**MYSQL**

**SQL (Structured Query Language)** is a standard programming language used to manage and manipulate **relational databases**. It allows you to **create, read, update, and delete** data — commonly referred to as **CRUD operations**.

MySQL is an **open-source relational database management system (RDBMS)** developed by **Oracle Corporation**. It stores data in **tables (rows and columns)** and uses **SQL (Structured Query Language)** for querying and managing the data.

(https://onecompiler.com/sqlserver/)

1.Domain Driven Design

Educational Application

           Student, Teacher, Course, Department, Subject

Banking Application

           Account, Customer, Fund Transfer …

Retail Application

           Product, Order, Cart, Payment

**Crud Operation**

-- create

create database manage;

use manage;

create table students (

student\_id int primary key,

name varchar(20),

course varchar(25),

joining\_date date

);

-- insert

insert into students values

(1,'Harish','Data Science','2025-07-17'),

(2,'Vishal','Data Engineering','2025-07-18'),

(3,'Vasanth','Data Science','2025-07-20'),

(4,'Saravanan', 'Data Engineering','2025-07-20');

-- fetch

-- select \* from students;

-- select name,course from students;

-- Using where

-- select \* from students where course='Data Engineering';

-- select \* from students where joining\_date > '2025-07-18';

-- VARIATIONS – (and, in, between)

-- select \* from students where course='Data Engineering' and joining\_date='2025-07-20';

-- select \* from students where course in ('Data Science','Python');

-- select \* from students where joining\_date between '2025-07-18' and '2025-07-22';

-- LIKE

-- select \* from students where name like 'V%';

-- select \* from students where name like '%l';

-- select \* from students where name like '%an%';

-- UPDATE

-- update students set course='Advanced Data Engineering' where student\_id=1;

-- update students set joining\_date='2025-07-18' where name='Harish';

-- DELETE

delete from students where student\_id=2;

delete from students where joining\_date<'2025-07-18';

select \* from students;

**Learning Subquery**

CREATE DATABASE simple\_sql;

USE simple\_sql;

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100),

department VARCHAR(50),

salary INT,

age INT

);

INSERT INTO employees VALUES

(1, 'Amit', 'HR', 30000, 25),

(2, 'Neha', 'IT', 45000, 28),

(3, 'Rahul', 'IT', 50000, 30),

(4, 'Divya', 'Sales', 40000, 26),

(5, 'Kiran', 'Sales', 35000, 24),

(6, 'Meena', 'HR', 32000, 29);

SUBQUERY

* A **subquery** is a query **nested inside another SQL query**. It returns a value or result set used by the main query.
* Used in where, having, select.
* Acts as value or condition

INLINE SUBQUERY

* A subquery **used inside the FROM clause** and treated like a temporary table (alias required).
* The outer query queries the **result set** of the inner query.
* Used in from
* Acts as a temporary table needs alias name

-- SUBQUERY

select \* from employees

where salary>

(select avg(salary) from employees);

-- INLINE QUERY

select dept\_avg.department,dept\_avg.avg\_salary

from(

select department,avg(salary) as avg\_salary

from employees

group by department) as dept\_avg;

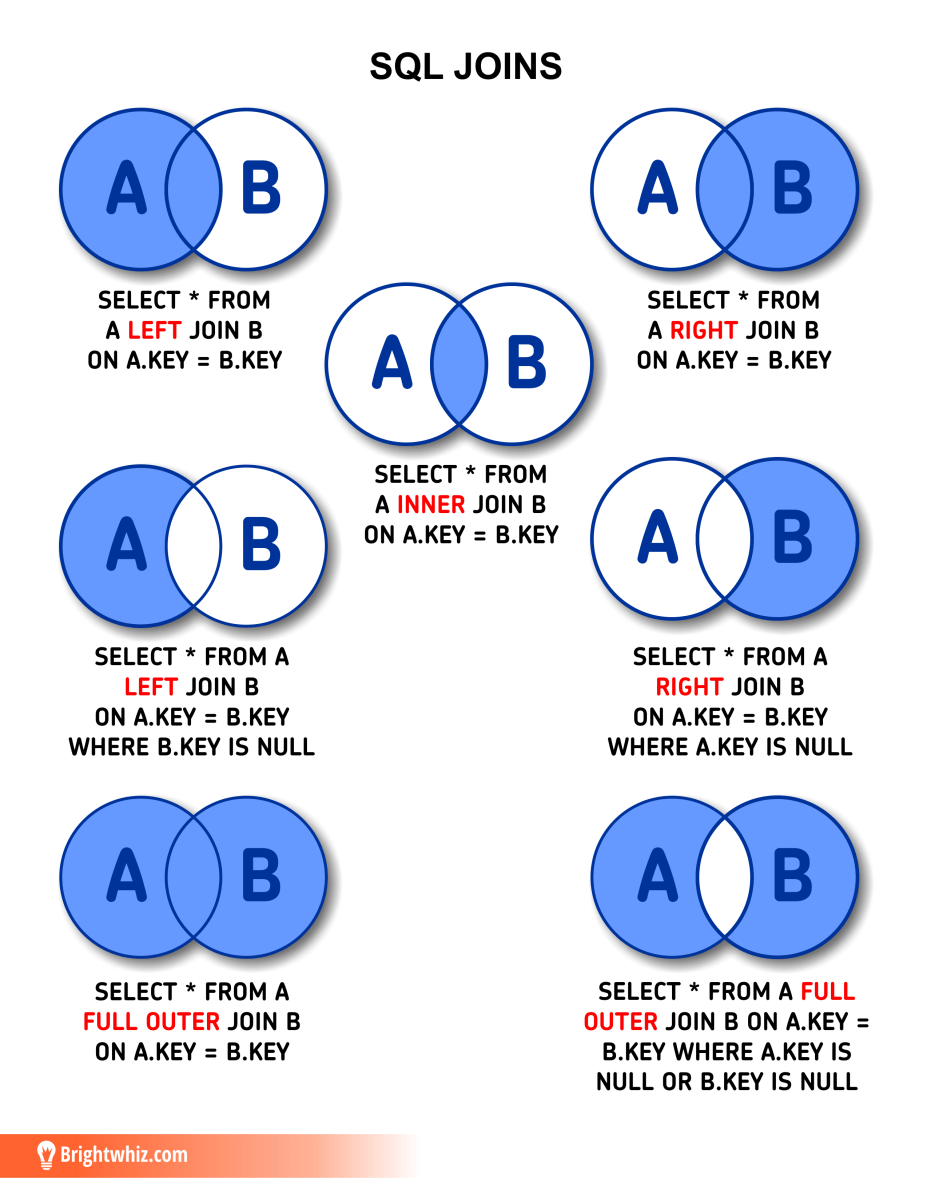
-- RANK

select emp\_name,department,salary,

rank() over (order by salary desc) as salary\_rank

from employees;

**Joins**



create database analytics\_practice;

use analytics\_practice;

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(100),

city VARCHAR(50)

);

INSERT INTO customers VALUES

(1, 'Amit Sharma', 'Delhi'),

(2, 'Neha Reddy', 'Hyderabad'),

(3, 'Rahul Iyer', 'Mumbai'),

(4, 'Divya Mehta', 'Chennai');

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

product\_name VARCHAR(100),

order\_amount INT,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

INSERT INTO orders VALUES

(101, 1, 'Laptop', 55000),

(102, 2, 'Mouse', 500),

(103, 1, 'Keyboard', 1500),

(104, 3, 'Monitor', 7000),

(105, 2, 'Printer', 8500);

-- INNER JOIN

select customers.customer\_name,orders.product\_name,orders.order\_amount

from customers

inner join orders

on customers.customer\_id=orders.customer\_id;

-- LEFT JOIN

select customers.customer\_name,orders.product\_name

from customers

left join orders

on customers.customer\_id=orders.customer\_id;

-- RIGHT JOIN

select orders.product\_name,customers.customer\_name

from customers

right join orders

on customers.customer\_id=orders.customer\_id;

-- JOINS WITH CONDITION

select customers.customer\_name,orders.product\_name,orders.order\_amount

from customers

join orders

on customers.customer\_id=orders.customer\_id

where orders.order\_amount>5000;

select o.order\_id,c.customer\_name,c.city,o.product\_name,o.order\_amount

From orders o

join customers c

on o.customer\_id=c.customer\_id;

-- JOIN WITH GROUP BY

select c.customer\_name,count(o.order\_id) as Total\_orders

from customers c

join orders o

on o.customer\_id = c.customer\_id

group by c.customer\_name

having Total\_orders >1;

select c.customer\_name,sum(order\_amount) as total\_spent

from customers c

join orders o

on c.customer\_id = o.customer\_id

group by c.customer\_name;

select c.customer\_name

from customers c

left join orders o

on c.customer\_id=o.customer\_id

where o.order\_id is null;

select c.city,count(o.order\_id) as Order\_count

from customers c

join orders o

on c.customer\_id=o.customer\_id

group by c.city;