RESTAURANT MANAGEMENT SYSTEM

Ali Moallem

**Introduction**

The restaurant management system project aims to develop a comprehensive solution to address various operational challenges faced by restaurants. By efficiently managing inventory, customer relationships, orders, reservations, employees, financial transactions, and data analysis, the system seeks to enhance operational efficiency, improve customer satisfaction, and drive business growth.

The system will enable restaurant owners and managers to streamline processes, optimize resources, and make informed decisions to ensure smooth operations and deliver exceptional dining experiences to customers.

Key objectives of the project include:

1. Inventory Management: Keep track of available stock levels, monitor usage, and ensure timely replenishment of inventory items to prevent stockouts and wastage.

2. Customer Relationship Management (CRM): Capture customer information, preferences, and feedback to personalize service, tailor marketing efforts, and foster long-term customer loyalty.

3. Reservation Handling: Manage table reservations efficiently, optimize seating capacity, and ensure smooth coordination between front-of-house and kitchen operations to accommodate customer bookings effectively.

4. Employee Scheduling and Management: Schedule shifts, assign tasks, and track employee performance to ensure adequate staffing levels, optimize labor costs, and maintain high standards of service.

5. Financial Tracking: Record and monitor financial transactions such as sales, payments, and invoices to track revenue, expenses, and profitability accurately.

6. Data Analysis: Utilize data analytics tools to derive insights, identify trends, and make data-driven decisions for improving operations, menu offerings, and overall business performance.

By addressing these key areas, the restaurant management system aims to empower restaurant owners and managers with the tools and insights needed to run their establishments efficiently, deliver exceptional dining experiences, and achieve sustainable growth in a competitive industry landscape.

**Related Work and Technologies**

Programs used:

• SQL Management System

• Microsoft Word

• Microsoft PowerPoint

• Draw.io

* Canva

How it was used

SQL Management System was used to implement the code of the System.

The Microsoft office word and PowerPoint were both used respectively to make the project report and project representation.

Draw.io alongside MySQL Workbench were used to draw the diagram.

We used Canva to make the prototype of the website and to have an idea of the menu.

**Entity-Relationship Model for E-commerce System:**

In this system, there are ten main entities: Inventory, Customer, Employee, Menu, Reservation, Table, Order, Invoice, Cooks and Shift. Every entity has its own attributes.

1. Entities:

* Inventory
* Customer
* Menu
* Reservation
* Table
* Employee
* Invoice
* Shift
* Cooks
* Order

B) Attributes for each entity:

1. Inventory:

InventoryID Primary Key

ItemName

Quantity

1. Customer:

CustomerID Primary Key

CustomerName

Feedback

1. Menu:

Item Primary Key

Category

Price

CustomerID Foreign Key

1. Reservation:

* ReservationID Primary Key
* ReservationTime
* GroupSize
* ReservationDate
* CustomerID Foreign Key
* TableNumber Foreign Key

1. Table:

* TableNumber Primary Key
* Capacity
* EmployeeID Foreign Key
* ReservationID Foreign Key

1. Employee:

* EmployeeID Primary Key
* Name
* Position Foreign Key
* InventoryID Foreign Key

1. Invoice:

* InvoiceID Primary Key
* InvoiceDate
* PaymentStatus
* TableNumber Foreign Key
* OrderID Foreign Key

1. Shift:

* ShiftID Primary Key
* Date
* StartTime
* EndTime
* EmployeeID Foreign Key

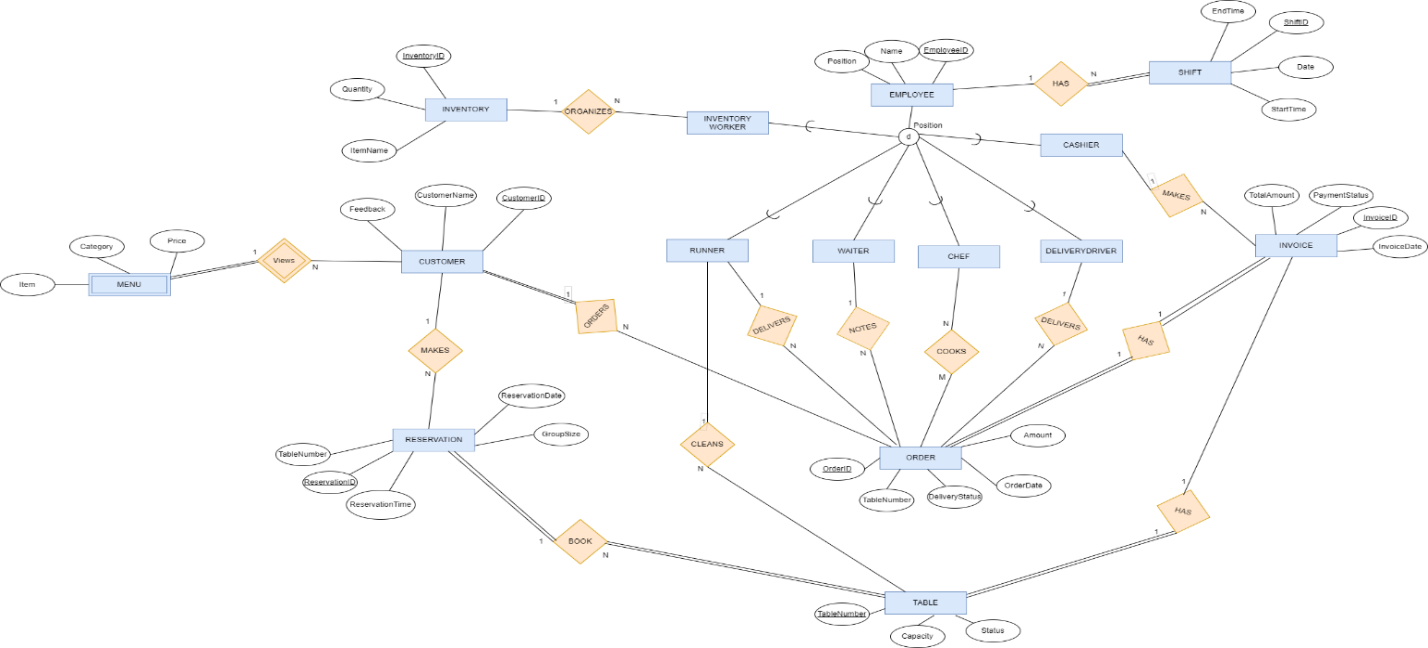
1. Cooks:

* EmployeeID Primary Key and Foreign Key
* OrderID Primary Key and Foreign Key

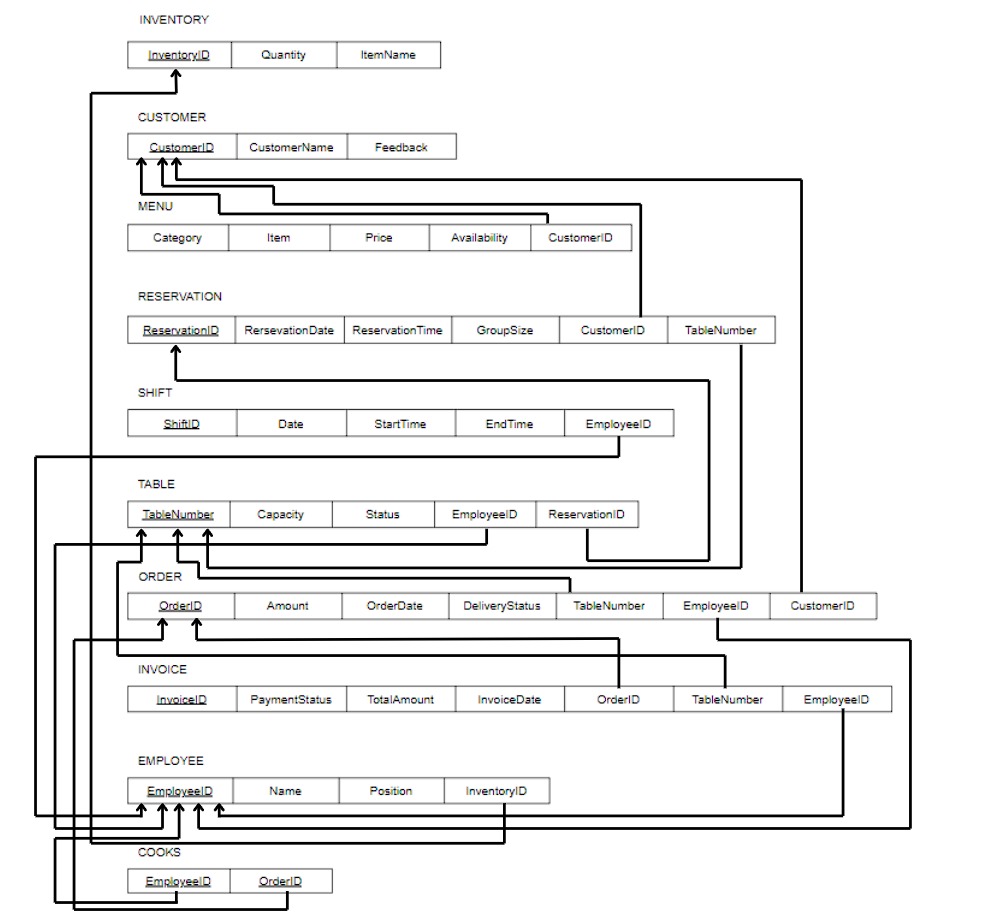
1. Order:

* OrderID Primary Key
* DeliveryStatus
* OrderDate
* Amount
* EmployeeID Foreign Key
* CustomerID Foreign Key
* TableNumber Foreign Key

**Diagrams**

ER Diagram

Schema

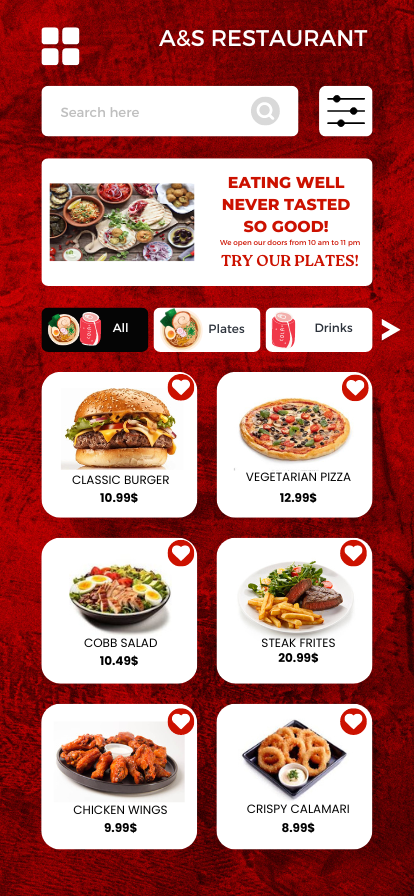


**Prototype**

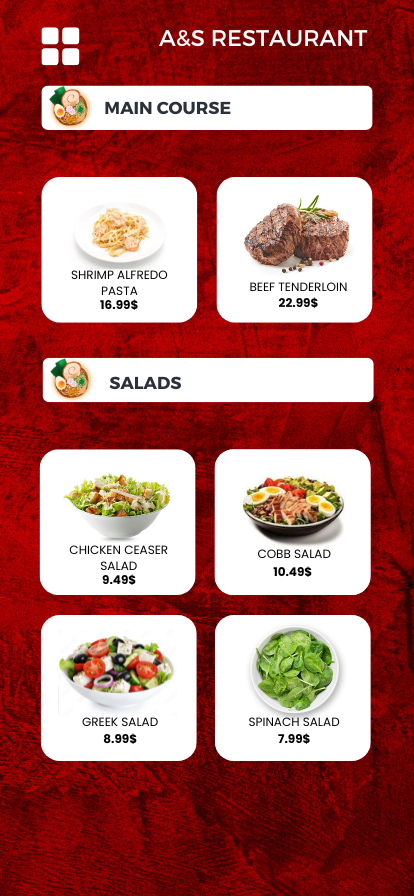
A user could:

* view the menu
* give feedback about the restaurant

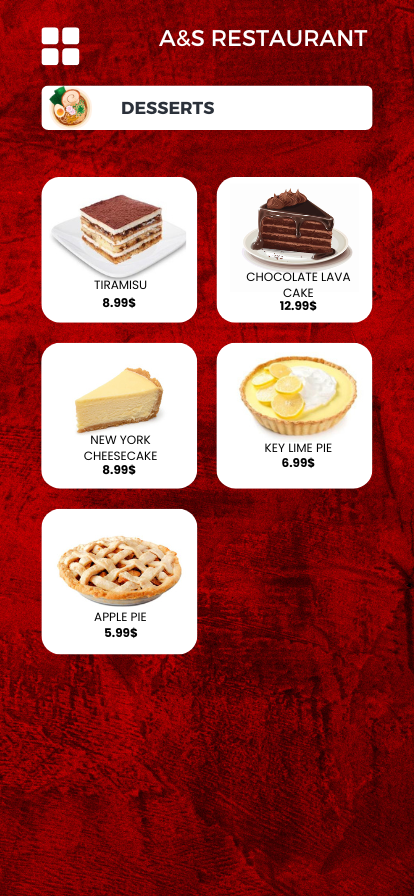
**Website preview:**



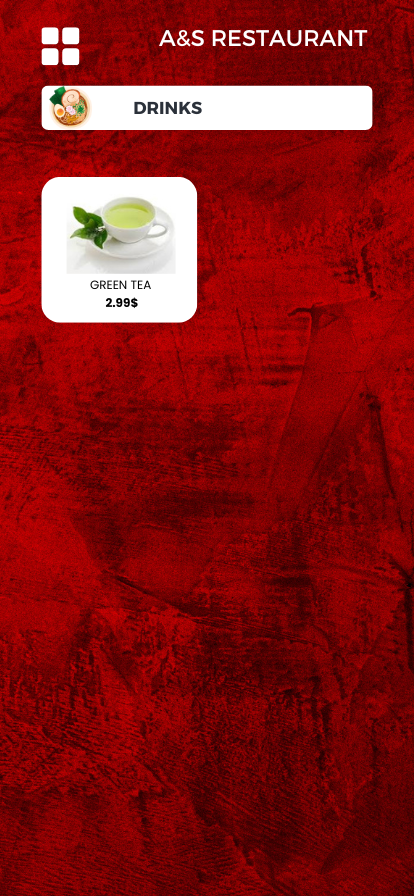


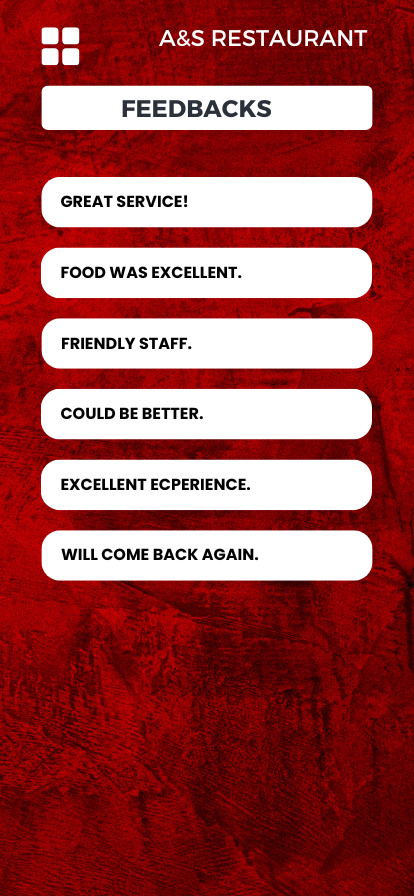












**Implementation:**

SQL creating tables Queries:

**Creating the tables:**

Create database Restaurant

use Restaurant

CREATE TABLE Inventory(

InventoryID int IDENTITY(1,1),

ItemName varchar(max) NOT NULL,

Quantity int,

Primary Key(InventoryID)

);

CREATE TABLE Customer(

CustomerID int IDENTITY(1,1),

CustomerName varchar(max) NOT NULL,

Feedback varchar(max),

Primary Key(CustomerID)

);

CREATE TABLE Employee1(

EmployeeID int IDENTITY(1,1),

Name varchar(max) NOT NULL,

Position varchar(max) NOT NULL,

InventoryID int,

Primary Key(EmployeeID),

Foreign Key(InventoryID) references Inventory(InventoryID) on delete cascade on update cascade,

);

CREATE TABLE Menu(

Item varchar(max) NOT NULL,

Category varchar(max) NOT NULL,

Price money NOT NULL,

CustomerID int,

Foreign Key(CustomerID) references Customer(CustomerID) on delete cascade on update cascade

);

CREATE TABLE Shift1(

ShiftID int IDENTITY(1,1),

Date1 date NOT NULL,

StartTime time NOT NULL,

EndTime time NOT NULL,

EmployeeID int,

Foreign Key(EmployeeID) references Employee1(EmployeeID) on delete cascade on update cascade,

Primary Key(ShiftID)

);

CREATE TABLE Reservation(

ReservationID int IDENTITY(1,1),

ReservationTime time NOT NULL,

GroupSize int NOT NULL,

ReservationDate date NOT NULL,

CustomerID int,

TableNumber int,

Primary Key(ReservationID),

Foreign Key(CustomerID) references Customer(CustomerID) on delete cascade on update cascade,

);

CREATE TABLE Table1(

TableNumber int unique,

Capacity int NOT NULL,

Status varchar(max) NOT NULL,

EmployeeID int ,

ReservationID int,

Primary Key (TableNumber),

FOREIGN KEY (ReservationID) REFERENCES Reservation(ReservationID) on delete cascade on update cascade,

Foreign Key(EmployeeID) references Employee1(EmployeeID) on delete cascade on update cascade,

);

CREATE TABLE Order1(

OrderID int IDENTITY(1,1),

DeliveryStatus varchar(max) NOT NULL,

OrderDate date NOT NULL,

Amount int NOT NULL,

EmployeeID int,

CustomerID int,

TableNumber int,

Primary Key(OrderID),

Foreign Key(EmployeeID) references Employee1(EmployeeID) on delete cascade on update cascade,

Foreign Key(CustomerID) references Customer(CustomerID) on delete cascade on update cascade,

Foreign Key(TableNumber) references Table1(TableNumber)

);

CREATE TABLE Invoice(

InvoiceID int IDENTITY(1,1),

InvoiceDate date NOT NULL,

PaymentStatus varchar(max),

TotalAmount int NOT NULL,

TableNumber int,

OrderID int,

EmployeeID int,

Primary Key(InvoiceID),

Foreign Key(TableNumber) references Table1(TableNumber) on delete cascade on update cascade,

Foreign Key(OrderID) references Order1(OrderID),

Foreign Key(EmployeeID) references Employee1(EmployeeID)

);

CREATE TABLE Cooks(

EmployeeID int NOT NULL,

OrderID int NOT NULL,

Foreign Key(EmployeeID) references Employee1(EmployeeID),

Foreign Key(OrderID) references Order1(OrderID),

Primary Key(EmployeeID,OrderID)

);

Insertion:

INSERT INTO Inventory (ItemName, Quantity)

VALUES

('Tomatoes, Onions, Lettuce, Rice, Potatoes', 500),

('Chicken, Pasta, Beef, Cheese, Fish', 250);

INSERT INTO Customer (CustomerName, Feedback)

VALUES

('Sophie Evans', 'Great service!'),

('William Thompson', 'The food was excellent.'),

('Evelyn Hernandez', NULL),

('Henry Carter', 'Friendly staff.'),

('Amelia Davis', NULL),

('Jack Moore', 'Could be better.'),

('Charlotte Taylor', 'Excellent experience.'),

('Daniel Adams', NULL),

('Harper Wilson', 'Will come back again.'),

('Matthew Parker', NULL),

('Harry Potter', NULL),

('James Bond', NULL);

INSERT INTO Employee1 (Name, Position, InventoryID)

VALUES

('John Doe', 'Waiter', Null),

('Jane Smith', 'Chef', Null),

('Michael Johnson', 'DeliveryDriver', Null),

('David Brown', 'Runner', Null),

('Sarah Miller', 'Cashier', Null),

('Robert Jones', 'Waiter', Null),

('Jennifer Davis', 'Chef', Null),

('Daniel Wilson', 'DeliveryDriver', Null),

('Andrew Clark', 'Runner', Null),

('Emma Garcia', 'Cashier', Null),

('Matthew Rodriguez', 'InventoryWorker',1),

('Olivia Martinez', 'Chef', Null),

('William Hernandez', 'DeliveryDriver', Null),

('Sophia Adams', 'Runner', Null),

('James Turner', 'Runner', Null),

('Ella Scott', 'Runner', Null),

('Nathan Phillips', 'InventoryWorker',2);

INSERT INTO Reservation (ReservationTime, GroupSize, ReservationDate,TableNumber, CustomerID)

VALUES

('18:30:00', 4, '2024-04-20', 2, 1),

('19:15:00', 6, '2024-04-21', 3 ,2),

('20:45:00', 4, '2024-04-22',6, 3),

('21:30:00', 2, '2024-04-23', 9,4),

('22:20:00', 8, '2024-04-24', 12,5),

('23:00:00', 6, '2024-04-25', 6,6),

('19:45:00', 4, '2024-04-26', 4,7),

('20:30:00', 2, '2024-04-27', 20,8),

('21:10:00', 6, '2024-04-28', 15,9),

('22:40:00', 4, '2024-04-29', 12,10);

INSERT INTO Table1 (TableNumber,Capacity, Status, EmployeeID, ReservationID)

VALUES

(2,4, 'Occupied', 4, 1),

(3,6, 'Available', NULL, NULL),

(6,4, 'Occupied', 9, 2),

(9,2, 'Available', NULL, NULL),

(12,8, 'Occupied', 14, 3),

(13,6, 'Available', NULL, NULL),

(14,4, 'Occupied', 15, 4),

(15,2, 'Available', NULL, NULL),

(19,6, 'Occupied', 16, 5),

(20,4, 'Available', NULL, NULL);

INSERT INTO Order1 (DeliveryStatus, OrderDate, Amount, EmployeeID, CustomerID, TableNumber)

VALUES

('InRestaurant', '2024-04-20', 50, 4, 1, 2),

('Takeaway', '2024-04-20', 30, NULL, 2, NULL),

('Delivery', '2024-04-20', 70, 3, 3, NULL),

('Delivery', '2024-04-23', 25, 8, 4, NULL),

('InRestaurant', '2024-04-24', 40, 9, 5, 6),

('Takeaway', '2024-04-25', 60, NULL, 6, NULL),

('Delivery', '2024-04-26', 55, 13, 7, NULL),

('Delivery', '2024-04-26', 45, 3, 8, NULL),

('InRestaurant', '2024-04-28', 35, 14, 9, 12),

('InRestaurant', '2024-04-28', 20, 15, 10, 20),

('Delivery', '2024-04-29', 20, 8, 8, NULL),

('InRestaurant', '2024-04-30', 32, 4, 11, 19),

('InRestaurant', '2024-04-30', 22, 15, 12, 15);

INSERT INTO Invoice (InvoiceDate, PaymentStatus, TotalAmount, TableNumber, OrderID, EmployeeID)

VALUES

('2024-04-20', 'ByCard', 242, 2, 1, 5),

('2024-04-20', 'ByCash', 150, NULL, 2, 5),

('2024-04-20', 'ByCard', 412, 6, 5, 5),

('2024-04-23', 'ByCard', 250, 12, 9, 10),

('2024-04-24', 'ByCash', 200, 20, 10, 10),

('2024-04-25', 'ByCard', 389, NULL, 6, 5),

('2024-04-26', 'ByCard', 287, NULL, 7, 10),

('2024-04-26', 'ByCash', 231, NULL, 8, 10),

('2024-04-28', 'ByCard', 210, 19, 3, 5),

('2024-04-28', 'ByCash', 120, 6,4, 10);

INSERT INTO Menu (Item, Category, Price)

VALUES

('Classic Burger', 'Main Course', 10.99),

('Vegetarian Pizza', 'Main Course', 12.99),

('Mushroom Risotto', 'Main Course', 14.99),

('Grilled Salmon', 'Main Course', 18.99),

('Pasta Carbonara', 'Main Course', 13.99),

('Steak Frites', 'Main Course', 20.99),

('Fish and Chips', 'Main Course', 15.99),

('Margherita Pizza', 'Main Course', 11.99),

('Shrimp Alfredo Pasta', 'Main Course', 16.99),

('Beef Tenderloin', 'Main Course', 22.99),

('Chicken Caesar Salad', 'Salad', 9.49),

('Greek Salad', 'Salad', 8.99),

('Cobb Salad', 'Salad', 10.49),

('Spinach Salad', 'Salad', 7.99),

('Garlic Bread', 'Appetizer', 5.99),

('Cheese Platter', 'Appetizer', 12.99),

('Bruschetta', 'Appetizer', 6.99),

('Crispy Calamari', 'Appetizer', 8.99),

('Chicken Wings', 'Appetizer', 9.99),

('Tiramisu', 'Dessert', 8.99),

('Chocolate Lava Cake', 'Dessert', 7.99),

('New York Cheesecake', 'Dessert', 9.99),

('Key Lime Pie', 'Dessert', 6.99),

('Apple Pie', 'Dessert', 5.99),

('Soda', 'Drink', 2.99),

('Iced Tea', 'Drink', 2.49),

('Lemonade', 'Drink', 2.79),

('Espresso', 'Drink', 3.49),

('Cappuccino', 'Drink', 3.99),

('Hot Chocolate', 'Drink', 3.49),

('Margarita', 'Drink', 7.49),

('Mojito', 'Drink', 6.99),

('Green Tea', 'Drink', 2.99);

INSERT INTO Cooks (EmployeeID, OrderID)

VALUES

(2, 1),

(7, 2),

(12, 3),

(2, 4),

(7, 5),

(7, 6),

(12, 7),

(2, 8),

(12, 9),

(2, 10);

INSERT INTO Shift1 (Date1, StartTime, EndTime, EmployeeID)

VALUES

('2024-03-20', '08:00:00', '16:00:00', 1),

('2024-03-20', '08:00:00', '16:00:00', 2),

('2024-01-22', '16:00:00', '00:00:00', 3),

('2024-01-01', '16:00:00', '00:00:00', 4),

('2024-01-01', '16:00:00', '00:00:00', 5),

('2024-01-01', '08:00:00', '16:00:00', 6),

('2024-02-06', '08:00:00', '16:00:00', 7),

('2024-02-08', '16:00:00', '00:00:00', 8),

('2024-01-26', '16:00:00', '00:00:00', 9),

('2024-02-28', '08:00:00', '16:00:00', 10),

('2024-03-22', '16:00:00', '00:00:00', 11),

('2024-02-16', '16:00:00', '00:00:00', 12),

('2024-02-10', '08:00:00', '16:00:00', 13),

('2024-05-12', '08:00:00', '16:00:00', 14),

('2024-03-09', '16:00:00', '00:00:00', 15),

('2024-03-09', '16:00:00', '00:00:00', 16),

('2024-02-12', '08:00:00', '16:00:00', 17);

50 SQL Queries:

--1 Display all tables

select \* from Inventory

select \* from Employee1

select \* from Customer

select \* from Menu

select \* from Reservation

select \* from Table1

select \* from Order1

select \* from Invoice

select \* from Cooks

select \* from Shift1

--2 Select the id's of customers who didn't make any orders

SELECT c.CustomerID From Customer as c,Order1 as o

where c.CustomerID = o.CustomerID

AND o.CustomerID IS NULL;

--3 Select Employees who work in the invetories

Select e.\* from Employee1 as e

where e.InventoryID is not null;

--4 Display how many items in total we have and displaying how many items in each category

SELECT COUNT(\*) AS [Total Items] FROM Menu;

SELECT COUNT(\*) AS [Main Courses] FROM Menu WHERE Category = 'Main Course';

SELECT COUNT(\*) AS Salads FROM Menu WHERE Category = 'Salad';

SELECT COUNT(\*) AS Appetizers FROM Menu WHERE Category = 'Appetizer';

SELECT COUNT(\*) AS Desserts FROM Menu WHERE Category = 'Dessert';

SELECT COUNT(\*) AS Drinks FROM Menu WHERE Category = 'Drink';

--5 Display the names of customers who left a feedback

SELECT CustomerName FROM Customer WHERE Feedback IS NOT NULL;

--6 Display the item with the highest price and it's price in the menu

SELECT Item, Price FROM Menu

WHERE Price = (SELECT MAX(Price) FROM Menu);

--7 Calculating the items sold

SELECT SUM(Amount) AS [Items Sold] FROM Order1;

--8 Calculating the total income from invoices

SELECT SUM(TotalAmount) AS [Total Income] FROM Invoice;

--9 Calculating the income on 2024-04-20

SELECT SUM(TotalAmount) AS [Income of 2024-04-20] FROM Invoice where InvoiceDate = '2024-04-20';

--10 List the reservations of the date 2024-04-24

SELECT \* FROM Reservation WHERE ReservationDate = '2024-04-24';

--11 Checking how many available tables we have and how many occupied

SELECT COUNT(\*) AS AvailableTables FROM Table1 WHERE Status = 'Available';

SELECT COUNT(\*) AS AvailableTables FROM Table1 WHERE Status = 'Occupied';

--12 Setting the feedback of Evelyn Hernandez of id 3 to 'Excellent service'

UPDATE Customer SET Feedback = 'Excellent service!' WHERE CustomerID = 3;

select \* from Customer where CustomerID=3;

--13 Adding on the Menu item 'Seafood Paella' as a main course with price 24.99

INSERT INTO Menu (Item, Category, Price) VALUES ('Seafood Paella', 'Main Course', 24.99);

--14 List all orders made by 'Sophie Evans'

SELECT \* FROM Order1 WHERE CustomerID = (SELECT CustomerID FROM Customer WHERE CustomerName = 'Sophie Evans');

--15 Inserting the missing tables

INSERT INTO Table1 (TableNumber, Capacity, Status, EmployeeID, ReservationID)

VALUES

(1, 2, 'Available', NULL, NULL),

(4, 4, 'Available', NULL, NULL),

(5, 6, 'Available', NULL, NULL),

(7, 2, 'Available', NULL, NULL),

(8, 4, 'Available', NULL, NULL),

(10, 6, 'Available', NULL, NULL),

(11, 2, 'Available', NULL, NULL),

(16, 4, 'Available', NULL, NULL),

(17, 6, 'Available', NULL, NULL),

(18, 2, 'Available', NULL, NULL);

--16 Someone has Occupied table number 3

UPDATE Table1 SET Status = 'Occupied' WHERE TableNumber = 3;

--17 List the EmployeeIDs that starts work at 8 am and list those who start at 4

SELECT EmployeeID AS [Employees at 8 am] FROM Employee1 WHERE EmployeeID IN (SELECT EmployeeID FROM Shift1 WHERE StartTime='08:00:00');

SELECT EmployeeID AS [Employees at 4 pm] FROM Employee1 WHERE EmployeeID IN (SELECT EmployeeID FROM Shift1 WHERE StartTime='16:00:00');

--18 List the total number of items in the Inventory and in each Inventory

SELECT SUM(Quantity) AS TotalItems FROM Inventory;

SELECT SUM(Quantity) AS TotalItems FROM Inventory WHERE InventoryID=1;

SELECT SUM(Quantity) AS TotalItems FROM Inventory WHERE InventoryID=2;

--19 Calculating the price of each category such that 1 item from each item is bought

SELECT Category, SUM(Price) AS [Total Price] FROM Menu GROUP BY Category;

--20 Calculating the number of delivery orders

SELECT COUNT(\*) AS TotalDeliveredOrders FROM Order1 WHERE DeliveryStatus = 'Delivery';

--21 updating the quantity of the invetory where it has tomatoes

UPDATE Inventory SET Quantity = 600 WHERE ItemName LIKE '%Tomatoes%';

--22 List the Income and number of card and cash payments

SELECT SUM(TotalAmount) AS IncomeCardPayments, COUNT(\*) AS NumberCardPayments FROM Invoice WHERE PaymentStatus LIKE '%Card%';

SELECT SUM(TotalAmount) AS IncomeCashPayments, COUNT(\*) AS NumberCashPayments FROM Invoice WHERE PaymentStatus LIKE '%Cash%';

--23 List the id and name of the customer who didn't leave a feedback

SELECT c.CustomerID,c.CustomerName FROM Customer c WHERE c.Feedback IS NULL;

--24 Calculate the total amount and total number of orders for each employee who worked as a runner

SELECT

e.Name AS EmployeeName,

SUM(o.Amount) AS TotalAmount,

COUNT(o.OrderID) AS TotalOrders

FROM

Employee1 e, Order1 o

WHERE

e.EmployeeID = o.EmployeeID and

e.Position = 'Runner'

GROUP BY

e.Name;

--25 Identify the most popular reservation time slot (hourly) and the total number of reservations made during that slot

SELECT

DATEPART(HOUR, r.ReservationTime) AS ReservationHour,

COUNT(r.ReservationID) AS TotalReservations

FROM

Reservation r

GROUP BY

DATEPART(HOUR, r.ReservationTime)

ORDER BY

TotalReservations DESC;

--26 Calculate the total amount of money spent by every customer in descending order

SELECT c.CustomerName, SUM(i.TotalAmount) AS TotalSpent

FROM Customer c, Invoice i, Order1 o WHERE c.CustomerID = o.CustomerID and o.OrderID=i.OrderID

GROUP BY c.CustomerName

ORDER BY TotalSpent DESC;

--27 Calculate the average order price for each category in the Menu

SELECT Category, AVG(Price) AS AvgOrderPrice FROM Menu GROUP BY Category;

--28 Find the total income on weekends Saturday and Sunday

SELECT SUM(i.TotalAmount) AS Income

FROM Invoice i ,Order1 o

WHERE DATENAME(WEEKDAY, o.OrderDate) IN ('Saturday', 'Sunday');

--29 List of the names of customers who placed orders on April 24, 2020

SELECT c.CustomerName FROM Customer c ,Order1 o

WHERE c.CustomerID = o.CustomerID

AND o.OrderDate = '2024-04-20';

--30 List the ids and names of customers who made reservations for tables with a capacity of 6 or more

SELECT c.CustomerID,c.CustomerName FROM Customer c, Reservation r, Table1 t

WHERE c.CustomerID = r.CustomerID

AND r.TableNumber = t.TableNumber

AND t.Capacity >= 6;

--31 The percentage of orders in reataurant, delivered and takeaway compared to total orders

SELECT (COUNT(\*) \* 100.0) / (SELECT COUNT(\*) FROM Order1) AS InRestaurantPercentage

FROM Order1 WHERE DeliveryStatus = 'InRestaurant';

SELECT (COUNT(\*) \* 100.0) / (SELECT COUNT(\*) FROM Order1) AS DeliveryPercentage

FROM Order1 WHERE DeliveryStatus = 'Delivery';

SELECT (COUNT(\*) \* 100.0) / (SELECT COUNT(\*) FROM Order1) AS TakeawayPercentage

FROM Order1 WHERE DeliveryStatus = 'Takeaway';

--32 List of the names of employees who are either chefs or waiters

SELECT Name FROM Employee1 WHERE Position = 'Chef' or EmployeeID IN (SELECT EmployeeID FROM Employee1 WHERE Position = 'Waiter');

--33 Find the total number of orders made by each customer

SELECT c.CustomerID, c.CustomerName, COUNT(o.OrderID) AS TotalOrders

FROM Customer c

LEFT JOIN Order1 o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.CustomerName;

--34 Calculate the total amount ordered by customers who left feedback

SELECT SUM(o.Amount) AS TotalAmountOrdered FROM Order1 o, Customer c

WHERE o.CustomerID = c.CustomerID

AND c.Feedback IS NOT NULL;

--35 displaying menu from higher price to lowest price

SELECT Item, Category

FROM Menu

ORDER BY Price DESC;

--36 Changing position datatype

ALTER TABLE Employee1 ALTER COLUMN Position VARCHAR(50);

--37 update the inventory quantity where there is chicken

UPDATE Inventory

SET Quantity = 300

WHERE ItemName LIKE '%Chicken%';

select \* from Inventory

--38 updating the price of Greek Salad

UPDATE Menu

SET Price = 7.99

WHERE Item = 'Greek Salad';

select \* from Menu

--39 changing date of reservation id=7

UPDATE Reservation

SET ReservationDate = '2024-05-05'

WHERE ReservationID = 7;

--40 Delete an item from the Menu by ItemName

DELETE FROM Menu

WHERE Item = 'Tiramisu';

--41 Listing employees where position is chef

SELECT \*

FROM Employee1

WHERE Position = 'Chef';

--42 Displaying The employees that is runner and has shift from 8 am

SELECT e.EmployeeID FROM Employee1 e, Shift1 s

WHERE e.EmployeeID=s.EmployeeID

AND StartTime='08:00:00'

AND Position='Runner'

--43 Inventory that has less than 300 as quantity

SELECT ItemName, Quantity

FROM Inventory

WHERE Quantity < 300;

--44 Retrieve orders made by customer who has id=5

SELECT \*

FROM Order1

WHERE CustomerID = 5;

--45 checking if any employee doesn't have a shift

SELECT Name

FROM Employee1

WHERE EmployeeID NOT IN (SELECT EmployeeID FROM Shift1);

--46 List the names of customers who bought more than 50 items

SELECT c.CustomerName

FROM Customer c, Order1 o

where c.CustomerID = o.CustomerID

and o.Amount > 50;

--47 Retrieve the details of the table with the highest capacity

SELECT TOP 1 \*

FROM Table1

ORDER BY Capacity DESC;

--48 List the names of employees who have worked shifts on weekends (Saturday and Sunday(1))

SELECT e.Name

FROM Employee1 e, Shift1 s

WHERE e.EmployeeID = s.EmployeeID

AND DATEPART(dw, s.Date1) IN (1, 7);

--49 Find the total number of orders made on each day of the week

SELECT DATEPART(dw, OrderDate) AS DayOfWeek, COUNT(OrderID) AS TotalOrders

FROM Order1

GROUP BY DATEPART(dw, OrderDate);

--50 Retrieve the details of the reservation with the highest group size

SELECT \*

FROM Reservation

WHERE GroupSize = (SELECT MAX(GroupSize) FROM Reservation);