, follow the following steps. The step for two most popular OS platform, Windows and Linux are discussed here.

DOWNLOADING MySQL [Windows Environment]:

Installation file for MySQL may be downloaded from the link:

http://dev.mysql.com/downloads/mysql/5.1.html#downloads

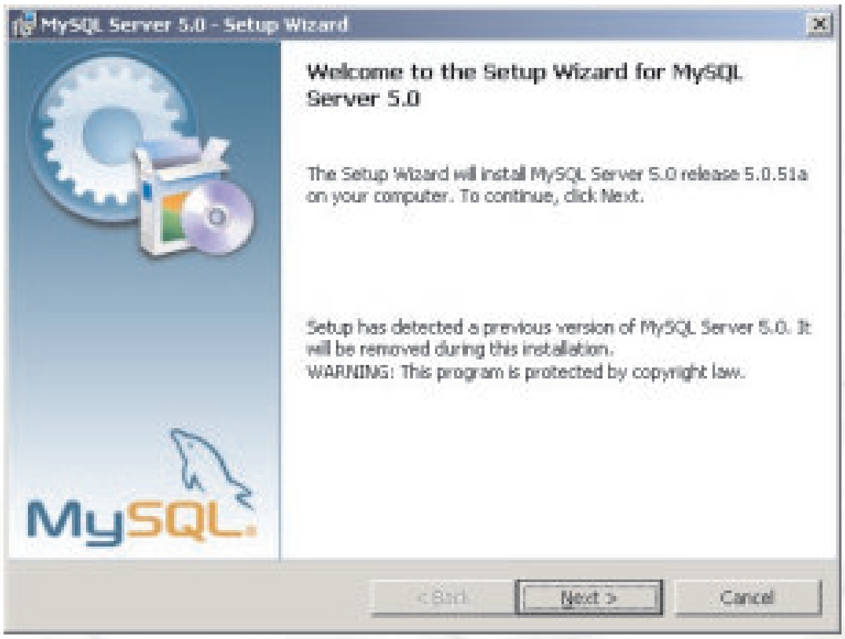
(Choose appropriate download link as per the operating system)



Click on the "Download" button for the Community Server and choose from the list of supported platforms (i.e., operating systems that it will run on), which include 32-bit and 64-bit Windows, several different Linux, Solaris, Mac OS X, and a few others.

INSTALLING MySQL:

After the installation file has finished downloading, double-click it, which begins the MySQL Setup Wizard.

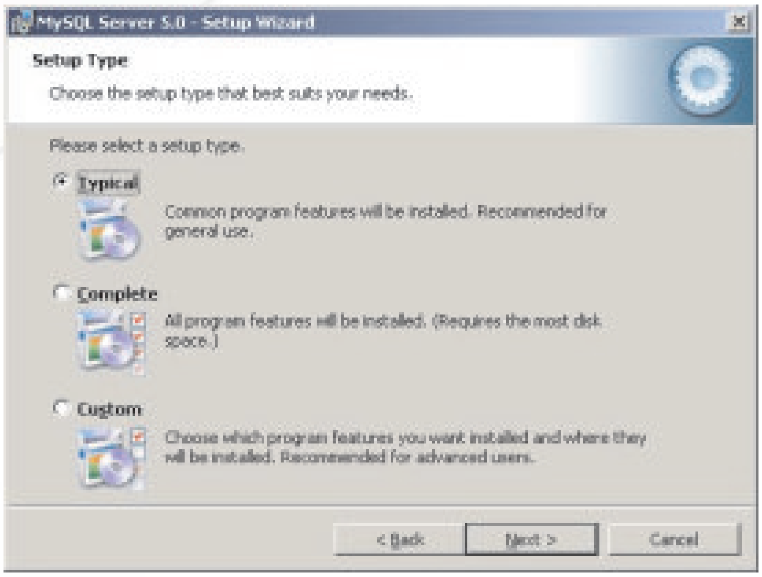


Welcome to the Setup Wizard for MySQL Server 5.0

The Setup Wizard will install MySQL Server 5.0 release 5.0.51a on your computer. To continue, click Next.

Setup has detected a previous version of MySQL Server 5.0. It will be removed during this installation.
WARNING: This program is protected by copyright law.

At the welcome dialog box, click the "Next" button.


MySQL Server 5.0 - Setup Wizard

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

[✓ Typical]
Common program features will be installed. Recommended for general use.

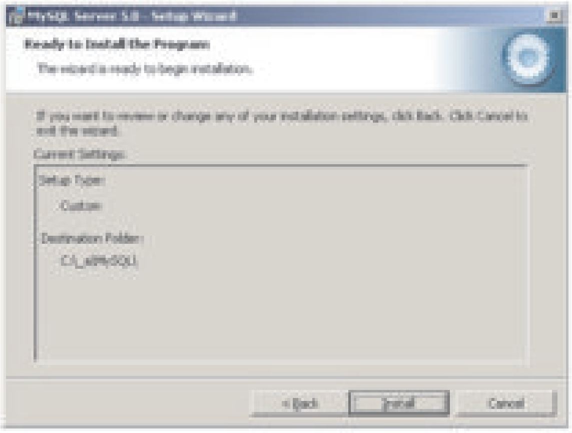
Complete
All program features will be installed. (Requires the most disk space.)

Custom
Choose which program features you want installed and where they will be installed. Recommended for advanced users.

The MySQL Setup Wizard allows us to choose the installation directory on the computer, and whether or not to have optional components installed. In the "Setup Type" dialog box, choose "Typical" from the three options. MySQL will be installed in the default directory, "C:\Program Files\MySQL\MySQL Server. Click the "Next" button.

Now it is ready to install MySQL's files. Click the "Install" button.

After the Setup is complete, we should configure the new server.


MySQL Server 5.0 - Setup Wizard

Ready to Install the Program

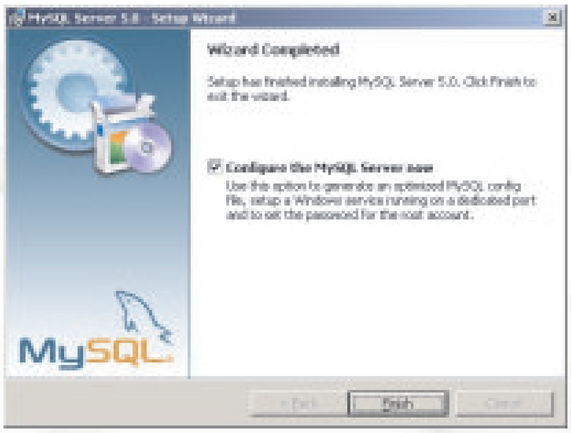
The wizard is ready to begin installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Current Settings:

Setup Type:
Custom

Destination Folder:
C:_a\MySQL\

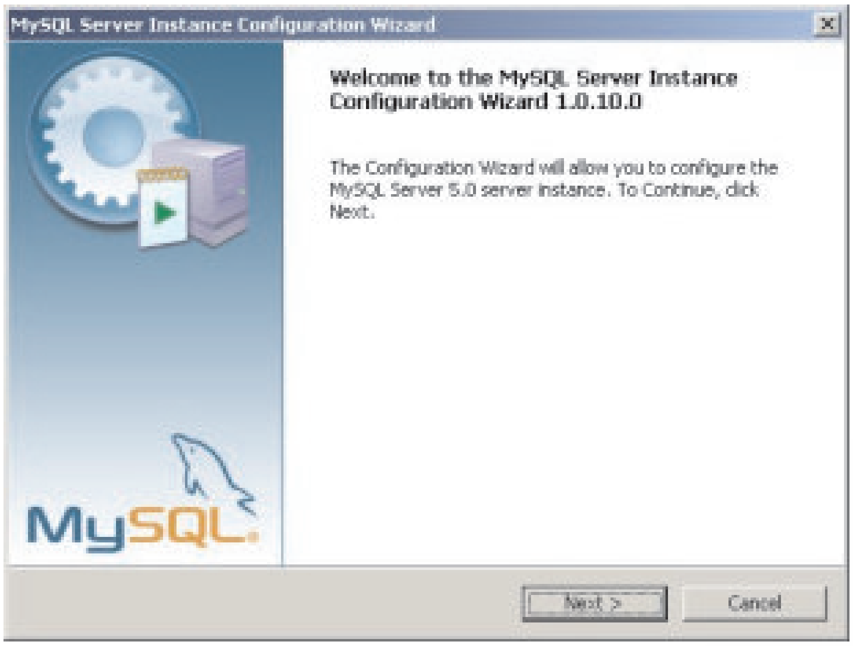

MySQL Server 5.0 - Setup Wizard

Wizard Completed

Setup has finished installing MySQL Server 5.0. Click Finish to exit the wizard.

[✓ Configure the MySQL Server now]
Use this option to generate an optimized MySQL config file, setup a Windows service running on a dedicated port and to set the password for the root account.

CONFIGURING MySQL:

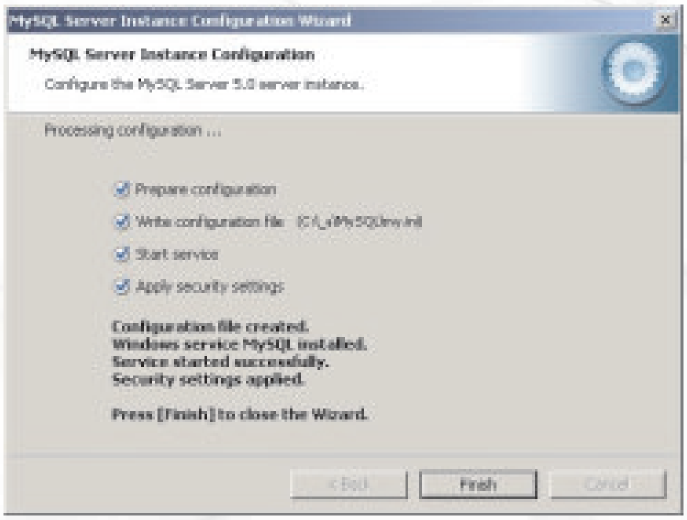

MySQL Server Instance Configuration Wizard

Welcome to the MySQL Server Instance Configuration Wizard 1.0.10.0

The Configuration Wizard will allow you to configure the MySQL Server 5.0 server instance. To Continue, click Next.

At the initial Server Instance Configuration Wizard dialog box, click the "Next" button. Keep selecting the default options provided in subsequent windows. If the configuration does not encounter any errors, then information will be prompted that the configuration file was created, MySQL server was installed and started, and the security settings applied.

Note: In the process of configuration of MySQL, a prompt for password will be displayed - Here you should enter a password and remember this password, as it will be required each time to start MySQL


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Processing configuration ...

- ✓ Prepare configuration
- ✓ Write configuration file (C:_x\MySQL\my.ini)
- ✓ Start service
- ✓ Apply security settings

Configuration file created.
Windows service MySQL installed.
Service started successfully.
Security settings applied.

Press [Finish] to close the Wizard.

Testing MySQL:

Follow the steps to start MySQL

```
Start> Programs>MySQL>….>MySQL Command Line Client
```

OR

Goto the folder

```
C:\Program Files\MySQL\MySQL Server 5.1\bin   [Assuming C:\ drive
as the drive having MySQL]
```

And Click on the file

```
MySQL.EXE
```

MySQL will prompt a message to provide password (it requires the same password which was entered during the installation)

```
Enter Password:****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version:  5.0.51a-community-nt MySQL Community Edition
(GPL)
Type 'help;' or '\h' for help.  Type '\c' to clear the buffer.
Mysql>
```

To exit from MySQL, type QUIT or EXIT

```
Mysql>QUIT
```

The above steps ensure successful installation and configuration of MySQL database server. Next time in the MySQL prompt, one can create and use databases, create tables and execute SQL queries.

Downloading MySQL [Linux Environment]:

Installation of the binary version of MySQL, release 4.0.20, to run on Linux is as follows:

Installation file for MySQL may be downloaded from the link:

http://dev.mysql.com/downloads/mysql/5.1.html#downloads

(Choose appropriate download link as per the desired operating system)

Create MySQL User Account:

```
# cd /usr/local
# groupadd mysql
# useradd -c "MySQL Software Owner" -g mysql mysql
# passwd mysql
Changing password for user mysql.
password: all authentication tokens updated successfully.
```

Installing Binary Version:

Unzip the files and change the directory to mysql

```
# cd mysql
# scripts/mysql_install_db --user=mysql
```

```
Preparing db table
Preparing host table
Preparing user table
Preparing func table
  . . .
  . . .
  . . .
The latest information about MySQL is available on the web at
http://www.mysql.com
Support MySQL by buying support/licenses at
https://order.mysql.com
```

Start and Stop The Database Software:

```
Starting the MySQL Database
# su -
# cd /usr/local/mysql
# bin/mysqld_safe --user=mysql &
```

Starting mysqld daemon with databases from /usr/local/mysql/data

Stopping the MySQL Database

```
# su -
# cd /usr/local/mysql
# bin/mysqladmin -u root shutdown
040803 23:36:27  mysqld ended
[1]+  Done      bin/mysqld_safe --user=mysql
```

Know more

Visit the following website to find a vast list of free and open source softwares available:
http://en.wikipedia.org/wiki/List_of_free_and_open_source_software_packages

Summary

- A database is an organised collection of data.
- Data is stored in a relational database in one or more tables.
- A group of rows and columns forms a Table.
- The horizontal subset of a Table is known as a Row/Tuple.
- The vertical subset of a Table is known as a Column/Attribute.
- A Candidate key is an attribute (or a set of attributes) that uniquely identifies a row. A Primary Key is one of the candidate keys.
- Only one of the Candidate keys is selected as the primary key of a table. All other candidate keys are called Alternate keys.

Multiple Choice Questions

1. **A relation can have only one ________ key and may have more than one _______ keys.**

   a) Primary, Candidate

   b) Candidate, Alternate

   c) Candidate, Primary

   d) Alternate, Candidate

2. **The vertical subset of a table is known as:**

   a) Tuple

   b) Row

   c) Attribute

   d) Relation

3. **If software is released under open source, it means:**

   a) It is expensive.

   b) Its source code is available to the user.

   c) It belongs to a company.

   d) It is a DBMS.

4. **Which of the following columns in a Student table can be used as the primary key?**

   a) Class

   b) Section

   c) First Name

   d) Admission No

5. **A tuple is also known as a ___________________________.**

   a) table

   b) relation

   c) row

   d) field

6. **An attribute is also known as a__________________________.**

   a) table

   b) relation

   c) row

   d) column

7. **A field or a combination of fields in a table that has a unique value for each row is called:**

   a) Candidate key.

   b) Foreign key.

   c) Main key.

   d) Alternate key.

Exercises

1. **Answer the following questions:**

   a) Define the following terms:

      i) Database

      ii) Table

      iii) Primary key

      iv) Candidate key

      v) Alternate key

   b) What is the relationship between a Database and a Table?

   c) What is DBMS? Write names of any two DBMSs.

   d) How is data organized in a table?

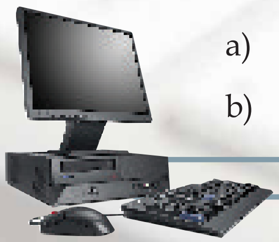
   e) What is a Primary key? What is its purpose in a table?

   f) What is MySQL?

2. **Distinguish between the following pairs**

   a) Row and Column

   b) Primary key and Candidate key.



9 MySQL

Learning Objectives

After studying this lesson the students will be able to:

- ❖ State categories of SQL statements.
- ❖ Create a database
- ❖ Create a table.
- ❖ Add rows to a table.
- ❖ Retrieve data in various ways from table using SELECT statement.
- ❖ Display data in a sorted way using ORDER BY clause.
- ❖ Modify data stored in a table.
- ❖ View structure of a table
- ❖ Modify structure of table
- ❖ Delete rows from a table

In the previous lesson, you have learnt that Relational Databases use tables to store data. A table simply refers to a two dimensional representation of data using columns and rows. MySQL lets us manipulate and manage these tables in an efficient way. We have learnt that MySQL is a Relational Database Management System. In this lesson we will learn about SQL (Structured Query Language).It is a Standard language used for accessing and manipulating relational databases.

Ms. Sujata is a Class teacher of Class XI. She wants to store data of her students i.e. Names and marks secured, in a database. A database is used to house data in the form of tables. She uses a CREATE DATABASE statement to create a new database named School.

```
mysql> CREATE DATABASE School;
```

Once the above mentioned statement gets executed, a database with the name School is created on her system. Now she has to open the database to work on it. For this USE statement is required. She opens the School database:

Statement entered by user

```
mysql> USE School;
Database Changed
```

Display by system

Now, MySQL prompt can accept any query related to the database School.

! *Semicolon is standard way to end SQL statement.*

Creating a table

After creating a database, the next step is creation of tables in the database. For this CREATE TABLE statement is used.

Syntax:

```
CREATE TABLE <TableName>(<ColumnName1> <Data Type1>,
<ColumnName2> <Data Type2>,… ,<ColumnNameN> <Data TypeN>);
```

Since Ms. Sujata is just learning, she initially creates a simple table named Learner with only two columns RollNo and Name in the School database.

To do this, she enters the following statement:

```
mysql> CREATE TABLE Learner
          (
                RollNo     INTEGER,
                Name  VARCHAR(25)
          );
```

!

- Give meaningful name to a table. If a table will store information about students, name it STUDENTS, not Abc or Person.
- Table names and column names are not case sensitive. For example, STUDENTS is treated the same as STuDents or students.

We will study about the CREATE TABLE statement in detail later in this lesson.

What if Ms. Sujata wants to see the names of all the tables in the database? At any point of time, she can view names of all the tables contained in the current database by using SHOW TABLES statement as shown below:

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_school  |
+------------------+
| Learner           |
|------------------+
1 row in set (0.00 sec)
```

Once the table named Learner is created, Ms. Sujata would like to add data of students in the table, which is also known as populating table with rows. To add row(s) in the table she uses the INSERT INTO statement:

Syntax:

```
INSERT INTO <TableName>
VALUES (<Value1>,<Value2>,... ,<ValueN>);
```

She inserts 4 rows :

```
mysql> INSERT INTO Learner VALUES (14,'Aruna Asaf Ali');
mysql> INSERT INTO Learner VALUES (12,'Tarun Sinha');
mysql> INSERT INTO Learner VALUES (16,'John Fedrick');
mysql> INSERT INTO Learner VALUES (10,'Yogi Raj Desai');
```

! In INSERT statement:

Character, date and Time data should be enclosed in Quotes.

Numeric values should not be enclosed in quotes.

Now that she has added 4 rows in the table, she wants to view the contents of the table. How can she do that? To view the contents of the table, she uses the following SELECT statement. In the simplest way, SELECT statement is used like this:

Syntax:

SELECT * FROM <TableName>;

So, she types the statement:

```
mysql> SELECT * FROM Learner;
+---------------------------+
|RollNo  | Name             |
+---------------------------+
|  14    | Aruna Asaf Ali   |
|  12    | Tarun Sinha      |
|  16    | John Fedrick     |
|  10    | Yogi Raj Desai   |
+---------------------------+
```

In the above statement, FROM clause states which table to look in for data.

Any time to know the database currently in use, the SELECT DATABASE() statement can be used.

```
mysql> SELECT DATABASE();
DATABASE()
school
1 row in set (0.0 sec)
```

! *Statements in MySQL are not case sensitive. It means select DATABASE(); or SELECT DATABASE(); or SELECT database(); would all work the same way.*

Some Terminologies

Keyword: A keyword refers to a special word that has a special meaning to SQL. For example, SELECT and FROM are keywords.

Clause : A clause is a portion of an SQL statement. Each clause is identified by a keyword.

For example, consider the statement

```
SELECT name FROM Learner;
```

Here SELECT name is a clause. SELECT is a statement as well as a clause. SELECT clause is everything from keyword SELECT until keyword FROM. SELECT statement is the entire command.

FROM Learner is a FROM clause, which specifies the table from which data has to be selected.

Statement : A statement is a combination of two or more clauses. For example,

```
SELECT name FROM Learner;
```

is a statement.

MySQL Data Types

Well, before we learn more about making a table, there is one thing we need to understand first: Data Types. They indicate the type of data that you are storing in a given table column. So, what are the different Data Types available in MySQL? Here is a list of some of the most common ones and what type of values they hold:

Class	Data Type	Description	Example
Text	CHAR(size)	A fixed-length string from 1 to 255 characters in length right-padded with spaces to the specified length when stored. Values must be enclosed in single quotes or double quotes.	'Maths' "TexT"
	VARCHAR(size)	A variable-length string from 1 to 255 characters in length; for example VARCHAR(25).Values must be enclosed in single quotes or double quotes.	'Computer' "Me and u"
Numeric	DECIMAL(size,d)	It can represent number with or without the fractional part. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter	17.32 345
	INT Or INTEGER	It is used for storing integer values. You can specify a width upto 11 digits.	76

Date	DATE	It represents the date including day, month and year	'2009-07-02'
TIME()	TIME	It represents time. Format: HH:MM:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'	

Categories of SQL Commands

SQL commands can be classified into the following categories:

1. Data Definition Language (DDL) Commands

The DDL part of SQL permits database tables to be created or deleted. It also defines indices (keys), specifies links between tables, and imposes constraints on tables. Examples of DDL commands in SQL are:

- **CREATE DATABASE -** creates a new database
- **CREATE TABLE -** creates a new table
- **ALTER TABLE -** modifies a table
- **DROP TABLE -** deletes a table

2. The Data Manipulation Language (DML) Commands

The query and update commands form the DML part of SQL: Examples of DDL commands are:

- **SELECT -** extracts data from a table
- **UPDATE -** updates data in a table
- **DELETE -** deletes data from a table
- **INSERT INTO -** inserts new data into a table

CREATE TABLE

Ms. Sujata feels good that she has successfully created a table named Learner with 2 columns using CREATE TABLE statement. She now creates a table named Student with

four columns. When tables are created its columns are named, data types and sizes are supplied for each column. While creating a table at least one column must be specified.

Syntax:

```
CREATE TABLE <table-name> (< column name><data type> [ <size>],
(< column name><data type> [ <size>], …);
```

Example:

```
mysql> USE school;
Database changed
```

```
mysql> CREATE TABLE Student(
Rollno INTEGER,
Name VARCHAR(25),
Gender  CHAR(1),
Marks1 DECIMAL(4,1));
Query OK, 0 rows affected (0.16 sec)
```

! *If table Student already exists in database school, then the error message "Table Student already exists" is displayed.*

Each column in the table is given a unique name. In the example above the column names are Rollno, Name etc. This doesn't mean each column that is named has to be unique within the entire database. It only has to be unique within the table where it exists. Also notice that the names do not use any spaces.

!*When naming tables and columns be sure to keep it simple with letters and numbers. Spaces and symbols are invalid characters except for underscore(_). Column names like first_name,last_name,email are valid column names.*

Viewing Structure of Table

The DESCRIBE statement can be used to see the structure of a table as indicated in the Create Statement. It displays the Column names, their data types, whether Column must contain data ,whether the Column is a Primary key etc.

Syntax:

```
DESCRIBE <table name>;
  OR
DESC <table name>;
```

Statement entered by user

```
mysql> DESCRIBE Student;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| Rollno | int(11)      | YES  |     | NULL    |       |
| Name   | varchar(25)  | YES  |     | NULL    |       |
| Gender | char(1)      | YES  |     | NULL    |       |
| Marks1 | decimal(4,1) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Output shown by system

Ms. Sujata adds some rows in the Student table using the INSERT INTO statement:

```
INSERT INTO Student VALUES (1,'Siddharth Sehgal','M',93);
INSERT INTO Student VALUES (2,'Gurpreet Kaur','F',91);
INSERT INTO Student VALUES (3,'Monica Rana','F',93);
INSERT INTO Student VALUES (4,'Jatinder Sen','M',78);
INSERT INTO Student VALUES (5,'George Jacob','M',76);
INSERT INTO Student VALUES (6,'Deepa Bhandari','F',77);
INSERT INTO Student VALUES (7,'Akshay Nath','M',65);
```

Changing Structure of table

When we create a table we define its structure. We can also change its structure i.e. add, remove or change its column(s) using the ALTER TABLE statement.

Syntax:

```
ALTER TABLE <table_name>  ADD/DROP <column_name> [datatype];
ALTER TABLE <table> MODIFY <column> <new_definition>;
```

Example:

Ms. Sujata adds a column named Games.

mysql> ALTER TABLE Student ADD Games VARCHAR(20);

Now she wants to check the structure of the table to see that the new column Games is added.

```
mysql> DESCRIBE Student;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| Rollno | int(11)      | YES  |     | NULL    |       |
| Name   | varchar(25)  | YES  |     | NULL    |       |
| Gender | char(1)      | YES  |     | NULL    |       |
| Marks1 | decimal(4,1) | YES  |     | NULL    |       |
| Games  | varchar(20)  | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

After execution of the above ALTER TABLE statement, the Games column is added and a NULL value is assigned to all the rows in this column.

```
mysql> SELECT * FROM Student;
+--------+-------------------+--------+--------+-------+
| Rollno | Name              | Gender | Marks1 | Games |
+--------+-------------------+--------+--------+-------+
|      1 | Siddharth Sehgal  | M      |   93.0 | NULL  |
|      2 | Gurpreet Kaur     | F      |   91.0 | NULL  |
|      3 | Monica Rana       | F      |   93.0 | NULL  |
|      4 | Jatinder Sen      | M      |   78.0 | NULL  |
|      5 | George Jacob      | M      |   76.0 | NULL  |
|      6 | Deepa Bhandari    | F      |   77.0 | NULL  |
|      7 | Akshay Nath       | M      |   65.0 | NULL  |
+--------+-------------------+--------+--------+-------+
```

Now, suppose we want to change the newly added Games column to hold integers(in place of character data) using ALTER TABLE statement:

```
mysql> ALTER TABLE Student MODIFY games INTEGER;
```

To delete a column of a table the ALTER TABLE statement is used with Drop clause.

Ms. Sujata deletes the Games column using the ALTER TABLE statement:

```
mysql> ALTER TABLE Student DROP Games;
```

```
mysql> DESC student;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| Rollno | int(11)      | YES  |     | NULL    |       |
| Name   | varchar(25)  | YES  |     | NULL    |       |
| Gender | char(1)      | YES  |     | NULL    |       |
| Marks1 | decimal(4,1) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

The above display shows that Games column is removed from the table.

! *The word "DESC" can also be used in place of "DESCRIBE"*

Retrieving Information with SELECT Statement

The SELECT statement is used to fetch data from one or more database tables.

Retrieving Single Column

Here is the syntax of SELECT statement to retrieve a single column from a table:

Syntax:

```
SELECT <column name> FROM <table name>;
```

Example:

Ms. Sujata wants to display Roll numbers of all her students. She uses the following statement:

```
mysql> SELECT Rollno FROM Student;
+----------+
|  Rollno  |
+----------+
|     1    |
|     2    |
|     3    |
|     4    |
|     5    |
|     6    |
|     7    |
+----------+
7 rows in set (0.00 sec)
```

Retrieving Multiple Columns

We can display more than one column(s) from a table using SELECT statement:

Syntax:

SELECT <column name1>,<column name2> FROM <table name>;

Example:

Now, Ms. Sujata displays two columns :Roll numbers and names of all the students.

```
mysql> SELECT Rollno, Name  FROM Student;
+----------+-----------------+
|  Rollno  | Name            |
+----------+-----------------+
|    1     | Siddharth Sehgal |
|    2     | Gurpreet Kaur   |
```

```
|      3      | Monica Rana        |
|      4      | Jatinder Sen       |
|      5      | George Jacob       |
|      6      | Deepa Bhandari     |
|      7      | Akshay Nath        |
+-----------+-------------------+
7 rows in set (0.00 sec)
```

Changing the order of display of Columns

We can display columns in any order by specifying the columns in that order in SELECT statement . The following statement displays Names first and then Roll numbers from the table Student.

```
mysql> SELECT Name,Rollno FROM Student;
+-------------------+--------+
| Name              | Rollno |
+-------------------+--------+
| Siddharth Sehgal  |      1 |
| Gurpreet Kaur     |      2 |
| Monica Rana       |      3 |
| Jatinder Sen      |      4 |
| George Jacob      |      5 |
| Deepa Bhandari    |      6 |
| Akshay Nath       |      7 |
+-------------------+--------+
7 rows in set (0.00 sec)
```

In the Output, notice that the first column displaying names is left-justified and the second column displaying roll numbers is right justified. The format of output follows the pattern that character data is left justified and numeric data is right justified.

Retrieving all Columns

To see all the columns of the table, we can write * in place of names of all the columns. The columns are displayed in the order in which they are stored in the table.

Ms. Sujata uses the following statement to see all the columns of her table:

```
mysql> SELECT * FROM Student;
+--------+------------------+--------+--------+
| Rollno | Name             | Gender | Marks1 |
+--------+------------------+--------+--------+
|    1   | Siddharth Sehgal | M      |   93.0 |
|    2   | Gurpreet Kaur    | F      |   91.0 |
|    3   | Monica Rana      | F      |   93.0 |
|    4   | Jatinder Sen     | M      |   78.0 |
|    5   | George Jacob     | M      |   76.0 |
|    6   | Deepa Bhandari   | F      |   77.0 |
|    7   | Akshay Nath      | M      |   65.0 |
+--------+------------------+--------+--------+
7 rows in set (0.00 sec)
```

! *The asterisk (*) means "All". SELECT * means display all columns*

Eliminating duplicate values

By default data is displayed from all the rows of the table, even if the data in the result is duplicated. Using the keyword DISTINCT, the duplicate values can be eliminated in the result. When DISTINCT keyword is specified, only one instance of the duplicated data is shown. The following query without the DISTINCT keyword shows 7 rows while the same query with DISTINCT keyword shows 6 rows as duplicate data 93 is displayed only once.

```
mysql> SELECT Marks1 FROM Student;
+--------+
| Marks1  |
+--------+
|   93.0  |
|   91.0  |
|   93.0  |
|   78.0  |
|   76.0  |
|   77.0  |
|   65.0  |
+--------+
7 rows in set (0.00 sec)
```

93 displayed twice

```
mysql> SELECT DISTINCT Marks1 FROM Student;
+--------+
| Marks1  |
+--------+
|   93.0  |
|   91.0  |
|   78.0  |
|   76.0  |
|   77.0  |
|   65.0  |
+--------+
6 rows in set (0.00 sec)
```

Retrieving Data From All Rows

If we write the keyword ALL in place of DISTINCT, then the result of SELECT query displays all the values including duplicate values. The output is the same as what we get when we do not write DISTINCT keyword in the SELECT query.

Using Arithmetic Operators with SELECT

Arithmetic operators perform mathematical calculations. In SQL the following arithmetic operators are used:

Operator	What it does
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (or remainder)

Modulus operator (%) returns the remainder of a division.

We can perform simple arithmetic computations on data using SELECT statement. Ms. Sujata thinks what if all my students had secured 5 marks more. She enters the following statement to display marks of all students increased by 5.

```
mysql> SELECT Marks1+5 FROM Student;
+----------+
| Marks1+5 |
+----------+
|   98.0   |
|   96.0   |
|   98.0   |
|   83.0   |
|   81.0   |
|   82.0   |
|   70.0   |
+----------+
7 rows in set (0.02 sec)
```

Marks1 column is displayed increased by 5. The actual values are not increased in the table.

Here are some more examples:

```
mysql> SELECT Name,Marks1+0.05*Marks1 FROM Student ;
mysql> SELECT Name,Marks1-10 FROM Student ;
mysql> SELECT Name,Marks1/2 FROM Student ;
```

! *Using these operators on tables does not create new columns in the tables or change the actual data values. The results of the calculations appear only in the output.*

In the above examples, arithmetic calculations were based on Student table. Arithmetic calculations may not always be based on tables. For example when we want to compute 7*3+1, there is no table to be referenced. In such queries no FROM clause is used :

```
mysql> SELECT 7*3+1;
+-------+
| 7*3+1  |
+-------+
|  22    |
+-------+
1 row in set (0.09 sec)
```

```
mysql> SELECT 71+34;
+-------+
| 71+34  |
+-------+
|   105  |
+-------+
1 row in set (0.00 sec)
```

Using Column Alias

Till now, we have seen that when the result of an SQL statement is displayed, the heading displayed at the top of column is same as the column name in the table or the arithmetic operation being done on the Column.

While displaying marks from the table Student, Ms. Sujata wants the output to display a column heading (for Marks) that is easier to understand and is more meaningful and presentable like "Marks Secured" instead of Marks1. Column alias lets different name (heading) to appear for a column than the actual one in the output. She enters the statement like this:

```
mysql> SELECT Marks1 AS "Marks Secured" FROM Student;
+---------------+
| Marks Secured |
+---------------+
|          93.0 |
|          91.0 |
|          93.0 |
|          78.0 |
|          76.0 |
|          77.0 |
|          65.0 |
+---------------+
7 rows in set (0.00 sec)
```

Notice that the column Marks1 has been given the column heading "Marks Secured" . If a column alias consists of more than one word ,then it should be enclosed in quotes as in "Marks Secured",otherwise error message is displayed.

! Using Column Alias does not rename a column. It simply displays a different column name in the output.

The AS keyword between the column name and alias is optional. We can also write **`SELECT  Marks1  "Marks Secured" FROM Student;`**

Putting text in Query output

Can Ms. Sujata make the query output more presentable by inserting items such as symbols or text in the query output ? Yes. She can. She uses the following statement.

```
mysql> SELECT Rollno,Name,'has secured marks',marks1 FROM
 student;
+--------+------------------+-------------------+--------+
| Rollno | Name             | has secured marks | marks1 |
+--------+------------------+-------------------+--------+
|      1 | Siddharth Sehgal | has secured marks |   93.0 |
|      2 | Gurpreet Kaur    | has secured marks |   91.0 |
|      3 | Monica Rana      | has secured marks |   93.0 |
|      4 | Jatinder Sen     | has secured marks |   78.0 |
|      5 | George Jacob     | has secured marks |   76.0 |
|      6 | Deepa Bhandari   | has secured marks |   77.0 |
|      7 | Akshay Nath      | has secured marks |   65.0 |
+--------+------------------+-------------------+--------+
7 rows in set (0.00 sec)
```

The text 'has secured marks' is displayed with every row of the table.

Retrieving specific rows - WHERE clause

Tables usually contain many rows. Mostly, we do not want to display all the rows of a table. Certain rows can be displayed based on the criteria for selection of rows using the keyword WHERE. The WHERE clause is used to filter records. It is used to extract only those records that fulfill a specified criterion.

Syntax:

SELECT <column name1> [,<column name> ,….] FROM <table name>

WHERE <condition>;

Ms. Sujata wants to display the names and marks of all those students who have secured marks above 80, she enters:

```
mysql> SELECT Name,Marks1 FROM Student WHERE Marks1 > 80;
+-------------------+----------+
| Name              |  Marks1  |
+-------------------+----------+
| Siddharth Sehgal  |   93.0   |
| Gurpreet Kaur     |   91.0   |
| Monica Rana       |   93.0   |
+-------------------+----------+
3 rows in set (0.00 sec)
```

She thinks "What would be the marks of my students if they were increased by 5 for all those students who secured marks below 80?" She enters the statement:

```
mysql> SELECT Name,Marks1+5 FROM Student WHERE marks1 <80;
+-----------------+----------+
| Name            | Marks1+5 |
+-----------------+----------+
| Jatinder Sen    |   83.0   |
| George Jacob    |   81.0   |
| Deepa Bhandari  |   82.0   |
| Akshay Nath     |   70.0   |
+-----------------+----------+
4 rows in set (0.00 sec)
```

Relational Operators

Ms. Sujata has used the relational operator "<" in the statement entered above. Relational operators are used to compare two values. The result of the comparison is True or False. They are used with WHERE clause. Given below are all the relational operators used in MySQL alongwith their functions:

Operator	What it does
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!= or <>	Not equal to

She enters the following statement to see the details of students who have secured at least 93 marks.

```
mysql> SELECT * FROM Student WHERE Marks1>=93;
+--------+------------------+--------+--------+
| Rollno | Name             | Gender | Marks1 |
+--------+------------------+--------+--------+
|   1    | Siddharth Sehgal | M      |  93.0  |
|   3    | Monica Rana      | F      |  93.0  |
+--------+------------------+--------+--------+
2 rows in set (0.06 sec)
```

! When we use relational operators with character data type, < means earlier in the alphabet and > means later in the alphabet. 'Aman' < 'Ayan' as 'm' comes before 'y' in alphabet.

Some more examples of queries involving relational expressions:

```
mysql> SELECT Name,Marks1 FROM Student WHERE Marks1 <60;
mysql> SELECT * FROM Student WHERE Name = 'Gurpreet Kaur';
mysql> SELECT RollNo,Marks1 FROM Student WHERE Rollno <=3;
mysql> SELECT RollNo,Marks1 FROM Student WHERE Rollno <>3;
mysql> SELECT RollNo,Marks1 FROM Student WHERE Name <>'Mani
Kumar';
```

Logical Operators

OR, AND, NOT logical operators are used in SQL. Logical operators OR and AND are used to connect relational expressions in the WHERE clause. If any of the comparisons are true, OR returns TRUE. AND requires both conditions to be true in order to return TRUE. NOT negates a condition. If a condition returns a True value, adding NOT causes the condition to return a False value and vice versa.

The symbol || can be used in place of OR, && can be used in place of AND, ! can be used in place of NOT operator.

Ms. Sujata uses the following statement (with Logical operator AND) to display Roll numbers and names of students who have secured marks above 70 but below 80.

```
mysql> SELECT Rollno, Name,Marks1 FROM Student WHERE Marks1 >
70 AND Marks1 < 80;
+--------+----------------+--------+
| Rollno | Name           | Marks1 |
+--------+----------------+--------+
|    4   | Jatinder Sen   |  78.0  |
|    5   | George Jacob   |  76.0  |
|    6   | Deepa Bhandari |  77.0  |
+--------+----------------+--------+
3 rows in set (0.01 sec)
```

Some example of SQL statements with Logical operators are shown below.

```
mysql> SELECT Empnumber, EmpName FROM Employee WHERE
Department = 'Accoumts' OR Department = 'Personnel';

mysql> SELECT Empnumber, EmpName FROM Employee WHERE
Department = 'Accoumts' AND Designation = 'Manager';

mysql> SELECT Empnumber, EmpName FROM Employee WHERE
NOT(Designation = 'Manager');

mysql> SELECT Name,TotalMarks FROM Candidate WHERE
writtenmarks>80 || Interviewmarks>10;

SELECT Name,TotalMarks FROM Candidate WHERE
writtenmarks>80 && Interviewmarks>10;
```

Using Parenthesis in WHERE clause

Sometimes we have to write a criterion using a combination of AND and OR. The parentheses not only help us visually see how things are grouped together but they also let the DBMS know exactly what to do.

```
SELECT *
    FROM Emp
    WHERE first_name='Amit' AND (last_name='Sharma' OR
    last_name='Verma');
```

So, how does that work? It simply states that we are looking for anyone with the first name as Amit and the last name as Sharma or Verma. They must have the first name as Amit but can have the last name as either Sharma or Verma.

Condition based on Range

The BETWEEN operator defines the range of values within which the column values must fall into to make the condition true. The range includes both the upper and lower values.

Ms. Sujata uses the following statement to display roll numbers and marks of students who have secured marks in the range 70 to 80 (including 70 and 80).

```
mysql> SELECT Rollno,Name,Marks1 FROM Student WHERE Marks1
BETWEEN 70 AND 80;
+--------+----------------+--------+
| Rollno | Name           | Marks1 |
+--------+----------------+--------+
|   4    | Jatinder Sen   |   78.0 |
|   5    | George Jacob   |   76.0 |
|   6    | Deepa Bhandari |   77.0 |
+--------+----------------+--------+
3 rows in set (0.06 sec)
```

The following statement displays roll numbers and marks of students who have secured marks other than the ones in the range 70 to 80(including 70 and 80).

```
mysql> SELECT Rollno,Name,Marks1 FROM Student WHERE Marks1
NOT BETWEEN 70 AND 80;
```

! BETWEEN displays all values between the lower and the upper values including the lower and the upper values.

To display marks in the range 70 to 80, Ms. Sujata could have used the following statement to give the same output as the one using BETWEEN operator.

```
mysql> SELECT Rollno,Name,Marks1 FROM Student WHERE Marks1>=70
AND Marks1<=80;
```

Condition based on a List

The IN operator selects values that match any value in the given list of values .If we want to display data of Students whose marks are 68 or 76 or 78, we can use the IN operator like this:

```
mysql> SELECT Rollno, Name, Marks1 FROM Student WHERE Marks1 IN
(68,76,78);

+------------+--------------------+-------------+
| Rollno     | Name               | Marks1      |
+------------+--------------------+-------------+
|     4      | Jatinder Sen       |    78.0     |
|     5      | George Jacob       |    76.0     |
+------------+--------------------+-------------+
2 rows in set (0.00 sec)
```

In an Employee table, to display rows where State is 'DELHI' or 'MUMBAI' or 'UP', we write the query like this:

```
SELECT * FROM Employee WHERE State IN ('DELHI','MUMBAI','UP');
```

In an Employee table, to display all rows except those that have State as 'DELHI' or 'MUMBAI' or 'UP', we write the query like this:

```
SELECT * FROM Employee WHERE State NOT IN
('DELHI','MUMBAI','UP');
```

Till now Ms. Sujata's table Student has 7 rows. She wants to populate it with some more rows.She uses the following INSERT INTO statement.

```
INSERT INTO Student VALUES (8,'Samdisha Sen','F',76);
INSERT INTO Student VALUES (9,'Geeta Sen Sharma','F',91);
INSERT INTO Student VALUES (10,'Geet Kadamb','M',66);
INSERT INTO Student VALUES (11,'Aman Ali','M',92);
INSERT INTO Student VALUES (12,'Ayan Ali','M',87);
```

She checks that the table has the new rows inserted by using the following SELECT statement:

```
SELECT * FROM Student;
+--------+------------------+--------+--------+
| Rollno | name             | Gender | Marks1 |
+--------+------------------+--------+--------+
|    1   | Siddharth Sehgal |   M    |   93   |
|    2   | Gurpreet Kaur    |   F    |   91   |
|    3   | Monica Rana      |   F    |   93   |
|    4   | Jatinder Sen     |   M    |   78   |
|    5   | George Jacob     |   M    |   76   |
|    6   | Deepa Bhandari   |   F    |   77   |
|    7   | Akshay Nath      |   M    |   65   |
|    8   | Samdisha Sen     |   F    |   76   |
|    9   | Geeta Sen Sharma |   F    |   91   |
|   10   | Geet Kadamb      |   M    |   66   |
|   11   | Aman Ali         |   M    |   92   |
|   12   | Ayan Ali         |   M    |   87   |
+--------+------------------+--------+--------+
12 rows in set (0.00 sec)
```

Condition based on pattern matches

Sometimes while trying to remember somebody's name, you remember a part of his/her name but not the exact name. In such cases, MySQL has wildcards to help you out. % and _ are two wild card characters. The percent (%) symbol is used to represent any sequence of zero or more characters. The underscore (_) symbol is used to represent a single character.

LIKE clause is used to fetch data which matches the specified pattern from a table. The LIKE clause tells the DBMS that we won't be doing a strict comparison like = or < or > but we will be using wildcards in our comparison.

Syntax:

```
SELECT <column name>, [<column name>...]
WHERE <column name> LIKE Pattern [AND [OR]] <Condition2>;
```

For example, Ms. Sujata wants to display details of students who have their names ending with 'Sen', she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE '%Sen';
+--------+--------------+--------+----------+
| Rollno | Name         | Gender | Marks1   |
+--------+--------------+--------+----------+
|      4 | Jatinder Sen | M      |     78.0 |
|      8 | Samdisha Sen | F      |     76.0 |
+--------+--------------+--------+----------+
2 rows in set (0.00 sec)
```

To display rows from the table Student with names starting with 'G', she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE 'G%';
+--------+-----------------+--------+----------+
| Rollno | name            | Gender | Marks1   |
+--------+-----------------+--------+----------+
|      2 | Gurpreet Kaur   | F      |     91.0 |
```

```
|      5  | George Jacob       | M        |     76.0  |
|      9  | Geeta Sen Sharma   | F        |     91.0  |
|     10  | Geet Kadamb        | M        |     66.0  |
+--------+------------------+--------+----------+
4 rows in set (0.02 sec)
```

To display rows that have names starting with 'G' and ending with 'b', she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE 'G%b';
+----------+------------------+------------+----------+
| Rollno   | name             | Gender     | Marks1   |
+----------+------------------+------------+----------+
|      5   | George Jacob     | M          |     76.0 |
|     10   | Geet Kadamb      | M          |     66.0 |
+----------+------------------+------------+----------+
2 rows in set (0.00 sec)
```

To display rows from the table Student that have 'Sen' anywhere in their names, she enters:

```
mysql> SELECT * FROM Student WHERE Name LIKE '%Sen%';
+----------+------------------+------------+----------+
| Rollno   | name             | Gender     | Marks1   |
+----------+------------------+------------+----------+
|      4   | Jatinder Sen     | M          |    78    |
|      8   | Samdisha Sen     | F          |    76    |
|      9   | Geeta Sen Sharma | F          |    91    |
+----------+------------------+------------+----------+
3 rows in set (0.00 sec)
```

To display rows that have names starting with 'A' and then having any 4 characters and ending with 'Ali', she uses underscore wild card like this:

```
mysql> SELECT * FROM Student WHERE Name LIKE 'A____Ali';
+----------+-----------------+------------+----------+
| Rollno   | name            | Gender     | Marks1   |
+----------+-----------------+------------+----------+
|     11   | Aman Ali        | M          |     92.0 |
|     12   | Ayan Ali        | M          |     87.0 |
+----------+-----------------+------------+----------+
2 rows in set (0.00 sec)
```

Some more examples

'Am%' matches any string starting with Am.

'%Singh%' matches any string containing 'Singh'

'%a' matches any string ending with 'a'

'___' matches any string that is exactly 3 characters long.

'__%' matches any string that has at least 2 characters.

'___g' matches any string that is 4 characters long with any 3 characters in the beginning but 'g' as the 4th character.

The keyword NOT LIKE is used to select the rows that do not match the specified pattern.

To display rows from the table Student that have names not starting with 'G', she enters:

```
mysql> SELECT * FROM Student WHERE Name NOT LIKE 'G%';
```

Precedence of Operators

All the operators have precedence. Precedence is the order in which different operators are evaluated in the same expression. When evaluating an expression containing multiple operators, operators with higher precedence are evaluated before evaluating those with lower precedence. Operators with equal precedence are evaluated from left to right within the expression. Parenthesis can be used to change the preference of an operator. Various operators in descending order of precedence (top to bottom) are listed below:

```
!
- (unary minus)
^
*, /, DIV, %, MOD
-, +
=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, IN
BETWEEN,
NOT
&&, AND
||, OR
```

NULL

Sometimes, you don't know the represent value for a column. In a table, you can store these unknowns as NULL. NULL means a value that is unavailable, unassigned, unknown or inapplicable. NULL is not the same as zero or a space or any other character. . In a table NULL is searched for using IS NULL keywords.

```
SELECT * FROM Student WHERE Name IS NULL;
SELECT * FROM Employee WHERE Commission IS NULL;
```

NOT NULL values in a table can be searched using IS NOT NULL.

```
SELECT * FROM Employee WHERE Commission IS NOT NULL;
```

!

If any column value involved in an arithmetic expression is NULL, the result of the arithmetic expression is also NULL.

Sorting the Results- ORDER BY

The result obtained using SELECT statement is displayed in the order in which the rows were entered in the table using the INSERT INTO statement. The results of the SELECT statement can be displayed in the ascending or descending values of a single column or multiple columns using ORDER BY clause.

```
SELECT <column name>, [<column name>…]
[WHERE <Condition list>]
ORDER BY <column name>;
```

Now, Ms. Sujata wants to display data of students in ascending order of their marks, she enters the following statement:

```
mysql> SELECT * FROM Student ORDER BY Marks1;
+--------+------------------+--------+--------+
| Rollno | Name             | Gender | Marks1 |
+--------+------------------+--------+--------+
|      7 | Akshay Nath      | M      |   65.0 |
|     10 | Geet Kadamb      | M      |   66.0 |
|      8 | Samdisha Sen     | F      |   76.0 |
|      5 | George Jacob     | M      |   76.0 |
|      6 | Deepa Bhandari   | F      |   77.0 |
|      4 | Jatinder Sen     | M      |   78.0 |
|     12 | Ayan Ali         | M      |   87.0 |
|      9 | Geeta Sen Sharma | F      |   91.0 |
|      2 | Gurpreet Kaur    | F      |   91.0 |
|     11 | Aman Ali         | M      |   92.0 |
|      3 | Monica Rana      | F      |   93.0 |
|      1 | Siddharth Sehgal | M      |   93.0 |
+--------+------------------+--------+--------+
12 rows in set (0.08 sec)
```

Similarly, to display data of students in ascending order of their names (meaning alphabetically sorted on names), she uses the following statement:

```
mysql> SELECT * FROM Student ORDER BY Name;
+--------+------------------+--------+--------+
| Rollno | Name             | Gender | Marks1 |
+--------+------------------+--------+--------+
|      7 | Akshay Nath      | M      |   65.0 |
|     11 | Aman Ali         | M      |   92.0 |
|     12 | Ayan Ali         | M      |   87.0 |
|      6 | Deepa Bhandari   | F      |   77.0 |
|     10 | Geet Kadamb      | M      |   66.0 |
|      9 | Geeta Sen Sharma | F      |   91.0 |
|      5 | George Jacob     | M      |   76.0 |
|      2 | Gurpreet Kaur    | F      |   91.0 |
|      4 | Jatinder Sen     | M      |   78.0 |
|      3 | Monica Rana      | F      |   93.0 |
|      8 | Samdisha Sen     | F      |   76.0 |
|      1 | Siddharth Sehgal | M      |   93.0 |
+--------+------------------+--------+--------+
12 rows in set (0.00 sec)
```

To display data in descending order, DESC keyword is used in ORDER BY clause. However it is not necessary to specify ASC for ascending order as it is the default order.

Ms. Sujata uses the following statement to display details of her students in descending order of marks.

```
mysql> SELECT * FROM Student ORDER BY Marks1 DESC;
+--------+------------------+--------+--------+
| Rollno | Name             | Gender | Marks1 |
+--------+------------------+--------+--------+
|      1 | Siddharth Sehgal | M      |   93.0 |
|      3 | Monica Rana      | F      |   93.0 |
|     11 | Aman Ali         | M      |   92.0 |
```

```
|       2  | Gurpreet Kaur      | F        |   91.0  |
|       9  | Geeta Sen Sharma   | F        |   91.0  |
|      12  | Ayan Ali           | M        |   87.0  |
|       4  | Jatinder Sen       | M        |   78.0  |
|       6  | Deepa Bhandari     | F        |   77.0  |
|       8  | Samdisha Sen       | F        |   76.0  |
|       5  | George Jacob       | M        |   76.0  |
|      10  | Geet Kadamb        | M        |   66.0  |
|       7  | Akshay Nath        | M        |   65.0  |
+--------+------------------+--------+--------+
12 rows in set (0.00 sec)
```

Ms. Sujata wants to display all the rows of the table Student in ascending order of Marks1. But if several students have the same value for Marks1, for them she wants the display to be in ascending order of names.

She can order results on more than one column like this:

```
mysql> SELECT * FROM Student ORDER BY Marks1,Name;
+--------+------------------+--------+--------+
| Rollno | Name               | Gender | Marks1 |
+--------+------------------+--------+--------+
|       7  | Akshay Nath        | M        |   65.0  |
|      10  | Geet Kadamb        | M        |   66.0  |
|       5  | George Jacob       | M        |   76.0  |
|       8  | Samdisha Sen       | F        |   76.0  |
|       6  | Deepa Bhandari     | F        |   77.0  |
|       4  | Jatinder Sen       | M        |   78.0  |
|      12  | Ayan Ali           | M        |   87.0  |
|       9  | Geeta Sen Sharma   | F        |   91.0  |
|       2  | Gurpreet Kaur      | F        |   91.0  |
|      11  | Aman Ali           | M        |   92.0  |
|       3  | Monica Rana        | F        |   93.0  |
|       1  | Siddharth Sehgal   | M        |   93.0  |
+--------+------------------+--------+--------+
12 rows in set (0.00 sec)
```

The following statement displays rows in descending order of marks but if several students have the same value for marks, for them the display is in ascending order of names.

```
mysql> SELECT * FROM Student ORDER BY Marks1 DESC,Name;
```

Ms. Sujata wants to display details of students who have secured marks above 90 in ascending order of names. She uses the following statement:

```
mysql> SELECT * FROM Student WHERE Marks1 > 90 ORDER BY Name;
+--------+-----------------+--------+--------+
| Rollno | Name            | Gender | Marks1 |
+--------+-----------------+--------+--------+
|     11 | Aman Ali        | M      |   92.0 |
|      9 | Geeta Sen Sharma| F      |   91.0 |
|      2 | Gurpreet Kaur   | F      |   91.0 |
|      3 | Monica Rana     | F      |   93.0 |
|      1 | Siddharth Sehgal| M      |   93.0 |
+--------+-----------------+--------+--------+
5 rows in set (0.02 sec)
```

She can also write ORDER BY 2 in place of ORDER BY Name as Name is the second column.

Sorting on Column Alias

If a Column alias is defined on a column, we can use it for displaying rows in an ascending or descending order using ORDER BY clause:

```
mysql> SELECT Name, Marks1 AS Total
 FROM Student
ORDER BY Total;
```

Column Alias

Column Alias

More About Inserting Rows

We have already used the INSERT INTO statement to add new rows to a table. It requires three values:

- the name of the table
- the names of the columns in the table which have to be populated (optional)
- corresponding values for the columns.

Syntax:

```
INSERT INTO <tablename>[<column list>] VALUES (<value>, <value>
...);
```

Ms. Sujata uses the following statement to add a row to her table named Student.

```
mysql>  INSERT INTO Student VALUES (13,'Mani Kumar','M',97);
Query OK, 1 row affected (0.06 sec)
```

In the above example note that the column names for which data values are populated are not specified. We can omit the column names if values have to be inserted for every column in a table but in such a case the data values for each column must match exactly the default order in which they appear in the table (as shown in the DESCRIBE statement), and a value must be provided for each column.

Ms. Sujata could have explicitly specified all the column names of the table or specific columns for which data is to be inserted and the corresponding data values for those columns like this:

```
INSERT INTO Student(Rollno,Name,Gender,Marks1) VALUES
(13,'Mani Kumar','M',97);
Query OK, 1 row affected (0.06 sec)
```

Ms. Sujata wants to insert a row for Student with roll number 14 who secured 45 marks. She however does not have that student's name. The following INSERT INTO statement inserts values for specific columns namely Rollno and Marks1. Those columns that are not specified in the list will have the default values (if defined) else NULLs will be inserted.

```
mysql>  INSERT INTO Student(Rollno,Marks1) VALUES (14,45);
Query OK, 1 row affected (0.05 sec)
```

Since values are provided only for Roll number and marks, Ms. Sujata uses the SELECT statement and notices the word NULL displayed for Name and Gender for Roll number 14 :

```
mysql> SELECT * FROM Student WHERE Rollno =14;
+--------+------+--------+----------+
| Rollno  | name | Gender  | Marks1    |
+--------+------+--------+----------+
|     14  | NULL | NULL    |     45.0 |
+--------+------+--------+----------+
1 row in set (0.00 sec)
```

Explicitly Inserting NULL Values

We have learnt that if a column can hold NULL values, it can be omitted from the INSERT INTO statement. INSERT INTO statement will automatically insert a null value in that column. This is called Implicitly inserting a NULL value.

```
mysql> INSERT INTO Student(Rollno,Name,Gender)
Values(15,'Charvi Chanana','F');
Query OK, 1 row affected (0.11 sec)
```

In the above INSERT INTO statement Marks1 column is omitted, therefore NULL value will be inserted for it.

We can also explicitly add NULL value by using the NULL keyword in the VALUES list for those columns that can hold null values.

```
mysql> INSERT INTO Student Values(14,'Siddharth
Sehgal','M',NULL);
Query OK, 1 row affected (0.11 sec)
```

! *A NULL value means no value has been entered for that column i.e. the value for that column is not known.*

Inserting Date Values

The default way to store a date in MySQL is with the type DATE. Below is the format of a DATE.

YYYY-MM-DD

To insert the current date into a table, MySQL's built-in function CURDATE() can be used in the query. Following are some examples of inserting date values.

```
mysql> INSERT INTO my_table (idate) VALUES (19970505);
mysql> INSERT INTO my_table (idate) VALUES ('97-05-05');
mysql> INSERT INTO my_table (idate) VALUES ('1997.05.05');
mysql> INSERT INTO my_table (idate) VALUES ('0000-00-00');
```

! While Inserting data:

- Text values must be enclosed in quotes.
- Standard date format is "yyyy-mm-dd".
- Standard time format is "hh:mm:ss".
- Quotes are required around the standard date and time formats.

UPDATE STATEMENT

In the table student, Ms. Sujata entered a student's marks as 93. Suppose, that student found out that one of her answers was unchecked and got her marks increased by 1. How would Ms. Sujata change it in the table? She can use the UPDATE statement to modify existing data in the table.

(a) Syntax:

```
UPDATE <table_name>
SET <column name> = <value>, [ <column name> = <value>, …]
[WHERE <condn>];
```

The statement can be used to update one or more columns together. WHERE clause helps in updation of particular rows in a table.

The following statement sets the marks(Mark1) of all the rows to 94.

```
UPDATE Student SET Marks1 = 94;
```

The following statement sets the marks(Mark1) of the row with name as 'Monica Rana' to 94.

```
mysql> UPDATE Student SET Marks1 = 94 WHERE name =
'Monica Rana';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

The marks displayed from the table shows 94 marks now:

```
mysql> SELECT Name,Marks1 FROM Student WHERE Name =
 'Monica Rana';
Output:
+---------------+--------+
| Name          | Marks1 |
+---------------+--------+
| Monica Rana   |    94  |
+---------------+--------+
1 row in set (0.00 sec)
```

What if Ms. Sujata wants to change the name and marks both at the same time? Multiple columns can also be updated at one time. The following statement changes the name to "Chhavi Chanana" and Marks to 90 for the roll number 15.

```
mysql> UPDATE Student SET name = 'Chhavi  Chanana',
Marks1= 90  WHERE Rollno = 15;
Output:
Query OK , 1 row affected (0.8 sec)
```

DELETE STATEMENT

Sometimes students leave school or an employee leaves an organization. Their rows have to be deleted from the table. Deleting data from a table is very simple. DELETE statement is used to delete rows from a table. DELETE removes the entire row, not the individual column values. Care must be taken while using this statement as accidentally important data may get deleted.

Syntax:

mysql> DELETE FROM < tablename> [Where < condn>];

One of the students with Roll number 14 has left the school and Ms. Sujata wants to delete his/her row. She uses the following statement to delete the row with roll number 14.

```
mysql> DELETE FROM Student WHERE Rollno = 14;
Query OK, 1 row affected (0.03 sec)
```

DELETE statement can be used to delete all rows of the table also . The following statement can be used to delete all the rows from Student table.

```
mysql> DELETE from Student;
mysql > Select * FROM Student;
+--------+---------+--------+-------+
|Rollno  | Name    | Gender |Marks1 |
+--------+---------+--------+-------+
|        |         |        |       |
+--------+---------+--------+-------+
0 row in set (0.01 sec)
```

Know more

The MySQL database management system contains an enormous amount of functionality and power. Using a simple set of statements for inserting, retrieving, deleting and updating data, we can develop quite a useful set of databases and tables.

To learn more about MySQL you may visit the website:

http://www.mysqltutorial.org

Summary

1. CREATE DATABASE statement is used to create a new database.
2. CREATE TABLE statement is used to create a new table.
3. INSERT INTO statement is used to insert a new row in a table.
4. The SELECT statement is used to fetch data from one or more database tables.
5. SELECT * means display all columns.
6. The WHERE clause is used to select specific rows.
7. The DESCRIBE statement is used to see the structure of a table.
8. We can change the structure of a table ie. add, remove or change its column(s) using the ALTER TABLE statement.
9. The keyword DISTINCT is used to eliminate redundant data from display.
10. (a) Logical operators OR and AND are used to connect relational expressions in the WHERE clause.

    (b) Logical operator NOT is used to negate a condition.
11. The BETWEEN operator defines the range of values that the column values must fall into to make the condition true.
12. The IN operator selects values that match any value in the given list of values.
13. % and _ are two wild card characters. The percent (%) symbol is used to represent any sequence of zero or more characters. The underscore (_) symbol is used to represent a single character.
14. NULL represents a value that is unavailable, unassigned, unknown or inapplicable.
15. The results of the SELECT statement can be displayed in the ascending or descending order of a single column or columns using ORDER BY clause.
16. UPDATE statement is used to modify existing data in a table.
17. DELETE statement is used to delete rows from a table.

Multiple Choice questions

1. **Which statement is used to extract data from a table?**

   A. SELECT

   B. DISPLAY

   C. READ

   D. EXTRACT

2. **How do you select all the columns from a table named "Employee"?**

   A. `SELECT [all] FROM Employee;`

   B. `SELECT Employee;`

   C. `SELECT * BY Employee;`

   D. `SELECT * FROM Employee ;`

3. **How do you select a column named "IName" from a table named "Inventory"?**

   A. `SELECT  Inventory  FROM Iname;`

   B. `DISPLAY  Iname   FROM Inventory;`

   C. `SELECT  Iname   FROM Inventory;`

   D. `SELECT  Iname, Inventory  FROM Iname;`

4. **Which of the following are valid column names?**

   A. Marks Eng

   B. 66_Marks

   C. Marks_Eng

   D. #Eng_Marks

5. **SELECT statement can be used to perform these functions.**

   A. Insert rows into a table.

   B. Choose and display columns from a table.

   C. Modify data in a table.

   D. Select and display structure of a table

6. **Which statement is used to insert new data in a table?**

   A. ADD RECORD

   B. INSERT RECORD

   C. INSERT INTO

   D. INSERT ROW

7. **How would you display all those rows from a table named "Friends" where the value of the column "Hobbies" is "SWIMMING"**

   A. `SELECT ALL FROM Friends WHERE Hobbies IS 'SWIMMING';`

   B. `SELECT * FROM Friends WHERE Hobbies='SWIMMING';`

   C. `SELECT * FROM Friends WHERE Hobbies = 'Swimming" ;`

   D. `SELECT ALL  FROM Friends WHERE Hobbies  'SWIMMING' ;`

8. **Which statement is used to modify data in a table?**

   A. `CHANGE`

   B. `MODIFY`

   C. `UPDATE`

   D. `SAVE AS`

9. **Which SQL statement is used to delete data from a table?**

   A. `DELETE`

   B. `DROP`

   C. `TRUNCATE`

   D. `REMOVE`

10. **How do you select all the rows from a table named "Student" where the value of the column "FName" starts with "G"?**

   A. `SELECT * FROM Student WHERE FName LIKE 'G_' ;`

   B. `SELECT * FROM Student WHERE FName='G';`

   C. `SELECT * FROM Student WHERE FName LIKE 'G%' ;`

   D. `SELECT * WHERE Student WHERE FName='%G%' ;`

11. The OR operator displays a record if ANY of the conditions listed are true. The AND operator displays a record if ALL of the conditions listed are true

A. False

B. True

12. Which keyword is used to return only different values in a column?

A. `DIFFERENT`

B. `EXCLUSIVE`

C. `DISTINCT`

D. `UNIQUE`

13. Which SQL keyword(s) is/are used to sort the rows in the output:

A. `SORTED ORDER`

B. `SORT`

C. `SORT BY`

D. `ORDER BY`

14. How would you return all the rows from a table named "Item" sorted in descending order on the column "IName"?

A. `SELECT * FROM Item SORT 'IName' DESC;`

B. `SELECT * FROM Item ORDER BY IName DESC ;`

C. `SELECT * FROM Item ORDER IName DESC ;`

D. `SELECT * FROM Item SORT BY 'IName' DESC ;`

15. How can you insert a new row into the "Store" table?

A. `INSERT (1,'Abc Rice') INTO Store;`

B. `INSERT VALUES (1,'Abc Rice') INTO Store ;`

C. `INSERT INTO Store VALUES (1,'Abc Rice');`

D. `ADD ROW Store values(1,'Abc Rice');`

16. **Which statement is appropriate to change the first name "Madhur" to "Mridul" in the "FName" column in the 'Student' table?**

    A. UPDATE Student SET FName='Mridul' WHERE FName='Madhur' ;

    B. MODIFY Student SET FName='Madhur' INTO FName='Mridul ;

    C. UPDATE Student SET FName='Madhur' INTO FName='Mridul' ;

    D. UPDATE Student SET FName='Madhur' WHERE FName='Mridul' ;

17. **How can you delete the rows with marks below 33 in the 'Student' Table?**

    A. DELETE FROM Student WHERE marks <=33;

    B. DELETE marks < 33 FROM Student ;

    C. DELETE ROW marks <33 FROM Student;

    D. DELETE FROM Student WHERE marks <33;

Exercises

Answer the following questions.

a) Define the following terms:

   i) Keyword ii) Clause iii) Statement

b) Which statement is used to select a database and make it current?

c) How is a database related to table(s)?

d) Write SQL statement to view names of all the tables contained in the current database

e) Which keyword is used to eliminate redundant data?

f) In INSERT INTO statement, while giving data values, how should text values be supplied?

g) What is NULL? What happens when you perform arithmetic calculations on NULL values?

h) Write SQL statement to display the result of arithmetic expression 78*2 on screen?

i) Is there any difference in the output of the following statements :

```
Select * from Emp;
SELECT * FROM EMP;
```

j) List two categories into which MySQL statements can be categorized. Give examples of 2 statements in each category.

k) Write the minimum number of column(s) that must be specified while creating a table using CREATE TABLE statement.

l) What happens if you give a CREATE TABLE statement to create a table named 'Item' and a table named 'Item' already exists?

m) Write any 4 things that are displayed when you display the structure of a table using DESCRIBE statement.

n) Consider the following INSERT INTO statement:

```
INSERT INTO Emp(Empid,Salary) VALUES ('I201',25000);
```

What values are assigned to the columns Empid and Salary respectively?

o) In which order are the columns displayed when we give a SELECT ___________ statement?

p) What is the difference in the output of SELECT statement if we write the keyword ALL in place of DISTINCT?

q) What is the purpose of a Column Alias?

r) Write the purpose of the following SELECT statement?

```
SELECT  Itemcode,IName AS "Item Title"  FROM Item;
```

s) What does the Modulus arithmetic operator do?

t) List six relational operators used in SQL.

u) How are operators with equal priority evaluated in an expression?

v) Which statement is used to modify data in a table?

w) Which statement is used to remove a column from a table?

x) What is the purpose of "ORDER BY" clause?

y) What value is assigned to a Nullable field if its value is not assigned in the INSERT INTO statement?

Lab Exercise

1. **Consider the following table named "GYM" with details about Fitness products being sold in the store.**

Table Name : **GYM**

PrCode stores Codes of Products

PrName stores names of Products

(UnitPrice is in Rs.)

PrCode	PrName	UnitPrice	Manufacturer
P101	Cross Trainer	25000	Avon Fitness
P102	TreadMill	32000	AG Fitline
P103	Massage Chair	20000	Fit Express
P104	Vibration Trainer	22000	Avon Fitness
P105	Bike	13000	Fit Express

Write SQL statements to do the following:

a) Display the names of all the products in the store.

b) Display the names and unit price of all the products in the store

c) Display the names of all the products with unit price less than Rs.20000.00

d) Display details of all the products with unit price in the range 20000 to 30000

e) Display names of all products by the manufacturer "Fit Express"

f) Display all rows sorted in descending order of unit price.

g) Add a new row for product with the details: "P106","Vibro Exerciser", 23000, manufacturer : "Avon Fitness".

h) Change the Unit Price data of all the rows by applying a 10% discount reduction on all the products.

i) Display details of all products with manufacturer name starting with "A"

2. **Consider the following tables Employee and Department.**

Employee

ECode	LastName	FirstName	Department
101	Sharma	Amit	Sales
102	Arora	Shiv	Personnel
103	Lakshmi	KS	Accounts
104	Rajlani	Shivika	Accounts
105	Thakral	Satvik	Sales

Department

DepCode	DepName	Budget
101	Sales	200000
102	Personnel	150000
104	Accounts	300000

Write SQL statements to do the following:

a) Display the last names and first names of all employees.

b) Display the Department names of all employees, without duplicates.

c) Display all the details of employees with last name as "Lakshmi".

d) Display all the details of employees whose last name is ""Rajlani" or "Sharma".

e) Display the codes and first names of all employees of 'Accounts' department.

f) Display department names of departments with budget above 18000.

g) Display all the details of employees whose First name begins with "S".

h) Display department details(from Department table) in descending order of Budget amount.

i) Change the Department name "Sales" to "Marketing" everywhere in the table "Employee" and "Department"

j) Add a new row with appropriate data values in Department table.

k) Create the table Department with columns of appropriate data types.

10 Functions in MySQL

Learning Objectives

After studying this lesson the students will be able to

Distinguish between two types of functions.

State the syntax and working of most of the Numeric, String and date/Time functions.

Functions are a powerful feature of SQL. Using these functions, we can find sum of values stored in a column or convert all characters of a name to lowercase or round off salaries in a column to two decimal places and so on. MySQL supports many functions to manipulate data. We can broadly categorize functions into two types: Single Row functions and Multiple Row Functions.

$$\begin{tikzcd}[column sep=small] & \text{Functions} \arrow[dl, dash] \arrow[dr, dash] & \\ \text{Single Row functions} & & \text{Multiple Row Functions} \end{tikzcd}$$

Single-row functions: Single row functions operate on a single value to return a single value. They can accept one or more arguments but return only one result per row. When applied on a table, they return a single result for every row of the queried table. They are further categorized into:

- Numeric functions
- String functions
- Date and Time functions

Multiple Row Functions (also called Aggregate Functions): Multiple row functions operate on a set of rows to return a single value. Examples include SUM(), AVG() and COUNT().

(Note : Multiple Row functions will be discussed in detail in Class XII)

Let us consider the following table named Employee with 5 rows. We will be referring to it in our lesson to learn about Functions.

```
CREATE TABLE Employee(
        id           int,
        first_name   VARCHAR(15),
        last_name    VARCHAR(15),
        date_join    DATE,
        salary       DECIMAL(8,2),
       city         VARCHAR(10)
     );
```

The rows in Employee table are as follows:

```
mysql> SELECT * FROM Employee;
+----+------------+-----------+------------+----------+---------+
| id | first_name | last_name | date_join  | salary   | city    |
+----+------------+-----------+------------+----------+---------+
|  1 | Amit       | Sharma    | 1996-07-25 | 25000.00 | Delhi   |
|  2 | Deeksha    | Verma     | 1995-06-27 | 30000.00 | Pune    |
|  3 | Navkiran   | Ahluwalia | 1990-02-20 | 32000.50 | Delhi   |
|  4 | Mamta      | Sharma    | 1989-08-18 | 37500.50 | Mumbai  |
|  5 | Bhawna     | Ahlurkar  | 2010-03-01 | 42389.50 | Chennai |
+----+------------+-----------+------------+----------+---------+
5 rows in set (0.00 sec)
```

A) Numeric Functions:

MySQL numeric functions perform operations on numeric values and return numeric values. The following table tells us about the numeric functions of MySQL and what they do.

Sno	Name & Syntax	Description	Example

1	POWER(X,Y)
	or
	POW(X,Y)
		Returns the value of X raised to the power of Y.	`a)mysql> SELECT POW(2,4);`
			`Result: 16`
			`b)mysql> SELECT POW(2,-2);`
			`Result: 0.25`
			`c)mysql> SELECT POW(-2,3);`
			`Result:-8`
			`d)mysql> SELECT id,salary, POWER(salary,2) FROM employee;`
			`Result:`
			`+----+----------+-----------------+`
			`| id | salary   | power(salary,2) |`
			`+----+----------+-----------------+`
			`|  1 | 25000.00 |       625000000 |`
			`|  2 | 30000.00 |       900000000 |`
			`|  3 | 32000.50 |   1024032000.25 |`
			`|  4 | 37500.50 |   1406287500.25 |`
			`|  5 | 42389.50 |   1796869710.25 |`
			`+----+----------+-----------------+`
			`5 rows in set (0.00 sec)`

2	ROUND(X,D)
	ROUND(X)
		a) Rounds the argument X to D decimal places.
		b) If number of decimal places is not specified or is zero, the number rounds to the nearest integer OR (0 decimal places).
			`a) mysql> SELECT ROUND(-1.23);`
			`Result: -1`
			`b) mysql> SELECT ROUND(-1.58);`
			`Result: -2`
			`c) mysql> SELECT ROUND(1.43);`
			`Result: 1`
			`d) mysql> SELECT ROUND(6.298, 1);`
			`Result: 6.3`
			`e) mysql> SELECT ROUND(6.235, 0);`
			`Result: 6`
			`f) mysql> SELECT ROUND(56.235, -1);`
			`Result: 60`

		c) If negative value is specified for precision, it counts off that value left from the decimal point.
		
		d) If positive value is specified for precision, it counts off that value right from the decimal point.
			g) `mysql> SELECT id,ROUND(salary,0) FROM employee;`
			Result:
			`+------+-----------------+`
			`| id   | round(salary,0) |`
			`+------+-----------------+`
			`|    1 |           25000 |`
			`|    2 |           30000 |`
			`|    3 |           32001 |`
			`|    4 |           37501 |`
			`|    5 |           42390 |`
			`+------+-----------------+`
			`5 rows in set (0.00 sec)`

3	TRUNCATE (X,D)	Returns the number X, truncated to D decimal places. If D is 0, the result has no decimal point or fractional part.If D is negative, it causes D digits left of the decimal point of the value X to become zero.	a) `mysql> SELECT TRUNCATE(7.543,1);` Result: 7.5 b) `mysql> SELECT TRUNCATE(4.567,0);` Result: 4 c) `mysql> SELECT TRUNCATE(-7.45,1);` Result: -7.4 d) `mysql> SELECT TRUNCATE(346,-2);` Result: 300 e) `mysql> SELECT id,TRUNCATE(salary,0) FROM employee;`

		Note: TRUNCATE does not round a number. It simply chops off digits from a number.	`Result:`
			`+------+--------------------+`
			`| id   | truncate(salary,0) |`
			`+------+--------------------+`
			`|  1   |              25000 |`
			`|  2   |              30000 |`
			`|  3   |              32000 |`
			`|  4   |              37500 |`
			`|  5   |              42389 |`
			`+------+--------------------+`
			`5 rows in set (0.00 sec)`

B) String(Character) Functions

String functions operate on character type data. String functions are used to extract, change, format or alter character strings. They return either character or numeric values. The following table tells us about the Character functions of MySQL and what they do.

Sno	Name & Syntax	Description	Example

1	LENGTH(str)	Returns the length of a column or a string in bytes.	`a) mysql> SELECT LENGTH`
			`   ('Informatics');`
			`        Result: 11`
			`b) mysql> SELECT LENGTH(First_Name)`
			`   FROM Employee;`
			`Result:`
			`+--------------------+`
			`| LENGTH(First_Name) |`
			`+--------------------+`
			`|                  4 |`
			`|                  7 |`
			`|                  8 |`
			`|                  5 |`
			`|                  6 |`
			`+--------------------+`
			`5 rows in set (0.00 sec)`

2	CONCAT(str1, str2,...)	Returns the string that results from concatenating the arguments. May have one or more arguments.	`a) mysql> SELECT CONCAT ('My', 'S', 'QL');`
			`Result: 'MySQL'`
			`b) mysql> SELECT CONCAT('Class', NULL, 'XI');`
			`Result: NULL`
			`c) mysql> SELECT CONCAT(First_Name,'',Last_Name) FROM Employee;`
			`Result:`
			`+---------------------------------+`
			`| CONCAT(First_Name,' ',Last_Name) |`
			`+---------------------------------+`
			`| Amit Sharma                     |`
			`| Deeksha Verma                   |`
			`| Navkiran Ahluwalia              |`
			`| Mamta Sharma                    |`
			`| Bhawna Ahlurkar                 |`
			`+---------------------------------+`
			`5 rows in set (0.00 sec)`

3	INSTR (str,substr)	Returns the position of the first occurrence of substring substr in string str.	`a) mysql> SELECT INSTR ('Informatics', 'for');`
			`Result: 3`
			`b) mysql> SELECT INSTR ('Computers', 'pet');`
			`Result: 0`
			`c) mysql> SELECT INSTR (First_Name,'Kiran') FROM Employee;`
			`Result:`
			`+--------------------------+`
			`| INSTR(First_Name,'Kiran') |`
			`+--------------------------+`
			`|                        0 |`
			`|                        0 |`
			`|                        4 |`
			`|                        0 |`
			`|                        0 |`
			`+--------------------------+`
			`5 rows in set (0.00 sec)`

4	LOWER(str)
	or
	LCASE(str)
		Returns the argument (str) in lowercase i.e. it changes all the characters of the passed string to lowercase.	a) `mysql> SELECT LOWER ('INFORMATICS');`
			`Result: 'informatics'`
			b) `mysql> SELECT LOWER(Last_Name) FROM Employee;`
			`Result:`
			`+------------------+`
			`| LOWER(Last_Name) |`
			`+------------------+`
			`| sharma           |`
			`| verma            |`
			`| ahluwalia        |`
			`| sharma           |`
			`| ahlurkar         |`
			`+------------------+`
			`5 rows in set (0.00 sec)`

5	UPPER(str)
	or
	UCASE(str)
		Returns the argument in uppercase. i.e. it changes all the characters of the passed string to uppercase.	a) `mysql> SELECT UPPER ('Informatics');`
			`Result: 'INFORMATICS'`
			b) `mysql> SELECT UPPER(Last_Name) FROM Employee;`
			`Result:`
			`+------------------+`
			`| UPPER(Last_Name) |`
			`+------------------+`
			`| SHARMA           |`
			`| VERMA            |`
			`| AHLUWALIA        |`
			`| SHARMA           |`
			`| AHLURKAR         |`
			`+------------------+`
			`5 rows in set (0.00 sec)`

6	LEFT(str,n)	Returns the specified number of characters (n)from the left side of string str.	`a) mysql> SELECT`
			`   LEFT('Informatics', 3);`
			`Result: 'Inf'`
			
			`b) mysql>select`
			`   LEFT(first_name,3)FROM Employee;`
			`Result:`
			`+--------------------+`
			`| LEFT(first_name,3) |`
			`+--------------------+`
			`| Ami                |`
			`| Dee                |`
			`| Nav                |`
			`| Mam                |`
			`| Bha                |`
			`+--------------------+`
			`5 rows in set (0.00 sec)`

7	RIGHT(str,n)	Returns the specified number of characters (n)from the right side of string str.	`a) mysql> SELECT`
			`   RIGHT('Informatics', 4);`
			`   Result: 'tics'`
			`b) mysql> select`
			`   RIGHT(first_name,3) FROM`
			`   Employee;`
			`Result:`
			`+---------------------+`
			`| RIGHT(first_name,3) |`
			`+---------------------+`
			`| mit                 |`
			`| sha                 |`
			`| ran                 |`
			`| mta                 |`
			`| wna                 |`
			`+---------------------+`
			`5 rows in set (0.00 sec)`

8	LTRIM(str)	Removes leading spaces i.e. removes spaces from the left side of the string str.	`a) mysql> SELECT LTRIM (' Informatics');`
			`Result: 'Informatics'`
			`b) mysql> SELECT LTRIM(First_Name) FROM Employee;`
			`Result:`
			`+-------------------+`
			`| LTRIM(First_Name) |`
			`+-------------------+`
			`| Amit              |`
			`| Deeksha           |`
			`| Navkiran          |`
			`| Mamta             |`
			`| Bhawna            |`
			`+-------------------+`
			`5 rows in set (0.00 sec)`

9	RTRIM(str)	Removes trailing spaces i.e. removes spaces from the right side of the string str.	`a) mysql> SELECT RTRIM ('Informatics ');`
			`Result: 'Informatics'`
			`b) mysql> SELECT RTRIM(First_Name) FROM Employee;`
			`Result:`
			`+-------------------+`
			`| RTRIM(First_Name) |`
			`+-------------------+`
			`| Amit              |`
			`| Deeksha           |`
			`| Navkiran          |`
			`| Mamta             |`
			`| Bhawna            |`
			`+-------------------+`
			`5 rows in set (0.02 sec)`

10	TRIM(str)	Removes both leading and trailing spaces from the string str.	`a)mysql> SELECT TRIM('  Informatics  ');`
			`Result: 'Informatics'`
			`b) mysql> SELECT TRIM(First_Name) FROM Employee;`
			`Result:`
			`+------------------+`
			`| TRIM(First_Name) |`
			`+------------------+`
			`| Amit             |`
			`| Deeksha          |`
			`| Navkiran         |`
			`| Mamta            |`
			`| Bhawna           |`
			`+------------------+`
			`5 rows in set (0.00 sec)`

11	SUBSTRING (str,m,n)
	Or
	MID(str,m,n)
		Returns the specified number of characters from the middle of the string. There are 3 arguments. The first argument is the source string. The second argument is the position of first character to be displayed. The third argument is the number of characters to be displayed.	`a) mysql> SELECT SUBSTRING('Informatics',3);`
			`Result: 'formatics'`
			`b) mysql> SELECT SUBSTRING('Informatics' FROM 4);`
			`Result: 'ormatics'`
			`c) mysql> SELECT SUBSTRING('Informatics',3,4);`
			`Result: 'form'`
			`d) mysql> SELECT SUBSTRING('Computers', -3);`
			`Result: 'ers'`
			`e) mysql> SELECT SUBSTRING('Computers', -5, 3);`
			`Result: 'ute'`
			`f) mysql> SELECT SUBSTRING('Computers' FROM -4 FOR 2);`
			`Result: 'te'`

		If the third argument is missing, then starting from the position specified, the rest of the string is returned.
		
		It is also possible to use a negative value for the second argument ie. position (pos). In such a case, the beginning of the substring is pos characters from the end of the string,
		
		Note: SUBSTR is the same as SUBSTRING
			`g) mysql> SELECT MID('Informatics',`
			`   3,4);`
			`   Result: 'form'`
			`h) mysql> select`
			`   MID(first_name,3,2) FROM`
			`   Employee;`
			`Result:`
			`+---------------------+`
			`| MID(first_name,3,2) |`
			`+---------------------+`
			`| it                  |`
			`| ek                  |`
			`| vk                  |`
			`| mt                  |`
			`| aw                  |`
			`+---------------------+`
			`5 rows in set (0.00 sec)`

12	ASCII(str)	Returns the ASCII value of the leftmost character of the string str. Returns 0 if str is an empty string. Returns NULL if str is NULL.	`a) mysql> SELECT ASCII('2');`
			`   Result: 50`
			`   (ASCII value of character '2')`
			`b) mysql> SELECT ASCII('dx');`
			`   Result: 100`
			`   (ASCII value of d)`
			`c) mysql> SELECT ASCII('A');`
			`   Result: 65`
			`   (ASCII value of 'A')`

C) Date and Time Functions

Date and Time functions allow us to perform many types of tasks on Date type data.The default date format in MySQL is YYYY-MM-DD.

Sno	Name & Syntax	Description	Example
1	CURDATE()	Returns the current date in YYYY-MM-DD format or YYYYMMDD format, depending on whether the function is used in a string or numeric context.	a) `mysql> SELECT CURDATE();` Result: '2010-02-26'
2	NOW()	Returns the current date and time in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uuuuuu format, depending on whether the function is used in a string or numeric context.	a) `mysql> SELECT NOW();` Result: '2010-02-26 21:30:26'
3	SYSDATE()	Returns the current date and time in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uuuuuu format, depending on whether the function	a) `mysql> SELECT SYSDATE();` Result: '2010-02-26 21:30:26' b) `mysql> SELECT SYSDATE() + 0;` Result: 20100226213026.000000

		is used in a string or numeric context. Note : SYSDATE() returns the time at which the function executes. SYSDATE() differs from NOW() which returns a constant time that indicates the time at which the statement began to execute. *** For difference between SYSDATE() and NOW() refer to NOTE at the end of this table.**	

4	DATE(expr)	Extracts the date part of a date or datetime expression	a) **`mysql> SELECT DATE('2010-02-26 01:02:03');`**
			`Result: '2010-02-26'`
			b) **`mysql> SELECT DATE('2009-10-16 01:02:03')`**
			`Result: '2009-10-16'`

5	MONTH(date)	Returns the numeric month from the date passed, in the range 0 to 12. It returns 0 for dates such as '0000-00-00' or '2010-00-00' that have a zero month part.	a) **`mysql> SELECT MONTH('2010-02-26');`** `Result: 2` b) **`mysql> select id,date_join,month(date_join) from employee;`**

			Result:
			`+----+------------+------------------+`
			`| id | date_join  | month(date_join) |`
			`+----+------------+------------------+`
			`|  1 | 1996-07-25 |                7 |`
			`|  2 | 1995-06-27 |                6 |`
			`|  3 | 1990-02-20 |                2 |`
			`|  4 | 1989-08-18 |                8 |`
			`|  5 | 2010-03-01 |                3 |`
			`+----+------------+------------------+`
			`5 rows in set (0.00 sec)`

6	YEAR(date)	Returns the year for date passed in the range 0 to 9999. Returns values like 1998, 2010,1996 and so on.	a) `mysql> SELECT YEAR('2010-02-26');`
			Result: 2010
			b) `mysql> SELECT id,date_join,year (date_join) from employee;`
			Result:
			`+----+------------+-----------------+`
			`| id | date_join  | year(date_join) |`
			`+----+------------+-----------------+`
			`|  1 | 1996-07-25 |            1996 |`
			`|  2 | 1995-06-27 |            1995 |`
			`|  3 | 1990-02-20 |            1990 |`
			`|  4 | 1989-08-18 |            1989 |`
			`|  5 | 2010-03-01 |            2010 |`
			`+----+------------+-----------------+`
			`5 rows in set (0.00 sec)`

7	DAYNAME (date)	If you want to know which day you were born on. Was it a Monday or a Friday? Use DAYNAME function. It	a) `mysql> SELECT YEAR('2009-07-21');` Result: 'Tuesday' b) `mysql> Select id,date_join,dayname (date_join) from employee;`

		returns the name of the weekday for the date passed	Result:
			`+----+------------+--------------------+`
			`| id | date_join  | dayname(date_join) |`
			`+----+------------+--------------------+`
			`|  1 | 1996-07-25 | Thursday           |`
			`|  2 | 1995-06-27 | Tuesday            |`
			`|  3 | 1990-02-20 | Tuesday            |`
			`|  4 | 1989-08-18 | Friday             |`
			`|  5 | 2010-03-01 | Monday             |`
			`+----+------------+--------------------+`
			`5 rows in set (0.00 sec)`

8	DAYOFMONTH(date)	Returns the day of the month in the range 0 to 31.	a) `mysql> SELECT DAYOFMONTH('2009-07-21');`
			Result: 21
			b) `mysql> select id,date_join,dayofmonth(date_join) from employee;`
			Result:
			`+----+------------+-----------------------+`
			`| id | date_join  | dayofmonth(date_join) |`
			`+----+------------+-----------------------+`
			`|  1 | 1996-07-25 |                    25 |`
			`|  2 | 1995-06-27 |                    27 |`
			`|  3 | 1990-02-20 |                    20 |`
			`|  4 | 1989-08-18 |                    18 |`
			`|  5 | 2010-03-01 |                     1 |`
			`+----+------------+-----------------------+`
			`5 rows in set (0.00 sec)`

9	DAYOFWEEK (date)	Returns the day of week in number as 1 for Sunday, 2 for Monday and so on.	a) `mysql> SELECT DAYOFWEEK('2009-07-21');` Result: 3 b) `mysql> select id,date_join,dayofweek(date_join) from employee;`

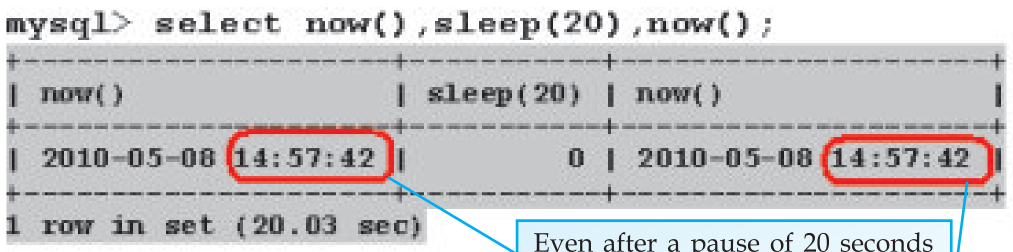
			Result:
			`+----+------------+---------------------+`
			`| id | date_join  | dayofweek(date_join) |`
			`+----+------------+---------------------+`
			`|  1 | 1996-07-25 |                   5 |`
			`|  2 | 1995-06-27 |                   3 |`
			`|  3 | 1990-02-20 |                   3 |`
			`|  4 | 1989-08-18 |                   6 |`
			`|  5 | 2010-03-01 |                   2 |`
			`+----+------------+---------------------+`
			`5 rows in set (0.00 sec)`

10	DAYOFYEAR (date)	Return the day of the year for the given date in numeric format in the range 1 to 366.	a) `mysql> SELECT DAYOFYEAR('2009-07-21');`
			Result: 202
			b) `mysql> SELECT DAYOFYEAR('2009-01-01');`
			Result: 1
			c) `mysql> select id,date_join,dayofyear (date_join) from employee;`
			Result:
			`+----+------------+---------------------+`
			`| id | date_join  | dayofyear(date_join) |`
			`+----+------------+---------------------+`
			`|  1 | 1996-07-25 |                 207 |`
			`|  2 | 1995-06-27 |                 178 |`
			`|  3 | 1990-02-20 |                  51 |`
			`|  4 | 1989-08-18 |                 230 |`
			`|  5 | 2010-03-01 |                  60 |`
			`+----+------------+---------------------+`
			`5 rows in set (0.00 sec)`

Note

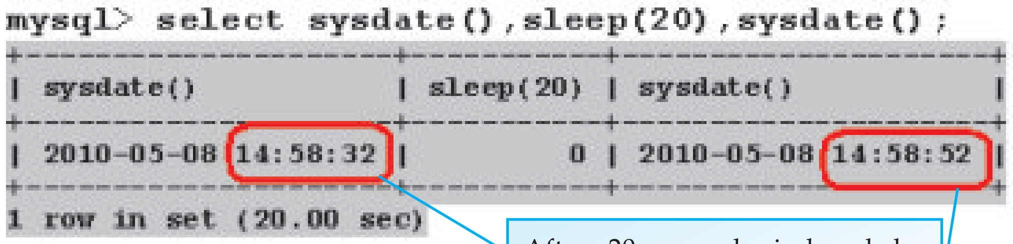
Difference between NOW() and SYSDATE()

(Sleep(argument) pauses for the number of seconds given by the argument.)



```
mysql> select now(),sleep(20),now();
+---------------------+-----------+---------------------+
| now()               | sleep(20) | now()               |
+---------------------+-----------+---------------------+
| 2010-05-08 14:57:42 |         0 | 2010-05-08 14:57:42 |
+---------------------+-----------+---------------------+
1 row in set (20.03 sec)
```

Even after a pause of 20 seconds induced by sleep(20),now() shows the same time ie. the time at which the statement began to execute.



```
mysql> select sysdate(),sleep(20),sysdate();
+---------------------+-----------+---------------------+
| sysdate()           | sleep(20) | sysdate()           |
+---------------------+-----------+---------------------+
| 2010-05-08 14:58:32 |         0 | 2010-05-08 14:58:52 |
+---------------------+-----------+---------------------+
1 row in set (20.00 sec)
```

After 20 seconds induced by sleep(20), sysdate() shows time incremented by 20 seconds.

Summary

1. Functions perform some operations and return a value.
2. Single row functions operate on a single value to return a single value.
3. Multiple Row functions operate on a set of rows to return a single value.
4. Numeric functions perform operations on numeric values and return numeric values.
5. String functions operate on character type data. They return either character or numeric values.
6. Date and Time functions allow us to manipulate Date type data.

Multiple Choice Questions

1) ________________functions operate on a single value to return a single value

   (a) Multiple Row

   (b) Aggregate

   (c) Single Row

   (d) Summation

2) SUM, AVG,COUNT are examples of ____________________functions.

   (a) Date

   (b) String

   (c) Multiple Row

   (d) Single Row

3) SELECT POW(-3,2) will display the output:

   (a) -6

   (b) -9

   (c) 9

   (d) 6

4) SELECT TRUNCATE(7.956,2) will result in

   (a) 7.95

   (b) 7.96

   (c) 8

   (d) 8.0

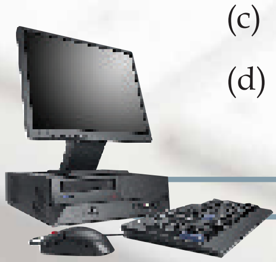
5) INSTR(str,str2) returns the position of the first occurrence of

   (a) Str in "MySQL"

   (b) Str in str2

   (c) str2 in str

   (d) str2 in "SQL"



6) **Any String function returns**

(a) Only string

(b) Only number

(c) String or number

(d) String, number or date type data.

Answer the following questions.

1. Define a Function.

2. List 3 categories of single row functions. Give two examples in each category.

3. How are numeric functions different from String functions?

4. Which function is used to display the system date?

5. Which Date function displays the result like "Monday" or "Tuesday" etc.

6. Name a

    i) date function that returns a number.

    ii) String function that returns a number.

    iii) date function that returns a date.

7. Write SQL statements to do the following:

    a) Using the three separate words "We," "study," and "MySQL," produce the following output:

    "We study MySQL"

    b) Use the string "Internet is a boon" and extract the string "net".

    c) Display the length of the string "Informatics Practices".

    d) Display the position of "My" in "Enjoying MySQL".

    e) Display the name of current month.

    f) Display the date 10 years from now. Label the column "Future."

    g) Display the day of week on which your birthday will fall or fell in 2010.

8. Write the output that the following statements will produce:

   a) `SELECT ROUND(7.3456, 2);`

   b) `SELECT TRUNCATE(2.3456, 2);`

   c) `SELECT DAYOFMONTH('2009-08-25');`

   d) `SELECT MONTH('2010-02-26');`

   e) `SELECT RIGHT('Informatics', 4);`

Lab Exercises

1. Create the following table named "Charity" and write SQL queries for the tasks that follow:

Table: Charity

P_Id	LastName	FirstName	Address	City	Contribution
1	Bindra	Jaspreet	5B, Gomti Nagar	Lucknow	3500.50
2	Rana	Monica	21 A, Bandra	Mumbai	2768.00
3	Singh	Jatinder	8, Punjabi Bagh	Delhi	2000.50
4	Arora	Satinder	K/1, Shere Punjab Colony	Mumbai	1900.00
5	Krishnan	Vineeta	A-75,Adarsh Nagar		

(Contribution is in Rs.)

I. Display all first names in lowercase

II. Display all last names of people of Mumbai city in uppercase

III. Display Person Id along with First 3 characters of his/her name.

IV. Display first name concatenated with last name for all the employees.

V. Display length of address along with Person Id

VI. Display last 2 characters of City and Person ID.

VII. Display Last Names and First names of people who have "at" in the second or third position in their first names.

VIII. Display the position of 'a' in Last name in every row.

IX. Display Last Name and First name of people who have "a" as the last character in their First names.

X. Display the first name and last name concatenated after removing the leading and trailing blanks.

XI. Display Person Id, last names and contribution rounded to the nearest rupee of all the persons.

XII. Display Person Id, last name and contribution with decimal digits truncated of all the persons.

XIII. Display Last name, contribution and a third column which has contribution divided by 10. Round it to two decimal points.

2. Consider the table "Grocer" and write SQL queries for the tasks that follow:

Table: Grocer

Item_Id	ItemName	UnitPrice	Quantity (kg)	Date_Purchase
1	Rice	52.50	80	2010-02-01
2	Wheat	25.40	50	2010-03-09
3	Corn	50.80	100	2010-03-11
4	Semolina	28.90	50	2010-01-15

(Unit Price is per kg price)

I. Display Item name, unit price along with Date of purchase for all the Items.

II. Display Item name along with Month (in number) when it was purchased for all the items.

III. Display Item name along with year in which it was purchased for all the items.

IV. Display Item Id, Date of Purchase and day name of week (e.g. Monday) on which it was purchased for all the items.

V. Display names of all the items that were purchased on Mondays or Tuesdays.

VI. Display the day name of the week on which Rice was purchased.

VII. Display the Item name and unit price truncated to integer value (no decimal digits)of all the items.

VIII. Display current date