DBMS Project

Food Ordering System

Team Members :

Name : Suraj Kumar Roll number: 23CS8071

Name: Madhusudan Das Roll number: 23CS8072

Name: Samriddha Halder Roll number: 23CS8073

Name : Aditya Gupta Roll number: 23CS8074

Name : Shubham Maity Roll number: 23CS8075

Introduction:

In this project, we have designed a database management system to organize and store information related to a food ordering system. This database contains details about customers, menu items, orders they place, and transaction records such as quantities and total prices. It also tracks order dates and categorizes menu items for easy reference. The primary objective of this project is to efficiently store and retrieve customer and order information. Our design facilitates convenient management of data by computerizing the ordering process, eliminating the need for manual entry and record-keeping systems. By implementing this system, we aim to streamline operations, enhance data accuracy, and provide a scalable solution for managing food orders effectively.

## Assumptions We Made for the ER Model :

1. **Keeping Orders Simple with One Table**: The introduction didn’t hint at splitting things up into separate tables for order summaries and item details, so we figured it’s easier to stick with one Order table, just like the schema shows. The OrderID acts like a handy tag that groups all the items a customer picks in one go—like a single receipt with everything listed together.
2. **No Extra Players Involved**: The intro keeps it straightforward, talking only about customers, menu items, and orders. So, we didn’t bring in any side characters like staff or payment stuff—it’s just the core trio we’re working with here, staying true to what the schema gives us.
3. **Figuring Out the TotalPrice**: We assumed the TotalPrice is just the Quantity times the Price for each item in an order. It lines up perfectly with the example data—like how 2 Burgers at $5.99 each come out to $11.98. Simple math that makes sense for a food order!
4. **Built for Ease and Efficiency**: The whole vibe of the introduction—focusing on "efficient storage and retrieval" and ditching manual work for a computerized setup—fits like a glove with the schema’s relational design. It’s all about making life easier with clear connections and solid rules to keep the data organized.

## Entity-Relationship Model :

## MenuItems

### Is Ordered

## Customers

## Orders

## Places

Creation Of Tables :

1.Table Customers

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

Contact VARCHAR(20)

);

INSERT INTO Customers (CustomerID, Name, Contact) VALUES

(1, 'Alice', '1234567890'),

(2, 'Bob', '9876543210'),

(3, 'Charlie', '5554443333'),

(4, 'Priya', '9198765432'),

(5, 'Rahul', '9187654321'),

(6, 'Aisha', '9176543210'),

(7, 'James', '4478901234'),

(8, 'Sophie', '8612345678'),

(9, 'Vikram Singh', '9190123456'),

(10, 'Joe Goldberg', '3467890123'),

(11, 'Anjali', '9178901234'),

(12, 'Thomas', '12025550123'),

(13, 'Neha Gupta', '9198761234'),

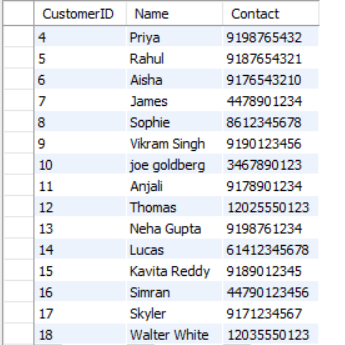
(14, 'Lucas', '61412345678'),

(15, 'Kavita Reddy', '9189012345'),

(16, 'Simran', '44790123456'),

(17, 'Skyler', '9171234567'),

(18, 'Walter White', '12035550123');



2.Table MenuItem

CREATE TABLE MenuItems (

ItemID INT PRIMARY KEY,

Name VARCHAR(100),

Price DECIMAL(10, 2),

Category VARCHAR(50)

);

INSERT INTO MenuItems VALUES

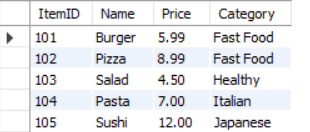
(101, 'Burger', 5.99, 'Fast Food'),

(102, 'Pizza', 8.99, 'Fast Food'),

(103, 'Salad', 4.50, 'Healthy'),

(104, 'Pasta', 7.00, 'Italian'),

(105, 'Sushi', 12.00, 'Japanese');



3.Table Orders

CREATE TABLE Orders (

OrderID INT,

CustomerID INT,

ItemID INT,

Quantity INT,

OrderDate DATE,

TotalPrice DECIMAL(10, 2),

PRIMARY KEY (OrderID, ItemID),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (ItemID) REFERENCES MenuItems(ItemID)

);

INSERT INTO Orders (OrderID, CustomerID, ItemID, Quantity, OrderDate, TotalPrice) VALUES

(1001, 1, 101, 2, '2025-03-29', 11.98),

(1001, 1, 102, 1, '2025-03-29', 8.99),

(1002, 2, 103, 3, '2025-04-01', 13.50),

(1003, 3, 104, 1, '2025-04-02', 7.00),

(1004, 1, 105, 2, '2025-04-03', 24.00),

(1005, 2, 101, 1, '2025-04-04', 5.99),

(1006, 2, 101, 2, '2025-04-05', 11.98),

(1007, 3, 103, 1, '2025-04-05', 4.50),

(1008, 3, 102, 1, '2025-04-06', 8.99),

(1009, 4, 101, 2, '2025-04-06', 11.98),

(1010, 4, 103, 1, '2025-04-06', 4.50),

(1011, 5, 102, 1, '2025-04-06', 8.99),

(1012, 5, 104, 2, '2025-04-06', 14.00),

(1013, 6, 105, 1, '2025-04-06', 12.00),

(1014, 6, 101, 1, '2025-04-06', 5.99),

(1015, 7, 102, 2, '2025-04-06', 17.98),

(1016, 7, 103, 1, '2025-04-06', 4.50),

(1017, 8, 105, 2, '2025-04-06', 24.00),

(1018, 8, 104, 1, '2025-04-06', 7.00),

(1019, 9, 101, 3, '2025-04-06', 17.97),

(1020, 9, 102, 1, '2025-04-06', 8.99),

(1021, 10, 103, 2, '2025-04-06', 9.00),

(1022, 10, 105, 1, '2025-04-06', 12.00),

(1023, 11, 104, 2, '2025-04-06', 14.00),

(1024, 11, 101, 1, '2025-04-06', 5.99),

(1025, 12, 102, 1, '2025-04-06', 8.99),

(1026, 12, 103, 3, '2025-04-06', 13.50),

(1027, 13, 105, 1, '2025-04-06', 12.00),

(1028, 13, 104, 1, '2025-04-06', 7.00),

(1029, 14, 101, 2, '2025-04-06', 11.98),

(1030, 14, 102, 1, '2025-04-06', 8.99),

(1031, 15, 103, 1, '2025-04-06', 4.50),

(1032, 15, 105, 2, '2025-04-06', 24.00),

(1033, 16, 104, 1, '2025-04-06', 7.00),

(1034, 16, 101, 1, '2025-04-06', 5.99),

(1035, 17, 102, 2, '2025-04-06', 17.98),

(1036, 17, 103, 1, '2025-04-06', 4.50),

(1037, 18, 105, 1, '2025-04-06', 12.00),

(1038, 18, 104, 2, '2025-04-06', 14.00),

(1039, 4, 102, 1, '2025-04-06', 8.99),

(1040, 5, 101, 1, '2025-04-06', 5.99),

(1041, 6, 103, 2, '2025-04-06', 9.00),

(1042, 7, 105, 1, '2025-04-06', 12.00),

(1043, 8, 102, 1, '2025-04-06', 8.99),

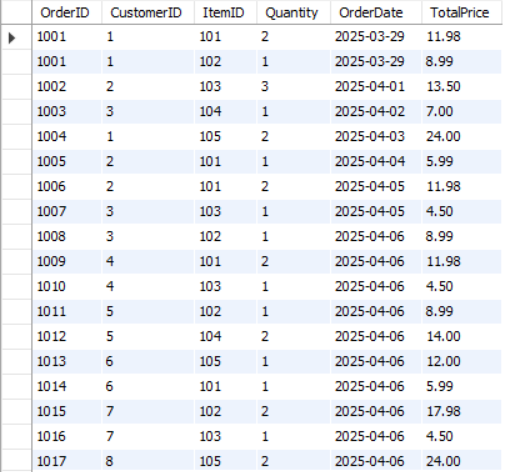
(1044, 9, 104, 1, '2025-04-06', 7.00),

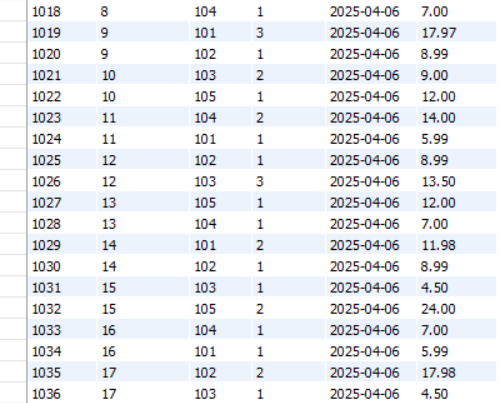
(1045, 10, 101, 2, '2025-04-06', 11.98),

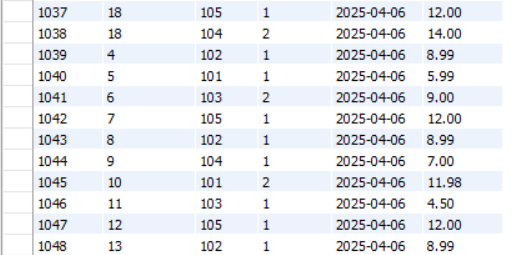
(1046, 11, 103, 1, '2025-04-06', 4.50),

(1047, 12, 105, 1, '2025-04-06', 12.00),

(1048, 13, 102, 1, '2025-04-06', 8.99);

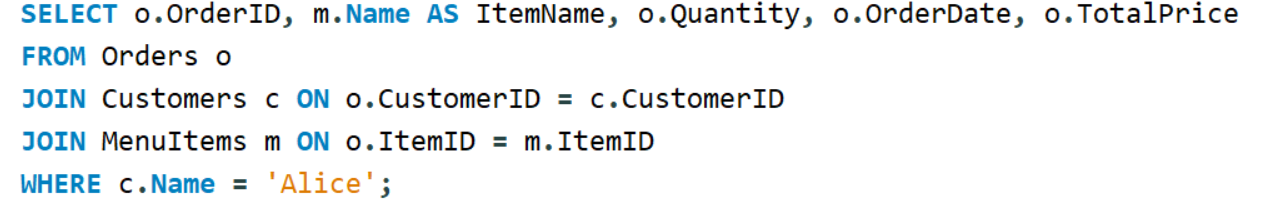


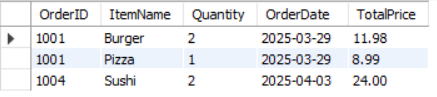




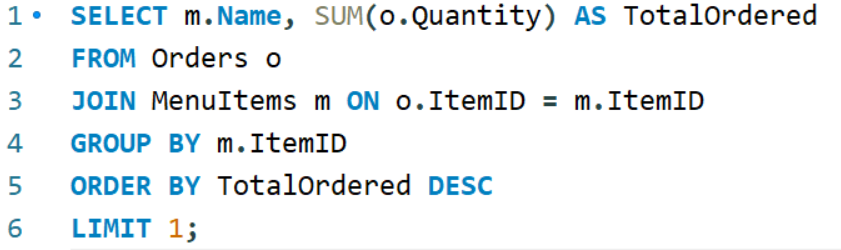
Queries :

Q1)Retrieve all orders placed by a specific customers



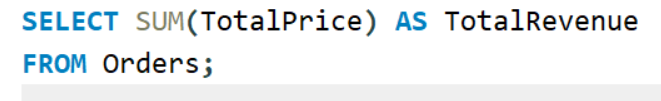


Q2) Find the most popular menu item



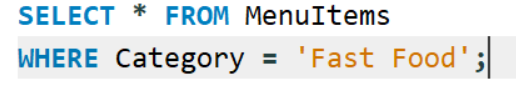


Q3)Calculate total revenue generated from orders



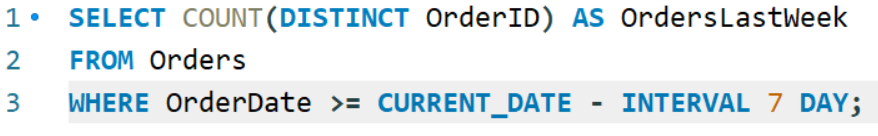


Q4)List all menu items in a specific category



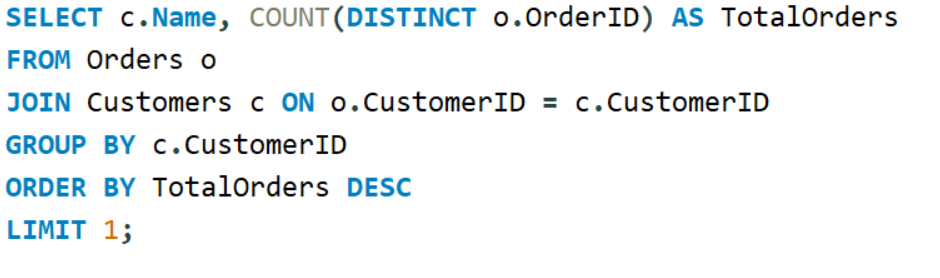


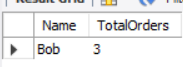
Q5)Count numbers of orders placed in the last week



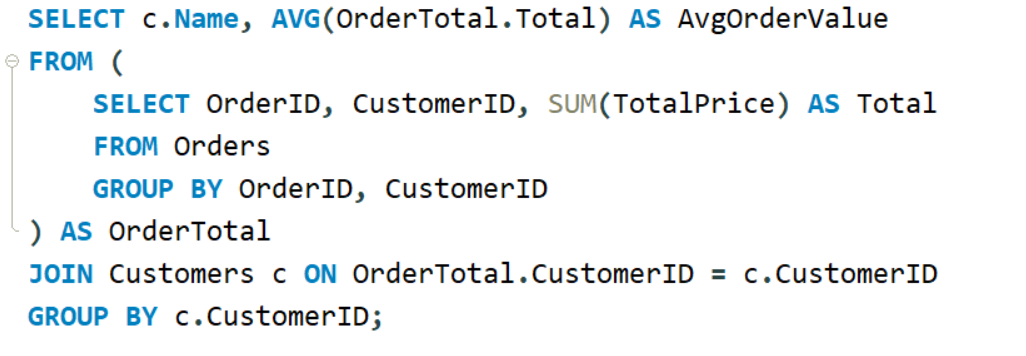


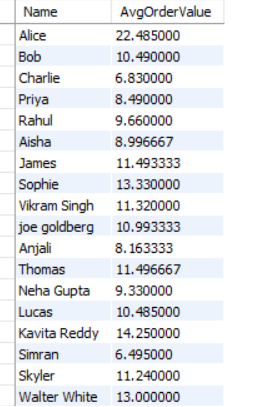
Q6) Identify customers who placed the highest number of orders



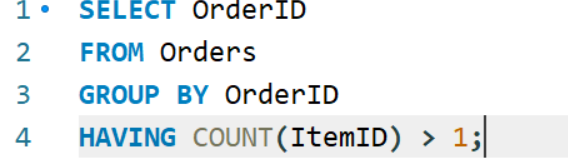


Q7)Find the average orders value per customers



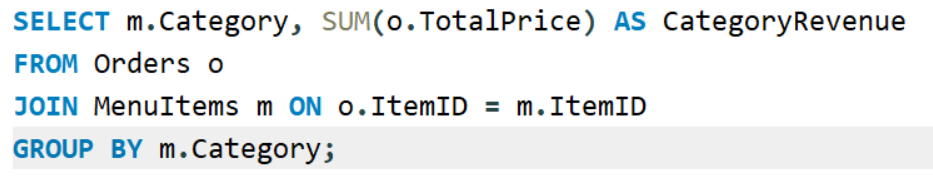


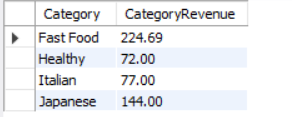
Q8)Retrieve orders that included multiple items



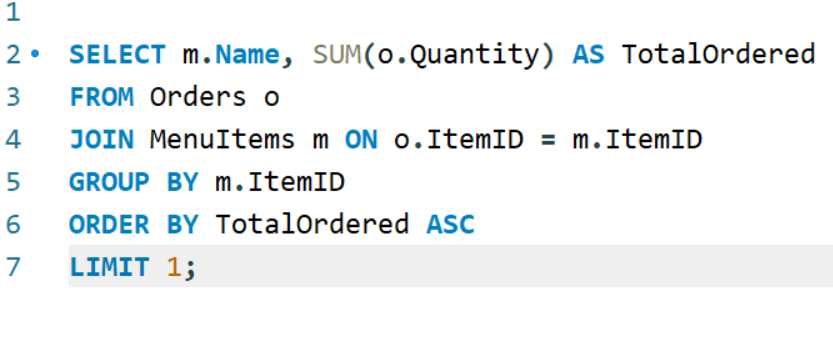


Q9)Get the total revenue for each menu category





Q10) identify the least ordered item





Relational Schema:

Customers Orders MenuItem

CustomersID(PK) OrderID(PK) ItemID(PK)

Name CustomerID Name

Contact ItemID(PK) Price

Quantity Contact

OrderDate

TotalPrice

Conclusion :

The Food Ordering System project successfully demonstrates the application of database management system (DBMS) principles to create an efficient and organized solution for managing food orders. By designing and implementing three interrelated tables—Customers, MenuItems, and Orders—we have established a robust relational database that effectively stores and links customer information, menu offerings, and order details. The Customers table captures essential user data, the MenuItems table provides a structured catalog of food options with prices and categories, and the Orders table ties it all together, tracking transactions with a composite primary key and foreign key constraints to ensure data integrity.