DBMS Lab File

**Name: Sanika Santosh Kadam**

**PRN: 2124UCSF1112**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR.No** | **Program No** | **Aim** |
| 1 | 1 | Install and set up MySQL. Create a database and a table to store employee details. Perform basic operations like INSERT, UPDATE, and DELETE using SELECT queries. |
| 2 | 2 | Create a table for storing information. Insert sample data and perfect basic operations: INSERT, UPDATE, DELETE, and SELECT. |
| 3 | 3 | Create a table with columns for Employee ID, Name, Salary, Joining Date, and Active Status using different data types. Insert sample data and perform queries to manipulate and retrieve data. |
| 4 | 4 | Create a table to store employee information with constraints like Primary Key, Foreign Key, and Unique |
| 5 | 5 | Create a table for Customer details with various integrity constraints like NOT NULL, CHECK, and DEFAULT. Insert valid and invalid data to test these constraints and  ensure data integrity. |
| 6 | 6 | Use DDL commands to create tables and DML commands to insert, update, and delete data. Write SELECT queries to retrieve and verify data changes. |
| 7 | 7 | Create a Sales table and use aggregate functions like COUNT, SUM, AVG, MIN, and MAX to summarize sales data and calculate statistics |
| 8 | 8 | Given Customers and Orders tables, write SQL queries to perform INNER JOIN, LEFT JOIN, and RIGHT JOIN to retrieve combined data for customer orders. |

**Program :1**

**Aim: Install and set up MySQL. Create a database and a table to store employee details. Perform basic operations like INSERT, UPDATE, and DELETE using SELECT queries**

Query:

CREATE DATABASE EmployeeDB;

USE EmployeeDB;

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

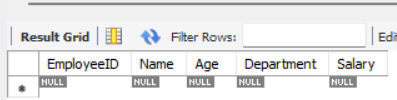
Name VARCHAR(100) NOT NULL,

Age INT CHECK (Age > 18),

Department VARCHAR(50),

Salary DECIMAL(10,2)

);



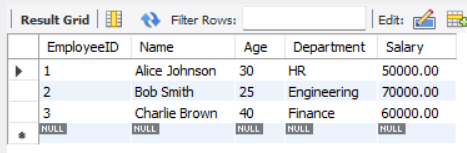
INSERT INTO Employee (Name, Age, Department, Salary)

VALUES

('Alice Johnson', 30, 'HR', 50000.00),

('Bob Smith', 25, 'Engineering', 70000.00),

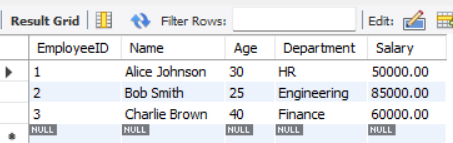
('Charlie Brown', 40, 'Finance', 60000.00);



UPDATE Employee

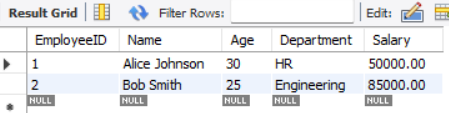
SET Salary = Salary + 5000

WHERE Name = 'Bob Smith';



DELETE FROM Employee

WHERE Name = 'Charlie Brown';



SELECT \* FROM Employee;

**Program : 2**

**Aim: Create a table for storing information. Insert sample data and perfect basic operations: INSERT, UPDATE, DELETE, and SELECT.**

CREATE DATABASE School;

CREATE CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

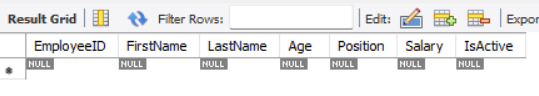
Age INT CHECK (Age >= 18),

Position VARCHAR(50),

Salary DECIMAL(10,2),

IsActive BOOLEAN DEFAULT TRUE

);



INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Position, Salary, IsActive)

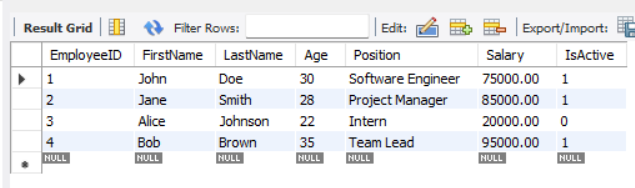
VALUES

(1, 'John', 'Doe', 30, 'Software Engineer', 75000.00, TRUE),

(2, 'Jane', 'Smith', 28, 'Project Manager', 85000.00, TRUE),

(3, 'Alice', 'Johnson', 22, 'Intern', 20000.00, FALSE),

(4, 'Bob', 'Brown', 35, 'Team Lead', 95000.00, TRUE);

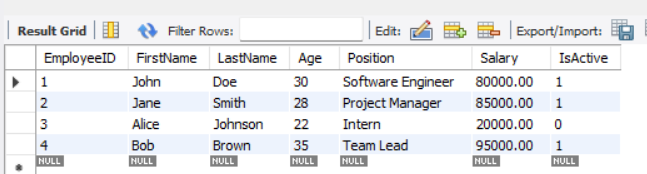


UPDATE Employees

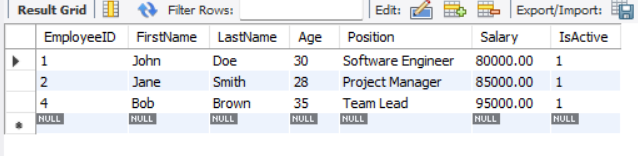
SET Salary = 80000.00

WHERE EmployeeID = 1;

SELECT \* FROM Employees;



DELETE FROM Employees WHERE EmployeeID = 3;



**Program : 3**

**Aim: Create a table with columns for EmployeeID, Name, Salary, JoiningDate, and ActiveStatus using different data types. Insert sample data and perform queries to manipulate and retrieve data.**

Query:

CREATE DATABASE Company;

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

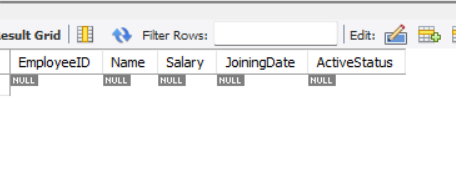
Name VARCHAR(100) NOT NULL,

Salary DECIMAL(10,2) CHECK (Salary > 0),

JoiningDate DATE NOT NULL,

ActiveStatus BOOLEAN DEFAULT TRUE

);



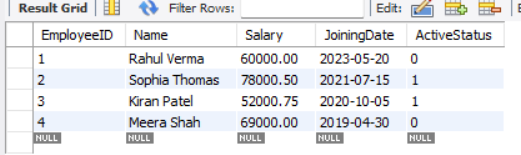
INSERT INTO Employee (Name, Salary, JoiningDate, ActiveStatus) VALUES

('Rahul Verma', 60000.00, '2023-05-20', FALSE),

('Sophia Thomas', 78000.50, '2021-07-15', TRUE),

('Kiran Patel', 52000.75, '2020-10-05', TRUE),

('Meera Shah', 69000.00, '2019-04-30', FALSE);



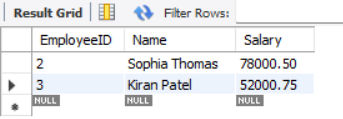
-- Retrieve All Employees

SELECT \* FROM Employee;

-- Retrieve Active Employees

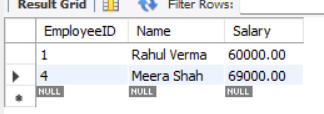
SELECT EmployeeID, Name, Salary FROM Employee WHERE

ActiveStatus = TRUE;



SELECT EmployeeID, Name, Salary FROM Employee WHERE

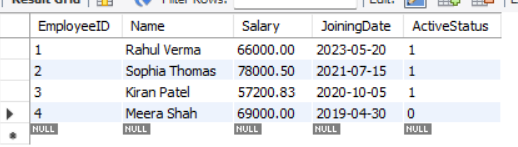
ActiveStatus = False;



UPDATE Employee SET Salary = Salary \* 1.10 WHERE EmployeeID = 1;

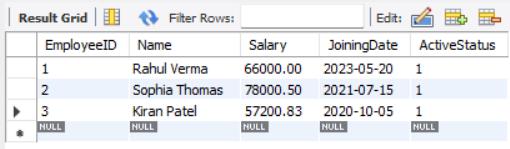
-- Change Active Status of an Employee

UPDATE Employee SET ActiveStatus = TRUE WHERE EmployeeID = 1;



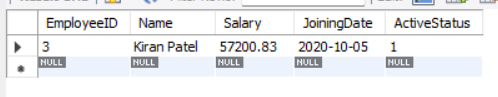
-- Delete an Employee Record

DELETE FROM Employee WHERE EmployeeID = 4;



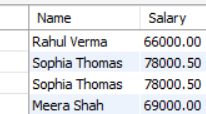
-- Retrieve Employees Who Joined in a Specific Year

SELECT \* FROM Employee WHERE YEAR(JoiningDate) = 2020



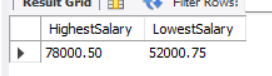
-- Retrieve Employees with Salary Greater Than a Specific Amount

SELECT Name, Salary FROM Employee WHERE Salary >60000;



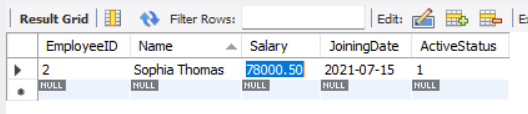
-- Find the Highest &amp; Lowest Salary in the Organization

SELECT MAX(Salary) AS HighestSalary, MIN(Salary) AS LowestSalary FROM Employee;



-- Retrieve the Top 3 Highest Paid Employees

SELECT \* FROM Employee ORDER BY Salary DESC LIMIT 3;



**Program 4:**

**Aim: Create a table to store employee information with constraints like Primary Key, Foreign Key, and Unique.**

Query:

CREATE TABLE Department (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50) UNIQUE

);

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

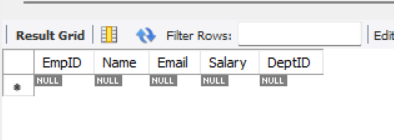
Name VARCHAR(100) NOT NULL,

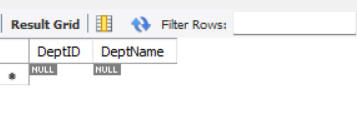
Email VARCHAR(100) UNIQUE,

Salary DECIMAL(10,2) CHECK (Salary >0),

DeptID INT REFERENCES Department(DeptID)

);

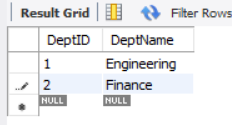




-- Insert Valid Data

INSERT INTO Department (DeptID, DeptName) VALUES (1, 'Enginerring');

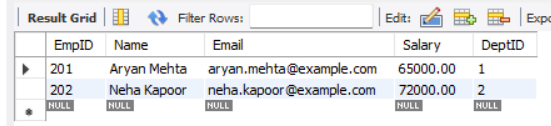
INSERT INTO Department (DeptID, DeptName) VALUES (2, 'Finance');



INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES

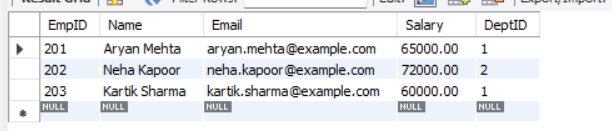
(201, 'Aryan Mehta', 'aryan.mehta@example.com', 65000.00, 1),

(202, 'Neha Kapoor', 'neha.kapoor@example.com', 72000.00, 2);



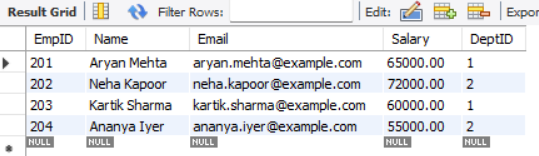
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES

(203, 'Kartik Sharma', 'kartik.sharma@example.com', 60000.00, 1);



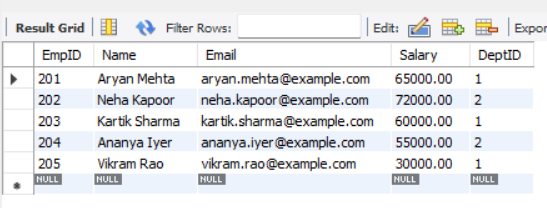
INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES

(204, 'Ananya Iyer', 'ananya.iyer@example.com', 55000.00, 2);



INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID) VALUES

(205, 'Vikram Rao', 'vikram.rao@example.com', 30000.00, 1);



**Practical 5: Testing Employee Constraints**

**Aim:** **Create a table for Customer details with various integrity constraints like NOT NULL, CHECK, and DEFAULT. Insert valid and invalid data to test these constraints and ensure data integrity.**

Query:

CREATE DATABASE Employee;

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

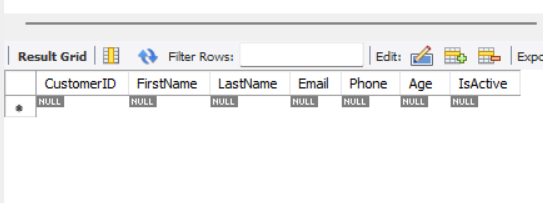
Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

Age INT CHECK (Age >= 18),

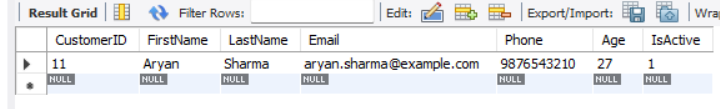
IsActive BOOLEAN DEFAULT TRUE

);



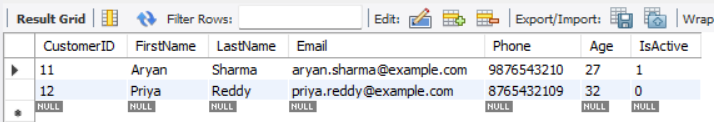
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)

VALUES (11, 'Aryan', 'Sharma', 'aryan.sharma@example.com', 9876543210, 27, TRUE);



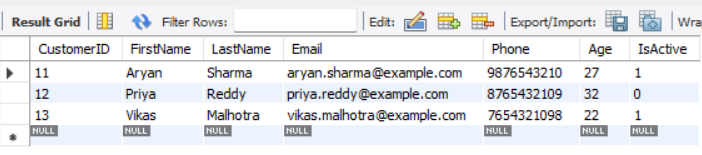
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)

VALUES (12, 'Priya', 'Reddy', 'priya.reddy@example.com', 8765432109, 32, FALSE);



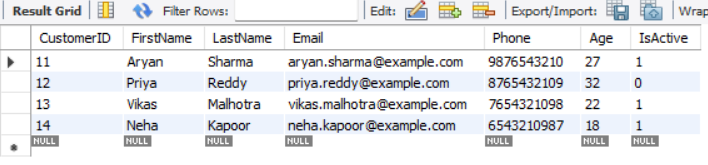
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)

VALUES (13, 'Vikas', 'Malhotra', 'vikas.malhotra@example.com', 7654321098, 22, TRUE);



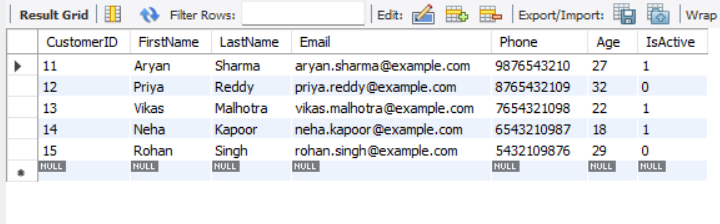
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)

VALUES (14, 'Neha', 'Kapoor', 'neha.kapoor@example.com', 6543210987, 18, TRUE);



INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)

VALUES (15, 'Rohan', 'Singh', 'rohan.singh@example.com', 5432109876, 29, FALSE);



**Practical 6:**

**Aim: Use DDL commands to create tables and DML commands to insert, update, and delete data. Write SELECT queries to retrieve and verify data changes.**

Query:

CREATE DATABASE Employee;

CREATE TABLE students (

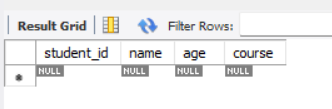
student\_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

age INT CHECK (age > 0),

course VARCHAR(50) DEFAULT 'Undeclared'

);

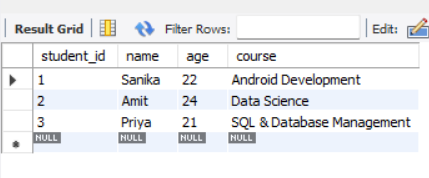


INSERT INTO students (student\_id, name, age, course) VALUES

(1, 'Sanika', 22, 'Android Development'),

(2, 'Amit', 24, 'Data Science'),

(3, 'Priya', 21, 'SQL & Database Management');

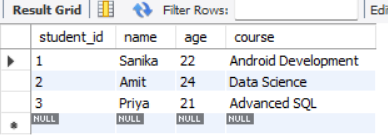


-- Updates (DML Commands)

UPDATE students

SET course = 'Advanced SQL'

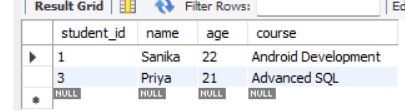
WHERE student\_id = 3;



--Delete using DML

DELETE FROM students

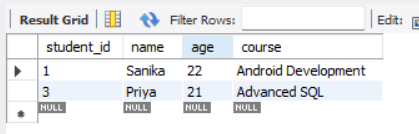
WHERE student\_id = 2;



-- Retrieving Data (SELECT)

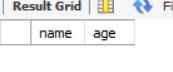
a) Get all students:

SELECT \* FROM students;



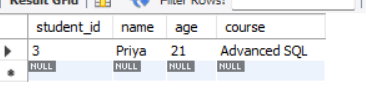
b) Find students enrolled in SQL & Database Management:

SELECT name, age FROM students WHERE course = 'SQL & Database Management';



c) Verify Priya's updated course:

SELECT \* FROM students WHERE student\_id = 3;



**Program 7:**

**Aim: Create a Sales table and use aggregate functions like COUNT, SUM, AVG, MIN, and MAX to summarize sales data and calculate statistics.**

Query:

-- Create Customer Table

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

Phone VARCHAR(15) UNIQUE NOT NULL,

Age INT CHECK (Age >= 18),

RegisteredDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Corrected default date

Balance DECIMAL(10,2) CHECK (Balance >= 0),

Status ENUM('Active', 'Inactive') DEFAULT 'Active'

);



-- Create Sales Table

CREATE TABLE Sales (

SaleID INT AUTO\_INCREMENT PRIMARY KEY, -- Unique sale ID

Product VARCHAR(100) NOT NULL, -- Product name

Quantity INT CHECK (Quantity > 0), -- Quantity must be positive

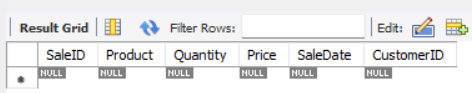
Price DECIMAL(10,2) CHECK (Price > 0), -- Price must be positive

SaleDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Default sale date

CustomerID INT, -- Foreign key reference

CONSTRAINT fk\_customer FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID) -- Links to Customer table

);



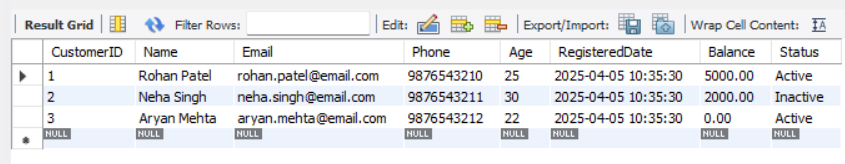
-- Insert Valid Customer Data

INSERT INTO Customer (CustomerID, Name, Email, Phone, Age, Balance, Status) VALUES

(1, 'Rohan Patel', 'rohan.patel@email.com', '9876543210', 25, 5000.00, 'Active'),

(2, 'Neha Singh', 'neha.singh@email.com', '9876543211', 30, 2000.00, 'Inactive'),

(3, 'Aryan Mehta', 'aryan.mehta@email.com', '9876543212', 22, 0.00, 'Active');



-- Insert Sales Data (Correct Table)

INSERT INTO Sales (Product, Quantity, Price, SaleDate, CustomerID) VALUES

('Refrigerator', 1, 45000.00, '2025-04-01', 1),

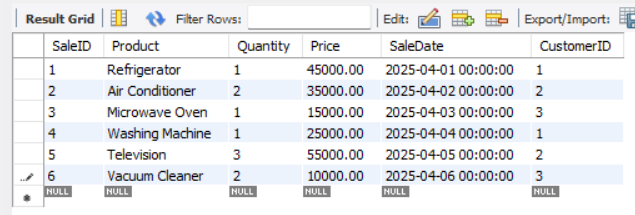
('Air Conditioner', 2, 35000.00, '2025-04-02', 2),

('Microwave Oven', 1, 15000.00, '2025-04-03', 3),

('Washing Machine', 1, 25000.00, '2025-04-04', 1),

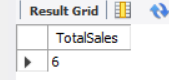
('Television', 3, 55000.00, '2025-04-05', 2),

('Vacuum Cleaner', 2, 10000.00, '2025-04-06', 3);

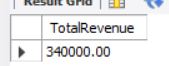


-- Aggregate Queries to Analyze Sales

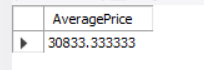
SELECT COUNT(\*) AS TotalSales FROM Sales; -- Total sales transactions



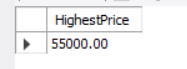
SELECT SUM(Quantity \* Price) AS TotalRevenue FROM Sales; -- Total revenue



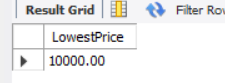
SELECT AVG(Price) AS AveragePrice FROM Sales; -- Average product price



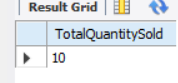
SELECT MAX(Price) AS HighestPrice FROM Sales; -- Highest sale price



SELECT MIN(Price) AS LowestPrice FROM Sales; -- Lowest sale price



SELECT SUM(Quantity) AS TotalQuantitySold FROM Sales; -- Total quantity of all items sold



**Program : 8**

**Aim: Given Customers and Orders tables, write SQL queries to perform INNER JOIN, LEFT JOIN, and RIGHT JOIN to retrieve combined data for customer orders.**

Query:

-- Create Customers table with modified attributes

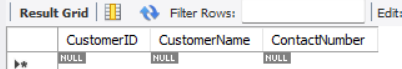
CREATE TABLE Customers (

Cust\_ID INT PRIMARY KEY,

Cust\_Name VARCHAR(100),

Phone\_Number VARCHAR(15)

);



-- Create Orders table with modified attributes

CREATE TABLE Orders (

Ord\_ID INT PRIMARY KEY,

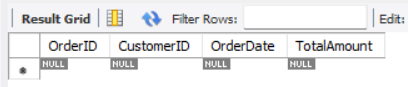
Cust\_ID INT,

Ord\_Date DATE,

Total\_Cost DECIMAL(10,2),

FOREIGN KEY (Cust\_ID) REFERENCES Customers(Cust\_ID)

);



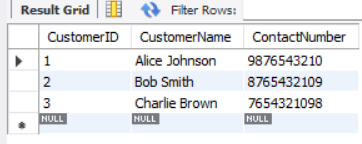
-- Insert sample data into Customers

INSERT INTO Customers (Cust\_ID, Cust\_Name, Phone\_Number) VALUES

(1, 'Alice Johnson', '9876543210'),

(2, 'Bob Smith', '8765432109'),

(3, 'Charlie Brown', '7654321098');



-- Insert sample data into Orders

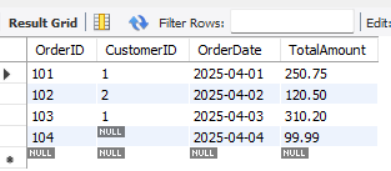
INSERT INTO Orders (Ord\_ID, Cust\_ID, Ord\_Date, Total\_Cost) VALUES

(101, 1, '2025-04-01', 250.75),

(102, 2, '2025-04-02', 120.50),

(103, 1, '2025-04-03', 310.20),

(104, NULL, '2025-04-04', 99.99); -- Order with no associated customer

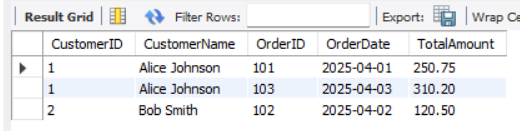


-- INNER JOIN: Retrieves records where there is a match in both tables

SELECT Customers.Cust\_ID, Customers. Cust\_Name, Orders.Ord\_ID, Orders.Ord\_Date, Orders.Total\_Cost

FROM Customers

INNER JOIN Orders ON Customers.Cust\_ID = Orders.Cust\_ID;

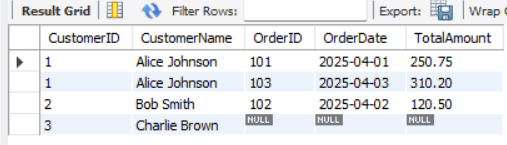


-- LEFT JOIN: Retrieves all customers, including those without orders

SELECT Customers.Cust\_ID, Customers.Cust\_Name, Orders.Ord\_ID, Orders.Ord\_Date, Orders.Total\_Cost

FROM Customers

LEFT JOIN Orders ON Customers.Cust\_ID = Orders.Cust\_ID;



-- RIGHT JOIN: Retrieves all orders, including those without associated customers

SELECT Customers.Cust\_ID, Customers.Cust\_Name, Orders.Ord\_ID, Orders.Ord\_Date, Orders.Total\_Cost

FROM Customers

RIGHT JOIN Orders ON Customers.Cust\_ID = Orders.Cust\_ID;

