

LINKS

<https://onecompiler.com/sqlserver/> - one compiler (Online)

<https://dev.mysql.com/downloads/mysql/> - MySQL Server (Community Edition)

<https://www.mysql.com/products/workbench/> - MySQL Workbench

Domain Driven Design

Educational Application

           Student, Teacher, Course, Department, Subject

Banking Application

           Account, Customer, Fund Transfer …

Retail Application

           Product, Order, Cart, Payment

MySQL,

* MySQL is an open-source relational database management system (RDBMS).
* It uses Structured Query Language (SQL) for accessing and managing data.

Data Storage

* Stores data in tables, rows, and columns.
* Uses a fixed schema (predefined structure).

Advantages

* Free and open-source (with enterprise options).
* Fast read operations and reliable performance.
* Well-documented and has a large community.

Limitations

* Not ideal for unstructured or semi-structured data.
* Vertical scaling is often required for large loads.
* Less flexible than NoSQL systems when schema changes are frequent.

TABLE CREATION

create table students(

student\_id int primary key,

name varchar(100),

dept varchar(100),

join\_date Date

);

insert into students(student\_id,name,dept,join\_date) VALUES

(1, 'Amit Sharma', 'Data Engineering', '2025-09-15'),

(2, 'Neha Verma','Data Science','2025-09-17'),

(3,'Rohit Iyer','Data Engineering','2025-09-20');

VARIATIONS

select \* from students;

select name, dept from students;

select \* from students where dept = 'Data Engineering';

select \* from students where join\_Date > '2025-09-15';

select \* from students

where dept='Data Engineering' and join\_Date > '2025-09-15';

-- students in either Data Science or AI

select \* from students

where dept in('Data Science', 'AI');

-- students who joined between sept 15 and sept 20

select \* from students

where join\_Date between '2025-09-15' and '2025-09-20';

LIKE

select \* from students where name like 'A%';

select \* from students where name like '%a';

select \* from students where name like '%it%';

UPDATE

update students

set dept='Advanced Data Engineering'

where student\_id=1;

update students

set join\_Date='2025-09-18'

where name = 'Neha Verma';

select \* from students;

update students

set join\_Date=DATE\_ADD('2025-09-18',interval 1 day);

DELETE

delete from students

where student\_id=2;

delete from students

where join\_date < '2025-09-16';

select \* from students;

CRUD operations - create, read, update, delete

EXERCISES

create database retail\_store;

use retail\_store;

create table products(

products\_id int primary key,

product\_name varchar(50),

category varchar(50),

price decimal(10,2),

stock\_quantity int,

added\_Date date

);

insert into products(products\_id,product\_name,category,price,stock\_quantity,added\_Date)

values

(1,'Headphones','Electronics',300.00,80,'2023-01-01'),

(2,'Mouse','Electronics',500.50,15,'2023-02-01'),

(3,'Laptop','Electronics',45500.00,8,'2023-03-01'),

(4,'Sofas','Furniture',85599.00,52,'2023-04-01'),

(5,'Chairs','Furniture',599.50,66,'2023-05-01');

select \* from products;

CREATE DATABASE analytics\_practice;

USE analytics\_practice;

CREATE TABLE sales\_data (

sale\_id INT PRIMARY KEY,

employee\_name VARCHAR(100),

region VARCHAR(50),

sale\_amount DECIMAL(10,2),

sale\_date DATE

);

INSERT INTO sales\_data VALUES

(1, 'Amit Sharma', 'North', 12000.50, '2024-01-15'),

(2, 'Neha Reddy', 'East', 8500.00, '2024-01-16'),

(3, 'Faizan Ali', 'North', 10000.00, '2024-01-20'),

(4, 'Divya Iyer', 'South', 13000.00, '2024-01-21'),

(5, 'Kiran Mehta', 'East', 9000.00, '2024-01-22'),

(6, 'Amit Sharma', 'North', 15000.00, '2024-02-05'),

(7, 'Neha Reddy', 'East', 8000.00, '2024-02-10'),

(8, 'Faizan Ali', 'North', 7000.00, '2024-02-15'),

(9, 'Divya Iyer', 'South', 14000.00, '2024-02-18'),

(10, 'Kiran Mehta', 'East', 6500.00, '2024-02-20');

-- show employees whose total sales > average sales of all employees

select employee\_name

from sales\_data

group by employee\_name

having sum(sale\_amount)>

(select avg(sale\_amount) from sales\_data);

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);

-- show employees who earn more than the average salary

select \* from employees

where salary > (select avg(salary) from employees); -- subquery

-- show department-wise average salary using a derived table

-- when we write the subquery in from that is called as inline subquery

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

from(

select department, avg(salary) as avg\_salary

from employees

group by department

)as dept\_avg;

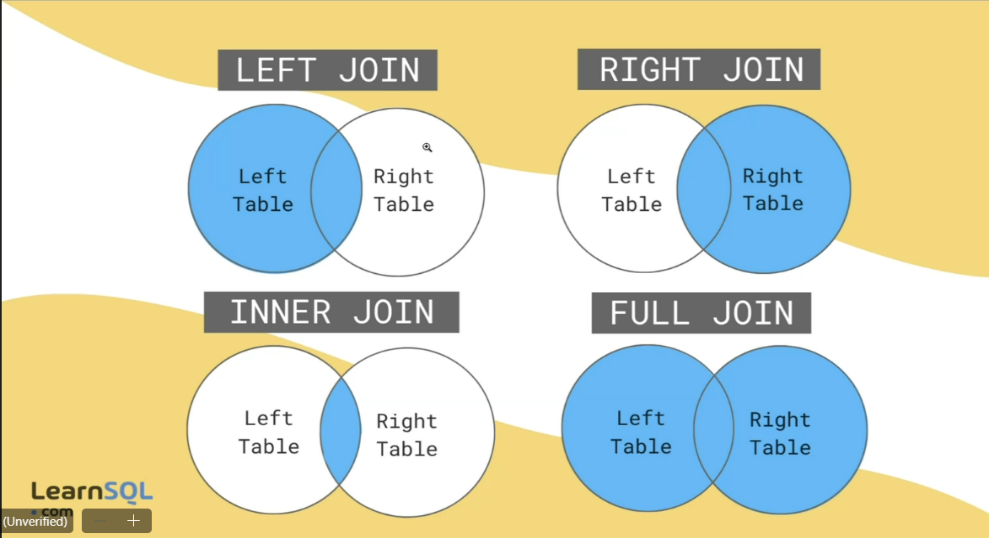
-- show employees with their rank based on salary (highest first)

-- rank() over(order by) is an analytic function

select emp\_name, department, salary,

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

from employees;

JOINS

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,orders.order\_amount

from customers

left join orders

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

-- right join

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

from customers

right join orders

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

-- full join is not supported in this compiler

-- join 2 tables where order\_amount is greater than 5000

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;

-- providing alias

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

from orders o

join customers c

using(customer\_id);

-- more than 1 orders

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

from customers c

join orders o using(customer\_id)

group by c.customer\_name

having total\_orders>1;

-- total amount spent by each customer

select c.customer\_name,sum(o.order\_amount)

as total\_spent

from customers c join orders o

using(customer\_id)

group by c.customer\_name;

-- the customers who have not placed any orders

select c.customer\_name

from customers c

left join orders o using(customer\_id)

where o.order\_id is null;

-- get which city is placing how many orders

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

from customers c join orders o

using(customer\_id)

group by c.city;

create database hex;

use hex;

create table departments (

dept\_id int primary key,

dept\_name varchar(100)

);

insert into departments values

(1,'Human Resources'),

(2, 'Engineering'),

(3,'Marketing');

create table employees(

emp\_id int primary key,

emp\_name varchar(100),

dept\_id int,

salary int

);

insert into employees values

(101,'Amit Sharma',1,30000),

(102,'Neha Reddy',2,45000),

(103,'Faizan Ali',2,48000),

(104,'Divya Mehta',3,35000),

(105,'Ravi Verma',NULL,28000);

-- JOIN-BASED Questions

-- 1. Show all employees with their department names.

select e.emp\_id,e.emp\_name,d.dept\_name from departments d

join employees e using(dept\_id);

-- 2. List employees who do not belong to any department.

select e.emp\_id,e.emp\_name from departments d

right join employees e using(dept\_id)

where d.dept\_id is null;

-- 3. Display the total number of employees in each department.

select d.dept\_name,count(e.emp\_id) as total\_employees from departments d

left join employees e using(dept\_id)

group by dept\_name;

-- 4. Show departments with no employees.

select d.dept\_name from departments d

left join employees e using(dept\_id)

where e.emp\_id is null;

-- 5. List employee names and department names for those who earn more than 40,000.

select e.emp\_name, d.dept\_name from departments d

join employees e using(dept\_id)

where e.salary > 40000;

ASSIGNMENT -1

-- Task 1: Create a Table

create table products(

products\_id int primary key,

product\_name varchar(50),

category varchar(50),

price decimal(10,2),

stock\_quantity int,

added\_Date date

);

-- Task 2: Insert Records

insert into products(products\_id, product\_name, category, price, stock\_quantity, added\_Date)

values

(1,'Headphones','Electronics',300.00,80,'2023-01-01'),

(2,'Mouse','Electronics',500.50,15,'2023-02-01'),

(3,'Laptop','Electronics',45500.00,8,'2023-03-01'),

(4,'Sofas','Furniture',85599.00,52,'2023-04-01'),

(5,'Chairs','Furniture',599.50,66,'2023-05-01');

-- Task 3: Write Queries

-- 1. List all products.

select \* from products;

-- 2. Display only product\_name and price.

select product\_name,price from products;

-- 3. Find products with stock\_quantity less than 10.

select \* from products where stock\_quantity < 10;

-- 4. Find products with price between 500 and 2000.

select \* from products where price between 500 and 2000;

-- 5. Show products added after 2023-01-01 .

select \* from products where added\_Date > '2023-01-01';

-- 6. List all products whose names start with ‘S’.

select \* from products where product\_name like 'S%';

-- 7. Show all products that belong to either Electronics or Furniture.

select \* from products where category in ('Electronics','Furniture');

-- Task 4: Update & Delete

-- 1. Update the price of one product.

update products

set price = 399.50 where products\_id=1;

-- 2. Increase stock of all products in a specific category by 5.

update products

set stock\_quantity=stock\_quantity+5 where category='Electronics';

-- 3. Delete one product based on its product\_id .

delete from products where products\_id=5;

-- 4. Delete all products with stock\_quantity = 0.

delete from products where stock\_quantity=0;

select \* from products;