Restaurant Management System (RMS).

Name:Siddarth S
Date: 30-07-2024

Project Module 1

1) Use Case Specification Template: Populate the template with your project details and expected functionalities.

2) Create DDL (Data Definition Language): Define the database schema.

3) Create ER Diagram (Entity-Relationship Diagram): Visual representation of the database.

4) Extract Data: Example queries for data extraction.

5) Prepare and Upload Document: Compile everything into a document and upload it to GitHub.

---

1) Use Case Specification :

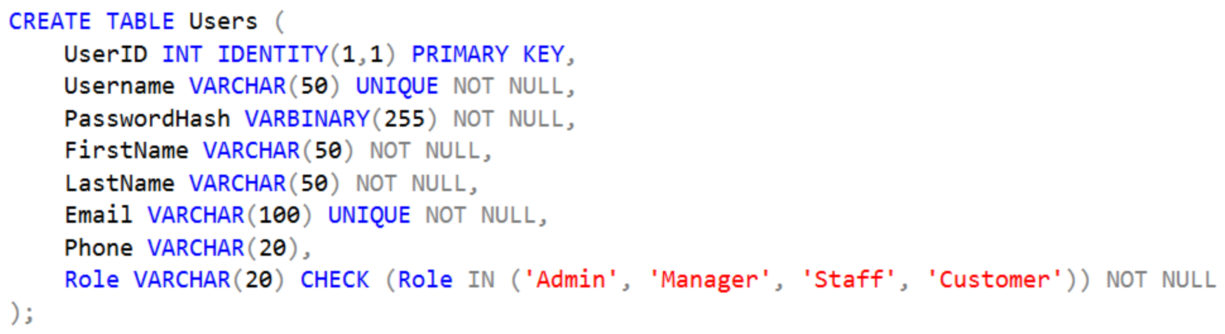
This is attached in a separate Excel file.

2) Create DDL (Data Definition Language):

Table 1 : Users

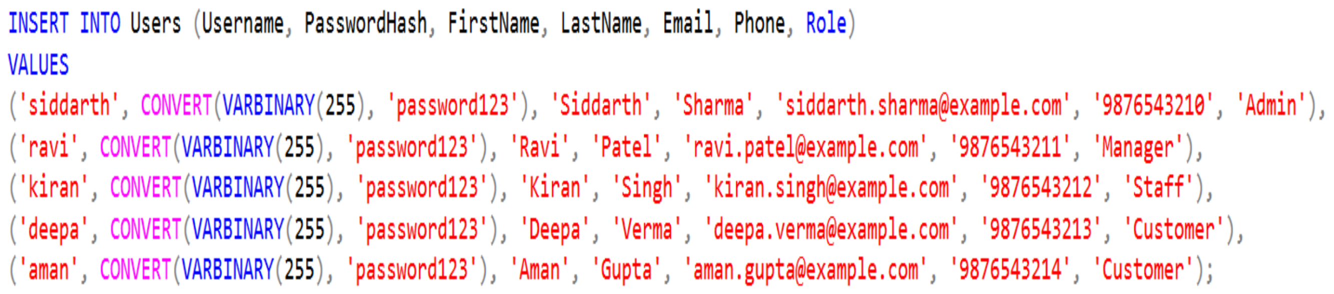
Purpose: This table stores user information, including login credentials and contact details. It is used to manage different user roles such as Admin, Manager, Staff, and Customer.

Create Table Users:



```
CREATE TABLE Users (
    UserID INT IDENTITY(1,1) PRIMARY KEY,
    Username VARCHAR(50) UNIQUE NOT NULL,
    PasswordHash VARBINARY(255) NOT NULL,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(20),
    Role VARCHAR(20) CHECK (Role IN ('Admin', 'Manager', 'Staff', 'Customer')) NOT NULL
);
```

Insert Some values in Table Users :



```
INSERT INTO Users (Username, PasswordHash, FirstName, LastName, Email, Phone, Role)
VALUES
('siddarth', CONVERT(VARBINARY(255), 'password123'), 'Siddarth', 'Sharma', 'siddarth.sharma@example.com', '9876543210', 'Admin'),
('ravi', CONVERT(VARBINARY(255), 'password123'), 'Ravi', 'Patel', 'ravi.patel@example.com', '9876543211', 'Manager'),
('kiran', CONVERT(VARBINARY(255), 'password123'), 'Kiran', 'Singh', 'kiran.singh@example.com', '9876543212', 'Staff'),
('deepa', CONVERT(VARBINARY(255), 'password123'), 'Deepa', 'Verma', 'deepa.verma@example.com', '9876543213', 'Customer'),
('aman', CONVERT(VARBINARY(255), 'password123'), 'Aman', 'Gupta', 'aman.gupta@example.com', '9876543214', 'Customer');
```

Table 2 : MenuItem

Purpose: This table holds information about the menu items available in the restaurant, including their names, descriptions, prices, and availability status.

Create Table MenuItem:

```
CREATE TABLE MenuItems (
    MenuItemID INT IDENTITY(1,1) PRIMARY KEY,
    ItemName VARCHAR(100) NOT NULL,
    Description VARCHAR(255),
    Price DECIMAL(10, 2) NOT NULL,
    Available BIT NOT NULL
);
```

Insert Some Values in Table MenuItems:

```
INSERT INTO MenuItems (ItemName, Description, Price, Available)
VALUES
('Butter Chicken', 'Rich and creamy chicken curry', 250.00, 1),
('Paneer Butter Masala', 'Creamy tomato-based paneer curry', 200.00, 1),
('Chole Bhature', 'Spicy chickpeas served with fried bread', 150.00, 1),
('Masala Dosa', 'Crispy rice pancake with spicy potato filling', 120.00, 1),
('Biryani', 'Fragrant rice dish with meat or vegetables', 180.00, 1);
```

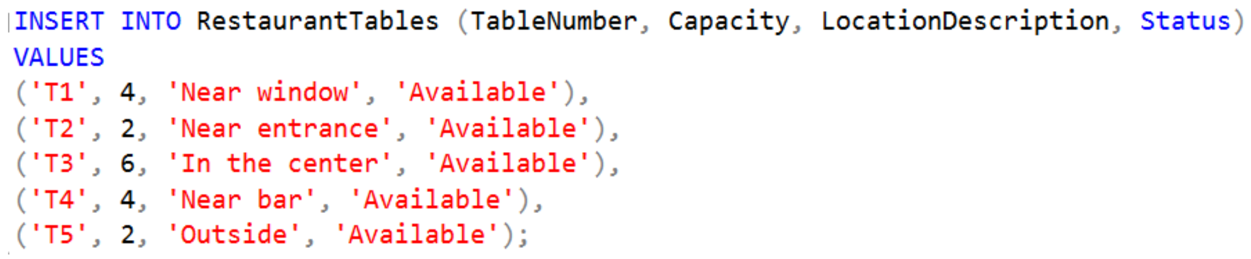
Table 3: RestaurantTables

Purpose: This table keeps track of the tables in the restaurant, their capacity, location, and current status (whether they are available, occupied, or reserved).

Create Table RestaurantTables :

```
CREATE TABLE RestaurantTables (
    TableID INT IDENTITY(1,1) PRIMARY KEY,
    TableNumber VARCHAR(10) UNIQUE NOT NULL,
    Capacity INT NOT NULL,
    LocationDescription VARCHAR(255),
    Status VARCHAR(20) CHECK (Status IN ('Available', 'Occupied', 'Reserved')) NOT NULL
);
```

Insert Some Values in Table RestaurantTables:

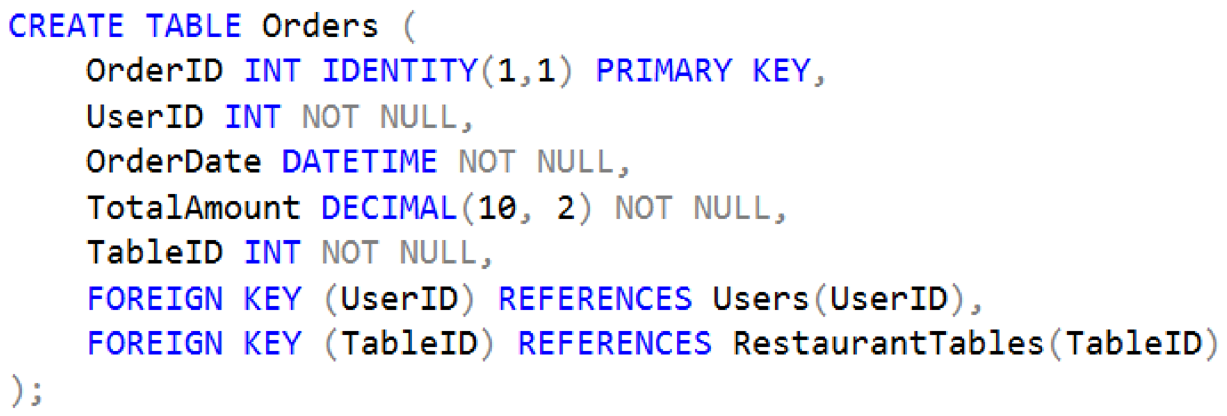


```
INSERT INTO RestaurantTables (TableNumber, Capacity, LocationDescription, Status)
VALUES
('T1', 4, 'Near window', 'Available'),
('T2', 2, 'Near entrance', 'Available'),
('T3', 6, 'In the center', 'Available'),
('T4', 4, 'Near bar', 'Available'),
('T5', 2, 'Outside', 'Available');
```

Table 4 :Orders

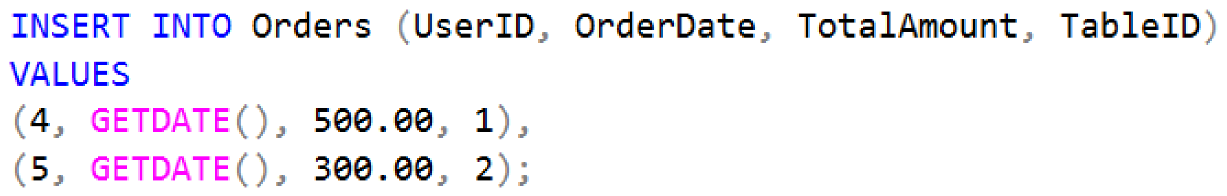
Purpose: This table records customer orders, including the user who placed the order, the order date, total amount, and the table associated with the order.

Create Table Orders:



```
CREATE TABLE Orders (
    OrderID INT IDENTITY(1,1) PRIMARY KEY,
    UserID INT NOT NULL,
    OrderDate DATETIME NOT NULL,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    TableID INT NOT NULL,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (TableID) REFERENCES RestaurantTables(TableID)
);
```

Insert Some Values in Table Orders:

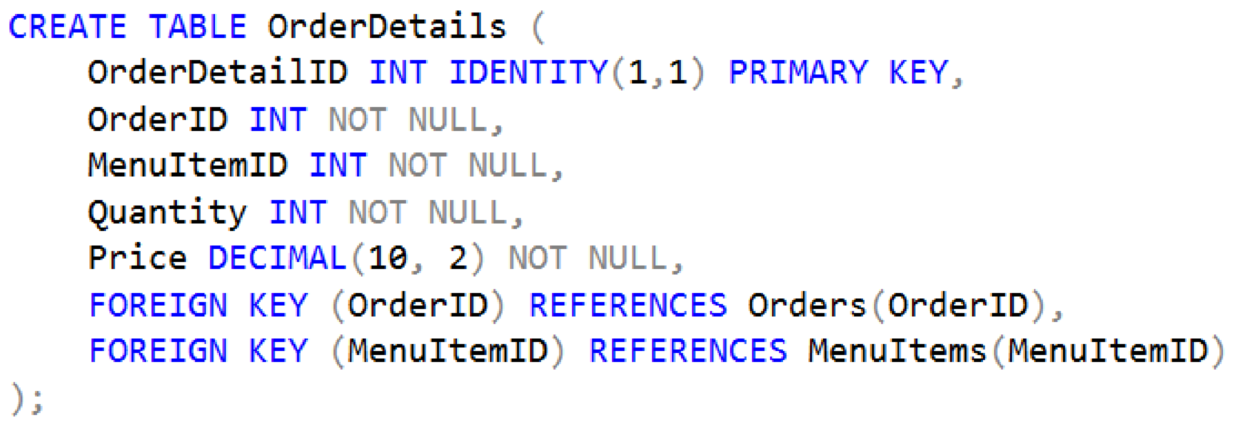


```
INSERT INTO Orders (UserID, OrderDate, TotalAmount, TableID)
VALUES
(4, GETDATE(), 500.00, 1),
(5, GETDATE(), 300.00, 2);
```

Table 5: OrderDetails

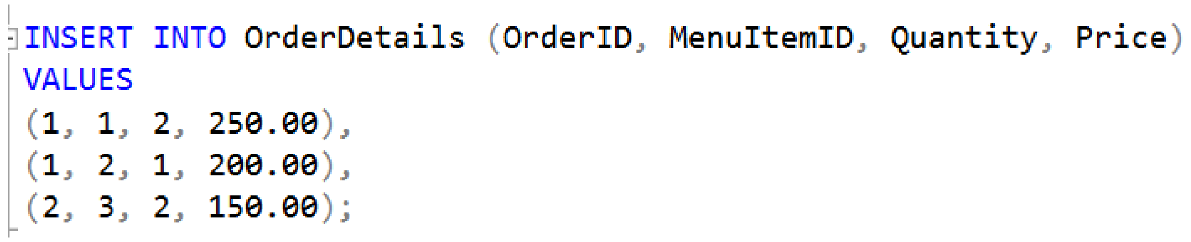
Purpose: This table provides detailed information about each item in an order, including the order ID, menu item ID, quantity, and price.

Create Table OrderDetails:



```sql
CREATE TABLE OrderDetails (
    OrderDetailID INT IDENTITY(1,1) PRIMARY KEY,
    OrderID INT NOT NULL,
    MenuItemID INT NOT NULL,
    Quantity INT NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (MenuItemID) REFERENCES MenuItems(MenuItemID)
);
```

Insert Some Values in Table OrderDetails:

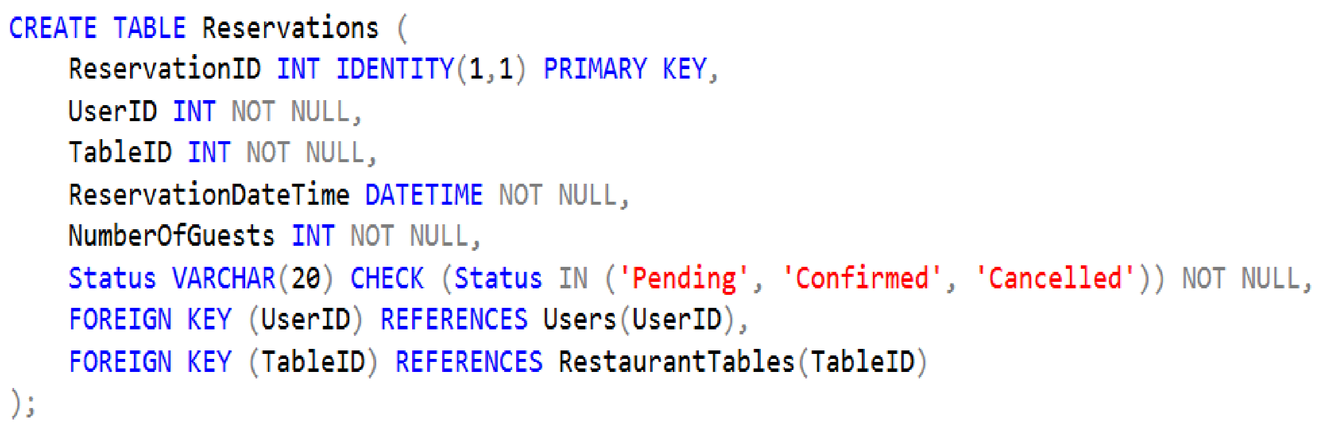


```sql
INSERT INTO OrderDetails (OrderID, MenuItemID, Quantity, Price)
VALUES
(1, 1, 2, 250.00),
(1, 2, 1, 200.00),
(2, 3, 2, 150.00);
```

Table 6: Reservation

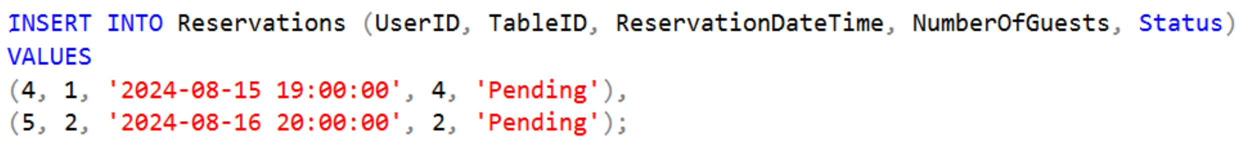
Create Table Reservation :

Purpose: This table manages reservations, including the user who made the reservation, the table reserved, the reservation date and time, the number of guests, and the reservation status (pending, confirmed, or canceled).



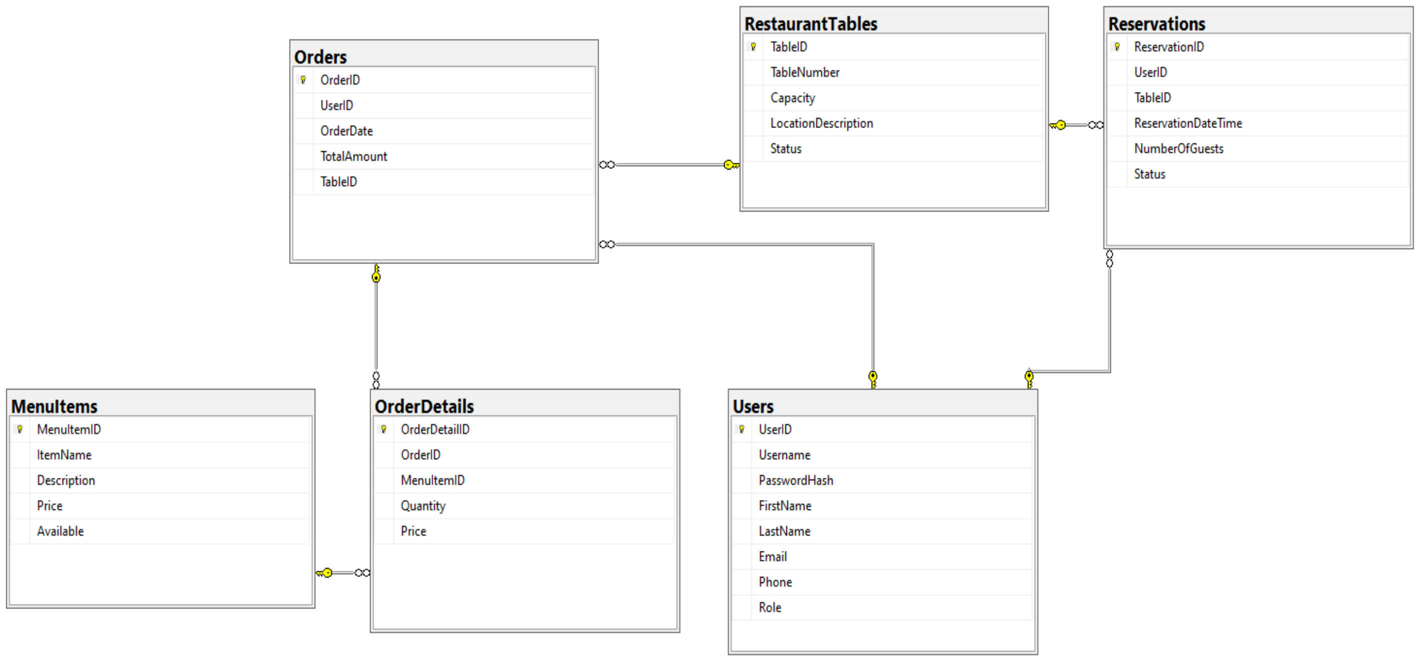
```
CREATE TABLE Reservations (
    ReservationID INT IDENTITY(1,1) PRIMARY KEY,
    UserID INT NOT NULL,
    TableID INT NOT NULL,
    ReservationDateTime DATETIME NOT NULL,
    NumberOfGuests INT NOT NULL,
    Status VARCHAR(20) CHECK (Status IN ('Pending', 'Confirmed', 'Cancelled')) NOT NULL,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (TableID) REFERENCES RestaurantTables(TableID)
);
```

Insert Some Values in Table Reservation :



```
INSERT INTO Reservations (UserID, TableID, ReservationDateTime, NumberOfGuests, Status)
VALUES
(4, 1, '2024-08-15 19:00:00', 4, 'Pending'),
(5, 2, '2024-08-16 20:00:00', 2, 'Pending');
```

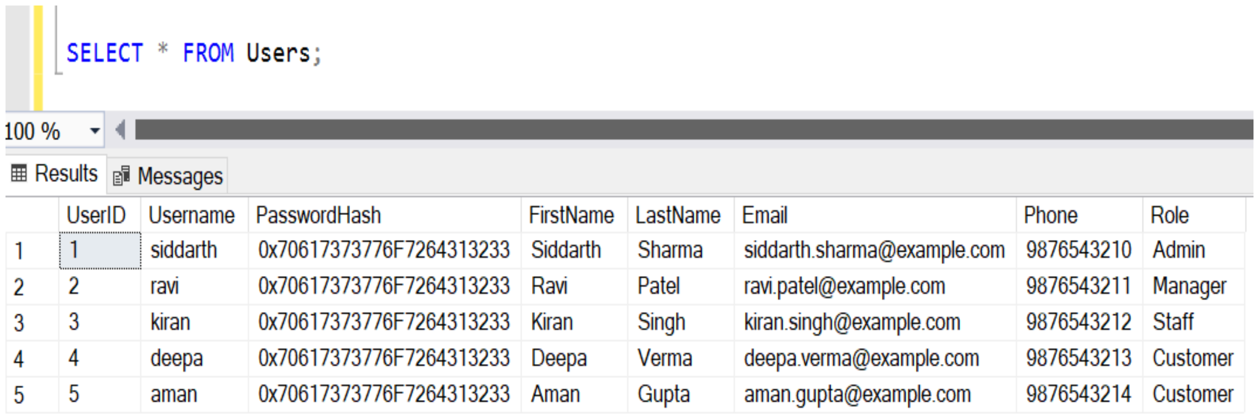
3) Create ER Diagram (Entity-Relationship Diagram):



4) Extract Data:

1.Retrieve All Records from a Table

- **Query Task:** Select all records from the `Users` table.
- **Query & Output:**

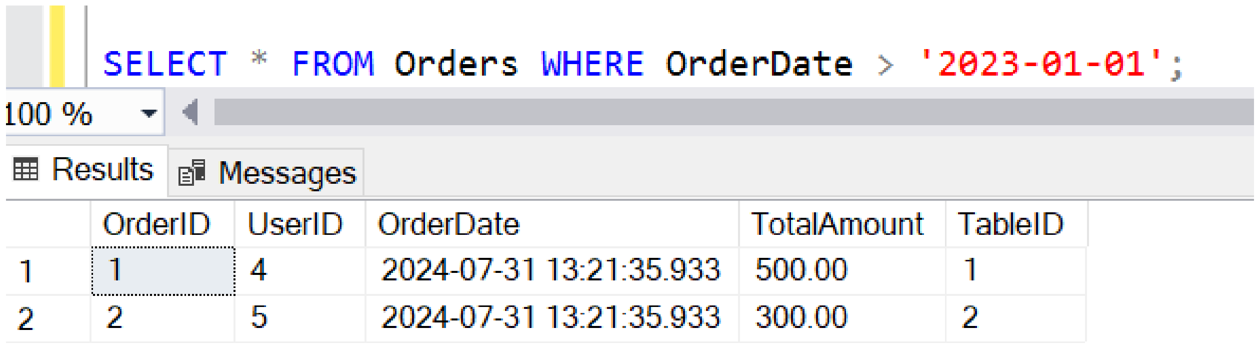


```
SELECT * FROM Users;
```

	UserID	Username	PasswordHash	FirstName	LastName	Email	Phone	Role
1	1	siddarth	0x70617373776F7264313233	Siddarth	Sharma	siddarth.sharma@example.com	9876543210	Admin
2	2	ravi	0x70617373776F7264313233	Ravi	Patel	ravi.patel@example.com	9876543211	Manager
3	3	kiran	0x70617373776F7264313233	Kiran	Singh	kiran.singh@example.com	9876543212	Staff
4	4	deepa	0x70617373776F7264313233	Deepa	Verma	deepa.verma@example.com	9876543213	Customer
5	5	aman	0x70617373776F7264313233	Aman	Gupta	aman.gupta@example.com	9876543214	Customer

2.Filter Records Based on a Condition

- **Query Task:**Select all orders from the Orders table where the order date is after January 1, 2023.
- **Query & Output:**



```
SELECT * FROM Orders WHERE OrderDate > '2023-01-01';
```

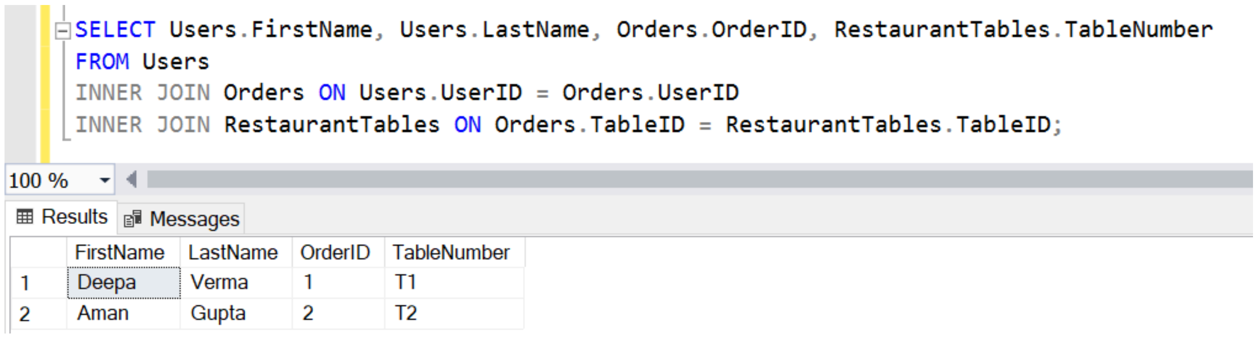
	OrderID	UserID	OrderDate	TotalAmount	TableID
1	1	4	2024-07-31 13:21:35.933	500.00	1
2	2	5	2024-07-31 13:21:35.933	300.00	2

3.Join Two Tables

Query Task: Retrieve the names of customers along with their order IDs and table numbers from the `Users`, `Orders`, and `RestaurantTables` tables.

Hint: Use an `INNER JOIN` to combine data from both tables based on a common column.

SQL Query and Output:



```
SELECT Users.FirstName, Users.LastName, Orders.OrderID, RestaurantTables.TableNumber
FROM Users
INNER JOIN Orders ON Users.UserID = Orders.UserID
INNER JOIN RestaurantTables ON Orders.TableID = RestaurantTables.TableID;
```

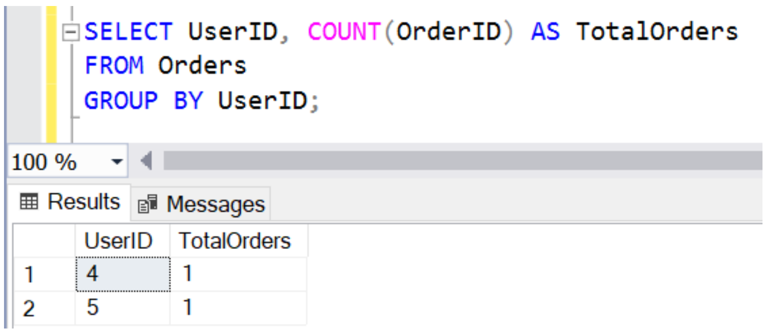
	FirstName	LastName	OrderID	TableNumber
1	Deepa	Verma	1	T1
2	Aman	Gupta	2	T2

4.Aggregate Data Using Group By

Query Task: Find the total number of orders placed by each customer.

Hint: Use the `GROUP BY` clause to group records and `COUNT` to aggregate.

SQL Query and Output:

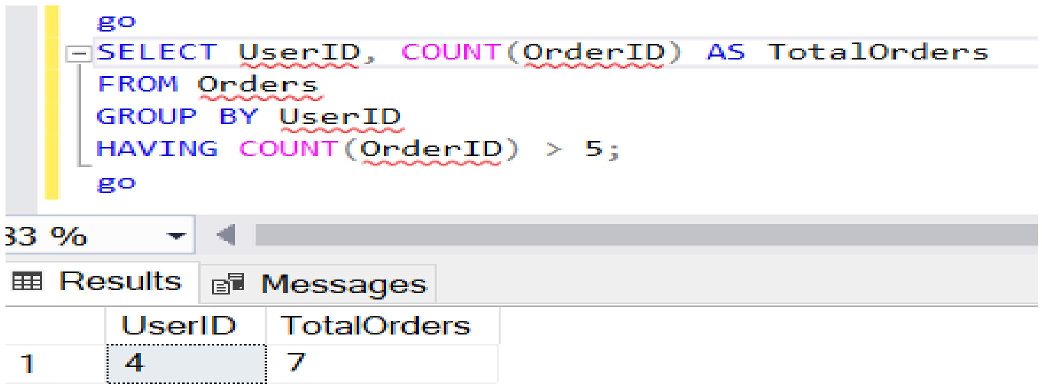


```
SELECT UserID, COUNT(OrderID) AS TotalOrders
FROM Orders
GROUP BY UserID;
```

	UserID	TotalOrders
1	4	1
2	5	1

5.Filter Groups Using HAVING

- **Query Task:**Query Task: Retrieve the customer IDs and their total number of orders, but only for customers who have placed more than 5 orders.
- **Query & Output:**



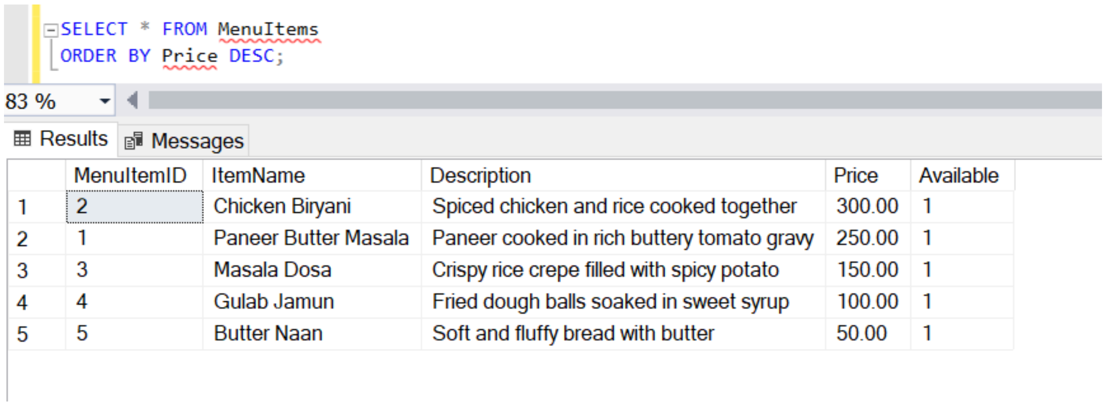
```
go
SELECT UserID, COUNT(OrderID) AS TotalOrders
FROM Orders
GROUP BY UserID
HAVING COUNT(OrderID) > 5;
go
```

	UserID	TotalOrders
1	4	7

6.Order Results Using ORDER BY

- **Query Task:** Select all menu items from the `MenuItems` table and order them by price in descending order.
- **Hint:** Use the `ORDER BY` clause to sort the results.

- **Query & Output:**



```
SELECT * FROM MenuItems
ORDER BY Price DESC;
```

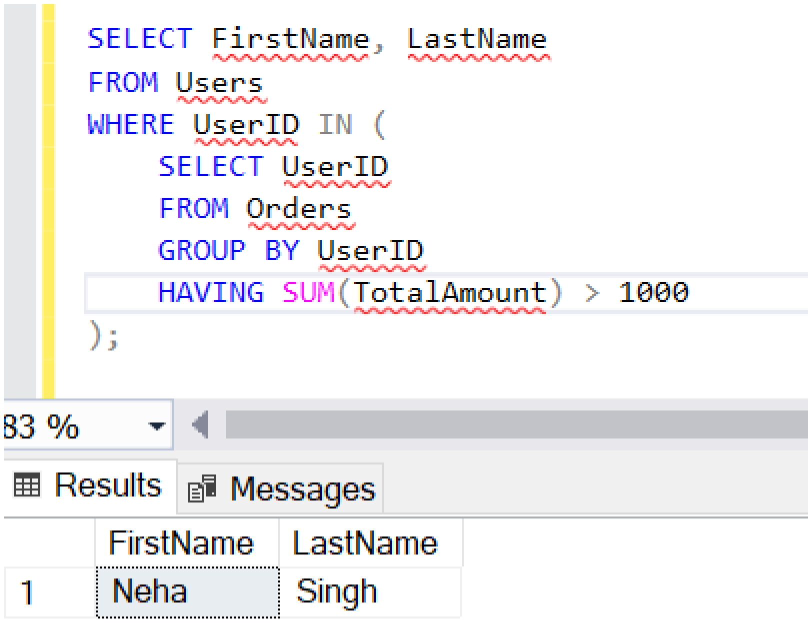
83 %

Results | Messages

	MenuItemID	ItemName	Description	Price	Available
1	2	Chicken Biryani	Spiced chicken and rice cooked together	300.00	1
2	1	Paneer Butter Masala	Paneer cooked in rich buttery tomato gravy	250.00	1
3	3	Masala Dosa	Crispy rice crepe filled with spicy potato	150.00	1
4	4	Gulab Jamun	Fried dough balls soaked in sweet syrup	100.00	1
5	5	Butter Naan	Soft and fluffy bread with butter	50.00	1

7.Retrieve Data with a Subquery

- **Query Task:** Find the names of customers who have placed orders with a total amount greater than Rs.1000.
- **Hint:** Use a subquery to calculate the total order amount for each customer.
- **Query & Output:**



```
SELECT FirstName, LastName
FROM Users
WHERE UserID IN (
    SELECT UserID
    FROM Orders
    GROUP BY UserID
    HAVING SUM(TotalAmount) > 1000
);
```

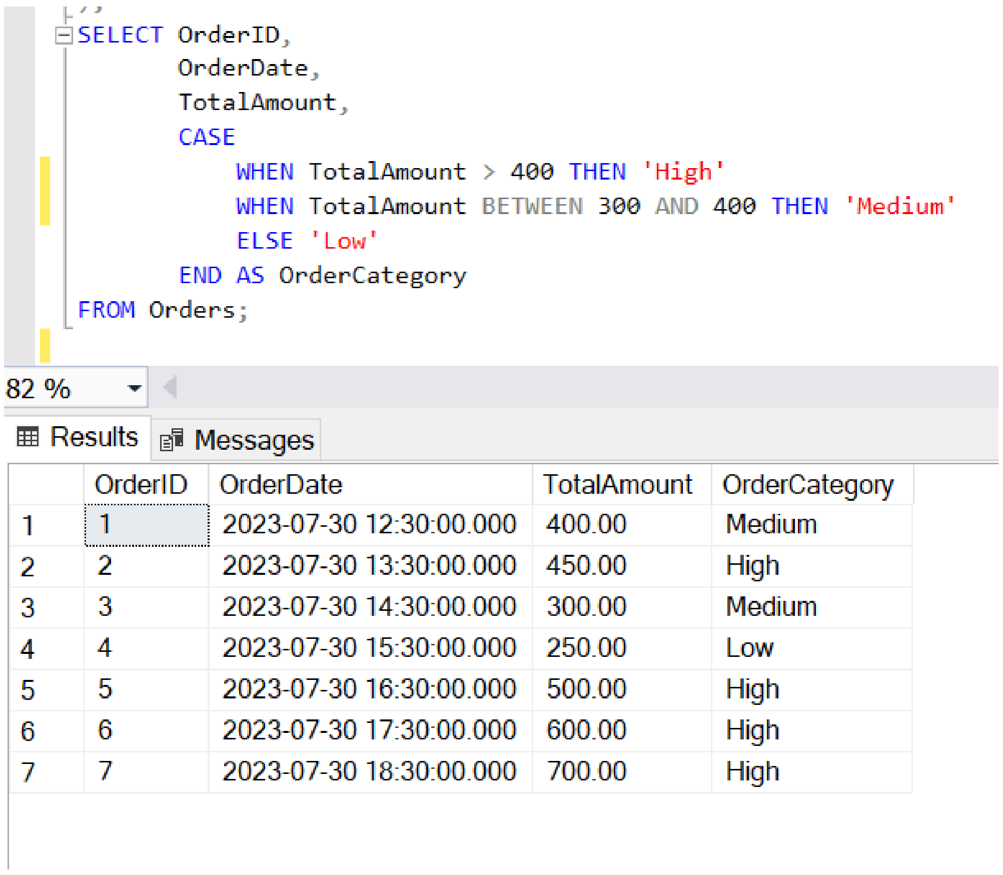
83 %

Results | Messages

	FirstName	LastName
1	Neha	Singh

8.Use CASE Statements

- **Query Task:** Categorize orders based on their total amount into 'High', 'Medium', and 'Low'.
- **Query & Output:**



```
SELECT OrderID,
       OrderDate,
       TotalAmount,
       CASE
           WHEN TotalAmount > 400 THEN 'High'
           WHEN TotalAmount BETWEEN 300 AND 400 THEN 'Medium'
           ELSE 'Low'
       END AS OrderCategory
FROM Orders;
```

	OrderID	OrderDate	TotalAmount	OrderCategory
1	1	2023-07-30 12:30:00.000	400.00	Medium
2	2	2023-07-30 13:30:00.000	450.00	High
3	3	2023-07-30 14:30:00.000	300.00	Medium
4	4	2023-07-30 15:30:00.000	250.00	Low
5	5	2023-07-30 16:30:00.000	500.00	High
6	6	2023-07-30 17:30:00.000	600.00	High
7	7	2023-07-30 18:30:00.000	700.00	High