MySQL

- SQL is the most Widely used commercial relational database language:-

- SQL statements:-

1. Data retrieval :- SELECT.

2. Data manipulation language (DML):- The SQL DML provides commands to query information to insert tuples into, delete tuples from, and modify tuples.
   INSERT, UPDATE, DELETE, MERGE.

3. Data definition language (DDL):- The SQL DDL provides commands to delete relations, and modify relations.
   CREATE, ALTER, DROP, RENAME, TRUNCATE.

4. Transaction control :- COMMIT, ROLLBACK, SAVEPOINT.

5. Data Control language (DCL):- GRANT, REVOKE.

- SQL query is not case-sensitive. But, it is good to use capital letter for the query.

- Create Database
- Syntax :-

CREATE DATABASE database name ;

- Selecting a database

USE database ; // -- to use particular database
SELECT database ; --

- Deleting a database

DROP DATABASE database ;

- Creating Tables
- A Table is a collection of related data held in a table format within a database

id	name	city
101	Raju	Delhi
102	Sham	Noida

- See different types of datatypes in starting of copy.
- CREATE TABLE students (
id INT,
name VARCHAR(100));
- Checking your table
- DESC table name ;

Adding data into a Table :

```
INSERT INTO Students (id, name, class)
VALUES
(101, "Paul", 5)
```

- If we adding data in the same order it is created then we can ignore column giving part.

```
INSERT INTO Students
VALUES
(102, "RAJ", 8),
(103, "MAN", 10);
```

Reading data from a Table

```
SELECT * FROM <table_Name>
```

- Display all columns of the table.

```
SELECT name FROM <table_Name>
```

- Display only name column

```
SELECT name, id FROM <table_Name>
```

- Display only name & id column.

```
SELECT * FROM <table_Name>
WHERE id = 103;
```

- Display id = 103, student data all columns.

- **Modify / Update data from a Table.**

```
UPDATE <table-name>
SET Contact = 9998
WHERE name = "RAJU";
```

- It will update the contact info where name matches.

- **DELETE data from a Table.**

DELETE, DROP, TURNCATE

- DELETE :-

Delete rows one by one, or delete all rows.

```
DELETE FROM <table-name>
WHERE name = "RAJ";
```

- DROP :-

It removes the whole table and its structure.

```
DROP TABLE <table-Name>;
```

- TURNCATE :-
- It deletes all rows. We cannot delete one row from this as it does not have a WHERE clause. Structure still exists of the table.

```
TURNCATE TABLE <table-Name>;
```

- NOT NULL.

- CREATE TABLE customer1
(
    id INT NOT NULL,
    name VARCHAR(100) NOT NULL
);

- After using "NOTNULL" we are not able to enter the NULL values and we cannot either left it blank.

- Default

CREATE TABLE employee(
    name VARCHAR(100),
    acctype VARCHAR(50) DEFAULT 'Savings'
);

- Primary Key
- The primary key constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only one primary key.

- CREATE TABLE customer1 (
    acc_no INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
);

- AUTO_Increment

- CREATE TABLE Customers
(
acc_no INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100) NOT NULL,
acc_type VARCHAR(50) NOT NULL DEFAULT 'Savings'
);

- ALIAS

- To make it more readable we can change the name by using "AS" keyword.

SELECT acc_no AS "Account No." FROM Customers1;

- WHERE

- Show condition.

SELECT * FROM Students
WHERE name = "RAJU";

- String functions
- CONCAT (first_col, secd_col)

CONCAT (first_word, second_word, ...)

- It will join the strings

SELECT CONCAT ('Hey', 'Bro');

output = HeyBro.

- Concat two column as one.

SELECT emp_id, CONCAT (fname, lname) as FullName From employees;

- We can even insert string in it.

SELECT empid, CONCAT(fname, 'ABe') as NewName From employees;

- CONCAT_WS

- Concat with Separater. It will join columns words with a Separater Symbol.

- We can perform this task by using just CONCAT, but this function will make it much easier.

- SELECT CONCAT_WS(':', emp_id, fname, desig) FROM employees;

  or

  SELECT CONCAT(emp_id, ':', fname, ':', desig)

- Substring

: It will extract the part from a string.

- SELECT SUBSTRING('Hey Brother', '1', '4');
  1234

  Output = Hey

- It excludes the last line.

  SELECT SUBSTR('Hey Brother', '5');

  Output = Brother.

- It will print until the last line.

  SELECT SUBSTR('Hey Brother', '-7');

  Output = Brother.

- It (will

REPLACE

- It will replace or change.
- REPLACE ('Hey Bro', 'Hey', 'Hello');

output ⇒ Hello Bro

- We can even replace column info.

```
SELECT REPLACE (emp_id, 10, 1000) as NewId
FROM employees;
```

(emp_id: Column name; 10: Current value; 1000: Changed value)

It will update this value to every record of that column

- **Reverse and Upper & Lower case**

- Reverse:- Reverse the words.

```
SELECT emp_id REVERSE (fname) AS fname
FROM employees;
```

- Upper case:-

```
SELECT emp_id, UPPE (fname) or
SELECT emp_id, UCASE (fname)
```

- Lower case:-

```
SELECT emp_id, LOWER (fname) or
SELECT emp_id, LCASE (fname).
```

- CHAR_LENGTH
- SELECT CHAR_LENGTH("Hello World");
- return the length of the word.
- Other functions
- To insert in the middle of string.
- SELECT INSERT('Hey Wassup', 5, 0, 'Raju');

output:- Hey Raju Wassup.

String — Position upto shifting of second word — Word to be inserted.

- SELECT LEFT('Abcdefghi', 3)

output = Abc

- SELECT RIGHT('Abcdefghi', 3)

output = ghi

- SELECT REPEAT('Haha', 10);

Word — Repetition.

- TRIM :- Remove the extra spaces

SELECT TRIM(' Hello ');

output => Hello.

DISTINCT

- It returns unique values from a column.
  DISTINCT
- SELECT dept FROM employees;
- It also uses with combination of colums with unique val

ORDER BY

- Used to arrange it in alphabetical order or any order
- SELECT * FROM Employees
  ORDER By fname;
- Reverse Sorting.
  SELECT * FROM employees
  ORDER By fname DESC;
- We can use column number.
  SELECT * FROM employees
  ORDER By 4;
  ↓
  column number
- We can use two colums too.
  SELECT dept, fname FROM employees
  ORDER By dept, fname

* It first Sort it according to "dept" then if we have multiple entries from one dept then it Sort that part.

- LIKE
- Search Word.
- SELECT * FROM employees
  WHERE desig LIKE "%cash%";

  desig → column name

  % → starting anything / ending anything

  middle somewhere contain cash (not case sensitive)
- SELECT * FROM employees
  WHERE fname LIKE "____";

  ____ → word length of 4

  returns names whose character length is 4.
- WHERE fname LIKE "R___";

  returns names which starting with R and of length 4.
- WHERE fname LIKE "%g";

  return names whose ending letter is 'g'.

XX MIT

- Adding column to an existing table.

```
ALTER TABLE employees
ADD COLUMN
Salary INT NOT NULL
DEFAULT 45000;
```

- LIMIT

- It gives a range of values from a table.

- SELECT * FROM employees LIMIT 3;

output :- It will show first 3 records.

- SELECT * FROM employees LIMIT 2, 3;

output :- It will show 3 records after number 2 record (primary key).

- SELECT * FROM employees ORDER BY salary DESC LIMIT 3;

output :- It will show top 3 salary person.

COUNT

- It returns the total no. of occurance.
- SELECT COUNT(*) FROM employees;

Output => Returns the total no. of records, like 9, 4, etc.

- SELECT COUNT(DISTINCT Dept) FROM employees;

Output => Returns the total no. of unique departments.

- SELECT COUNT(emp_ID) FROM employees WHERE desig = 'Manager';

Output => Returns the total no. of person having designation manager.

GROUP BY

- It will form groups in the table.
- It does not include duplicate data.

SELECT dept, COUNT(emp_Id) FROM employees GROUP BY dept;

Output :- It will firstly form groups in dept and then show, how many persons are there in that dept.

MIN & MAX.

- MIN returns the lowest value and MAX returns the highest value.

- SELECT MAX(Salary) FROM employees;

output ⇒ returns the value with highest Salary.

- SELECT fname, MAX(Salary) FROM employees;

output ⇒ It returns the first name and the highest Salary value not the person name having highest salary from the table.

- To get that value we need Sub-Query.

- SELECT fname, Salary FROM employees
WHERE
Salary = (SELECT MAX(Salary) FROM employees);

→ Here the query inside the small brackets will execute first and then the above query will execute.

SUM & AVG

- SUM, gives the total sum.
  AVG, gives the average amount.

  SELECT AVG (Salary) FROM employees.

- We can use this SUM & AVG with other query to make it more useful.

Data Types

CHAR vs VARCHAR.
(0-255) (0-65,535)

- CHAR is a fixed-length datatype.
  e.g. ⇒ name CHAR (4);
  If we input only "M" it will still add 3 white-spaces and make it 4 bytes long and then store it. We can't enter more than 4 length
- In case VARCHAR, it is mostly use we size is not so fixed.
  Eg. ⇒ name CHAR (4)
  Now if we enter "M" it will take only 1 byte. We can't enter more than 4 length
- But in case of using all length like name CHAR (4) 'MANO', it will take 4 bytes.
  name VARCHAR(4) 'MANO', it will take 5 bytes
- CHAR is mostly used for fixed length word like country codes (IN etc

Decimal

- Maximum value = 65 including decimals.

DECIMAL(5, 2) → Digits after decimals.
↓ Total digits

e.g. ⇒ 119.12 ✓ (total digits = 5)
28.15 ✓ (4)
1150.15 ✗ (6) Not allowed if DECIMAL(5, 2).
15.878 (It will not give an error but give warning and also stores upto two decimal places.

Float & Double.

Float :- upto 7 digits, takes 4 bytes of memory.

Double :- upto 15 digits, takes 8 bytes of memory.

float display upto = 123.456.

Double display upto = 123.456789009884

- Date, Time, Datetime
- Date format : YYYY-mm-dd,
- Time format : HH:MM:SS.
- Date time : YYYY-mm-dd HH:MM:SS.
- CREATE TABLE person
  (d DATE, t TIME, dt DATETIME);

  INSERT INTO person
  VALUES
  ('2002-04-17', '23:00:00', '2002-05-16 22:05:02');
- We can use ('2002:04:17') but it will still stores in '2002-04-17' order and gives warning.
- DATE TIME Function

  CURDATE, CURTIME, NOW
- SELECT CURDATE();

  Output :- Return current date.
- INSERT INTO person values (CURDATE(), CURTIME(), NOW());

- DAYNAME, DAYOFMONTH, DAYOFWEEK.
- SELECT DAYNAME('2007-02-03');

  output :- Saturday.
- SELECT DAYOFMONTH('2007-02-03');

  output :- 3
- SELECT DAYOFWEEK('2007-02-03');

  output :- 7. (as counting beigns from Sunday - 1, Monday - 2)
- Date Formatting.
- DATE_FORMAT(NOW(), '%D %a at %T')

  %D → Date, %a → Day, %T → Time

  output :- 21st Tue at 21:20:28

  DATE_FORMAT(NOW(), '%m/%d/%y')

  output :- 04/16/23.

- DATE MATH

- DATEDIFF (expr1, expr2)

→ returns two date difference.

- SELECT DATEDIFF('2023-04-20', '2023-03-15');

output = 36 days.
Always comes in days

- DATE_ADD (date, Interval expr unit).

- SELECT DATE_ADD('2023-04-21', INTERVAL 1 month);

output :- 2023-05-21.

- DATE_SUB ('2023-04-21', INTERVAL 1 month);

output :- 2023-03-21.

- TIMEDIFF (expr1, expr2)

TIMEDIFF ('20:00:00', '18:00:00')

DEFULT & ON UPDATE TIMESTAMP

- Here two new function is used when we create something and then we updated that thing like in an eg. of an blog.

```
CREATE TABLE blogs (
blog VARCHAR (150),
ct DATETIME DEFAULT CURRENT_TIMESTAMP,
ut DATETIME ON UPDATE CURRENT_TIMESTAMP
);
```

```
INSERT INTO blogs (blog)
VALUES ('This is my blog');
```

output:-

blog	ct	ut
This is my blog	2023-04-21 22:28:02	NULL

```
UPDATE blogs
SET blog = 'This is my 2nd blog';
```

output:-

blog	ct	ut
This is my 2nd blog	2023-04-21 22:28:02	2023-04-21 22:30:02

- **Operators**
- **Relational Operators:-**

<	:	less than
>	:	Greater than
<=	:	Less than or equal to
>=	:	Greater than or equal to
=	:	Equal to
!=	:	Not equal to.

- SELECT * FROM employees
  WHERE Salary < 65000;
- **Logical operators:-**
- AND :- Condition 1 AND Condition 2

  When both the conditions are true.

SELECT * FROM employees
WHERE Salary = 45000 AND dept = 'IT';

- OR :- condition1 OR Condition 2

  When either of the condition is true.

IN, NOT IN

- "IN" is use to find that particular record.

```
SELECT * FROM employees
WHERE dept = 'Account'
OR dept = 'Cash'
OR dept = 'Loan';
```

or

```
SELECT * FROM employees
WHERE dept IN ('Account', 'Cash', 'Loan');
```

output :- Show records where dept matched with either of one department.

- "NOT IN" is use to do just reverse of IN.

```
SELECT * FROM employees
WHERE dept NOT IN ('Account', 'Cash', 'Loan');
```

output :- Show records of every dept. except the given dept.

- **Between**
- Returns the between range.

```
SELECT * FROM employees
WHERE
Salary BETWEEN 65000 AND 75000;
```

- **CASE**
- Used to determine condition.

```
SELECT fname, salary,
CASE
WHEN salary >=40000 THEN 'High salary'
ELSE 'Low salary'
END
AS 'salary category'
FROM employees;
```

output:

fname	Salary	Salary category
RAJU	37000	Low salary
SHAM	45000	High salary

- We can use multiple cases.

```
CASE
WHEN salary >= 50000 THEN 'High Salary'
WHEN salary >= 40000 AND salary < 50000
THEN 'Mid salary'
ELSE 'LOW salary'
AS 'salary category'
FROM employees;
```

IS NULL & NOT LIKE

- To check NULL values in column. It returns the NULL values

- SELECT * FROM person
  WHERE jt IS NULL;

  output ⇒ It will show all records where jt column contains NULL values.

- NOT LIKE, just opposite of LIKE.

  SELECT * FROM employees
  WHERE fname NOT LIKE 'A%.';

  output ⇒ Returns values where starting letter is not A of column fname.

- UNIQUE constraints
- Used to keep unique values.
- CREATE TABLE Contact (mob VARCHAR (15) UNIQUE);
- CHECK constraints
- To keep an check to the values entered in the columns.
- CREATE TABLE Contact (
mob VARCHAR(15) CHECK(LENGTH(mob) >=10)
- Now it will generate an error if we try to enter value less than 10.
- To specify the exact error like "mobile no. is less than 10 digits" then we can use CONSTRAINT keyword.

```
mob VARCHAR (15) UNIQUE,
CONSTRAINT mobile_no._less_than_10_digits
CHECK (LENGTH (mob) >=10);
```

Altering Tables

- We can modify our table structure.
- To add column into table. (by default mySQL put NULL values to newly added column).

```
ALTER TABLE contact
ADD COLUMN name VARCHAR(50);
```

- To remove column.

```
ALTER TABLE Contact
DROP COLUMN city;
```

- To rename a column.

```
ALTER TABLE Contacts
RENAME COLUMN name To Full_name;
```

- To rename a table.

```
ALTER TABLE Contacts
RENAME TABLE contacts To my_contacts;
```

- Modify the ~~Column~~ data types

- ALTER TABLE mycontact MODIFY mob VARCHAR(20) DEFAULT 'unknown';

- To change name and property both at the same time.

ALTER TABLE contacts CHANGE name emp_name VARCHAR (50) NOT NULL;

Here we are changing the "name" to "emp_name" and add property "NOT NULL".

- Add & Drop constraints

- To add or drop constraints to an existing table.

- To ~~add~~ add or drop constraints we first need to find the constraints

SELECT CONSTRAINT_TYPE, CONSTRAINT_NAME FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS WHERE TABLE_NAME = 'contact2';

↓

Table name.

- For adding constraints.

```
ALTER TABLE mycontact
ADD CONSTRAINT mob_digits_check
CHECK (LENGTH (mob) >=10);
```

- Relationship

1:1 , 1:MANY , MANY:MANY

- Foreign key
- CREATE TABLE orders (

```
    -
    -
    -
FOREIGN KEY (cust_id) REFERENCES
customers (cust_id) );
```

- JOIN

- Join operation is used to combine rows from two or more tables based on a related column between them.

- Types of join

Cross Join.
Inner Join.
Left Join.
Right Join.

- **Cross Join**

Every row from one table is combined with every row from another table.

```
SELECT * FROM customers, orders;
```

- output :- It will show all datas of both the tables. It is very confusing that's why we do not use this method.

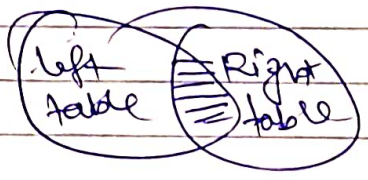
- **Inner Join**

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

- ```
  SELECT * FROM customers
  INNER JOIN orders
  ON orders.cust_id = customers.cust_id;
  ```

- Using Group by function and also showing total amount spend by a particular person.

```
SELECT name, SUM(amount) FROM
customers INNER JOIN orders
ON orders.cust_id = customers.cust_id
Group By name.
```



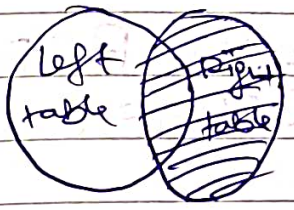
Note:- In MYSQL INNER JOIN is the default join so we can use simply JOIN.
```

LEFT JOIN

- Returns all rows from the left (or first) table and the matching rows from the right (or second) table.

```
SELECT * FROM customers
LEFT JOIN
orders
ON orders.cust_ID = customers.cust_ID.
```

Right JOIN



- Returns all rows from the right (or second) table and the matching rows from the left (or first) table.

```
SELECT * FROM customers
RIGHT JOIN
orders
ON orders.cust_ID = customers.cust_ID.
```

ON DELETE CASCADE

- When Join tables one is customer and other one is order. : Then if we try to delete any records from customer table by using normal delete operation then it won't allow to do this as that record is linked to order table.

- If we want to delete this, then we have to use "ON DELETE CASCADE." in the order table in the foreign key.

```
CREATE TABLE orders (
    - - -
    ^ ——
    - ——
    FOREIGN KEY (Cust_id) REFERENCES
    customer(Cust_id) ON DELETE CASCADE.
```

- ## VIEW & Virtual table

- We can create an virtual table to stored the relationship data.

CREATE VIEW inst_info AS.

- ## Having clause

- If we are using Group By function in the query and wants to add an condition then we cannot use "WHERE" clause. we have to use the "Having" clause. It is same as WHERE.

- "WITH ROLLUP" we use this to add an row in the last which shows the total.