SQL (Structured Query language):- (Sequel)

Let's START with SQL:-)

Database:- Collection of Data its called as database.

DBMS:- A software application to manage our data.

User → Software Application →

Database.

Database

Relational (Use tables to store data)

My SQL
Oracle
Maria DB

Non-Relational (Da is not stored im tables)

Mongo DB

Why SQL?

We need a language to interact with databases.

So we use SQL to interact with DB, do some CRUD operations on the data.

Them What iis MySQL:-

My SQL iis specific Relational DBMS that uses SQL as its quering language

History of SQL:-

SQL originated in the 1970s from IBM's research on RDB. It started as SEQUEL, later SQL due to trade-mark issues.

SQL:-

SQL is a programming language that is used to communicate and manipulate data iin databases.

It helps us to peroform CRUD (Create, Read, Update, Delete) operations in DB.

How SQl helps us?

SQL allows users to perform a variety of tasks related to data bases.

- Retrieving data - Extracting precise information from a database through database.
- Manipulating Data - Adding, Modifyin, or removing records with a DB.

- Defining data - Creating and adjusting the structure of a database, including tables, views and indexes.
- Controlling Data - Managing database access by granting or revoking permissions

My SQL Server :-
Database server where data is stored, managed and accessed.

My SQL WorkBench :- It is visual tool which is used for database design development, administration and management.
It provides a User interface to interact with MySQL server.

Let's install the server first :-

- Go to the MySQL official website : https://www.mysql.com/.
- Go to downloads.
- Select My SQL community (GPL) downloads at the bottom of the page.
- Choose My SQL community Server, select the version and click on download.
- Follow the instructions and set the

root password. This password would be asked while creating a new connection.

Types of SQL Commands:-

SQL Commands are divided into different categories based on their functionalities.

1. Data Query language (DQL) Commands -
DQL is used to retrieve data from the database.
Commands: Select

2. Data Manipulation language (DML) commands -
DML is used to manipulate data stored in the database.
Commands: Insert, Update, Delete.

3. Data definition language (DDL) Commands
DDL is used to define the structure and schema of the database
Commands:- Create, Alter, Drop, Truncate, Rename.

4. Data Control language (DCL) commands -
DCL deals with the control and security

of data within the database.
Commands : Grant, Revoke.

5. Transaction Control language (TCL) Commands -

TCL is used to manage transactions within a database.
Commands : Commit, Rollback, Save-point.

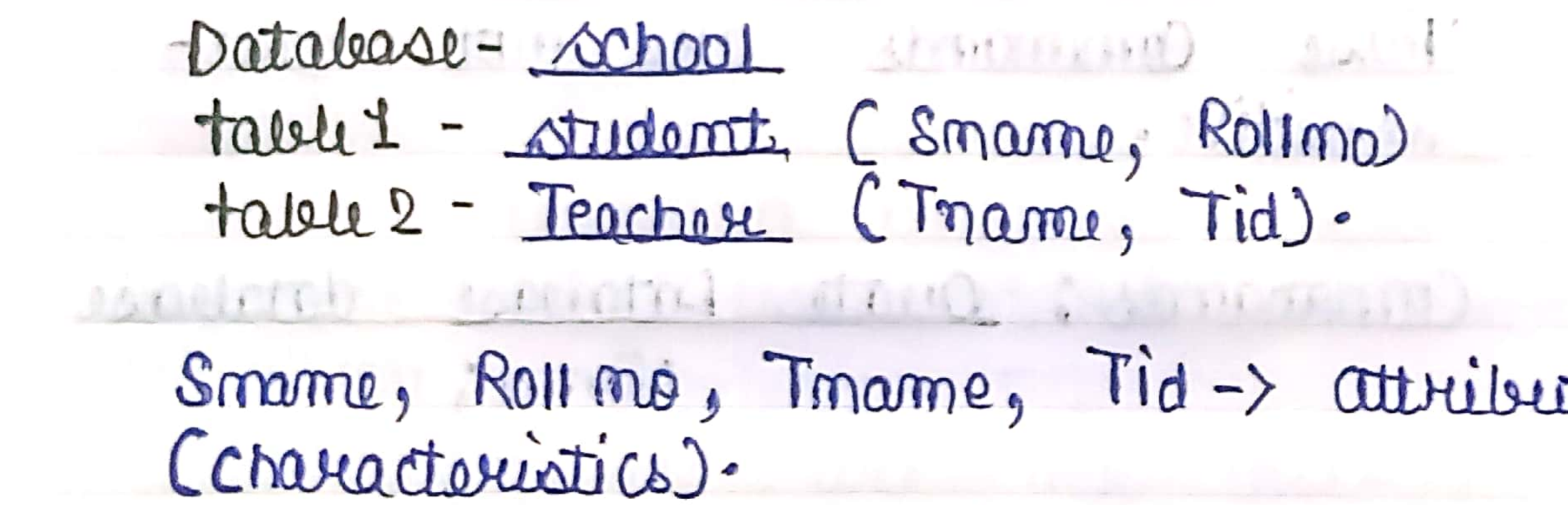
Creation of Database :-

Lets understand database design from an example, Consider a college database.

Database - School
table 1 - student (Sname, Rollno)
table 2 - Teacher (Tname, Tid).

Sname, Rollno, Tname, Tid -> attributes (characteristics).

Creation of Database :-



Creation of Database :-

Steps to create a database :

1. Choose DBMS (Database Management system).
2. Connect to the server using a command-line tool or a GUI.
3. Create a new Database
4. Once the database is created, you can use the USE statement to create tables in the database.
5. Create tables and insert data.

Creating a new database :-

We use the Create database statement to create a new database.
These Commands are not case-sensitive.

Commands : Create Database database Name;

Also to avoid error, we can use:

Commands :- Create database if not exists database Name;

If not exists and if exists clauses are commonly used in conjunction with the Create table and to drop

table statements to avoid errors.

Deleting a database

We use the drop database statement to delete a database.
Dropping a database means deleting the entire database, including all tables, data and other objects within it, Drop is a DDL Command.
These Commands are not case-sensitive.

Command: Drop database databaseName;

Also, to avoid error we can use:

Command: Drop database if Exists databaseName;

Using a Database:-

We use the Use database statement to use a database.
These commands are not case-sensitive.

Command: Use databaseName;

Showing a database:-

We use the Show Databases statement to see all the databases present in a server.

Command: Show Databases;

Table

Creating a table

We use the Create Table statement to create a table in DB.

Command:

Create Table Table Name (
    Column1 Data Type 1 Constraint 1,
    Column2 Data Type 2 Constraint 2,
    Column 3 Data type 3 Constraint 3,
    -- additional columns if needed
);

CREATE - DDL Command.

Eg:-

```
Create Table employee (
    empidINT primary key,
    name Varchar(50),
    salary INT
);
```

employee

empid	name	salary

Inserting values into table:-

INSERT - DML Command

Insert into table name (Column 1, Column 2... Column N) Values (Value 1, Value 2. ... Value n).

Example :-

- Insert into employee (empid, name, salary) values (1, "Ray", 1200), (2, "Rahul", 1100), (3, "Harshit", 1100).

employee

empid	name	salary
1	Ray	1200
2	Rahul	1100
3	Harshit	1100

- Insuct into employee values (1, "Ray", 1200), (2, "Rahul", 1100), (3, "Harshit, 1100).

Seeing table in a database:-

SHOW

Eg:-

Show Tables;

It helps us to see all the tables in a given database.

employee

emplId	name	salary
1	Raj	1200
2	Rahul	1100
3	Harshit	1100

To see the values in the table :-

SELECT :-

Eg :-

To see specific values of a column:

Select emplId from employee;

To see all the values or the entire table.

Select * From employee;

then it will show the entire table like

Let's Create a Database for instagram.

1. Create a database

- CReate database if not exists instagram DB;

2. Use db to Create tables

- Use intagram DB;

3. Create table ~~if not~~ into the db.

- CREATE Table if not exists users(userid int primary key, User Name Varchar(50) email Varchar(100)
);

- Create table if not exists posts (postId INT primary key, userId INT, Caption Varchar(100),
);

4. Insert values in the tables.

- Insert into users (userid, user name, email)
values
(1, "niti", abc@gmail.com),
(1, "ray", xy2@gmail.com),
(1, "harshit", abc2@gmail.com);

- Insert into posts (postid, userid, captions)
values
(101, 561, "light"),
(102, 562, "air"),
(103, 563, "water"),

5. You can now see all the tables in database.

- Use Database instagram dB;
  Show Tables;

6. All the values in a specific tables

- select * from Users;
  select * from posts;

Data types in SQL:-

Data types are used to specify the type of data that column can store.

Numeric	Character/String	Date & Time
• Integer/INT.	• Char(n)	• Date
• SmallINT.	• Varchar(n)	• Time
• Big INT.	• Text	• Datetime
• Decimal.		• Timestamp
• Float.		
• Double.		

Boolean	Binary
• Boolean	• Binary(n)
	• Varbinary(n)
	• BLOB

Data types

Numeric data types

1. **INT** - Use for storing whole no's without decimal points (-2,147, 483, 648 to 2,147, 483, 647) (Signed integer).

2. **BIG INT** - Use for storing large whole numbers. (-9, 223, 372,036, 854, 775, 808. to 9, 223, 372, 036, 854, 775, 807).

3. **Float** - Used for storing decimal no's (4-byte).

4. **Double** - Used for storing decimal no's (8-byte).

5. **Decimal (p,s)** - Used for exact numeric representation. p is the precision & s is the scale.

Command:

Create Table example (id INT);

By default all the numeric data types can have negative as well as positive values. This restrict the range so if we know there is only positive value stored we use Unsigned attribute (0-255).

For eg:- Salary can never be in '-' ive or age.

Command :-

Create Table example (C
id INT unsigned
);

Character datatypes :-

1. Char (n) :- Fixed- length character strings can be stored. (0-255)
2. Varchar (n) :- Variable - length character strings can be stored. (0-255)
3. Text :- Variable - length character string with no specified limit.

Command :-

Create Table example (C
name Varchar (50)
);

Boolean data types :-

1. Boolean - Used to store a true or false value.

Command :-

Create Table example (C
is Active Boolean
);

Binary Data types :-

1. Binary (n) - Used to fixed - length binary data.
2. Varbinary(n) :- Used to storing Variable - length binary data.
3. BLOB (Binary Large Object) :- Used for storing large amounts of binary data (var len).

Command :

```
Create Table document (
data BLOB
);
```

Constraints in SQL :-

Constraints - Constraints define rules or conditions that must be satisfied by the data in the table.

Common constraints include uniqueness, nullability, default values etc.

- Unique Constraint : Ensures values in a column are unique across the table.
- Not Null Constraint :- Ensures a Column cannot have a null value.
- Check Constraint : Enforces a conditi

to be true for each row.

- Default Constraint : Provides a default value for a column if no value is specified.
- Primary key : Enforces the unique ness of values in one or more columns.
- Foreign Key : Enforces a link between two tables by referencing a column in one table that is a primary key in another key.

Constraints in SQL:-

- Unique Constraint:

```
Create Table example1 (
phoneNbr INT Unique);
```

- Not Null Constraint:

```
Create Table example1 (
address Varchar (50) Not Null);
```

- Check constraint:

```
Create Table example1 (
age INT check (age>=18));
```

- Default Constraint :-

Create Table example 1 (
enrolled Varchar (20) Default 'no');

- Primary Key constraint :

Create Table employee (id INT Primary Key, name Varchar (255));

Or

Create Table employee (
id INT,
name Varchar (255)
Primary Key (id)
);

- Foreign key constraint :

Create Table orders (
order item No INT primary Key,
CustId INT,
Foreign Key (CustId) Refrences Customer
(CustId));

- Keys in SQL :-

- Primary Key - A primary Key is a unique identifier for each record in the table. It ensures that each row can be uniquely identified and accessed within the table.

- Foreign Key - A foreign key is a field in a table that refers to the primary key of another table. It establishes relationship between Tables.

Primary Key :- A primary key is a key which uniquely identifies each record in a table. It ensures that each tuple or record can be uniquely identified within the table.
It is always Unique + Not Null.

ID	Name	Hometown
123	Rahul	Kolkata
245	Raj	Kolkata
434	Riti	Delhi

Foreign Key :- A Foreign key is a field in a table that refers to the primary key in another table. It establishes a relationship between two tables.

Student
(Base / Referenced table)

Roll no.	Name	Hometown
1.	Rahul	Kolkata
2.	Raj	Kolkata
3.	Riti	Delhi

Subject
(Referencing table)

Roll no.	Name	Subject
1.	Rahul	Maths
2.	Raj	SST
3.	Riti	Science

Referenced table - Table having primary key (pk).

Referencing table - Table having foreign key (fk).

Student
(Referenced table)

Roll no.	Name.	Hometown
1	Rahul	Kolkata
2	Raj	Kolkata
3	Riti	Delhi

↓
(Pk)

Subject
(Refuencing table)

Roll no.	SubjectID	Sub
1	S1	Math
2	S2	SST
3	S3	EVS

↓
(Fk)

Foreign Key

id	Name	C_ID
1	Rahul	100
2	Ray	101
3	Riti	102

Student
(Referenced table)

C_ID	Name	tutor	id
100	Hindi	Ram	1
101	Maths	Mohan	2
102	Eng	Priya	3

Course
(Refuencing table)

Foreign Key :-
Foreign Key helps to perform operation related to the parent table, such as joining table or ensuring referential integrity.

Query :-
Create table childtable Name (
Child Id INT primary Key,
baseId INT,
Foreign Key (baseid) References base table
Name (baseId)
);

Cascading in foreign Key :-
Cascading are a set of rules which dictate what actions should be taken automatically when a referenced row in the parent table is modified or deleted.

1. Cascade- If a row in the parent table is updated or deleted, all related rows in the child table will be automatically updated or deleted.

2. Set Null :- If a row in the parent table is updated or deleted, all corres-

pending foreign key values in the child table will be set to Null.

3. Restrict or No Action - Blocks the modification or deletion of a referenced row in the parent table if related rows exist in the child table, thus maintaining referential integrity

These Cascading Actions help maintain the integrity of the data across related tables in the database

1. ON DELETE Cascade.
2. ON UPDATE Cascade.

- ON DELETE Cascade - The ON Delete Cascade clause indicates that if a row in the parent table (parent_table) is deleted, all corresponding rows in the child table (Child_table) will be automatically deleted as well.

Query:-

Create Table Childtable Name (
ChildId INT primary Key,
baseId INT,
Foreign Key (baseId) References base Table
name (baseId)

ON Delete Cascade
);

2. ON UPDATE Cascade :- The ON UPDATE Cascade clause indicates that if a row in the parent table (parent_table) is updated, all corresponding rows in the child table (Child_table) will be automatically updated as well.

Query :-
Create Table Child table Name (
ChildId INT primary Key,
baseId INT,
Foreign Key (baseId) References parent table Name (ChildId)
ON update Cascade
);

Let's Make a database for a Company xyz :-

- Make a database for a company xyz.

Create Database xyz

- Make an employee table in the xyz database.

Create Table employee (
id INT Primary Key,

name Varchar(50),
age INT,
department Varchar(50)
City Varchar(50),
salary INT);

Retrieving data from Table :-

3. Fill details in the table -

Insert into employee (id, name, age, department, city, salary)
Values
(1, "Rahul", 25, "IT", "Mumbai", 1500)
(2, "afsana", 26, "HR", "Pune", 2000)

4. See all the the data in the data table.

UPDATE Command -

The UPDATE Command in SQL is used to modify existing records in a table. If you get a safe mode error while executing queries turn this query.

Query - set SQL_safe_update=0;

Query :-
UPDATE table_name.
SET Columns Name 1 = Value 1 (to be set),

Column Name 2 = Value 2 (to be set)
Where Condition;

UPDATE Command (Practice Question)

1. Write a Query to update the salary for all employee in the 'HR' department to 50000.

Query:-
Update employee
Set salary = 50000
Where department = "HR";

2. Write a query to update the name of an employee from Raaj to Raj.

Query:-
Update employe
Set name = "Raj"
Where name = "raaj"

DELETE Command

The DELETE Command in SQL is used to remove records from a table.

Query:-
Delete from table_name
Where condition;

1. Write a query to delete all records from the employee table where department is 'HR'.

Query :-

```
Delete from employee
Where department = "HR";
```

2. Write a query to Delete the record of an employee having name as Raj -

Query :-

```
Delete from employee
Where name = "Raj";
```

Retrieving data from table :-

Select command - Select is a DQL (Data Query Language) command. It is used to retrieve data from a database.

We can provide specific columns from which we can retrive data.

select Column1, column2 from table Name; → to retrieve data present in specific column in a table.

select * from table Name; → to retrieve all the data present in table.

Where clause - It filters the rows based on specified conditions.

Query: Select Col1 Col2 from tableName
where condition;

ex:- Select * from employee where age>20;

SQL Commands

DQL	DML	DDL
Select	Insert Update Delete	Create Alter Drop Truncate Rename

Alter Command -

Alter is a DDL command used to modify (change) existing database objects, such as tables, indexes or constraints (schema).

Let's see all the things Alter can help us to do. So Mostly it is used to modify the schema, so we will mostly see how it can help in Modification of columns like - addition of new column, deletion of column,

modification of column and much more.

- Add a column -

Query:
Alter Table table name
Add column Name datatype constraint;

- Drop a column:-
Query:
Alter Table table Name
Drop column column Name;

- Modify the data type of an existing column:-

Modify Clause - The Modify clause is oftenly used within an Alter table statement in SQL. It allows us to change the definition or properties of an existing column in a table.

Query:
Alter table tableName
Modify column new datatype;

The above command modifies column Name to a new data type.

- Change the name of an existing Column :-

Change - The Change command is oftenly used within an Alter Table statement in SQL. It helps to change the name or data type of a column within a table -

Query:

Alter Table tableName
Change oldcolumnName newcolumnName
newdatatype;

The Above Command changes the old column Name & also its datatype.

- Rename the name of an existing Column :-

Rename Command : Rename Command is used to change the name of an existing database object, such as a table, column, index, or constraint

Query :-

Alter Table tableName
Rename Column Old Column Name to
new Column Name;

The Above command renames the old column N th to new column Name.

Rename Command:-
This Command is used to change the name of an existing database object, such as a table, column, index or constraint.

Query:
Rename Table old table Name to new table Name;
The above command renames the old table Name to new table Name.

Column Renaming:-
Alter Table table name
Rename Column old column name to new column name;

Database Renaming:-
Rename Database old database name to new database name;

Truncate Command:-
this command removes all rows from the given table, leaving the table empty but preserving its structure.

Query:-
Truncate Table tableName;

Diff between Truncate, Delete & Drop:

Truncate	Delete	Drop
• Remove all rows from a table. • Truncate Table tablename;	• Used to remove specific rows from a table based on a condition. • Delete from tablename where condition;	• Used to completely remove table. • Drop table TABLEname;

- Using Distinct to retrieve Unique Values :-

Distinct - Distinct keywords is used within the select statement to retrieve unique values from a column or combination of columns.

Query:-
Select Distinct Col1 from tableName;
→ Retrieve a list of unique values of Col1.

Select Distinct Col1, Col2 from tableName;
→ Return unique combo of Col1 & Col2.

Operators in SQL :-

To perform operations on data in SQL we use operators.

Query :- Select Col1 Col2 from table Name where condition (use operator);

Types :-

- Arithmetic operators - Addition (+), Subtraction (-), Multiplication (*), division (/), Modulus (%).

Query -
select * from employee where Age+1 = 60;

- Comparision operators - equal to (=), not equal to (<> or !=), greater than (>), less than (<), greater than or equal to (>=), less than or equal to (<=)

Query :-
select * from employe where age > 20;

- Logical operators :-

1. AND: It combines two conditions and return true if both are true.

Query :- select * from employee where city = 'Pune' AND age > 18;

2. OR: It combines two conditions and returns true if either is true.
Query: Select * from employee Where city = 'Pune' OR age > 18;

3. NOT: It reverses the result of a condition, returns true if the condition is false.
Query: Select * from employe Where department NOT IN ('IT', 'HR');

- In Operator: IN (Check if a value matches in a list of values)
Query: Select * From employe Where department IN ('IT', 'HR');

- Is Null / is Not Null Operators: Is Null (Checks for null values), is not Null (Checks for not null values).

Query: Select * from employee Where department is Not Null;

- Bitwise operators:
AND (&), OR (|).

- Like & Wildcard Operators:
Like operator is used to search for

a specified pattern in a column. It uses wild card operators for matching patterns.

1. (%) percent sign - It Matches for any sequence of zero or more characters.

Query: select * from employe where name Like 'A%';

2. _ (underscore):- It matches for any single character.

Query: select * from employe where name Like '_A%';

3. Between Operator - Checks if a value is within a range of values.

Query: select * from Where salary Between 1200 & 1500;

Clauses in SQL:-

Clauses are like tools | conditions that helps us to make queries more specific or decide what data to fetch

Ex:- Where, Group by, Having, Order by, Limit.

Query:- Select Col1, Col2
From tableName
Clause Condition;

Where Clause - It filters the rows based on specified conditions.

Query:- Select Col1, Col2
From tableName
Where Condition;

Eg:- Select * from employee where Age>20;

Limit Clause

The Limit clause in SQL is used to restrict the number of rows returned by a query.

Query:-

Select Col1, Col2 from tableName
Limit No of Rows

Eg:- Select * from employee Limit 2;

Sorting data with the Order by clause.

Order by - It is used to sort the results in ascending or descending order. By default it returns the result in ascending order.

Query:-
Select col1, col2 from tableName
Order by col1 (ASC / DESC), col2 (ASC/DESC)

Eg:- select * from employee order by salary DESC;

⊙ Write a SQL Query to fetch the details of employees having id as 1 -

Query:-
Select * from employee
where id=1;

⊙ Write a SQL Query to fetch the details of employees having id as 1 and city as Mumbai.

Query:-
Select * from employee
where id=1 and city = "Mumbai";

⑤ Write a SQL Query to fetch the details of employees having salary greater than 1200 & city as Mumbai.

Query:-

```
Select * from employe
Where salary > 1200 and city = "Mumbai";
```

⑥ Write a SQL Query to fetch the details of employees who are not from Mumbai.

Query:-

```
Select * from employees
where city Not in ("Mumbai");
```

⑦ Write a SQL Query to fetch the details of employees having the Maximum salary.

Query:-

```
Select * from employe
Order by salary DESC;
```

⑧ Write a SQL Query to fetch the details of 2 employees having the Maximum salary.

Query:-

```
Select * from employee
Order by salary DESC
Limit 2;
```

Aggregate functions :-

Aggregate functions performs some operations on a set of values and then returns a single value summarizing the data. These are used with select statements to perform calculations.

Types :-

- Count ()
- Sum ()
- Avg ()
- Min ()
- Max ()
- Group_CONCAT()

Count () - It counts the number of rows in a table or the number of Non-Null values in a column. This counts how many things are in a list or a group.

Query :- Select Count (name) from employee;
→ this will tell the no. of employees in a company.

Sum () :- It calculates the sum of all values in a numeric column.

This adds up all the numbers in a list.

~~Query :-~~

~~Select * from~~

Select Sum (Salary) from employees ; → this will tell the total amount company is paying to its employees.

Avg() - It computes the avg. of all values in a numeric column. It finds the average, or the "middle" no., of all the numbers in a list.

Query :- Select AVG (salary) from employee ;
→ this will tell the avg amount company is paying to its employees.

Min() - It helps to find the smallest number in a list.

Query : Select Min (salary) from employee; → this will tell the minimum salary company is paying to its employees.

Max() - It finds the Maximum value in a column,

Query : Select MAX (salary) from employee;
→ this will tell the max salary company is paying to its employees.

Grouping data with the Group by Clause :-

Group by clause - This is used to group rows that have the same values into together. It helps to organise data into groups so that you can do calculations, like finding totals or averages, for each group.

This query retrieves the first n rows from the table.

Query :-

```
Select Col1, aggregatefun (Col2)
from table Name
Group by Col1;
```

Eg :-

```
Select department, Avg (salary) As avgsal
    from employee
Group by department;
```

Having Clause

The Having Clause is just like Clause but the main difference is it works on aggregated data. It is used with the Group by Clause. It helps to filter groups based on given con

Conditions · ditions ·

Query :-
Select Col1, Col2 aggregatedfun (col3)
from tableName
Group by Col1 Col2
Having conditions ;

Eg :-
Select department, Avg (Salary) As avgsal
from employee
Group by department
Having avgsal > 1500 ;

Group by and Having clause :-

Three queries demonstrate how to use

(a) Group by to categorize data &

(b) Having to filter group data based on specified conditions in SQL.

Differences :-

Where	Having
• Used to filter rows from the result based on condition applied to	• Used to filter rows from the result based on condition

a row before the aggregation

- It is used with select, update or DELETE commands
- Select * from table Name where condition;

applied to a row after the aggregation

- It is used with Group by and aggregate functions.
- Select col1, col2 aggregate fun(col3) from table Name Group by col1, col2 Having condition;

The general order of SQL commands :-

SNo.	Command	Usecase
1.	Select	Retrieve from database
2.	From	Identify the table
3.	Where	filter rows based on some conditions
4.	Group by	Group rows that have the same values
5.	Having	filter groups based on some conditions
6.	Order by	Sort the result set either asc/desc
7.	Limit	Limit the no. of rows returned.

Joins in SQL:-

Joins are used to combine rows from two or more tables based on a related or shared or common column between them. There are commonly 4 types of Joins including INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, SELF JOIN, CROSS JOIN.

id	Name	Age
1.	Riya	17
2.	Rahul	18
3.	Ram	17

Student

id	Course-id	Course-name
1	101	Eng
2	102	Hin
3	103	Phe

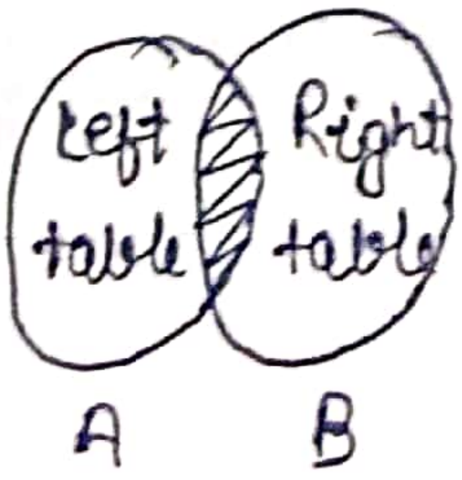
Course

Q. Is foriegn Key important for performing Joins?

→ Joins can be performed based on any columns that establish a relationship between tables, not just fk constraints. So, its not necessary.

Types of Joins in SQL:-

1. Inner Join



Student

rollno	name
1	Ram
2	Rahul
3	Riti

Course

rollno	name
2	Hindi
23	Eng
34	Math

2. Left Join / Left Outer Join :-

Student

Roll no	name
1	Ram
2	Rahul
3	Riti

Course

roll no	C_name
2	Hindi
3	Eng
4	Maths

3. Right Join / Right Outer Join :-

Student

roll no	name
1	Ram
2	Rahul
3	Riti

Course

roll no	C_name
2	Hindi
3	Eng
4	Maths

4. Full Join / Full Outer Join :-

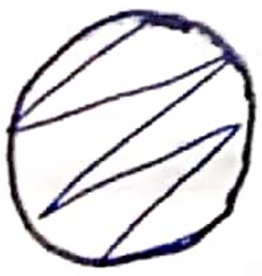
Student

roll no	name
1	Ram
2	Rahul
3	Riti

Course

roll no	C_name
2	Hindi
3	Eng
4	Maths

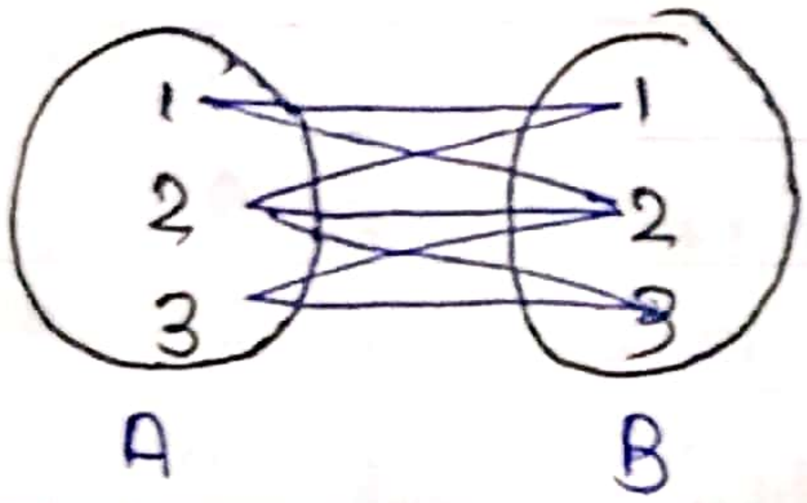
5. Self Join :-



Roll no.	Name
1	Ram
2	Rahul
3	Riti

Students

6. Cross Join :-



Joins in SQL :-

1. Inner Join :- It helps us in getting the rows that have matching values in both tables, according to the given Join condition.

Query :

```
Select columns
from table 1
INNER Join table 2
On table 1. colName = table 2. col Name ;
```

id	Name
101	Ram
102	Rahul
103	Rhi

Customer

id	a name
102	Fruit
103	Ball
104	Utensils

Order

It returns rows where there is a matching id in both sides

```
Select*
from Customer
INNER JOIN Order
ON Customer.id = Order id;
```

id	name
102	Rahul
103	Rhi

id	O_name
102	fruit
108	Ball

2. Left Join - It is used to fetch all the records from the left table along with matched records from the right table.
If there are no matching records in the right table, Null values are returned for the columns of the right table.

Query :-

```
Select columns
from table 1
Left Join table 2
ON table.Col Name = table 2. Col Name.
```

Left table - the table specified before the Left Join Keyword.

Right table - The table specified after the Left Join Keyword.

Query :-
Select *
from Customer
left Join Order
ON Customer.id = Order.id

id	name
101	Ram
102	Rahul
103	Riti

id	O_name
null	null
102	fruit
103	Ball

3. Right Join :- It is used To fetch all the records from the right table along with matched records from the left table.
If there are no matching records in the left table, Null values are returned for the columns of the left table.

Query -
Select Columns
from table1
Right Join table2
ON table1.ColName = table2.ColName;

-

Select *
from Customer
Right Join ~~Join~~ Order
ON Customer.id = Order.id.

id	o_name
102	Fruits
103	Wall
104	Utensils

id	name
102	Rahul
103	Riti
null	null

4. Full Join :- It returns the matching rows of both left and right table and also includes all rows from both tables even if they don't have matching rows.
If there is no match, Null values are returned for the columns of the missing table.
In MySQL, the syntax for a full joint is different compared to other SQL data bases like Postgre SQL.
MySQL doesnot support the full Join Keyword directly. So, we use a combination of Left Join, Right Join & Union to achieve the result.

Query :-

```
Select *
from Customer
Left Join Order
On Customer.id = Order.id;
Union
Select*
from Customer
Right Join Order
```

on Customer.id = order.id.

Result:-

id	name	id	o.name
101	Ram	null	null
102	Rahul	102	fruit
103	Riti	103	Ball
null	null	104	Utensils

5. Cross Join :- It combines each row of the first table with every row of the second table.

Query :-

```
Select *
from table1
Cross Join table2;
```

It results in a new table where the no. of rows is equal to the product of the no. of rows is equal to the product of the no. of rows in each table. (m*n).

Result :-

id	name		id	o_name
101	Ram		1	fruit
101	Ram		2	Ball
102	Rahul		1	fruit
103	Rahul		2	Ball

6. Self Join - A self Join in SQL is a type of Join where a table is joined with itself. It is a type of inner Join.

Query -

```
Select columns
from table as t1
Join table as t2
On t1.colName = t2.colName
```

t1 and t2 are aliases for the tables used to distinguish between the order rows.

S_id	name	mentor_id
1	Ram	null
2	Rahul	1
3	Riti	1
4	Riya	3

Ram (Mentor)
↙ ↘
Rahul (Mentor) Riti (Mentor)
↓
Riya (student)

```
Select S1.name as mentor_name, S2.name
as name
from student as S1
Join student as S2
where S1.s_id = S2.mentor_id
```

Result:-

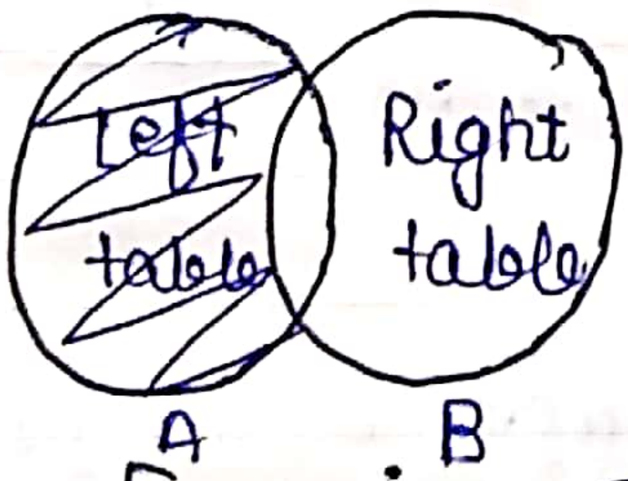
mentor_name	name
Ram	Riti
Ram	Rahul
Riti	Riya

Exclusive Join :-

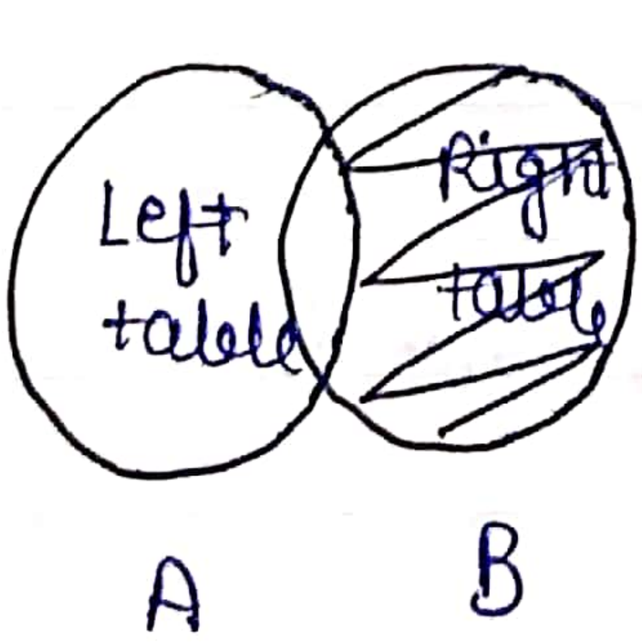
Exclusive joins are used when we want to retrieve data from two tables excluding mutual rows. They are a part of outer joins or full outer join.

Types :-

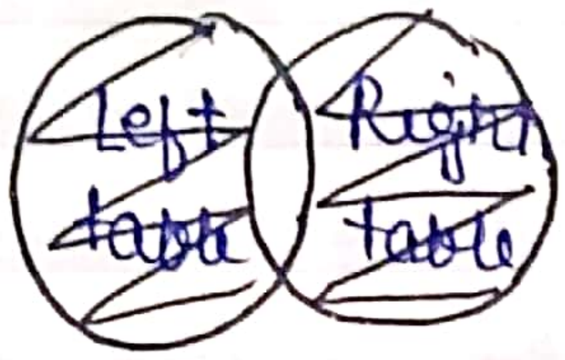
1. Left Exclusive Join -



2. Right Exclusive Join -



3. Full Exclusive Join -



- Left Exclusive Join - When we retrive records from the left table excluding the ones matching in both left & right table.

Query :-

```
Select Columns
from table 1
Left Join table 2
On table 1. colName = table 2. col Name;
Where table 2. col Name is Null;
```

id	name
101	Ram
102	Rahul
103	Riti

Customer

id	o_name
102	fruit
103	Ball
104	Umbrella

Order

- Right Exclusive Join - When we retrive records from the right table excluding the ones matching in both Left & right table.

Query :-

```
Select Columns
from table 1
Right Join table 2
On table 1. Col Name = table 2. Col Name;
Where table 1. Col Name is Null;
```

Full Exclusive Join :-

When we retrive records from the right table and left table excluding the ones matching in both left & right table.

Query :-

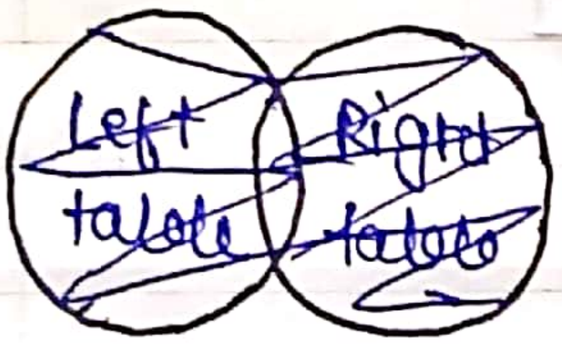
```
Select Columns
from table 1
Left Join table 2
ON. table 1. Col Name = table 2. Col Name;
Where table 2. Col Name is Null;
Union
Select Columns
from table 1
Right join table 2
On table 1. Col Name = table 2. Col Name;
Where table 1. Col Name is Null;
```

Union operators in SQL:-

Union operator in SQL is used to combine the results of two or more Select queries into a single result set and gives unique rows by removing duplicate rows.

Things we keep in mind:

1. Each Select command within the Union must retrieve the same number of columns.
2. The data types of columns in corresponding positions across Select statements should match.
3. Columns should be listed in the same order across all Select statements



Query:-

```
Select columns
from table1
Union
Select columns
from Table2;
```

id
1
2
3

+

id
2
3
4

→

id
1
2
3
4

Union All:- Union operator in SQL is used to combine the results of two or more Select queries into a single result set and gives all rows by not removing duplicate rows.

Query:-

```
Select Columnns
from table1
Union All
Select Columnns
from table2;
```

id
1
2
3

+

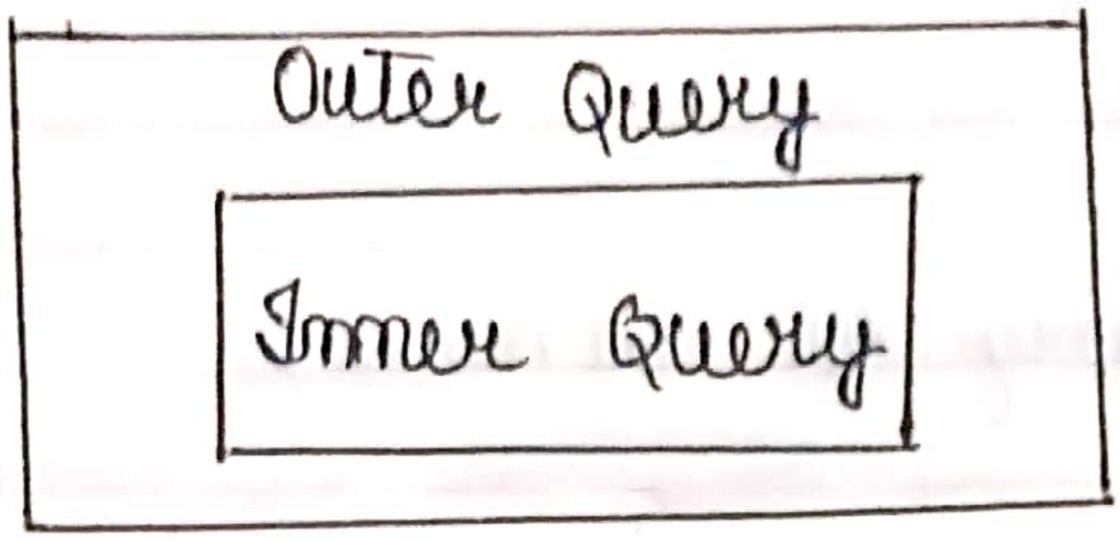
id
2
3
4

→

id
1
2
3
2
3
4

Subqueries / Inner Queries / Nested queries:-

SQL subquery is a query nested within another SQL statement. Whenever we want to retrieve data based on the result of another query we used nested queries.



⊙ He can we use subqueries?

⇒ Subqueries can be used in multiple ways:

- Subqueries can be used with clauses as Select, Insert, Update or delete to perform complex data retrieval.

Query:-

```
Select columns (subQuery)
from tableName;
```

It can be used in Multiple ways:

- It can be used with where clause to filter data based on conditions.

Query:-

```
Select
from table Name
Where column name operator (subquery);
```

How can we use Subqueries -

- Subqueries can be used in the form clause.

Query:-
Select *
From Subquery AS altName;

Eg:- Subqueries in Where:

1. Find all the employees who have salary greater than the min salary

- Find the main salary.
- Find the employee having salary greater than min salary.

id	Name	age	department	city	Salary
1	Rahul	25	IT	Mumbai	1500
2	Afsara	26	HR	Pune	2000
3	Abhimanyu	27	IT	Mumbai	2500
4	Aditya	25	Marketing	Surat	2400
5	Raj	24	Finance	Indore	1500

- To find the min salary.

Query:
Select Avg(salary) from employee

- To find all the employees having salary greater than min salary.

Query :-
Select name , salary
from employee
Where salary > (subquery)

Let's understand from example of using subqueries in Where :

2. Find the employees with the min. age.
   - Find the min age :
   - Find employee having the min age.

To find the Min age :-

Query :-
Select Min (age) from employee.

- To find employees having min. age.

Query :-
Select name, age
from employe
Where Age = (sub query);

1. Find the employees who is having age greater than Min_age.
   - Find Min age.
     Select min (age) As min_ age from employee

- Find employee having age > min age.

Query:
```
Select emp. name
from employee emp (subquery) AS subquery
Where emp.age > subquery. min_age ;
```

1. Print the employees with the average age and age of employees.

- Find avg age

⇒ Query:-
```
Select Avg (age) from employee.
```

Print the employee Age & Avg.age.

Query:-
```
Select (subquery) AS avg_age , age
from employee ;
```

Q. Find the nth highest salary in a given dataset.

Steps:-

1. Select the column which you want to show the final result i.e salary.
2. Order the salary in descending order so that you have the max at the first
3. Now the value of n could 1, 2, 3.. till n, so we have to make the

query in such a way so that whenever be the value of n it can provide the result.

4. So at the end of the query we will provide a LIMIT so that on data set which we have got after ordering the salary in descending order, we can fetch the nth highest one.

Limit - Limit clause is used to restrict the no. of rows returned by a Query.

- Limit n - It helps to retrieve a maximum of n rows from the beginning of the result set.
- Limit m,n - It helps to retrieve a specific range of rows where

m - no. of rows to skip from the beginning.

n - no. of rows to fetch after skipping

Find the nth highest salary in a given dataset.

Query:

```
Select Distinct salary
from tableName
Order by salary DESC
Limit n-1,1;
```

Stored Procedures:-

These are programs that can perform specific tasks based on the stored query. It is basically a collection of pre-written query statements grouped together under a specific name.

Query:- (to create procedure)
Create Procedure procedureName()
Begin
Query
End;

Query:- (to call the procedure)
Call procedureName()

Examples:- Stored procedure without params

Query 1:
Create procedure getAllOrderDetails()
Begin
Select * from orders;
End;

Query: (to call the procedure)
Call getAllOrderDetails();

- Sometimes we encounter an issue in SQL Workbench so we can use delimiter there

1 Query:-

```
Delimiter /
Create Procedure get All Order Details ()
Begin
Select * from Orders;
End /
!!
Demeter ;
```

Query: (to call the procedure)

```
Call get All Order Details ();
```

- Return the details of the order by id.

Query 2:

```
Create Procedure getAll Order Details By Id (IN id int
Begin
Select * from Orders Where id = id;
End;
```

Query: (to call the procedure)

```
Call get All Order Details By Id (2);
```

A view is a virtual table in SQL. It helps in providing a filtered view of data for security purposes.

Query:-

```
Create View view name AS
```

Select columns from Base Table Name; (Specify the columns to be included in the view).
It helps in Data Abstraction, Security and simplify complex queries.

- To see all the data in view.

Query:
Select * from View Name;

- To drop a view

Query:
Drop View if Exists View Name;