Reffer Link : http://www.studytonight.com/dbms/

2nd Refer Link : http://www.1keydata.com/sql/sql.html

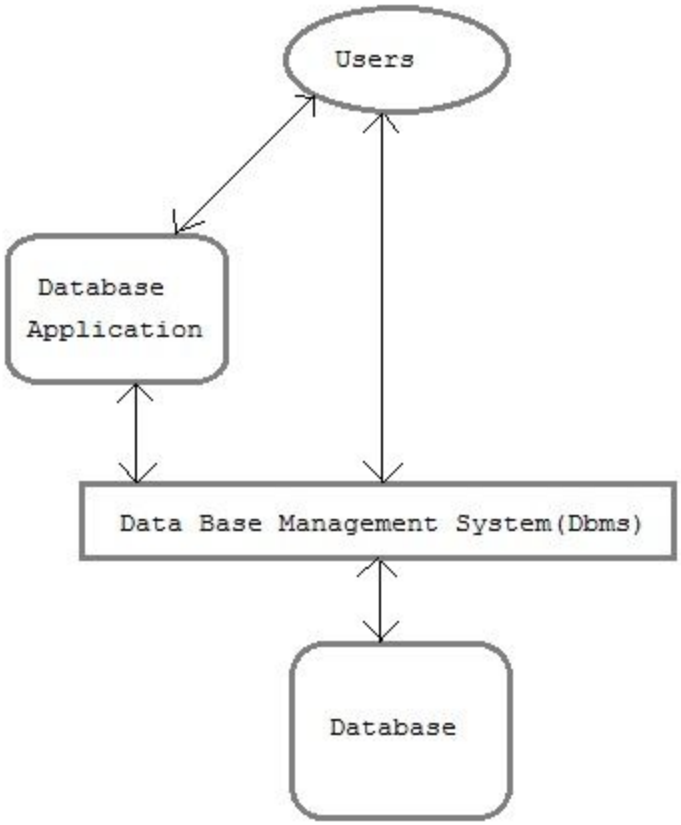
3rd Refer Link : http://www.w3schools.com/sql/default.asp

A **Database** is a collection of related data organised in a way that data can be easily accessed, managed and updated. Any piece of information can be a data, for example name of your school. Database is actually a place where related piece of information is stored and various operations can be performed on it.

A **DBMS** is a software that allows creation, definition and manipulation of database. Dbms is actually a tool used to perform any kind of operation on data in database. Dbms also provides protection and security to database. It maintains data consistency in case of multiple users. Here are some examples of popular dbms, MySql, Oracle, Sybase, Microsoft Access and IBM DB2 etc.

Components of Database System

The database system can be divided into four components.



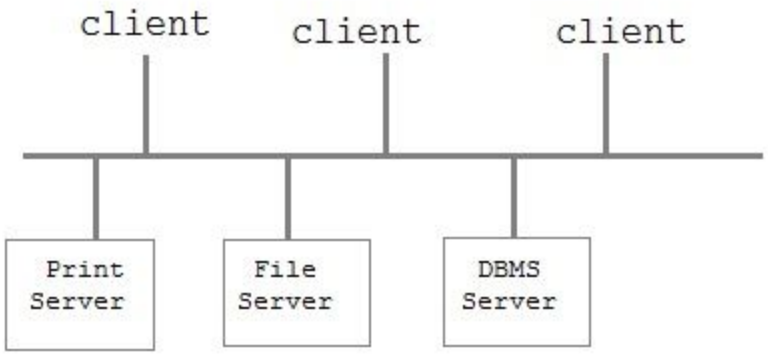
- **Users :** Users may be of various type such as DB administrator, System developer and End users.
- **Database application :** Database application may be Personal, Departmental, Enterprise and Internal
- **DBMS :** Software that allow users to define, create and manages database access, Ex: MySql, Oracle etc.
- **Database :** Collection of logical data.

Database Architecture

Database architecture is logically divided into two types.

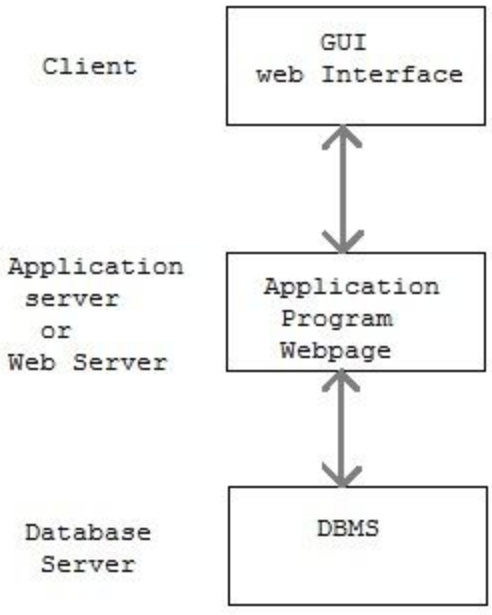
1. Logical two-tier Client / Server architecture
2. Logical three-tier Client / Server architecture

Two-tier Client / Server Architecture



Two-tier Client / Server architecture is used for User Interface program and Application Programs that runs on client side. An interface called ODBC(Open Database Connectivity) provides an API that allow client side program to call the dbms. Most DBMS vendors provide ODBC drivers. A client program may connect to several DBMS's. In this architecture some variation of client is also possible for example in some DBMS's more functionality is transferred to the client including data dictionary, optimization etc. Such clients are called **Data server**.

Three-tier Client / Server Architecture



Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called **Application server** or Web Server stores the web connectivity software and the business logic(constraints) part of application used to access the

right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client.

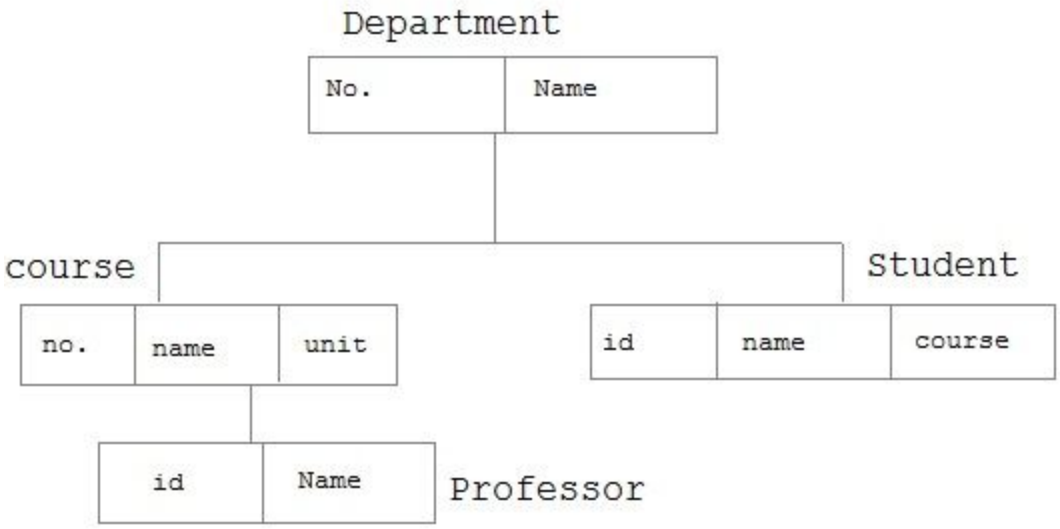
Database Model

A Database model defines the logical design of data. The model describes the relationships between different parts of the data. In history of database design, three models have been in use.

- Hierarchical Model
- Network Model
- Relational Model

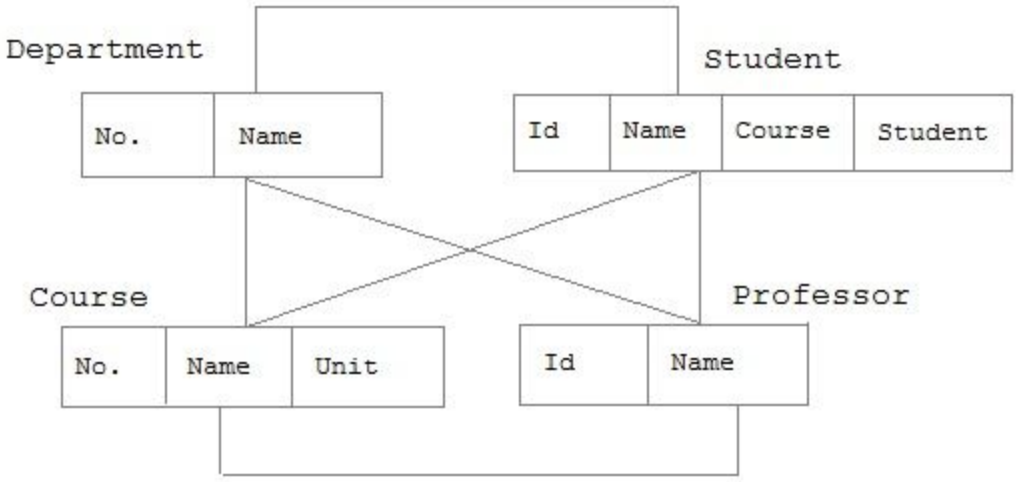
Hierarchical Model

In this model each entity has only one parent but can have several children . At the top of hierarchy there is only one entity which is called **Root**.



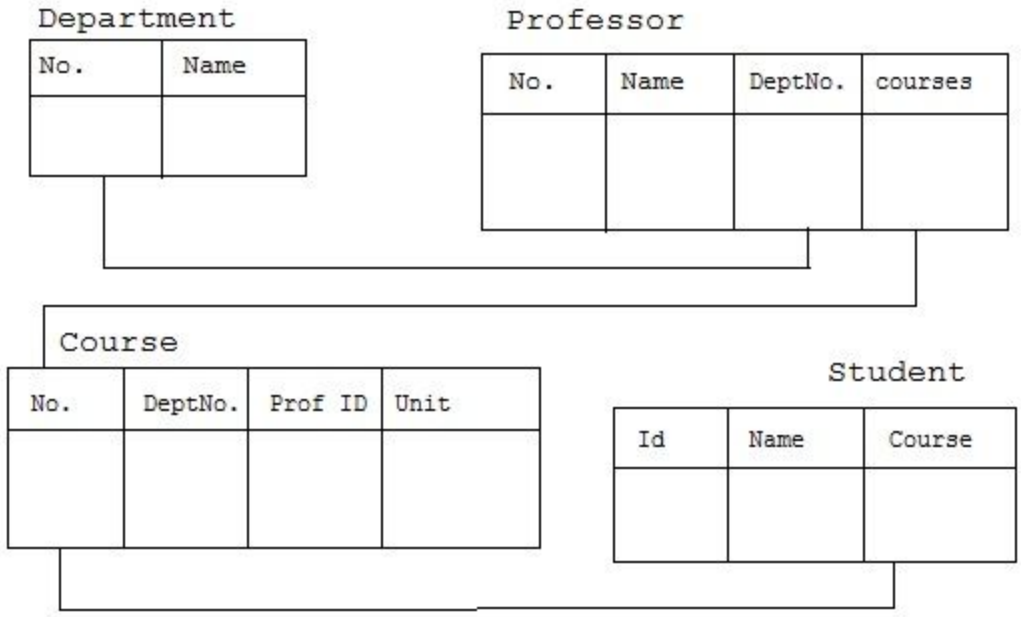
Network Model

In the network model, entities are organised in a graph,in which some entities can be accessed through sveral path



Relational Model

In this model, data is organised in two-dimesional tables called **relations**. The tables or relation are related to each other.



RDBMS Concepts

A **Relational Database management System**(RDBMS) is a database management system based on relational model introduced by E.F Codd. In relational model, data is represented in terms of tuples(rows).
RDBMS is used to manage Relational database. **Relational database** is a collection of organized set of tables from which data can be accessed easily. Relational Database is most commonly used database. It consists of number of tables and each table has its own primary key.

What is Table ?

In Relational database, a **table** is a collection of data elements organised in terms of rows and columns. A table is also considered as convenient representation of **relations**. But a table can have duplicate tuples while a true **relation** cannot have duplicate tuples. Table is the most simplest form of data storage. Below is an example of Employee table.

ID	Name	Age	Salary
1	Adam	34	13000
2	Alex	28	15000
3	Stuart	20	18000
4	Ross	42	19020

What is a Record ?

A single entry in a table is called a **Record** or **Row**. A **Record** in a table represents set of related data. For example, the above **Employee** table has 4 records. Following is an example of single record.

1	Adam	34	13000

What is Field ?

A table consists of several records(row), each record can be broken into several smaller entities known as **Fields**. The above **Employee** table consist of four fields, **ID**, **Name**, **Age** and **Salary**.

What is a Column ?

In **Relational** table, a column is a set of value of a particular type. The term **Attribute** is also used to represent a column. For example, in Employee table, Name is a column that represent names of employee.

Name
Adam

Alex
Stuart
Ross

Database Keys

Keys are very important part of Relational database. They are used to establish and identify relation between tables. They also ensure that each record within a table can be uniquely identified by combination of one or more fields within a table.

Super Key

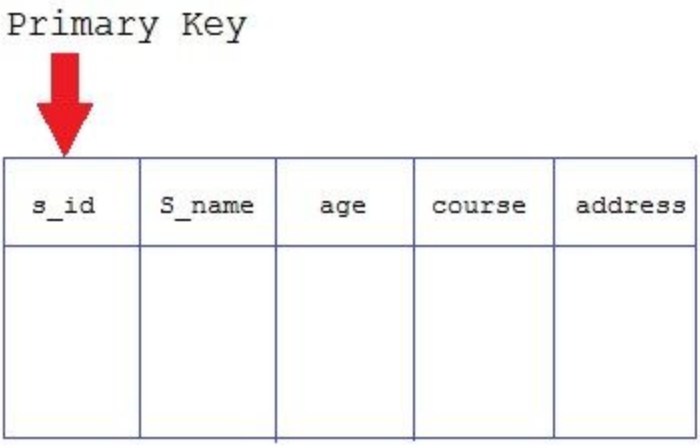
Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

Candidate Key

Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attribute that can act as a primary key for a table to uniquely identify each record in that table.

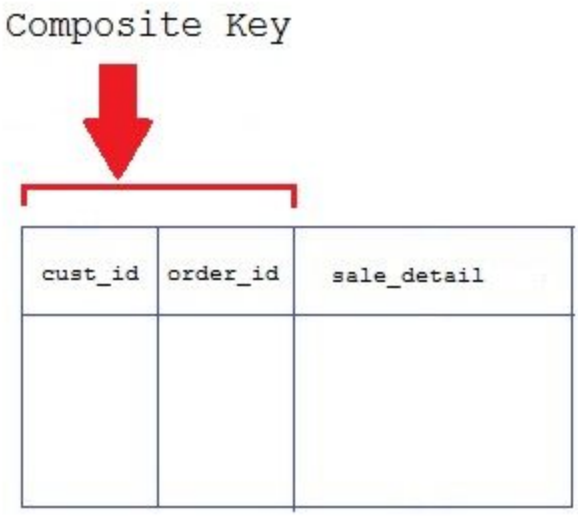
Primary Key

Primary key is a candidate key that is most appropriate to become main key of the table. It is a key that uniquely identify each record in a table.



Composite Key

Key that consist of two or more attributes that uniquely identify an entity occurance is called **Composite key**. But any attribute that makes up the **Composite key** is not a simple key in its own.



Secondary or Alternative key

The candidate key which are not selected for primary key are known as secondary keys or alternative keys

---

Non-key Attribute

Non-key attributes are attributes other than **candidate key** attributes in a table.

Non-prime Attribute

Non-prime Attributes are attributes other than **Primary attribute**.

Basic SQL :

Introduction to SQL

Structure Query Language(SQL) is a programming language used for storing and managing data in RDBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) uses **SQL** as the standard database language. SQL is used to perform all type of data operations in RDBMS.

SQL Command

SQL defines following data languages to manipulate data of RDBMS.

DDL : Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Command	Description
create	to create new table or database
alter	for alteration
truncate	delete data from table
drop	to drop a table
rename	to rename a table

DML : Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Command	Description
insert	to insert a new row
update	to update existing row
delete	to delete a row
merge	merging two rows or two tables

TCL : Transaction Control Language

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.

Command	Description
commit	to permanently save
rollback	to undo change
savepoint	to save temporarily

DCL : Data Control Language

Data control language provides command to grant and take back authority.

Command	Description
grant	grant permission of right
revoke	take back permission.

DQL : Data Query Language

Command	Description
select	retrieve records from one or more table

Basic SQL :

```
[root@rdbms ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

To Check available databases:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

To create Database :

```
mysql> create database StudentList;
Query OK, 1 row affected (0.00 sec)
```

Check Databases:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| StudentList        |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

To Use and table under the database OR To change the databases:

```
mysql> use StudentList;
Database changed
```

To check available tables in Databases :

```
mysql> show tables;
```

Empty set (0.00 sec)

Create a table student=>

```
mysql> create table student(StID int(5), StName char(20), StClass char(20));
Query OK, 0 rows affected (0.02 sec)
```

To see the description of the table

```
mysql> desc student;
+---------+----------+------+-----+---------+-------+
| Field   | Type     | Null | Key | Default | Extra |
+---------+----------+------+-----+---------+-------+
| StID    | int(5)   | YES  |     | NULL    |       |
| StName  | char(20) | YES  |     | NULL    |       |
| StClass | char(20) | YES  |     | NULL    |       |
+---------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

To Insert Data into Table :

```
mysql> insert into student (StID, StName, StClass) VALUES ('1001', 'Yugandhar',
'TengthClass');
Query OK, 1 row affected (0.00 sec)
```

To See all columns & rows from the table student:

```
mysql> select * from student;
+------+-----------+-------------+
| StID | StName    | StClass     |
+------+-----------+-------------+
| 1001 | Yugandhar | TengthClass |
+------+-----------+-------------+
1 row in set (0.00 sec)
```

Inserting more data into table :

```
mysql> insert into student Values('1002', 'Bhagya', 'XI-Class');
Query OK, 1 row affected (0.00 sec)

mysql> select *  from student;
+------+-----------+-------------+
| StID | StName          | StClass         |
+------+-----------+-------------+
| 1001 | Yugandhar | TengthClass |
| 1002 | Bhagya          | XI-Class        |
+------+-----------+-------------+
2 rows in set (0.00 sec)

mysql> insert into student (StId, StName) VALUES ('1003', 'Thriveni');
Query OK, 1 row affected (0.00 sec)

mysql> select * from student;
+------+-----------+-------------+
| StID | StName          | StClass         |
+------+-----------+-------------+
| 1001 | Yugandhar | TengthClass |
| 1002 | Bhagya          | XI-Class        |
| 1003 | Thriveni  | NULL        |
+------+-----------+-------------+
3 rows in set (0.00 sec)
```

Add new column into table:

```
mysql> alter table student add(FathersName char(20));
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> desc student;
+-------------+----------+------+-----+---------+-------+
| Field             | Type  | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| StID              | int(5)   | YES  |         | NULL |          |
| StName        | char(20) | YES  |       | NULL |          |
| StClass         | char(20) | YES  |       | NULL |          |
| FathersName | char(20) | YES  |      | NULL |          |
+-------------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> desc student;
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| StID        | int(5)   | YES  |     | NULL    |       |
| StName      | char(20) | YES  |     | NULL    |       |
| StClass     | char(20) | YES  |     | NULL    |       |
| FathersName | char(20) | YES  |     | NULL    |       |
| SchoolName  | char(20) | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> select * from student;
+------+-----------+-------------+-------------+------------+
| StID | StName    | StClass     | FathersName | SchoolName |
+------+-----------+-------------+-------------+------------+
| 1001 | Yugandhar | TengthClass | NULL        | NULL       |
| 1002 | Bhagya    | XI-Class    | NULL        | NULL       |
| 1003 | Thriveni  | NULL        | NULL        | NULL       |
+------+-----------+-------------+-------------+------------+
3 rows in set (0.00 sec)
```

```
mysql> select * from student;
+------+-----------+-------------+---------------+------------+
| StID | StName    | StClass     | FathersName   | SchoolName |
+------+-----------+-------------+---------------+------------+
| 1001 | Yugandhar | TengthClass | NULL          | NULL       |
| 1002 | Bhagya    | XI-Class    | NULL          | NULL       |
| 1003 | Thriveni  | NULL        | NULL          | NULL       |
| NULL | NULL      | NULL        | SreeRami Reddy | NULL      |
+------+-----------+-------------+---------------+------------+
4 rows in set (0.00 sec)
```

Delete last row from the above table:

```
mysql> delete from student where FathersName='SreeRami Reddy';
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from student;
+------+-----------+-------------+-------------+------------+
| StID | StName    | StClass     | FathersName | SchoolName |
+------+-----------+-------------+-------------+------------+
| 1001 | Yugandhar | TengthClass | NULL        | NULL       |
| 1002 | Bhagya    | XI-Class    | NULL        | NULL       |
| 1003 | Thriveni  | NULL        | NULL        | NULL       |
+------+-----------+-------------+-------------+------------+
3 rows in set (0.00 sec)
```

UPDATE record in Column of a table;

```
mysql> update student set StClass='XIIClass' where StName='Thriveni';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+------+-----------+-------------+-------------+------------+
| StID | StName    | StClass     | FathersName | SchoolName |
+------+-----------+-------------+-------------+------------+
| 1001 | Yugandhar | TengthClass | NULL        | NULL       |
| 1002 | Bhagya    | XI-Class    | NULL        | NULL       |
| 1003 | Thriveni  | XIIClass    | NULL        | NULL       |
+------+-----------+-------------+-------------+------------+
3 rows in set (0.00 sec)
```

Updating Father’sName in to table:

```
mysql> update student set FathersName='SreeRami Reddy' where StID='1001';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update student set FathersName='Sreenivas Reddy' where StID='1002';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update student set FathersName='Thimmanna' where StID='1003';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> select * from student;
+------+-----------+-------------+-----------------+------------+
| StID | StName    | StClass     | FathersName     | SchoolName |
+------+-----------+-------------+-----------------+------------+
| 1001 | Yugandhar | TengthClass | SreeRami Reddy  | NULL       |
| 1002 | Bhagya    | XI-Class    | Sreenivas Reddy | NULL       |
| 1003 | Thriveni  | XIIClass    | Thimmanna       | NULL       |
+------+-----------+-------------+-----------------+------------+
3 rows in set (0.00 sec)
```

To Rename a column name in Mysql:

```
mysql> select * from markslist;
+--------------+-----------------+----------+------------+------------+
| HallTicketNo | StudentName     | Class | Percentage | SchoolName |
+--------------+-----------------+----------+------------+------------+
|         1001 | Yugandhar Yetham | XClass   |         75 | NULL       |
|         1002 | Sudhakar Palle   | XIClass  |         70 | NULL       |
|         1002 | Mohan Palle     | XIIClass |      85 | NULL       |
+--------------+-----------------+----------+------------+------------+
3 rows in set (0.00 sec)

mysql> alter table markslist change SchoolName CollegeName varchar (20);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select *  from markslist;
+--------------+-----------------+----------+------------+-------------+
| HallTicketNo | StudentName     | Class | Percentage | CollegeName |
+--------------+-----------------+----------+------------+-------------+
|         1001 | Yugandhar Yetham | XClass   |         75 | NULL       |
|         1002 | Sudhakar Palle   | XIClass  |         70 | NULL       |
|         1002 | Mohan Palle     | XIIClass |      85 | NULL       |
+--------------+-----------------+----------+------------+-------------+
3 rows in set (0.00 sec)
```

Update Columns in Mysql :

```
mysql> update markslist set CollegeName='ZPHSchool' WHERE StudentName='Yugandhar
Yetham';
```

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update markslist set CollegeName='JrCollege ATP' WHERE StudentName='Sudhakar
Palle';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update markslist set CollegeName='JrCollege ATP' WHERE StudentName='Mohan
Palle';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from markslist;
+-------------+-----------------+----------+------------+---------------+
| HallTicketNo | StudentName           | Class | Percentage | CollegeName   |
+-------------+-----------------+----------+------------+---------------+
|          1001 | Yugandhar Yetham | XClass   |          75 | ZPHSchool          |
|          1002 | Sudhakar Palle   | XIClass  |          70 | JrCollege ATP |
|          1002 | Mohan Palle     | XIIClass |      85 | JrCollege ATP |
+-------------+-----------------+----------+------------+---------------+
3 rows in set (0.00 sec)
```

To Drop a column :

```
mysql> alter table markslist drop CollegeName;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select *  from markslist;
+-------------+-----------------+----------+------------+
| HallTicketNo | StudentName           | Class | Percentage |
+-------------+-----------------+----------+------------+
|          1001 | Yugandhar Yetham | XClass   |          75 |
|          1002 | Sudhakar Palle   | XIClass  |          70 |
|          1002 | Mohan Palle     | XIIClass |      85 |
+-------------+-----------------+----------+------------+
3 rows in set (0.00 sec)
```

To Modify a datatype for column of table.

```
mysql> alter table markslist modify Class int(20);
Query OK, 3 rows affected, 3 warnings (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> desc markslist;
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| HallTicketNo | int(20)  | YES  |     | NULL    |       |
| StudentName  | char(20) | YES  |     | NULL    |       |
| Class        | int(20)  | YES  |     | NULL    |       |
| Percentage   | int(20)  | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Rename a Table :

```
mysql> show tables;
+----------------------+
| Tables_in_StudentList |
+----------------------+
| markslist            |
| student              |
+----------------------+
2 rows in set (0.00 sec)

mysql> rename table markslist to MarksList;
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+----------------------+
| Tables_in_StudentList |
+----------------------+
| MarksList            |
| student              |
+----------------------+
2 rows in set (0.00 sec)
```

truncate will remove all records from the table:

```
mysql> select * from MarksList;
+-------------+-----------------+-------+------------+
| HallTicketNo | StudentName          | Class | Percentage |
+-------------+-----------------+-------+------------+
|          1001 | Yugandhar Yetham |   0 |       75 |
|          1002 | Sudhakar Palle   |       0 |       70 |
|          1002 | Mohan Palle     |         0 |       85 |
+-------------+-----------------+-------+------------+
3 rows in set (0.00 sec)

mysql> truncate table MarksList;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from MarksList;
Empty set (0.00 sec)
```

Drop a table from Database list:

```
mysql> drop table MarksList;
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+----------------------+
| Tables_in_StudentList |
+----------------------+
| student                    |
+----------------------+
1 row in set (0.00 sec)
```

To Drop a database

```
mysql> show databases;
+--------------------+
| Database                 |
+--------------------+
| information_schema |
| StudentList              |
| mysql            |
| test             |
+--------------------+
4 rows in set (0.01 sec)
```

```
mysql> drop database StudentList;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

Using WHERE Clause:

```
mysql> select * from student;
+-------+-----------+------+---------+
| st_id | name  | age  | address |
+-------+-----------+------+---------+
|   101 | Yugandhar | NULL | NULL    |
|   102 | Sudhakar  | NULL | NULL    |
|   103 | Mohan     | NULL | NULL    |
|   104 | priya     | NULL | NULL    |
|   105 | bhagya    | NULL | NULL    |
+-------+-----------+------+---------+
5 rows in set (0.00 sec)

mysql> update student set age='20', address='bangalore' where name='Yugandhar';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
|   102 | Sudhakar  | NULL | NULL      |
|   103 | Mohan     | NULL | NULL      |
|   104 | priya     | NULL | NULL      |
|   105 | bhagya    | NULL | NULL      |
+-------+-----------+------+-----------+
5 rows in set (0.00 sec)
```

SELECT Query

Select query is used to retrieve data from a tables. It is the most used SQL query. We can retrieve complete tables, or partial by mentioning conditions using WHERE clause.

```
mysql> select name from student where age=20;
+-----------+
| name            |
+-----------+
| Yugandhar |
| priya  |
+-----------+
2 rows in set (0.00 sec)

mysql> select * from student;
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
|   102 | Sudhakar  |   22 | Anantapur |
|   103 | Mohan           |   23 | Anantapur |
|   104 | priya   |   20 | bangalore |
|   105 | bhagya          |   27 | calcutta  |
+-------+-----------+------+-----------+
5 rows in set (0.00 sec)

mysql> select name, address from student where age=20;
+-----------+-----------+
| name            | address   |
+-----------+-----------+
| Yugandhar | ba
ngalore |
| priya  | bangalore |
+-----------+-----------+
2 rows in set (0.00 sec)


mysql> select * from student where name='Yugandhar';
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
```

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

Like clause

Like clause is used as condition in SQL query. **Like** clause compares data with an expression using wildcard operators. It is used to find similar data from the table.

Wildcard operators

There are two wildcard operators that are used in like clause.

- **Percent sign %** : represents zero, one or more than one character.
- **Underscore sign _** : represents only one character.

```
mysql> select * from student;
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
|   102 | Sudhakar  |   22 | Anantapur |
|   103 | Mohan     |   23 | Anantapur |
|   104 | priya   |   20 | bangalore |
|   105 | bhagya    |   27 | calcutta  |
+-------+-----------+------+-----------+
5 rows in set (0.00 sec)

mysql> select * from student where address like 'bang%';
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
|   104 | priya   |   20 | bangalore |
+-------+-----------+------+-----------+
2 rows in set (0.00 sec)



mysql> select * from student where address like '_a%';
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
```

```
|   101 | Yugandhar |   20 | bangalore |
|   104 | priya   |   20 | bangalore |
|   105 | bhagya          |   27 | calcutta  |
+-------+-----------+------+-----------+
3 rows in set (0.00 sec)

mysql> select * from student where name like '%a';
+-------+--------+------+-----------+
| st_id | name   | age  | address   |
+-------+--------+------+-----------+
|   104 | priya  |   20 | bangalore |
|   105 | bhagya |   27 | calcutta  |
+-------+--------+------+-----------+
2 rows in set (0.00 sec)
```

Order By Clause

Order by clause is used with **Select** statement for arranging retrieved data in sorted order. The **Order by** clause by default sort data in ascending order. To sort data in descending order **DESC** keyword is used with **Order by** clause.

```
mysql> select * from student;
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
|   102 | Sudhakar  |   22 | Anantapur |
|   103 | Mohan           |   23 | Anantapur |
|   104 | priya   |   20 | bangalore |
|   105 | bhagya          |   27 | calcutta  |
+-------+-----------+------+-----------+
5 rows in set (0.00 sec)

mysql> select * from student order by name;
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   105 | bhagya          |   27 | calcutta  |
|   103 | Mohan           |   23 | Anantapur |
|   104 | priya   |   20 | bangalore |
|   102 | Sudhakar  |   22 | Anantapur |
```

```
|   101 | Yugandhar |   20 | bangalore |
+-------+-----------+------+-----------+
5 rows in set (0.00 sec)

mysql> select name, st_id, age from student order by age;
+-----------+-------+------+
| name           | st_id | age  |
+-----------+-------+------+
| Yugandhar |   101 |   20 |
| priya  |   104 |   20 |
| Sudhakar  |   102 |   22 |
| Mohan          |   103 |   23 |
| bhagya         |   105 |   27 |
+-----------+-------+------+
5 rows in set (0.00 sec)
```

In DESC order …..=>

```
mysql> select st_id, name, age from student order by age desc;
+-------+-----------+------+
| st_id | name  | age  |
+-------+-----------+------+
|   105 | bhagya          |   27 |
|   103 | Mohan           |   23 |
|   102 | Sudhakar  |   22 |
|   101 | Yugandhar |   20 |
|   104 | priya   |   20 |
+-------+-----------+------+
5 rows in set (0.00 sec)
```

Group By Clause

Group by clause is used to group the results of a SELECT query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

```
mysql> select * from student;
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
|   102 | Sudhakar  |   22 | Anantapur |
```

```
|   103 | Mohan           |   23 | Anantapur |
|   104 | priya   |   20 | bangalore |
|   105 | bhagya          |   27 | calcutta  |
+-------+-----------+------+-----------+
5 rows in set (0.00 sec)


mysql> select name, age, address from student where age > 20 group by address;
+----------+------+-----------+
| name           | age  | address   |
+----------+------+-----------+
| Sudhakar |   22 | Anantapur |
| bhagya   |   27 | calcutta  |
+----------+------+-----------+
2 rows in set (0.00 sec)
```

HAVING Clause

having clause is used with SQL Queries to give more precise condition for a statement. It is used to mention condition in Group based SQL functions, just like WHERE clause.

```
mysql> show tables;
+------------------+
| Tables_in_Testing |
+------------------+
| emp_bonus              |
| student         |
+------------------+
2 rows in set (0.00 sec)

mysql> select * from emp_bonus;
+----------+-------+
| Employee | Bonus |
+----------+-------+
| A        |  1000 |
| B        |  2000 |
| A        |   500 |
| C        |   700 |
| B        |  1250 |
+----------+-------+
5 rows in set (0.00 sec)
```

```
mysql> select employee, sum(bonus) from emp_bonus group by employee having sum(bonus)
> 1000 ;
+----------+------------+
| employee | sum(bonus) |
+----------+------------+
| A        |       1500 |
| B        |       3250 |
+----------+------------+
2 rows in set (0.00 sec)
```

Distinct keyword

The **distinct** keyword is used with **Select** statement to retrieve unique values from the table. **Distinct** removes all the duplicate records while retrieving from database.

```
mysql> select distinct age from student;
+------+
| age  |
+------+
|   20 |
|   22 |
|   23 |
|   27 |
+------+
4 rows in set (0.00 sec)

mysql> select distinct address from student;
+-----------+
| address   |
+-----------+
| bangalore |
| Anantapur |
| calcutta  |
+-----------+
3 rows in set (0.00 sec)
```

AND & OR operator

AND and **OR** operators are used with **Where** clause to make more precise conditions for fetching data from database by combining more than one condition together.

AND operator

AND operator is used to set multiple conditions with *Where* clause.

```
mysql> select * from student;
+-------+-----------+------+-----------+
| st_id | name  | age  | address   |
+-------+-----------+------+-----------+
|   101 | Yugandhar |   20 | bangalore |
|   102 | Sudhakar  |   22 | Anantapur |
|   103 | Mohan     |   23 | Anantapur |
|   104 | priya   |   20 | bangalore |
|   105 | bhagya    |   27 | calcutta  |
+-------+-----------+------+-----------+
5 rows in set (0.00 sec)

mysql> select * from student where age > 20 AND address='Anantapur';
+-------+----------+------+-----------+
| st_id | name  | age  | address   |
+-------+----------+------+-----------+
|   102 | Sudhakar |   22 | Anantapur |
|   103 | Mohan    |   23 | Anantapur |
+-------+----------+------+-----------+
2 rows in set (0.00 sec)
```

OR operator

OR operator is also used to combine multiple conditions with *Where* clause. The only difference between AND and OR is their behaviour. When we use AND to combine two or more than two conditions, records satisfying all the condition will be in the result. But in case of OR, atleast one condition from the conditions specified must be satisfied by any record to be in the result.

```
mysql> select * from student where age > 20 OR address='Anantapur';
+-------+----------+------+-----------+
| st_id | name  | age  | address   |
+-------+----------+------+-----------+
```

```
|   102 | Sudhakar |   22 | Anantapur |
|   103 | Mohan    |   23 | Anantapur |
|   105 | bhagya   |   27 | calcutta  |
+-------+----------+------+-----------+
3 rows in set (0.00 sec)
```

Advanced SQL :

SQL Constraints

SQl Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.
Constraints can be divided into following two types,

- **Column level constraints :** limits only column data
- **Table level constraints :** limits whole table data

Constraints are used to make sure that the integrity of data is maintained in the database.
Following are the most used constraints that can be applied to a table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NOT NULL Constraint

NOT NULL constraint restricts a column from having a NULL value. Once **NOT NULL** constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value. One important point to note about NOT NULL constraint is that it cannot be defined at table level.

```
mysql> create table markslist (st_id int NOT NULL, Name varchar(60), Age int(20));
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+------------------+
| Tables_in_Testing |
```

```
+------------------+
| emp_bonus        |
| markslist        |
| student          |
+------------------+
3 rows in set (0.00 sec)

mysql> desc markslist;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| st_id | int(11)     | NO   |     | NULL    |       |
| Name  | varchar(60) | YES  |     | NULL    |       |
| Age   | int(20)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Updating a column with NOT NULL Constraint=>

```
mysql> alter table markslist change Name Name varchar(60) NOT NULL;
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> desc markslist;
+------------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| st_id  | int(11)     | NO   |     | NULL    |       |
| Name       | varchar(60) | NO   |   | NULL    |       |
| Age        | int(20)     | YES  |     | NULL    |       |
| TotalMarks | int(20)  | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> alter table markslist change Age Age int(20)NOT NULL;
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> desc markslist;
+------------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| st_id  | int(11)     | NO   |     | NULL    |       |
```

```
| Name       | varchar(60) | NO   |   | NULL |         |
| Age        | int(20)     | NO   |   | NULL |         |
| TotalMarks | int(20)     | YES  |   | NULL |         |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

UNIQUE Constraint

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data. UNIQUE constraint can be applied at column level or table level.

```
mysql> alter table markslist add (Percentage int UNIQUE);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> desc markslist;
+------------+-------------+------+-----+---------+-------+
| Field  | Type            | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| st_id  | int(11)         | NO   |     | NULL |         |
| Name       | varchar(60) | NO   |   | NULL |         |
| Age        | int(20)     | NO   |   | NULL |         |
| TotalMarks | int(20)     | YES  |   | NULL |         |
| Percentage | int(11)     | YES  | UNI | NULL    |         |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)


mysql> alter table markslist change st_id st_id int UNIQUE;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> desc markslist;
+------------+-------------+------+-----+---------+-------+
| Field  | Type            | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| st_id  | int(11)         | YES  | UNI | NULL    |         |
| Name       | varchar(60) | NO   |   | NULL |         |
| Age        | int(20)     | NO   |   | NULL |         |
| TotalMarks | int(20)     | YES  |   | NULL |         |
```

```
| Percentage | int(11)  | YES  | UNI | NULL    |          |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Primary Key Constraint

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

```
mysql> alter table markslist add primary key(TotalMarks);

mysql> desc markslist;
+------------+-------------+------+-----+---------+-------+
| Field  | Type             | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| st_id  | int(11)          | NO   | UNI | NULL    |          |
| Name           | varchar(60) | NO   | PRI |     |          |
| Age            | int(20)          | NO   |          | NULL |          |
| TotalMarks | int(20)  | YES  |          | NULL |          |
| Percentage | int(11)  | YES  | UNI | NULL    |          |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Foreign Key Constraint

FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand FOREIGN KEY, let's see it using two table.

```
mysql> create table Customer_Detail (c_id int, Customer_Name varchar(60), address varchar(60));
Query OK, 0 rows affected (0.02 sec)

mysql> show tables;
+------------------+
| Tables_in_Testing |
+------------------+
| Customer_Detail   |
| emp_bonus            |
| markslist       |
```

```
| student          |
+------------------+
4 rows in set (0.00 sec)

mysql> desc Customer_Detail;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| c_id          | int(11)     | YES  |     | NULL    |       |
| Customer_Name | varchar(60) | YES  |     | NULL    |       |
| address       | varchar(60) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> insert into Customer_Detail (c_id, Customer_Name, address) values ('101', 'Yugandhar',
'Noida');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Customer_Detail (c_id, Customer_Name, address) values ('102', 'Sudhakar',
'Delhi');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Customer_Detail (c_id, Customer_Name, address) values ('103', 'Mohan',
'ATP');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Customer_Detail;
+------+---------------+---------+
| c_id | Customer_Name | address |
+------+---------------+---------+
|  101 | Yugandhar     | Noida   |
|  102 | Sudhakar      | Delhi   |
|  103 | Mohan         | ATP     |
+------+---------------+---------+
3 rows in set (0.00 sec)
```

CHECK Constraint

CHECK constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

```
mysql> create table Order_Detail (Order_id int, Order_Name varchar(60), C_id int);
Query OK, 0 rows affected (0.00 sec)

mysql> desc Order_Detail;
+------------+-------------+------+-----+---------+-------+
| Field  | Type              | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Order_id   | int(11)    | YES  |          | NULL |          |
| Order_Name | varchar(60) | YES  |  | NULL |          |
| C_id             | int(11)          | YES  |          | NULL |          |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> alter table Order_Detail add CHECK(C_id > 100);
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> create table Order_Detail(Order_id int, Order_Name varchar(60), C_id int
CHECK(C_id>100));
Query OK, 0 rows affected (0.00 sec)

mysql> desc Order_Detail;
+------------+-------------+------+-----+---------+-------+
| Field  | Type              | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Order_id   | int(11)    | YES  |          | NULL |          |
| Order_Name | varchar(60) | YES  |  | NULL |          |
| C_id             | int(11)          | YES  |          | NULL |          |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

SQL Functions

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two catagories,

- Aggregrate Functions
- Scalar Functions

Aggregrate Functions

These functions return a single value after calculating from a group of values.Following are some frequently used Aggregrate functions.

1) AVG()

Average returns average value after calculating from values in a numeric column.

```
mysql> create table Emp(eid int, name varchar(20), age int, salary int);
Query OK, 0 rows affected (0.01 sec)

mysql> select * from Emp;
+-----+-------+------+--------+
| eid | name  | age  | salary |
+-----+-------+------+--------+
| 401 | Anu   |   22 |   9000 |
| 402 | Shane |   29 |   8000 |
| 403 | Rohan |   34 |   6000 |
| 404 | Scott |   44 |  10000 |
| 405 | Tiger |   35 |   8000 |
+-----+-------+------+--------+
5 rows in set (0.00 sec)

mysql> select avg(salary) from Emp;
+-------------+
| avg(salary) |
+-------------+
|   8200.0000 |
+-------------+
1 row in set (0.00 sec)
```

2) COUNT()

Count returns the number of rows present in the table either based on some condition or without condition.
Its general Syntax is,

```
mysql> select count(eid) from Emp;
```

```
+------------+
| count(eid) |
+------------+
|          5 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> select count(name) from Emp where salary=8000;
+-------------+
| count(name) |
+-------------+
|           2 |
+-------------+
1 row in set (0.00 sec)
```

Example of COUNT(distinct)

```
mysql> select count(distinct salary) from Emp;
+------------------------+
| count(distinct salary) |
+------------------------+
|                      4 |
+------------------------+
1 row in set (0.00 sec)
```

MAX()

MAX function returns maximum value from selected column of the table.

```
mysql> select max(salary) from Emp;
+-------------+
| max(salary) |
+-------------+
|       10000 |
+-------------+
1 row in set (0.00 sec)
```

6) MIN()

MIN function returns minimum value from a selected column of the table.

Syntax for MIN function is,

```
mysql> select min(salary) from Emp;
+-------------+
| min(salary) |
+-------------+
|        6000 |
+-------------+
1 row in set (0.00 sec)
```

7) SUM()

SUM function returns total sum of a selected columns numeric values.
Syntax for SUM is,

```
mysql> select sum(salary) from Emp;
+-------------+
| sum(salary) |
+-------------+
|       41000 |
+-------------+
1 row in set (0.00 sec)
```

Scalar Functions

Scalar functions return a single value from an input value. Following are soe frequently used Scalar Functions.

1) UCASE()

UCASE function is used to convert value of string column to Uppercase character.
Syntax of UCASE,

```
mysql> select UCASE(name) from Emp;
+-------------+
| UCASE(name) |
+-------------+
| ANU         |
| SHANE       |
| ROHAN       |
| SCOTT       |
| TIGER       |
+-------------+
```

5 rows in set (0.00 sec)

2) LCASE()

LCASE function is used to convert value of string column to Lowecase character.
Syntax for LCASE is,

```
mysql> select LCASE(name) from Emp;
+-------------+
| LCASE(name) |
+-------------+
| anu         |
| shane       |
| rohan       |
| scott       |
| tiger       |
+-------------+
5 rows in set (0.00 sec)
```

3) MID()

MID function is used to extract substrings from column values of string type in a table.
Syntax for MID function is,

```
mysql> select mid(name, 2, 3) from Emp;
+-----------------+
| mid(name, 2, 3) |
+-----------------+
| nu              |
| han             |
| oha             |
| cot             |
| ige             |
+-----------------+
5 rows in set (0.00 sec)
```

4) ROUND()

ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values. Syntax of Round function is,

Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join to itself known as, **Self Join**.

Types of Join

The following are the types of JOIN that we can use in SQL.

- Inner
- Outer
- Left
- Right

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows of from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.
Cross JOIN Syntax is,

```
mysql> select * from Customer_Detail;
+------+---------------+---------+
| c_id | Customer_Name | address |
+------+---------------+---------+
|  101 | Yugandhar     | Noida   |
|  102 | Sudhakar      | Delhi   |
|  103 | Mohan         | ATP   |
+------+---------------+---------+
3 rows in set (0.00 sec)

mysql> select * from Order_Detail;
+----------+------------+------+
| Order_id | Order_Name | C_id |
```

```
+----------+------------+------+
|         1001 | Order1 |  101 |
|         1001 | Order1 |   99 |
|         1001 | Order1 |   80 |
+----------+------------+------+
3 rows in set (0.00 sec)

mysql> select * from Customer_Detail CROSS JOIN Order_Detail;
+------+--------------+---------+----------+------------+------+
| c_id | Customer_Name | address | Order_id | Order_Name | C_id |
+------+--------------+---------+----------+------------+------+
|  101 | Yugandhar     | Noida   |       1001 | Order1 |  101 |
|  102 | Sudhakar      | Delhi   |       1001 | Order1 |  101 |
|  103 | Mohan         | ATP   |       1001 | Order1 |  101 |
|  101 | Yugandhar     | Noida   |       1001 | Order1 |   99 |
|  102 | Sudhakar      | Delhi   |       1001 | Order1 |   99 |
|  103 | Mohan         | ATP   |       1001 | Order1 |   99 |
|  101 | Yugandhar     | Noida   |       1001 | Order1 |   80 |
|  102 | Sudhakar      | Delhi   |       1001 | Order1 |   80 |
|  103 | Mohan         | ATP   |       1001 | Order1 |   80 |
+------+--------------+---------+----------+------------+------+
9 rows in set (0.00 sec)

mysql>
```

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query.
Inner Join Syntax is,

```
mysql> select * from Order_Detail, Customer_Detail where
Customer_Detail.c_id=Order_Detail.c_id;
+----------+------------+------+------+--------------+---------+
| Order_id | Order_Name | c_id | c_id | Customer_Name | address |
+----------+------------+------+------+--------------+---------+
|         1001 | Order1 |  101 |  101 | Yugandhar     | Noida   |
|         1002 | Order2 |  102 |  102 | Sudhakar      | Delhi   |
|         1003 | Order3 |  103 |  103 | Mohan         | ATP   |
+----------+------------+------+------+--------------+---------+
3 rows in set (0.00 sec)
```

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

```
mysql> select * from Customer_Detail natural join Order_Detail;
+------+--------------+---------+----------+------------+
| c_id | Customer_Name | address | Order_id | Order_Name |
+------+--------------+---------+----------+------------+
|  101 | Yugandhar     | Noida   |       1001 | Order1 |
|  102 | Sudhakar      | Delhi   |       1002 | Order2 |
|  103 | Mohan         | ATP   |       1003 | Order3 |
+------+--------------+---------+----------+------------+
3 rows in set (0.00 sec)
```

Outer JOIN

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join

The left outer join returns a result table with the **matched data** of two tables then remaining rows of the **left** table and null for the **right** table's column.

```
mysql> select * from Customer_Detail left outer join Order_Detail on
(Customer_Detail.c_id=Order_Detail.c_id);
+------+--------------+---------+----------+------------+------+
| c_id | Customer_Name | address | Order_id | Order_Name | c_id |
+------+--------------+---------+----------+------------+------+
|  101 | Yugandhar     | Noida   |       1001 | Order1 |  101 |
|  102 | Sudhakar      | Delhi   |       1002 | Order2 |  102 |
|  103 | Mohan         | ATP   |       1003 | Order3 |  103 |
|  104 | Priya         | KDT   |       NULL | NULL           | NULL |
|  105 | Indu          | KDT   |       NULL | NULL           | NULL |
+------+--------------+---------+----------+------------+------+
5 rows in set (0.00 sec)
```

Right Outer Join

The right outer join returns a result table with the **matched data** of two tables then remaining rows of the **right table** and null for the **left** table's columns.

```
mysql> select * from Customer_Detail right outer join Order_Detail on
(Customer_Detail.c_id=Order_Detail.c_id);
+------+---------------+---------+----------+------------+------+
| c_id | Customer_Name | address | Order_id | Order_Name | c_id |
+------+---------------+---------+----------+------------+------+
|  101 | Yugandhar     | Noida   |     1001 | Order1 |  101 |
|  102 | Sudhakar      | Delhi   |     1002 | Order2 |  102 |
|  103 | Mohan         | ATP   |     1003 | Order3 |  103 |
+------+---------------+---------+----------+------------+------+
3 rows in set (0.00 sec)
```

Full Outer Join

The full outer join returns a result table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

SQL Alias

Alias is used to give an alias name to a table or a column. This is quite useful in case of large or complex queries. Alias is mainly used for giving a short alias name for a column or a table with complex names.

```
mysql> select * from Customer_Detail as CD;
+------+---------------+---------+
| c_id | Customer_Name | address |
+------+---------------+---------+
|  101 | Yugandhar     | Noida   |
|  102 | Sudhakar      | Delhi   |
|  103 | Mohan         | ATP   |
|  104 | Priya         | KDT   |
|  105 | Indu          | KDT   |
+------+---------------+---------+
5 rows in set (0.00 sec)
```

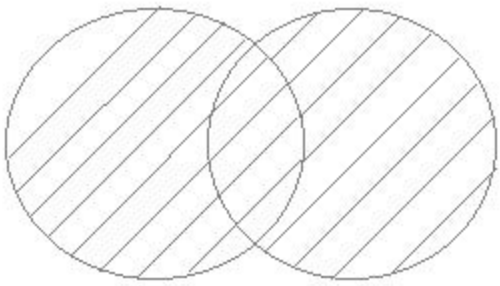
```
mysql> select Customer_Name as CN from Customer_Detail;
+-----------+
| CN     |
+-----------+
| Yugandhar |
| Sudhakar  |
| Mohan          |
| Priya  |
| Indu   |
+-----------+
5 rows in set (0.00 sec)
```

Set Operation in SQL

SQL supports few Set operations to be performed on table data. These are used to get meaningful results from data, under different special conditions.

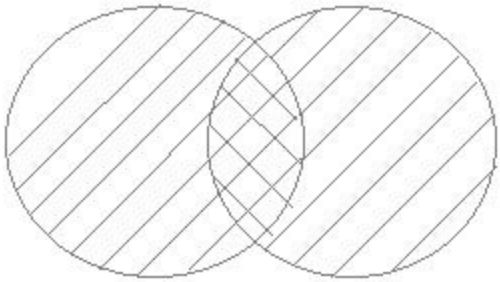
Union

UNION is used to combine the results of two or more Select statements. However it will eliminate duplicate rows from its result set. In case of union, number of columns and datatype must be same in both the tables.



Union All

This operation is similar to Union. But it also shows the duplicate rows.

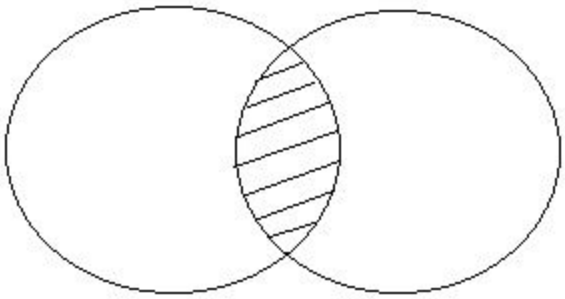


```
mysql> select * from first;
+------+------+
| ID   | Name |
+------+------+
|          1 | abhi |
|          2 | adam |
+------+------+
2 rows in set (0.00 sec)

mysql> select * from second;
+------+---------+
| ID   | Name    |
+------+---------+
|          2 | adam          |
|          3 | Chester |
+------+---------+
2 rows in set (0.00 sec)

mysql> select * from first UNION select * from second;
+------+---------+
| ID   | Name    |
+------+---------+
|          1 | abhi|
|          2 | adam          |
|          3 | Chester |
+------+---------+
3 rows in set (0.00 sec)

mysql> select * from first UNION ALL  select * from second;
+------+---------+
```

```
| ID   | Name    |
+------+---------+
|          1 | abhi|
|          2 | adam         |
|          2 | adam         |
|          3 | Chester |
+------+---------+
4 rows in set (0.00 sec)
```

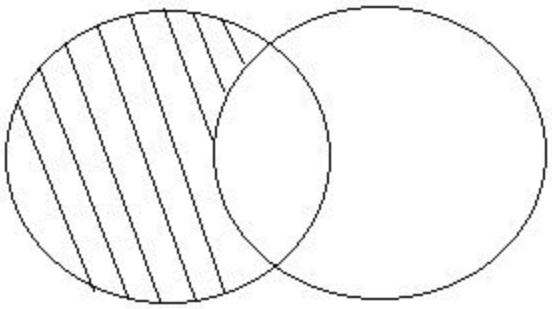
Intersect

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same. MySQL does not support INTERSECT operator.



Minus

Minus operation combines result of two Select statements and return only those result which belongs to first set of result. MySQL does not support INTERSECT operator.



SQL Sequence

Sequence is a feature supported by some database systems to produce unique values on demand. Some DBMS like **MySQL** supports AUTO_INCREMENT in place of Sequence. AUTO_INCREMENT is applied on columns, it automatically increments the column value by 1 each time a new record is entered into the table. Sequence is also some what similar to AUTO_INCREMENT but its has some extra features.

Creating Sequence

Syntax to create sequences is,
CREATE **Sequence** *sequence-name*
start with *initial-value*
increment by *increment-value*
maxvalue *maximum-value*
cycle|nocycle

initial-value specifies the starting value of the Sequence, **increment-value** is the value by which sequence will be incremented and **maxvalue** specifies the maximum value until which sequence will increment itself. **cycle** specifies that if the maximum value exceeds the set limit, sequence will restart its cycle from the begining. **No cycle** specifies that if sequence exceeds **maxvalue** an error will be thrown.

Example to create Sequence

The sequence query is following
CREATE **Sequence** seq_1
start with 1

increment by 1
maxvalue 999
cycle ;

Example to use Sequence

The **class** table,

ID	NAME
1	abhi
2	adam
4	alex

The sql query will be,
INSERT into class value(**seq_1.nextval**,'anu');

Result table will look like,

ID	NAME
1	abhi
2	adam
4	alex
1	anu

Once you use nextval the sequence will increment even if you don't Insert any record into the table.

SQL View

A view in SQL is a logical subset of data from one or more tables. View is used to restrict data access.

```
mysql> select * from Emp;
+-----+-------+------+--------+
| eid | name  | age  | salary |
+-----+-------+------+--------+
| 401 | Anu   |   22 |   9000 |
| 402 | Shane |   29 |   8000 |
| 403 | Rohan |   34 |   6000 |
```

```
| 404 | Scott |   44 |  10000 |
| 405 | Tiger |   35 |   8000 |
+-----+-------+------+--------+
5 rows in set (0.00 sec)
```

Syntax for creating a View,

```
mysql> create view emp_sal as select name, salary from Emp;
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+------------------+
| Tables_in_Testing |
+------------------+
| Customer_Detail   |
| Emp               |
| Order_Detail      |
| emp_bonus         |
| emp_sal           |
| first             |
| markslist         |
| salary            |
| student           |
+------------------+
9 rows in set (0.00 sec)

mysql> select * from emp_sal;
+-------+--------+
| name  | salary |
+-------+--------+
| Anu   |   9000 |
| Shane |   8000 |
| Rohan |   6000 |
| Scott |  10000 |
| Tiger |   8000 |
+-------+--------+
5 rows in set (0.00 sec)
```

How do we delete a view table from database?

```
mysql> show tables;
+------------------+
| Tables_in_Testing |
```

```
+-------------------+
| Customer_Detail   |
| Emp               |
| Order_Detail      |
| emp_bonus         |
| first             |
| markslist         |
| salary            |
| student           |
+-------------------+
8 rows in set (0.00 sec)
```

mysql> drop view salary;
Query OK, 0 rows affected (0.00 sec)

```
mysql> show tables;
+-------------------+
| Tables_in_Testing |
+-------------------+
| Customer_Detail   |
| Emp               |
| Order_Detail      |
| emp_bonus         |
| first             |
| markslist         |
| student           |
+-------------------+
7 rows in set (0.00 sec)
```