Sql Database
Data Types
character - CHAR(10) - static memory allocation, VARCHAR(1000- dynamic memory allocation),txt,blob,

text	- tiny text,text,mediumtext,longtext
blob	-
numeric	- int,decimal(5,5) - 999.99
int	- int, tinyint,smallint,mediumint,Bigint
decimal	- Float(p,s),double(p,s) p-precisition,s-scale

temporal Data
Date , datetime , timestamp , year , Time

By valan
My sql uses many different datatypes broken into three categories

- Numeric
- Date and time
- String types

DDL - Data definition language -> deals with your table structure- Auto commit

- Create
- Rename
- Alter Add,Modify,rename, drop
- Drop
- truncate

DML- Data Manipulation Language->deals with your data inside table records - Manual commit

- Insert
- Delete
- Update
- Select

TCL - Translation control language-> commit rollback

- Commit
- RollBack
- Savepoint
- Grant
- Revoke

DDL commands and syntax

To create a database
create database mydb;

To show the available database
Show databases

To use that database
Use mydb;

To create a table
Create table tbl_employee();
alter table tbl_employee1 add gender char(1);

To Describe the table structure
desc tbl_employee;

Alter
To alter the table - to modify our table structure
    alter table tbl_employee1 add gender char(1);

To modify your column name
alter table tbl_employee1 modify gender char(10);

To rename your column name
alter table tbl_employee1 rename column gender to egender;

To drop your column
alter table tbl_employee1 drop column egender;

To drop our table
drop table tbl_employee1;

DML - commands

Insert query
To insert data into MySql table, you would need to use the sql INSERT into command

Syntax-generic sql syntax
```
    Insert into table_name( field1,field2,.....fieldN)
Values
(value1,value2….valueN);
```

Ex
```
   create table tbl_employee(eid int(5),ename varchar(20),esalary int (5));
```
`insert into tbl_employee values(101,'jeyavel','2000');`

—---

There are two ways to pass the null value to the table

Implicitly:
```
insert into tbl_employee values(103,null,'3000');
```

Explicitly:
```
insert into tbl_employee (eid,ename) values(104,'bala');
```

Output:
```
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
|  102 | harrish |    2000 |
|  103 | null    |    3000 |
|  104 | bala    |    NULL |
+------+---------+---------+
```

4 rows in set (0.00 sec)

Select Query

The SQL SELECT command is used to fetch data from the MySql Table.

Syntax:

Here is generic SQL syntax of SELECT command to fetch data from the MySql table

Select field1,field2,...fieldN
From table_name1,table_name2…
[WHERE Clause]
[offset M][LIMIT N]

Ex: Output

```
 select eid, esalary from tbl_employee;
+------+---------+
| eid  | esalary |
+------+---------+
|  101 |    2000 |
|  102 |    2000 |
|  103 |    3000 |
|  104 |    NULL |
+------+---------+
```

Ex:

```
 select * from tbl_employee where esalary >2000;
+------+-------+---------+
| eid  | ename | esalary |
+------+-------+---------+
|  103 | null  |    3000 |
+------+-------+---------+
```

```
 select * from tbl_employee where esalary >=2000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
|  102 | harrish |    2000 |
|  103 | null    |    3000 |
+------+---------+---------+
```

EX:

```
mysql> select * from tbl_employee where ename ='bala';
+------+-------+---------+
| eid  | ename | esalary |
+------+-------+---------+
|  104 | bala  |    NULL |
+------+-------+---------+
1 row in set (0.00 sec)
```

```
mysql> select * from tbl_employee where ename !='bala';
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
|  102 | harrish |    2000 |
|  103 | null    |    3000 |
+------+---------+---------+
3 rows in set (0.00 sec)
```

By using relational operators we cannot compare null values

```
select * from tbl_employee where ename is null;
select * from tbl_employee where ename is not null;

select * from tbl_employee where esalary is null;
select * from tbl_employee where esalary is not null;

select * from tbl_employee where ename is not null and esalary =3000;
select * from tbl_employee where ename is not null or esalary =3000;
```

Is Null < = >

Comparison or Relational operators -> these operators is used for delete as well as update query

In operator

Output

```
 select * from tbl_employee where eid in (101,103,106);
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
|  103 | null    |    3000 |
+------+---------+---------+
2 rows in set (0.00 sec)
```

```
select * from tbl_employee where eid not in (101,103,106);
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | harrish |    2000 |
|  104 | bala    |    NULL |
|  105 | NULL    |    3000 |
+------+---------+---------+
3 rows in set (0.00 sec)
```

—---

Between Operator

```
select * from tbl_employee where esalary between 2000 and 4000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
|  102 | harrish |    2000 |
|  103 | null    |    3000 |
|  105 | NULL    |    3000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where esalary not between 2000 and 4000;
Empty set (0.00 sec)
```

I will display if we have negative values in the table

```
select * from tbl_employee where esalary  between 4000 and 2000;
Empty set (0.00 sec)
 select * from tbl_employee where ename like 'b%';
+------+-------+---------+
| eid  | ename | esalary |
+------+-------+---------+
|  104 | bala  |    NULL |
+------+-------+---------+
```

```
select * from tbl_employee where ename like '_a%';
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | harrish |    2000 |
|  104 | bala    |    NULL |
+------+---------+---------+
```

```
select * from tbl_employee where ename like '__y%';
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
+------+---------+---------+
```

—---

Update query
You can update one or more fields altogether.

- You can specify any condition using the where clause.
- You can specify any condition using where clause
- You can update the values in a single table at a time

EX:
There may be a requirement where the existing data in my sql table needs to be modified. You can do so by using the sql UPDATE command. This will modify any field of any MySQL table.
Syntax:

```
UPDATE table_name SET field1= new-value1,field2 = new-value2;
```

The WHERE cause is vary useful when you want to delete selected rows in a table

EX:

```
 update tbl_employee set esalary =0;
Query OK, 5 rows affected (0.02 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> select * from tbl_employee ;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |       0 |
|  102 | harrish |       0 |
|  103 | null    |       0 |
|  104 | bala    |       0 |
|  105 | NULL    |       0 |
+------+---------+---------+
5 rows in set (0.00 sec)
```

```
mysql> select @@autocommit from dual;
+--------------+
| @@autocommit |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)

mysql> set autocommit=0;
Query OK, 0 rows affected (0.02 sec)

mysql> select @@autocommit ;
+--------------+
| @@autocommit |
+--------------+
|            0 |
+--------------+
1 row in set (0.00 sec)
```

```
mysql> update tbl_employee set esalary =1000 where eid=101;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from tbl_employee ;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    1000 |
|  102 | harrish |       0 |
|  103 | null    |       0 |
|  104 | bala    |       0 |
|  105 | NULL    |       0 |
+------+---------+---------+
5 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from tbl_employee ;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |       0 |
|  102 | harrish |       0 |
|  103 | null    |       0 |
|  104 | bala    |       0 |
|  105 | NULL    |       0 |
+------+---------+---------+
5 rows in set (0.00 sec)
```

```
mysql> update tbl_employee set esalary =2000;
Query OK, 5 rows affected (0.00 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> select * from tbl_employee ;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
|  102 | harrish |    2000 |
|  103 | null    |    2000 |
|  104 | bala    |    2000 |
|  105 | NULL    |    2000 |
+------+---------+---------+
5 rows in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> update tbl_employee set ename= null,esalary= 0 where eid in (101,103,106);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from tbl_employee ;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | NULL    |       0 |
|  102 | harrish |    2000 |
|  103 | NULL    |       0 |
|  104 | bala    |    2000 |
|  105 | NULL    |    2000 |
+------+---------+---------+
5 rows in set (0.00 sec)

mysql> rollback;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from tbl_employee ;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    2000 |
|  102 | harrish |    2000 |
|  103 | null    |    2000 |
|  104 | bala    |    2000 |
|  105 | NULL    |    2000 |
+------+---------+---------+
5 rows in set (0.00 sec)
```

Delete Query:
If you want to delete a record from any MySql table, then you can use the Sql command DELETE FROM. you can use this command at the mySql> prompt as well as in any script like PHP.

Delete is in under the DML
Truncate and drop is under DDL
So we can use rollback in only delete keyword

DELETE FROM table_name[where Clause]

- If the WHERE clause is not specified , then all the records will be deleted from the given MySql table,
-
- You can specify any condition using the WHERE clause.
-
- You can delete records in a single table at a time.

Like clause

We have seen the SQL SELECT command to fetch data from the MySql table.We can also use a conditional clause called as the Where to select the required records.

A where clause with the ‘equal to’ sign (=) works fine where we want to do an exit match. Like if”employee_name =’sanjay’”.But there may be a requirement where we want to filter out all the results where employee_name should contain ”jay”. This can be handled using SQL LIKE clause along with the WHERE clause .
Syntax

SELECT field1,field2,...fieldN table_name1,table_name2…
WHERE field1 LIKE condition1 [AND[OR]] field2 = ‘somevalue’

- You can specify any condition using the WHERE clause.
- You can use the LIKE clause along with the WHERE clause.
- You can use the LIKE clause in place of the **equals to** sign.
- When LIKE is used along with % sign then it will work like a meta character search.
- You can specify more than one condition using **AND** or **OR** operators .
- A WHERE.. LIKE clause can be used along with DELETE or UPDATE SQL command also to specify a condition .

Sorting Results
- We have seen the SQL SELECT command to fetch data from a MySql table.When you selected rows. The MySql server is free to return them in any order , unless you instruct it otherwise by saying how to sort the result.
-
- But , you sirt a result set by adding an ORDER By clause the column or columns which you want to sort

Syntax:
The following code block is an generic SQL syntax of the SELECT command along with the ORDER BY clause to sort the data from a MySql table.

SELECT field1,field2…fieldN table_name1, table_name2…
ORDER BY , [field2…][ASC[DESC]]

- You can sort the returned on any field, if that field is being listed out.
- You can sort the result on more than one field.
- You can use the keyword ASC or DESC to get result in ascending or descending order , it's the ascending order.
- You can use the WHERE…LIKE clause in the usual way to put a condition

Ex

```
mysql> select * from tbl_employee order by eid;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | jeyavel |    4000 |
|  102 | harrish |    2000 |
|  104 | bala    |    2000 |
|  105 | NULL    |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee order by eid desc;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  105 | NULL    |    2000 |
|  104 | bala    |    2000 |
|  102 | harrish |    2000 |
|  101 | jeyavel |    4000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

My SQL Null Values

We have the SQL SELECT command along with the WHERE clause to fetch data form a my SQL table , but when try to give a condition, which compares the field or the column value to NULL , it does not work properly,
To handle such a situation , MySql provides three operators -

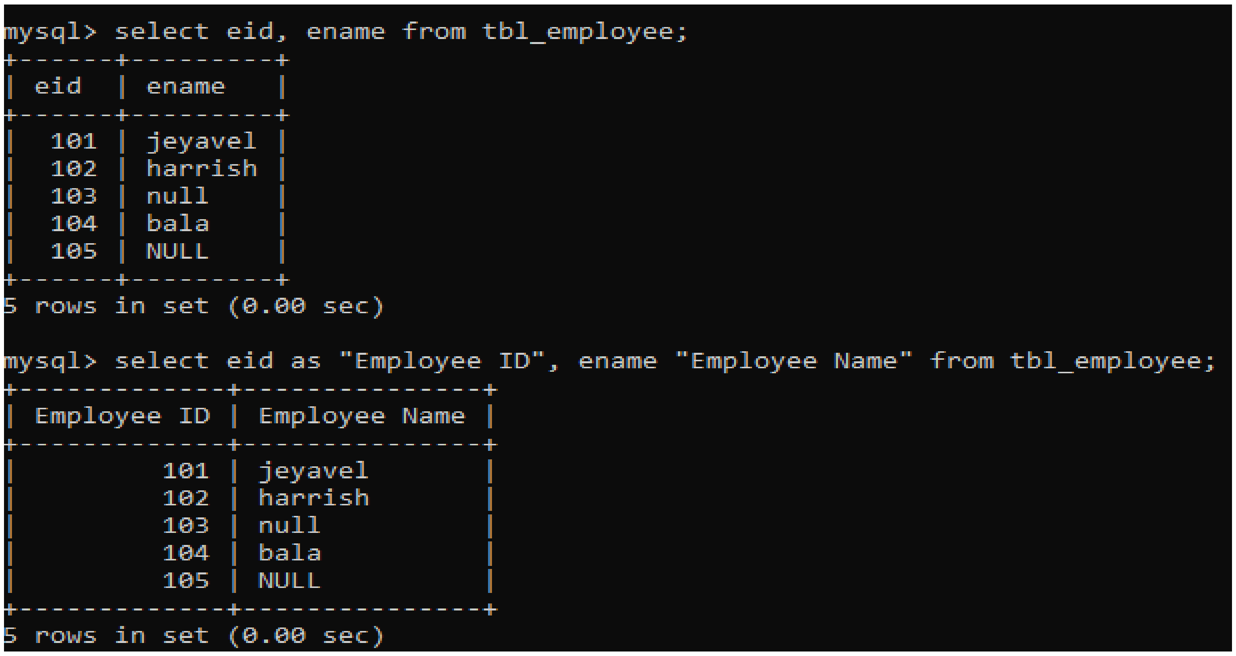
- **IS NULL** - This operator returns true , if the column value is NULL.
- **Is not NULL** - This operator return true , if the column value is not NULL .
- **< = >** - This operator compares values , which (unlike the = operator) is true even for two NULL Values .

The conditions involving NULL are special , you cannot use = NULL or ! = NULL to look for NULL values in columns . such comparisons fail because it is impossible to tell whether they are not true or not . Sometimes , even NULL = NULL fails,
To Look columns that are not NULL , use **IS NULL** or **IS NOT NULL.**

Alias in My sql

It is for display purpose
It never modify your table

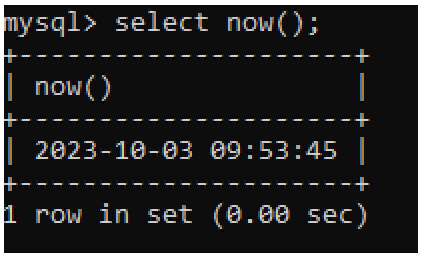
Example :



```
mysql> select eid, ename from tbl_employee;
+------+---------+
| eid  | ename   |
+------+---------+
|  101 | jeyavel |
|  102 | harrish |
|  103 | null    |
|  104 | bala    |
|  105 | NULL    |
+------+---------+
5 rows in set (0.00 sec)

mysql> select eid as "Employee ID", ename "Employee Name" from tbl_employee;
+-------------+---------------+
| Employee ID | Employee Name |
+-------------+---------------+
|         101 | jeyavel       |
|         102 | harrish       |
|         103 | null          |
|         104 | bala          |
|         105 | NULL          |
+-------------+---------------+
5 rows in set (0.00 sec)
```

Now() query

This query is used to achieve the current system timing in MySql



```
mysql> select now();
+---------------------+
| now()               |
+---------------------+
| 2023-10-03 09:53:45 |
+---------------------+
1 row in set (0.00 sec)
```

—--

Sub Query - this is very very important

Query inside another query (nested query) you can create n number of queries like this

At first inner query will be executed based on that the outer query will executed

You can write a query within in a query in MySql this is known as a subquery or, an inner query Or, a Nested query . Usually ,a subquery is embedded within the where clause.

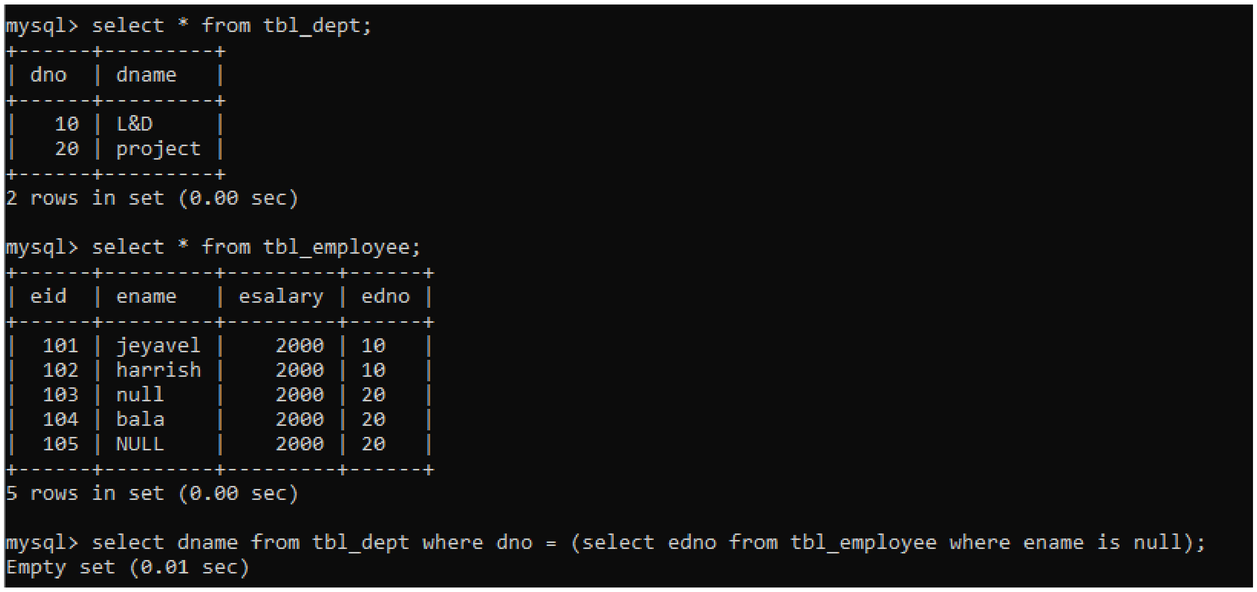
A subquery is used to return data that will be used in the main query as a condition
Types of subQuery : **Single-row subquery, multiple row subquery, multiple column subquery, correlated subquery, and nested subquery.**

It we can apply on

- **select**
- **update**
- **insert**
- **delete**

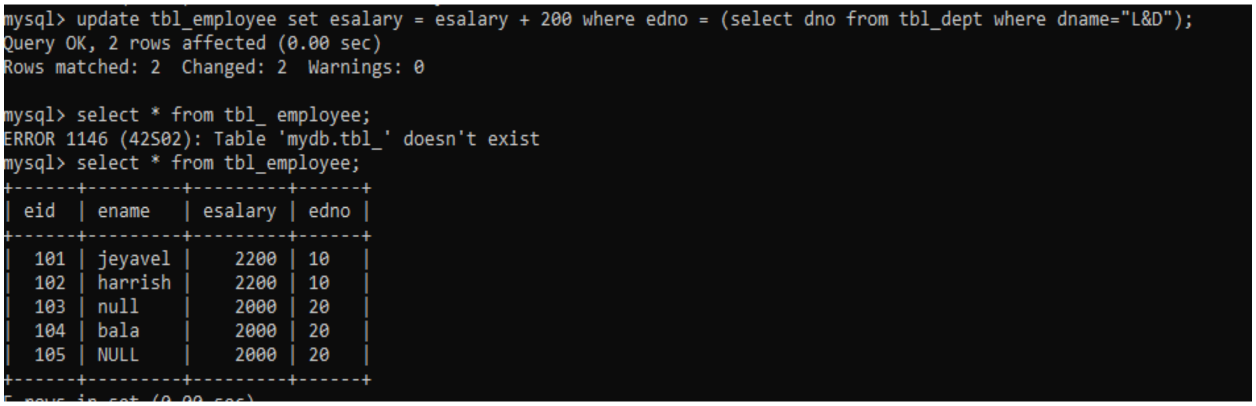
Single row subquery

Ex



```
mysql> select * from tbl_dept;
+------+---------+
| dno  | dname   |
+------+---------+
|   10 | L&D     |
|   20 | project |
+------+---------+
2 rows in set (0.00 sec)

mysql> select * from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | jeyavel |    2000 | 10   |
|  102 | harrish |    2000 | 10   |
|  103 | null    |    2000 | 20   |
|  104 | bala    |    2000 | 20   |
|  105 | NULL    |    2000 | 20   |
+------+---------+---------+------+
5 rows in set (0.00 sec)

mysql> select dname from tbl_dept where dno = (select edno from tbl_employee where ename is null);
Empty set (0.01 sec)
```

update tbl_employee set esalary = esalary + 200 where edno = (select dno from tbl_dept where dname="L&D");

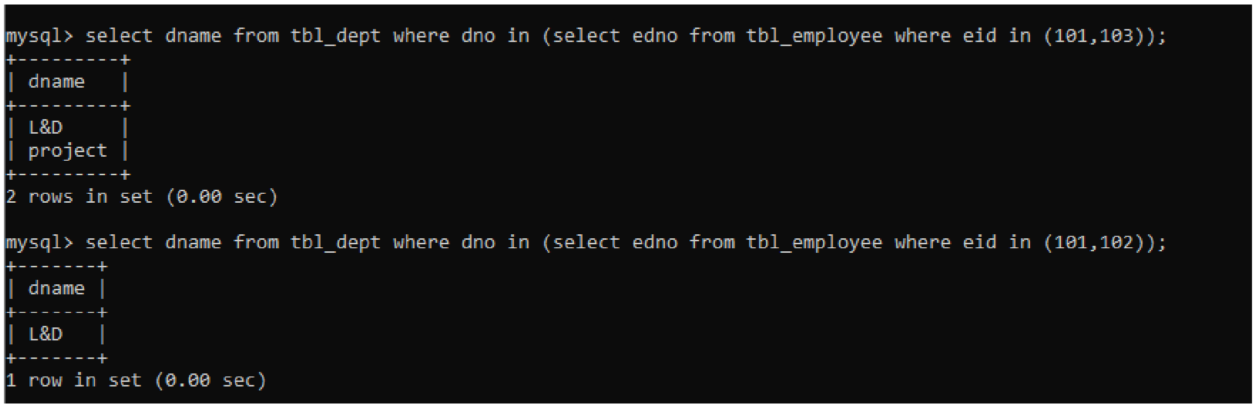


```
mysql> update tbl_employee set esalary = esalary + 200 where edno = (select dno from tbl_dept where dname="L&D");
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from tbl_ employee;
ERROR 1146 (42S02): Table 'mydb.tbl_' doesn't exist
mysql> select * from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | jeyavel |    2200 | 10   |
|  102 | harrish |    2200 | 10   |
|  103 | null    |    2000 | 20   |
|  104 | bala    |    2000 | 20   |
|  105 | NULL    |    2000 | 20   |
+------+---------+---------+------+
```

Multi row / Value Subquery

In multi row subquery we use instead of (**=**) sign we use **in** in this query to display multiple values

Ex



```
mysql> select dname from tbl_dept where dno in (select edno from tbl_employee where eid in (101,103));
+---------+
| dname   |
+---------+
| L&D     |
| project |
+---------+
2 rows in set (0.00 sec)

mysql> select dname from tbl_dept where dno in (select edno from tbl_employee where eid in (101,102));
+-------+
| dname |
+-------+
| L&D   |
+-------+
1 row in set (0.00 sec)
```

—---

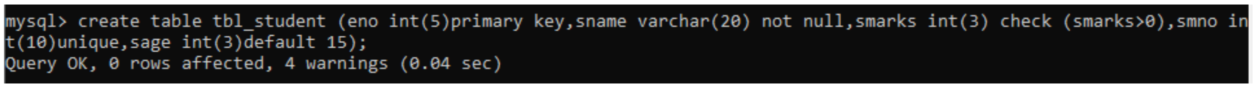
Constrains

- The constraint in MySql is used to specify the rule that allows or resists what values /data will be stored in the table.
-
- They provide a suitable method to ensure data accuracy and integrity inside the table

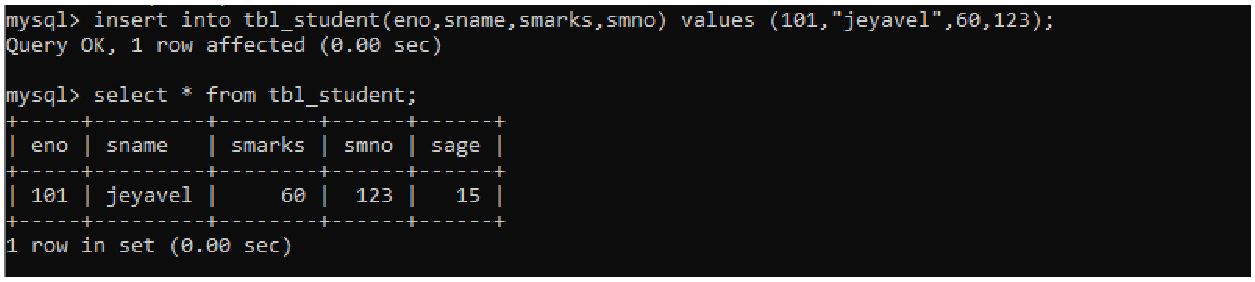
Used in MySql

- **NOT NULL**
- **CHECK**
- **DEFAULT**
- **PRIMARY KEY**
- **AUTO_INCREMENT**
- **UNIQUE**

Ex

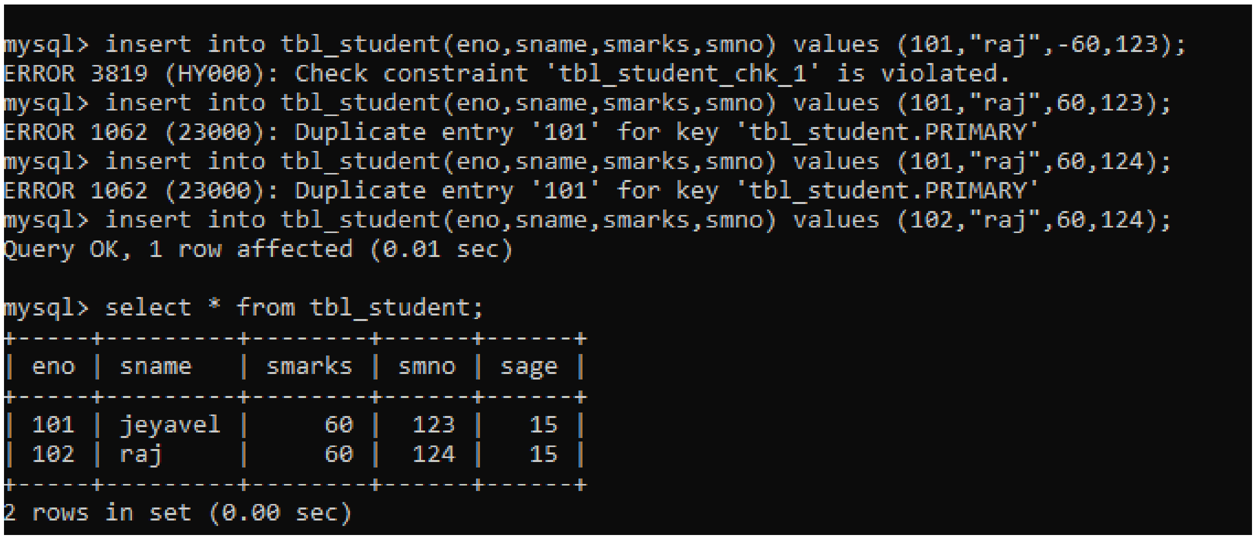


```
mysql> create table tbl_student (eno int(5)primary key,sname varchar(20) not null,smarks int(3) check (smarks>0),smno in
t(10)unique,sage int(3)default 15);
Query OK, 0 rows affected, 4 warnings (0.04 sec)
```



```
mysql> insert into tbl_student(eno,sname,smarks,smno) values (101,"jeyavel",60,123);
Query OK, 1 row affected (0.00 sec)

mysql> select * from tbl_student;
+-----+---------+--------+------+------+
| eno | sname   | smarks | smno | sage |
+-----+---------+--------+------+------+
| 101 | jeyavel |     60 |  123 |   15 |
+-----+---------+--------+------+------+
1 row in set (0.00 sec)
```

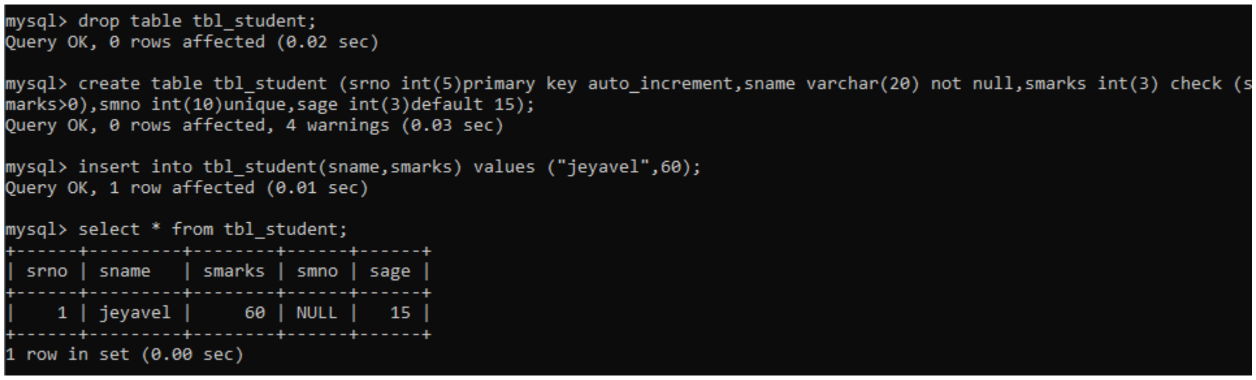
To check the primary key, Not Null , check , Default
Output :



```
mysql> insert into tbl_student(eno,sname,smarks,smno) values (101,"raj",-60,123);
ERROR 3819 (HY000): Check constraint 'tbl_student_chk_1' is violated.
mysql> insert into tbl_student(eno,sname,smarks,smno) values (101,"raj",60,123);
ERROR 1062 (23000): Duplicate entry '101' for key 'tbl_student.PRIMARY'
mysql> insert into tbl_student(eno,sname,smarks,smno) values (101,"raj",60,124);
ERROR 1062 (23000): Duplicate entry '101' for key 'tbl_student.PRIMARY'
mysql> insert into tbl_student(eno,sname,smarks,smno) values (102,"raj",60,124);
Query OK, 1 row affected (0.01 sec)

mysql> select * from tbl_student;
+-----+---------+--------+------+------+
| eno | sname   | smarks | smno | sage |
+-----+---------+--------+------+------+
| 101 | jeyavel |     60 |  123 |   15 |
| 102 | raj     |     60 |  124 |   15 |
+-----+---------+--------+------+------+
2 rows in set (0.00 sec)
```

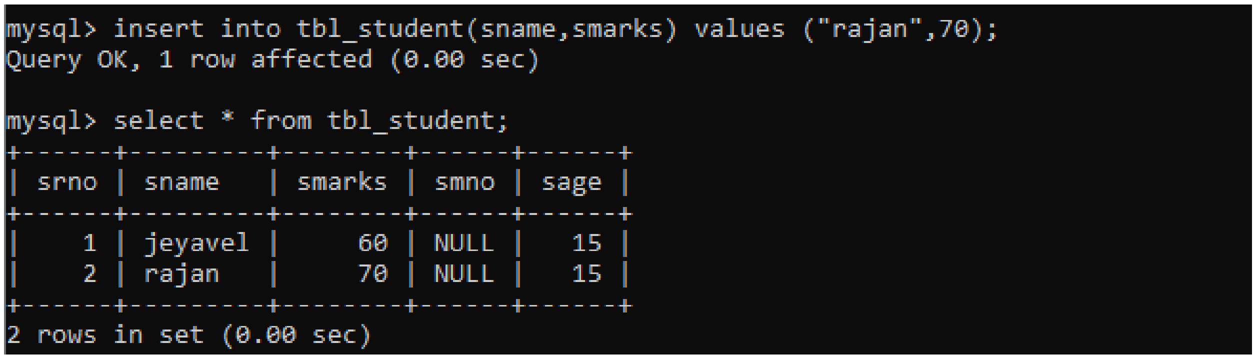
Auto increment

create table tbl_student (srno int(5)primary key **auto_increment**,sname varchar(20) not null,smarks int(3) check (smarks>0),smno int(10)unique,sage int(3)default 15);
Ex



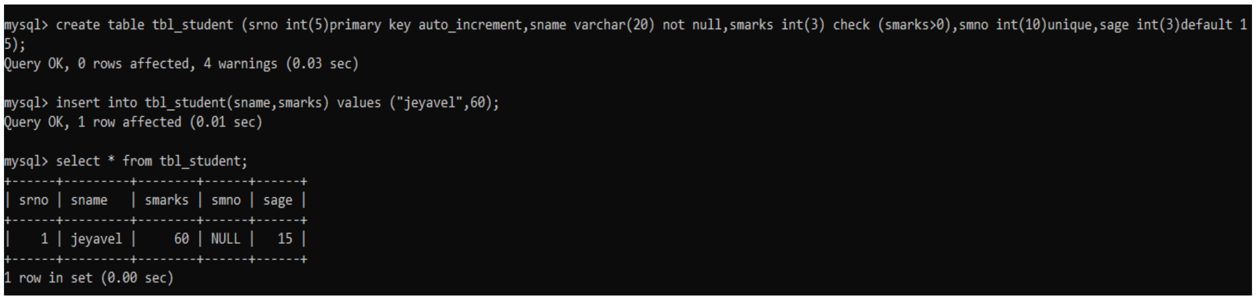
```
mysql> drop table tbl_student;
Query OK, 0 rows affected (0.02 sec)

mysql> create table tbl_student (srno int(5)primary key auto_increment,sname varchar(20) not null,smarks int(3) check (s
marks>0),smno int(10)unique,sage int(3)default 15);
Query OK, 0 rows affected, 4 warnings (0.03 sec)

mysql> insert into tbl_student(sname,smarks) values ("jeyavel",60);
Query OK, 1 row affected (0.01 sec)

mysql> select * from tbl_student;
+------+---------+--------+------+------+
| srno | sname   | smarks | smno | sage |
+------+---------+--------+------+------+
|    1 | jeyavel |     60 | NULL |   15 |
+------+---------+--------+------+------+
1 row in set (0.00 sec)
```

Ex 2:



```
mysql> insert into tbl_student(sname,smarks) values ("rajan",70);
Query OK, 1 row affected (0.00 sec)

mysql> select * from tbl_student;
+------+---------+--------+------+------+
| srno | sname   | smarks | smno | sage |
+------+---------+--------+------+------+
|    1 | jeyavel |     60 | NULL |   15 |
|    2 | rajan   |     70 | NULL |   15 |
+------+---------+--------+------+------+
2 rows in set (0.00 sec)
```

Primary key



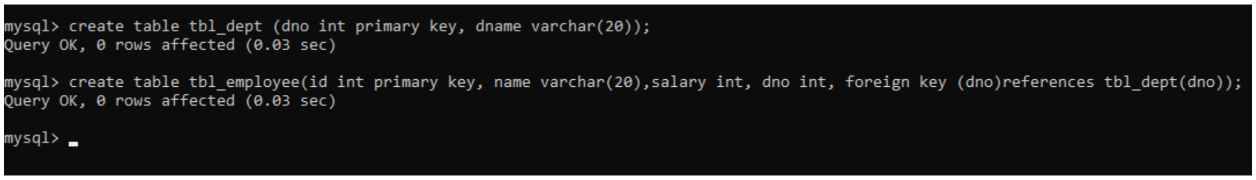
```
mysql> create table tbl_student (srno int(5)primary key auto_increment,sname varchar(20) not null,smarks int(3) check (smarks>0),smno int(10)unique,sage int(3)default 1
5);
Query OK, 0 rows affected, 4 warnings (0.03 sec)

mysql> insert into tbl_student(sname,smarks) values ("jeyavel",60);
Query OK, 1 row affected (0.01 sec)

mysql> select * from tbl_student;
+------+---------+--------+------+------+
| srno | sname   | smarks | smno | sage |
+------+---------+--------+------+------+
|    1 | jeyavel |     60 | NULL |   15 |
+------+---------+--------+------+------+
1 row in set (0.00 sec)
```

Primary key & foreign key relationship

Ex

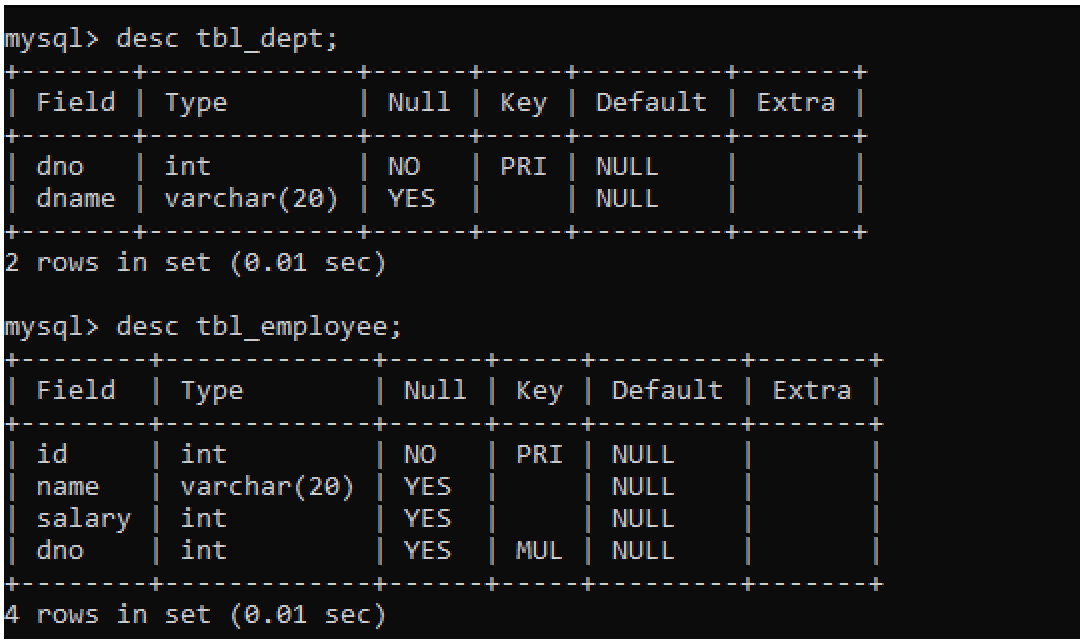
One table have multiple relationships with multiple table

First fill the child table and then update the parent table



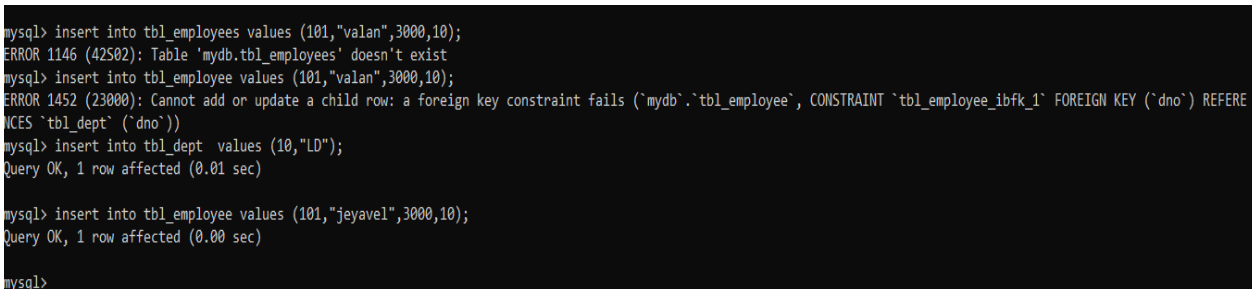
```
mysql> create table tbl_dept (dno int primary key, dname varchar(20));
Query OK, 0 rows affected (0.03 sec)

mysql> create table tbl_employee(id int primary key, name varchar(20),salary int, dno int, foreign key (dno)references tbl_dept(dno));
Query OK, 0 rows affected (0.03 sec)

mysql> _
```

Output



```
mysql> desc tbl_dept;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| dno   | int         | NO   | PRI | NULL    |       |
| dname | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> desc tbl_employee;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int         | NO   | PRI | NULL    |       |
| name   | varchar(20) | YES  |     | NULL    |       |
| salary | int         | YES  |     | NULL    |       |
| dno    | int         | YES  | MUL | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

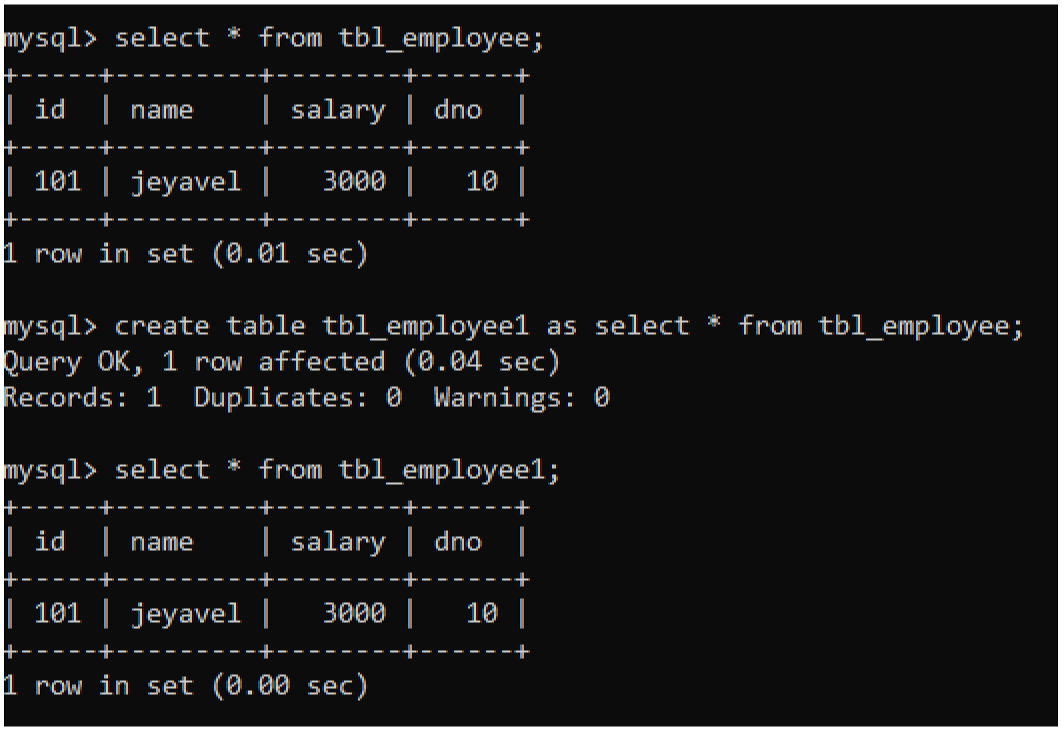
Error while updating the parent first



```
mysql> insert into tbl_employees values (101,"valan",3000,10);
ERROR 1146 (42S02): Table 'mydb.tbl_employees' doesn't exist
mysql> insert into tbl_employee values (101,"valan",3000,10);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`mydb`.`tbl_employee`, CONSTRAINT `tbl_employee_ibfk_1` FOREIGN KEY (`dno`) REFERE
NCES `tbl_dept` (`dno`))
mysql> insert into tbl_dept  values (10,"LD");
Query OK, 1 row affected (0.01 sec)

mysql> insert into tbl_employee values (101,"jeyavel",3000,10);
Query OK, 1 row affected (0.00 sec)

mysql>
```

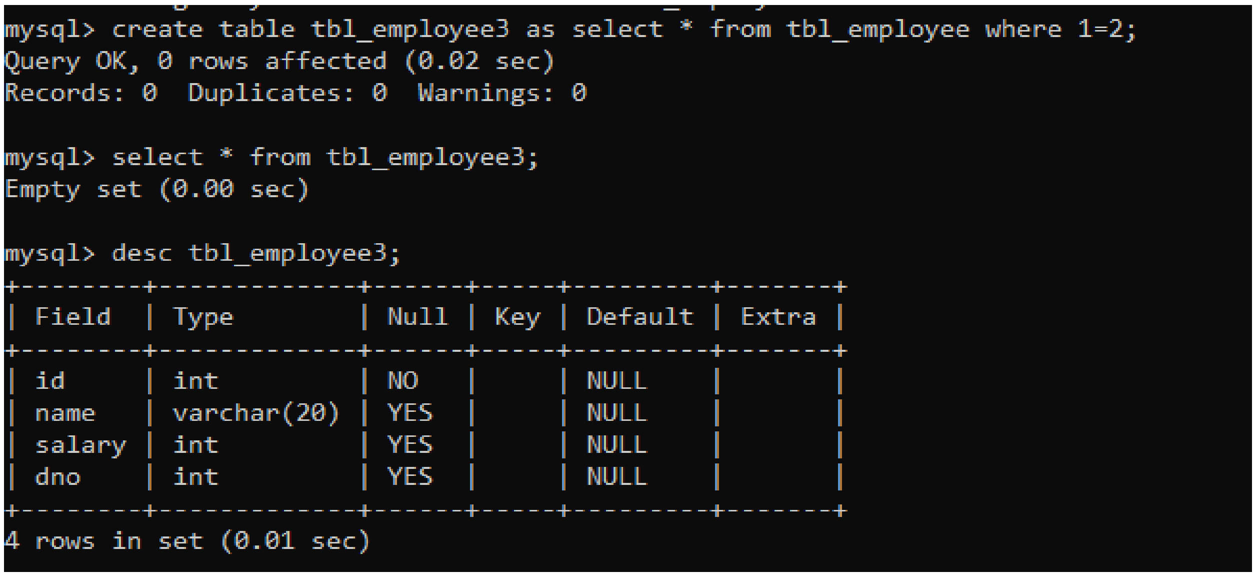
How to create a new table by copying the existing table

EX: create table tbl_employee1 as select * from tbl_employee;



```
mysql> select * from tbl_employee;
+-----+---------+--------+------+
| id  | name    | salary | dno  |
+-----+---------+--------+------+
| 101 | jeyavel |   3000 |   10 |
+-----+---------+--------+------+
1 row in set (0.01 sec)

mysql> create table tbl_employee1 as select * from tbl_employee;
Query OK, 1 row affected (0.04 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> select * from tbl_employee1;
+-----+---------+--------+------+
| id  | name    | salary | dno  |
+-----+---------+--------+------+
| 101 | jeyavel |   3000 |   10 |
+-----+---------+--------+------+
1 row in set (0.00 sec)
```

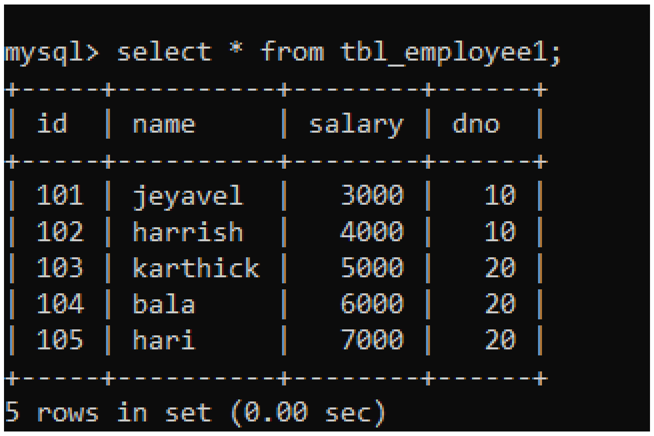
Use where we can insert false statements and get the structure of the table

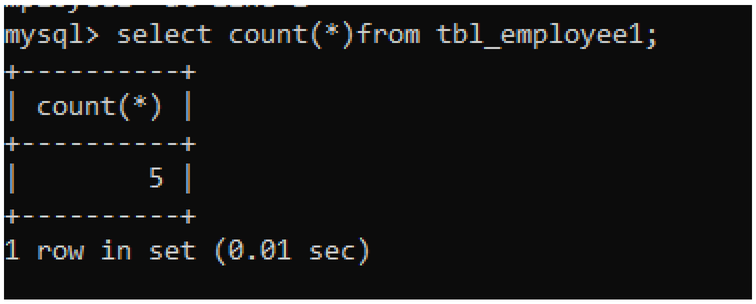


```
mysql> create table tbl_employee3 as select * from tbl_employee where 1=2;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from tbl_employee3;
Empty set (0.00 sec)

mysql> desc tbl_employee3;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int         | NO   |     | NULL    |       |
| name   | varchar(20) | YES  |     | NULL    |       |
| salary | int         | YES  |     | NULL    |       |
| dno    | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Count to find the total no of records in the table

select count(*)from tbl_employee1;

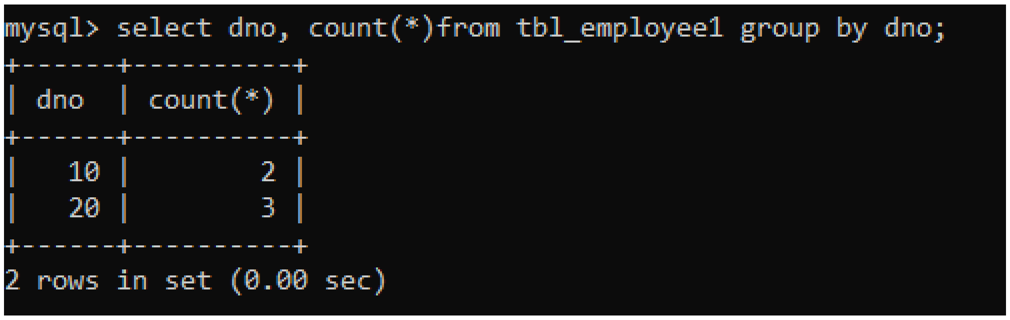

```
mysql> select * from tbl_employee1;
+-----+----------+--------+------+
| id  | name     | salary | dno  |
+-----+----------+--------+------+
| 101 | jeyavel  |   3000 |   10 |
| 102 | harrish  |   4000 |   10 |
| 103 | karthick |   5000 |   20 |
| 104 | bala     |   6000 |   20 |
| 105 | hari     |   7000 |   20 |
+-----+----------+--------+------+
5 rows in set (0.00 sec)
```


```
mysql> select count(*)from tbl_employee1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.01 sec)
```

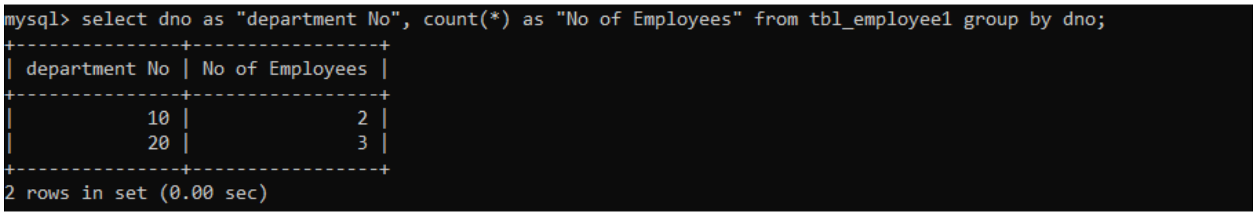
Group by -> next order by

Group by is used to group the data into applied condition or multiple values.
Both are used in multiple columns .
Order By using Multiple columns.
Ex: order by col1,col2;
Order by col1,col2,desc; G.F, S.R.F.

count() , sum(), Avg() , min() , max()


```
mysql> select dno, count(*)from tbl_employee1 group by dno;
+------+----------+
| dno  | count(*) |
+------+----------+
|   10 |        2 |
|   20 |        3 |
+------+----------+
2 rows in set (0.00 sec)
```

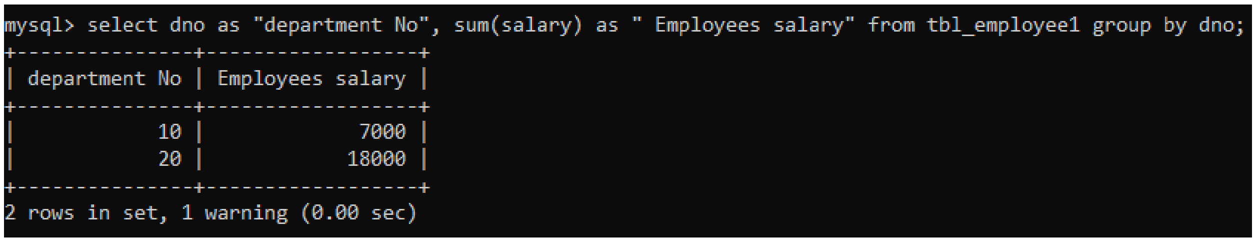
Use **alibi** in the **group by** for better understanding for the clients



```
mysql> select dno as "department No", count(*) as "No of Employees" from tbl_employee1 group by dno;
+---------------+-----------------+
| department No | No of Employees |
+---------------+-----------------+
|            10 |               2 |
|            20 |               3 |
+---------------+-----------------+
2 rows in set (0.00 sec)
```

sum()

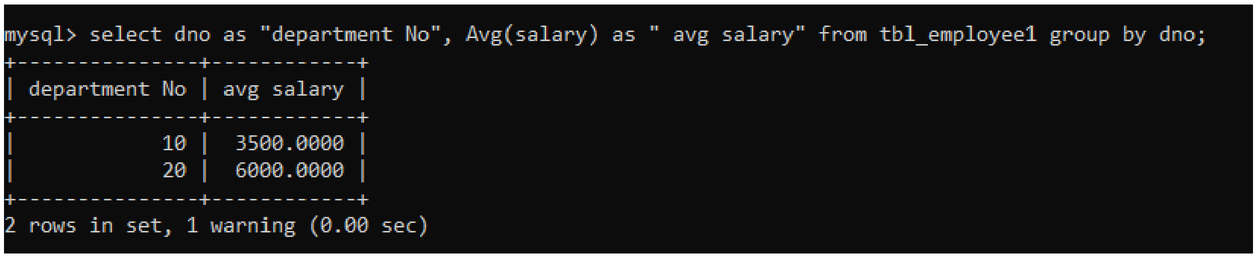
Ex



```
mysql> select dno as "department No", sum(salary) as " Employees salary" from tbl_employee1 group by dno;
+---------------+------------------+
| department No | Employees salary |
+---------------+------------------+
|            10 |             7000 |
|            20 |            18000 |
+---------------+------------------+
2 rows in set, 1 warning (0.00 sec)
```

Avg()

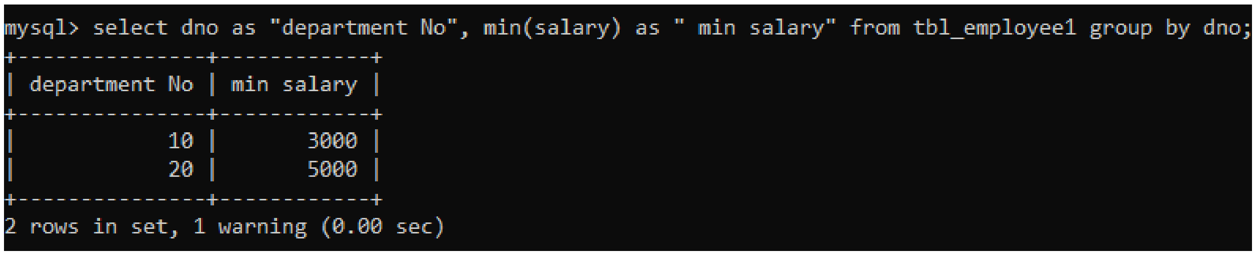
Ex



```
mysql> select dno as "department No", Avg(salary) as " avg salary" from tbl_employee1 group by dno;
+---------------+------------+
| department No | avg salary |
+---------------+------------+
|            10 |  3500.0000 |
|            20 |  6000.0000 |
+---------------+------------+
2 rows in set, 1 warning (0.00 sec)
```

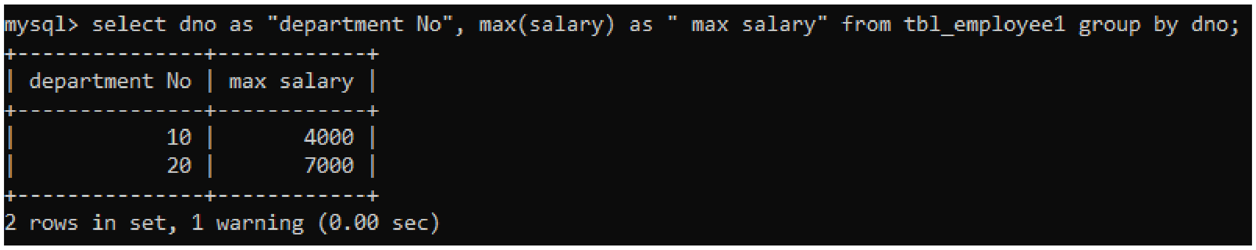
min()

Ex



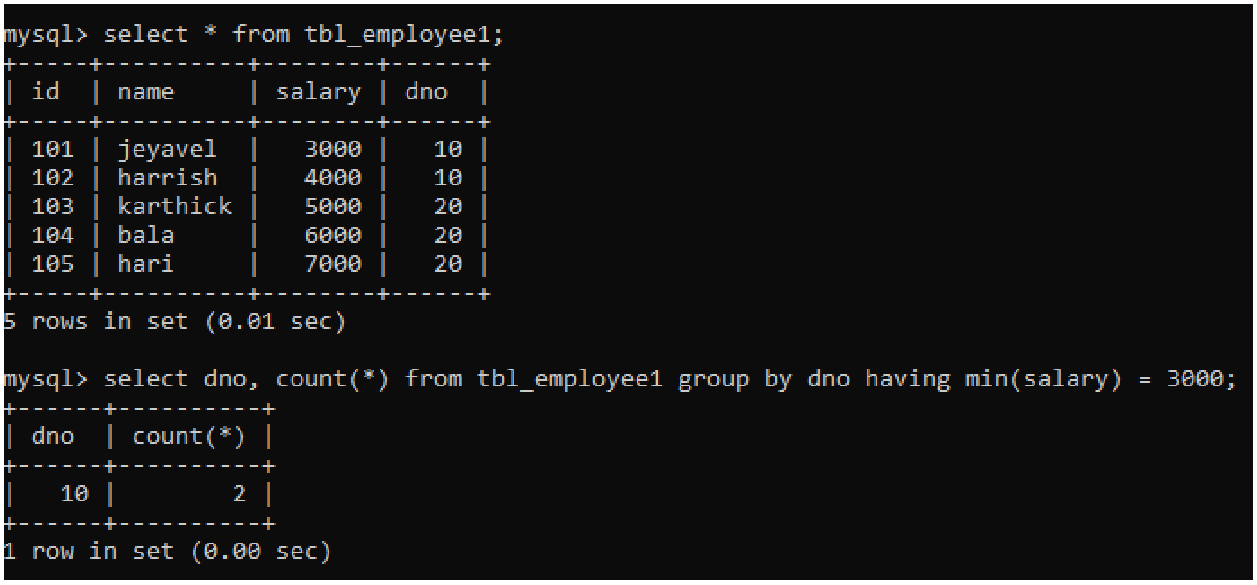
```
mysql> select dno as "department No", min(salary) as " min salary" from tbl_employee1 group by dno;
+---------------+------------+
| department No | min salary |
+---------------+------------+
|            10 |       3000 |
|            20 |       5000 |
+---------------+------------+
2 rows in set, 1 warning (0.00 sec)
```

max()
Ex



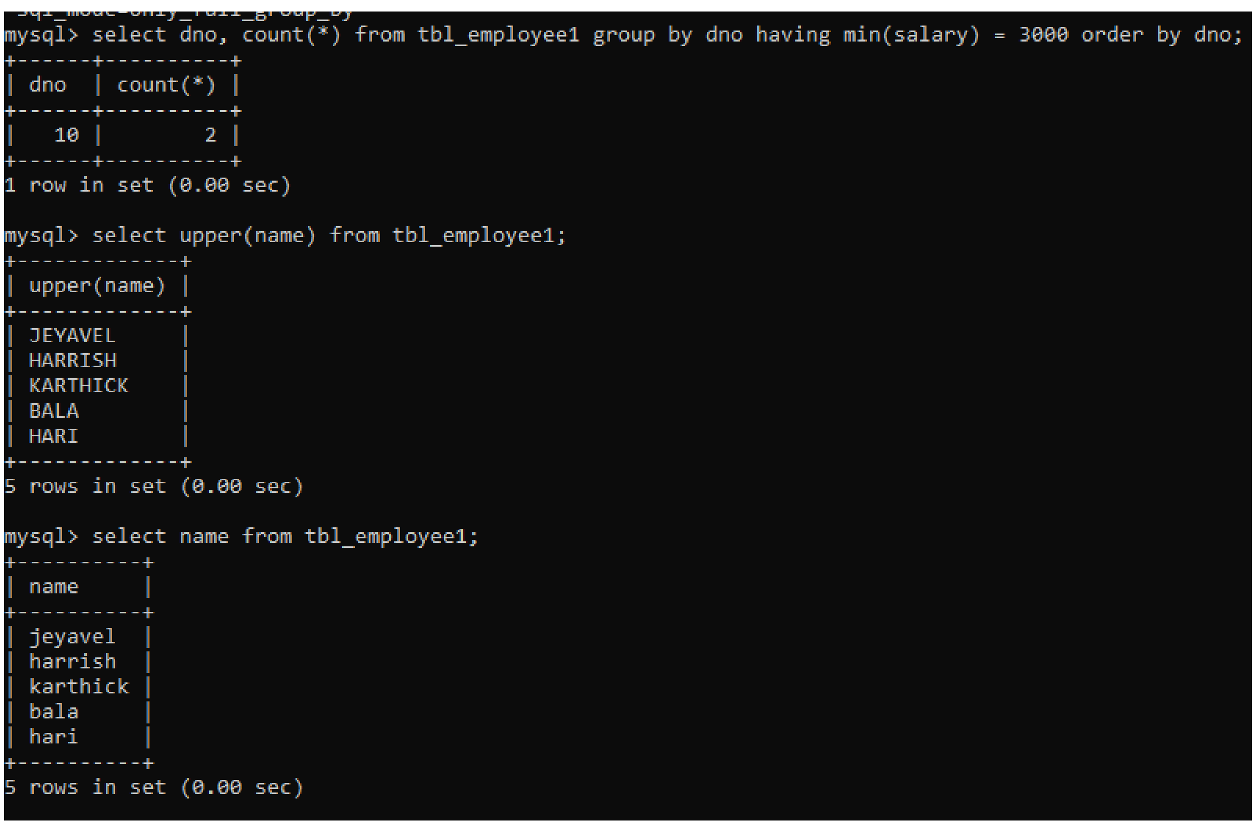
```
mysql> select dno as "department No", max(salary) as " max salary" from tbl_employee1 group by dno;
+---------------+------------+
| department No | max salary |
+---------------+------------+
|            10 |       4000 |
|            20 |       7000 |
+---------------+------------+
2 rows in set, 1 warning (0.00 sec)
```

Having keyword



```
mysql> select * from tbl_employee1;
+-----+----------+--------+------+
| id  | name     | salary | dno  |
+-----+----------+--------+------+
| 101 | jeyavel  |   3000 |   10 |
| 102 | harrish  |   4000 |   10 |
| 103 | karthick |   5000 |   20 |
| 104 | bala     |   6000 |   20 |
| 105 | hari     |   7000 |   20 |
+-----+----------+--------+------+
5 rows in set (0.01 sec)

mysql> select dno, count(*) from tbl_employee1 group by dno having min(salary) = 3000;
+------+----------+
| dno  | count(*) |
+------+----------+
|   10 |        2 |
+------+----------+
1 row in set (0.00 sec)
```

Cases -> ex upper case



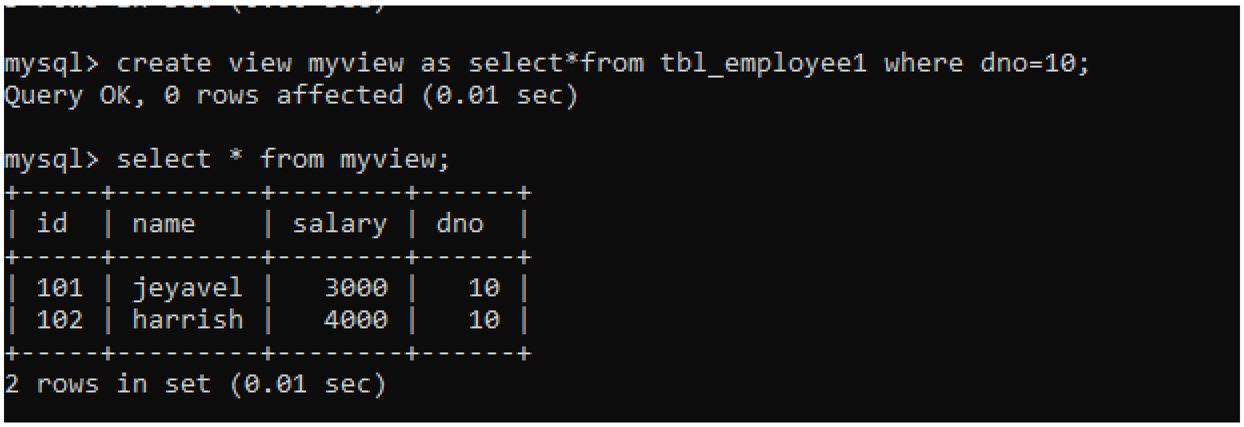
```
sql_mode=only_full_group_by
mysql> select dno, count(*) from tbl_employee1 group by dno having min(salary) = 3000 order by dno;
+------+----------+
| dno  | count(*) |
+------+----------+
|   10 |        2 |
+------+----------+
1 row in set (0.00 sec)

mysql> select upper(name) from tbl_employee1;
+-------------+
| upper(name) |
+-------------+
| JEYAVEL     |
| HARRISH     |
| KARTHICK    |
| BALA        |
| HARI        |
+-------------+
5 rows in set (0.00 sec)

mysql> select name from tbl_employee1;
+----------+
| name     |
+----------+
| jeyavel  |
| harrish  |
| karthick |
| bala     |
| hari     |
+----------+
5 rows in set (0.00 sec)
```

VIEW - vvl in business logic

- A view is the database object that has no values.
- If we change any data in the view table it will get reflected in the original table
- also same if the record change in the original table it will get reflected in the view table
- It is like hard disk and ram in computer
- **Simple View**
- **Complex View.**

  Simple views can only contain a **single base table.**
  Complex views can be **constructed on more than one base table.**
  In particular, complex views can contain: join conditions, a group by clause, order by clause.

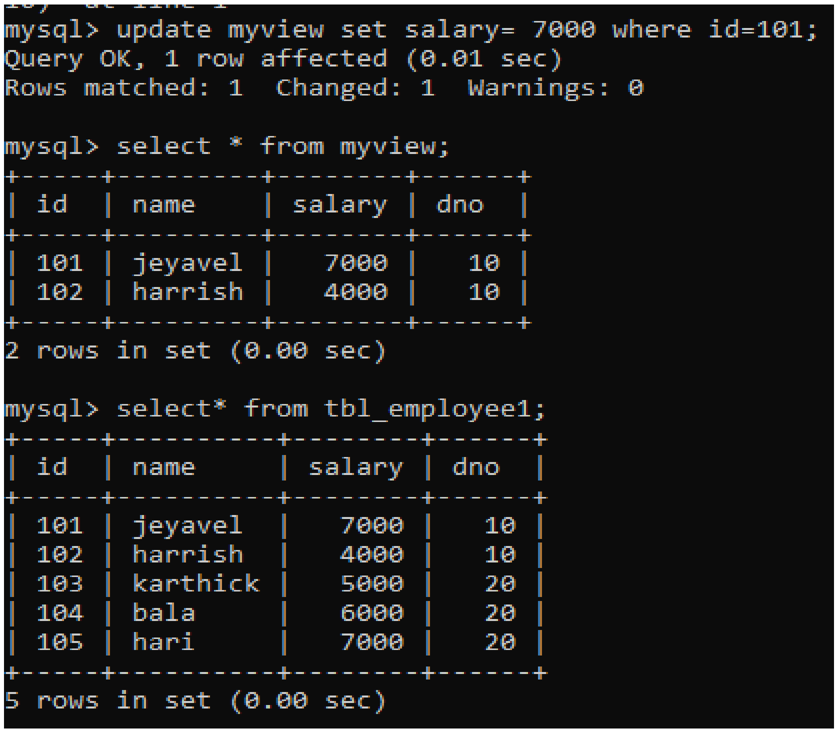
Output:



```
mysql> create view myview as select*from tbl_employee1 where dno=10;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from myview;
+-----+---------+--------+------+
| id  | name    | salary | dno  |
+-----+---------+--------+------+
| 101 | jeyavel |   3000 |   10 |
| 102 | harrish |   4000 |   10 |
+-----+---------+--------+------+
2 rows in set (0.01 sec)
```

Output

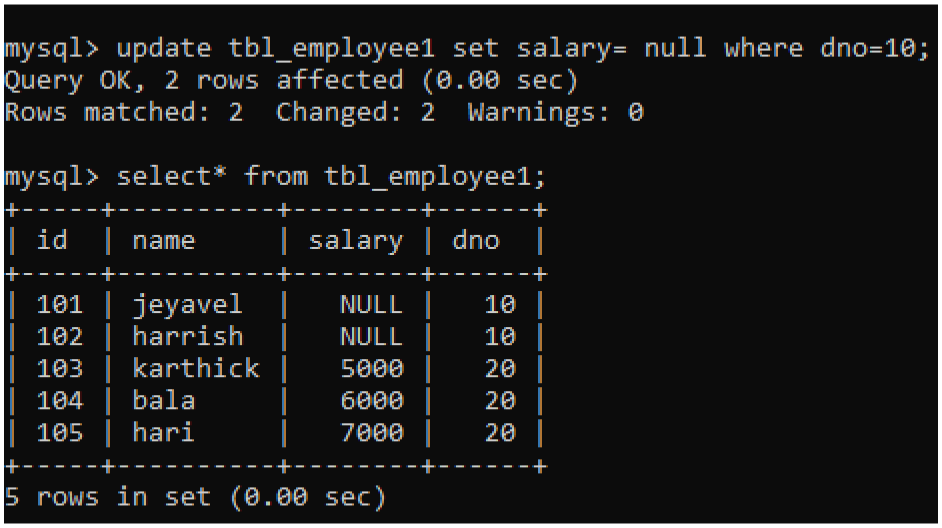
- If we change any data in the view it will get reflected in the original table



```
mysql> update myview set salary= 7000 where id=101;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from myview;
+-----+---------+--------+------+
| id  | name    | salary | dno  |
+-----+---------+--------+------+
| 101 | jeyavel |   7000 |   10 |
| 102 | harrish |   4000 |   10 |
+-----+---------+--------+------+
2 rows in set (0.00 sec)

mysql> select* from tbl_employee1;
+-----+----------+--------+------+
| id  | name     | salary | dno  |
+-----+----------+--------+------+
| 101 | jeyavel  |   7000 |   10 |
| 102 | harrish  |   4000 |   10 |
| 103 | karthick |   5000 |   20 |
| 104 | bala     |   6000 |   20 |
| 105 | hari     |   7000 |   20 |
+-----+----------+--------+------+
5 rows in set (0.00 sec)
```

Another Ex Output

- Also same if the record change in the original table it will get reflected in the view table



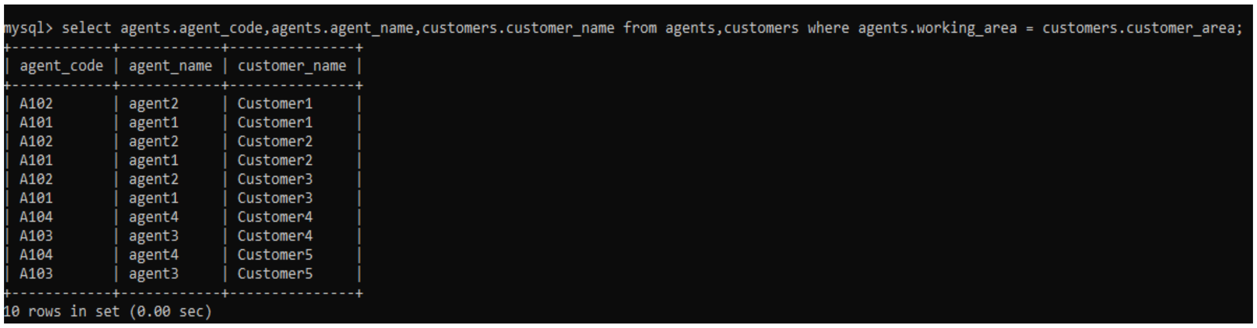
```
mysql> update tbl_employee1 set salary= null where dno=10;
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select* from tbl_employee1;
+-----+----------+--------+------+
| id  | name     | salary | dno  |
+-----+----------+--------+------+
| 101 | jeyavel  |   NULL |   10 |
| 102 | harrish  |   NULL |   10 |
| 103 | karthick |   5000 |   20 |
| 104 | bala     |   6000 |   20 |
| 105 | hari     |   7000 |   20 |
+-----+----------+--------+------+
5 rows in set (0.00 sec)
```

Join

There are 4 possible ways in joints
Using On to write a condition
Based on the condition it will display the display
Types of joints

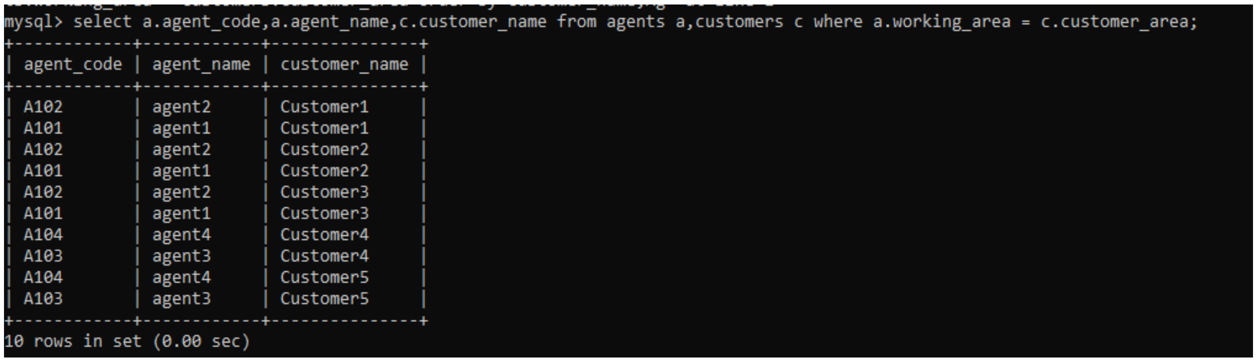
- Inner join - By default it is a inner join - it will display only the matching records from the table
- Equi join - (=)
- Non-Equi join - if we using other than equi will become the non equi join
- Outer join - will display as well as matching and non matching records
- Left outer join
- Right outer join
- Self join - join table within itself
- Cross join

Using table name to create join
Output:



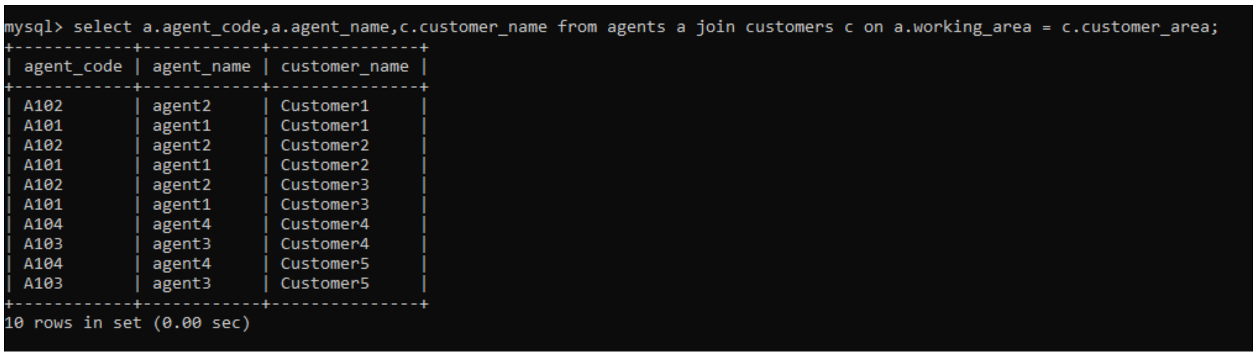
```
mysql> select agents.agent_code,agents.agent_name,customers.customer_name from agents,customers where agents.working_area = customers.customer_area;
+------------+------------+---------------+
| agent_code | agent_name | customer_name |
+------------+------------+---------------+
| A102       | agent2     | Customer1     |
| A101       | agent1     | Customer1     |
| A102       | agent2     | Customer2     |
| A101       | agent1     | Customer2     |
| A102       | agent2     | Customer3     |
| A101       | agent1     | Customer3     |
| A104       | agent4     | Customer4     |
| A103       | agent3     | Customer4     |
| A104       | agent4     | Customer5     |
| A103       | agent3     | Customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Using Alias:
Output:



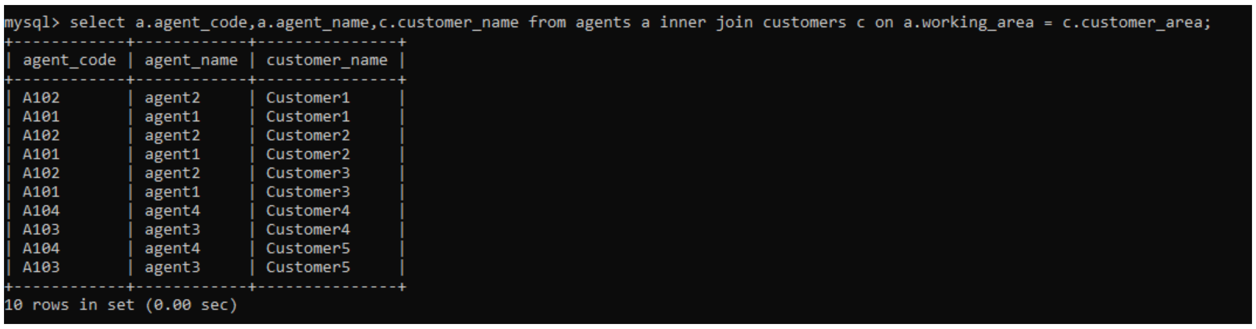
```
mysql> select a.agent_code,a.agent_name,c.customer_name from agents a,customers c where a.working_area = c.customer_area;
+------------+------------+---------------+
| agent_code | agent_name | customer_name |
+------------+------------+---------------+
| A102       | agent2     | Customer1     |
| A101       | agent1     | Customer1     |
| A102       | agent2     | Customer2     |
| A101       | agent1     | Customer2     |
| A102       | agent2     | Customer3     |
| A101       | agent1     | Customer3     |
| A104       | agent4     | Customer4     |
| A103       | agent3     | Customer4     |
| A104       | agent4     | Customer5     |
| A103       | agent3     | Customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Using join Keyword:
Output:



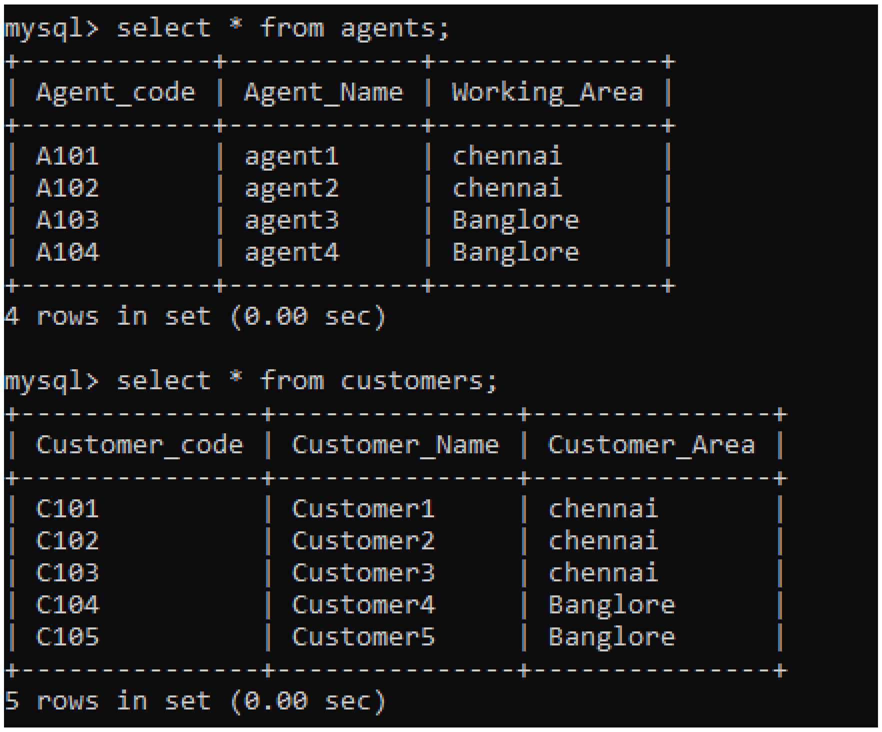
```
mysql> select a.agent_code,a.agent_name,c.customer_name from agents a join customers c on a.working_area = c.customer_area;
+------------+------------+---------------+
| agent_code | agent_name | customer_name |
+------------+------------+---------------+
| A102       | agent2     | Customer1     |
| A101       | agent1     | Customer1     |
| A102       | agent2     | Customer2     |
| A101       | agent1     | Customer2     |
| A102       | agent2     | Customer3     |
| A101       | agent1     | Customer3     |
| A104       | agent4     | Customer4     |
| A103       | agent3     | Customer4     |
| A104       | agent4     | Customer5     |
| A103       | agent3     | Customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Using inner join - by default it is an inner join
output



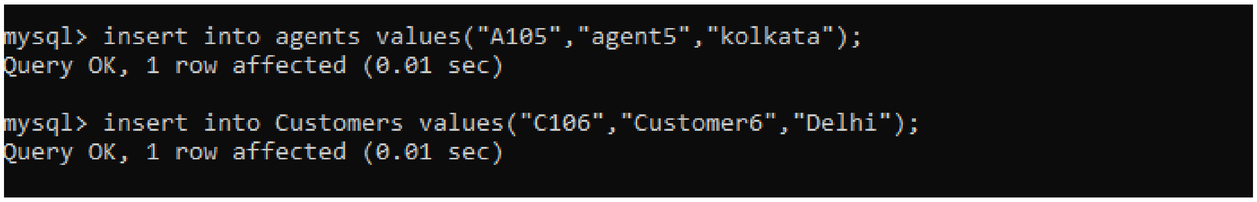
```
mysql> select a.agent_code,a.agent_name,c.customer_name from agents a inner join customers c on a.working_area = c.customer_area;
+------------+------------+---------------+
| agent_code | agent_name | customer_name |
+------------+------------+---------------+
| A102       | agent2     | Customer1     |
| A101       | agent1     | Customer1     |
| A102       | agent2     | Customer2     |
| A101       | agent1     | Customer2     |
| A102       | agent2     | Customer3     |
| A101       | agent1     | Customer3     |
| A104       | agent4     | Customer4     |
| A103       | agent3     | Customer4     |
| A104       | agent4     | Customer5     |
| A103       | agent3     | Customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Left outer join & right outer join
Output:



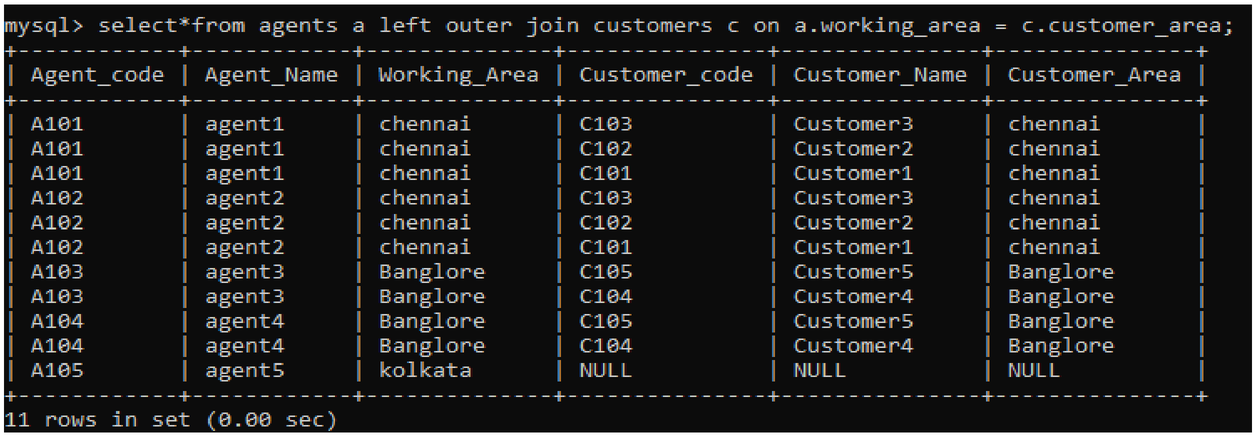
```
mysql> select * from agents;
+------------+------------+--------------+
| Agent_code | Agent_Name | Working_Area |
+------------+------------+--------------+
| A101       | agent1     | chennai      |
| A102       | agent2     | chennai      |
| A103       | agent3     | Banglore     |
| A104       | agent4     | Banglore     |
+------------+------------+--------------+
4 rows in set (0.00 sec)

mysql> select * from customers;
+---------------+---------------+---------------+
| Customer_code | Customer_Name | Customer_Area |
+---------------+---------------+---------------+
| C101          | Customer1     | chennai       |
| C102          | Customer2     | chennai       |
| C103          | Customer3     | chennai       |
| C104          | Customer4     | Banglore      |
| C105          | Customer5     | Banglore      |
+---------------+---------------+---------------+
5 rows in set (0.00 sec)
```

We have to add two new data to the tables to see the
output



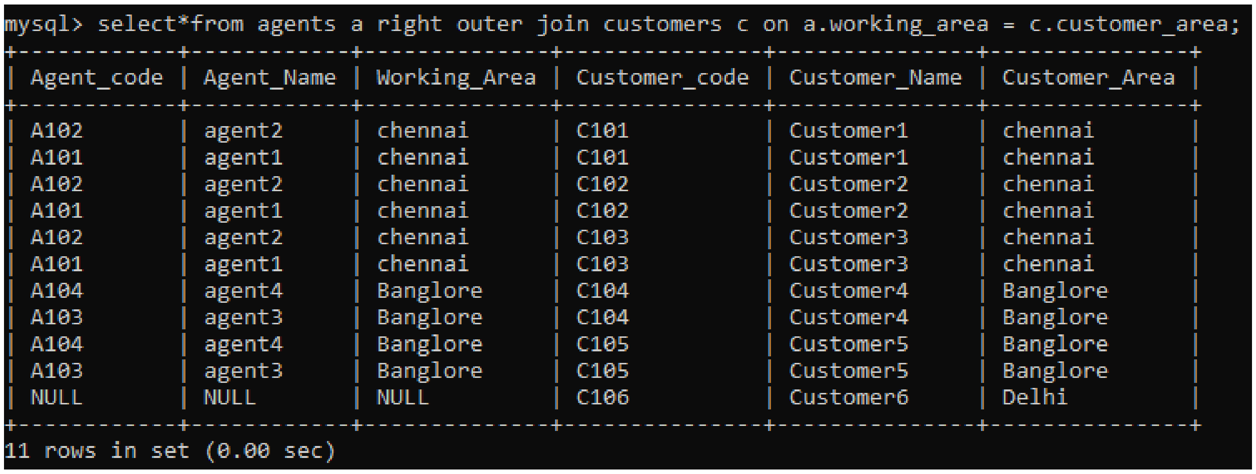
```
mysql> insert into agents values("A105","agent5","kolkata");
Query OK, 1 row affected (0.01 sec)

mysql> insert into Customers values("C106","Customer6","Delhi");
Query OK, 1 row affected (0.01 sec)
```

To verify the left outer join
 output



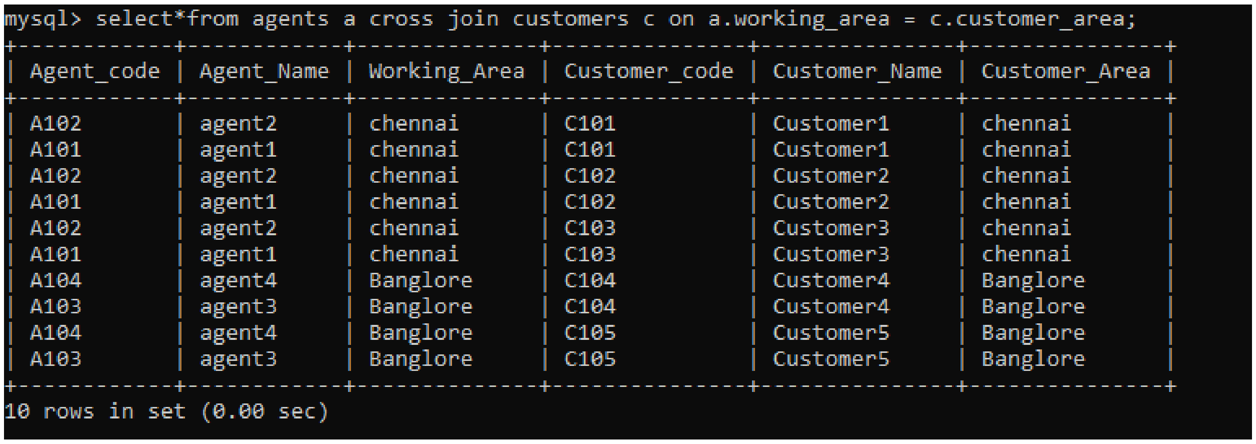
```
mysql> select*from agents a left outer join customers c on a.working_area = c.customer_area;
+------------+------------+--------------+---------------+---------------+---------------+
| Agent_code | Agent_Name | Working_Area | Customer_code | Customer_Name | Customer_Area |
+------------+------------+--------------+---------------+---------------+---------------+
| A101       | agent1     | chennai      | C103          | Customer3     | chennai       |
| A101       | agent1     | chennai      | C102          | Customer2     | chennai       |
| A101       | agent1     | chennai      | C101          | Customer1     | chennai       |
| A102       | agent2     | chennai      | C103          | Customer3     | chennai       |
| A102       | agent2     | chennai      | C102          | Customer2     | chennai       |
| A102       | agent2     | chennai      | C101          | Customer1     | chennai       |
| A103       | agent3     | Banglore     | C105          | Customer5     | Banglore      |
| A103       | agent3     | Banglore     | C104          | Customer4     | Banglore      |
| A104       | agent4     | Banglore     | C105          | Customer5     | Banglore      |
| A104       | agent4     | Banglore     | C104          | Customer4     | Banglore      |
| A105       | agent5     | kolkata      | NULL          | NULL          | NULL          |
+------------+------------+--------------+---------------+---------------+---------------+
11 rows in set (0.00 sec)
```

Right Outer join
Output:



```
mysql> select*from agents a right outer join customers c on a.working_area = c.customer_area;
+------------+------------+--------------+---------------+---------------+---------------+
| Agent_code | Agent_Name | Working_Area | Customer_code | Customer_Name | Customer_Area |
+------------+------------+--------------+---------------+---------------+---------------+
| A102       | agent2     | chennai      | C101          | Customer1     | chennai       |
| A101       | agent1     | chennai      | C101          | Customer1     | chennai       |
| A102       | agent2     | chennai      | C102          | Customer2     | chennai       |
| A101       | agent1     | chennai      | C102          | Customer2     | chennai       |
| A102       | agent2     | chennai      | C103          | Customer3     | chennai       |
| A101       | agent1     | chennai      | C103          | Customer3     | chennai       |
| A104       | agent4     | Banglore     | C104          | Customer4     | Banglore      |
| A103       | agent3     | Banglore     | C104          | Customer4     | Banglore      |
| A104       | agent4     | Banglore     | C105          | Customer5     | Banglore      |
| A103       | agent3     | Banglore     | C105          | Customer5     | Banglore      |
| NULL       | NULL       | NULL         | C106          | Customer6     | Delhi         |
+------------+------------+--------------+---------------+---------------+---------------+
11 rows in set (0.00 sec)
```

Cross join
Output:



```
mysql> select*from agents a cross join customers c on a.working_area = c.customer_area;
+------------+------------+--------------+---------------+---------------+---------------+
| Agent_code | Agent_Name | Working_Area | Customer_code | Customer_Name | Customer_Area |
+------------+------------+--------------+---------------+---------------+---------------+
| A102       | agent2     | chennai      | C101          | Customer1     | chennai       |
| A101       | agent1     | chennai      | C101          | Customer1     | chennai       |
| A102       | agent2     | chennai      | C102          | Customer2     | chennai       |
| A101       | agent1     | chennai      | C102          | Customer2     | chennai       |
| A102       | agent2     | chennai      | C103          | Customer3     | chennai       |
| A101       | agent1     | chennai      | C103          | Customer3     | chennai       |
| A104       | agent4     | Banglore     | C104          | Customer4     | Banglore      |
| A103       | agent3     | Banglore     | C104          | Customer4     | Banglore      |
| A104       | agent4     | Banglore     | C105          | Customer5     | Banglore      |
| A103       | agent3     | Banglore     | C105          | Customer5     | Banglore      |
+------------+------------+--------------+---------------+---------------+---------------+
10 rows in set (0.00 sec)
```

—--

Where keyword

Used in insert , select , update , delete

Syntax:
Where is mainly used for filtering the data in the table and to display
Select column1,column2, from table_name **where** condition;

Using **=** in where
Using **and keyword** in where
Using **or keyword** in where
Using **in keyword** in where
Using **not in keyword** in where
Using **between keyword** in where

—---

Limit - Keyword
Used to limit the table input
Ex:
If the table contains 12 rows we can say like
Limit 5;
It only display 5 rows;

—---

LIKE , NOT like - to filter the data by patterns like
starting letter with - r
And also numbers

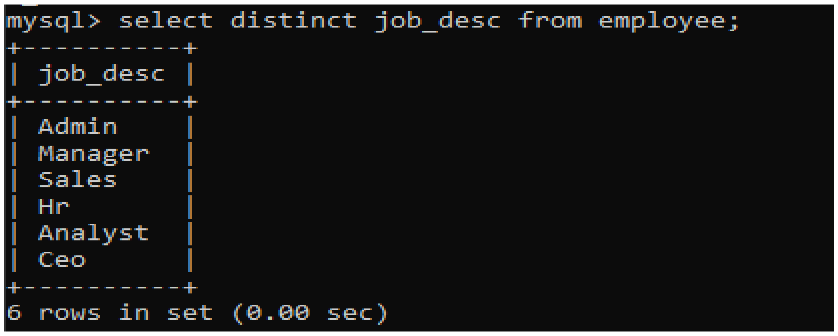
Wildcards - %(Zero or more characters) _(this underscore defines one character)

—---

Distinct - it means unique and not repeated

It does not repeat the value in the selected table

Ex



```
mysql> select distinct job_desc from employee;
+----------+
| job_desc |
+----------+
| Admin    |
| Manager  |
| Sales    |
| Hr       |
| Analyst  |
| Ceo      |
+----------+
6 rows in set (0.00 sec)
```

Order By , Order By DESC

To change the order of the table

Order by will be in the last of the table

After filtering the table order by will be executed

Custom order By

Syntax:

Function - performs a specific task

Mat related functions

sum() Avg() Min() Max()

String related functions

UCASE() - uppercase

CHAR_LENGTH()-length of the character

CONCAT()- CONCATENATION

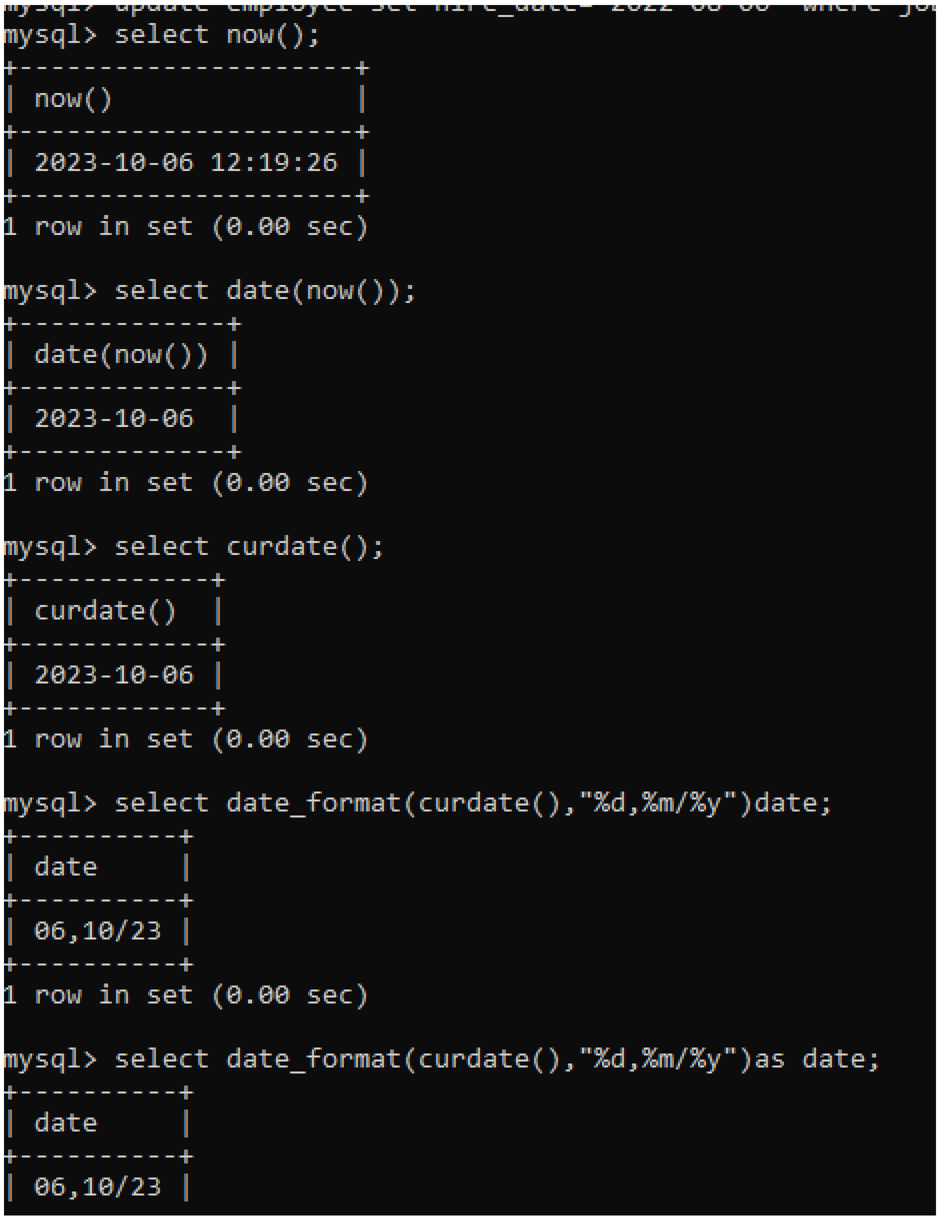
FORMAT()

LEFT()

refer

https://www.techonthenet.com/mysql/index.php

DATE



```
mysql> update employee set hire_date="2022-08-08" where jo
mysql> select now();
+---------------------+
| now()               |
+---------------------+
| 2023-10-06 12:19:26 |
+---------------------+
1 row in set (0.00 sec)

mysql> select date(now());
+-------------+
| date(now()) |
+-------------+
| 2023-10-06  |
+-------------+
1 row in set (0.00 sec)

mysql> select curdate();
+------------+
| curdate()  |
+------------+
| 2023-10-06 |
+------------+
1 row in set (0.00 sec)

mysql> select date_format(curdate(),"%d,%m/%y")date;
+----------+
| date     |
+----------+
| 06,10/23 |
+----------+
1 row in set (0.00 sec)

mysql> select date_format(curdate(),"%d,%m/%y")as date;
+----------+
| date     |
+----------+
| 06,10/23 |
```

Timestamp

Group by

It is used to group a particular column weather it is suitable for group by

The main purpose of the group by is to use the Aggregate functions
Like sum, avg , min, max etc

Having - to filter the values in group by

Order by

It is used to sorting a column after Group by is done .