task-3

1.Customer-table

create database task-3

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

email VARCHAR(100),

phone VARCHAR(20),

address TEXT,

join\_date DATE

);

INSERT INTO Customers (customer\_id, first\_name, last\_name, email, phone, address, join\_date)

VALUES

(1, 'Rahul', 'Sharma', 'rahul@gmail.com', '9876543210', '123, MG Road, Bangalore', '2023-05-10'),

(2, 'Priya', 'Verma', 'priya@gmail.com', '9876543211', '45, Kothrud, Pune', '2022-08-21'),

(3, 'Amit', 'Singh', 'amit@gmail.com', '9876543212', '12, Sector 22, Gurgaon', '2021-11-15'),

(4, 'Neha', 'Patel', 'neha@gmail.com', '9876543213', '23, Banjara Hills, Hyderabad', '2020-01-01'),

(5, 'Karan', 'Yadav', 'karan@gmail.com', '9876543214', '5, Juhu, Mumbai', '2023-07-12'),

(6, 'Sonal', 'Gupta', 'sonal@gmail.com', '9876543215', '6, C-Block, Delhi', '2021-02-19'),

(7, 'Vikram', 'Reddy', 'vikram@gmail.com', '9876543216', '8, Sadar Bazar, Jaipur', '2023-03-25'),

(8, 'Ananya', 'Kumar', 'ananya@gmail.com', '9876543217', '9, Kankarbagh, Patna', '2020-05-30'),

(9, 'Ravi', 'Jadhav', 'ravi@gmail.com', '9876543218', '19, New Vasai, Mumbai', '2022-12-10'),

(10, 'Pooja', 'Desai', 'pooja@gmail.com', '9876543219', '20, Salt Lake, Kolkata', '2021-06-15');

=====================================================================================================================

2.Product-Table

CREATE TABLE Products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

category VARCHAR(50),

price DECIMAL(10,2),

stock\_quantity INT

);

INSERT INTO Products (product\_id, product\_name, category, price, stock\_quantity)

VALUES

(1, 'iPhone 13', 'Electronics', 79999.99, 50),

(2, 'Samsung Galaxy S22', 'Electronics', 64999.99, 30),

(3, 'Sony Bravia TV', 'Electronics', 59999.99, 20),

(4, 'Dell Inspiron 15', 'Laptops', 45000.50, 15),

(5, 'HP Pavilion x360', 'Laptops', 60000.00, 25),

(6, 'Nike Running Shoes', 'Footwear', 2999.99, 100),

(7, 'Adidas T-shirt', 'Clothing', 1299.50, 200),

(8, 'Levi’s Jeans', 'Clothing', 1999.00, 150),

(9, 'Apple MacBook Pro', 'Laptops', 129999.00, 10),

(10, 'Canon EOS 80D', 'Cameras', 54999.99, 8);

================================================================================

3.Orders-table

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

total\_amount DECIMAL(10,2),

order\_status VARCHAR(20),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

INSERT INTO Orders (order\_id, customer\_id, order\_date, total\_amount, order\_status)

VALUES

(1, 1, '2023-06-01', 159999.99, 'Pending'),

(2, 2, '2023-07-15', 64999.99, 'Shipped'),

(3, 3, '2023-08-05', 87999.99, 'Pending'),

(4, 4, '2023-09-10', 90000.00, 'Completed'),

(5, 5, '2023-10-20', 1999.99, 'Shipped'),

(6, 6, '2023-11-11', 79999.00, 'Cancelled'),

(7, 7, '2023-12-05', 129999.99, 'Completed'),

(8, 8, '2023-12-10', 2999.50, 'Pending'),

(9, 9, '2023-12-25', 59999.00, 'Shipped'),

(10, 10, '2023-11-30', 54999.99, 'Shipped');

===============================================================================================

4.OrderDetails-table

CREATE TABLE OrderDetails (

order\_detail\_id INT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT,

unit\_price DECIMAL(10,2),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

INSERT INTO OrderDetails (order\_detail\_id, order\_id, product\_id, quantity, unit\_price)

VALUES

(1, 1, 1, 1, 79999.99),

(2, 1, 3, 1, 59999.99),

(3, 2, 2, 1, 64999.99),

(4, 3, 4, 1, 45000.50),

(5, 4, 9, 1, 129999.00),

(6, 5, 7, 1, 1299.50),

(7, 6, 6, 1, 2999.99),

(8, 7, 10, 1, 54999.99),

(9, 8, 8, 2, 1999.00),

(10, 9, 5, 1, 60000.00);

============================================================================================

5.Payments-table

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY,

order\_id INT,

payment\_date DATE,

payment\_amount DECIMAL(10,2),

payment\_method VARCHAR(20),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id)

);

INSERT INTO Payments (payment\_id, order\_id, payment\_date, payment\_amount, payment\_method)

VALUES

(1, 1, '2023-06-02', 159999.99, 'Credit Card'),

(2, 2, '2023-07-16', 64999.99, 'PayPal'),

(3, 3, '2023-08-06', 87999.99, 'Credit Card'),

(4, 4, '2023-09-11', 90000.00, 'Cash'),

(5, 5, '2023-10-21', 1999.99, 'Credit Card'),

(6, 6, '2023-11-12', 79999.00, 'PayPal'),

(7, 7, '2023-12-06', 129999.99, 'Credit Card'),

(8, 8, '2023-12-11', 2999.50, 'Cash'),

(9, 9, '2023-12-26', 59999.00, 'Credit Card'),

(10, 10, '2023-12-01', 54999.99, 'PayPal');

================================================================================

SQL Queries for the Case Study

1. Find the Total Number of Orders for Each Customer

SELECT customer\_id, COUNT(order\_id) AS total\_orders

FROM Orders

GROUP BY customer\_id;

==================================================================================

2. Find the Total Sales Amount for Each Product (Revenue per Product)

SELECT p.product\_id, p.product\_name, SUM(od.quantity \* od.unit\_price) AS total\_sales

FROM OrderDetails od

JOIN Products p ON od.product\_id = p.product\_id

GROUP BY p.product\_id, p.product\_name;

==============================================================================

3. Find the Most Expensive Product Sold

SELECT p.product\_id, p.product\_name, (od.quantity \* od.unit\_price) AS total\_revenue

FROM OrderDetails od

JOIN Products p ON od.product\_id = p.product\_id

ORDER BY total\_revenue DESC

LIMIT 1;

===============================================================================

4. Get the List of Customers Who Have Placed Orders in the Last 30 Days

SELECT DISTINCT c.customer\_id, c.first\_name, c.last\_name, c.email, c.phone

FROM Orders o

JOIN Customers c ON o.customer\_id = c.customer\_id

WHERE o.order\_date >= CURDATE() - INTERVAL 30 DAY;

=============================================================================

5. Calculate the Total Amount Paid by Each Customer

SELECT o.customer\_id, SUM(p.payment\_amount) AS total\_paid

FROM Payments p

JOIN Orders o ON p.order\_id = o.order\_id

GROUP BY o.customer\_id;

==============================================================================

6. Get the Number of Products Sold by Category

SELECT p.category, SUM(od.quantity) AS total\_products\_sold

FROM OrderDetails od

JOIN Products p ON od.product\_id = p.product\_id

GROUP BY p.category;

===================================================================================

7. List All Orders That Are Pending (i.e., Orders that haven't been shipped yet)

SELECT o.order\_id, o.customer\_id, c.first\_name, c.last\_name, o.order\_date, o.order\_status

FROM Orders o

JOIN Customers c ON o.customer\_id = c.customer\_id

WHERE o.order\_status = 'Pending';

=====================================================================================

8. Find the Average Order Value (Total Order Amount / Number of Orders)

SELECT SUM(o.total\_amount) / COUNT(o.order\_id) AS average\_order\_value

FROM Orders o;

====================================================================================

9. List the Top 5 Customers Who Have Spent the Most Money

SELECT o.customer\_id, c.first\_name, c.last\_name, SUM(p.payment\_amount) AS total\_spent

FROM Payments p

JOIN Orders o ON p.order\_id = o.order\_id

JOIN Customers c ON o.customer\_id = c.customer\_id

GROUP BY o.customer\_id

ORDER BY total\_spent DESC

LIMIT 5;

=====================================================================================

10. Find the Products That Have Never Been Sold

SELECT p.product\_id, p.product\_name

FROM Products p

LEFT JOIN OrderDetails od ON p.product\_id = od.product\_id

WHERE od.product\_id IS NULL;