Day 1 – SQL Notes

Date: 17-07-2025

Domain Driven Design

# Applications

Educational Application: Student, Teacher, Course, Department, Subject

Banking Application: Account, Customer, Fund Transfer …

Retail Application: Product, Order, Cart, Payment

# SQL Queries

## Students Table

-- create  
CREATE TABLE students (  
 student\_id int PRIMARY KEY,  
 name varchar(100),  
 course varchar(100),  
 join\_date DATE  
);  
  
-- insert  
INSERT INTO students values  
(1,'Amit','Data engineering','2025-08-09'),  
(2,'Rohan','Data science','2025-07-19'),  
(3,'Rohit','Data engineering','2025-09-10');  
  
SELECT \* FROM students;  
  
SELECT name,course FROM students;  
  
SELECT \* FROM students where course= 'Data engineering';  
  
SELECT \* FROM students where join\_date>'2025-07-19';

-- Matching patterns  
-- starting with A  
SELECT \* FROM students where name like 'A%';  
  
-- end with A  
SELECT \* FROM students where name like '%T';  
  
-- containing 'it'  
SELECT \* FROM students where name like '%it%';  
  
update students  
set course ='Advanced Data engineering'  
where student\_id = 1;  
  
update students  
set join\_date = '2025-09-01'  
where name = 'Rohan';  
  
-- delete a student by ID  
delete FROM students   
where student\_id = 2;  
  
-- delete students by condition  
delete FROM students  
where join\_date<'2025-08-09';

## CRUD OPERATION

CREATE | READ | UPDATE | DELETE

## Products Table Tasks

-- task 1  
create table products(  
product\_id int primary key,  
product\_name varchar(100),  
category varchar(100),  
price decimal(10,2),  
stock\_quantity int,  
added\_date date);  
  
-- task 2  
insert into products values   
(1, 'Smartphone', 'Electronics', 15000.00, 20, '2023-02-15'),  
(2, 'Sofa', 'Furniture', 12000.00, 5, '2023-03-10'),  
(3, 'Shoes', 'Fashion', 2500.00, 15, '2023-05-20'),  
(4, 'Study Table', 'Furniture', 4500.00, 8, '2023-07-01'),  
(5, 'Smartwatch', 'Electronics', 3500.00, 12, '2023-08-18');  
  
-- task 3  
SELECT \* FROM products;  
select product\_name, price from products;  
select \* from products where stock\_quantity<10;  
select \* from products where price between 500 and 2000;  
select \* from products where added\_date > '2023-01-01';  
select \* from products where product\_name like 'S%';  
select \* from products where category in ('Electronics', 'Furniture');  
  
-- task 4  
update products set price = 16000.00 where product\_id = 1;  
update products set stock\_quantity = stock\_quantity + 5 WHERE category = 'Furniture';  
delete from products where product\_id = 5;  
delete from products where stock\_quantity = 0;  
  
select \* from products;

-- DAY 1 17.07.2025 MYSQL

create database retail\_store;

use retail\_store;

create table products(

product\_id int primary key,

product\_name varchar(100),

category varchar(100),

price decimal(10,2),

stock\_quantity int,

added\_date date);

insert into products values (1, 'Smartphone', 'Electronics', 15000.00, 20, '2023-02-15'),

(2, 'Sofa', 'Furniture', 12000.00, 5, '2023-03-10'),

(3, 'Shoes', 'Fashion', 2500.00, 15, '2023-05-20'),

(4, 'Study Table', 'Furniture', 4500.00, 8, '2023-07-01'),

(5, 'Smartwatch', 'Electronics', 3500.00, 12, '2023-08-18');

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');

select \* from sales\_data;

**-- inline query or subquery**

**-- show employees whose total sales > avg 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);

select \* from employees;

**-- subquery (always where clause)**

**-- show employees who earn more than avg salary**

select \* from employees

where salary >(

select avg(salary) from employees);

**-- inline query(using from)**

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

FROM (

select department, avg(salary) as avg\_salary from employees

group by department)

as dept\_avg;

**-- analytic function [ RANK() ]**

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

select emp\_name, department, salary, RANK() OVER

(order by salary desc) as salary\_rank

from employees;

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);

select \* from customers;

select \* from orders;

**-- INNER JOIN [ intersection ]**

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 customers.customer\_name, orders.product\_name

from customers

right join orders

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

**-- FILTERING**

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

from customers

right join orders

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

where orders.order\_amount>5000;

**-- ALIAS**

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;

**-- HAVING**

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

from customers c

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

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

group by c.customer\_name;

**-- customers who have taken no orders**

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;