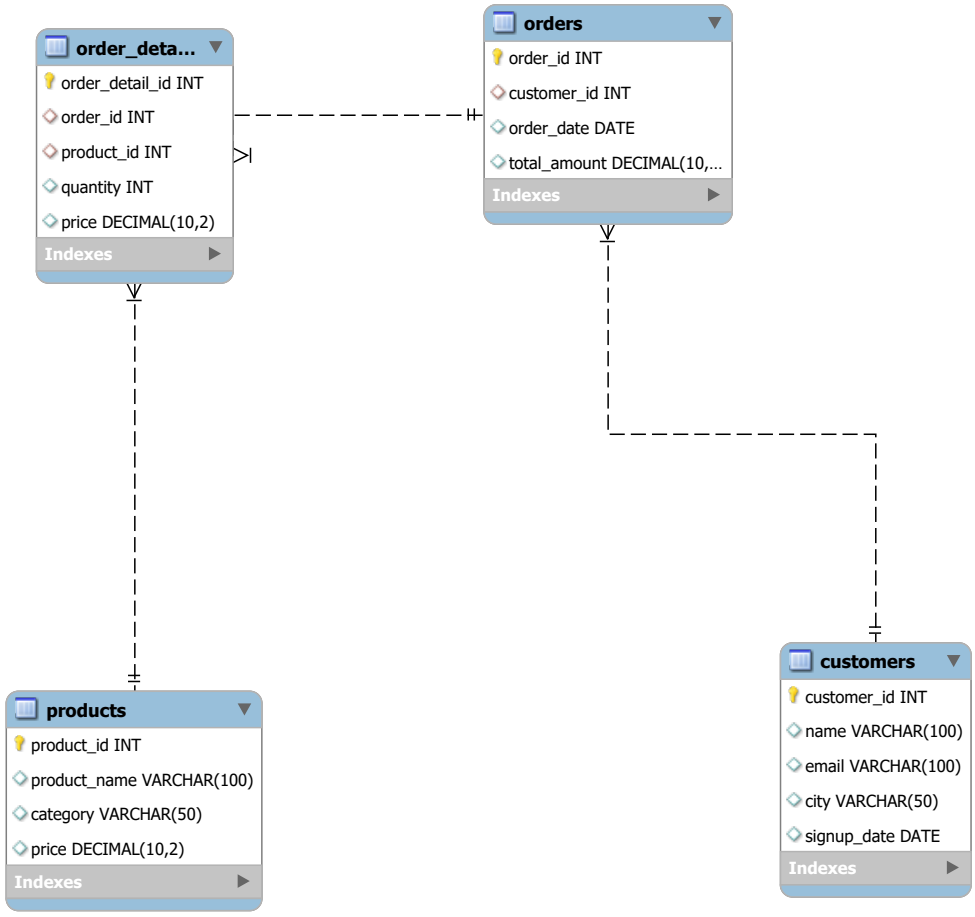
TABLE CREATION AND INSERTING DATA

```
CREATE TABLE Customers (
   customer_id INT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100),
   city VARCHAR(50),
   signup_date DATE
);

INSERT INTO Customers (customer_id, name, email, city, signup_date) VALUES
(1, 'Aarav Mehta', 'aarav.mehta@email.com', 'Mumbai', '2024-02-10'),
(2, 'Priya Nair', 'priya.nair@email.com', 'Bangalore', '2024-01-15'),
(3, 'Rohan Singh', 'rohan.singh@email.com', 'Delhi', '2023-11-05'),
(4, 'Meera Shah', 'meera.shah@email.com', 'Mumbai', '2024-03-20'),
(5, 'Vikram Patel', 'vikram.patel@email.com', 'Hyderabad', '2024-04-01');

CREATE TABLE Orders (
   order_id INT PRIMARY KEY,
   customer_id INT,
   order_date DATE,
   total_amount DECIMAL(10,2),
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
INSERT INTO Orders (order_id, customer_id, order_date, total_amount) VALUES
(101, 1, '2025-06-01', 80000.00),
(102, 2, '2025-05-15', 25000.00),
(103, 3, '2025-04-20', 999.00),
(104, 1, '2025-04-22', 125000.00),
(105, 4, '2025-05-29', 69999.00);

CREATE TABLE Products (
```

```
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50),
    price DECIMAL(10,2)
);

INSERT INTO Products (product_id, product_name, category, price) VALUES
(1, 'iPhone 15', 'Electronics', 79999.00),
(2, 'Samsung Galaxy S23', 'Electronics', 69999.00),
(3, 'Sony WH-1000XM5', 'Audio', 24999.00),
(4, 'Logitech Mouse', 'Accessories', 999.00),
(5, 'USB-C Cable', 'Accessories', 499.00),
(6, 'Google Pixel 8', 'Electronics', 74999.00);

CREATE TABLE Order_Details (
    order_detail_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    price DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

INSERT INTO Order_Details (order_detail_id, order_id, product_id, quantity, price) VALUES
(1001, 101, 1, 1, 79999.00),
(1002, 102, 3, 1, 24999.00),
(1003, 103, 4, 1, 999.00),
(1004, 104, 1, 1, 79999.00),
(1005, 104, 6, 1, 74999.00),
(1006, 105, 2, 1, 69999.00);
```


order_deta...
order_detail_id INT
order_id INT
product_id INT
quantity INT
price DECIMAL(10,2)
Indexes
orders
order_id INT
customer_id INT
order_date DATE
total_amount DECIMAL(10,...
Indexes
products
product_id INT
product_name VARCHAR(100)
category VARCHAR(50)
price DECIMAL(10,2)
Indexes
customers
customer_id INT
name VARCHAR(100)
email VARCHAR(100)
city VARCHAR(50)
signup_date DATE
Indexes

BASIC QUERIES

1. Get the list of **all customers.**
SELECT * FROM Customers;

2. Find all orders placed in the last 30 days.
SELECT * FROM Orders WHERE DATEDIFF(CURDATE(), order_date) <= 30;

3. Show product names and their prices.
SELECT product_name, price FROM Products;

4. Find the total number of products in each category.
SELECT category, COUNT(*) AS product_count FROM Products GROUP BY category;

FILTERING AND CONDITIONS

5. Get all customers from the city 'Mumbai'.
SELECT * FROM Customers WHERE city = 'Mumbai';

6. Find orders with a total amount greater than ₹5000.
SELECT * FROM Orders WHERE total_amount > 5000;

7. List customers who signed up after '2024-01-01'.
SELECT * FROM Customers WHERE signup_date > '2024-01-01';

JOINS

8. Show all orders along with the customer's name.
SELECT o.order_id, o.order_date, o.total_amount, c.name AS customer_name
FROM Orders o LEFT JOIN Customers c ON o.customer_id = c.customer_id;

9. List products purchased in each order.
SELECT od.order_id, p.product_name, od.quantity, od.price
FROM Order_Details od JOIN Products p ON od.product_id = p.product_id
ORDER BY od.order_id;

10. Find customers who have never placed an order.
SELECT c.customer_id, c.name
FROM Customers c LEFT JOIN Orders o ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;

AGGREGATION AND GROUPING

11. Find the total amount spent by each customer.
SELECT c.customer_id, c.name, SUM(o.total_amount) AS total_spent
FROM Customers c
LEFT JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id;

12. Which product has been sold the most (by quantity)?

```
SELECT p.product_id, p.product_name, SUM(od.quantity) AS total_quantity_sold
FROM Order_Details od
JOIN Products p ON od.product_id = p.product_id
GROUP BY p.product_id
ORDER BY total_quantity_sold DESC
LIMIT 1;
```

13. Find the average order value for each customer.

```
SELECT c.customer_id, c.name, AVG(o.total_amount) AS average_order_value
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id;
```

14. Total sales amount per product category.

```
SELECT p.category, SUM(od.quantity * od.price) AS total_sales_amount
FROM Order_Details od
JOIN Products p ON od.product_id = p.product_id
GROUP BY p.category;
```

SUBQUERIES

15. Find customers who spent more than the average spending.

```
SELECT c.customer_id, c.name, SUM(o.total_amount) AS total_spent
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
HAVING SUM(o.total_amount) > (
SELECT AVG(total_spent) FROM (
SELECT customer_id, SUM(total_amount) AS total_spent
FROM Orders
GROUP BY customer_id
) AS customer_avg
);
```

16. List products that have never been ordered.

```
SELECT product_name
FROM Products
WHERE product_id NOT IN (
SELECT DISTINCT product_id FROM Order_Details
);
```

17. Find the most recent order for each customer.

```
SELECT o.* FROM Orders o
WHERE order_date = (
SELECT MAX(o2.order_date) FROM Orders o2 WHERE o2.customer_id = o.customer_id );
```

ADVANCED QUERIES

18. Rank customers by total spending (highest first).

```
SELECT c.customer_id, c.name, SUM(o.total_amount) AS total_amount_spent,
RANK() OVER (ORDER BY SUM(o.total_amount) DESC) AS spending_rank
FROM Customers c  JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id;
```

19. Get the top 3 customers based on the number of orders placed.

```
SELECT c.customer_id, c.name, COUNT(*) AS number_of_orders
FROM Customers c  JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY number_of_orders DESC
LIMIT 3;
```

20. For each product, find how many unique customers have purchased it.

```
SELECT p.product_id, p.product_name, COUNT(DISTINCT o.customer_id) AS unique_customer_count
FROM Products p
JOIN Order_Details od ON p.product_id = od.product_id
JOIN Orders o ON od.order_id = o.order_id
GROUP BY p.product_id;
```