UNIT III: Basic structure of SQL queries, writing simple queries, Complex queries and nested Subqueries in SQL, Aggregate functions in SQL, Effect of NULL values on result, Defining a Relational Schema, View definitions and constraints, types of keys.

1.What is MySQL?

- MySQL is a fast, easy to use relational database. It is currently the most popular open-source database. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications.
- MySQL is used for many small and big businesses.
- It is written in C and C++.

Reasons of popularity

MySQL is becoming so popular because of these following reasons:

- MySQL is an open-source database so you don't have to pay a single penny to use it.
- MySQL is a very powerful program so it can handle a large set of functionalities of the most expensive and powerful database packages.
- MySQL is customizable because it is an open-source database and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.
- MySQL is quicker than other databases so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

2. MySQL Features:

- **Relational Database Management System (RDBMS):** MySQL is a relational database management system.
- **Easy to use:** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
- **Free to download:** MySQL is free to use and you can download it from MySQL official website.
- **It is scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
- **Compatible on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).
- **Allows roll-back:** MySQL allows transactions to be rolled back, commit and crash recovery.

- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

Disadvantages / Drawback of MySQL:

Following are the few disadvantages of MySQL:

- MySQL version less than 5.0 doesn't support ROLE, COMMIT and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

3. MySQL Data Types:

- A Data Type specifies a particular type of data, like integer, floating points, Boolean etc. It also identifies the possible values for that type, the operations that can be performed on that type and the way the values of that type are stored.
- MySQL supports a lot number of SQL standard data types in various categories. It uses many different data types broken into mainly three categories: numeric, date and time, and string types.

Numeric Data Type:

Data Type Syntax	**Description**
INT	A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
TINYINT	A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
SMALLINT	A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
MEDIUMINT	A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
BIGINT	A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
FLOAT(m,d)	A floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is

	the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float.
DOUBLE(m,d)	A double precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double.
DECIMAL(m,d)	An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.

Date and Time Data Type:

Data Type Syntax	**Maximum Size**	**Explanation**
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'yyyy-mm-dd'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'yyyy-mm-dd hh:mm:ss'.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	Default is 4 digits.

String Data Types:

Data Type Syntax	**Maximum Size**	**Explanation**
CHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARCHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.
TINYTEXT(size)	Maximum size of 255 characters.	Where size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.	Where size is the number of characters to store.
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characters to store.
BINARY(size)	Maximum size of 255 characters.	Where size is the number of binary characters to store. Fixed-length strings. Space padded on right to equal size characters. (introduced in MySQL 4.1.2)
VARBINARY(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string. (introduced in MySQL 4.1.2)

4.How to install MySQL:

Download MySQL

Follow these steps:

- Go to MySQL official website **http://www.mysql.com/downloads/**
- Choose the version number for MySQL community server which you want.

Installing MySQL on Windows

Your downloaded MySQL is neatly packaged with an installer. Download the installer package, unzip it anywhere and run setup.exe.

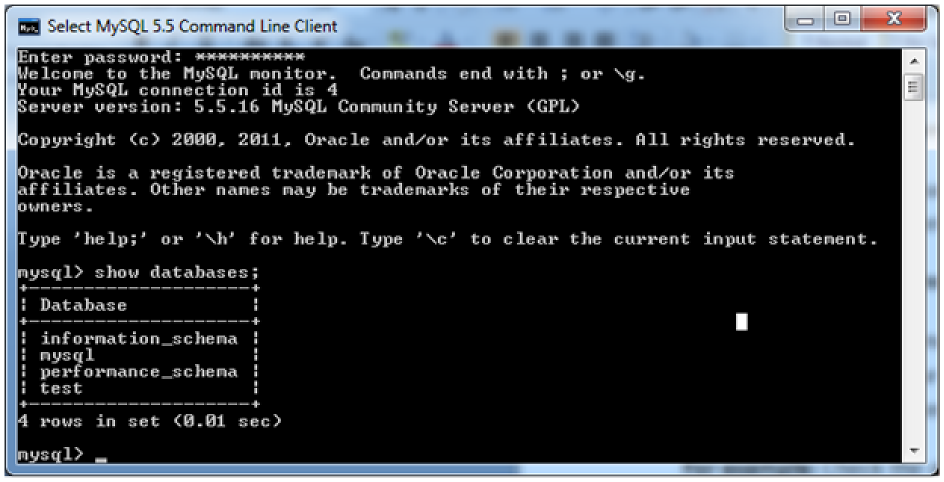
By default, this process will install everything under C:\mysql.

Verify MySQL installation

Once MySQL has been successfully installed, the base tables have been initialized, and the server has been started, you can verify its working via some simple tests.

Open your MySQL Command Line Client, it should be appeared with a mysql> prompt. If you have set any password, write your password here. Now, you are connected to the MySQL server and you can execute all the SQL command at mysql> prompt as follows:

For example: Check the already created databases with show databases command:

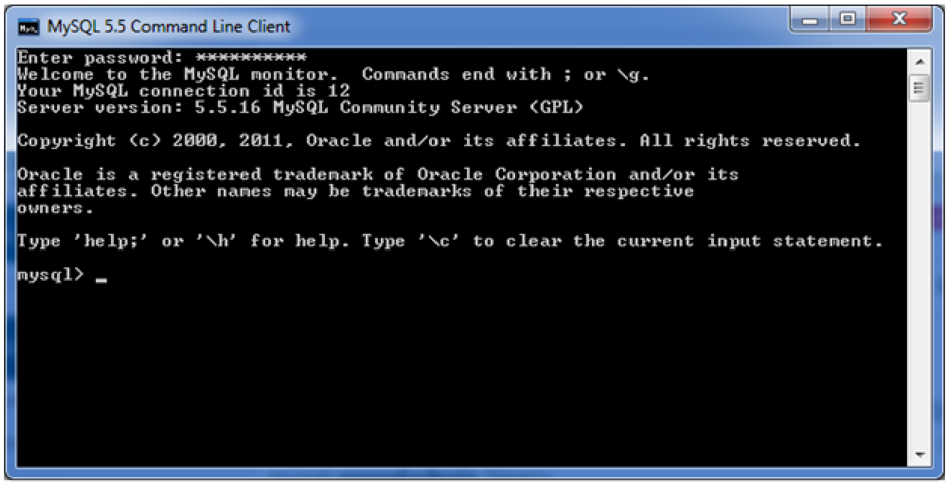


```
Enter password: ***********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.01 sec)

mysql> _
```

MySQL Create Database

You can create a MySQL database by using MySQL Command Line Client.

Open the MySQL console and write down password, if you set one while installation. You will get the following:



```
Enter password: ***********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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> _
```

Now you are ready to create database.

Syntax:

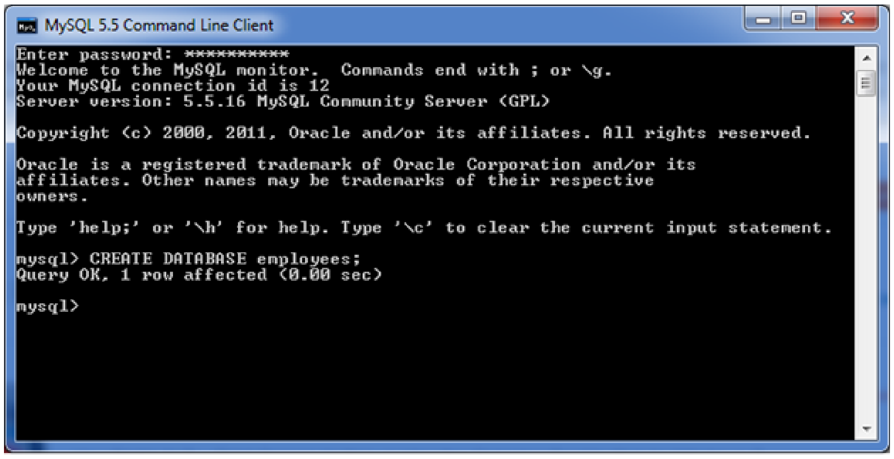
1. CREATE DATABASE database_name;

Example:

Let's take an example to create a database name "employees"

1. CREATE DATABASE employees;

It will look like this:

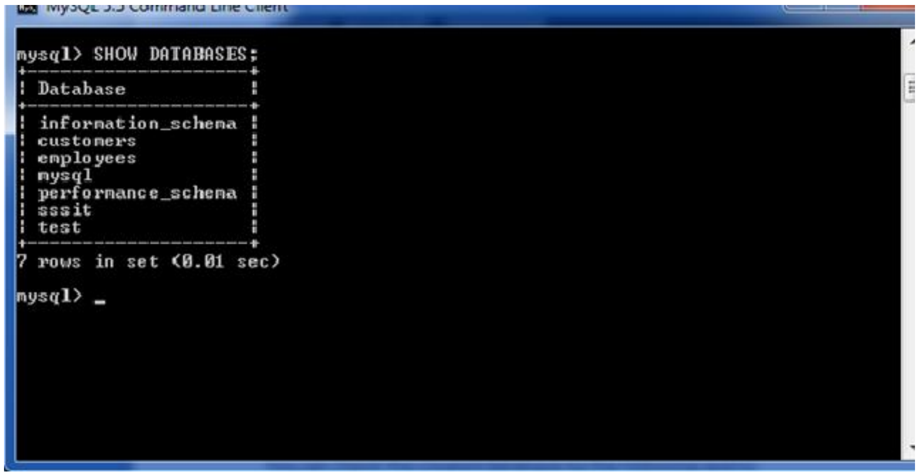


```
Enter password: ***********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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> CREATE DATABASE employees;
Query OK, 1 row affected (0.00 sec)

mysql>
```

You can check the created database by the following query:

1. SHOW DATABASES;

Output



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| employees          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql> _
```

MySQL SELECT Database

SELECT Database is used in MySQL to select a particular database to work with. This query is used when multiple databases are available with MySQL Server.

You can use SQL command **USE** to select a particular database.

Syntax:

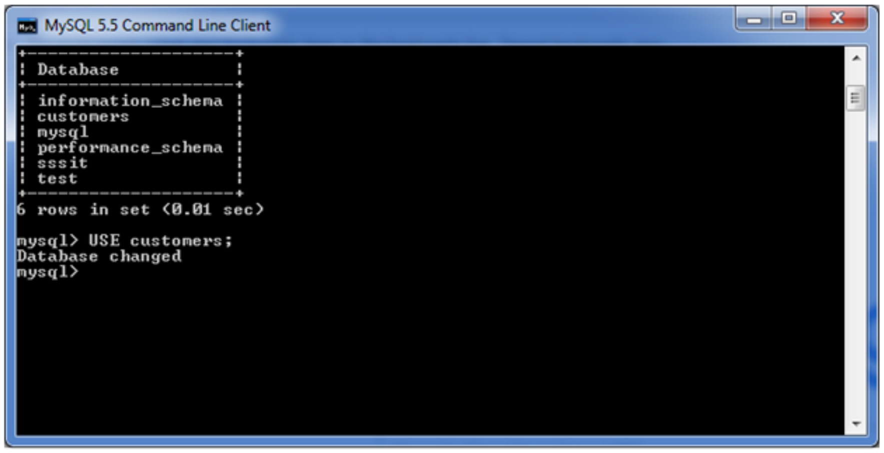
1. USE database_name;

Example:

Let's take an example to use a database name "customers".

1. USE customers;

It will look like this:



```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql>
```

MySQL Drop Database

You can drop/delete/remove a MySQL database easily with the MySQL command. You should be careful while deleting any database because you will lose your all the data available in your database.

Syntax:

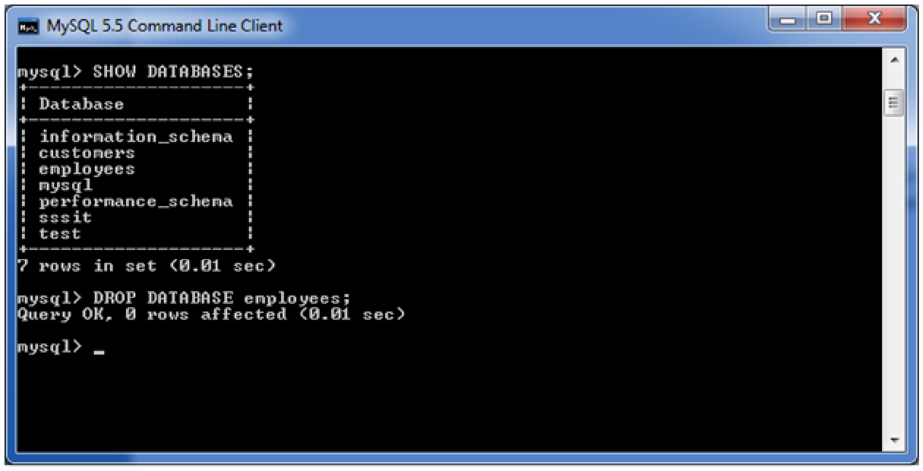
1. DROP DATABASE database_name;

Example:

Let's take an example to drop a database name "employees"

1. DROP DATABASE employees;

It will look like this:

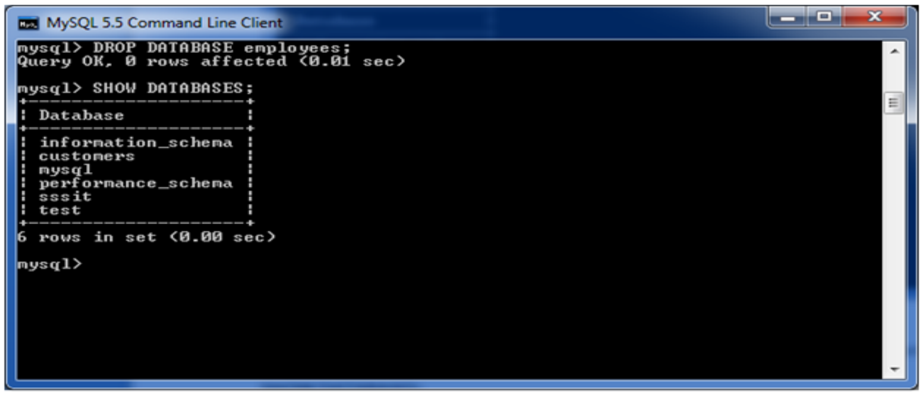


```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| employees          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql> DROP DATABASE employees;
Query OK, 0 rows affected (0.01 sec)

mysql> _
```

Now you can check that either your database is removed by executing the following query:

1. SHOW DATABASES;

Output:



```
mysql> DROP DATABASE employees;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.00 sec)

mysql>
```

5. Constraints in SQL:

A constraint is a mechanism which is used to stop or restrict invalid data into the table. Constraints can be divided into following two types,

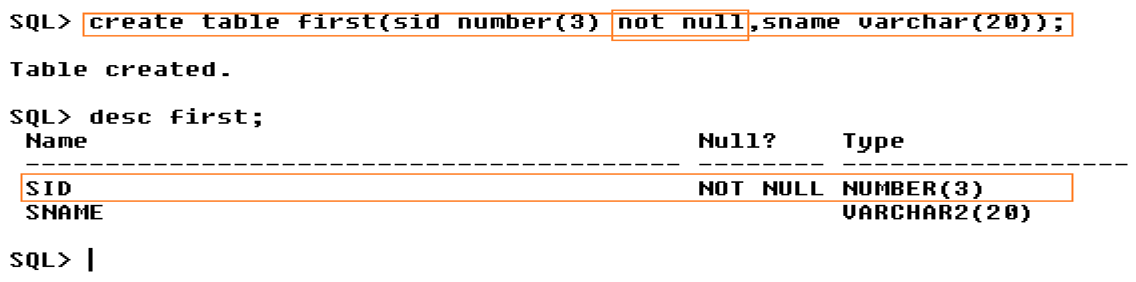
- **Column level constraints:** limits only column data
- **Table level constraints:** limits whole table data

<u>**Constraints in SQL:**</u> **1.NOT NULL. 2. UNIQUE. 3.DEFAULT 4. PRIMARY KEY 5. FOREIGN KEY**

<u>NOT NULL Constraint:</u>

- If you do not enter the value in any particular column in the table that column automatically take null.
- **Not null Constraints** does not allow entering null value on a particular column in a table.
- we can apply not null constraints on more than one column in same table.

Example:

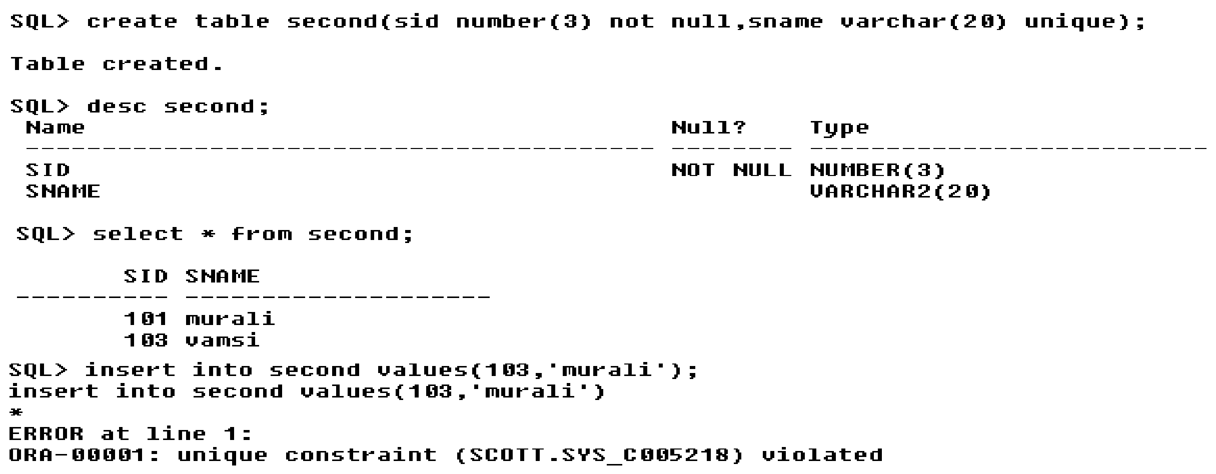


```
SQL> create table first(sid number(3) not null,sname varchar(20));

Table created.

SQL> desc first;
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
 SID                                       NOT NULL NUMBER(3)
 SNAME                                              VARCHAR2(20)

SQL> |
```

<u>Unique Constraint:</u>

- **Unique Constraints** does not allow duplicate values.
- We can apply unique constraints on more than one column in same table.

Example:

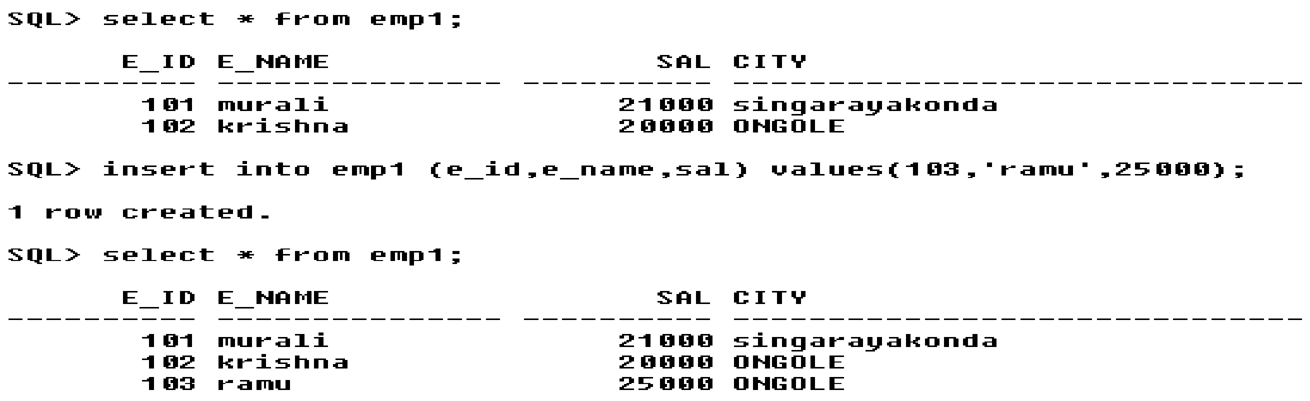


```
SQL> create table second(sid number(3) not null,sname varchar(20) unique);

Table created.

SQL> desc second;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SID                                       NOT NULL NUMBER(3)
 SNAME                                              VARCHAR2(20)

 SQL> select * from second;

       SID SNAME
 ---------- --------------------
       101 murali
       103 vamsi
SQL> insert into second values(103,'murali');
insert into second values(103,'murali')
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.SYS_C005218) violated
```

Default Constraints:

- ❖ Default Constraints is used to insert a default value into a column.
- ❖ The default value will be added to all new records, if no other value is specified.

Example:-



```
SQL> select * from emp1;

      E_ID E_NAME                      SAL CITY
---------- --------------- ---------- --------------------------------
       101 murali               21000 singarayakonda
       102 krishna              20000 ONGOLE

SQL> insert into emp1 (e_id,e_name,sal) values(103,'ramu',25000);

1 row created.

SQL> select * from emp1;

      E_ID E_NAME                      SAL CITY
---------- --------------- ---------- --------------------------------
       101 murali               21000 singarayakonda
       102 krishna              20000 ONGOLE
       103 ramu                 25000 ONGOLE
```

Primary Key Constraints

- ❖ It is used to define a key column of a table.
- ❖ Primary Key Constraints does not allow duplicate as well as null values.
- ❖ This constraint is supported with an index automatically.
- ❖ We cannot apply Primary Key constraints on more than one column in same table.

Primary key= NOT NULL + UNIQUE + INDEX

Example:

Mysql> create table dept (dept_no int(10) primary key, dept_name char(20));

FOREIGN KEY:

- ❖ A foreign key a column that is used to establish a link between two tables.
- ❖ In simple words we can say that, a foreign key is one column in a table that is used to point primary key column in another table.
- ❖ It allows NULL and Duplicate values.
- ❖ It can be related to either primary key or unique constraint column of other table.

$$\boldsymbol{primary\ key}/_{\boldsymbol{unique}} \quad < ------> \boldsymbol{foreign\ key}$$

Example:

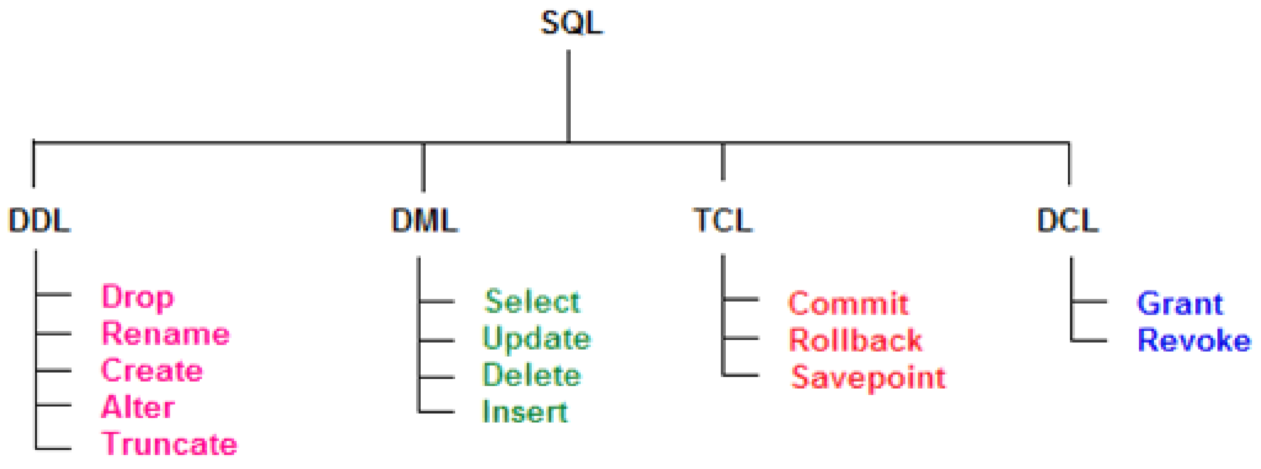
Mysql>create table emp (emp_id int(10) primary key, emp_name char(20), emp_age int(10),

salary int(10), dept_no int(10), foreign key (dept_no) references dept(dept_no));

6. Explain about SQL commands?

SQL Commands are mainly classified into four types, which are DDL command, DML command, TCL command and DCL command. **SQL is mainly divided into four sub languages**

- **Data Definition Language (DDL)**
- **Data Manipulation Language (DML)**
- **Transaction Control Language (TCL)**
- **Data Control Language (DCL)**



<u>DDL commands: -</u>

DDL is means **Data Definition Language,** it is used to define the database schema and structures.

- ✓ CREATE – to create database and its objects like (table, index, views, and triggers).
- ✓ ALTER – it can change the structure of the existing database.
- ✓ DROP – delete objects from the database.
- ✓ TRUNCATE – remove all records from a table, including all spaces allocated for the records are removed.
- ✓ RENAME – rename an object.

<u>DML commands: -</u>

DML is means **Data Manipulation Language** which deals with data manipulation, and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE etc, and it is used to store, modify, retrieve, delete and update data in database.

- ✓ SELECT – retrieve data from the database.
- ✓ INSERT – insert data into a table.
- ✓ UPDATE – updates existing data within a table.
- ✓ DELETE – Delete all records from a database table.

DCL commands: -

DCL is means **Data Control Language,** by using the DCL commands the data administrator can manage the data control from other users in the database environment.

- ✓ GRANT – allow users access rights (permissions) to database.
- ✓ REVOKE – withdraw user's access rights (permissions) given by using the GRANT command.

TCL commands: -

TCL is means Transaction Control Language which deals with transaction within a database.

- ✓ COMMIT – commits a Transaction.
- ✓ ROLLBACK – rollback a transaction in case of any error occurs.
- ✓ SAVEPOINT – to rollback the transaction making points within groups.

7.DDL:

7.1 MySQL CREATE TABLE

The MySQL CREATE TABLE command is used to create a new table into the database. A table creation command requires three things:

- Name of the table
- Names of fields
- Definitions for each field

Syntax: Following is a generic syntax for creating a MySQL table in the database.

```
1. CREATE TABLE table_name (column_name data_type(size),

                            column_name data_type(size),

                            …...);
```

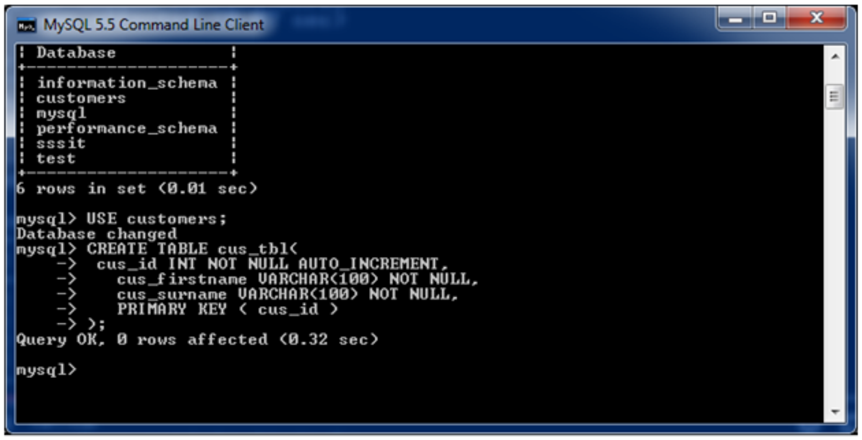
Example: Here, we will create a table named "cus_tbl" in the database "customers".

```
1. CREATE TABLE cus_tbl(
2.   cus_id INT NOT NULL AUTO_INCREMENT,
3.   cus_firstname VARCHAR(100) NOT NULL,
4.   cus_surname VARCHAR(100) NOT NULL,
5.   PRIMARY KEY ( cus_id )
6. );
```

Note: 1. Here, NOT NULL is a field attribute and it is used because we don't want this field to be NULL. If you will try to create a record with NULL value, then MySQL will raise an error.

2. The field attribute AUTO_INCREMENT specifies MySQL to go ahead and add the next available number to the id field.PRIMARY KEY is used to define a column as primary key. You can use multiple columns separated by comma to define a primary key.

Visual representation of creating a MySQL table:

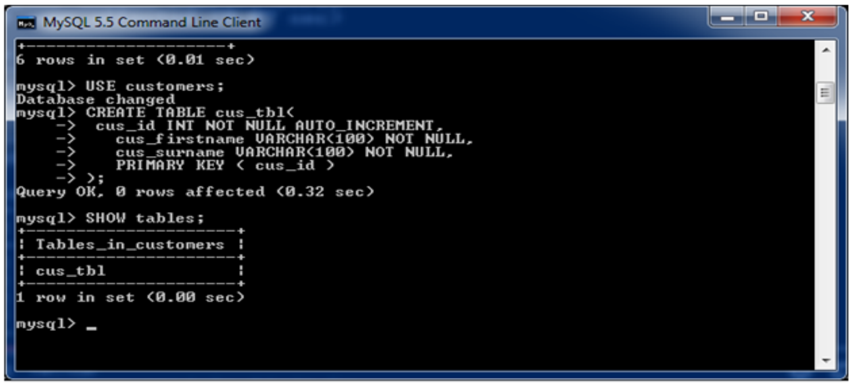


```
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql> CREATE TABLE cus_tbl(
    ->  cus_id INT NOT NULL AUTO_INCREMENT,
    ->    cus_firstname VARCHAR(100) NOT NULL,
    ->    cus_surname VARCHAR(100) NOT NULL,
    ->    PRIMARY KEY ( cus_id )
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql>
```

See the created table:

Use the following command to see the table already created:

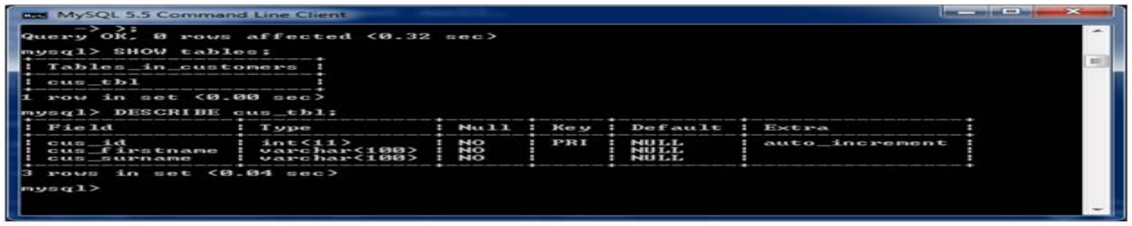
1. SHOW tables;



```
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql> CREATE TABLE cus_tbl(
    ->  cus_id INT NOT NULL AUTO_INCREMENT,
    ->    cus_firstname VARCHAR(100) NOT NULL,
    ->    cus_surname VARCHAR(100) NOT NULL,
    ->    PRIMARY KEY ( cus_id )
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql> SHOW tables;
+---------------------+
| Tables_in_customers |
+---------------------+
| cus_tbl             |
+---------------------+
1 row in set (0.00 sec)

mysql> _
```

See the table structure:

Use the following command to see the table already created: DESCRIBE cus_tbl;



```
    -> );
Query OK, 0 rows affected (0.32 sec)
mysql> SHOW tables;
+---------------------+
| Tables_in_customers |
+---------------------+
| cus_tbl             |
+---------------------+
1 row in set (0.00 sec)
mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.04 sec)
mysql>
```

7.2 MySQL ALTER Table

MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

1) ADD a column in the table

Syntax: ALTER TABLE table_name ADD new_column_name column_definition

[FIRST | AFTER column_name];

Parameters

table_name: It specifies the name of the table that you want to modify.

new_column_name: It specifies the name of the new column that you want to add to the table.

column_definition: It specifies the data type and definition of the column (NULL or NOT NULL, etc).

FIRST | AFTER column_name: It is optional. It tells MySQL where in the table to create the column. If this parameter is not specified, the new column will be added to the end of the table.

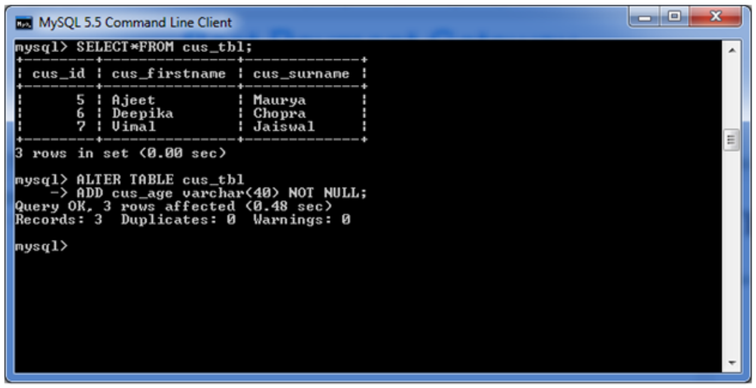
Example:

In this example, we add a new column "cus_age" in the existing table "cus_tbl".

Use the following query to do this:

1. ALTER TABLE cus_tbl ADD cus_age varchar(40) NOT NULL;

Output:

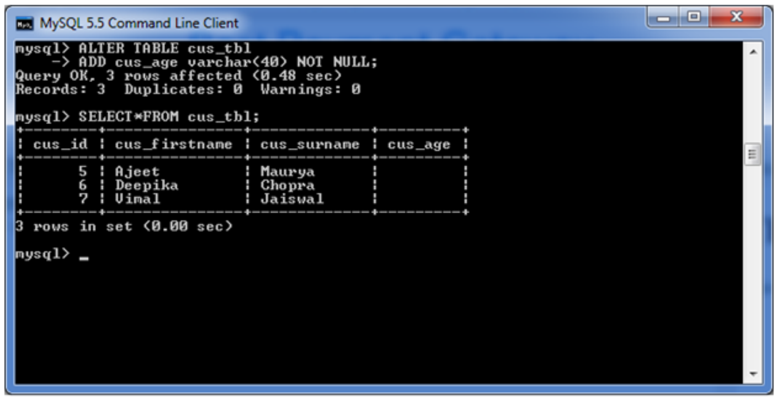


```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
|      6 | Deepika       | Chopra      |
|      7 | Vimal         | Jaiswal     |
+--------+---------------+-------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> ADD cus_age varchar(40) NOT NULL;
Query OK, 3 rows affected (0.48 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the recently added column:

1. SELECT* FROM cus_tbl;

Output:



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_age varchar(40) NOT NULL;
Query OK, 3 rows affected (0.48 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+---------+
| cus_id | cus_firstname | cus_surname | cus_age |
+--------+---------------+-------------+---------+
|      5 | Ajeet         | Maurya      |         |
|      6 | Deepika       | Chopra      |         |
|      7 | Vimal         | Jaiswal     |         |
+--------+---------------+-------------+---------+
3 rows in set (0.00 sec)

mysql> _
```

2) Add multiple columns in the table

Syntax:

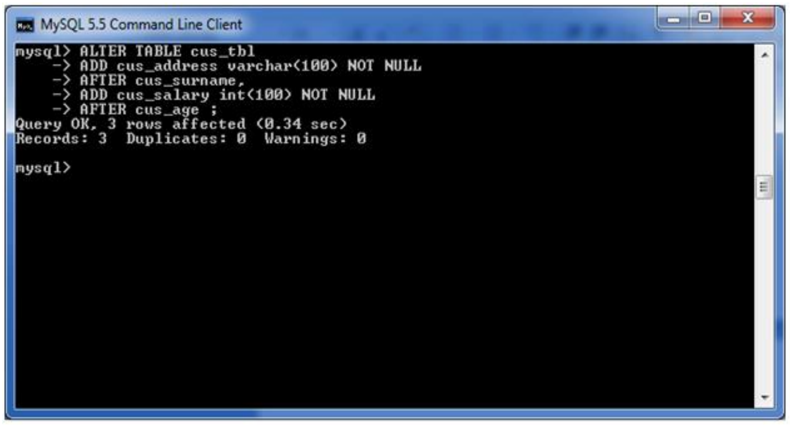
1. ALTER TABLE table_name ADD new_column_name column_definition
2. [FIRST | AFTER column_name],
3. ADD new_column_name column_definition [FIRST | AFTER column_name],
4. ...
5. ;

Example:

In this example, we add two new columns "cus_address", and cus_salary in the existing table "cus_tbl". cus_address is added after cus_surname column and cus_salary is added after cus_age column.

Use the following query to do this:

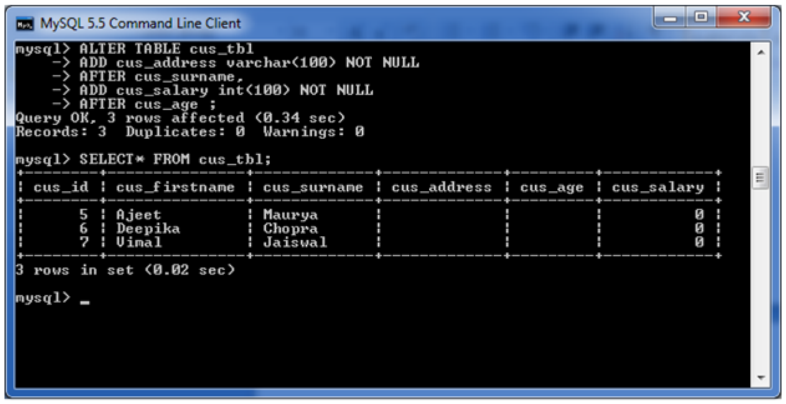
1. ALTER TABLE cus_tbl ADD cus_address varchar(100) NOT NULL
2. AFTER cus_surname, ADD cus_salary int(100) NOT NULL AFTER cus_age ;



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_address varchar(100) NOT NULL
    -> AFTER cus_surname,
    -> ADD cus_salary int(100) NOT NULL
    -> AFTER cus_age ;
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the recently added columns:

1. SELECT* FROM cus_tbl;



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_address varchar(100) NOT NULL
    -> AFTER cus_surname,
    -> ADD cus_salary int(100) NOT NULL
    -> AFTER cus_age ;
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.02 sec)

mysql> _
```

7.3 MODIFY column in the table

The MODIFY command is used to change the column definition of the table.

Syntax:

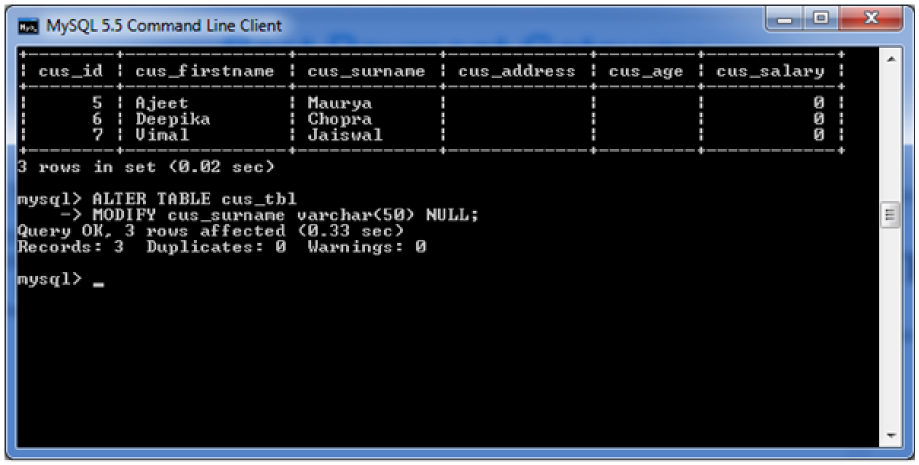
1. ALTER TABLE table_name MODIFY column_name column_definition
2. [FIRST | AFTER column_name];

Example:

In this example, we modify the column cus_surname to be a data type of varchar(50) and force the column to allow NULL values.

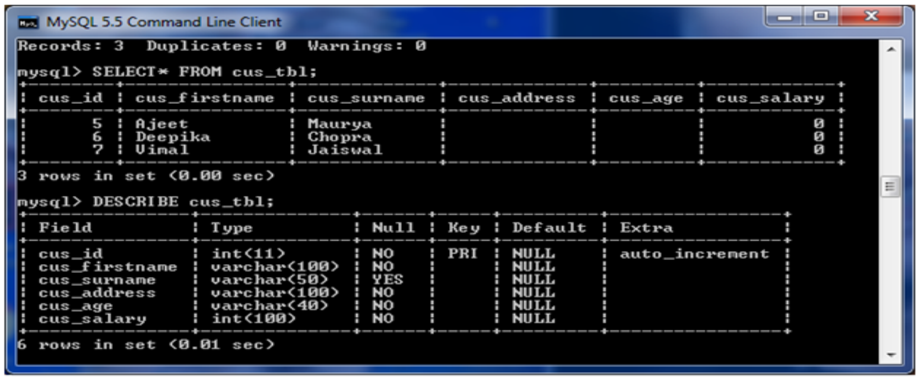
Use the following query to do this:

1. ALTER TABLE cus_tbl MODIFY cus_surname varchar(50) NULL;



```
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.02 sec)

mysql> ALTER TABLE cus_tbl
    -> MODIFY cus_surname varchar(50) NULL;
Query OK, 3 rows affected (0.33 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

See the table structure:



```
Records: 3  Duplicates: 0  Warnings: 0
mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(50)  | YES  |     | NULL    |                |
| cus_address   | varchar(100) | NO   |     | NULL    |                |
| cus_age       | varchar(40)  | NO   |     | NULL    |                |
| cus_salary    | int(100)     | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

7.4 DROP column in table

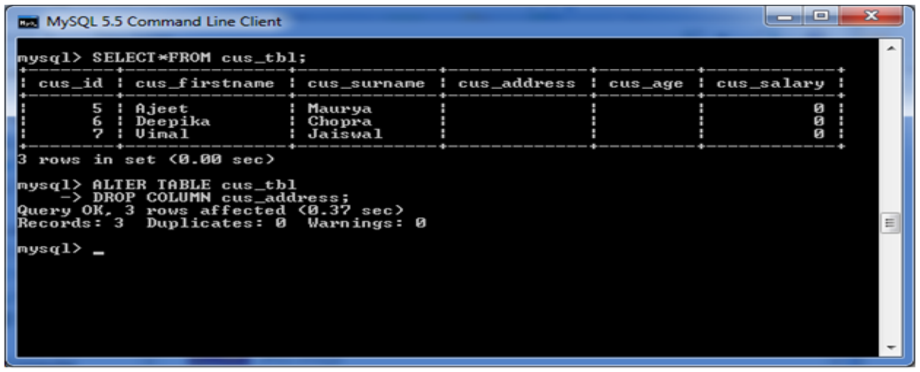
Syntax: ALTER TABLE table_name DROP COLUMN column_name;

Let's take an example to drop the column name "cus_address" from the table "cus_tbl".

Use the following query to do this:

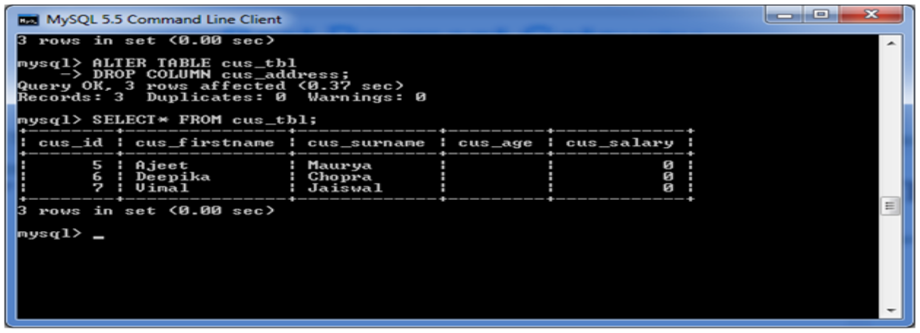
1. ALTER TABLE cus_tbl DROP COLUMN cus_address;

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> DROP COLUMN cus_address;
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

See the table structure:



```
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> DROP COLUMN cus_address;
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_age | cus_salary |
+--------+---------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |         |          0 |
|      6 | Deepika       | Chopra      |         |          0 |
|      7 | Vimal         | Jaiswal     |         |          0 |
+--------+---------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> _
```

7.5 RENAME column in table

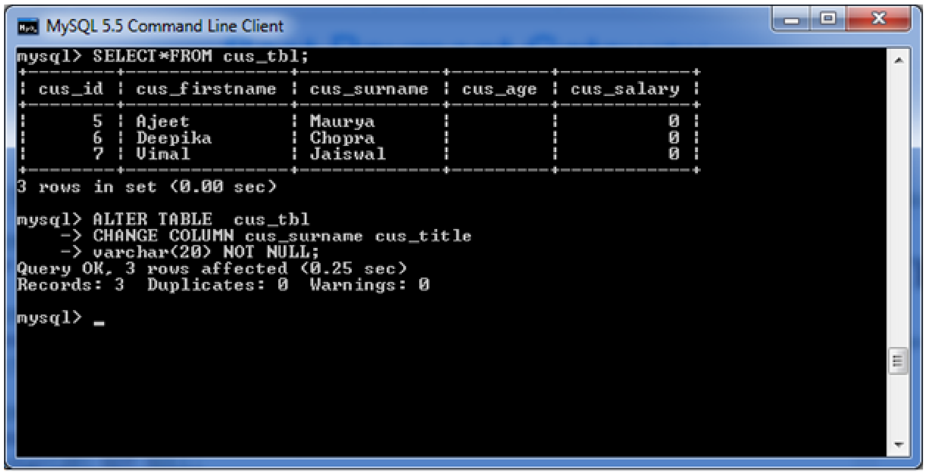
Syntax: ALTER TABLE table_name rename COLUMN old_column_name TO new_column_name;

Example: In this example, we will change the column name "cus_surname" to "cus_title".

Use the following query to do this:

1. ALTER TABLE cus_tbl rename COLUMN cus_surname TO cus_title varchar(20) NOT NULL;

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_age | cus_salary |
+--------+---------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |         |          0 |
|      6 | Deepika       | Chopra      |         |          0 |
|      7 | Vimal         | Jaiswal     |         |          0 |
+--------+---------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE  cus_tbl
    -> CHANGE COLUMN cus_surname cus_title
    -> varchar(20) NOT NULL;
Query OK, 3 rows affected (0.25 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

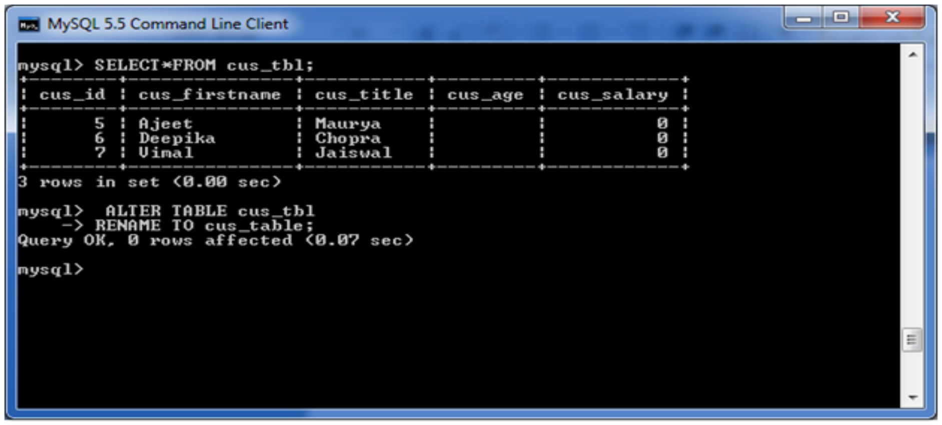
7.6 RENAME table

Syntax: ALTER TABLE old_table_name RENAME TO new_table_name;

Example: In this example, the table name cus_tbl is renamed as cus_table.

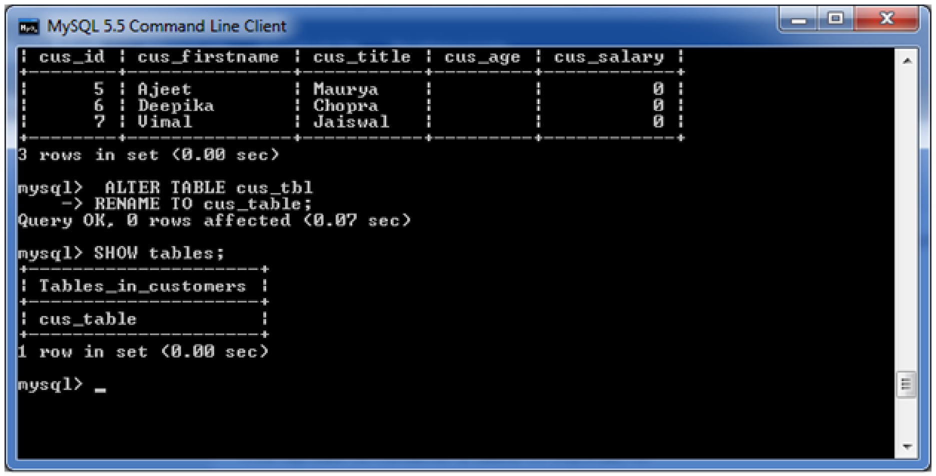
1. ALTER TABLE cus_tbl RENAME TO cus_table;

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-----------+---------+------------+
| cus_id | cus_firstname | cus_title | cus_age | cus_salary |
+--------+---------------+-----------+---------+------------+
|      5 | Ajeet         | Maurya    |         |          0 |
|      6 | Deepika       | Chopra    |         |          0 |
|      7 | Vimal         | Jaiswal   |         |          0 |
+--------+---------------+-----------+---------+------------+
3 rows in set (0.00 sec)

mysql>  ALTER TABLE cus_tbl
    -> RENAME TO cus_table;
Query OK, 0 rows affected (0.07 sec)

mysql>
```

See the renamed table:



MySQL 5.5 Command Line Client

```
| cus_id | cus_firstname | cus_title | cus_age | cus_salary |
+--------+---------------+-----------+---------+------------+
|      5 | Ajeet         | Maurya    |         |          0 |
|      6 | Deepika       | Chopra    |         |          0 |
|      7 | Vimal         | Jaiswal   |         |          0 |
+--------+---------------+-----------+---------+------------+
3 rows in set (0.00 sec)

mysql>  ALTER TABLE cus_tbl
    -> RENAME TO cus_table;
Query OK, 0 rows affected (0.07 sec)

mysql> SHOW tables;
+---------------------+
| Tables_in_customers |
+---------------------+
| cus_table           |
+---------------------+
1 row in set (0.00 sec)

mysql> _
```

7.7.MySQL TRUNCATE Table

MYSQL TRUNCATE statement removes the complete data without removing its structure.

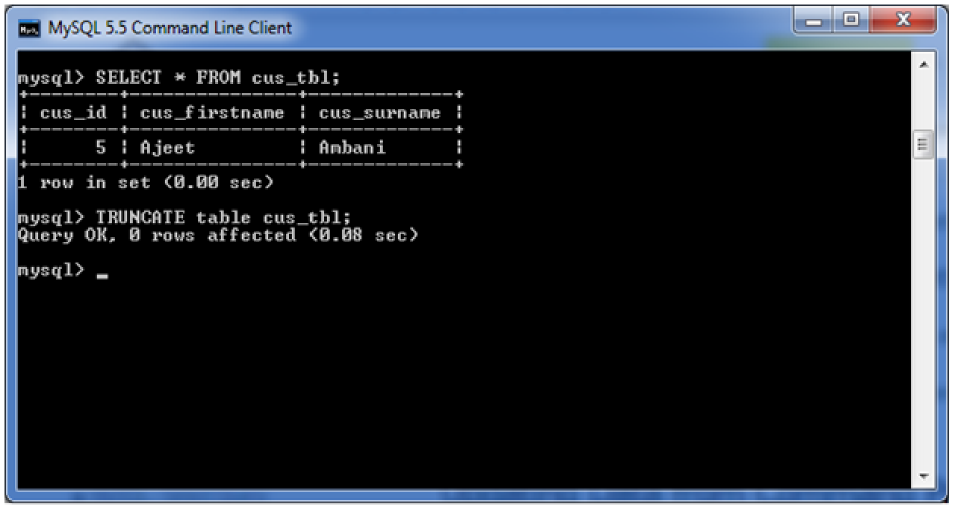
The TRUNCATE TABLE statement is used when you want to delete the complete data from a table without removing the table structure.

Syntax: TRUNCATE TABLE table_name;

Example: This example specifies how to truncate a table. In this example, we truncate the table "cus_tbl".

1. TRUNCATE TABLE cus_tbl;

Output:



MySQL 5.5 Command Line Client

```
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Ambani      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> TRUNCATE table cus_tbl;
Query OK, 0 rows affected (0.08 sec)

mysql> _
```

7.8.MySQL DROP Table

MYSQL DROP table statement removes the complete data with structure.

Syntax: DROP TABLE table_name;

Example: This example specifies how to drop a table. In this example, we are dropping the table "cus_tbl".

```
DROP TABLE  cus_tbl;
```

MySQL TRUNCATE Table vs DROP Table

You can also use DROP TABLE command to delete complete table but it will remove complete table data and structure both. You need to re-create the table again if you have to store some data. But in the case of TRUNCATE TABLE, it removes only table data not structure. You don't need to re-create the table again because the table structure already exists.

8.DML:

8.1 MySQL INSERT Statement

MySQL INSERT statement is used to insert data in MySQL table within the database. We can insert single or multiple records using a single query in MySQL.

Syntax:

The SQL INSERT INTO command is used to insert data in MySQL table. Following is a generic syntax:

1. INSERT INTO table_name (field1, field2,...fieldN)
2. VALUES (value1, value2,...valueN);

Field name is optional. If you want to specify partial values, field name is mandatory.

Syntax for all fields:

1. INSERT INTO table_name VALUES (value1, value2,...valueN);

MySQL INSERT Example 1: for all fields

If you have to store all the field values, either specify all field name or don't specify any field.

Example: INSERT INTO emp VALUES (7, 'Sonoo', 40000);

Or,

1. INSERT INTO emp(id,name,salary) VALUES (7, 'Sonoo', 40000);

MySQL INSERT Example 2: for partial fields

In such case, it is mandatory to specify field names.

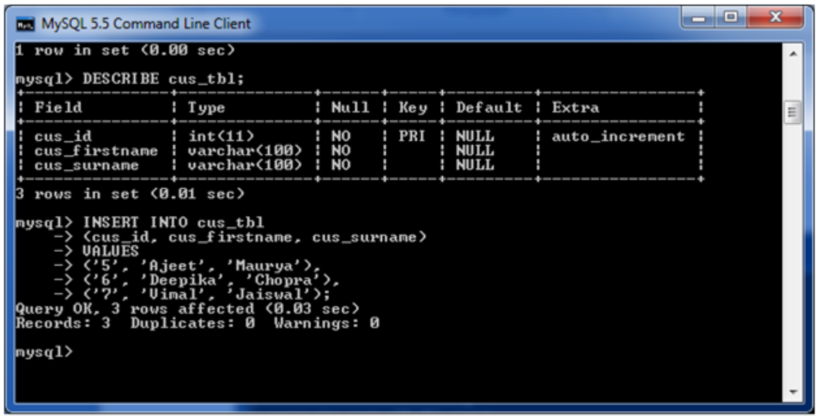
1. INSERT INTO emp(id,name) VALUES (7, 'Sonoo');

MySQL INSERT Example 3: inserting multiple records

Here, we are going to insert record in the "cus_tbl" table of "customers" database.

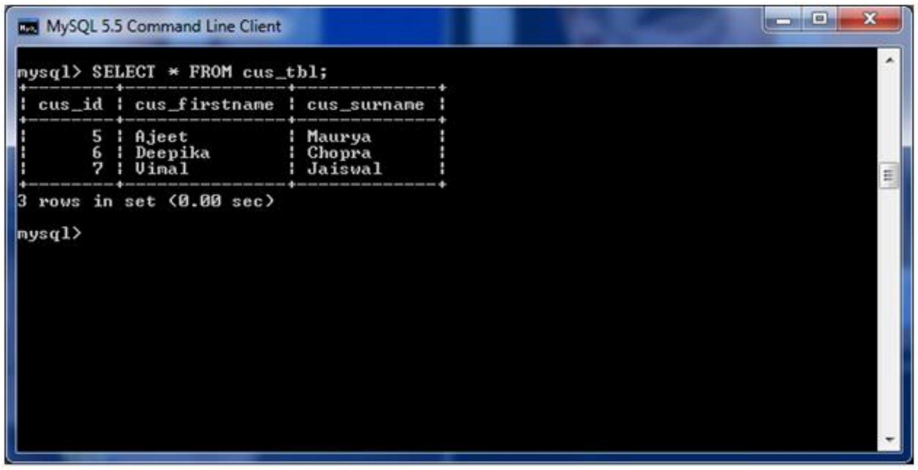
1. INSERT INTO cus_tbl (cus_id, cus_firstname, cus_surname) VALUES (5, 'Ajeet', 'Maurya'),
2. (6, 'Deepika', 'Chopra'), (7, 'Vimal', 'Jaiswal');

Visual Representation:



```
1 row in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)

mysql> INSERT INTO cus_tbl
    -> (cus_id, cus_firstname, cus_surname)
    -> VALUES
    -> ('5', 'Ajeet', 'Maurya'),
    -> ('6', 'Deepika', 'Chopra'),
    -> ('7', 'Vimal', 'Jaiswal');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the data within the table by using the SELECT command:



```
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
|      6 | Deepika       | Chopra      |
|      7 | Vimal         | Jaiswal     |
+--------+---------------+-------------+
3 rows in set (0.00 sec)

mysql>
```

8.2 MySQL UPDATE Query

MySQL UPDATE statement is used to update data of the MySQL table within the database. It is used when you need to modify the table.

Syntax:

Following is a generic syntax of UPDATE command to modify data into the MySQL table:

1. UPDATE table_name SET field1=new-value1, field2=new-value2 [WHERE Clause]

Note:

- One or more field can be updated altogether.
- Any condition can be specified by using WHERE clause.
- You can update values in a single table at a time.
- WHERE clause is used to update selected rows in a table.

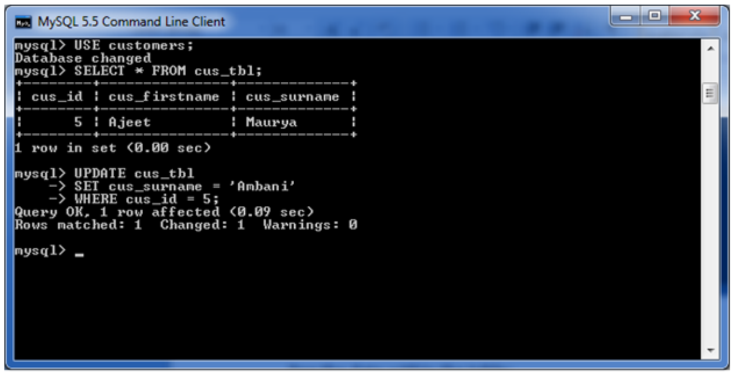
Example:

Here, we have a table "cus_tbl" within the database "customers". We are going to update the data within the table "cus_tbl".

This query will update cus_surname field for a record having cus_id as 5.

1. UPDATE cus_tbl SET cus_surname = 'Ambani' WHERE cus_id = 5;

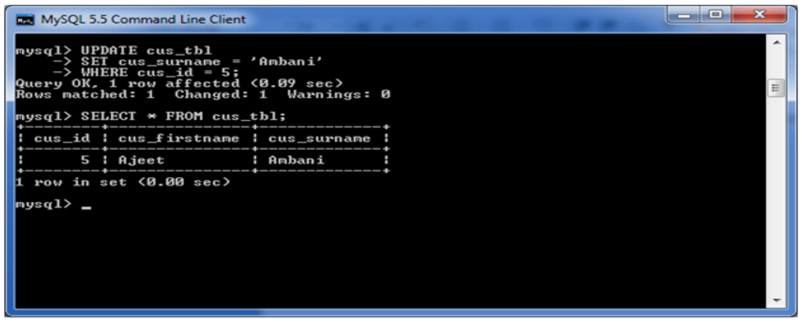
Visual Representation:



```
mysql> USE customers;
Database changed
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> UPDATE cus_tbl
    -> SET cus_surname = 'Ambani'
    -> WHERE cus_id = 5;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> _
```

Output by SELECT query:

1. SELECT * FROM cus_tbl;



```
mysql> UPDATE cus_tbl
    -> SET cus_surname = 'Ambani'
    -> WHERE cus_id = 5;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Ambani      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> _
```

8.3 MySQL DELETE Statement

MySQL DELETE statement is used to delete data from the MySQL table within the database. By using delete statement, we can delete records on the basis of conditions.

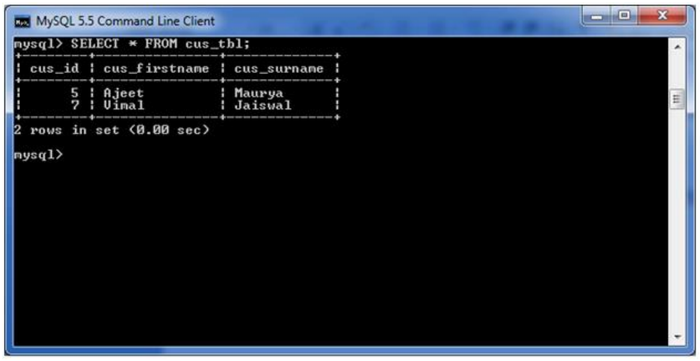
Syntax:

1. DELETE FROM table_name   WHERE (Condition specified);

Example:

1. DELETE FROM cus_tbl WHERE cus_id = 6;

Output:



```
mysql> SELECT * FROM cus_tbl;
+---------+---------------+-------------+
| cus_id  | cus_firstname | cus_surname |
+---------+---------------+-------------+
|       5 | Ajeet         | Maurya      |
|       7 | Vimal         | Jaiswal     |
+---------+---------------+-------------+
2 rows in set (0.00 sec)

mysql>
```

8.4 MySQL SELECT Statement

The MySQL SELECT statement is used to fetch data from the one or more tables in MySQL. We can retrieve records of all fields or specified fields.

Syntax for specified fields:

1. SELECT expressions FROM tables [WHERE conditions];

Syntax for all fields:

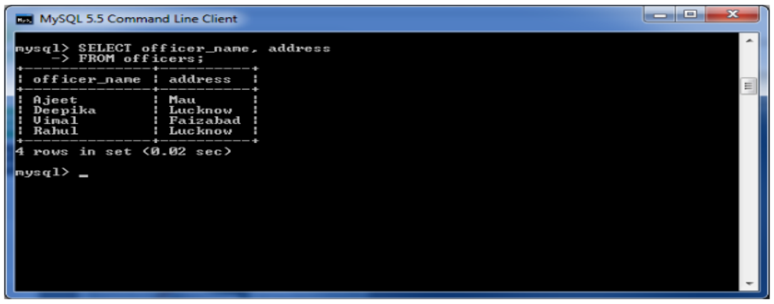
1. SELECT * FROM tables [WHERE conditions];

MySQL SELECT Example 1: for specified fields

In such case, it is mandatory to specify field names.

Example:

1. SELECT officer_name, address FROM officers

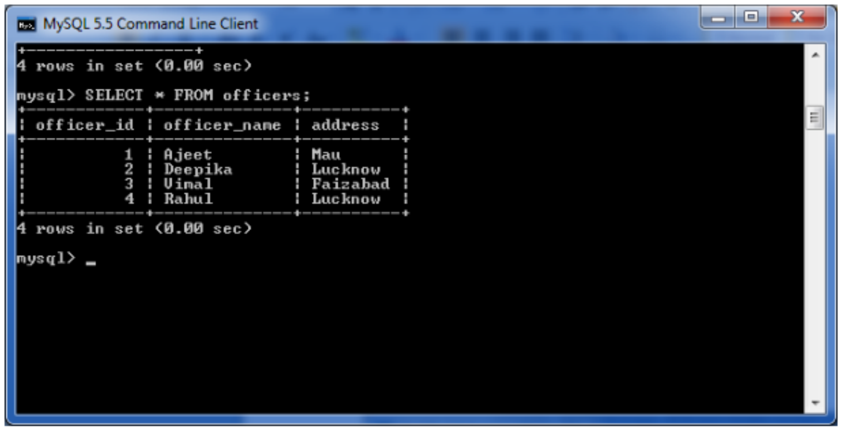


```
mysql> SELECT officer_name, address
    -> FROM officers;
+--------------+----------+
| officer_name | address  |
+--------------+----------+
| Ajeet        | Mau      |
| Deepika      | Lucknow  |
| Vimal        | Faizabad |
| Rahul        | Lucknow  |
+--------------+----------+
4 rows in set (0.02 sec)

mysql> _
```

MySQL SELECT Example 2: for all fields

In such case, we can specify either all fields or * (asterisk) symbol.

1. SELECT * FROM officers

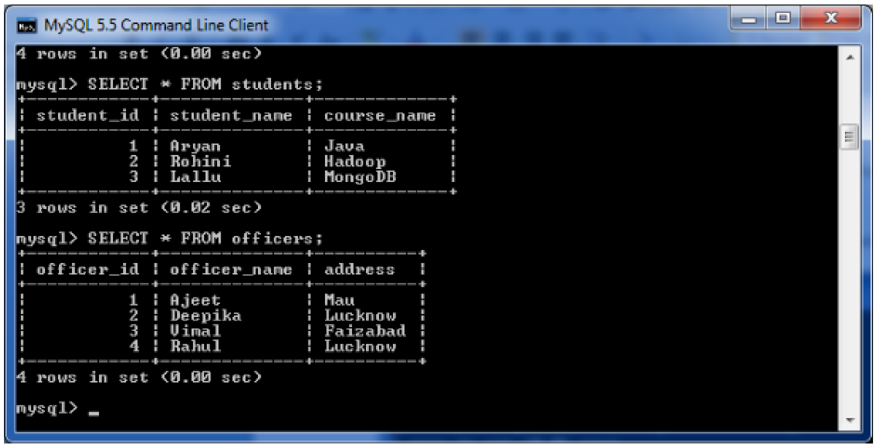


```
+------------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> _
```

MySQL SELECT Example 3: from multiple tables

MySQL SELECT statement can also be used to retrieve records from multiple tables by using JOIN statement.

Let's take two tables "students" and "officers", having the following data.

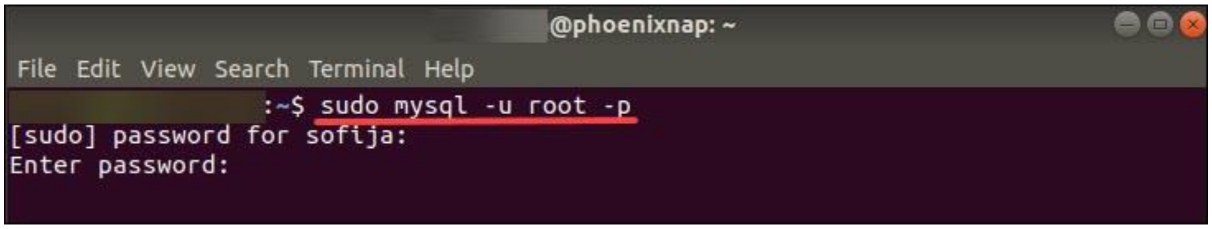


```
4 rows in set (0.00 sec)

mysql> SELECT * FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.02 sec)

mysql> SELECT * FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> _
```

9. How to Create New MySQL User

1. Before you can create a new MySQL user, you need to open a terminal window and launch the MySQL shell as the root user. To do so, enter the following command: sudo mysql –u root –p

2. Type in the root password for this account and press Enter.



```
[redacted]:~$ sudo mysql -u root -p
[sudo] password for sofija:
Enter password:
```

The prompt should change to show that you are in the **mysql>** shell.

3. Next, create a new MySQL user with: CREATE USER 'username' IDENTIFIED BY 'password';

Replace **username** and **password** with a username and password of your choice.

Alternatively, you can set up a user by specifying the machine hosting the database.

- If you are working on the machine with MySQL, use **username@localhost** to define the user.
- If you are connecting remotely, use **username@ip_address**, and replace **ip_address** with the actual address of the remote system hosting MySQL.

Therefore, the command will be: CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';

Or CREATE USER 'username'@'ip_address' IDENTIFIED BY 'password';

You can also create a user that can connect from any machine with the command:

CREATE USER 'username'@'%' IDENTIFIED BY 'password';

How to Grant Permissions in MySQL

Before logging in with a new account, make sure you have set the permissions for the user.

Permissions are actions that the user is allowed to perform in the database. Depending on how much authority you want your user to have, you can grant them one, several or all of the following privileges:

- **All Privileges:** The user account has full access to the database
- **Insert:** The user can insert rows into tables
- **Delete:** The user can remove rows from tables
- **Create:** The user can create entirely new tables and databases

- **Drop:** The user can drop (remove) entire tables and databases
- **Select:** The user gets access to the select command, to read the information in the databases
- **Update:** The user can update table rows
- **Grant Option:** The user can modify other user account privileges

The basic syntax used to grant privileges to a user account is:

```
GRANT permission_type ON database.table TO 'username'@'localhost';
```

For example, **to grant insert privileges to a MySQL user** you would run the command: `GRANT INSERT ON *.* TO 'username'@'localhost';`

You can replace the privilege level according to your needs. Run the command for each privilege you wish to grant.If you want to limit the user's access to a specific database, name that database before the dot. Likewise, you can restrict a user's access to a particular table by naming it after the dot, as in the command below:

```
GRANT INSERT *database_name.table_name* TO 'username'@'localhost';
```

MySQL User Management

This section will help you list the privileges held by a user account, take privileges away from a user, and completely delete a user account. It will also show you how to log out of the root MySQL user account, and log back in under the account you've just created.

How to List MySQL User Account-Privileges

To display all the current privileges held by a user: `SHOW GRANTS username;`

How to Grant All Privileges on a Database in MySQL

To **grant all privileges to MySQL User on all databases** use the command:

```
GRANT ALL PRIVILEGES ON *.* TO 'database_user'@'localhost';
```

However, you can also grant all privileges to a user account on a **specific** database with the following command: `GRANT ALL PRIVILEGES ON database_name.* TO 'database_user'@'localhost';`

To grant all privileges to a user account over a specific table from a database type:

`GRANT ALL PRIVILEGES ON database_name.table_name TO 'database_user'@'localhost';`

Revoke Privileges MySQL User Account

To take back privileges from a specific user, use the **REVOKE** command. It works similar to the GRANT command, its basic syntax being: REVOKE permission_type ON database.table TO 'username'@'localhost';

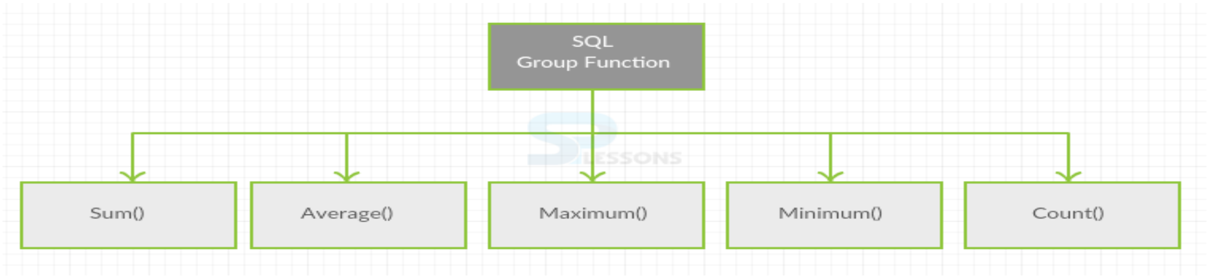
Remove an Entire User Account

To delete a MySQL user account use the command: DROP USER 'username'@'localhost';

10. Aggregate functions in SQL/MySQL Functions:

Function is a type of command which accepts 'n' number of values as input and return a single value. MySQL Functions can be of different types:

- Sum()
- Average()
- Maximum()
- Minimum()
- Count()



Sum(): Calculate the set of values of an expression.

Syntax : Select Sum<column_name> from <table_name>;

Examples: mysql> select sum(salary) from employee;

mysql> select sum(salary)as total salary from employee;

Avg(): Perform the average of an expression.

Syntax: Select avg(column_name) from <table_name>;

Examples: mysql>select avg(salary) from employee;

Maximum(): Perform the maximum value of an expression.

Syntax: Select maximum<column_name>from <table_name>;

Examples: mysql> select max(salary) from employee;

Minimum(): Perform the minimum value of an expression.

Syntax: Select minimum<column_name>from <table_name>;

Examples: mysql> select min(salary) from employee;

Count(): Perform count of an expression.

Syntax: select count<column_name> from <table_name>;

Examples: mysql> select count(emp_id) from employee;

11.MySQL Clauses

Clause is defined as a set of rules, that makes to understand the concepts of MySQL command in Database. MySQL Clauses are very similar to SQL clause, except some functional operations.

1. Distinct clause.
2. Where clause.
3. Group By clause.
4. Having clause.
5. Order by clause .

1. To eliminate duplicate values in a column use Distinct clause.

   mysql> select * from employee;

   mysql> select distinct deptno from employee;

2. Where clause will specify the condition for the result set in the table.
   mysql> select * from employee where ename='murali';

3. To divide the data of a table into groups based on single column use Group By clause.
   mysql> select deptno, max(salary) from employee group by deptno;
   mysql> select deptno, min(salary) from employee group by deptno;

4. To specify a condition with group by clause use Having clause.
   mysql> select deptno,max (salary) from employee group by deptno having max(salary)>13000;

   mysql> select deptno,max(salary) from employee group by deptno having min(salary)<13000;
5. To arrange data of a table either in ascending order or descending order based on single column use Orderby clause i.e., Select * from the table name will show all the rows from the table and order by clause is used to arrange the specific column in ascending or descending order.
   mysql> select * from employee order by salary;
   mysql> select * from employee order by salary desc;

12.Explain subqueries or co-related queries or nested queries ?

A sub query is a query inside another query. A sub query or inner query is used to generate the required information used as input for outer query. A sub query has characteristics. · A sub query is normally expressed inside parenthesis · The inner query or sub query is executed first
Sub queries used in several situations:

Rules of Nested Queries:-

1.The result of Inner Query is used in execution of Outer Query.

2.A subquery must always appear within pair of parentheses.

3.A sub-query must return only one column with multiple rows, that means you cannot use

“ select * ” in sub-query,but main query contain multiple columns with multiple rows.

4.You can use IN or not IN along with sub-query.

5.Sub-Query can be used with select,update,delete,insert statement along with operators like > , < , >= , <= , = , IN , BETWEEN.

Syntax for nested query:

select column_list from table_name where column_name operator (select column_name from table_name where condition);

Where sub queries:

The most common type of sub query used on inner select query on the right side of a where clause comparison expression.

Select ename, sal from emp where sal > (select avg(Sal) from emp);

This displays all rows from emp table whose salary is greater than the average salary If a sub query is used >,<,>=,<= in the condition.
Then sub query returns only single value.

In sub queries:

When you want to compare single attributewith the list of value we can use in sub queries inside.
For ex: Select * from emp where sal in (select max (sal) from emp group by dept no);
This displays all rows of employee whose salary is maximum in their departments.

Having sub query:

An sub queries used in the where clause, we can use a sub queries with a havingclause, Having clause is used to restrict the rows by using the condition in the group by query. For ex:Select dept no from emp group by deptno having avg (sal)> (select avg (sal) from emp);
It displays department whose avg salary is greater than the total avg salary.

From sub query:

As you already know, in the from clause, this specifies table name from which the data can be drawn; because the output of the select statement is another table.

we Can use a select queries in the from clause. Consider the query:

Select ename, sat from (select * from emp);

13. MySQL Operators:

To retrieve data from the table based on some rules, then use conditional operators. Conditional operators return true or false esteem based on the instance of the variables.

The conditional operators in MySQL are classified into following types:

1.Comparison operators: comparison operators are = ,>,<,>=,<=,;

Consider the select query by using comparison operators

Select * from employee where salary >= 3000;

It displays all rows, all column of employee's whose salary is grater than or

equal to 3000.

2. Arithmetic operators:

Arithmetic operators are +,-,*,/ we can use the arithmetic operators with the table attribute in a conditional expression. Consider the select query by using arithmetic operator

select salary +1000 from employee;

3. Logical Operators (Boolean):

The logical operators are AND, OR, NOT, are used in between two or more conditional expression.

3.1 To specify multiple conditions in the column of MySQL table use Logical AND operator. i.e. MySQL Logical And operator compares two results and returns true if both of the results are true.

Syntax: Select * from <table_name> where <condition1> and <condition2> and <condition3>;

Examples: mysql> select * from employee;

mysql>select * from employee where ename='murali' and job='manager';

3.2 To specify multiple conditions on different columns in MySQL table use Logical OR operator. MySQL Logical OR operator compares two expressions and returns true if both of the expressions is valid.

Syntax : Select * from <table_name> where <condition1> or <condition2> or <condition3>;

Examples: mysql> select * from employee where ename='rambo' or sal>13000;

3.3 MySQL NOT EQUAL operator is used to return a set of rows from the table, after making sure that two expressions placed on either sides of the table are NOT NULL.

Syntax: Select * from <table_name> where <column_name> is not equal to;

Examples: mysql> select * from employee where job'business';

The above example tells that, when Not Equal operation is performed on a column name job<>business then it automatically display remaining employee who are working in another sectors.

4. Special operations:

4.1Between: MySQL BETWEEN AND operator checks whether a value is present between minimum and maximum range of an expression.

Syntax: Select * from <table_name> where <column_name> between minimum and maximum;

Examples: mysql> select * from employee where salary> 13000 and salary<14500;

The above example tells that, when between and operation is performed on a column name salary, then it automatically shows the salary, according to the condition and gives the output.

4.2.IS NULL: MySQL IS NULL operator will review whether a esteem is NULL.

Syntax : Select * from <table_name> where <column_name> is null;

Examples: mysql> select * from employee where commission is null;

This operator is used to checks for the null.

4.3.IS NOT NULL operator will review whether the esteem **IS Not Null** or not.

Syntax : select * from <table_name> where <column_name> is not null.

Examples: mysql> select * from employee where commission is not null;

4.3.Like: The Like operation is used with wildcard characters [%, -]
To find out the patterns within the string attributes % it means any and all following or preceding characters.

_ (Under score) it means any single characters

SQL allows case – sensitive searches with like operation Ex: select * from employee where ename like ' A%';
· It displays all rows of employees whose name start with A.

4.4.In: This operator is used to check whether all attribute values match any value with in a value list. The values in the list are all of same data types

Ex: select * from employee where salary in (10000,20000);
It displays all rows of employees whose salary is exactly 10000 and 20000

14. MySQL Views

MySQL Views are the database objects that can be created on a table to improve the performance of MySQL Server. A view has unique Constraints look on data from one or more tables. It can organize data in unique order, focus or hide some data.

MySQL Views comprises of a stored query accessible as a fundamental table composed of the outcome set. Beside standard tables a perspective does not shaped a part of the physical schema. It is a virtual table, dynamic in the database.

View is a stored query, and it can be attributed like a table.

MySQL Views object is mainly classified into

- Create view
- Alter view
- Drop view

Description: Once a view is created from base table, all DML operations can be performed on that view which effects its base table. This kind of view is called simple view. Create any number of columns in a table. The system privileges are required to create own schema, and can execute any object privilege on object type.

1. **Syntax: create view <table_name>as select <column_name> FROM <table_name> where <condition>;**

   mysql> create view vehicles as select name from cars;

2. **Syntax: Alter view <table_name> as select <column_name> from <table_name> where condition;**

   mysql> alter view cheapcars as select car_name from cars where cost select * from cheapcars;

   mysql> alter view cheapcars as rangeovers;

   mysql> select * from rangeovers;

3. **Syntax: Drop table <table_name>**

   mysql> drop table cars;

15. SQL FUNCTIONS:

➔All SQL functions are inbuilt functions.

➔These are classified as two types:1. Single row function 2. Multiple row function

1.SINGLE ROW FUNCTION:

➔There are the one who works on the single row and return one output for row.

EX: Conversion Function,Character Function (or) String Function,Numeric Function.

Conversion Function:

upper( ): This function convert a string to Uppercase.

Syn: select upper(string);

Eg: select upper("dbms") ;

O/p: DBMS

lower( ): This function convert a string to lowercase.

Syn: select lower(string);

Eg: select lower("Dbms") ; O/p: dbms

String Functions:-

These are accept character as input and return number or character value.

1. **concat():-** This function is used to combine two strings.

Syn: select concat(string1,string2);

Eg: select concat("cse","world");

O/p: cse world

2. **strcmp( ):-**This function is used to compare two strings.

Syn: select concat(string1,string2); ***Eg:***select strcmp("man" , "mom"); **O/p:** 1

3. **length( ):**This function is used to count the length of the string.

Ex: select length("cse"); **O/P:** length("cse") 3

4. substr( ):This function is used to return a portion of string from given start point to end pount.

Ex: select substr(“world”,2);

O/p: substr(“world”,2)

orld

5. instr( ): This fuction is used to return a numeric position of a character (or) string.

Ex: select instr(“world”, “l”)

O/P: instr(“world”, “l”);

4

6. lpad( ): This fuction is used to inert the symbol with the actual length of the string from left side.

Ex: select lpad(“world”,10, “*”);

O/P: lpad(“world”,10, “*”)

*****world

7. rpad( ): This function is used to insert the symbol with the actual length of the string from rpad side.

Ex: select rpad(“world”,10, “*”);

rpad(“world”,10, “*”) world*****

8. ltrim( ): This function is used to remove leading spaces in a given string from leftside.

Ex: select ltrim(“   world”);

O/P: world

9. rtrim( ): This function is used to remove leading spaces in a given string from rightside.

Ex: select ltrim(“world   ”);

O/P: world

3.NUMERIC FUNCTIONS:

1. **truncate( ):**

Ex: select trunc(28.7) #removes decimal part. O/P: 28

2. **round( ):**

   Ex: select round(27.6)

   O/P: 28

3. **mod( ):**

   Ex: select mod(27,6)

   O/P: 3 #remainder

4. **least( ):**

   Ex: select least(-27.5,-28.5) O/P: -28.5

5. **greatest( ):**

   Ex: select greatest(-27.5,-28.5) O/P:-27.5

6. **sqrt( ):**

   Ex: select sqtr(25) O/P: 5

7. **ceil( ):**

   Ex: select ceil(27.2)

   O/P: 28

   select ceil(-27.2) O/P: -27

8. **floor( ):**

   Ex: select floor(27.2) O/P: 27

   select floor(-27.2) O/P: -28

9. **power( ):**

   Ex: select power(8,2) O/P: 64

2. MULTIPLE ROW FUNCTIONS:

- ➢ These are works upon group of rows and return one result for the complete set of rows.
- ➢ These are also called as “group function” (or) “aggregate fuctions”.
- ➢ The following are the aggregate functions:

(a) sum( )

(b) avg( )

(c) count( )

(d) min( )

(e) max( )

(f) first( )

(h) last( )

(a) sum( ): This function is used to get the sum of numeric column.

Syntax: select sum(column_name) from table_name;

EX: select sum(salary) from employee;

O/p:

sum(salary)

(b) avg( ): This function is used to get the average of numeric column.

Syntax: select avg(column_name) from table_name;

EX: select avg(salary) from employee;

O/p:

avg(salary)

(c) count( ): This function is used to get the no.of rows in table.

Syntax: select count(*) from table_name;

EX: select count(*) from employee;

O/p:5

➔This fuction is also allows the where condition;

Ex: select count(*) from employee where name= “a”;

O/P: 1

(d) **min( ):** This function is used to get the minimum value from a column.

Syntax: select min(column_name) from table_name;

EX: select min(salary) from employee;

O/p: 30000

(e) **max( ):** This function is used to get the maximum value from a column.

Syntax: select max(column_name) from table_name;

EX: select max(salary) from employee;

O/p: 35000

(d) **first( ):** This function is used to get the first value of selected column.

Syntax: select column_name from table_name limit 1;

EX: select name from employee limit 1;

O/p: a

(e) **last( ):** This function is used to get the last value of selected column.

Syntax: select column_name from table_name order by column_name desc limit 1;

EX:select name from employee order by name desc limit 1;

O/p: e

16. NULL VALUES IN SQL:

➔The SQL NULL is used to represent a Missing value.

➔A NULL value in a table is value in a column that appears to be blank.

➔ A column with NULL value is “A Column with no value”.It is very important to understand that a NULL valur is different than 0 value (or) column contains spaces.

➔In general,each NULL value is different from every other NULL value in database.

IMPORTANCE OF NULL VALUE:

NULL values are **Not applicable:** Which means when a value doesn’t exist for an entity.

Ex:Some of the students are not contain middle_name.

(i) Unknown: Missing: Which means that value exist but unknown.

Ex:Just know the names of your friend don’t know the middle_name or last_name.

(ii) Not Known: Which means that no information about the existence.

➔We check NULL value by using IS NULL (or) IS NOT NULL operators.

➔ For example, create table student(sid int NOT NULL,first_name varchar(10),middle_name varchar(10),last_name varchar(10),marks int);

➔ In above example, NOT NULL specifies that column should always accept value of given datatype.There are one column that contains NOT NULL values, that is sid and remaining 4 column first_name,middle_name,last_name contains NULL values.

IS NOT NULL OPERATOR:

sid	first_name	middle_name	last_name	marks
1	a	b	c	70
2	d	e	f	75
3	g	h	i	NULL
4	NULL	j	k	78
5	l	NULL	m	80
6	n	o	p	85
7	NULL	NULL	q	90
8	r	s	t	NULL

9	NULL	NULL	NULL	95

Select sid ,first_name,middle_name,last_name,marks from student where marks IS NOT NULL;

OUTPUT:

sid	first_name	middle_name	last_name	marks
1	a	b	c	70
2	d	e	f	75
4	NULL	j	k	78
5	l	NULL	m	80
6	n	o	p	85
7	NULL	NULL	q	90
9	NULL	NULL	NULL	95

IS NULL OPERATOR:

select sid ,first_name,middle_name,last_name,marks from student where marks IS NULL;

OUTPUT:

sid	first_name	middle_name	last_name	marks
3	g	h	i	NULL
8	r	s	t	NULL

Replace NULL Values:-

There are different ways to replace NULL values. (a)IF NULL function

(b)case statement (c)COALESCE function

(a) *IF NULL function:*

select first_name, middle_name, last_name, IFNULL(marks,0) as marks from student;

OUTPUT:

sid	first_name	middle_name	last_name	marks
1	a	b	c	70
2	d	e	f	75
3	g	h	i	0
4	NULL	j	k	78
5	l	NULL	m	80
6	n	o	p	85
7	NULL	NULL	q	90
8	r	s	t	0
9	NULL	NULL	NULL	95

(b) case statement:

select first_name,last_name,case when marks IS NULL then 0 else marks end as marks from student;

OUTPUT:

first_name	last_name	marks
a	c	70
d	f	75
g	i	0
NULL	k	78
l	m	80
n	p	85
NULL	q	90
r	t	0
NULL	NULL	95

(c) ***COALESCE function:***

select sid,COALESCE(first_name,middle_name,last_name) as name,marks from student;

(OR)

select sid,COALESCE(first_name , middle_name , last_name , 'no name') as name,marks from student;

OUTPUT:

sid	name	marks
1	a	70
2	d	75
3	g	0
4	NULL	78
5	l	80
6	n	85
7	NULL	90
8	r	0
9	NULL	95

17.KEYS IN SQL:

➔ A key can be a single attribute or group of attributes,where combination may act as key.

➔Keys are plays major role in Relational_Database(RD).

Different types of keys: (1) Super key (5)Surrogate key

(2)Candidate key (6)Primary key

(3)Composite key (7)Foreign key.

(4)Secondary key

sid	name	phonenumber	age
1	a	9123456780	20
2	b	9876543210	21
3	a	8123467890	20
4	a	7123456890	21
5	c	3456788990	20

1.Super Key:

➔ A set of attributes within a table that can be uniquely identified each record within a table.

➔Super key is superset of candidate key.

➔In above table,

{sid},{sid,name},{phonenumber},

{sid,age,name},{name,phonenumber}...etc all are keys.

➔Here,sid is unique for every row of data. Hence, it can be used as identify each row uniquely.

➔{sid,name},here name of two students can be same but sid's are cannot be same.hence,this combination acts as a key.

➔ At the same time phonenumber for every student will be unique.hence,again phonenumber can be a key.

2.Candidate Key:

➔The minimal set of attributes which can be uniquely identified in a table.

➔It is an attribute or set of attributes that can be as primary key for a table to uniquely identify each record in a table.

➔They can be morethan one candidate keys in a table.

➔In above table,sid,phonenumber both are candidate keys for the student table.

➔A candidate key can never be null or empty and its value should be unique.

➔There can be morethan one candidate keys in a table.

3.Composite Key:

➔ If any single attribute of a table is not capable to being a key i.e., it can not identify each record uniquely.So,we combine two or more attributes to form a key is known as "Composite key".

4.Secondary Key:

➔ The candidate key which is not selected as primary key is known as "secondary key" (or) "alternate key".

5.Surrogate Key:

A key which can be unique in nature,not null and upadatable is called "Surrogate Key".

Ex:phone_number.

6.Primary Key:

➔Primary key contains unique values and never contains new values.

➔It is unique column in a table.

➔ A table can have only one primary key which consists of one or more columns.

7.Foreign Key:

➔It means it links two different tables together and column in one table that can be pointing to primary key in another table.

➔They act as cross reference between tables.

18. **Explain about JOINS in SQL?**

Join is used to retrieve the data from more than one table in a single query

There are following joins are available in SQL:-

1) Equi join.
2) Cartesian join[cross join].
3) Non-equi join.
4) Outer join.
5) Self join.

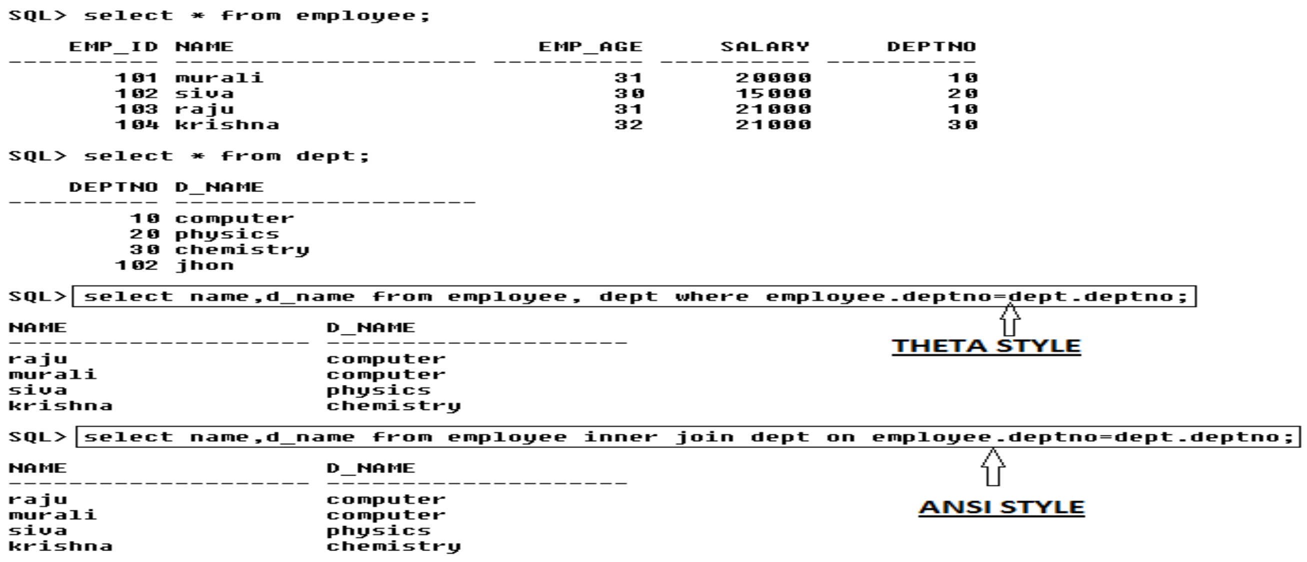
1) Equi join:-

✓ It is used to retrieve the data from more than 1 table based on "equality" condition.
✓ Tables must have common column between them to apply this join.

- ✓ If N tables are joined N-1 conditions are applied.

MYSQL> select name,d_name from emp,dept where emp.dno=dept.dno;
OR
MYSQL> select name,d_name from emp inner join dept on emp.dno=dept.dno;

Example:-

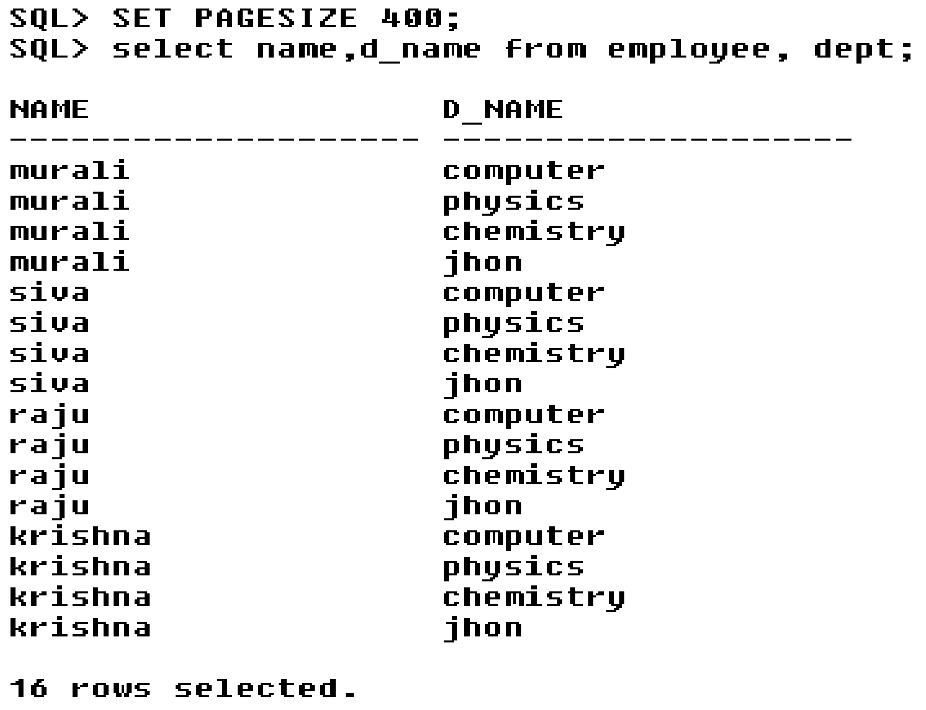


```
SQL> select * from employee;

    EMP_ID NAME                    EMP_AGE     SALARY     DEPTNO
---------- -------------------- ---------- ---------- ----------
       101 murali                       31      20000         10
       102 siva                         30      15000         20
       103 raju                         31      21000         10
       104 krishna                      32      21000         30

SQL> select * from dept;

    DEPTNO D_NAME
---------- --------------------
        10 computer
        20 physics
        30 chemistry
       102 jhon

SQL> select name,d_name from employee, dept where employee.deptno=dept.deptno;

NAME                 D_NAME
-------------------- --------------------
raju                 computer
murali               computer
siva                 physics
krishna              chemistry

SQL> select name,d_name from employee inner join dept on employee.deptno=dept.deptno;

NAME                 D_NAME
-------------------- --------------------
raju                 computer
murali               computer
siva                 physics
krishna              chemistry
```

2) <u>Cartesian join: -[cross join]</u>

- ✓ It is used to retrieve the data from the multiple tables without any conditions.
- ✓ It is used to retrieve data analysis report.
- ✓ No need to have common column between tables to apply this type of join.

<u>Example:-</u>

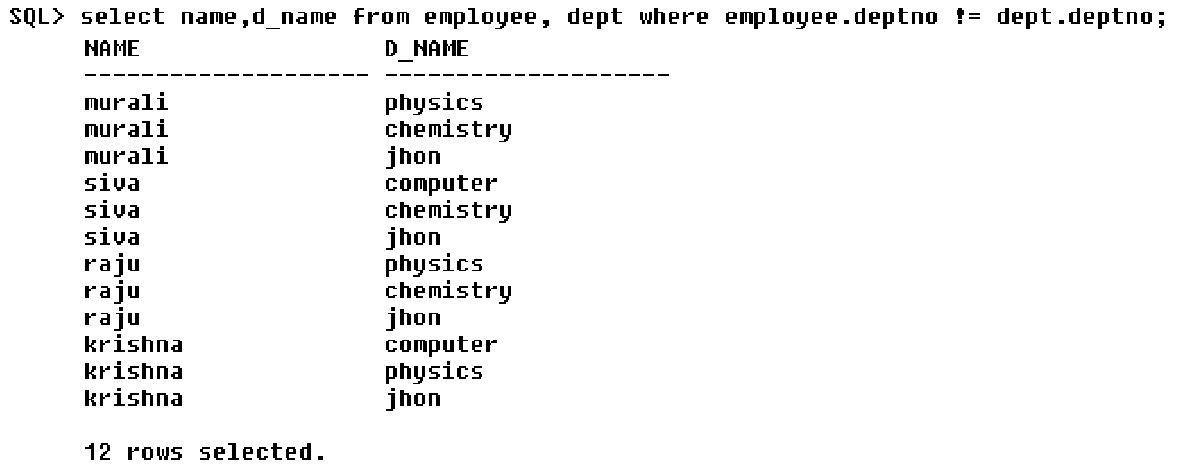


```
SQL> SET PAGESIZE 400;
SQL> select name,d_name from employee, dept;

NAME                 D_NAME
-------------------- --------------------
murali               computer
murali               physics
murali               chemistry
murali               jhon
siva                 computer
siva                 physics
siva                 chemistry
siva                 jhon
raju                 computer
raju                 physics
raju                 chemistry
raju                 jhon
krishna              computer
krishna              physics
krishna              chemistry
krishna              jhon

16 rows selected.
```

3) <u>Non-Equi join:-</u>

- ✓ It is used to retrieve the data from multiple tables based on other condition but not equal.
- ✓ Non Equi Join uses all comparison operators except the equal (=) operator like !=, >=, <=, <, >.

MYSQL>SELECT NAME,D_NAME FROM EMPLOYEE, DEPT WHERE EMPLOYEE.DEPTNO <> DEPT.DEPTNO;



```
SQL> select name,d_name from employee, dept where employee.deptno != dept.deptno;
     NAME                 D_NAME
     -------------------- --------------------
     murali               physics
     murali               chemistry
     murali               jhon
     siva                 computer
     siva                 chemistry
     siva                 jhon
     raju                 physics
     raju                 chemistry
     raju                 jhon
     krishna              computer
     krishna              physics
     krishna              jhon

     12 rows selected.
```

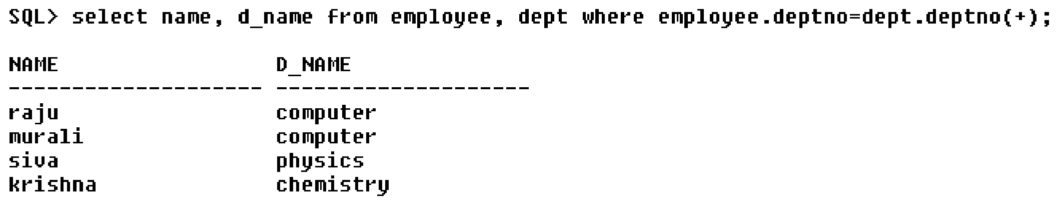
4) Outer Join – This join returns all the rows from one table and only those rows from second table which meets the condition.

Outer join is classified into 3 types:

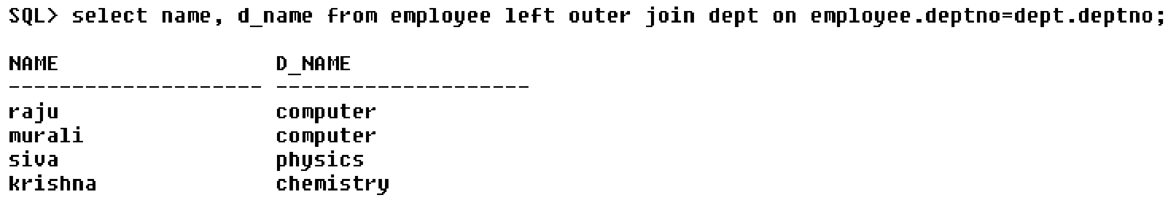
a. Left Outer Join
b. Right Outer Join
c. Full Outer Join

a. **Left Outer Join:**–

✓ Returns all the rows from left table(ie, first table) and rows that meet the condition from second table.
✓ All the rows from left table and only matching rows from the right table.



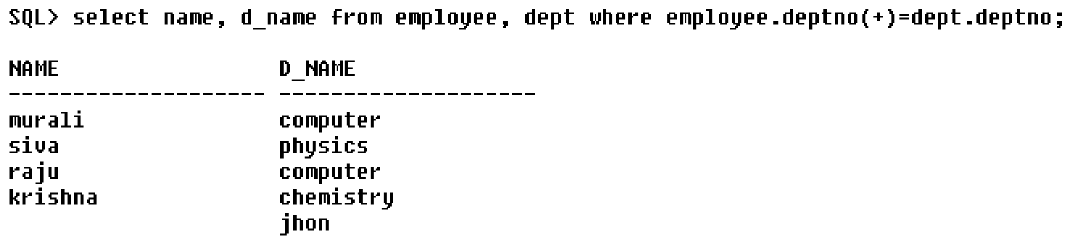
```
SQL> select name, d_name from employee, dept where employee.deptno=dept.deptno(+);

NAME                 D_NAME
-------------------- --------------------
raju                 computer
murali               computer
siva                 physics
krishna              chemistry
```



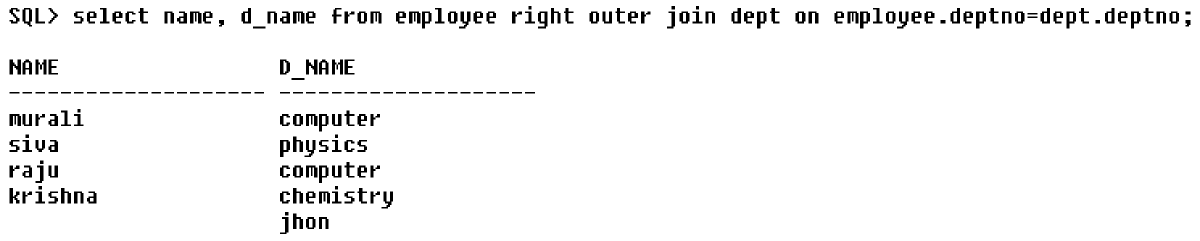
```
SQL> select name, d_name from employee left outer join dept on employee.deptno=dept.deptno;

NAME                 D_NAME
-------------------- --------------------
raju                 computer
murali               computer
siva                 physics
krishna              chemistry
```

b).Right Outer Join:-

✓ Returns all the rows from right table(ie, second table) and rows that meet the condition from first table.
✓ All rows from right table and only matching rows from the left table.



```
SQL> select name, d_name from employee, dept where employee.deptno(+)=dept.deptno;

NAME                 D_NAME
-------------------- --------------------
murali               computer
siva                 physics
raju                 computer
krishna              chemistry
                     jhon
```



```
SQL> select name, d_name from employee right outer join dept on employee.deptno=dept.deptno;

NAME                 D_NAME
-------------------- --------------------
murali               computer
siva                 physics
raju                 computer
krishna              chemistry
                     jhon
```

<u>c). Full Outer Join :–</u>

✓ Combines the results of both left and right outer joins.

✓ Non-matching records from both the tables will be left blank.

```
SQL> select name, d_name from employee full outer join dept on employee.deptno=dept.deptno;

NAME                 D_NAME
-------------------- --------------------
raju                 computer
murali               computer
siva                 physics
krishna              chemistry
                     jhon
```

5) <u>Self join</u> :-

- ✓ Here the table is joined (compared) to itself.
- ✓ The output shows the low salary employee names, high salary employee names from the employee table with itself.

```
SQL> select * from employee;

    EMP_ID NAME                    EMP_AGE     SALARY     DEPTNO
---------- -------------------- ---------- ---------- ----------
       101 murali                       31      20000         10
       102 siva                         30      15000         20
       103 raju                         31      21000         10
       104 krishna                      32      21000         30

SQL> select a.name low_salary_emp, b.name high_salary_emp from employee a, employee b where a.salary
<b.salary;

LOW_SALARY_EMP       HIGH_SALARY_EMP
-------------------- --------------------
murali               raju
murali               krishna
siva                 murali
siva                 raju
siva                 krishna
```

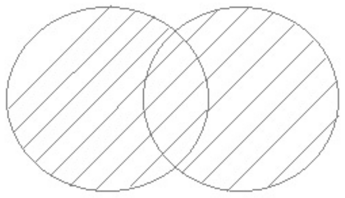
19. Explain about set operations/Relational set operators.

SQL SET Operators are behaving same like mathematical sets, these sql set operators are classified into four types, which is given below;

- ❖ **Union**
- ❖ **Union all**
- ❖ **Minus**
- ❖ **Intersect**

<u>Union:-</u>

Union operator returns all the value from the entire table. But it do not include duplicate values, it means it not return duplicate values.

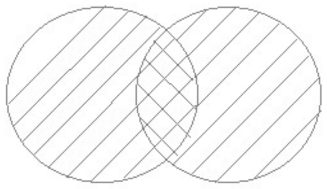


Example

```
SQL> select * from first;

        ID NAME
---------- --------------------
         1 murali
         2 raju

SQL> select * from second;

        ID NAME
---------- --------------------
         2 raju
         3 vamsi

SQL> select * from first union select * from second;

        ID NAME
---------- --------------------
         1 murali
         2 raju
         3 vamsi
```

Union all

Union all operator returns all the value from the entire table including duplicate values.

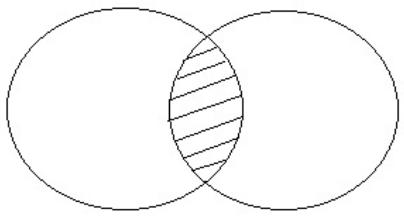


Example

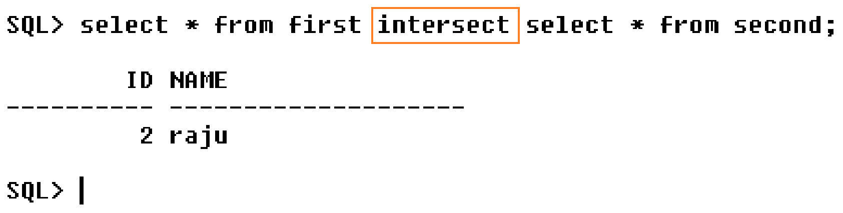
```
SQL> select * from first union all select * from second;

        ID NAME
---------- --------------------
         1 murali
         2 raju
         2 raju
         3 vamsi

SQL>
```

Intersect:-

Intersect operator return the common value from all the variables.



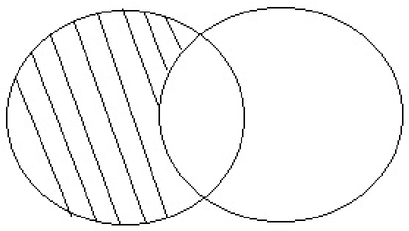
Example



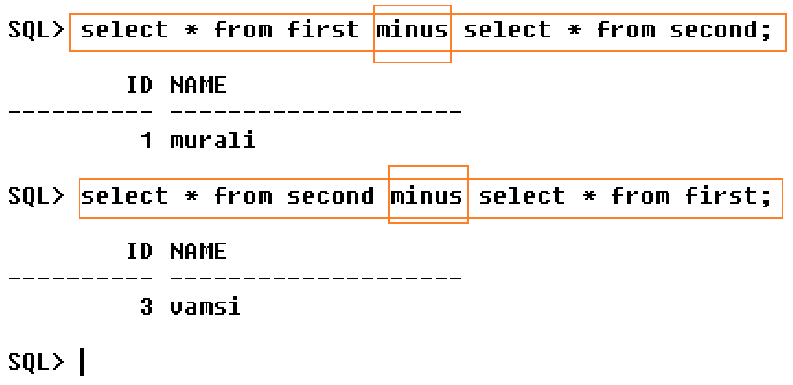
```
SQL> select * from first intersect select * from second;

        ID NAME
---------- --------------------
         2 raju

SQL> |
```

Minus

Minus operator return the values which are not available in first table but not available in second table.



Example



```
SQL> select * from first minus select * from second;

        ID NAME
---------- --------------------
         1 murali

SQL> select * from second minus select * from first;

        ID NAME
---------- --------------------
         3 vamsi

SQL> |
```

20.Date and Time in MySQL :

The DATE, DATETIME, and TIMESTAMP types are related. This section describes their characteristics, how they are similar, and how they differ.

The DATE type is used for values with a date part but no time part. MySQL retrieves and displays DATE values in 'YYYY-MM-DD' format. The supported range is '1000-01-01' to '9999-12-31'.

The DATETIME type is used for values that contain both date and time parts. MySQL retrieves and displays DATETIME values in 'YYYY-MM-DD HH:MM:SS' format. The supported range is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. The TIMESTAMP data type is used for values that contain both date and time parts. TIMESTAMP has a range of '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.

S. No.	Name & Description
1	ADDDATE() Adds dates
2	ADDTIME() Adds time
3	CONVERT_TZ() Converts from one timezone to another
4	CURDATE() Returns the current date
5	CURRENT_DATE(), CURRENT_DATE Synonyms for CURDATE()
6	CURRENT_TIME(), CURRENT_TIME Synonyms for CURTIME()
7	CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP Synonyms for NOW()
8	CURTIME() Returns the current time
9	DATE_ADD( ) Adds two dates
10	DATE_FORMAT() Formats date as specified
11	DATE_SUB() Subtracts two dates
12	DATE() Extracts the date part of a date or datetime expression
13	DATEDIFF() Subtracts two dates
14	DAY() Synonym for DAYOFMONTH()
15	DAYNAME() Returns the name of the weekday
16	DAYOFMONTH() Returns the day of the month (1-31)
17	DAYOFWEEK() Returns the weekday index of the argument
18	DAYOFYEAR() Returns the day of the year (1-366)
19	EXTRACT Extracts part of a date
20	FROM_DAYS() Converts a day number to a date
21	FROM_UNIXTIME() Formats date as a UNIX timestamp
22	HOUR() Extracts the hour
23	LAST_DAY Returns the last day of the month for the argument
24	LOCALTIME(), LOCALTIME Synonym for NOW()
25	LOCALTIMESTAMP, LOCALTIMESTAMP() Synonym for NOW()
26	MAKEDATE() Creates a date from the year and day of year
27	MAKETIME MAKETIME()
28	MICROSECOND() Returns the microseconds from argument
29	MINUTE() Returns the minute from the argument
30	MONTH() Returns the month from the date passed