SQL

20 August 2024 13:46

- **SEARCHING DATABASES :**

SHOW DATABASES;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pizzahut           |
| practice           |
| sqlcodes           |
| sys                |
+--------------------+
```

- **Creating the database = *CREATE DATABASE students;***

```
mysql> CREATE DATABASE students;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pizzahut           |
| practice           |
| sqlcodes           |
| students           |
| sys                |
+--------------------+
```

- **Using the database = USE students;**

```
mysql>  USE students;
Database changed
mysql>
```

- **SHOW TABLES**

```
mysql> show tables;
+--------------------+
| Tables_in_practice |
+--------------------+
| contacts           |
| cust               |
| customers          |
| users              |
+--------------------+
```

- **Creating the table = *CREATE TABLE student_info (id INT, name VARCHAR(20) , contact INT);***

```
mysql> CREATE TABLE student_info (id INT, name VARCHAR(20) , contact INT);
Query OK, 0 rows affected (0.02 sec)
```

- **Inserting the values into the table = *INSERT INTO student_info (id, name, contact) values (01, 'ravi', 963), (02, 'suri', 452)***

```
mysql> INSERT INTO student_info (id, name, contact) values (01, 'ravi', 963)
, (02, 'suri', 452)
    ->
    -> ;
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

- **Describing the table = this allows us to describe the values in the table**

- ***DESC students_info;***

```
mysql> desc student_info;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| id      | int         | YES  |     | NULL    |       |
| name    | varchar(20) | YES  |     | NULL    |       |
| contact | int         | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- **Reading the table = *SELECT * FROM student_info;***

```
mysql> SELECT * FROM student_info;
+------+------+---------+
| id   | name | contact |
+------+------+---------+
|    1 | ravi |     963 |
|    2 | suri |     452 |
+------+------+---------+
2 rows in set (0.00 sec)
```

- **Updating the table values = we can change the existing table values**

- ***UPDATE student_info SET contact = "420" WHERE id = 2;***

```
mysql> UPDATE student_info SET contact = "420" WHERE id = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student_info;
+------+------+---------+
| id   | name | contact |
+------+------+---------+
|    1 | ravi |     963 |
|    2 | suri |     420 |
+------+------+---------+
2 rows in set (0.00 sec)
```

- **Drop table = this deletes or drops the table from the database**
- ***DROP student_info;***

- **Null = generally SQL takes the values as null if not given. For example let's add 3rd student in our table with no contact data**
- ***INSERT INTO student_info(id, name) VALUES (3, "ramesh");***

```
mysql> INSERT INTO student_info(id, name) VALUES (3, "ramesh");
Query OK, 1 row affected (0.00 sec)

mysql> select * from student_info;
+------+--------+---------+
| id   | name   | contact |
+------+--------+---------+
|    1 | ravi   |     963 |
|    2 | suri   |     420 |
|    3 | ramesh |    NULL |
+------+--------+---------+
3 rows in set (0.00 sec)
```

- **NOT NULL = TO GO OVER THAT ISSUE WE CAN USE NOT NULL IN OUR SYNTAX IT GOES LIKE THIS. This not null ensures that the table has no null values**

CREATE TABLE customers (id INT NOT NULL, name VARCHAR(100) NOT NULL) ;

```
mysql> CREATE TABLE customers ( id INT NOT NULL, name VARCHAR(100) NOT NULL)
 ;
Query OK, 0 rows affected (0.01 sec)

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

mysql> insert into customers(id, name) values (101,"parth"), (102, "kumar");
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0


mysql> select * from customers;
+-----+-------+
| id  | name  |
+-----+-------+
| 101 | parth |
| 102 | kumar |
+-----+-------+
```

- **The following error pops up when we use NOT NULL and don't add values**

```
mysql> insert into customers(id) values (103);
ERROR 1364 (HY000): Field 'name' doesn't have a default value
mysql>
```

- **DEFAULT VALUES = This ensures us to have a default value instead of null so that we don't get the above error when we are inserting the data.**
- **For example most of the bank accounts are savings so we can set that as a default value so that we don't have to worry about entering it all the time.**

- ***CREATE TABLE CUSTOMERS2(ID INT NOT NULL, NAME VARCHAR(100) NOT NULL, ACC_TYPE VARCHAR(100) NOT NULL DEFAULT "Savings");***

```
mysql> desc customers2
    -> ;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| ID       | int          | NO   |     | NULL    |       |
| NAME     | varchar(100) | NO   |     | NULL    |       |
| ACC_TYPE | varchar(100) | NO   |     | Savings |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- **As you can see that acc_type has a default value (savings)**
- **Now we if we forget to add the values as well like below we will not get any error.**
  insert into customers2(id, name)
  values(101,'suresh'), (102, 'ramesh'), (103, 'Yougesh'), (104,"naresh");

```
mysql> select * from customers2;
+-----+---------+----------+
| ID  | NAME    | ACC_TYPE |
+-----+---------+----------+
| 101 | suresh  | Savings  |
| 102 | ramesh  | Savings  |
| 103 | Yougesh | Savings  |
| 104 | naresh  | Savings  |
+-----+---------+----------+
4 rows in set (0.00 sec)
```

- **PRIMARY KEY: A TABLE NEEDS TO HAVE A PRIMARY KEY AND IT CANNOT BE EMPTY, OR NULL. EACH VALUE IN THE PRIMARY KEY SHOULD BE UNIQUE WHICH HELPS US TO**

IDENTIFY OR EXTRACT THE REQUIRED DATA.

- **TO ADD A PRIMARY KEY WE CAN USE "PRIMARY KEY" BESIDE THE COLUMN NAME LIKE THIS**

CREATE TABLE customers3(acc_no INT PRIMARY KEY, name VARCHAR(100) NOT NULL, acc_type VARCHAR(100) NOT NULL DEFAULT "Savings");

```
mysql> desc customers3;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| acc_no   | int          | NO   | PRI | NULL    |       |
| name     | varchar(100) | NO   |     | NULL    |       |
| acc_type | varchar(100) | NO   |     | Savings |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> insert into customers3(acc_no, name)
    -> VALUES (96354, 'lisa'), (85412, 'jenny');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from customers3;
+--------+-------+----------+
| acc_no | name  | acc_type |
+--------+-------+----------+
|  85412 | jenny | Savings  |
|  96354 | lisa  | Savings  |
+--------+-------+----------+
2 rows in set (0.00 sec)
```

- **Now it will not allow us to add the same primary key. As mentioned above if I use jenny's primary key again it will throw an error.**

```
mysql> insert into customers3(acc_no, name)
    -> values ( 85412, "mona");
ERROR 1062 (23000): Duplicate entry '85412' for key 'customers3.PRIMARY'
```

- **AUTO_INCREMENT : By default it starts with 0 and increases by 1. This makes it easy to add values into the table without worry about assigning the values each time and we can also add values if we want.**

create table cust(acc_no INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100) NOT NULL,
acc_type VARCHAR(100) NOT NULL DEFAULT "current");

```
mysql> create table cust(acc_no INT PRIMARY KEY AUTO_INCREMENT,
    -> name VARCHAR(100) NOT NULL,
    -> acc_type VARCHAR(100) NOT NULL DEFAULT "current");
Query OK, 0 rows affected (0.02 sec)

mysql> desc cust;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| acc_no   | int          | NO   | PRI | NULL    | auto_increment |
| name     | varchar(100) | NO   |     | NULL    |                |
| acc_type | varchar(100) | NO   |     | current |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Now it will automatically assign the values to the table like below which makes the data input much easier.

```
mysql> insert into cust(name)
    -> values ("ram"), ("laxman"), ("arjun");
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from cust;
+--------+--------+----------+
| acc_no | name   | acc_type |
+--------+--------+----------+
|      1 | ram    | current  |
|      2 | laxman | current  |
|      3 | arjun  | current  |
+--------+--------+----------+
3 rows in set (0.00 sec)
```

- **ALIAS : WE CAN USE " as " TO USE THE TABLE WITH THE DESIRED NAME**
- **FOR EXAMPLE : CUSTOMERS_3 WE CAN USE IT SIMPLY AS "CUST" OR WE CAN ALSO USE IT FOR TABLE VALUES.**

SELECT acc_no as "account no", name as "customer Name" from cust;

```
mysql> SELECT acc_no as "account no", name as "customer Name" from cust;
+------------+---------------+
| account no | customer Name |
+------------+---------------+
|          1 | ram           |
|          2 | laxman        |
|          3 | arjun         |
+------------+---------------+
```

- **Selecting the specific values from the table**

```
mysql> select * from cust;
+--------+--------+----------+
| acc_no | name   | acc_type |
+--------+--------+----------+
|      1 | ram    | current  |
|      2 | laxman | current  |
|      3 | arjun  | current  |
|      4 | naveen | loan     |
|      5 | giri   | savings  |
|      6 | sai    | loan     |
+--------+--------+----------+
```

- **I need the names of the people with loan accounts.**

```
mysql> select name from cust where acc_type = "loan";
+--------+
| name   |
+--------+
| naveen |
| sai    |
+--------+
```

We can use where clause to extract the required data based on the condition we provide

```
mysql> select * from cust where acc_no = 4;
+--------+--------+----------+
| acc_no | name   | acc_type |
+--------+--------+----------+
|      4 | naveen | loan     |
+--------+--------+----------+
```

- **We can even update the values with "where" clause.**

```
mysql> update cust set acc_type = "loan" where acc_no = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from cust;
+--------+--------+----------+
| acc_no | name   | acc_type |
+--------+--------+----------+
|      1 | ram    | loan     |
|      2 | laxman | current  |
|      3 | arjun  | current  |
|      4 | naveen | loan     |
|      5 | giri   | savings  |
|      6 | sai    | loan     |
+--------+--------+----------+
```

- **We can also delete the particular row with where.**

```
mysql> DELETE FROM cust where acc_no = 4;
Query OK, 1 row affected (0.01 sec)

mysql> select * from cust;
+--------+--------+----------+
| acc_no | name   | acc_type |
+--------+--------+----------+
|      1 | ram    | loan     |
|      2 | laxman | current  |
|      3 | arjun  | current  |
|      5 | giri   | savings  |
|      6 | sai    | loan     |
+--------+--------+----------+
```

==========

STRING FUNCTIONS

- **CONCAT : TO ADD THE DATA**

```
mysql> SELECT CONCAT("hELLO", "WORLD");
+--------------------------+
| CONCAT("hELLO", "WORLD") |
+--------------------------+
| hELLOWORLD               |
+--------------------------+
```

- **Now lets add the values in the table using this**

```
+----+-----------+----------+------------+-------------------------------+
| ID | FirstName | LastName | Contact    | Email                         |
+----+-----------+----------+------------+-------------------------------+
|  1 | John      | Doe      | 1234567890 | johndoe@example.com           |
|  2 | Jane      | Smith    | 2345678901 | janesmith@example.com         |
|  3 | Michael   | Johnson  | 3456789012 | michaeljohnson@example.com    |
|  4 | Emily     | Davis    | 4567890123 | emilydavis@example.com        |
|  5 | David     | Martinez | 5678901234 | davidmartinez@example.com     |
|  6 | Sarah     | Lee      | 6789012345 | sarahlee@example.com          |
|  7 | James     | Brown    | 7890123456 | jamesbrown@example.com        |
|  8 | Olivia    | Wilson   | 8901234567 | oliviawilson@example.com      |
|  9 | Daniel    | Garcia   | 9012345678 | danielgarcia@example.com      |
| 10 | Sophia    | Miller   | 0123456789 | sophiamiller@example.com      |
| 11 | William   | Moore    | 1234567891 | williammoore@example.com      |
| 12 | Isabella  | Taylor   | 2345678902 | isabellataylor@example.com    |
| 13 | Alexander | Anderson | 3456789013 | alexanderanderson@example.com |
| 14 | Mia       | Thomas   | 4567890124 | miathomas@example.com         |
| 15 | Liam      | Jackson  | 5678901235 | liamjackson@example.com       |
```

- **After combining the values**

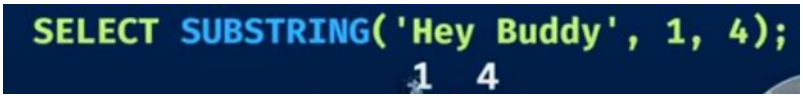
```
mysql> select id, concat(firstname, "", lastname) as FullName from users;
+----+-------------------+
| id | FullName          |
+----+-------------------+
|  1 | JohnDoe           |
|  2 | JaneSmith         |
|  3 | MichaelJohnson    |
|  4 | EmilyDavis        |
|  5 | DavidMartinez     |
|  6 | SarahLee          |
|  7 | JamesBrown        |
|  8 | OliviaWilson      |
|  9 | DanielGarcia      |
| 10 | SophiaMiller      |
| 11 | WilliamMoore      |
| 12 | IsabellaTaylor    |
| 13 | AlexanderAnderson |
| 14 | MiaThomas         |
| 15 | LiamJackson       |
```

- **CONCAT_WS : THIS ADDS A "_" WHILE ADDING THE COLUMS TOGETHER SO THAT IT WILL BE EASIER TO IDENTIFY.**

```
mysql> SELECT ID, CONCAT_WS('-', FIRSTNAME, LASTNAME) FROM USERS;
+----+-------------------------------------+
| ID | CONCAT_WS('-', FIRSTNAME, LASTNAME) |
+----+-------------------------------------+
|  1 | John-Doe                            |
|  2 | Jane-Smith                          |
|  3 | Michael-Johnson                     |
|  4 | Emily-Davis                         |
|  5 | David-Martinez                      |
|  6 | Sarah-Lee                           |
|  7 | James-Brown                         |
|  8 | Olivia-Wilson                       |
|  9 | Daniel-Garcia                       |
| 10 | Sophia-Miller                       |
+----+-------------------------------------+
```

- **We can add as many columns as we want like below and we can use any kind of separator like "-" or ':' or anything.**

```
mysql> SELECT ID, CONCAT_WS('-', FIRSTNAME, LASTNAME, CONTACT) AS DETAILS FR
OM USERS;
+----+----------------------------+
| ID | DETAILS                    |
+----+----------------------------+
|  1 | John-Doe-1234567890        |
|  2 | Jane-Smith-2345678901      |
|  3 | Michael-Johnson-3456789012 |
|  4 | Emily-Davis-4567890123     |
|  5 | David-Martinez-5678901234  |
|  6 | Sarah-Lee-6789012345       |
|  7 | James-Brown-7890123456     |
|  8 | Olivia-Wilson-8901234567   |
|  9 | Daniel-Garcia-9012345678   |
| 10 | Sophia-Miller-0123456789   |
+----+----------------------------+
```

- **SUB STRING : THIS WILL CUT THE WORD OR DATA ACCORDING TO ITS POSITION.**
- 

-

```
mysql> SELECT SUBSTRING("HELLO bUDDY", 1,4);
+------------------------------+
| SUBSTRING("HELLO bUDDY", 1,4) |
+------------------------------+
| HELL                         |
+------------------------------+
```

```
mysql> SELECT SUBSTRING("HEY bUDDY", 1,4);
+----------------------------+
| SUBSTRING("HEY bUDDY", 1,4) |
+----------------------------+
| HEY                        |
+----------------------------+
1 row in set (0.00 sec)
```

- **We can also perform actions from the end**

```
mysql> select substring("hey buddy, how are you. I am jose", -4);
+-------------------------------------------------+
| substring("hey buddy, how are you. I am jose", -4) |
+-------------------------------------------------+
| jose                                            |
+-------------------------------------------------+
```

- **SUB string in the table**

```
mysql> select firstname, substring(contact, 1,4) as "first_four_contact" fro
m users;
+-----------+--------------------+
| firstname | first_four_contact |
+-----------+--------------------+
| John      | 1234               |
| Jane      | 2345               |
| Michael   | 3456               |
| Emily     | 4567               |
| David     | 5678               |
| Sarah     | 6789               |
| James     | 7890               |
| Olivia    | 8901               |
| Daniel    | 9012               |
| Sophia    | 0123               |
+-----------+--------------------+
```

- **Replace : This is used to replace the values in the table. The syntax you need to follow**


```
mysql> select replace("hello Buddy", "hello", "hey");
+---------------------------------------+
| replace("hello Buddy", "hello", "hey") |
+---------------------------------------+
| hey Buddy                             |
+---------------------------------------+
1 row in set (0.00 sec)
```

- **Ex in table :**

```
mysql> select replace(ID, 1, 10) as new_id, firstname from users;
+--------+-----------+
| new_id | firstname |
+--------+-----------+
| 10     | John      |
| 2      | Jane      |
| 3      | Michael   |
| 4      | Emily     |
| 5      | David     |
| 6      | Sarah     |
| 7      | James     |
| 8      | Olivia    |
| 9      | Daniel    |
| 100    | Sophia    |
+--------+-----------+
```

- **You can see that 1 has been replaced with 10 and if you see closely since we used primary key and auto_increment the last 10th value changed to 100 to keep the uniqueness.**

- **REVERSE : This just reverses the values that's it.**

```
mysql> select reverse("Hello");
+------------------+
| reverse("Hello") |
+------------------+
| olleH            |
+------------------+
1 row in set (0.00 sec)
```

- **Upper/Lower : They convert the letters to upper and lowercases.**
-
```
mysql> select ucase("hello");
+----------------+
| ucase("hello") |
+----------------+
| HELLO          |
+----------------+
```

-
```
mysql> select lcase("HELLO");
+----------------+
| lcase("HELLO") |
+----------------+
| hello          |
+----------------+
```

-
```
mysql> select id, Upper(firstname), Lower(lastname) from users;
+----+------------------+-----------------+
| id | Upper(firstname) | Lower(lastname) |
+----+------------------+-----------------+
|  1 | JOHN             | doe             |
|  2 | JANE             | smith           |
|  3 | MICHAEL          | johnson         |
|  4 | EMILY            | davis           |
|  5 | DAVID            | martinez        |
|  6 | SARAH            | lee             |
|  7 | JAMES            | brown           |
|  8 | OLIVIA           | wilson          |
|  9 | DANIEL           | garcia          |
| 10 | SOPHIA           | miller          |
+----+------------------+-----------------+
```

- **CHAR LENGTH: Used to find the length of the string**

```
mysql> select firstname, char_length(firstname) as length from users;
+-----------+--------+
| firstname | length |
+-----------+--------+
| John      |      4 |
| Jane      |      4 |
| Michael   |      7 |
| Emily     |      5 |
| David     |      5 |
| Sarah     |      5 |
| James     |      5 |
| Olivia    |      6 |
| Daniel    |      6 |
| Sophia    |      6 |
+-----------+--------+
```

- **We can use this to filter or find the data according to their char length**

```
mysql> select * from users where char_length(firstname) > 6;
+----+-----------+----------+------------+---------------------------+
| ID | FirstName | LastName | Contact    | Email                     |
+----+-----------+----------+------------+---------------------------+
|  3 | Michael   | Johnson  | 3456789012 | michaeljohnson@example.com |
+----+-----------+----------+------------+---------------------------+
```

- **INSERT : used to insert the values In between.**

```
mysql> select insert("hello world", 7, 0, "Remo ");
+--------------------------------------+
| insert("hello world", 7, 0, "Remo ") |
+--------------------------------------+
| hello Remo world                     |
+--------------------------------------+
```

- **LEFT/RIGHT : Used to extract the values from right or left**

```
mysql> select left("hey buddy raju", 3);
+---------------------------+
| left("hey buddy raju", 3) |
+---------------------------+
| hey                       |
+---------------------------+
1 row in set (0.00 sec)

mysql> select right("hey buddy raju", 4);
+----------------------------+
| right("hey buddy raju", 4) |
+----------------------------+
| raju                       |
+----------------------------+
```

- **REPEAT : Used to repeat the strings**

```
mysql> SELECT REPEAT("hello ", 5);
+-----------------------------+
| REPEAT("hello ", 5)         |
+-----------------------------+
| hello hello hello hello hello  |
+-----------------------------+
1 row in set (0.00 sec)
```

- **TRIM : To trim off the extra spaces**

```
mysql> select trim("    Hello!    ");
+------------------------+
| trim("    Hello!    ") |
+------------------------+
| Hello!                 |
+------------------------+
```

==========

- **DISTINCT : This is used to get the unique values from the table. It omits the repeated values.**

```
mysql> select * from contacts;
+----+-----------+----------+------------+--------------+
| ID | FirstName | LastName | Contact    | City         |
+----+-----------+----------+------------+--------------+
|  1 | John      | Doe      | 1234567890 | Dallas       |
|  2 | Jane      | Smith    | 2345678901 | Chicago      |
|  3 | Michael   | Johnson  | 3456789012 | Chicago      |
|  4 | Emily     | Davis    | 4567890123 | Houston      |
|  5 | David     | Martinez | 5678901234 | Dallas       |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |
|  7 | James     | Brown    | 7890123456 | Chicago      |
|  8 | Olivia    | Wilson   | 8901234567 | San Diego    |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas       |
+----+-----------+----------+------------+--------------+
```

```
mysql> select distinct city from contacts;
+--------------+
| city         |
+--------------+
| Dallas       |
| Chicago      |
| Houston      |
| Philadelphia |
| San Diego    |
+--------------+
```

- **ORDER BY : This is used while sorting the data. We can sort the data in ascending or descending order or by alphabetical order.**

```
mysql> select * from contacts;
+----+-----------+----------+------------+--------------+
| ID | FirstName | LastName | Contact    | City         |
+----+-----------+----------+------------+--------------+
|  1 | John      | Doe      | 1234567890 | Dallas       |
|  2 | Jane      | Smith    | 2345678901 | Chicago      |
|  3 | Michael   | Johnson  | 3456789012 | Chicago      |
|  4 | Emily     | Davis    | 4567890123 | Houston      |
|  5 | David     | Martinez | 5678901234 | Dallas       |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |
|  7 | James     | Brown    | 7890123456 | Chicago      |
|  8 | Olivia    | Wilson   | 8901234567 | San Diego    |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas       |
+----+-----------+----------+------------+--------------+
```

- **Sorting in alphabetical order**

```
mysql> select id, firstname from contacts order by firstname;
+----+-----------+
| id | firstname |
+----+-----------+
|  9 | Daniel    |
|  5 | David     |
|  4 | Emily     |
|  7 | James     |
|  2 | Jane      |
|  1 | John      |
|  3 | Michael   |
|  8 | Olivia    |
|  6 | Sarah     |
+----+-----------+
```

- **Sorting in descending order**

```
mysql> select id, firstname from contacts order by id desc;
+----+-----------+
| id | firstname |
+----+-----------+
|  9 | Daniel    |
|  8 | Olivia    |
|  7 | James     |
|  6 | Sarah     |
|  5 | David     |
|  4 | Emily     |
|  3 | Michael   |
|  2 | Jane      |
|  1 | John      |
+----+-----------+
```

- **LIKE : We can use this to find the particular item.**

```
mysql> select * from contacts where city LIKE '%dallas%';
+----+-----------+----------+------------+--------+
| ID | FirstName | LastName | Contact    | City   |
+----+-----------+----------+------------+--------+
|  1 | John      | Doe      | 1234567890 | Dallas |
|  5 | David     | Martinez | 5678901234 | Dallas |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas |
+----+-----------+----------+------------+--------+
```

```
mysql> select * from contacts where city LIKE 'C_____';
+----+-----------+----------+------------+---------+
| ID | FirstName | LastName | Contact    | City    |
+----+-----------+----------+------------+---------+
|  2 | Jane      | Smith    | 2345678901 | Chicago |
|  3 | Michael   | Johnson  | 3456789012 | Chicago |
|  7 | James     | Brown    | 7890123456 | Chicago |
+----+-----------+----------+------------+---------+
```

- **How to alter the existing table:**

```
ALTER TABLE employees
ADD COLUMN
salary INT NOT NULL
```

- 


```
mysql> alter table contacts add column salary int not null default 25000;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from contacts;
+----+-----------+----------+------------+--------------+--------+
| ID | FirstName | LastName | Contact    | City         | salary |
+----+-----------+----------+------------+--------------+--------+
|  1 | John      | Doe      | 1234567890 | Dallas       |  25000 |
|  2 | Jane      | Smith    | 2345678901 | Chicago      |  25000 |
|  3 | Michael   | Johnson  | 3456789012 | Chicago      |  25000 |
|  4 | Emily     | Davis    | 4567890123 | Houston      |  25000 |
|  5 | David     | Martinez | 5678901234 | Dallas       |  25000 |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |  25000 |
|  7 | James     | Brown    | 7890123456 | Chicago      |  25000 |
|  8 | Olivia    | Wilson   | 8901234567 | San Diego    |  25000 |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas       |  25000 |
+----+-----------+----------+------------+--------------+--------+
```

- **LIMIT : It is used to get the values up to a certain limit**

```
mysql> select * from contacts LIMIT 5;
+----+-----------+----------+------------+---------+--------+
| ID | FirstName | LastName | Contact    | City    | salary |
+----+-----------+----------+------------+---------+--------+
|  1 | John      | Doe      | 1234567890 | Dallas  |  45100 |
|  2 | Jane      | Smith    | 2345678901 | Chicago |  32500 |
|  3 | Michael   | Johnson  | 3456789012 | Chicago |  45190 |
|  4 | Emily     | Davis    | 4567890123 | Houston |  55555 |
|  5 | David     | Martinez | 5678901234 | Dallas  |  25000 |
+----+-----------+----------+------------+---------+--------+
```

```
mysql> select * from contacts LIMIT 3,3;
+----+-----------+----------+------------+--------------+--------+
| ID | FirstName | LastName | Contact    | City         | salary |
+----+-----------+----------+------------+--------------+--------+
|  4 | Emily     | Davis    | 4567890123 | Houston      |  55555 |
|  5 | David     | Martinez | 5678901234 | Dallas       |  25000 |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |  52463 |
+----+-----------+----------+------------+--------------+--------+
```

- **We can combine it with order by and get the value of the highest salaried person**

```
mysql> select * from contacts ORDER BY SALARY DESC LIMIT 1;
+----+-----------+----------+------------+---------+--------+
| ID | FirstName | LastName | Contact    | City    | salary |
+----+-----------+----------+------------+---------+--------+
|  4 | Emily     | Davis    | 4567890123 | Houston |  55555 |
+----+-----------+----------+------------+---------+--------+
```

- **COUNT : This gives the count of records in the table.**

```
mysql> select count(*) from contacts;
+----------+
| count(*) |
+----------+
|        9 |
+----------+
```

- **We can use this to count the unique values as**

well.

- ```
  mysql> select count(distinct city) from contacts;
  +----------------------+
  | count(distinct city) |
  +----------------------+
  |                    5 |
  +----------------------+
  ```
- We can also use this to find the particular count based on particular values as well.

```
mysql> select count(id) , city from contacts where city = "dallas";
+-----------+--------+
| count(id) | city   |
+-----------+--------+
|         3 | Dallas |
+-----------+--------+
```

- GROUP BY : This groups all the similar values and provides the output.

- ```
  mysql> select city from contacts group by city;
  +--------------+
  | city         |
  +--------------+
  | Dallas       |
  | Chicago      |
  | Houston      |
  | Philadelphia |
  | San Diego    |
  +--------------+
  ```

- How to use it. Suppose we need how many customers are there in dallas then we can use this.

- ```
  mysql> select city, count(id) from contacts group by city;
  +--------------+-----------+
  | city         | count(id) |
  +--------------+-----------+
  | Dallas       |         3 |
  | Chicago      |         3 |
  | Houston      |         1 |
  | Philadelphia |         1 |
  | San Diego    |         1 |
  +--------------+-----------+
  ```

- The above tables gives us the count of ids in the particular cities.

==========
===

- MIN & MAX :

- ```
  mysql> SELECT MAX(SALARY) FROM CONTACTS;
  +-------------+
  | MAX(SALARY) |
  +-------------+
  |       55555 |
  +-------------+
  1 row in set (0.00 sec)

  mysql> SELECT MIN(SALARY) FROM CONTACTS;
  +-------------+
  | MIN(SALARY) |
  +-------------+
  |       23565 |
  +-------------+
  ```

- SUB QUERIES : This is one of the most important topics of sql. Here we write a query inside another query.
- In sub queries first the query inside the brackets gets executed then it moves on to the first query.

- ```
  SELECT emp_id, fname, salary FROM employees
    WHERE
  salary = (SELECT MAX(salary) FROM employees);
  ```
- ```
  mysql> select id, firstname from contacts
      -> where
      -> salary = (select max(salary) from contacts);
  +----+-----------+
  | id | firstname |
  +----+-----------+
  |  4 | Emily     |
  +----+-----------+
  ```

==========

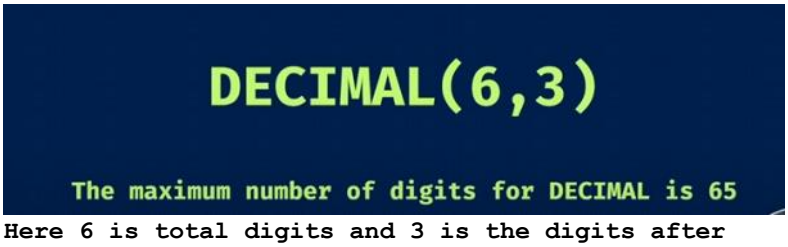
- **SUM & AVG**
- ```
  mysql> SELECT SUM(SALARY) FROM CONTACTS;
  +-------------+
  | SUM(SALARY) |
  +-------------+
  |      340885 |
  +-------------+
  1 row in set (0.00 sec)

  mysql> SELECT AVG(SALARY) FROM CONTACTS;
  +-------------+
  | AVG(SALARY) |
  +-------------+
  |  37876.1111 |
  +-------------+
  ```
- ```
  mysql> SELECT CITY, SUM(SALARY) FROM CONTACTS GROUP BY CITY;
  +--------------+-------------+
  | CITY         | SUM(SALARY) |
  +--------------+-------------+
  | Dallas       |       95100 |
  | Chicago      |      101255 |
  | Houston      |       55555 |
  | Philadelphia |       52463 |
  | San Diego    |       36512 |
  +--------------+-------------+
  5 rows in set (0.00 sec)

  mysql> SELECT CITY, COUNT(ID) ,SUM(SALARY) FROM CONTACTS GROUP BY CITY;
  +--------------+-----------+-------------+
  | CITY         | COUNT(ID) | SUM(SALARY) |
  +--------------+-----------+-------------+
  | Dallas       |         3 |       95100 |
  | Chicago      |         3 |      101255 |
  | Houston      |         1 |       55555 |
  ```

==========

DATA TYPES

- **DECIMAL : Generally int data type won't store 59.69 it will store 59 instead so to eradicate that error we use decimal.**
- 

- **Here 6 is total digits and 3 is the digits after decimal.**

- 


```
mysql> create table num(price decimal(5,2));
```

```
mysql> desc num;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| price | decimal(5,2) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

```
mysql> insert into num values (155.78);
Query OK, 1 row affected (0.00 sec)

mysql> select * from num;
+--------+
| price  |
+--------+
| 155.78 |
+--------+
```

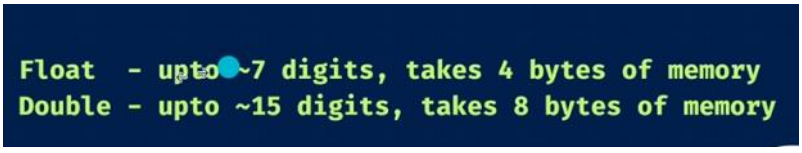
- No matter how many digits we enter after the decimal it will round them off to the prescribed value.
- Ex: As we have set the limit to 5 this won't get executed

```
mysql> insert into num values (185445.5632174),(23654.3642);
ERROR 1264 (22003): Out of range value for column 'price' at row 1
```

- The value after the decimal converted to 2 points

```
mysql> insert into num values (256.696969);
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> select * from num;
+--------+
| price  |
+--------+
| 155.78 |
| 155.59 |
| 256.70 |
+--------+
```

- **FLOAT / DOUBLE:**
- 


```
mysql> CREATE TABLE NUM1(F FLOAT ,D DOUBLE );
Query OK, 0 rows affected (0.02 sec)

mysql> DESC NUM1;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| F     | float  | YES  |     | NULL    |       |
| D     | double | YES  |     | NULL    |       |
+-------+--------+------+-----+---------+-------+
```

- **AS the name suggests the float takes less values and double takes more values after the decimal.**

```
mysql> insert into num1 values(123.123456789, 123.123456789);
Query OK, 1 row affected (0.01 sec)

mysql> select * from num1;
+---------+---------------+
| F       | D             |
+---------+---------------+
| 123.123 | 123.123456789 |
```

==========

- **DATE :**

- **TIME:**

- **DATETIME :**

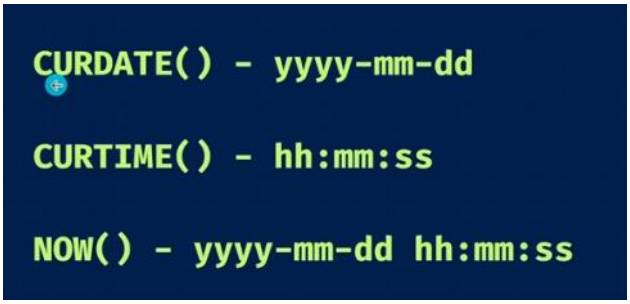
- **We can add the values as date time and date time in the table using these.**

```
mysql> INSERT INTO PERSON VALUES('2024-11-25','12:01:00', '2024-11-25 12:02:
23');
Query OK, 1 row affected (0.00 sec)

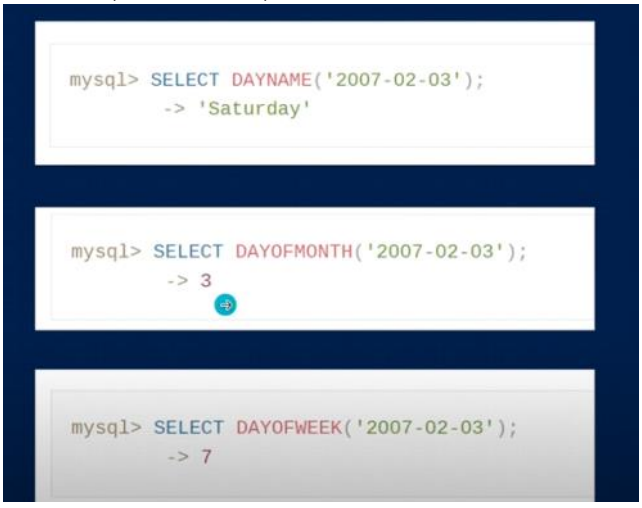
mysql> SELECT * FROM PERSON;
+------------+----------+---------------------+
| d          | t        | dt                  |
+------------+----------+---------------------+
| 2024-11-25 | 12:01:00 | 2024-11-25 12:02:23 |
+------------+----------+---------------------+
```

==========

- **DATE TIME FUNCTIONS:**

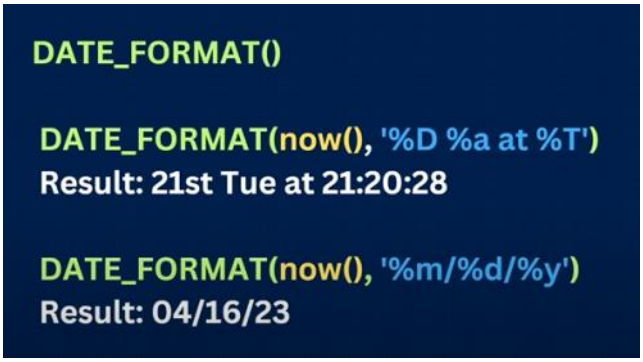


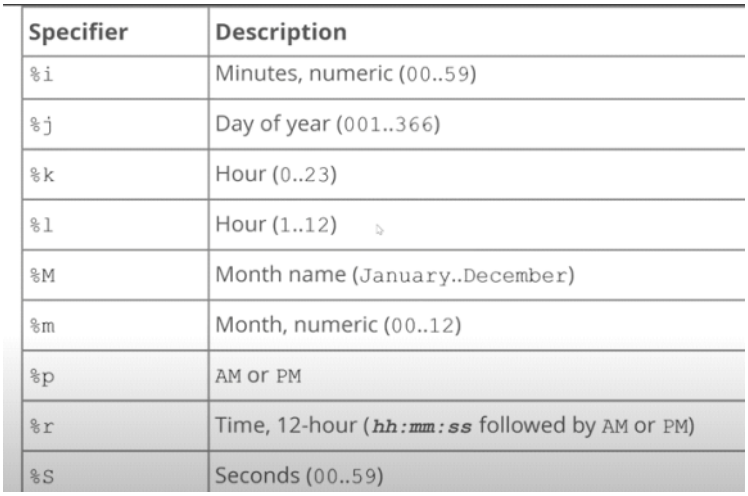
-
```
mysql> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2024-08-23 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT CURTIME();
+-----------+
| CURTIME() |
+-----------+
| 15:43:46  |
+-----------+
```
-
```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2024-08-23 15:43:53 |
+---------------------+
1 row in set (0.00 sec)
```
- **DAYNAME, DAYMONTH, DAY OF WEEK:**
- 

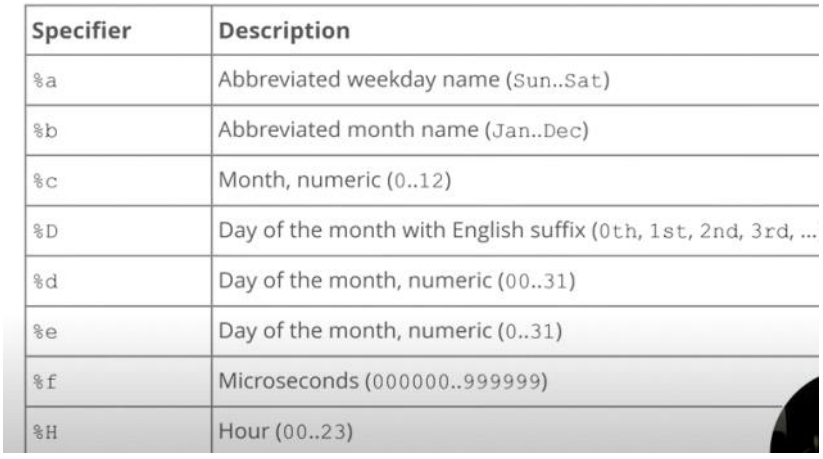
- **MONTHNAME & HOUR:**
- 

- 

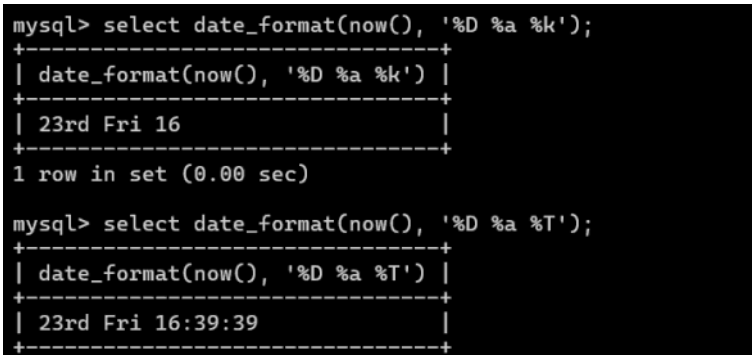
- 

- 

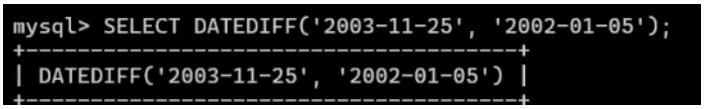
Specifier	Description
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (*hh:mm:ss* followed by AM or PM)
%S	Seconds (00..59)



Specifier	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)

- 

- DATEDIFF : WE USE THIS TO FIND THE DIFFERENCE BETWEEN THE DATES

- 

- ```
  mysql> SELECT DATEDIFF('2003-11-25', '2002-01-05');
  +-------------------------------------+
  | DATEDIFF('2003-11-25', '2002-01-05') |
  +-------------------------------------+
  |                                 689 |
  +-------------------------------------+
  ```

- DATE ADD / DATE SUB :
- 

- ```
  mysql> SELECT DATE_ADD(NOW(), INTERVAL 4 DAY);
  +---------------------------------+
  | DATE_ADD(NOW(), INTERVAL 4 DAY) |
  +---------------------------------+
  | 2024-08-27 16:49:37             |
  +---------------------------------+
  1 row in set (0.00 sec)
  ```
- ```
  mysql> SELECT DATE_SUB(NOW(), INTERVAL 7 DAY);
  +---------------------------------+
  | DATE_SUB(NOW(), INTERVAL 7 DAY) |
  +---------------------------------+
  | 2024-08-16 16:50:05             |
  +---------------------------------+
  1 row in set (0.00 sec)
  ```
- FOR EXAMPLE WE HAVE DATE OF BIRTHS AND WE NEED TO FIND OUT WHEN THEY WILL TURN INTO 25 YEARS WE CAN USE THIS
- ```
  mysql> SELECT DATE_ADD('2003-11-25', INTERVAL 25 YEAR);
  +-----------------------------------------+
  | DATE_ADD('2003-11-25', INTERVAL 25 YEAR) |
  +-----------------------------------------+
  | 2028-11-25                              |
  +-----------------------------------------+
  ```
- ==========
  ==========

- TIME DIFF:
- 

  ==========

- ON UPDATE : Consider this use case if a user tweets a tweet at a particular time and then updates after sometime. To capture both the time we can use this.

```
mysql> INSERT INTO TWEET (TWEETS) VALUES ('HEY THERE EVERYONE');
Query OK, 1 row affected (0.00 sec)

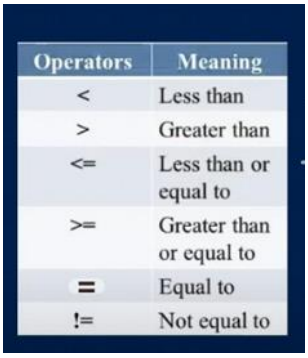
mysql> SELECT * FROM TWEET;
+--------------------+---------------------+------------+
| tweets             | created_at          | updated_at |
+--------------------+---------------------+------------+
| HEY THERE EVERYONE | 2024-08-23 17:04:10 | NULL       |
+--------------------+---------------------+------------+
```

- NOW LETS UPDATE IT

```
mysql> UPDATE TWEET SET TWEETS = "THIS IS CHANGED NOW";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM TWEET;
+--------------------+---------------------+---------------------+
| tweets             | created_at          | updated_at          |
+--------------------+---------------------+---------------------+
| THIS IS CHANGED NOW | 2024-08-23 17:04:10 | 2024-08-23 17:05:15 |
+--------------------+---------------------+---------------------+
```

- ==========

- **OPERATORS:**
- 

Operators	Meaning
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
!=	Not equal to

- **LOGICAL OPERATORS:**
- 

- **AND : This is used when you need the two conditions to be true. For ex: you need a boy under 18 years**
- **Then you need to use age <18 and gender = Male**

- **OR : This is used when you want either of the conditions to be true. It will return answer if one of the condition is true.**
- **For the above condition : age <18 or gender = Male**
- **In the above example you will either get ppl(including girls as well) below age 18 and boys who may be above 18.**

==========

- **IN & NOT IN: IN is used to find the values that are in the particular column and NOT IN is opposite to it.**

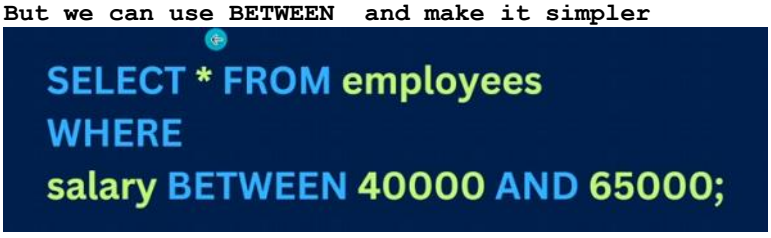
```
mysql> SELECT * FROM CONTACTS WHERE CITY IN ("DALLAS", "cHICAGO");
+----+-----------+----------+------------+---------+--------+
| ID | FirstName | LastName | Contact    | City    | salary |
+----+-----------+----------+------------+---------+--------+
|  1 | John      | Doe      | 1234567890 | Dallas  |  45100 |
|  2 | Jane      | Smith    | 2345678901 | Chicago |  32500 |
|  3 | Michael   | Johnson  | 3456789012 | Chicago |  45190 |
|  5 | David     | Martinez | 5678901234 | Dallas  |  25000 |
|  7 | James     | Brown    | 7890123456 | Chicago |  23565 |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas  |  25000 |
+----+-----------+----------+------------+---------+--------+
```

NOT IN:

```
mysql> SELECT * FROM CONTACTS WHERE CITY not IN ("DALLAS", "cHICAGO");
+----+-----------+----------+------------+--------------+--------+
| ID | FirstName | LastName | Contact    | City         | salary |
+----+-----------+----------+------------+--------------+--------+
|  4 | Emily     | Davis    | 4567890123 | Houston      |  55555 |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |  52463 |
|  8 | Olivia    | Wilson   | 8901234567 | San Diego    |  36512 |
+----+-----------+----------+------------+--------------+--------+
```

- **BETWEEN : The name is self explanatory we use this to find the values in between two values.**
- **Generally we find it like this**

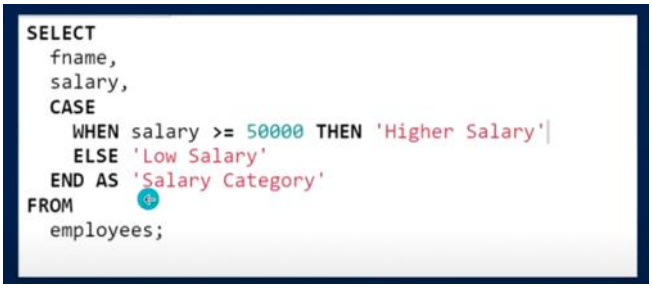
- **But we can use BETWEEN and make it simpler**



==========

- **CASE :**

- This works as an IF statement in SQL as per my knowledge. We can separate the values in the table on a condition. Like if age >=18 can vote if not cannot vote.
- SYNTAX:



```
SELECT
  fname,
  salary,
  CASE
    WHEN salary >= 50000 THEN 'Higher Salary'
    ELSE 'Low Salary'
  END AS 'Salary Category'
FROM
  employees;
```

```
mysql> SELECT FIRSTNAME, SALARY, CASE WHEN SALARY >=40000 THEN 'HIGH'
    -> ELSE 'LOW' END AS 'CATEGORY' FROM CONTACTS;
+-----------+--------+----------+
| FIRSTNAME | SALARY | CATEGORY |
+-----------+--------+----------+
| John      |  45100 | HIGH     |
| Jane      |  32500 | LOW      |
| Michael   |  45190 | HIGH     |
| Emily     |  55555 | HIGH     |
| David     |  25000 | LOW      |
| Sarah     |  52463 | HIGH     |
| James     |  23565 | LOW      |
| Olivia    |  36512 | LOW      |
| Daniel    |  25000 | LOW      |
+-----------+--------+----------+
```

- WE CAN ALSO INCLUDE MORE CONDITIONS IN THE CASE LIKE BELOW

```
SELECT
  fname,
  salary,
  CASE
    WHEN salary >= 50000 THEN 'Higher Salary'
    WHEN salary >= 40000
    AND salary < 50000 THEN 'Mid Salary'
    ELSE 'Low Salary'
  END AS 'Salary Category'
FROM
  employees;
```

==========

- IS NULL : This is used to find the null values in the table.

==========

- **UNIQUE : As the name is self-explanatory it only takes in the unique values like primary key but this has another use case. If we need phone numbers then we can use this unique function to take the exact details without any errors.**

```
CREATE TABLE contacts(
    name VARCHAR(50),
    mob VARCHAR(15) UNIQUE CHECK (LENGTH(mob) >= 10)
);
```

- **When the user tries to input 10 or more digits it will accept but if the digits are less than 10 then it throws an error as below**

```
mysql> create table phone(num varchar(15) unique check (length(num) >= 10));
Query OK, 0 rows affected (0.04 sec)


mysql> desc phone;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| num   | varchar(15) | YES  | UNI | NULL    |       |
+-------+-------------+------+-----+---------+-------+
1 row in set (0.01 sec)

mysql> insert into phone values (1236547896);
Query OK, 1 row affected (0.01 sec)

mysql> insert into phone values (123654);
ERROR 3819 (HY000): Check constraint 'phone_chk_1' is violated.
```

```
mysql> insert into phone values (123654);
ERROR 3819 (HY000): Check constraint 'phone_chk_1' is violated.
mysql>
```

- **If someone else is working on the database and finds this error. It will be hard for them to find out what's wrong so let's add a constraint.**

```
CREATE TABLE contacts(
    name VARCHAR(50),
    mob VARCHAR(15) UNIQUE,
    CONSTRAINT mob_no_less_than_10digits CHECK (LENGTH(mob) >= 10)
);
```

- **This is how it works:**

```
mysql> create table phone2(contacts varchar(15) unique, constraint phn_no_le
ss_than_10 check(length(contacts)>=10));
Query OK, 0 rows affected (0.03 sec)

mysql> desc phone2;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| contacts | varchar(15) | YES  | UNI | NULL    |       |
+----------+-------------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql> insert into phone2 values (123654);
ERROR 3819 (HY000): Check constraint 'phn_no_less_than_10' is violated.
```

- **==========**
  ==========

- **ALTER : USING THIS WE CAN ALTER THE TABLE IT MEANS WE CAN ADD NEW COLUMNS TO IT.**
- **Let me provide you an example this is the table I have.**

```
mysql> select * from contacts;
+----+-----------+----------+------------+--------------+--------+
| ID | FirstName | LastName | Contact    | City         | salary |
+----+-----------+----------+------------+--------------+--------+
|  1 | John      | Doe      | 1234567890 | Dallas       |  45100 |
|  2 | Jane      | Smith    | 2345678901 | Chicago      |  32500 |
|  3 | Michael   | Johnson  | 3456789012 | Chicago      |  45190 |
|  4 | Emily     | Davis    | 4567890123 | Houston      |  55555 |
|  5 | David     | Martinez | 5678901234 | Dallas       |  25000 |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |  52463 |
|  7 | James     | Brown    | 7890123456 | Chicago      |  23565 |
|  8 | Olivia    | Wilson   | 8901234567 | San Diego    |  36512 |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas       |  25000 |
+----+-----------+----------+------------+--------------+--------+
```

- **Let me add some new column to it like gender.**

```
mysql> ALTER TABLE CONTACTS ADD COLUMN gender VARCHAR(10);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

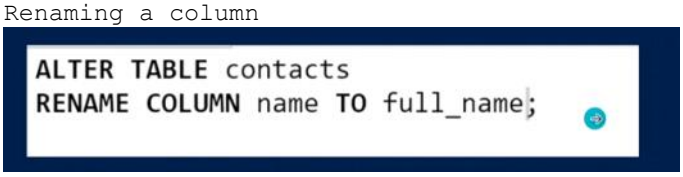
mysql> select * from contacts;
+----+-----------+----------+------------+--------------+--------+--------+
| ID | FirstName | LastName | Contact    | City         | salary | gender |
+----+-----------+----------+------------+--------------+--------+--------+
|  1 | John      | Doe      | 1234567890 | Dallas       |  45100 | NULL   |
|  2 | Jane      | Smith    | 2345678901 | Chicago      |  32500 | NULL   |
|  3 | Michael   | Johnson  | 3456789012 | Chicago      |  45190 | NULL   |
|  4 | Emily     | Davis    | 4567890123 | Houston      |  55555 | NULL   |
|  5 | David     | Martinez | 5678901234 | Dallas       |  25000 | NULL   |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |  52463 | NULL   |
|  7 | James     | Brown    | 7890123456 | Chicago      |  23565 | NULL   |
|  8 | Olivia    | Wilson   | 8901234567 | San Diego    |  36512 | NULL   |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas       |  25000 | NULL   |
+----+-----------+----------+------------+--------------+--------+--------+
```

- **WE CAN ALSO REMOVE THE COLUMN USING THIS**

```
mysql> ALTER TABLE CONTACTS DROP COLUMN GENDER  ;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM CONTACTS;
+----+-----------+----------+------------+--------------+--------+
| ID | FirstName | LastName | Contact    | City         | salary |
+----+-----------+----------+------------+--------------+--------+
|  1 | John      | Doe      | 1234567890 | Dallas       |  45100 |
|  2 | Jane      | Smith    | 2345678901 | Chicago      |  32500 |
|  3 | Michael   | Johnson  | 3456789012 | Chicago      |  45190 |
|  4 | Emily     | Davis    | 4567890123 | Houston      |  55555 |
|  5 | David     | Martinez | 5678901234 | Dallas       |  25000 |
|  6 | Sarah     | Lee      | 6789012345 | Philadelphia |  52463 |
|  7 | James     | Brown    | 7890123456 | Chicago      |  23565 |
|  8 | Olivia    | Wilson   | 8901234567 | San Diego    |  36512 |
|  9 | Daniel    | Garcia   | 9012345678 | Dallas       |  25000 |
+----+-----------+----------+------------+--------------+--------+
```

- **==========**

- **HOW TO RENAME A COLUMN OR TABLE NAME**
- Renaming a column
- 


```
mysql> alter table contacts rename column id to emp_id;
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from contacts;
+--------+-----------+----------+------------+--------------+--------+
| emp_id | FirstName | LastName | Contact    | City         | salary |
+--------+-----------+----------+------------+--------------+--------+
|      1 | John      | Doe      | 1234567890 | Dallas       |  45100 |
|      2 | Jane      | Smith    | 2345678901 | Chicago      |  32500 |
|      3 | Michael   | Johnson  | 3456789012 | Chicago      |  45190 |
|      4 | Emily     | Davis    | 4567890123 | Houston      |  55555 |
|      5 | David     | Martinez | 5678901234 | Dallas       |  25000 |
|      6 | Sarah     | Lee      | 6789012345 | Philadelphia |  52463 |
|      7 | James     | Brown    | 7890123456 | Chicago      |  23565 |
|      8 | Olivia    | Wilson   | 8901234567 | San Diego    |  36512 |
|      9 | Daniel    | Garcia   | 9012345678 | Dallas       |  25000 |
+--------+-----------+----------+------------+--------------+--------+
```

- **How to rename a table**

-
```
ALTER TABLE contacts
RENAME TO mycontacts;
```

-
```
mysql> alter table contacts RENAME TO employees;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from employees;
+--------+-----------+----------+------------+--------------+--------+
| emp_id | FirstName | LastName | Contact    | City         | salary |
+--------+-----------+----------+------------+--------------+--------+
|      1 | John      | Doe      | 1234567890 | Dallas       |  45100 |
|      2 | Jane      | Smith    | 2345678901 | Chicago      |  32500 |
|      3 | Michael   | Johnson  | 3456789012 | Chicago      |  45190 |
|      4 | Emily     | Davis    | 4567890123 | Houston      |  55555 |
|      5 | David     | Martinez | 5678901234 | Dallas       |  25000 |
|      6 | Sarah     | Lee      | 6789012345 | Philadelphia |  52463 |
|      7 | James     | Brown    | 7890123456 | Chicago      |  23565 |
|      8 | Olivia    | Wilson   | 8901234567 | San Diego    |  36512 |
|      9 | Daniel    | Garcia   | 9012345678 | Dallas       |  25000 |
+--------+-----------+----------+------------+--------------+--------+
```

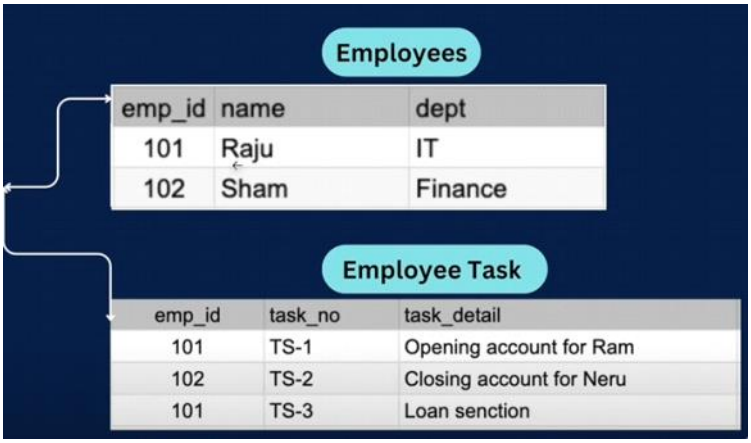
- **ALTERING THE DATATYPE IN THE COLUMN**
- **Lets add default values to the existing values**
-
```
ALTER TABLE contacts
MODIFY mob VARCHAR(15) DEFAULT 'unknown';
```

- **==========**

- **RELATIONSHIP :**
-
  - One to One
  - One to Many
  - Many to Many

- **ONE TO ONE :**
-


- **ONE TO MANY :**

- 

- **As you can see that raju is assigned to 2 tasks**

- **MANY TO MANY : IN SIMPLE WORDS MANY TABLES ARE LINKED WITH EACH OTHER.**

- **In most of the cases we come across ONE TO MANY relations.**

==========

- **FOREIGN KEY : When we use a primary key as a reference key in another table then it is called as foreign key.**
- **While creating a table we can mention the foreign key in it**
-
```
CREATE TABLE orders(
  ord_id INT AUTO_INCREMENT PRIMARY KEY,
  date DATE,
  amount DECIMAL(10, 2),
  cust_id INT,
  FOREIGN KEY (cust_id) REFERENCES customers(cust_id)
);
```

- **How to check the foreign key**

SELECT constraint_name, column_name, referenced_table_name
FROM information_schema.key_column_usage
WHERE table_name = 'orders';

-
```
mysql> SELECT CONSTRAINT_NAME, COLUMN_NAME, REFERENCED_TABLE_NAME FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE WHERE TABLE_NAME='orders';
+-----------------+-------------+-----------------------+
| CONSTRAINT_NAME | COLUMN_NAME | REFERENCED_TABLE_NAME |
+-----------------+-------------+-----------------------+
| PRIMARY         | ord_id      | NULL                  |
| orders_ibfk_1   | cust_id     | customers             |
+-----------------+-------------+-----------------------+
```

- **One thing about the foreign key is that if you use that in your table and try to enter the wrong foreign key value in it then you will get an error.**
- **==========**

- **JOINS: This is used to join different tables based on a particular column which is common between them.**
- **For ex : I have the following two tables**

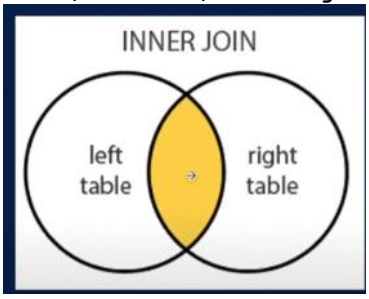
```
mysql> select * from pizzas;
+---------+----------------+--------+-------+
| PizzaID | PizzaName      | Size   | Price |
+---------+----------------+--------+-------+
|       1 | Margherita     | Small  |  8.99 |
|       2 | Pepperoni      | Medium | 10.99 |
|       3 | BBQ Chicken    | Large  | 12.99 |
|       4 | Veggie Delight | Medium |  9.99 |
|       5 | Hawaiian       | Large  | 11.99 |
+---------+----------------+--------+-------+
5 rows in set (0.00 sec)

mysql> select * from pizzaorders;
+---------+--------------+---------+----------+------------+
| OrderID | CustomerName | PizzaID | Quantity | OrderDate  |
+---------+--------------+---------+----------+------------+
|     101 | John Doe     |       1 |        2 | 2024-08-24 |
|     102 | Jane Smith   |       3 |        1 | 2024-08-24 |
|     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
```

- **TYPES OF JOINS**
- **CROSS JOIN : Every row from one table is combined with every row from another table.**

```
mysql> select * from pizzas, pizzaorders;
+---------+----------------+--------+-------+---------+---------------+---------+----------+------------+
| PizzaID | PizzaName      | Size   | Price | OrderID | CustomerName  | PizzaID | Quantity | OrderDate  |
+---------+----------------+--------+-------+---------+---------------+---------+----------+------------+
|       5 | Hawaiian       | Large  | 11.99 |     101 | John Doe      |       1 |        2 | 2024-08-24 |
|       4 | Veggie Delight | Medium |  9.99 |     101 | John Doe      |       1 |        2 | 2024-08-24 |
|       3 | BBQ Chicken    | Large  | 12.99 |     101 | John Doe      |       1 |        2 | 2024-08-24 |
|       2 | Pepperoni      | Medium | 10.99 |     101 | John Doe      |       1 |        2 | 2024-08-24 |
|       1 | Margherita     | Small  |  8.99 |     101 | John Doe      |       1 |        2 | 2024-08-24 |
|       5 | Hawaiian       | Large  | 11.99 |     102 | Jane Smith    |       3 |        1 | 2024-08-24 |
|       4 | Veggie Delight | Medium |  9.99 |     102 | Jane Smith    |       3 |        1 | 2024-08-24 |
|       3 | BBQ Chicken    | Large  | 12.99 |     102 | Jane Smith    |       3 |        1 | 2024-08-24 |
|       2 | Pepperoni      | Medium | 10.99 |     102 | Jane Smith    |       3 |        1 | 2024-08-24 |
|       1 | Margherita     | Small  |  8.99 |     102 | Jane Smith    |       3 |        1 | 2024-08-24 |
|       5 | Hawaiian       | Large  | 11.99 |     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
|       4 | Veggie Delight | Medium |  9.99 |     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
|       3 | BBQ Chicken    | Large  | 12.99 |     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
|       2 | Pepperoni      | Medium | 10.99 |     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
|       1 | Margherita     | Small  |  8.99 |     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
|       5 | Hawaiian       | Large  | 11.99 |     104 | Bob Brown     |       1 |        1 | 2024-08-23 |
|       4 | Veggie Delight | Medium |  9.99 |     104 | Bob Brown     |       1 |        1 | 2024-08-23 |
|       3 | BBQ Chicken    | Large  | 12.99 |     104 | Bob Brown     |       1 |        1 | 2024-08-23 |
|       2 | Pepperoni      | Medium | 10.99 |     104 | Bob Brown     |       1 |        1 | 2024-08-23 |
|       1 | Margherita     | Small  |  8.99 |     104 | Bob Brown     |       1 |        1 | 2024-08-23 |
|       5 | Hawaiian       | Large  | 11.99 |     105 | Charlie Lee   |       4 |        2 | 2024-08-22 |
|       4 | Veggie Delight | Medium |  9.99 |     105 | Charlie Lee   |       4 |        2 | 2024-08-22 |
|       3 | BBQ Chicken    | Large  | 12.99 |     105 | Charlie Lee   |       4 |        2 | 2024-08-22 |
|       2 | Pepperoni      | Medium | 10.99 |     105 | Charlie Lee   |       4 |        2 | 2024-08-22 |
|       1 | Margherita     | Small  |  8.99 |     105 | Charlie Lee   |       4 |        2 | 2024-08-22 |
+---------+----------------+--------+-------+---------+---------------+---------+----------+------------+
```

- **INNER JOIN : Returns only the rows where there is a match between the specified columns in both the left(or first) and right(or second) tables.**



- **Only the yellow part is showed in the result**

  For ex : I have two following tables let me do an inner join

```
mysql> select * from pizzas;
+---------+----------------+--------+-------+
| PizzaID | PizzaName      | Size   | Price |
+---------+----------------+--------+-------+
|       1 | Margherita     | Small  |  8.99 |
|       2 | Pepperoni      | Medium | 10.99 |
|       3 | BBQ Chicken    | Large  | 12.99 |
|       4 | Veggie Delight | Medium |  9.99 |
|       5 | Hawaiian       | Large  | 11.99 |
+---------+----------------+--------+-------+
```

```
mysql> select * from pizzaorders;
+---------+---------------+---------+----------+------------+
| OrderID | CustomerName  | PizzaID | Quantity | OrderDate  |
+---------+---------------+---------+----------+------------+
|     101 | John Doe      |       1 |        2 | 2024-08-24 |
|     102 | Jane Smith    |       3 |        1 | 2024-08-24 |
|     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
|     104 | Bob Brown     |       1 |        1 | 2024-08-23 |
|     105 | Charlie Lee   |       4 |        2 | 2024-08-22 |
+---------+---------------+---------+----------+------------+
```

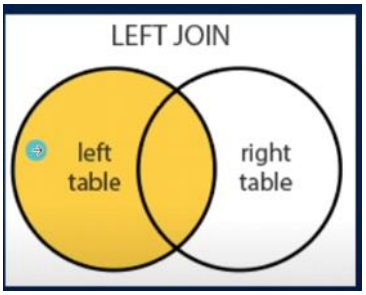
  After the inner join you can see that the 5th pizza id is not present because there were no orders placed on that pizza.

```
mysql> select * from pizzas INNER JOIN pizzaorders on pizzaorders.pizzaid = pizzas.pizzaid;
+---------+----------------+--------+-------+---------+---------------+---------+----------+------------+
| PizzaID | PizzaName      | Size   | Price | OrderID | CustomerName  | PizzaID | Quantity | OrderDate  |
+---------+----------------+--------+-------+---------+---------------+---------+----------+------------+
|       1 | Margherita     | Small  |  8.99 |     101 | John Doe      |       1 |        2 | 2024-08-24 |
|       1 | Margherita     | Small  |  8.99 |     104 | Bob Brown     |       1 |        1 | 2024-08-23 |
|       2 | Pepperoni      | Medium | 10.99 |     103 | Alice Johnson |       2 |        3 | 2024-08-23 |
|       3 | BBQ Chicken    | Large  | 12.99 |     102 | Jane Smith    |       3 |        1 | 2024-08-24 |
|       4 | Veggie Delight | Medium |  9.99 |     105 | Charlie Lee   |       4 |        2 | 2024-08-22 |
+---------+----------------+--------+-------+---------+---------------+---------+----------+------------+
```

- **If you use some previous tricks you can simply find that what pizza earned how much with this**

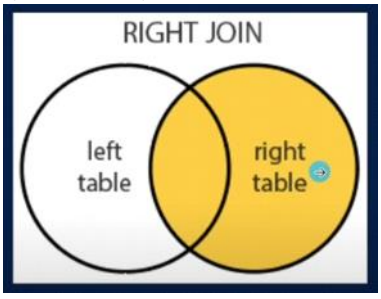
```
mysql> select  pizzaname , sum(price) from pizzas INNER JOIN pizzaorders on
pizzaorders.pizzaid = pizzas.pizzaid group by  pizzaname;
+----------------+------------+
| pizzaname      | sum(price) |
+----------------+------------+
| Margherita     |      17.98 |
| Pepperoni      |      10.99 |
| BBQ Chicken    |      12.99 |
| Veggie Delight |       9.99 |
+----------------+------------+
```

- **LEFT JOIN : RETURNS ALL ROWS FROM THE LEFT(OR FIRST) TABLE AND THE MATCHING ROWS FROM THE RIGHT (OR SECOND) TABLE.**

- 


- ```
mysql> SELECT * FROM PIZZAS
    -> LEFT JOIN
    -> PIZZAORDERS
    -> ON PIZZAORDERS.PIZZAID = PIZZAS.PIZZAID;
+---------+---------------+--------+-------+---------+--------------+---------+----------+------------+
| PizzaID | PizzaName | Size | Price | OrderID | CustomerName | PizzaID | Quantity | OrderDate |
+---------+---------------+--------+-------+---------+--------------+---------+----------+------------+
1	Margherita	Small	8.99	101	John Doe	1	2	2024-08-24
1	Margherita	Small	8.99	104	Bob Brown	1	1	2024-08-23
2	Pepperoni	Medium	10.99	103	Alice Johnson	2	3	2024-08-23
3	BBQ Chicken	Large	12.99	102	Jane Smith	3	1	2024-08-24
4	Veggie Delight	Medium	9.99	105	Charlie Lee	4	2	2024-08-22
5	Hawaiian	Large	11.99	NULL	NULL	NULL	NULL	NULL
+---------+---------------+--------+-------+---------+--------------+---------+----------+------------+
```

- **Here pizzas is the first (left) table and pizzaorders is the right table.**

- **RIGHT JOIN : RETURNS ALL ROWS FROM THE RIGHT(OR SECOND) TABLE AND THE MATCHING ROWS FROM THE LEFT (OR FIRST) TABLE.**
- 


- ```
mysql> select * from pizzas right join pizzaorders on pizzaorders.pizzaid = pizzas.pizzaid;
+---------+---------------+--------+-------+---------+--------------+---------+----------+------------+
| PizzaID | PizzaName     | Size   | Price | OrderID | CustomerName | PizzaID | Quantity | OrderDate  |
+---------+---------------+--------+-------+---------+--------------+---------+----------+------------+
|       1 | Margherita    | Small  |  8.99 |     101 | John Doe     |       1 |        2 | 2024-08-24 |
|       3 | BBQ Chicken   | Large  | 12.99 |     102 | Jane Smith   |       3 |        1 | 2024-08-24 |
|       2 | Pepperoni     | Medium | 10.99 |     103 | Alice Johnson|       2 |        3 | 2024-08-23 |
|       1 | Margherita    | Small  |  8.99 |     104 | Bob Brown    |       1 |        1 | 2024-08-23 |
|       4 | Veggie Delight| Medium |  9.99 |     105 | Charlie Lee  |       4 |        2 | 2024-08-22 |
+---------+---------------+--------+-------+---------+--------------+---------+----------+------------+
```

==========

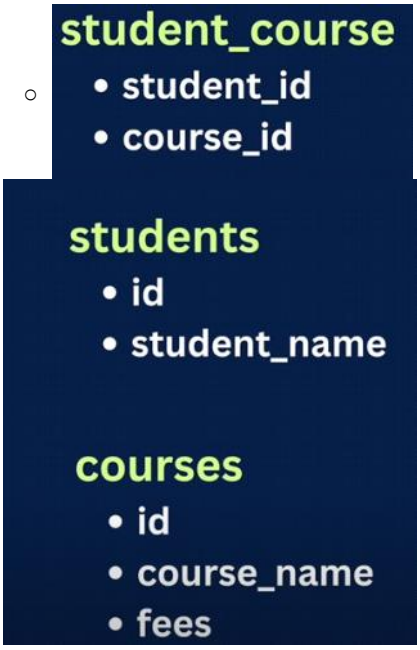
- **DELETE : Suppose we want to delete a pizza or customer from our database then we can use DELETE.**
- **When we try to delete the name from the foreign key table we will get an error that it cannot be deleted.**
- ```
mysql> DELETE FROM PIZZAS
    -> WHERE PIZZANAME = 'PEPPERONI';
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key cons
traint fails (`practice`.`pizzaorders`, CONSTRAINT `pizzaorders_ibfk_1` FORE
IGN KEY (`PizzaID`) REFERENCES `pizzas` (`PizzaID`))
```
```

- **To eradicate this we can use the below method while creating the table itself.**

```
CREATE TABLE orders(
    ord_id INT AUTO_INCREMENT PRIMARY KEY,
    date DATE,
    amount DECIMAL(10, 2),
    cust_id INT,
    FOREIGN KEY (cust_id) REFERENCES customers(cust_id) ON DELETE CASCADE
)


INSERT INTO orders (date, amount, cust_id)
VALUES (CURDATE(), 100.50, 1), (CURDATE(), 500.40, 2), (CURDATE(), 300.30, 1);
```

==========

- **MANT TO MANY :**
  - 

- **HERE STUDENT COURSE IS THE JUNCTION TABLE USING THAT WE CAN JOIN THE OTHER TWO TABLES**

```
# Joins with Many to Many relationship
SELECT student_name, course_name FROM students
JOIN
    student_course ON student_course.student_id=students.id
JOIN
    courses ON student_course.course_id=courses.id;
```

==========

- **VIEW(virtual table) : Instead of using the same query every time we can simply create a virtual table of it and use that to query the required values.**
- **To create a view you just need to follow the below format.**
- **CREATE VIEW table_name AS**
- **Select pizzaname, price from pizzas join pizzaorder on pizzas.pizzaid = pizzaorder.pizzaid;**

- **The above query will create a virtual table which will contain the mentioned details and it can be handy for certain queries.**

- **To remove the virtual table you just need to enter the following query.**

- ***Drop view table_name;***

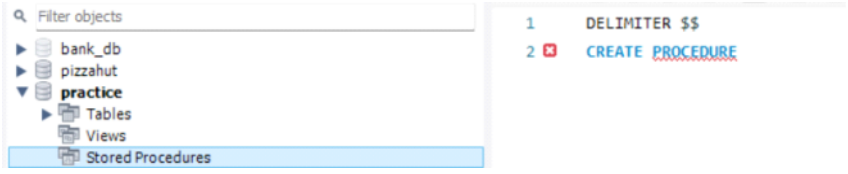
==========

- **HAVING CLAUSE : Whenever we want to group by something and in that we need something specific like fee_paid >35000 then we can use this.**

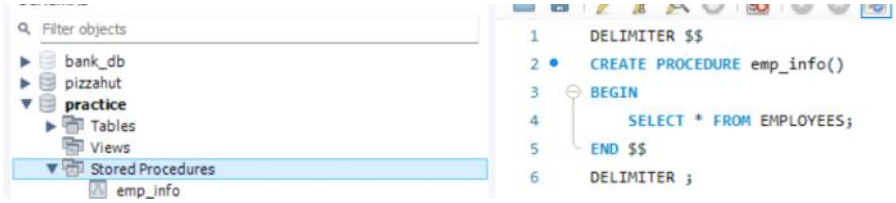
==========

- **WITH ROLLUP : This just adds up all the values and gives the result in a new row.**

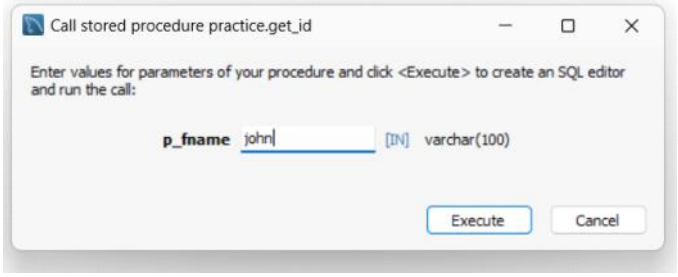
```
mysql> select ifnull(firstname, "Total"), sum(salary) from employees group b
y firstname with rollup;
+---------------------------+-------------+
| ifnull(firstname, "Total") | sum(salary) |
+---------------------------+-------------+
| Daniel                    |       25000 |
| David                     |       25000 |
| Emily                     |       55555 |
| James                     |       23565 |
| Jane                      |       32500 |
| John                      |       45100 |
| Michael                   |       45190 |
| Olivia                    |       36512 |
| Sarah                     |       52463 |
| Total                     |      340885 |
+---------------------------+-------------+
```

- **STORED ROUTINE : An SQL statement or a set of SQL statement that can be stored on database server which can be called no of times.**
- **There are two types of STORED Routine**
  - **STORED Procedure: These are routines that contain a series of SQL statements and procedural logic.**
    - **Often used for performing actions like data modification, transaction control, and executing sequences of statements.**
    - **The syntax goes as follows. For visual representation I'm using workbench.**
    - 

  - **As of now there are no stored procedures now let me create one.**

- 

  - **When I click on it I will easily get the employees data and this would come in handy as I don't need to write that query every time.**

- **Passing arguments to the stored procedure :**
  - **We can also create certain procedures on a particular basis. For ex if we enter the name we need to get their id.**
  - ```
    DELIMITER $$
    CREATE PROCEDURE get_id(IN p_fname varchar(100))
    BEGIN
        SELECT emp_id FROM EMPLOYEES
        where firstname = p_fname;
    END $$
    DELIMITER ;
    ```
  - **So when we execute this we are asked for an input name like this:**
  - 

  - **After entering the name we will get the id of that employee.**
  - 

  - **User defined Functions:**

- ```
  DELIMITER $$

  CREATE FUNCTION get_sum(p_fname VARCHAR(50)) RETURNS VARCHAR(50)
  DETERMINISTIC NO SQL READS SQL DATA

  BEGIN
      DECLARE v_max INT;
      DECLARE v_name VARCHAR(50);

      Select MAX(salary) INTO v_max from employees;
      Select fname into v_name from employees
          where salary=v_max;

      return v_name;
  END$$

  DELIMITER ;
  ```

- **The query goes like this**

```
DELIMITER $$
CREATE FUNCTION get_max_salary_empname() RETURNS VARCHAR(100)
DETERMINISTIC NO SQL READS SQL DATA
BEGIN
    DECLARE v_max INT;
    DECLARE v_emp_name VARCHAR(100);


    SELECT MAX(salary) INTO v_max from employees;
    SELECT firstname into v_emp_name from employees
    where salary = v_max;

    return v_emp_name;
END $$
DELIMITER ;
```

- **Output :**

practice.get_max_salary_empname()
Emily

\

==========

- **WINDOWS FUNCTIONS : Window functions also known as analytic functions allow you to perform calculations across a set of rows related to the current row. Defined by an OVER() clause**
- **Over() function is used just to sort the data like we need here are some of its usecases.**
- **The below query provides the max salary for each city**

```
mysql> select emp_id, firstname,city, salary, max(salary) over(partition by
city) as max_sal from employees;
+--------+-----------+--------------+--------+---------+
| emp_id | firstname | city         | salary | max_sal |
+--------+-----------+--------------+--------+---------+
|      2 | Jane      | Chicago      |  32500 |   45190 |
|      3 | Michael   | Chicago      |  45190 |   45190 |
|      7 | James     | Chicago      |  23565 |   45190 |
|      1 | John      | Dallas       |  45100 |   45100 |
|      5 | David     | Dallas       |  25000 |   45100 |
|      9 | Daniel    | Dallas       |  25000 |   45100 |
|      4 | Emily     | Houston      |  55555 |   55555 |
|      6 | Sarah     | Philadelphia |  52463 |   52463 |
|      8 | Olivia    | San Diego    |  36512 |   36512 |
+--------+-----------+--------------+--------+---------+
```

- **This provides the total salary given in each city.**

```
mysql> select emp_id, firstname,city, salary, sum(salary) over(order by city
) as total_sal from employees;
+--------+-----------+--------------+--------+-----------+
| emp_id | firstname | city         | salary | total_sal |
+--------+-----------+--------------+--------+-----------+
|      2 | Jane      | Chicago      |  32500 |    101255 |
|      3 | Michael   | Chicago      |  45190 |    101255 |
|      7 | James     | Chicago      |  23565 |    101255 |
|      1 | John      | Dallas       |  45100 |    196355 |
|      5 | David     | Dallas       |  25000 |    196355 |
|      9 | Daniel    | Dallas       |  25000 |    196355 |
|      4 | Emily     | Houston      |  55555 |    251910 |
|      6 | Sarah     | Philadelphia |  52463 |    304373 |
|      8 | Olivia    | San Diego    |  36512 |    340885 |
+--------+-----------+--------------+--------+-----------+
```

- **More functions :**

- 

- **ROW NUMBER :**

```
mysql> select ROW_NUMBER() OVER() AS ROW_NO, emp_id,city, salary from employ
ees;
+--------+--------+--------------+--------+
| ROW_NO | emp_id | city         | salary |
+--------+--------+--------------+--------+
|      1 |      1 | Dallas       |  45100 |
|      2 |      2 | Chicago      |  32500 |
|      3 |      3 | Chicago      |  45190 |
|      4 |      4 | Houston      |  55555 |
|      5 |      5 | Dallas       |  25000 |
|      6 |      6 | Philadelphia |  52463 |
|      7 |      7 | Chicago      |  23565 |
|      8 |      8 | San Diego    |  36512 |
|      9 |      9 | Dallas       |  25000 |
+--------+--------+--------------+--------+
```

- **RANK : This provides the rank based on the values. Remember it will give the ranks from smallest to largest like below.**

```
mysql> SELECT EMP_ID, FIRSTNAME, SALARY,
    -> RANK() OVER(ORDER BY SALARY) AS RANK_SAL FROM EMPLOYEES;
+--------+-----------+--------+----------+
| EMP_ID | FIRSTNAME | SALARY | RANK_SAL |
+--------+-----------+--------+----------+
|      7 | James     |  23565 |        1 |
|      5 | David     |  25000 |        2 |
|      9 | Daniel    |  25000 |        2 |
|      2 | Jane      |  32500 |        4 |
|      8 | Olivia    |  36512 |        5 |
|      1 | John      |  45100 |        6 |
|      3 | Michael   |  45190 |        7 |
|      6 | Sarah     |  52463 |        8 |
|      4 | Emily     |  55555 |        9 |
+--------+-----------+--------+----------+
```

- **So just use desc you are all set**

```
mysql> SELECT EMP_ID, FIRSTNAME, SALARY,
    -> RANK() OVER(ORDER BY SALARY desc ) AS RANK_SAL FROM EMPLOYEES;
+--------+-----------+--------+----------+
| EMP_ID | FIRSTNAME | SALARY | RANK_SAL |
+--------+-----------+--------+----------+
|      4 | Emily     |  55555 |        1 |
|      6 | Sarah     |  52463 |        2 |
|      3 | Michael   |  45190 |        3 |
|      1 | John      |  45100 |        4 |
|      8 | Olivia    |  36512 |        5 |
|      2 | Jane      |  32500 |        6 |
|      5 | David     |  25000 |        7 |
|      9 | Daniel    |  25000 |        7 |
|      7 | James     |  23565 |        9 |
+--------+-----------+--------+----------+
```

- **If you observe the above ranks closely you will see that 8 th rank is missing that is because we have two 7 ranks to eradicate this we use dense rank**
- **DENSE RANK:**

```
mysql> SELECT EMP_ID, FIRSTNAME, SALARY,
    -> DENSE_RANK() OVER(ORDER BY SALARY desc ) AS RANK_SAL FROM EMPLOYEES;
+--------+-----------+--------+----------+
| EMP_ID | FIRSTNAME | SALARY | RANK_SAL |
+--------+-----------+--------+----------+
|      4 | Emily     |  55555 |        1 |
|      6 | Sarah     |  52463 |        2 |
|      3 | Michael   |  45190 |        3 |
|      1 | John      |  45100 |        4 |
|      8 | Olivia    |  36512 |        5 |
|      2 | Jane      |  32500 |        6 |
|      5 | David     |  25000 |        7 |
|      9 | Daniel    |  25000 |        7 |
|      7 | James     |  23565 |        8 |
+--------+-----------+--------+----------+
```