🟩 Hands-On Lab: SQL for Data Analysis

Scenario

You're working as a **junior data analyst** for an online retail company. You'll be using **SQL** to explore and manipulate the company's **Customer**, **Orders**, and **Products** databases.

📂 Step 0: Create the Database & Sample Tables

```
-- Create Database
CREATE DATABASE retail_db;
USE retail_db;

-- Create Customers Table
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    city VARCHAR(50),
    country VARCHAR(50)
);

-- Create Products Table
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50),
    price DECIMAL(10,2)
);

-- Create Orders Table
CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    product_id INT,
```

```
    order_date DATE,
    quantity INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

🟢 4.1 Introduction to Databases: Tables, Rows, Columns, Keys

Step 1: Insert Sample Data

```
INSERT INTO Customers VALUES
(1, 'Alice', 'Brown', 'New York', 'USA'),
(2, 'John', 'Smith', 'Los Angeles', 'USA'),
(3, 'Maria', 'Garcia', 'Madrid', 'Spain');

INSERT INTO Products VALUES
(101, 'Laptop', 'Electronics', 1200.00),
(102, 'Smartphone', 'Electronics', 800.00),
(103, 'Desk Chair', 'Furniture', 150.00);

INSERT INTO Orders VALUES
(1001, 1, 101, '2025-08-01', 1),
(1002, 2, 103, '2025-08-02', 2),
(1003, 1, 102, '2025-08-03', 1);
```

✅ **Task:** View all customers:

```
SELECT * FROM Customers;
```

🟢 4.2 Basic SQL Queries

Step 2: SELECT, WHERE, LIMIT, ORDER BY

```sql
-- Select specific columns
SELECT first_name, last_name, city FROM Customers;

-- Filter customers from USA
SELECT * FROM Customers
WHERE country = 'USA';

-- Show top 2 most expensive products
SELECT * FROM Products
ORDER BY price DESC
LIMIT 2;
```

✅ **Task:** Find all orders for customer_id = 1.

🟢 4.3 Filtering and Aggregations

Step 3: GROUP BY, HAVING, COUNT, AVG, SUM

```sql
-- Count orders per customer
SELECT customer_id, COUNT(order_id) AS total_orders
FROM Orders
GROUP BY customer_id;

-- Average product price per category
SELECT category, AVG(price) AS avg_price
FROM Products
GROUP BY category;

-- Total quantity ordered per customer (only those with more than 1 item)
SELECT customer_id, SUM(quantity) AS total_items
FROM Orders
GROUP BY customer_id
HAVING SUM(quantity) > 1;
```

✅ **Task:** Find total sales amount per customer.

🟢 4.4 Joins and Subqueries

Step 4: INNER, LEFT, RIGHT, FULL JOIN

```
-- Inner Join to see order details with customer names
SELECT o.order_id, c.first_name, p.product_name, o.quantity
FROM Orders o
INNER JOIN Customers c ON o.customer_id = c.customer_id
INNER JOIN Products p ON o.product_id = p.product_id;

-- Left Join: Show all customers even if they have no orders
SELECT c.first_name, o.order_id
FROM Customers c
LEFT JOIN Orders o ON c.customer_id = o.customer_id;
```

✅ **Task:** Use a subquery to find customers who ordered products in the “Electronics” category.

```
SELECT DISTINCT first_name, last_name
FROM Customers
WHERE customer_id IN (
    SELECT customer_id FROM Orders
    WHERE product_id IN (
        SELECT product_id FROM Products
        WHERE category = 'Electronics'
    )
);
```

🟢 4.5 Data Manipulation

Step 5: INSERT, UPDATE, DELETE

```sql
-- Insert a new product
INSERT INTO Products VALUES (104, 'Office Desk', 'Furniture', 250.00);

-- Update product price
UPDATE Products
SET price = 1300.00
WHERE product_id = 101;

-- Delete an order
DELETE FROM Orders
WHERE order_id = 1003;
```

✅ **Task:** Add a new customer, then place an order for them.

🎯 Lab Completion

By completing this lab, students will:

- Understand **database structure**
- Write **basic and advanced queries**
- Perform **aggregations and joins**
- Manipulate data with **INSERT, UPDATE, DELETE**
- Apply **real-world SQL skills** to analysis
