SQL COMMANDS

- ✓ **Show Databases Command -:** The most common way to get a list of the MySQL databases is by using the MySQL client to connect to the MySQL server and run the SHOW DATABASES command.

Syntax -: show databases;
Example :-

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

mysql>
```

- ✓ **Create Databases -:**
  Example :-

```
mysql> create database first;
Query OK, 1 row affected (0.24 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| first              |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.04 sec)

mysql>
```

- ✓ **USE -:** The USE statement tells MySQL to use the named database as the default (current) database for subsequent statements. This statement requires some privilege for the database or some object within it.

The named database remains the default until the end of the session or another USE statement is issued:
Syntax -:
USE db_name;
Example :-

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| first              |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.04 sec)

mysql> use first;
Database changed
mysql>
```

- ✓ **Create Table -:** A table is basic unit of storage. It is composed of rows and columns. To create a table we will name the table and the columns of the table.
  It must begin with a letter and can be up to 30 characters long.
  It must not be duplicate and not any reserved word.

Syntax to create a table is -:

CREATE TABLE tablename (column_name1 datatype (size), column_name2 datatype (size) …);

Example is

CREATE TABLE STUDENT(Roll_no int,Name varchar(15),AGE smallint);

```
mysql> CREATE TABLE STUDENT(Roll_no int,Name varchar(15),AGE smallint);
Query OK, 0 rows affected (1.90 sec)

mysql> show tables;
+-----------------+
| Tables_in_first |
+-----------------+
| student         |
+-----------------+
1 row in set (0.20 sec)

mysql>
```

- ✓ **Describe Statement -:** As the name suggests, DESCRIBE is used to describe something. Since in database we have tables, that's why we use DESCRIBE or DESC(both are same) command to describe the structure of a table.

Syntax:

DESCRIBE one;

OR

DESC one;

Example :

```
mysql> describe student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | YES  |     | NULL    |       |
| name   | varchar(10) | YES  |     | NULL    |       |
| class  | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql>
```

- ✓ **The Insertion Of Data Into Table :-** Once a table is created, the most natural thing to do is load this with data to be manipulated later i.e. to insert the rows in a table. The data in a table can be inserted in three ways.

Syntax:-

INSERT INTO <table name >(<columnname1>,<columnname2>) VALUES(<expression1>,<expression 2>);

OR

INSERT INTO <tablename>VALUES(<expression1 >,<expression2>);

OR

INSERT INTO <tablename> VALUES('<&columnname1>' ,'<&columnname2>');

Example :-

```
mysql> insert into student values(1,'ajay','bca');
Query OK, 1 row affected (0.10 sec)

mysql> insert into student values(1,'ajay','mca');
Query OK, 1 row affected (0.18 sec)
```

- ✓ **Select Query -:** SELECT QUERY is used to fetch the data from the MySQL database. Databases store data for later retrieval. The purpose of MySQL Select is to return from the database tables, one or more rows that match a given criteria. Select query can be used in scripting language like PHP, Ruby, or you can execute it via the command prompt.

Syntax:-

Select colname1,colname2,colname3,……. From tablename;

Select * from tablename;

Example :-

```
mysql> select rollno from student;
+--------+
| rollno |
+--------+
|      1 |
|      1 |
|      3 |
|      2 |
+--------+
4 rows in set (0.04 sec)

mysql> select * from student;
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|      1 | ajay  | bca   |
|      1 | ajay  | mca   |
|      3 | ajay  | NULL  |
|      2 | vijay | NULL  |
+--------+-------+-------+
4 rows in set (0.00 sec)
```

- ✓ **Where Clause -:** We have seen the SQL SELECT command to fetch data from a MySQL table. We can use a conditional clause called the WHERE Clause to filter out the results. Using this WHERE clause, we can specify a selection criteria to select the required records from a table.

Syntax:-

Select colname1,colname2,colname3,……. From tablename where condition;

Example :-

```
mysql> select * from student where rollno=1;
+--------+------+-------+
| rollno | name | class |
+--------+------+-------+
|      1 | ajay | bca   |
|      1 | ajay | mca   |
+--------+------+-------+
2 rows in set (0.03 sec)
```

Comparison Operators

Operator	Description	Example
=	Checks if the values of the two operands are equal or not, if yes, then the condition becomes true.	(A = B) is not true.
!=	Checks if the values of the two operands are equal or not, if the values are not equal then the condition becomes true.	(A != B) is true.
>	Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of the left operand is less than the value of the	(A < B) is true.

	right operand, if yes then the condition becomes true.	
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true.	(A <= B) is true.

- ✓ **Order By Clause** -: The Order By Keyword Is Used To Sort The Result-Set In Ascending Or Descending Order.

The Order By Keyword Sorts The Records In Ascending Order By Default. To Sort The Records In Descending Order, Use The Desc Keyword.

Syntax:-
SELECT column1, column2, ... FROM table_name ORDER BY column1, column2, ... ASC|DESC;
Example :-

```
mysql> select * from student ORDER BY rollno;
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|      1 | ajay  | bca   |
|      1 | ajay  | mca   |
|      2 | vijay | NULL  |
|      3 | ajay  | NULL  |
+--------+-------+-------+
4 rows in set (0.00 sec)

mysql> select * from student ORDER BY rollno DESC;
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|      3 | ajay  | NULL  |
|      2 | vijay | NULL  |
|      1 | ajay  | bca   |
|      1 | ajay  | mca   |
+--------+-------+-------+
4 rows in set (0.00 sec)
```

Example :-

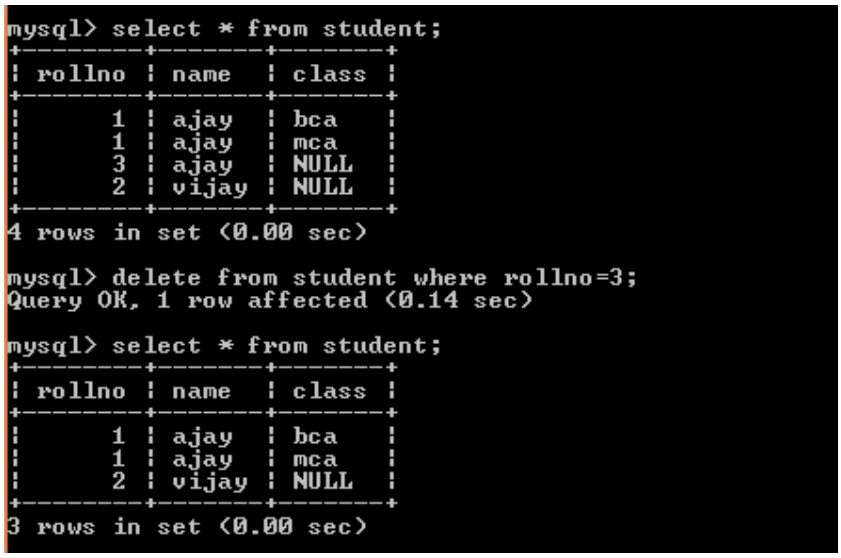
```
mysql> select * from student WHERE ROLLNO=1 ORDER BY rollno DESC,CLASS;
+--------+------+-------+
| rollno | name | class |
+--------+------+-------+
|      1 | ajay | bca   |
|      1 | ajay | mca   |
+--------+------+-------+
2 rows in set (0.03 sec)
```

- ✓ **DELETE -:** The DELETE statement is used to delete existing records in a table.

Syntax:-
DELETE FROM table_name WHERE condition;
Example :-
Delete from student;
All data delete.

```
mysql> delete from student;
Query OK, 3 rows affected (0.13 sec)

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

Example :-



```
mysql> select * from student;
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|      1 | ajay  | bca   |
|      1 | ajay  | mca   |
|      3 | ajay  | NULL  |
|      2 | vijay | NULL  |
+--------+-------+-------+
4 rows in set (0.00 sec)

mysql> delete from student where rollno=3;
Query OK, 1 row affected (0.14 sec)

mysql> select * from student;
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|      1 | ajay  | bca   |
|      1 | ajay  | mca   |
|      2 | vijay | NULL  |
+--------+-------+-------+
3 rows in set (0.00 sec)
```

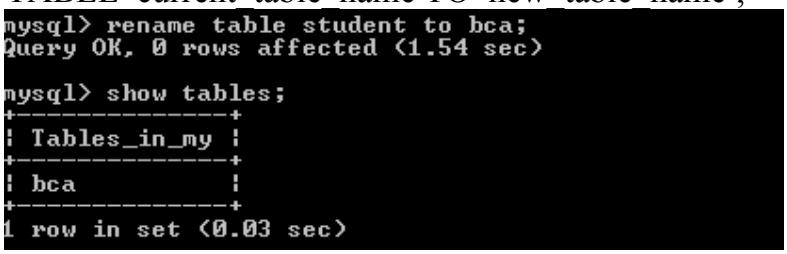
- ✓ **Rename Command -:** The rename command is used to change the name of an existing database object(like Table,Column) to a new name.

  Renaming a table does not make it to lose any data is contained within it.

Syntax:-

RENAME TABLE current_table_name TO new_table_name ;

Example :



```
mysql> rename table student to bca;
Query OK, 0 rows affected (1.54 sec)

mysql> show tables;
+--------------+
| Tables_in_my |
+--------------+
| bca          |
+--------------+
1 row in set (0.03 sec)
```

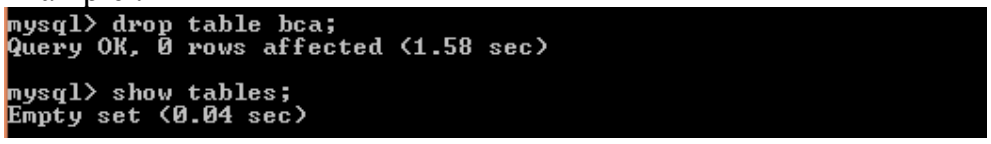
- ✓ **Drop Table Statement -:** The DROP TABLE statement is used to drop an existing table in a database.

Note: Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!

Syntax:-

Drop Table Name;

Example :-



```
mysql> drop table bca;
Query OK, 0 rows affected (1.58 sec)

mysql> show tables;
Empty set (0.04 sec)
```

- ✓ **Truncate Table -:** The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

Syntax:-

TRUNCATE TABLE table_name;

Example :-

```
mysql> select * from student;
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|      2 | vijay | bca   |
|      1 | ajay  | bca   |
|      3 | jay   | bca   |
+--------+-------+-------+
3 rows in set (0.00 sec)

mysql> TRUNCATE TABLE student;
Query OK, 0 rows affected (1.88 sec)

mysql> select * from student;
Empty set (0.10 sec)
```

- ✓ **Update Statement -:** The UPDATE statement is used to modify the existing records in a table.

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

Syntax:-
UPDATE table_name SET column1 =new value1, column2 = newvalue2, ... WHERE condition;
Example :-

```
mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | bca   |
|      2 | raj    | bca   |
|      3 | ajay   | bca   |
|      4 | vijay  | bca   |
+--------+--------+-------+
4 rows in set (0.00 sec)

mysql> update student set name='nisha';
Query OK, 4 rows affected (0.11 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> select * from student;
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|      1 | nisha | bca   |
|      2 | nisha | bca   |
|      3 | nisha | bca   |
|      4 | nisha | bca   |
+--------+-------+-------+
4 rows in set (0.04 sec)
```

```
mysql> update student set name='vikash' where rollno=3;
Query OK, 1 row affected (0.15 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | nisha  | bca   |
|      2 | nisha  | bca   |
|      3 | vikash | bca   |
|      4 | nisha  | bca   |
+--------+--------+-------+
4 rows in set (0.02 sec)
```

- ✓ **Count() Function -:** The COUNT() function returns the number of rows that matches a specified criterion.

Syntax:-
SELECT COUNT(column_name) FROM table_name WHERE condition;
Example :-

```
mysql> select count(rollno) from student;
+---------------+
| count(rollno) |
+---------------+
|             4 |
+---------------+
1 row in set (0.00 sec)
```

Note -: group functions ignore null values.

Example :-

```
mysql> select count(rollno) from student;
+---------------+
| count(rollno) |
+---------------+
|             7 |
+---------------+
1 row in set (0.00 sec)

mysql> select count(name) from student;
+-------------+
| count(name) |
+-------------+
|           4 |
+-------------+
1 row in set (0.00 sec)

mysql> select count(*) from student;
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

- ✓ **Avg() Function** -: The AVG() function returns the average value of a numeric column.

Syntax -:
SELECT AVG(column_name) FROM table_name WHERE condition;
Example :-

```
mysql> select avg(rollno) from student;
+-------------+
| avg(rollno) |
+-------------+
|      2.5000 |
+-------------+
1 row in set (0.00 sec)
```

- ✓ **SUM() function -:** The SUM() function returns the total sum of a numeric column.

Syntax -:
SELECT SUM(column_name) FROM table_name WHERE condition;
Example :-

```
mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | nisha  | bca   |
|      2 | raj    | bca   |
|      3 | vikash | bca   |
|      4 | vijay  | bca   |
+--------+--------+-------+
4 rows in set (0.00 sec)

mysql> select sum(rollno) from student;
+-------------+
| sum(rollno) |
+-------------+
|          10 |
+-------------+
1 row in set (0.04 sec)

mysql> select sum(rollno) "total" from student;
+-------+
| total |
+-------+
|    10 |
+-------+
```

Arithmetic operator command

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo

Example -:

```
mysql> update student set rollno=rollno+2000;
Query OK, 7 rows affected (0.11 sec)
Rows matched: 7  Changed: 7  Warnings: 0

mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|   2001 | nisha  | bca   |
|   2002 | raj    | bca   |
|   2003 | vikash | bca   |
|   2004 | vijay  | bca   |
|   2005 | NULL   | NULL  |
|   2006 | NULL   | NULL  |
|   2007 | NULL   | NULL  |
+--------+--------+-------+
7 rows in set (0.04 sec)

mysql> select 100-40;
+--------+
| 100-40 |
+--------+
|     60 |
+--------+
1 row in set (0.00 sec)
```

```
mysql> select 100*40;
+--------+
| 100*40 |
+--------+
|   4000 |
+--------+
1 row in set (0.00 sec)

mysql> select 100/40;
+--------+
| 100/40 |
+--------+
| 2.5000 |
+--------+
1 row in set (0.00 sec)

mysql> select 100%40;
+--------+
| 100%40 |
+--------+
|     20 |
+--------+
1 row in set (0.00 sec)
```

Logical Operators

Operator	Description
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

Example -:

```
mysql> select * from student where name in ("vikash","nisha","vijay","ankush");
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|   2001 | nisha  | bca   |
|   2003 | vikash | bca   |
|   2004 | vijay  | bca   |
+--------+--------+-------+
3 rows in set (0.00 sec)
```

```
mysql> select * from student where name not in ("vikash","nisha","vijay","ankush
");
+--------+------+-------+
| rollno | name | class |
+--------+------+-------+
|   2002 | raj  | bca   |
+--------+------+-------+
1 row in set (0.00 sec)
```

```
mysql> select * from student where rollno>1 and name<>"vikash";
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|   2001 | nisha | bca   |
|   2002 | raj   | bca   |
|   2004 | vijay | bca   |
+--------+-------+-------+
3 rows in set (0.00 sec)

mysql> select * from student where rollno>1 or name<>"vikash";
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|   2001 | nisha  | bca   |
|   2002 | raj    | bca   |
|   2003 | vikash | bca   |
|   2004 | vijay  | bca   |
|   2005 | NULL   | NULL  |
|   2006 | NULL   | NULL  |
|   2007 | NULL   | NULL  |
+--------+--------+-------+
7 rows in set (0.00 sec)

mysql> select * from student where rollno>1 like name<>"vikash";
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|   2001 | nisha  | bca   |
|   2002 | raj    | bca   |
|   2003 | vikash | bca   |
|   2004 | vijay  | bca   |
+--------+--------+-------+
4 rows in set, 1 warning (0.04 sec)
```

```
mysql> select * from student where name like "______";
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|   2003 | vikash | bca   |
+--------+--------+-------+
1 row in set (0.04 sec)

mysql> select * from student where name like "a%";
Empty set (0.00 sec)

mysql> select * from student where name like "n%";
+--------+-------+-------+
| rollno | name  | class |
+--------+-------+-------+
|   2001 | nisha | bca   |
+--------+-------+-------+
1 row in set (0.00 sec)
```

- ✓ **Between Operator -:** The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

Syntax-:
SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;

Example -:

```
mysql> select * from student where rollno between 2002 and 4007;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|   2002 | raj    | bca   |
|   2003 | vikash | bca   |
|   2004 | vijay  | bca   |
|   2005 | NULL   | NULL  |
|   2006 | NULL   | NULL  |
|   2007 | NULL   | NULL  |
+--------+--------+-------+
6 rows in set (0.00 sec)
```

- ✓ **Is Operator –:** In databases, NULL is unknown, not applicable or missing information, therefore, you cannot use the comparison operators (=, >,<, etc.,) to check whether a value is NULL or not.

Syntax-:
SELECT * FROM table_name WHERE column_name is (not) null;
Example -:

```
mysql> select * from student where name is not null;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|   2001 | nisha  | bca   |
|   2002 | raj    | bca   |
|   2003 | vikash | bca   |
|   2004 | vijay  | bca   |
+--------+--------+-------+
4 rows in set (0.00 sec)

mysql> select * from student where name is null;
+--------+------+-------+
| rollno | name | class |
+--------+------+-------+
|   2005 | NULL | NULL  |
|   2006 | NULL | NULL  |
|   2007 | NULL | NULL  |
+--------+------+-------+
3 rows in set (0.00 sec)
```

- ✓ **Alter Table Statement -:** The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.
Keyword-: add,drop,modify.

Syntax-:
ALTER TABLE table_name ADD (column_name datatype, column_name datatype,….. column_name datatype);
Note -: by default store null value.
Add Example -:

```
mysql> alter table student add (fee int,farher_name varchar(10));
Query OK, 0 rows affected (1.86 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| rollno      | int         | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| fee         | int         | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.05 sec)

mysql> select * from student;
+--------+--------+-------+------+-------------+
| rollno | name   | class | fee  | farher_name |
+--------+--------+-------+------+-------------+
|   2001 | nisha  | bca   | NULL | NULL        |
|   2002 | raj    | bca   | NULL | NULL        |
|   2003 | vikash | bca   | NULL | NULL        |
|   2004 | vijay  | bca   | NULL | NULL        |
|   2005 | NULL   | NULL  | NULL | NULL        |
|   2006 | NULL   | NULL  | NULL | NULL        |
|   2007 | NULL   | NULL  | NULL | NULL        |
+--------+--------+-------+------+-------------+
7 rows in set (0.04 sec)
```

Drop Example -:

```
mysql> alter table student drop fee;
Query OK, 0 rows affected (3.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| rollno      | int         | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

Modify Example -:

```
mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| rollno      | int         | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> alter table student modify rollno varchar(10);
Query OK, 7 rows affected (2.97 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| rollno      | varchar(10) | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

- ✓ **Change Data Type -:**

Example -:

```
mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| rollno      | varchar(10) | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> alter table student change rollno roll_no int;
Query OK, 7 rows affected (3.63 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| roll_no     | int         | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

- ✓ **Alter Table - Add Column -:** To add a column in a table**.**

syntax:
ALTER TABLE table_name ADD column_name datatype;

```
mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| roll_no     | int         | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)

mysql> alter table student add section varchar(5) after class;
Query OK, 0 rows affected (2.65 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| roll_no     | int         | YES  |     | NULL    |       |
| name        | varchar(10) | YES  |     | NULL    |       |
| class       | varchar(10) | YES  |     | NULL    |       |
| section     | varchar(5)  | YES  |     | NULL    |       |
| farher_name | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.04 sec)
```

Example -:

- ✓ **Group By Statement -:** The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Syntax-:
SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) ORDER BY column_name(s);
Example -:

```
mysql> select * from student;
+---------+--------+-------+---------+-------------+
| roll_no | name   | class | section | farher_name |
+---------+--------+-------+---------+-------------+
|       1 | vikash | bca   | a       | xyz1        |
|       1 | neha   | mca   | a       | xyz2        |
|       1 | vikay  | mca   | b       | xyz3        |
|       1 | vijay  | bca   | b       | xyz4        |
+---------+--------+-------+---------+-------------+
4 rows in set (0.00 sec)

mysql> select count(roll_no),section from student group by section;
+----------------+---------+
| count(roll_no) | section |
+----------------+---------+
|              2 | a       |
|              2 | b       |
+----------------+---------+
2 rows in set (0.03 sec)
```

SUM -: select sum(salary),city from emp group by city;
Example -:

```
mysql> select * from emp;
+------------+--------+--------+
| city       | gender | salary |
+------------+--------+--------+
| Chandigarh | mail   |  40000 |
| Punjab     | mail   |  50000 |
| Chandigarh | female |  40000 |
| Punjab     | femail |  20000 |
+------------+--------+--------+
4 rows in set (0.00 sec)
```

```
mysql> select sum(salary),city from emp group by city;
+-------------+------------+
| sum(salary) | city       |
+-------------+------------+
|       80000 | Chandigarh |
|       70000 | Punjab     |
+-------------+------------+
2 rows in set (0.00 sec)

mysql> select city,gender,sum(salary),city from emp group by city;
+------------+--------+-------------+------------+
| city       | gender | sum(salary) | city       |
+------------+--------+-------------+------------+
| Chandigarh | mail   |       80000 | Chandigarh |
| Punjab     | mail   |       70000 | Punjab     |
+------------+--------+-------------+------------+
2 rows in set (0.00 sec)

mysql> select city,gender,sum(salary) from emp group by city,gender;
+------------+--------+-------------+
| city       | gender | sum(salary) |
+------------+--------+-------------+
| Chandigarh | mail   |       40000 |
| Punjab     | mail   |       50000 |
| Chandigarh | female |       40000 |
| Punjab     | femail |       20000 |
+------------+--------+-------------+
4 rows in set (0.00 sec)
```

- ✓ **HAVING Clause -:** The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

Syntax-:
SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) HAVING condition ORDER BY column_name(s);
Example -:

```
mysql> select count(class),section from student group by section having sum(sect
ion)<=2;
+--------------+---------+
| count(class) | section |
+--------------+---------+
|            2 | a       |
|            2 | b       |
+--------------+---------+
2 rows in set, 4 warnings (0.00 sec)

mysql> select count(class),section from student group by section having sum(sect
ion)<=1;
+--------------+---------+
| count(class) | section |
+--------------+---------+
|            2 | a       |
|            2 | b       |
+--------------+---------+
2 rows in set, 4 warnings (0.00 sec)
```

- ✓ **Primary Key On Create Table -:** Creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

Syntax-:

- CREATE TABLE Persons (ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255),Age int,PRIMARY KEY (ID));

- CREATE TABLE Persons (ID int NOT NULL PRIMARY KEY,LastName varchar(255) NOT NULL,FirstName varchar(255),Age int);

- CREATE TABLE Persons (ID int NOT NULL,LastName varchar(255) NOT NULL,FirstName varchar(255),Age int,CONSTRAINT PK_Persons PRIMARY KEY (ID,LastName));

Example -:

```
mysql> create table student1 (roll int primary key,name varchar(20));
Query OK, 0 rows affected (2.27 sec)

mysql> desc student1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| roll  | int         | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.04 sec)
```

Multiple Columns Primary Key -:

Example -:

```
mysql> create table student2 (rollno int,name varchar(20),primary key(rollno,nam
e));
Query OK, 0 rows affected (0.58 sec)

mysql> desc student2;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | NO   | PRI | NULL    |       |
| name   | varchar(20) | NO   | PRI | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.14 sec)

mysql> insert into student2 values (4255,"vikash");
Query OK, 1 row affected (0.14 sec)

mysql> select * from student2;
+--------+--------+
| rollno | name   |
+--------+--------+
|   4255 | vikash |
+--------+--------+
1 row in set (0.00 sec)
```

- ✓ **Primary Key On Alter Table -:** To create a PRIMARY KEY constraint on the "ID" column when the table is already created

Syntax-:

- ALTER TABLE Persons ADD PRIMARY KEY (ID);
- ALTER TABLE Persons ADD CONSTRAINT PK_table PRIMARY KEY (ID,LastName);
- ALTER TABLE Persons DROP PRIMARY KEY;
- ALTER TABLE Persons DROP CONSTRAINT PK_ Persons;

Example -:

```
mysql> desc student3;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| roll  | int         | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.05 sec)

mysql> alter table student3 add primary key (name);
ERROR 1062 (23000): Duplicate entry 'nisha' for key 'student3.PRIMARY'
mysql> alter table student3 add primary key (roll);
Query OK, 0 rows affected (3.15 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc student3;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| roll  | int         | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.05 sec)
```

Drop Primary Key Example -:

```
mysql> desc student3;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| roll  | int         | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.05 sec)

mysql> alter table student3 drop primary key;
Query OK, 3 rows affected (3.52 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> desc student3;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| roll  | int         | NO   |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.06 sec)
```

- ✓ **Foreign Key Constraint -:** The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Syntax-:

- CREATE TABLE Orders (OrderID int NOT NULL,OrderNumber int NOT NULL, PersonID int,PRIMARY KEY (OrderID),FOREIGN KEY (PersonID) REFERENCES Persons(PersonID));

- CREATE TABLE Orders (OrderID int NOT NULL,OrderNumber int NOT NULL, PersonID int,PRIMARY KEY (OrderID),CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)REFERENCES Persons(PersonID));

Example -:

```
mysql> desc student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | NO   | PRI | NULL    |       |
| sname  | varchar(10) | YES  |     | NULL    |       |
| class  | varchar(10) | YES  |     | NULL    |       |
| fees   | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

```
mysql> select * from fees;
+------+
| roll |
+------+
| 4006 |
+------+
1 row in set (0.00 sec)

mysql> insert into fees values(4007);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
fails (`my`.`fees`, CONSTRAINT `fees_ibfk_1` FOREIGN KEY (`roll`) REFERENCES `s
tudent` (`rollno`))
```

- ✓ **Foreign Key On Alter Table -:** To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created

Syntax-:

- ALTER TABLE Orders ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

- ALTER TABLE Orders ADD CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

Example -:

```
mysql> desc fees;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| roll  | int  | YES  | MUL | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.01 sec)

mysql> alter table fees add foreign key (roll) references student(rollno);
Query OK, 3 rows affected (0.21 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> show create table fees;
```

- ✓ **Drop A Foreign Key Constraint -:**

Syntax-:

- ALTER TABLE Orders DROP FOREIGN KEY FK_PersonOrder;
- ALTER TABLE Orders DROP CONSTRAINT FK_PersonOrder;

Example -:

```
mysql> alter table fees drop foreign key fees_ibfk_1;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> alter table fees add constraint fk_roll foreign key (roll) references st
udent(rollno);
Query OK, 3 rows affected (0.22 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> desc fees;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| roll  | int  | YES  | MUL | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

- ✓ **Unique Constraint On Create Table -:**

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

Syntax-:

- CREATE TABLE Persons(ID int NOT NULL UNIQUE,LastName varchar(255) NOT NULL,FirstName varchar(255),Age int);

- CREATE TABLE Persons(ID int NOT NULL,LastName varchar(255) NOT NULL,
- FirstName varchar(255),Age int,UNIQUE (ID));

- CREATE TABLE Persons(ID int NOT NULL,LastName varchar(255) NOT NULL,
- Firstname varchar(255),Age int,CONSTRAINT UC_Person UNIQUE (ID,LastName));

Example -:

```
mysql> use my;
Database changed
mysql> create table st1 (rollno int unique,name varchar(10));
Query OK, 0 rows affected (1.47 sec)

mysql> desc st1;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | YES  | UNI | NULL    |       |
| name   | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.08 sec)
```

- **Unique Constraint On Alter Table -:**To Create A Unique Constraint On The "Id" Column When The Table Is Already Created.

Syntax-:

- ALTER TABLE Persons ADD UNIQUE (ID);

- ALTER TABLE Persons ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);

Example -:

```
mysql> desc student3;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| roll  | int         | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.05 sec)

mysql> alter table st1 add  unique(name);
Query OK, 0 rows affected (0.52 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc st1;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | YES  | UNI | NULL    |       |
| name   | varchar(10) | YES  | UNI | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.05 sec)
```

- ✓ **DROP a UNIQUE Constraint -:**

Syntax-:

- ALTER TABLE Persons DROP INDEX UC_Person;

- ALTER TABLE Persons DROP CONSTRAINT UC_Person;

Example -:

```
mysql> ALTER TABLE st1 DROP CONSTRAINT rollno;
Query OK, 0 rows affected (1.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc st1;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | YES  |     | NULL    |       |
| name   | varchar(10) | YES  | UNI | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.05 sec)
```

- ✓ **Not Null On Create Table** -: The "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

- By default, a column can hold NULL values.
- The Not NULL constraint enforces a column to NOT accept NULL values.
- This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

Syntax-:
CREATE TABLE Persons (ID int NOT NULL,LastName varchar(255) NOT NULL, FirstName varchar(255) NOT NULL,Age int);

Example -:

```
mysql> create table st2 (roll_no int,name varchar(20) not null,age int not null)
;
Query OK, 0 rows affected (1.37 sec)

mysql> desc st2;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| roll_no | int         | YES  |     | NULL    |       |
| name    | varchar(20) | NO   |     | NULL    |       |
| age     | int         | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.10 sec)

mysql> insert into st2 values(1,"vikash",22);
Query OK, 1 row affected (0.18 sec)

mysql> insert into st2 values(1,"vikash",);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near ')' at
 line 1
mysql> insert into st2 values(1,"vikash",null);
ERROR 1048 (23000): Column 'age' cannot be null
mysql>
```

- ✓ **Not Null On Alter Table** -: To create a NOT NULL constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

Syntax-:
ALTER TABLE Persons MODIFY Age int NOT NULL;
Example -:

```
mysql> ALTER TABLE st1 modify rollno int not null;
Query OK, 0 rows affected (2.38 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc st1;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | NO   |     | NULL    |       |
| name   | varchar(10) | YES  | UNI | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.07 sec)
```

- ✓ **Check On Create Table -:**

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a column it will allow only certain values for this column.
- If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

Creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

Syntax-:
CREATE TABLE Persons (ID int NOT NULL,LastName varchar(255) NOTNULL,
FirstName varchar(255),Age int,CHECK (Age>=18));

Example -:

```
mysql> create table st3 (rollno int,name varchar(20), age int check(age>=18));
Query OK, 0 rows affected (1.03 sec)

mysql> insert into st3 values(1,"vikash",18);
Query OK, 1 row affected (0.25 sec)

mysql> insert into st3 values(1,"vikash",17);
ERROR 3819 (HY000): Check constraint 'st3_chk_1' is violated.
mysql> create table st4 (fees int,salary int, check(fees>=1000 and salary>=10000
));
Query OK, 0 rows affected (1.09 sec)

mysql> desc st3;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int         | YES  |     | NULL    |       |
| name   | varchar(20) | YES  |     | NULL    |       |
| age    | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.04 sec)

mysql> desc st4;
+--------+------+------+-----+---------+-------+
| Field  | Type | Null | Key | Default | Extra |
+--------+------+------+-----+---------+-------+
| fees   | int  | YES  |     | NULL    |       |
| salary | int  | YES  |     | NULL    |       |
+--------+------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

- ✓ **Check On Alter Table -:** To create a CHECK constraint on the "Age" column when the table is already created

Syntax-:
ALTER TABLE Persons ADD CHECK (Age>=18);
Example -:

```
mysql> alter table st4 add (check(fees>=1000),check(salary<=10000));
Query OK, 0 rows affected (3.45 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table st4;
+-------+----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
------------------+
| Table | Create Table


                  |
+-------+----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
------------------+
| st4   | CREATE TABLE `st4` (
  `fees` int DEFAULT NULL,
  `salary` int DEFAULT NULL,
  CONSTRAINT `st4_chk_1` CHECK ((`fees` >= 1000)),
  CONSTRAINT `st4_chk_2` CHECK ((`salary` <= 10000))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+----------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
------------------+
1 row in set (0.00 sec)
```

- ✓ **Drop A Check Constraint** -: To drop a CHECK constraint.

Syntax-:
ALTER TABLE Persons DROP CONSTRAINT CHK_PersonAge;

Example -:

```
mysql> show create table st4;
+-------+----------------------------------------------------------------------------+
| Table | Create Table                                                               |
+-------+----------------------------------------------------------------------------+
| st4   | CREATE TABLE `st4` (
  `fees` int DEFAULT NULL,
  `salary` int DEFAULT NULL,
  CONSTRAINT `st4_chk_1` CHECK (((`fees` >= 1000) and (`salary` >= 10000)))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+----------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> alter table st4 drop constraint `st4_chk_1`;
Query OK, 0 rows affected (0.41 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc st4;
+--------+------+------+-----+---------+-------+
| Field  | Type | Null | Key | Default | Extra |
+--------+------+------+-----+---------+-------+
| fees   | int  | YES  |     | NULL    |       |
| salary | int  | YES  |     | NULL    |       |
+--------+------+------+-----+---------+-------+
```

- ✓ **DEFAULT on CREATE TABLE -:** The sets a DEFAULT value for the "City" column when the "Persons" table is created:
  - The DEFAULT constraint is used to set a default value for a column.
  - The default value will be added to all new records, if no other value is specified.

Syntax-:
CREATE TABLE Persons (ID int NOT NULL,LastName varchar(10) NOT NULL,
FirstName varchar(10),Age int,City varchar(255) DEFAULT 'Sandnes');
Example -:

```
mysql> create table st1 (rollno int default 0,name varchar(15));
Query OK, 0 rows affected (0.85 sec)

mysql> insert into st1 (name) values("vikash"),("ajay");
Query OK, 2 rows affected (0.36 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from st1;
+--------+--------+
| rollno | name   |
+--------+--------+
|      0 | vikash |
|      0 | ajay   |
+--------+--------+
2 rows in set (0.00 sec)
```

- ✓ **Default On Alter Table** -: To create a DEFAULT constraint on the "City" column when the table is already created

Syntax-:
ALTER TABLE table_name ALTER col_name SET DEFAULT 'name';

Example -:

```
mysql> alter table st1 alter name set default "neha";
Query OK, 0 rows affected (0.33 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from st1;
+--------+--------+
| rollno | name   |
+--------+--------+
|      0 | vikash |
|      0 | ajay   |
+--------+--------+
2 rows in set (0.04 sec)

mysql> insert into st1 (rollno) values(1),(2);
Query OK, 2 rows affected (0.13 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from st1;
+--------+--------+
| rollno | name   |
+--------+--------+
|      0 | vikash |
|      0 | ajay   |
|      1 | neha   |
|      2 | neha   |
+--------+--------+
4 rows in set (0.00 sec)
```

- ✓ **Auto Increment Field** -: Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

Often this is the primary key field that we would like to be created automatically every time a new record is inserted.
The following SQL statement defines the "Personid" column to be an auto-increment primary key field in the "Persons" table:

Syntax-:
CREATE TABLE Persons (Personid int NOT NULL AUTO_INCREMENT,LastName varchar(255) NOT NULL,FirstName varchar(255),Age int,PRIMARY KEY (Personid));
Example -:

```
mysql> CREATE TABLE st2 (rollno int NOT NULL AUTO_INCREMENT,LastName varchar(255
) NOT NULL,FirstName varchar(255),Age int,PRIMARY KEY (rollno));
Query OK, 0 rows affected (1.27 sec)

mysql> select * from st2;
Empty set (0.01 sec)

mysql> insert into st2 values(0,'bhartwaj','roshni',22);
Query OK, 1 row affected (0.10 sec)

mysql> select * from st2;
+--------+----------+-----------+------+
| rollno | LastName | FirstName | Age  |
+--------+----------+-----------+------+
|      1 | sharma   | vikash    |   21 |
|      2 | bhartwaj | roshni    |   22 |
|      3 | bhartwaj | roshni    |   22 |
|      6 | singh    | alpana    |   22 |
|      7 | bhartwaj | roshni    |   22 |
+--------+----------+-----------+------+
5 rows in set (0.00 sec)
```

- ✓ **Gender enum -:**

Syntax-:
Create table st2 (gender enum(“male”,”fenale”));

Example -:

```
mysql> Create table st3 (gender enum('male','fenale'));
Query OK, 0 rows affected (1.72 sec)

mysql> insert into st3 values('male');
Query OK, 1 row affected (0.10 sec)

mysql> insert into st3 values('abcd');
ERROR 1265 (01000): Data truncated for column 'gender' at row 1
mysql> desc st3;
+--------+----------------------+------+-----+---------+-------+
| Field  | Type                 | Null | Key | Default | Extra |
+--------+----------------------+------+-----+---------+-------+
| gender | enum('male','fenale') | YES  |     | NULL    |       |
+--------+----------------------+------+-----+---------+-------+
1 row in set (0.04 sec)

mysql> select * from st3;
+--------+
| gender |
+--------+
| male   |
+--------+
1 row in set (0.00 sec)
```

- ✓ **Lower case –:**

Syntax-:
Select lcase(col_name) from table_name;
Update -: update table_name set col_name=lcase(cal_name);
Example -:

```
mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | bca   |
|      3 | ajay   | bca   |
|      8 | komal  | bca   |
|      7 | neha   | bca   |
+--------+--------+-------+
4 rows in set (0.00 sec)

mysql> select lcase(name) from student;
+--------------+
| lcase(name)  |
+--------------+
| vikash       |
| ajay         |
| komal        |
| neha         |
+--------------+
4 rows in set (0.09 sec)

mysql>
```

```
mysql> update student set name=lcase(name);
Query OK, 0 rows affected (0.00 sec)
Rows matched: 4  Changed: 0  Warnings: 0

mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | bca   |
|      3 | ajay   | bca   |
|      8 | komal  | bca   |
|      7 | neha   | bca   |
+--------+--------+-------+
4 rows in set (0.00 sec)
```

- ✓ **Dual table -:**

Syntax-:
Select ucase("hello") from dual;

Example -:

```
mysql> select ucase("hello") from dual;
+----------------+
| ucase("hello") |
+----------------+
| HELLO          |
+----------------+
1 row in set (0.05 sec)

mysql>
```

- ✓ **Ascii -:**

Syntax-:
Select ascii(“v”) from dual;
Example -:

```
mysql> select ascii("v") from dual;
+------------+
| ascii("v") |
+------------+
|        118 |
+------------+
1 row in set (0.03 sec)
```

- ✓ **Char_length -:**

Syntax-:
Select char_length(‘vikash’) from dual;
Example -:

```
mysql> Select char_length('vikash') from dual;
+-----------------------+
| char_length('vikash') |
+-----------------------+
|                     6 |
+-----------------------+
1 row in set (0.04 sec)

mysql>
```

- ✓ **Concat -:**

Syntax-:
Select concat(‘vikash’,’sharma’) from dual;
Example -:

```
mysql> Select concat('vikash','sharma') from dual;
+---------------------------+
| concat('vikash','sharma') |
+---------------------------+
| vikashsharma              |
+---------------------------+
1 row in set (0.02 sec)

mysql> Select concat_ws("-",'vikash','sharma') from dual;
+-----------------------------------+
| concat_ws("-",'vikash','sharma') |
+-----------------------------------+
| vikash-sharma                     |
+-----------------------------------+
1 row in set (0.00 sec)
```

- ✓ **Left, Right and Mid -:**

Syntax-:
Select left(‘vikash’,4) from dual;
Select right(‘vikash’,4) from dual;

Select mid('vikash',3,2) from dual;
Example -:

```
mysql> Select left('vikash',4) from dual;
+------------------+
| left('vikash',4) |
+------------------+
| vika             |
+------------------+
1 row in set (0.00 sec)

mysql> Select right('vikash',4) from dual;
+-------------------+
| right('vikash',4) |
+-------------------+
| kash              |
+-------------------+
1 row in set (0.00 sec)

mysql> Select mid('vikash',3,2) from dual;
+-------------------+
| mid('vikash',3,2) |
+-------------------+
| ka                |
+-------------------+
1 row in set (0.00 sec)
```

- ✓ **Trim -:** left and right side space remove.
  Syntax-:
  Select trim(' Vikash ') from dual;
  Example -:

```
mysql> Select trim('     vikash      ') from dual;
+--------------------------+
| trim('     vikash      ') |
+--------------------------+
| vikash                   |
+--------------------------+
1 row in set (0.00 sec)

mysql> Select trim(leading 'v' from 'vikash') from dual;
+---------------------------------+
| trim(leading 'v' from 'vikash') |
+---------------------------------+
| ikash                           |
+---------------------------------+
1 row in set (0.03 sec)
```

- ✓ **Strcmp** -: compare string
  Syntax-:
  Select strcmp('Vikash','vikash) from dual;
  Example -:

```
mysql> Select strcmp('vikash','vikash') from dual;
+---------------------------+
| strcmp('vikash','vikash') |
+---------------------------+
|                         0 |
+---------------------------+
1 row in set (0.00 sec)

mysql> Select strcmp('vikash','aikash') from dual;
+---------------------------+
| strcmp('vikash','aikash') |
+---------------------------+
|                         1 |
+---------------------------+
1 row in set (0.00 sec)
```

```
mysql> Select strcmp('vikash','zhikash') from dual;
+-----------------------------+
| strcmp('vikash','zhikash') |
+-----------------------------+
|                          -1 |
+-----------------------------+
1 row in set (0.00 sec)
```

- ✓ **Abs, ceil and floor -:**

  Syntax-:
  Select abs(-4255) from dual;
  Select ceil(12.5) from dual;
  Select floor(12.5) from dual;
  Example -:

```
mysql> Select abs(-4255) from dual;
+-------------+
| abs(-4255) |
+-------------+
|        4255 |
+-------------+
1 row in set (0.03 sec)

mysql> Select ceil(12.5) from dual;
+-------------+
| ceil(12.5) |
+-------------+
|          13 |
+-------------+
1 row in set (0.00 sec)

mysql> Select floor(12.5) from dual;
+--------------+
| floor(12.5) |
+--------------+
|           12 |
+--------------+
1 row in set (0.00 sec)
```

- ✓ **Greatest, mod, rand(random number), sqrt(squar root) and pow(power)** -:

  Syntax-:
  Select greatest(99,105,11,12,13) from dual;
  Select mod(12,3) from dual;
  Select pow(12,2) from dual;
  Select rand() from dual;
  Select sqrt(25) from dual;
  Example -:

```
mysql> Select greatest(12,13,14,105,116) from dual;
+-----------------------------+
| greatest(12,13,14,105,116) |
+-----------------------------+
|                         116 |
+-----------------------------+
1 row in set (0.04 sec)

mysql> Select mod(12,5) from dual;
+------------+
| mod(12,5) |
+------------+
|          2 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> Select pow(12,5) from dual;
+-----------+
| pow(12,5) |
+-----------+
|    248832 |
+-----------+
1 row in set (0.04 sec)

mysql> Select rand() from dual;
+--------------------+
| rand()             |
+--------------------+
| 0.08172052268099088 |
+--------------------+
1 row in set (0.05 sec)

mysql> Select sqrt(25) from dual;
+----------+
| sqrt(25) |
+----------+
|        5 |
+----------+
1 row in set (0.04 sec)
```

- ✓ **Round function -:**Two decimal places round.

  Syntax-:

  Select round(25.374896) from dual;

  Example -:

```
mysql> Select round(25.637589) from dual;
+------------------+
| round(25.637589) |
+------------------+
|               26 |
+------------------+
1 row in set (0.03 sec)

mysql> Select round(25.111589) from dual;
+------------------+
| round(25.111589) |
+------------------+
|               25 |
+------------------+
1 row in set (0.00 sec)
```

- ✓ **Truncate function -:** decimal poit all truncate .

  Syntax-:

  Select truncate(25.374896) from dual;

Example -:

```
mysql> Select truncate(25.111589,2) from dual;
+------------------------+
| truncate(25.111589,2) |
+------------------------+
|                  25.11 |
+------------------------+
1 row in set (0.00 sec)
```

- ✓ **IsNull function -:**

  Syntax-:

  Select isnull(class) from dual;

Example -:

```
mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | bca   |
|      3 | ajay   | bca   |
|      8 | komal  | bca   |
|      7 | neha   | bca   |
+--------+--------+-------+
4 rows in set (0.00 sec)
```

```
mysql> update student set class=null;
Query OK, 4 rows affected (0.12 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | NULL  |
|      3 | ajay   | NULL  |
|      8 | komal  | NULL  |
|      7 | neha   | NULL  |
+--------+--------+-------+
4 rows in set (0.01 sec)

mysql> select isnull(class) from student;
+---------------+
| isnull(class) |
+---------------+
|             1 |
|             1 |
|             1 |
|             1 |
+---------------+
4 rows in set (0.00 sec)
```

- ✓ **Date function -:**

Syntax-:

Select now() from dual;
Select curdate() from dual;
Select curtime() from dual;
Select date(now()) from dual;

Example -:

```
mysql> select now() from dual;
+---------------------+
| now()               |
+---------------------+
| 2021-06-11 14:21:55 |
+---------------------+
1 row in set (0.03 sec)

mysql> select curdate() from dual;
+------------+
| curdate()  |
+------------+
| 2021-06-11 |
+------------+
1 row in set (0.02 sec)

mysql> select curtime() from dual;
+-----------+
| curtime() |
+-----------+
| 14:22:27  |
+-----------+
1 row in set (0.00 sec)

mysql> select date(now()) from dual;
+-------------+
| date(now()) |
+-------------+
| 2021-06-11  |
+-------------+
1 row in set (0.03 sec)

mysql>
```

- ➢ **Extract function -:**

Syntax-:

Select extract(day from now()) from dual;

Example -:

```
mysql> select extract(day from now()) from dual;
+-------------------------+
| extract(day from now()) |
+-------------------------+
|                      11 |
+-------------------------+
1 row in set (0.03 sec)

mysql> select extract(month from now()) from dual;
+---------------------------+
| extract(month from now()) |
+---------------------------+
|                         6 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select extract(second from now()) from dual;
+---------------------------+
| extract(second from now()) |
+---------------------------+
|                         7 |
+---------------------------+
1 row in set (0.00 sec)
```

- **Date add function -:**

Syntax-:

Select date_add(now(),interval 1 year) from dual;

Example -:

```
mysql> select date_add(now(),interval 1 year) from dual;
+-----------------------------------+
| date_add(now(),interval 1 year)   |
+-----------------------------------+
| 2022-06-11 14:34:59               |
+-----------------------------------+
1 row in set (0.03 sec)
```

- **Date diff function -:**

Syntax-:

Select datediff(now(),"2022-06-11") from dual;

Example -:

```
mysql> Select datediff(now(),"2022-06-11") from dual;
+-------------------------------+
| datediff(now(),"2022-06-11") |
+-------------------------------+
|                          -365 |
+-------------------------------+
1 row in set (0.00 sec)
```

- **Date sub function -:**

Syntax-:

Select date_sub(now(),interval 1 year) from dual;

Example -:

```
mysql> Select date_sub(now(),interval 1 year) from dual;
+-----------------------------------+
| date_sub(now(),interval 1 year)   |
+-----------------------------------+
| 2020-06-11 14:51:39               |
+-----------------------------------+
1 row in set (0.00 sec)
```

- **Cast function -:**

Syntax-:

Select cast("2", as decimal) from dual;

Example -:

```
mysql> Select cast("8284935160" as decimal) from dual;
+--------------------------------+
| cast("8284935160" as decimal)  |
+--------------------------------+
|                     8284935160 |
+--------------------------------+
1 row in set (0.00 sec)
```

- **Date format function -:**

Syntax-:

Select date_format(now(),"%d %m %y") from dual;

Example -:

```
mysql> Select date_format(now(),"%d %m %y") from dual;
+--------------------------------+
| date_format(now(),"%d %m %y")  |
+--------------------------------+
| 11 06 21                       |
+--------------------------------+
1 row in set (0.03 sec)

mysql> Select date_format(now(),"%d %M %y") from dual;
+--------------------------------+
| date_format(now(),"%d %M %y")  |
+--------------------------------+
| 11 June 21                     |
+--------------------------------+
1 row in set (0.03 sec)

mysql> Select date_format(now(),"%D %M %Y") from dual;
+--------------------------------+
| date_format(now(),"%D %M %Y")  |
+--------------------------------+
| 11th June 2021                 |
+--------------------------------+
1 row in set (0.00 sec)
```

- **Convert function -:**

Syntax-:

Select convert("2" ,as decimal) from dual;

Example -:

```
mysql> Select convert('8284935160',decimal) from dual;
+--------------------------------+
| convert('8284935160',decimal)  |
+--------------------------------+
|                     8284935160 |
+--------------------------------+
1 row in set (0.00 sec)
```

- **Distinct function -:**

Syntax-:

Select distinct(class) from table_name;

Example -:

```
mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | NULL  |
|      3 | ajay   | NULL  |
|      8 | komal  | NULL  |
|      7 | neha   | NULL  |
+--------+--------+-------+
4 rows in set (0.00 sec)
```

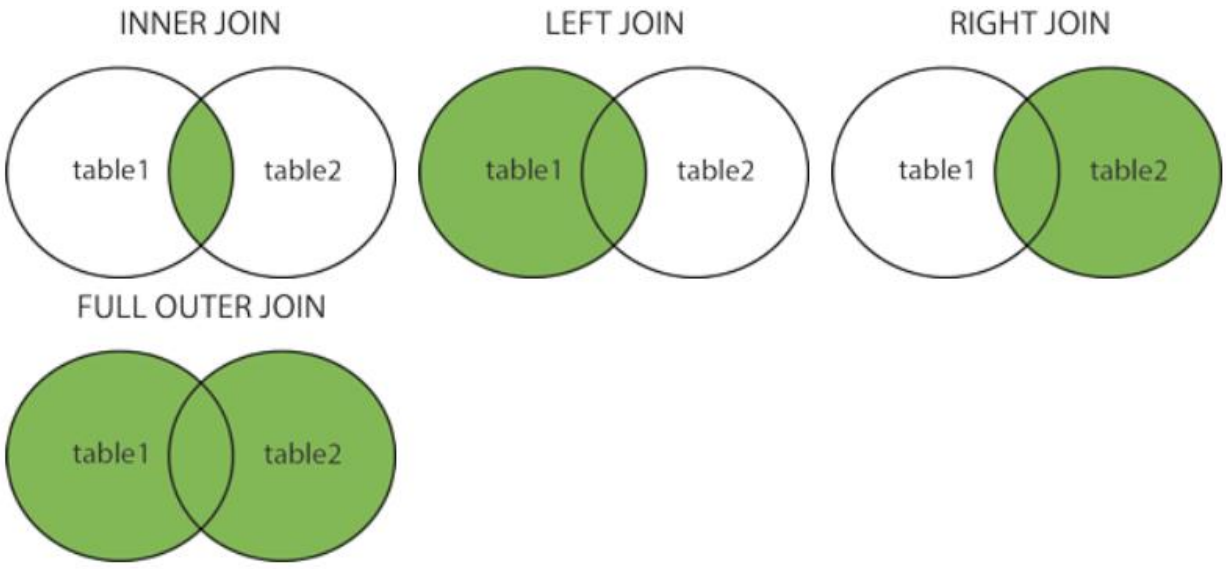
```
mysql> select distinct(class) from student;
+-------+
| class |
+-------+
| NULL  |
+-------+
1 row in set (0.00 sec)

mysql> select distinct(class),rollno from student;
+-------+--------+
| class | rollno |
+-------+--------+
| NULL  |      1 |
| NULL  |      3 |
| NULL  |      8 |
| NULL  |      7 |
+-------+--------+
4 rows in set (0.00 sec)
```

- **JOIN -:** A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

Different Types of JOINs

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



INNER JOIN -:

Syntax-:

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM 1table_name
INNER JOIN 2table_name ON Orders.CustomerID=Customers.CustomerID;

Example -:

```
mysql> select student.name,fees.fees from student join fees on student.rollno=fe
es.rollno;
+--------+-------+
| name   | fees  |
+--------+-------+
| vikash | 10000 |
| ajay   | 20000 |
| komal  | 30000 |
+--------+-------+
3 rows in set (0.00 sec)

mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | NULL  |
|      3 | ajay   | NULL  |
|      8 | komal  | NULL  |
|      7 | neha   | NULL  |
+--------+--------+-------+
4 rows in set (0.03 sec)

mysql> select * from fees;
+--------+-------+
| rollno | fees  |
+--------+-------+
|      1 | 10000 |
|      3 | 20000 |
|      8 | 30000 |
|      9 | 40000 |
+--------+-------+
4 rows in set (0.00 sec)

mysql> select * from student inner join fees on student.rollno=fees.rollno;
+--------+--------+-------+--------+-------+
| rollno | name   | class | rollno | fees  |
+--------+--------+-------+--------+-------+
|      1 | vikash | NULL  |      1 | 10000 |
|      3 | ajay   | NULL  |      3 | 20000 |
|      8 | komal  | NULL  |      8 | 30000 |
+--------+--------+-------+--------+-------+
3 rows in set (0.00 sec)

mysql> select student.name,fees.fees from student inner join fees on student.rol
lno=fees.rollno;
+--------+-------+
| name   | fees  |
+--------+-------+
| vikash | 10000 |
| ajay   | 20000 |
| komal  | 30000 |
+--------+-------+
3 rows in set (0.00 sec)
```

EQUAL JOIN (=) -:

```
mysql> select student.name,fees.fees from student,fees where student.rollno=fees
.rollno;
+--------+-------+
| name   | fees  |
+--------+-------+
| vikash | 10000 |
| ajay   | 20000 |
| komal  | 30000 |
+--------+-------+
3 rows in set (0.00 sec)

mysql> select * from student inner join fees on student.rollno=fees.rollno where
 student.rollno>4;
+--------+-------+-------+--------+-------+
| rollno | name  | class | rollno | fees  |
+--------+-------+-------+--------+-------+
|      8 | komal | NULL  |      8 | 30000 |
+--------+-------+-------+--------+-------+
1 row in set (0.00 sec)
```

NATURAL JOIN -:

Syntax-:

SELECT * FROM 1table_name Natural JOIN 2table name;

Example -:

```
mysql> SELECT * FROM student Natural JOIN fees;
+--------+--------+-------+-------+
| rollno | name   | class | fees  |
+--------+--------+-------+-------+
|      1 | vikash | NULL  | 10000 |
|      3 | ajay   | NULL  | 20000 |
|      8 | komal  | NULL  | 30000 |
+--------+--------+-------+-------+
3 rows in set (0.02 sec)
```

LEFT JOIN -: The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

Syntax-:

SELECT column_name(s) FROM table1 LEFT JOIN table2 ON table1.column_name = table2.column_name;

Example -:

```
mysql> select * from student left join fees on student.rollno=fees.rollno;
+--------+--------+-------+--------+-------+
| rollno | name   | class | rollno | fees  |
+--------+--------+-------+--------+-------+
|      1 | vikash | NULL  |      1 | 10000 |
|      3 | ajay   | NULL  |      3 | 20000 |
|      8 | komal  | NULL  |      8 |  NULL |
|      8 | komal  | NULL  |      8 | 30000 |
|      7 | neha   | NULL  |   NULL |  NULL |
+--------+--------+-------+--------+-------+
5 rows in set (0.05 sec)

mysql> select * from student left join fees on student.rollno=fees.rollno where
fees.rollno is null;
+--------+------+-------+--------+------+
| rollno | name | class | rollno | fees |
+--------+------+-------+--------+------+
|      7 | neha | NULL  |   NULL | NULL |
+--------+------+-------+--------+------+
1 row in set (0.00 sec)
```

RIGHT JOIN -: The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

Syntax-:

SELECT column_name(s) FROM table1 RIGHT JOIN table2 ON table1.column_name = table2.column_name;

Example -:

```
mysql> select * from student right join fees on student.rollno=fees.rollno;
+--------+--------+-------+--------+-------+
| rollno | name   | class | rollno | fees  |
+--------+--------+-------+--------+-------+
|      1 | vikash | NULL  |      1 | 10000 |
|      3 | ajay   | NULL  |      3 | 20000 |
|      8 | komal  | NULL  |      8 | 30000 |
|   NULL | NULL   | NULL  |      9 | 40000 |
|   NULL | NULL   | NULL  |      9 |  NULL |
|      8 | komal  | NULL  |      8 |  NULL |
+--------+--------+-------+--------+-------+
6 rows in set (0.00 sec)
```

```
mysql> select * from student right join fees on student.rollno=fees.rollno where
 fees.rollno is null;
Empty set (0.00 sec)
```

- **UNION OPERATOR -:** The UNION operator is used to combine the result-set of two or more SELECT statements.
- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

Syntax-:

SELECT column_name(s) FROM table1 UNION SELECT column_name(s) FROM table2;

Example -:

```
mysql> select rollno from student union select rollno from fees;
+--------+
| rollno |
+--------+
|      1 |
|      3 |
|      8 |
|      7 |
|      9 |
+--------+
5 rows in set (0.02 sec)

mysql> select name from student union select fees from fees;
+--------+
| name   |
+--------+
| vikash |
| ajay   |
| komal  |
| neha   |
| 10000  |
| 20000  |
| 30000  |
| 40000  |
| NULL   |
+--------+
9 rows in set (0.00 sec)
```

UNION ALL -: The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:

Syntax-:

SELECT column_name(s) FROM table1 UNION ALL SELECT column_name(s) FROM table2;

Example -:

```
mysql> select name from student union all select fees from fees;
+--------+
| name   |
+--------+
| vikash |
| ajay   |
| komal  |
| neha   |
| 10000  |
| 20000  |
| 30000  |
| 40000  |
| NULL   |
| NULL   |
+--------+
10 rows in set (0.00 sec)
```

```
mysql> select rollno from student union all select rollno from fees;
+--------+
| rollno |
+--------+
|      1 |
|      3 |
|      8 |
|      7 |
|      1 |
|      3 |
|      8 |
|      9 |
|      9 |
|      8 |
+--------+
10 rows in set (0.00 sec)
```

FULL OUTER JOIN -: The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Tip : FULL OUTER JOIN and FULL JOIN are the same.

Syntax-:

SELECT column_name(s) FROM table1 FULL OUTER JOIN table2 ON table1.column_name = table2.column_name WHERE condition;

Or

SELECT column_name(s) FROM table1 LEFT JOIN table2 ON table1.column_name = table2.column_name
UNION
SELECT column_name(s) FROM table1 RIGHT JOIN table2 ON table1.column_name = table2.column_name;

Example -:

```
mysql> select * from student left join fees on student.rollno=fees.rollno
    -> union
    -> select * from student right join fees on student.rollno=fees.rollno;
+--------+--------+-------+--------+-------+
| rollno | name   | class | rollno | fees  |
+--------+--------+-------+--------+-------+
|      1 | vikash | NULL  |      1 | 10000 |
|      3 | ajay   | NULL  |      3 | 20000 |
|      8 | komal  | NULL  |      8 |  NULL |
|      8 | komal  | NULL  |      8 | 30000 |
|      7 | neha   | NULL  |   NULL |  NULL |
|   NULL | NULL   | NULL  |      9 | 40000 |
|   NULL | NULL   | NULL  |      9 |  NULL |
+--------+--------+-------+--------+-------+
7 rows in set (0.00 sec)
```

CROSS JOIN -:

Example -:

```
mysql> SELECT * FROM student,fees;
+--------+--------+-------+--------+-------+
| rollno | name   | class | rollno | fees  |
+--------+--------+-------+--------+-------+
|      7 | neha   | NULL  |      1 | 10000 |
|      8 | komal  | NULL  |      1 | 10000 |
|      3 | ajay   | NULL  |      1 | 10000 |
|      1 | vikash | NULL  |      1 | 10000 |
|      7 | neha   | NULL  |      3 | 20000 |
|      8 | komal  | NULL  |      3 | 20000 |
|      3 | ajay   | NULL  |      3 | 20000 |
|      1 | vikash | NULL  |      3 | 20000 |
|      7 | neha   | NULL  |      8 | 30000 |
|      8 | komal  | NULL  |      8 | 30000 |
|      3 | ajay   | NULL  |      8 | 30000 |
|      1 | vikash | NULL  |      8 | 30000 |
|      7 | neha   | NULL  |      9 | 40000 |
|      8 | komal  | NULL  |      9 | 40000 |
|      3 | ajay   | NULL  |      9 | 40000 |
|      1 | vikash | NULL  |      9 | 40000 |
|      7 | neha   | NULL  |      9 |  NULL |
|      8 | komal  | NULL  |      9 |  NULL |
|      3 | ajay   | NULL  |      9 |  NULL |
|      1 | vikash | NULL  |      9 |  NULL |
|      7 | neha   | NULL  |      8 |  NULL |
|      8 | komal  | NULL  |      8 |  NULL |
|      3 | ajay   | NULL  |      8 |  NULL |
|      1 | vikash | NULL  |      8 |  NULL |
+--------+--------+-------+--------+-------+
24 rows in set (0.00 sec)
```

- **VIEWS -:** Views Are Virtual Tables That Do Not Store Any Data Of Their Own But Display Data Stored In Other Tables. In Other Words, Views Are Nothing But Sql Queries. A View Can Contain All Or A Few Rows From A Table. A Mysql View Can Show Data From One Table Or Many Tables.

Syntax-:

CREATE VIEW `view_name` AS SELECT statement;

Example -:

```
mysql> create view view2 as select * from student;
Query OK, 0 rows affected (0.58 sec)

mysql> select * from view2;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | NULL  |
|      3 | ajay   | NULL  |
|      8 | komal  | NULL  |
|      7 | neha   | NULL  |
+--------+--------+-------+
4 rows in set (0.18 sec)

mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | NULL  |
|      3 | ajay   | NULL  |
|      8 | komal  | NULL  |
|      7 | neha   | NULL  |
+--------+--------+-------+
4 rows in set (0.00 sec)

mysql> insert into view2 values(10,'raju','mca');
Query OK, 1 row affected (0.20 sec)
```

```
mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | vikash | NULL  |
|      3 | ajay   | NULL  |
|      8 | komal  | NULL  |
|      7 | neha   | NULL  |
|     10 | raju   | mca   |
+--------+--------+-------+
5 rows in set (0.00 sec)
```

DROP VIEW -:

Example -:

```
mysql> create view view3 as select name from student;
Query OK, 0 rows affected (0.11 sec)

mysql> select * from view3;
+--------+
| name   |
+--------+
| vikash |
| ajay   |
| komal  |
| neha   |
| raju   |
+--------+
5 rows in set (0.00 sec)

mysql> drop view view3;
Query OK, 0 rows affected (0.27 sec)
```

CREATE OR REPLACE VIEW -:

Example -:

```
mysql> create view view2 as select * from student;
Query OK, 0 rows affected (0.22 sec)

mysql> insert into view2 values(10,'raju','mca');
Query OK, 1 row affected (0.24 sec)

mysql> create or replace view view2 as select rollno,name from student
    -> where rollno=1;
Query OK, 0 rows affected (0.21 sec)

mysql> select * from view2;
+--------+--------+
| rollno | name   |
+--------+--------+
|      1 | vikash |
+--------+--------+
1 row in set (0.00 sec)
```

select view_definition,table_name from information_schema.views where table_name='view2';

Example -:

```
mysql> select view_definition,table_name from information_schema.views where tab
le_name='view2';
+--------------------------------------------------------------------------------
--------------------------------------------------+------------+
| VIEW_DEFINITION
                                                  | TABLE_NAME |
+--------------------------------------------------------------------------------
--------------------------------------------------+------------+
| select `my`.`student`.`rollno` AS `rollno`,`my`.`student`.`name` AS `name` fro
m `my`.`student` where (`my`.`student`.`rollno` = 1) | view2      |
+--------------------------------------------------------------------------------
--------------------------------------------------+------------+
1 row in set (0.01 sec)
```

desc information_schema.views;
select table_schema,table_name,definer from information_schema.views;
Example -:

```
mysql> desc information_schema.views;
+----------------------+---------------------------------+------+-----+---------
+-------+
| Field                | Type                            | Null | Key | Default
| Extra |
+----------------------+---------------------------------+------+-----+---------
+-------+
| TABLE_CATALOG        | varchar(64)                     | YES  |     | NULL
|       |
| TABLE_SCHEMA         | varchar(64)                     | YES  |     | NULL
|       |
| TABLE_NAME           | varchar(64)                     | YES  |     | NULL
|       |
| VIEW_DEFINITION      | longtext                        | YES  |     | NULL
|       |
| CHECK_OPTION         | enum('NONE','LOCAL','CASCADED') | YES  |     | NULL
|       |
| IS_UPDATABLE         | enum('NO','YES')                | YES  |     | NULL
|       |
| DEFINER              | varchar(288)                    | YES  |     | NULL
|       |
| SECURITY_TYPE        | varchar(7)                      | YES  |     | NULL
|       |
| CHARACTER_SET_CLIENT | varchar(64)                     | NO   |     | NULL
|       |
| COLLATION_CONNECTION | varchar(64)                     | NO   |     | NULL
|       |
+----------------------+---------------------------------+------+-----+---------
+-------+
10 rows in set (0.11 sec)
```

select view_definition,table_name from information_schema.views where table_name='view2';
Example -:

```
mysql> select IS_updatable,table_name from information_schema.views where table_
name='view2';
+--------------+------------+
| IS_UPDATABLE | TABLE_NAME |
+--------------+------------+
| YES          | view2      |
+--------------+------------+
1 row in set (0.00 sec)
```

NOT UPDATABLE -:

Example -:

```
mysql> create view view3 as select sum(rollno) from student;
Query OK, 0 rows affected (0.22 sec)

mysql> select IS_updatable,table_name from information_schema.views where table_
name='view3';
+--------------+------------+
| IS_UPDATABLE | TABLE_NAME |
+--------------+------------+
| NO           | view3      |
+--------------+------------+
1 row in set (0.00 sec)
```

create view view4 as select STUDEnt.rollno,fees.fees from student inner join fees on student.rollno=fees.rollno;
Example -:

```
mysql> create view view4 as select STUDEnt.rollno,fees.fees from student
    -> inner join fees on student.rollno=fees.rollno;
Query OK, 0 rows affected (0.22 sec)
```

```
mysql> select * from view4;
+--------+-------+
| rollno | fees  |
+--------+-------+
|      1 | 10000 |
|      3 | 20000 |
|      8 | 30000 |
|      8 |  NULL |
+--------+-------+
4 rows in set (0.24 sec)

mysql> select IS_updatable,table_name from information_schema.views where table
name='view4';
+--------------+------------+
| IS_UPDATABLE | TABLE_NAME |
+--------------+------------+
| YES          | view4      |
+--------------+------------+
1 row in set (0.00 sec)
```

SUBQUERY -:

Syntax-:

select name from student where rollno=(select rollno from fees where fees=10000);

- Single row subquery (=, not in)
- Multirow subquery (in, not in, any[<, >, = in], all[<, >)
- Multicolumn subquery
- Correlated subquery

  Example -:

```
mysql> select name from student where rollno=(select rollno from fees where
    -> fees=10000);
+--------+
| name   |
+--------+
| sharma |
+--------+
1 row in set (0.06 sec)
```

```
mysql> select name from student where rollno in (select rollno from fees where
    -> fees=null);
Empty set (0.00 sec)

mysql> select name from student where rollno not in (select rollno from fees whe
re
    -> fees=20000);
+--------+
| name   |
+--------+
| sharma |
| komal  |
| neha   |
| raju   |
| raju   |
+--------+
5 rows in set (0.00 sec)
```

```
mysql> select rollno from fees where rollno >any (select rollno from student);
+--------+
| rollno |
+--------+
|      3 |
|      8 |
|      9 |
|      9 |
|      8 |
+--------+
5 rows in set (0.05 sec)
```

```
mysql> select * from student;
+--------+--------+-------+
| rollno | name   | class |
+--------+--------+-------+
|      1 | sharma | NULL  |
|      3 | ajay   | NULL  |
|      8 | komal  | NULL  |
|      7 | neha   | NULL  |
|     10 | raju   | mca   |
|     10 | raju   | mca   |
+--------+--------+-------+
6 rows in set (0.00 sec)

mysql> select * from fees;
+--------+-------+
| rollno | fees  |
+--------+-------+
|      1 | 10000 |
|      3 | 20000 |
|      8 | 30000 |
|      9 | 40000 |
|      9 |  NULL |
|      8 |  NULL |
+--------+-------+
6 rows in set (0.00 sec)
```

```
mysql> select rollno from fees where rollno >all (select rollno from student);
Empty set (0.00 sec)

mysql> select rollno from fees where rollno <all (select rollno from student);
Empty set (0.00 sec)

mysql> select rollno from student where rollno >all (select rollno from fees);
+--------+
| rollno |
+--------+
|     10 |
|     10 |
+--------+
2 rows in set (0.00 sec)

mysql> select rollno from student where rollno <all (select rollno from fees);
Empty set (0.00 sec)
```

Exists and not exists operators -:

```
mysql> select rollno from fees where exists (select rollno from student where  f
ees.rollno=student.rollno);
+--------+
| rollno |
+--------+
|      1 |
|      3 |
|      8 |
|      8 |
+--------+
4 rows in set (0.00 sec)
```

- **From Clause -:**

Syntax-:

select max(rollno) from (select rollno from fees) as sample;
select max(rollno) from (select rollno from fees) as sample;
select (rollno) from (select rollno from fees) as sample;

Example -:

```
mysql> select max(rollno) from (select rollno from student) as sample;
+-------------+
| max(rollno) |
+-------------+
|          10 |
+-------------+
1 row in set (0.00 sec)
```

```
mysql> select max(rollno) from (select rollno from fees) as sample;
+-------------+
| max(rollno) |
+-------------+
|           9 |
+-------------+
1 row in set (0.00 sec)

mysql> select (rollno) from (select rollno from fees) as sample;
+--------+
| rollno |
+--------+
|      1 |
|      3 |
|      8 |
|      9 |
|      9 |
|      8 |
+--------+
6 rows in set (0.00 sec)
```

ROLLUP -: The ROLLUP in MySQL is a modifier used to produce the summary output, including extra rows that represent super-aggregate (higher-level) summary operations. It enables us to sum-up the output at multiple levels of analysis using a single query.

Syntax-:

select city,gender,sum(salary) from emp group by city,gender with rollup;

select gender,sum(salary) from emp group by gender with rollup;

Example -:

```
mysql> select city,gender,sum(salary) from emp
    ->  group by city,gender with rollup;
+------------+--------+-------------+
| city       | gender | sum(salary) |
+------------+--------+-------------+
| Chandigarh | female |       40000 |
| Chandigarh | mail   |       40000 |
| Chandigarh | NULL   |       80000 |
| Punjab     | femail |       20000 |
| Punjab     | mail   |       50000 |
| Punjab     | NULL   |       70000 |
| NULL       | NULL   |      150000 |
+------------+--------+-------------+
7 rows in set (0.00 sec)
```

PL/SQL Introduction

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Typically, each block performs a logical action in the program. A block has the following structure:

- **DECLARE**

declaration statements;

- **BEGIN**

executable statements

- **EXCEPTIONS**

exception handling statements

- **END;**

- Declare section starts with DECLARE keyword in which variables, constants, records as cursors can be declared which stores data temporarily. It basically consists definition of PL/SQL identifiers. This part of the code is optional.

- Execution section starts with BEGIN and ends with END keyword.This is a mandatory section and here the program logic is written to perform any task like loops and conditional statements. It supports all DML commands, DDL commands and SQL*PLUS built-in functions as well.

- Exception section starts with EXCEPTION keyword.This section is optional which contains statements that are executed when a run-time error occurs. Any exceptions can be handled in this section.

➢ **Let us see an example to see how to display a message using PL/SQL :**

Example-:

```
1  --PL/SQL BLOCK
2    BEGIN
3       DBMS_OUTPUT.PUT_LINE('HELLO VIKASH SHARMA');
4    END;
5
```

```
Statement processed.
HELLO VIKASH SHARMA
```

- **Variables -:**

Syntax for declaration of variables:

variable_name datatype [NOT NULL := value];

Example-:

```
1  --VARIABLES
2  DECLARE
3  X NUMBER(2);
4    BEGIN
5       X:=2;
6       DBMS_OUTPUT.PUT_LINE(X);
7    END;
8
```

```
Statement processed.
2
```

- **Assignment Operator**

Example-:

```
1  --VARIABLES
2  --ASSICNMENT OPERATOR
3  DECLARE
4  X NUMBER(2):=80;
5    BEGIN
6       DBMS_OUTPUT.PUT_LINE(X);
7    END;
```

```
Statement processed.
80
```

- **Constant Number**

Example-:

```
1  --VARIABLES
2  --ASSICNMENT OPERATOR
3  DECLARE
4  X NUMBER(2):=80;
5  CNST CONSTANT NUMBER:=14;
6    BEGIN
7       DBMS_OUTPUT.PUT_LINE(X+CNST);
8       DBMS_OUTPUT.PUT_LINE(X+CNST-X);
9    END;
10
```

```
Statement processed.
94
14
```

- **Existing Table Fetch Data In Pl Sql**

Example-:

```
1   --DEPT
2       --DEPTNO,DNAME,LOC
3
4   DECLARE
5   DNO NUMBER;
6   DN VARCHAR(20);
7   BEGIN
8   SELECT DEPTNO,DNAME INTO DNO,DN FROM DEPT WHERE DEPTNO=10;
9   DBMS_OUTPUT.PUT_LINE(DNO || '     ' || DN);
10  END;
```

```
Statement processed.
10    ACCOUNTING
```

- %TYPE -:

Example-:

```
1   --EMP
2       --EMPNO,ENAME,SALARY,HRD,DEPTNO
3
4   DECLARE
5   ENO EMP.EMPNO%TYPE;
6   ENM EMP.ENAME%TYPE;
7   BEGIN
8   SELECT EMPNO,ENAME INTO ENO,ENM FROM EMP WHERE EMPNO=7782;
9   DBMS_OUTPUT.PUT_LINE(ENO || '     ' || ENM);
10  END;
```

```
Statement processed.
7782    CLARK
```

- %ROWTYPE -:

Example-:

```
1   --EMP
2       --EMPNO,ENAME,SALARY,HRD,DEPTNO
3
4   DECLARE
5   ENM EMP%ROWTYPE;
6   BEGIN
7   SELECT * INTO ENM FROM EMP WHERE EMPNO=7782;
8   DBMS_OUTPUT.PUT_LINE(ENM.EMPNO || '     ' || ENM.ENAME ||'   '|| ENM.SAL);
9   END;
```

```
Statement processed.
7782    CLARK   2450
```