CODING CHALLENGE

Name: ANIROOP GUPTA

Q) Create Database and Create tables and Insert the data into it.

```
-- Create Customers table
CREATE DATABASE StoreDB;

-- Use Database
USE StoreDB;

-- Create Customers table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    City VARCHAR(50)
);

-- Create Orders table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    Amount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Insert data into Customers
INSERT INTO Customers (CustomerID, CustomerName, City)
VALUES
(1, 'Aniroop', 'Kanpur'),
(2, 'Akash', 'Pune'),
(3, 'Subrat', 'Jaipur'),
(4, 'Harish', 'Goa'),
(5, 'Shreyansh', 'Delhi'),
(6, 'Anjum', 'Noida'),
(7, 'Nahin', 'Kashmir'),
(8, 'Savi', 'Patna'),
(9, 'Prajjwal', 'Merrut'),
(10, 'Ambati', 'Chennai');

Select * from Customers;
```

	CustomerID	CustomerName	City
1	1	Aniroop	Kanpur
2	2	Akash	Pune
3	3	Subrat	Jaipur
4	4	Harish	Goa
5	5	Shreyansh	Delhi
6	6	Anjum	Noida
7	7	Nahin	Kashmir
8	8	Savi	Patna
9	9	Prajjwal	Merrut
10	10	Ambati	Chennai

```sql
-- Insert data into Orders
INSERT INTO Orders (OrderID, CustomerID, OrderDate, Amount)
VALUES
(101, 1, '2024-01-15', 150.00),
(102, 2, '2024-02-20', 200.00),
(103, 1, '2024-03-10', 300.00),
(104, 3, '2024-03-15', 120.00),
(105, 3, '2024-04-05', 180.00),
(106, 4, '2024-05-10', 220.00),
(107, 5, '2024-05-20', 90.00),
(108, 6, '2024-06-15', 400.00),
(109, 7, '2024-06-20', 250.00),
(110, 8, '2024-07-10', 180.00);

Select * from Orders;
```

	OrderID	CustomerID	OrderDate	Amount
1	101	1	2024-01-15	150.00
2	102	2	2024-02-20	200.00
3	103	1	2024-03-10	300.00
4	104	3	2024-03-15	120.00
5	105	3	2024-04-05	180.00
6	106	4	2024-05-10	220.00
7	107	5	2024-05-20	90.00
8	108	6	2024-06-15	400.00
9	109	7	2024-06-20	250.00
10	110	8	2024-07-10	180.00

1. Querying Data by Using Joins and Subqueries and subtotal.

- **JOINS:** Combine rows from two or more tables based on a related column.
- **SUBQUERIES:** A query inside another query to provide intermediate results.
- **SUBTOTAL:** Calculates a sum, average, or other aggregate of specific data within a group.

1) Find the `OrderID`, `CustomerName`, and `Amount` for each order.

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.Amount
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

	OrderID	CustomerName	Amount
1	101	Aniroop	150.00
2	102	Akash	200.00
3	103	Aniroop	300.00
4	104	Subrat	120.00
5	105	Subrat	180.00
6	106	Harish	220.00
7	107	Shreyansh	90.00
8	108	Anjum	400.00
9	109	Nahin	250.00
10	110	Savi	180.00

2) Find all customers and their orders, including customers without orders.

```
SELECT Customers.CustomerName, Orders.OrderID, Orders.Amount
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

	CustomerName	OrderID	Amount
1	Aniroop	101	150.00
2	Aniroop	103	300.00
3	Akash	102	200.00
4	Subrat	104	120.00
5	Subrat	105	180.00
6	Harish	106	220.00
7	Shreyansh	107	90.00
8	Anjum	108	400.00
9	Nahin	109	250.00
10	Savi	110	180.00
11	Prajjwal	NULL	NULL
12	Ambati	NULL	NULL

3) List the customers who have placed at least one order with an amount greater than $150.

```
SELECT CustomerName
FROM Customers
WHERE CustomerID IN (
    SELECT CustomerID
    FROM Orders
    WHERE Amount > 150
```

```
);
```

	CustomerName
1	Aniroop
2	Akash
3	Subrat
4	Harish
5	Anjum
6	Nahin
7	Savi

4) Find the total amount of orders placed in the first quarter of 2024

```
SELECT SUM(Amount) AS TotalQ1Amount
FROM Orders
WHERE OrderDate BETWEEN '2024-01-01' AND '2024-03-31';
```

	TotalQ1Amount
1	770.00

5) Retrieve the total amount each customer has spent, but only for those customers who have at least one order greater than $150.

```
SELECT Customers.CustomerID, Customers.CustomerName, SUM(Orders.Amount) AS TotalAmount
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
WHERE Customers.CustomerID IN (
    SELECT CustomerID
    FROM Orders
    WHERE Amount > 150
)
GROUP BY Customers.CustomerID, Customers.CustomerName;
```

	CustomerID	CustomerName	TotalAmount
1	1	Aniroop	450.00
2	2	Akash	200.00
3	3	Subrat	300.00
4	4	Harish	220.00
5	6	Anjum	400.00
6	7	Nahin	250.00
7	8	Savi	180.00

2. Manipulating Data by Using GROUP BY and HAVING Clauses

• **GROUP BY:** Groups rows with the same value in a specified column to perform aggregate calculations.

• **HAVING Clause:** Filters groups created by `GROUP BY` based on aggregate conditions (like totals or counts).

1) Find Total Amount per Customer

```
SELECT CustomerID, SUM(Amount) AS TotalAmount
FROM Orders
GROUP BY CustomerID;
```

	CustomerID	TotalAmount
1	1	450.00
2	2	200.00
3	3	300.00
4	4	220.00
5	5	90.00
6	6	400.00
7	7	250.00
8	8	180.00

2) Find Customers with Total Orders Over $250

```
SELECT CustomerID, SUM(Amount) AS TotalAmount
FROM Orders
GROUP BY CustomerID
HAVING SUM(Amount) > 250;
```

	CustomerID	TotalAmount
1	1	450.00
2	3	300.00
3	6	400.00

ER DIAGRAM