**#Task-4**

-- Create Tables

CREATE TABLE Restaurants (

RestaurantID INT PRIMARY KEY,

Name VARCHAR(100),

City VARCHAR(50),

CuisineType VARCHAR(50),

Rating DECIMAL(2, 1),

AverageCostForTwo DECIMAL(10, 2)

);

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Phone VARCHAR(15),

City VARCHAR(50),

JoinDate DATE

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

RestaurantID INT,

OrderDate DATE,

OrderAmount DECIMAL(10, 2),

OrderStatus VARCHAR(20),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)

);

CREATE TABLE Reviews (

ReviewID INT PRIMARY KEY,

CustomerID INT,

RestaurantID INT,

Rating DECIMAL(2, 1),

Comment TEXT,

ReviewDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)

);

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY,

OrderID INT,

PaymentMethod VARCHAR(20),

Amount DECIMAL(10, 2),

PaymentDate DATE,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

-- Insert Sample Data

INSERT INTO Restaurants VALUES

(1, 'Spicy Delight', 'Mumbai', 'Indian', 4.7, 800),

(2, 'Pasta Paradise', 'Mumbai', 'Italian', 4.5, 1200),

(3, 'Dragon Bowl', 'Delhi', 'Chinese', 4.3, 700),

(4, 'Tandoori Flames', 'Mumbai', 'Indian', 4.6, 900),

(5, 'Sushi Spot', 'Bangalore', 'Japanese', 4.8, 1500),

(6, 'Pizza Haven', 'Mumbai', 'Italian', 4.2, 1000),

(7, 'Curry House', 'Delhi', 'Indian', 4.4, 850),

(8, 'Burger Bliss', 'Chennai', 'American', 4.1, 500),

(9, 'Noodle Nest', 'Mumbai', 'Chinese', 4.6, 750),

(10, 'Healthy Bites', 'Pune', 'Vegan', 4.9, 1300);

INSERT INTO Customers VALUES

(1, 'John', 'Doe', '9876543210', 'Mumbai', '2023-01-15'),

(2, 'Jane', 'Smith', '8765432109', 'Delhi', '2023-03-10'),

(3, 'Amit', 'Shah', '7654321098', 'Mumbai', '2023-05-20'),

(4, 'Rita', 'Kumar', '6543210987', 'Bangalore', '2023-07-25'),

(5, 'Raj', 'Patel', '5432109876', 'Pune', '2023-09-01');

INSERT INTO Orders VALUES

(1, 1, 1, '2023-11-01', 1200, 'Delivered'),

(2, 2, 3, '2023-11-05', 800, 'Cancelled'),

(3, 3, 2, '2023-11-10', 1500, 'Delivered'),

(4, 4, 5, '2023-11-15', 2000, 'Pending'),

(5, 5, 10, '2023-11-20', 1300, 'Delivered');

INSERT INTO Reviews VALUES

(1, 1, 1, 5.0, 'Excellent food!', '2023-11-02'),

(2, 3, 2, 4.5, 'Great pasta.', '2023-11-11'),

(3, 5, 10, 4.8, 'Loved the healthy options.', '2023-11-21');

INSERT INTO Payments VALUES

(1, 1, 'Card', 1200, '2023-11-01'),

(2, 3, 'Wallet', 1500, '2023-11-10'),

(3, 5, 'Cash', 1300, '2023-11-20');

-- Assignment Queries

-- 1. Retrieve the names and locations of restaurants with a rating of 4.5 or higher.

SELECT Name, City FROM Restaurants WHERE Rating >= 4.5;

-- 2. Find the total number of orders placed by each customer.

SELECT CustomerID, COUNT(OrderID) AS TotalOrders FROM Orders GROUP BY CustomerID;

-- 3. List all restaurants offering "Italian" cuisine in "Mumbai".

SELECT Name FROM Restaurants WHERE CuisineType = 'Italian' AND City = 'Mumbai';

-- 4. Calculate the total revenue generated by each restaurant from completed orders.

SELECT r.Name, SUM(o.OrderAmount) AS TotalRevenue

FROM Orders o

JOIN Restaurants r ON o.RestaurantID = r.RestaurantID

WHERE o.OrderStatus = 'Delivered'

GROUP BY r.Name;

-- 5. Retrieve the most recent order placed by each customer.

SELECT CustomerID, MAX(OrderDate) AS MostRecentOrder FROM Orders GROUP BY CustomerID;

-- 6. List customers who have not placed any orders yet.

SELECT c.FirstName, c.LastName

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE o.OrderID IS NULL;

-- 7. Identify the most reviewed restaurants.

SELECT r.Name, COUNT(rv.ReviewID) AS TotalReviews

FROM Reviews rv

JOIN Restaurants r ON rv.RestaurantID = r.RestaurantID

GROUP BY r.Name

ORDER BY TotalReviews DESC;

-- 8. Find the most preferred payment method.

SELECT PaymentMethod, COUNT(PaymentID) AS UsageCount

FROM Payments

GROUP BY PaymentMethod

ORDER BY UsageCount DESC

LIMIT 1;

-- 9. List the top 5 restaurants by total revenue.

SELECT r.Name, SUM(o.OrderAmount) AS TotalRevenue

FROM Orders o

JOIN Restaurants r ON o.RestaurantID = r.RestaurantID

WHERE o.OrderStatus = 'Delivered'

GROUP BY r.Name

ORDER BY TotalRevenue DESC

LIMIT 5;

-- 10. Show the details of all cancelled orders along with the customer's and restaurant's names.

SELECT o.OrderID, c.FirstName, c.LastName, r.Name AS RestaurantName, o.OrderAmount, o.OrderDate

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Restaurants r ON o.RestaurantID = r.RestaurantID

WHERE o.OrderStatus = 'Cancelled';