CC-215

PROJECT

DBMS

SUBMITTED TO:

SEHRISH KHAN

SUBMITTED BY:

SHANZAY MAHMOOD

NEELAM SHAHZADI

ROLL_NO:

110852 & 110828

Department of InformatIon technology
UnIversIty of the punjab lahore,
pakIstan

CONTENTS:

COMMANDS:

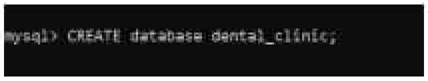
IMPLEMENTATIONS OF DIFFERENT COMMANDS ON MYSQL ;

○ DENTAL_CLINIC DATABASE:

First of all create database which you want to made as per according to your desire by using the syntax.

Syntax:

CREATE DATABASE DATABASE_NAME;

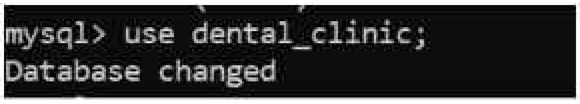


```
mysql> CREATE database dental_clinic;
```

- Use your already made database as by use the words as;

Syntax:

USE database_name:



```
mysql> use dental_clinic;
Database changed
```

➢CREATE command:

It's used for the creation of tables.

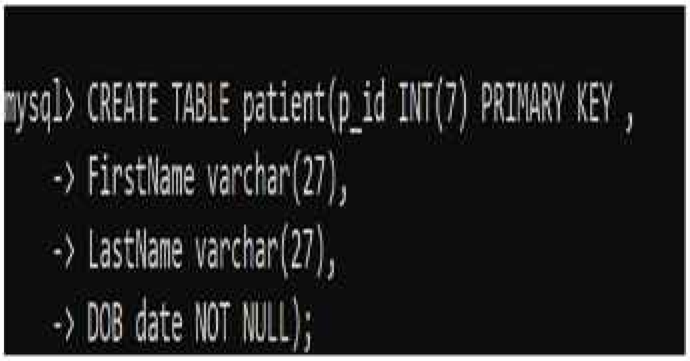
Evolving the following statement;

Syntax:

CREATE TABLE TABLE_NAME(COLUMN_1,COLUMN_2.....);

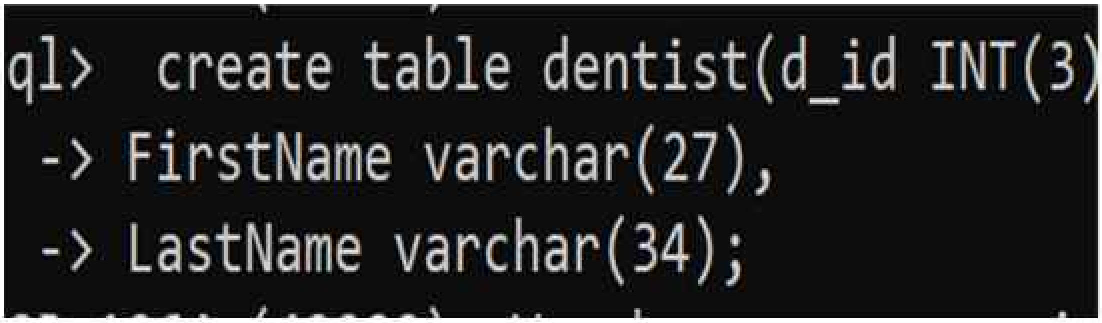
Example:

Patient table:



```
mysql> CREATE TABLE patient(p_id INT(7) PRIMARY KEY ,
    -> FirstName varchar(27),
    -> LastName varchar(27),
    -> DOB date NOT NULL);
```

Dentist Table;



```
ql>  create table dentist(d_id INT(3)
 -> FirstName varchar(27),
 -> LastName varchar(34);
```

Use of DESCRIBE COMMAND:

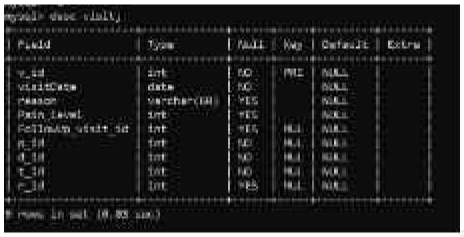
It's also can be used as the **DESC** shortly.

Syntax:

DESC TABLE_NAME;

Examples:

VISIT TABLE:



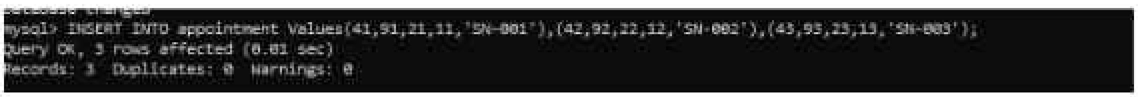
```
mysql> desc visit;
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| v_id              | int         | NO   | PRI | NULL    |       |
| visitDate         | date        | NO   |     | NULL    |       |
| reason            | varchar(64) | YES  |     | NULL    |       |
| Pain_level        | int         | YES  |     | NULL    |       |
| Followup_visit_id | int         | YES  | MUL | NULL    |       |
| p_id              | int         | NO   | MUL | NULL    |       |
| d_id              | int         | NO   | MUL | NULL    |       |
| t_id              | int         | NO   | MUL | NULL    |       |
| c_id              | int         | YES  | MUL | NULL    |       |
+-------------------+-------------+------+-----+---------+-------+
9 rows in set (0.03 sec)
```

➢ INSERT COMMAD:

Syntax:

INSERT INTO TABLE_NAME VALUES (COLUMN_1'S VALUES,C_2,.......);

Example:



```
mysql> INSERT INTO appointment Values(41,91,21,11,'SN-001'),(42,92,22,12,'SN-002'),(43,93,23,13,'SN-003');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

➢SELECT * command:

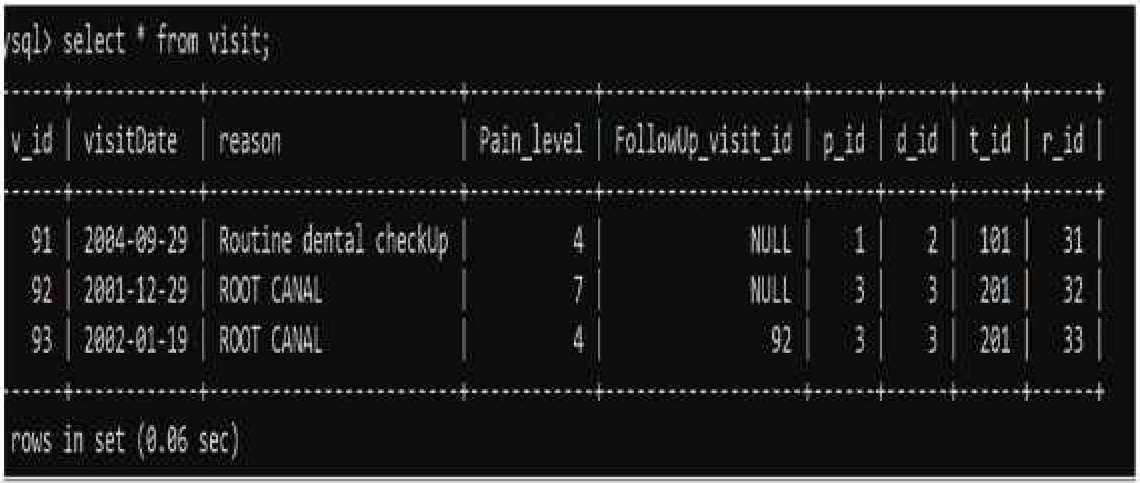
This command is used for the displaying all and every single attribute's **values** on the command prompt.

Syntax:

```
SELECT * FROM TABLE_NAME;
```

Example:

VISIT table:



```
ysql> select * from visit;
-----+------------+-----------------------+------------+------------------+------+------+------+------+
v_id | visitDate  | reason                | Pain_level | FollowUp_visit_id | p_id | d_id | t_id | r_id |
-----+------------+-----------------------+------------+------------------+------+------+------+------+
  91 | 2004-09-29 | Routine dental checkUp |          4 |             NULL |    1 |    2 |  101 |   31 |
  92 | 2001-12-29 | ROOT CANAL            |          7 |             NULL |    3 |    3 |  201 |   32 |
  93 | 2002-01-19 | ROOT CANAL            |          4 |               92 |    3 |    3 |  201 |   33 |
-----+------------+-----------------------+------------+------------------+------+------+------+------+
rows in set (0.06 sec)
```

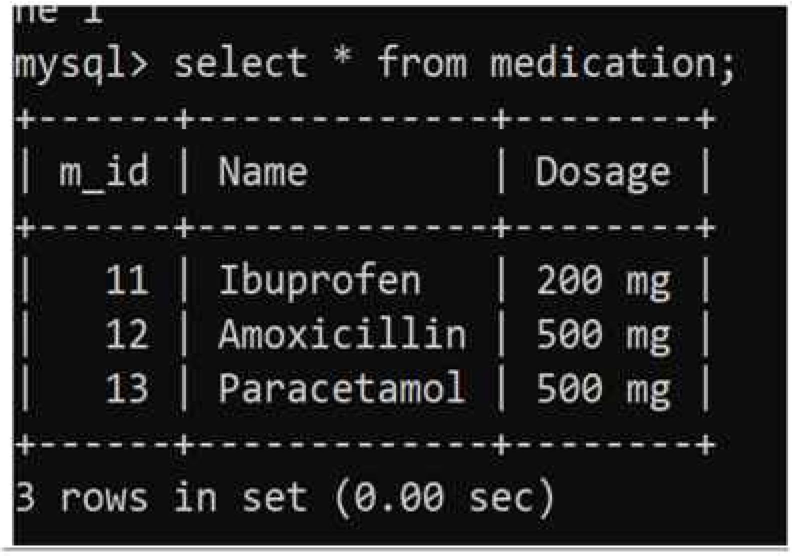
➢***SELECT one/multiple column :***

This command is beneficial at that support while you are dealing with to come out the single one and the multiple attributes from a table rather then all of the attributes. Only a **certain** or **specific** values will be come out by this.

Syntax;

```
SELECT COLUMN_NAME FROM TABLE_NAME;
```

Example:



```
mysql> select * from medication;
+------+-------------+--------+
| m_id | Name        | Dosage |
+------+-------------+--------+
|   11 | Ibuprofen   | 200 mg |
|   12 | Amoxicillin | 500 mg |
|   13 | Paracetamol | 500 mg |
+------+-------------+--------+
3 rows in set (0.00 sec)
```

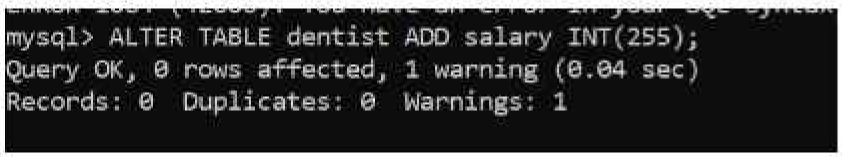
➢ALTER command:

Used to alter the table and including other attributes or **inserts** more data into the table.

Syntax:

ALTER TABLE TABLE_NAME ADD COLUMN_NAME CONSTRAINT;

Example:



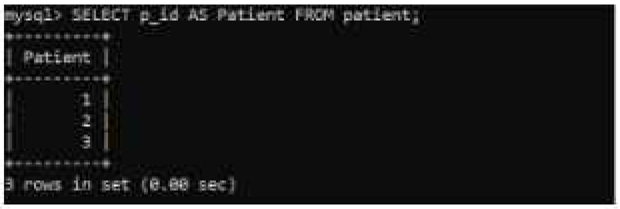
```
mysql> ALTER TABLE dentist ADD salary INT(255);
Query OK, 0 rows affected, 1 warning (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 1
```

AS COMMAND on MySQL:

This is basically used to make an alias of the attribute. It's helpful where you want to access the already attribute name with the new one as you want.

Syntax:

SELECT COLUMN_NAME AS ALIAS FROM TABLE_NAME;



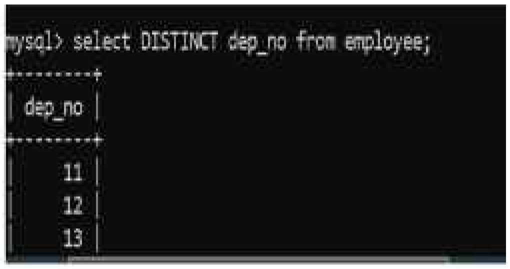
```
mysql> SELECT p_id AS Patient FROM patient;
+---------+
| Patient |
+---------+
|       1 |
|       2 |
|       3 |
+---------+
3 rows in set (0.00 sec)
```

➢ *DISTINCT COMMAND:*

This command is used for duplicacy removal from your table if exist.

Syntax:

SELECT DISTINCT COLUMN_NAME FROM TABLE_NAME;



```
ysql> select DISTINCT dep_no from employee;
+--------+
| dep_no |
+--------+
|     11 |
|     12 |
|     13 |
```

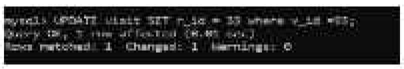
➢ WHERE clause in SQL:

Where command is work as such like fro the condition .

Syntax:

SELECT * FROM TABLE_NAME WHERE COLUMN="VALUE";

Example:



```
ysql> UPDATE visit SET r_id = 33 where v_id =93;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

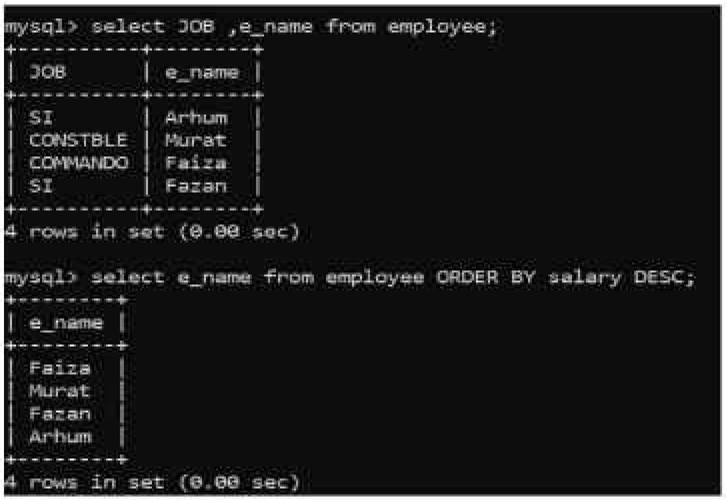
➢ *ORDER BY command:*

It's used for the ascending or descending order sorting. By default ascending sort is done by the complier.

Syntax;

SELECT COLUMN_NAME FROM TABLE_NAME ORDER BY COLUMN_2 DESC;

Example:



```
mysql> select JOB ,e_name from employee;
+----------+--------+
| JOB      | e_name |
+----------+--------+
| SI       | Arhum  |
| CONSTBLE | Murat  |
| COMMANDO | Faiza  |
| SI       | Fazan  |
+----------+--------+
4 rows in set (0.00 sec)

mysql> select e_name from employee ORDER BY salary DESC;
+--------+
| e_name |
+--------+
| Faiza  |
| Murat  |
| Fazan  |
| Arhum  |
+--------+
4 rows in set (0.00 sec)
```

➢ *OPERATORS IN SQL:*

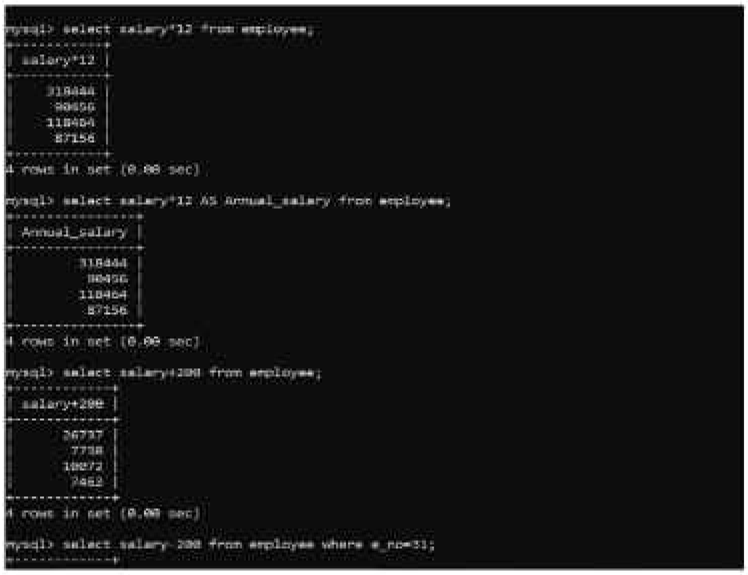
Arithmetic operators including (**+**,**-**,*****,**/**).

Syntax:

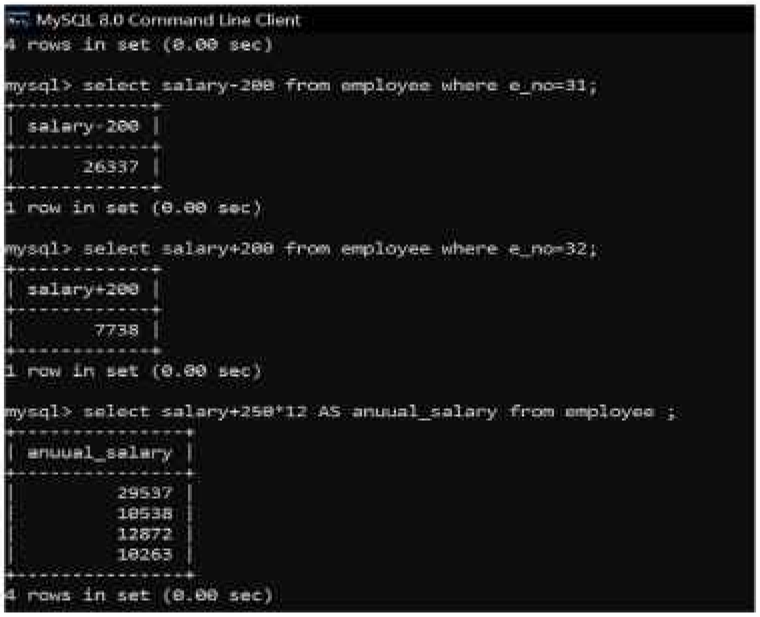
SELECT COLUMN_NAME OPERATOR FROM TABLE_NAME WHERE COLUMN_NAME="VALUES";

Or

SELECT COLUMN_NAME FROM TABLE COLUMN_NAME OPERATOR ANYOPERATION;



```
mysql> select salary*12 from employee;
+-----------+
| salary*12 |
+-----------+
|    318444 |
|     90456 |
|    118464 |
|     87156 |
+-----------+
4 rows in set (0.00 sec)

mysql> select salary*12 AS Annual_salary from employee;
+---------------+
| Annual_salary |
+---------------+
|        318444 |
|         90456 |
|        118464 |
|         87156 |
+---------------+
4 rows in set (0.00 sec)

mysql> select salary+200 from employee;
+------------+
| salary+200 |
+------------+
|      26737 |
|       7738 |
|      10072 |
|       7462 |
+------------+
4 rows in set (0.00 sec)

mysql> select salary-200 from employee where e_no=31;
+------------+
```



```
MySQL 8.0 Command Line Client
4 rows in set (0.00 sec)

mysql> select salary-200 from employee where e_no=31;
+------------+
| salary-200 |
+------------+
|      26337 |
+------------+
1 row in set (0.00 sec)

mysql> select salary+200 from employee where e_no=32;
+------------+
| salary+200 |
+------------+
|       7738 |
+------------+
1 row in set (0.00 sec)

mysql> select salary+250*12 AS anuual_salary from employee ;
+---------------+
| anuual_salary |
+---------------+
|         29537 |
|         10538 |
|         12872 |
|         10263 |
+---------------+
4 rows in set (0.00 sec)
```

➢**Relational operators:**

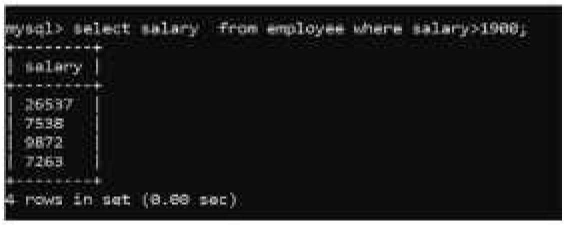
These operators including **(>,< ,>=,<=,==,!=)**

Syntax:

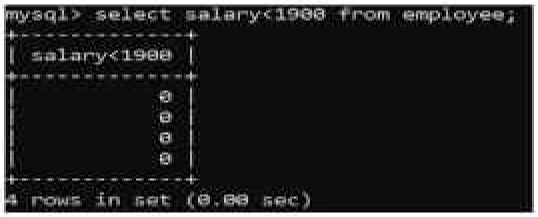
SELECT COLUMN_NAME OPERATOR FROM TABLE_NAME WHERE COLUMN_NAME="VALUES";

Or

SELECT COLUMN_NAME FROM TABLE_NAME WHERE COLUMN OPERATOR CONDITION;



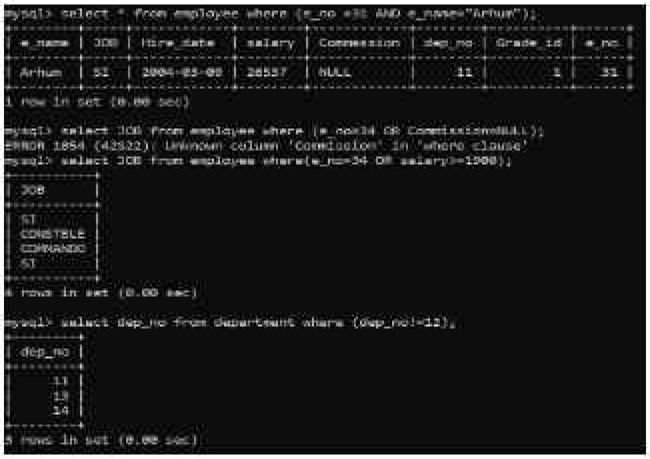
```
mysql> select salary  from employee where salary>1900;
+--------+
| salary |
+--------+
| 26537  |
| 7538   |
| 9872   |
| 7263   |
+--------+
4 rows in set (0.00 sec)
```



```
mysql> select salary<1900 from employee;
+-------------+
| salary<1900 |
+-------------+
|           0 |
|           0 |
|           0 |
|           0 |
+-------------+
4 rows in set (0.00 sec)
```

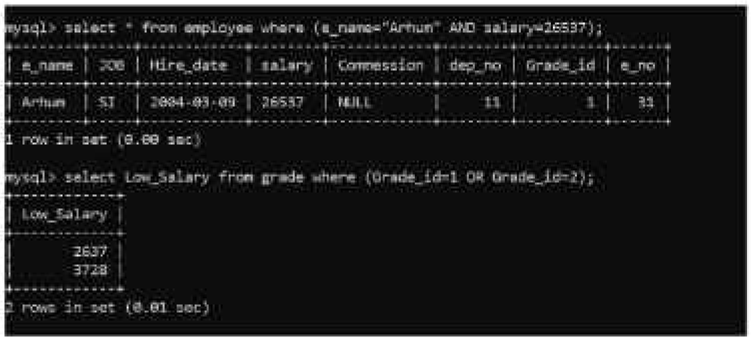
Logical Operators:

AND ,OR ,NOT.



```
mysql> select * from employee where (e_no =31 AND e_name="Arhum");
+--------+-----+------------+--------+------------+--------+----------+------+
| e_name | JOB | Hire_date  | salary | Commession | dep_no | Grade_id | e_no |
+--------+-----+------------+--------+------------+--------+----------+------+
| Arhum  | SI  | 2004-03-08 | 26537  | NULL       |     11 |        1 |   31 |
+--------+-----+------------+--------+------------+--------+----------+------+
1 row in set (0.00 sec)

mysql> select JOB from employee where (e_no=34 OR Commission=NULL);
ERROR 1054 (42S22): Unknown column 'Commission' in 'where clause'
mysql> select JOB from employee where(e_no=34 OR salary>=1900);
+----------+
| JOB      |
+----------+
| SI       |
| CONSTBLE |
| COMMANDO |
| SI       |
+----------+
4 rows in set (0.00 sec)

mysql> select dep_no from department where (dep_no!=12);
+--------+
| dep_no |
+--------+
|     11 |
|     13 |
|     14 |
+--------+
3 rows in set (0.00 sec)
```

➢ ***<u>SQL QUERIES BY AND,OR OPERATOR:</u>***



```
mysql> select * from employee where (e_name="Arhum" AND salary=26537);
+--------+-----+------------+--------+------------+--------+----------+------+
| e_name | JOB | Hire_date  | salary | Commession | dep_no | Grade_id | e_no |
+--------+-----+------------+--------+------------+--------+----------+------+
| Arhum  | SI  | 2004-03-09 | 26537  | NULL       |     11 |        1 |   31 |
+--------+-----+------------+--------+------------+--------+----------+------+
1 row in set (0.00 sec)

mysql> select Low_Salary from grade where (Grade_id=1 OR Grade_id=2);
+------------+
| Low_Salary |
+------------+
|       2637 |
|       3728 |
+------------+
2 rows in set (0.01 sec)
```

➢AND BETWEEN:

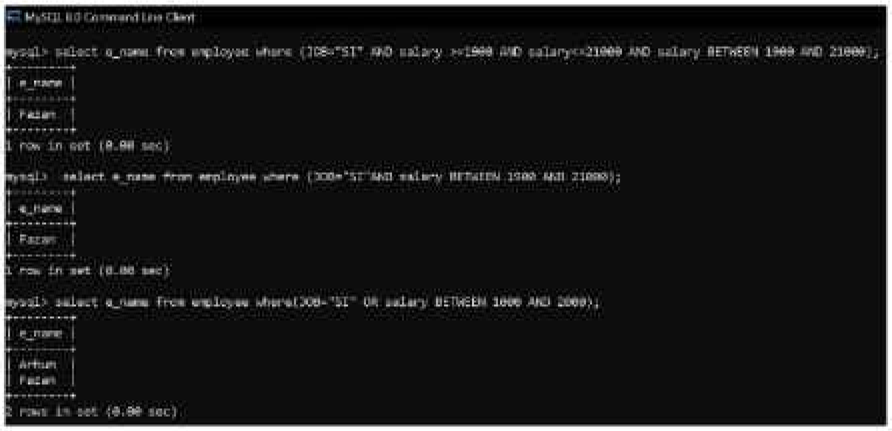
It's used where you want to evaluate the values from the range.Within that specific range.

Syntax:

SELECT * FROM TABLE_NAME WHERE(COND_1 AND COND_2 BETWEEN COND_3);

As you can use any column_name rather then the asterisk inside the query.

Example:



MySQL 8.0 Command Line Client

```
mysql> select e_name from employee where (JOB="SI" AND salary >=1000 AND salary<=21000 AND salary BETWEEN 1000 AND 21000);
+--------+
| e_name |
+--------+
| Fazan  |
+--------+
1 row in set (0.00 sec)

mysql>  select e_name from employee where (JOB="SI"AND salary BETWEEN 1000 AND 21000);
+--------+
| e_name |
+--------+
| Fazan  |
+--------+
1 row in set (0.00 sec)

mysql> select e_name from employee where(JOB="SI" OR salary BETWEEN 1000 AND 2000);
+--------+
| e_name |
+--------+
| Arhum  |
| Fazan  |
+--------+
2 rows in set (0.00 sec)
```

- OR BETWEEN is same as that of the AND BETWEEN.

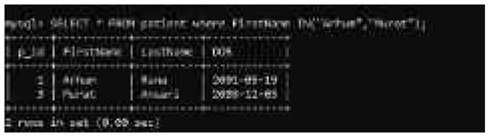
➢IN COMMAND:

It's use to come out the certain value from the list of the table. It's basically used to reduce length which is taken by multiple AND or OR operators.

Syntax:

SELECT COLUMN_NAME FROM TABLE_NAME WHERE COLUMN_NAME IN(V_1,V_2);

Examples:



```
mysql> SELECT * FROM patient where FirstName IN("Arhum","Murat");
+------+-----------+----------+------------+
| p_id | FirstName | LastName | DOB        |
+------+-----------+----------+------------+
|    1 | Arhum     | Rana     | 2001-09-19 |
|    3 | Murat     | Ansari   | 2000-12-05 |
+------+-----------+----------+------------+
2 rows in set (0.00 sec)
```

➢***LIKE COMMAND:***

As if you want to evaluate a specific personality from a table whose spelling you are specifying in the query or any other name whose end or start or at any middle element you'll be mentioned by you. This command is beneficial at that spot while you want to take a name whose character you'll not be known is advanced.

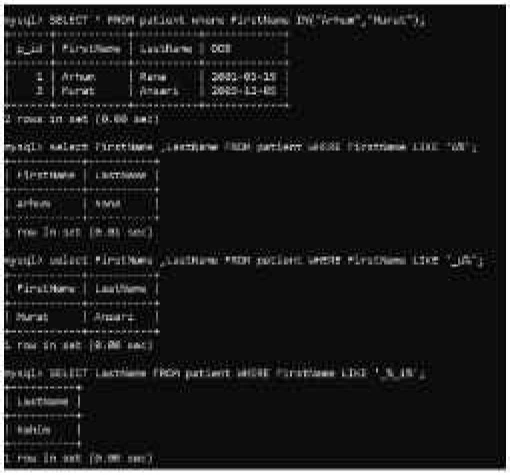
➢ ***Wildcard Characters;***

- **%**(Represents the single ,null or multiple characters)
- _(Represents a single character)

Syntax:

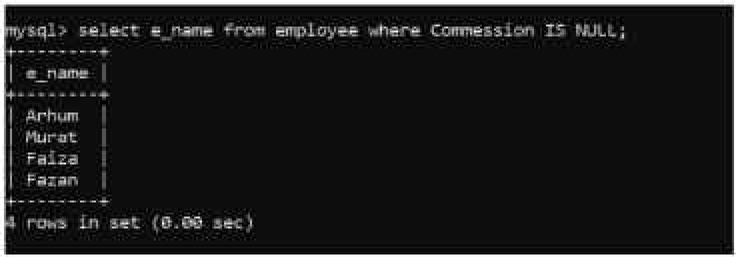
SELECT C_1,C_2 FROM TABLE_NAME WHERE C_NAME LIKE PATTERN;

Example:



```
mysql> SELECT * FROM patient where FirstName IN("Arhum","Murat");
+------+-----------+----------+------------+
| p_id | FirstName | LastName | DOB        |
+------+-----------+----------+------------+
|    1 | Arhum     | Rana     | 2001-03-15 |
|    3 | Murat     | Ansari   | 2003-12-05 |
+------+-----------+----------+------------+
2 rows in set (0.00 sec)

mysql> select FirstName ,LastName FROM patient WHERE FirstName LIKE '[illegible]';
+-----------+----------+
| FirstName | LastName |
+-----------+----------+
| Arhum     | Rana     |
+-----------+----------+
1 row in set (0.01 sec)

mysql> select FirstName ,LastName FROM patient WHERE FirstName LIKE '_[illegible]';
+-----------+----------+
| FirstName | LastName |
+-----------+----------+
| Murat     | Ansari   |
+-----------+----------+
1 row in set (0.00 sec)

mysql> SELECT LastName FROM patient WHERE FirstName LIKE '_[illegible]';
+----------+
| LastName |
+----------+
| Kabir    |
+----------+
1 row in set (0.00 sec)
```

➢ *NULL:*

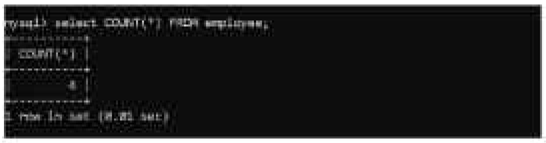


```
mysql> select e_name from employee where Commession IS NULL;
+--------+
| e_name |
+--------+
| Arhum  |
| Murat  |
| Faiza  |
| Fazan  |
+--------+
4 rows in set (0.00 sec)
```

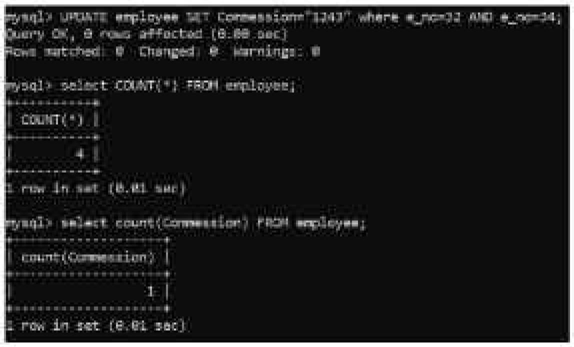
AGGREGRATION FUNCTIONS:

- COUNT(COLUMN_NAME)
- COUNT(*)
- AVG(COLUMN_NAME)
- MAX(COLUMN_NAME)
- MIN(COLUMN_NAME)
- SUM(COUMN_NAME)

➢ *COUNT COMMAND:*



```
mysql> select COUNT(*) FROM employee;
+----------+
| COUNT(*) |
+----------+
|        4 |
+----------+
1 row in set (0.01 sec)
```



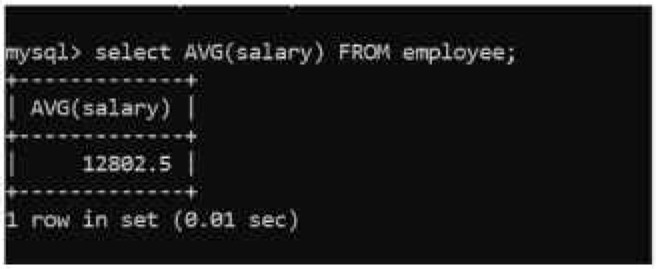
```
mysql> UPDATE employee SET Commession="1243" where e_no=32 AND e_no=34;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> select COUNT(*) FROM employee;
+----------+
| COUNT(*) |
+----------+
|        4 |
+----------+
1 row in set (0.01 sec)

mysql> select count(Commession) FROM employee;
+-------------------+
| count(Commession) |
+-------------------+
|                 1 |
+-------------------+
1 row in set (0.01 sec)
```

- **COUNT(*)** will display the all the colums which is include in your tables including the **null** values and also the **duplicate** values.
- **COUNT(column_name)** will display ONLY the **singly** values and not null vaues.

➢ ***AVERAGE FUNCTIONS:***

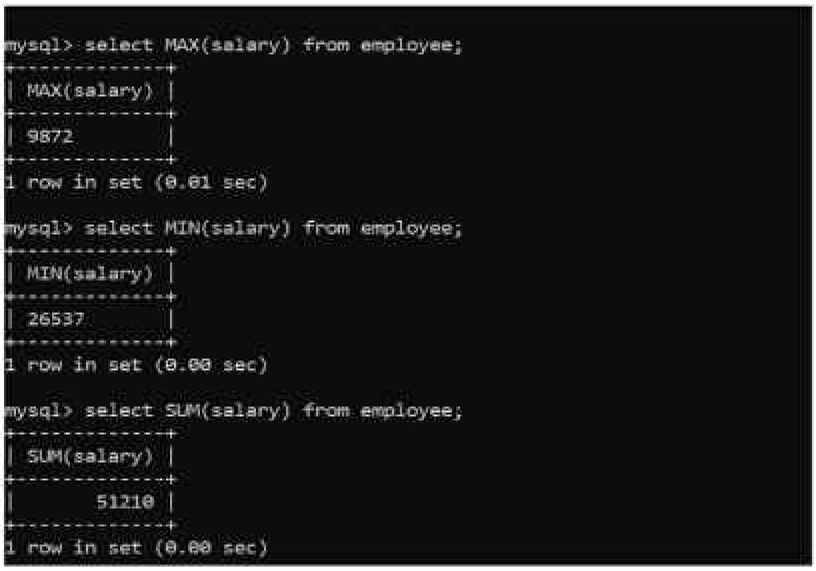
Syntax:

SELECT AVG(COLUMN_NAME) FROM TABLE_NAME;



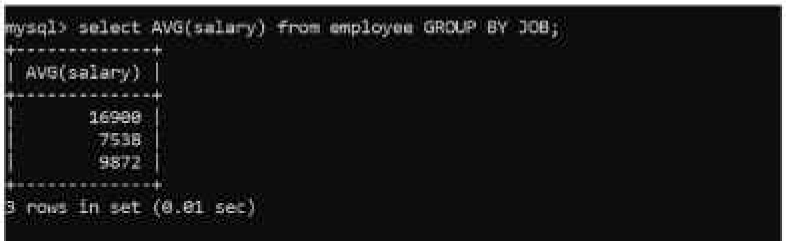
```
mysql> select AVG(salary) FROM employee;
+-------------+
| AVG(salary) |
+-------------+
|     12802.5 |
+-------------+
1 row in set (0.01 sec)
```

OTHERS functions:



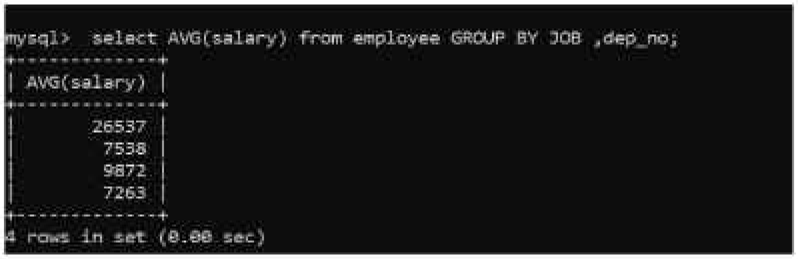
```
mysql> select MAX(salary) from employee;
+-------------+
| MAX(salary) |
+-------------+
| 9872        |
+-------------+
1 row in set (0.01 sec)

mysql> select MIN(salary) from employee;
+-------------+
| MIN(salary) |
+-------------+
| 26537       |
+-------------+
1 row in set (0.00 sec)

mysql> select SUM(salary) from employee;
+-------------+
| SUM(salary) |
+-------------+
|       51210 |
+-------------+
1 row in set (0.00 sec)
```

➢ **GROUP BY :**



```
mysql> select AVG(salary) from employee GROUP BY JOB;
+-------------+
| AVG(salary) |
+-------------+
|       16900 |
|        7538 |
|        9872 |
+-------------+
3 rows in set (0.01 sec)
```

- If you want to view groups of two different attributes.

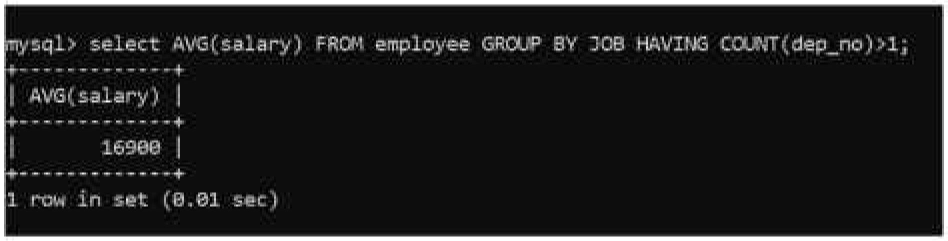


```
mysql>  select AVG(salary) from employee GROUP BY JOB ,dep_no;
+-------------+
| AVG(salary) |
+-------------+
|       26537 |
|        7538 |
|        9872 |
|        7263 |
+-------------+
4 rows in set (0.00 sec)
```

➢ <u>**HAVING COMMAND**</u> **:**

It's use as such to apply the conditions indirectly which is not done with the help of where command as while you are dealing with the **GROUP BY** in sql. So for the sack of that purpose you have to use the HAVING to filter out of data or attributes as you want from the table.

• <u>**Practical Implementations:**</u>



```
mysql> select AVG(salary) FROM employee GROUP BY JOB HAVING COUNT(dep_no)>1;
+-------------+
| AVG(salary) |
+-------------+
|       16900 |
+-------------+
1 row in set (0.01 sec)
```