CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

UNIVERSITY INSTITUTE OF COMPUTING

CASE STUDY REPORT
ON
RESTAURANT MANAGEMENT SYSTEM

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

Submitted By:

Name: Sumit Kalyan
UID: 23BCA10316
Section: 4-‘B’

Submitted To:

Name: Mr.Arvinder Singh
Designation: Assistant Professor

INTRODUCTION

The **Restaurant Management System** is a comprehensive database project designed to streamline and manage the various operations of a restaurant using structured SQL queries and relational database concepts. This system allows efficient handling of essential components such as customer information, employee details, menu items, order processing, payment tracking, and table reservations.

The primary objective of this project is to provide a centralized system that improves data organization, enhances customer service, and facilitates insightful reporting for better decision-making. By using SQL, we ensure the data is well-structured, relational, and easily accessible for real-time operations and analysis.

This database system includes multiple interrelated tables such as Customers, Employees, Menu, Orders, Order_Details, Payments, and Reservations. It supports functionalities like order management, tracking payment methods, analyzing top-selling items, managing employee roles, and handling customer bookings efficiently.

With well-designed SQL queries, this system can generate vital reports like total revenue, order breakdowns, popular menu items, employee performance, and customer engagement, making it an ideal tool for modern restaurant management.

TECHNIQUES

The primary technology used in this project is MySQL, an open-source relational database management system. The following techniques have been implemented:

- **Entity-Relationship Modeling** for data structure visualisation.
- **Normalisation** to organise data efficiently and remove redundancy.
- **SQL Queries** for data manipulation and retrieval.
- **Use of Constraints** like PRIMARY KEY, FOREIGN KEY to enforce relationships.
- **Join operations** to combine data from multiple tables.
- **Aggregate Functions** to summarize and analyze data.
- **Filtering and Sorting** to extract meaningful insights from the dataset.
- **Stored Procedures and Views** (optional enhancements) for automation.

The goal is to simulate a real-time cinema database with multiple users accessing the system concurrently. Though our current system is simplified, it lays the foundation for large-scale enterprise software.

SYSTEM CONFIGURATION

Hardware Requirements

- **Processor:** Intel i5 / Ryzen 5 or higher
- **RAM:** 8 GB minimum
- **Storage:** 256 GB SSD / 500 GB HDD
- **Display:** 14" or larger

Software Requirements

- **OS:** Windows 10/11 or Ubuntu 20.04+
- **DBMS:** MySQL Server 8.0+
- **Interface Tool:** MySQL Workbench / phpMyAdmin
- **ER Tool:** Draw.io / dbdiagram.io
- **Editor:** VS Code / Notepad++

Database Details

- **Name:** vansh_db
- **Tables:** Customers, Orders, OrderDetails, Products, Suppliers, Employees
- **Relations:** Primary & Foreign Keys, Constraints for integrity

INPUT

The input for the Restaurant Management System is the structured data provided to different tables of the database. This data is inserted using SQL INSERT statements and reflects real-world entities and interactions within a restaurant environment. The input includes:

- **Customer Details**:
  Inputs include customer name, phone number, email, and address. These details help in identifying and managing customer interactions, orders, and reservations.

- **Employee Details**:
  Data such as employee name, role (e.g., Chef, Waiter, Manager), salary, and contact number are stored to manage staff operations.

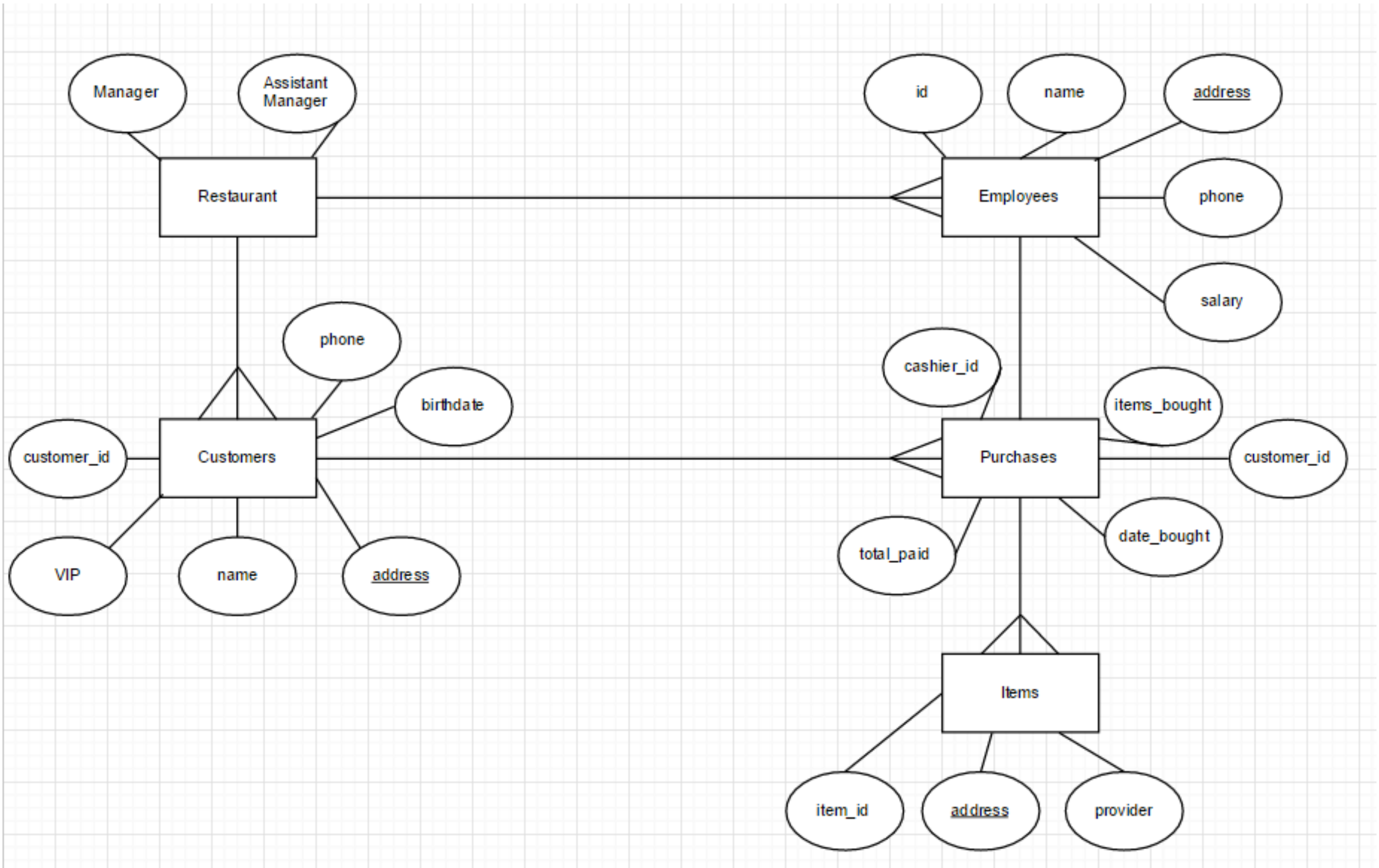
- **Menu Items**:
  Inputs include item name, category (Main, Starter, Dessert, Drink), price, and availability status. This helps manage the list of offerings at the restaurant.

- **Order Details**:
  Each order records the customer placing the order, the employee handling it, the total bill, and the date/time. Linked order items include the quantity and price of each menu item ordered.

- **Payments**:
  Inputs capture the order ID, payment method (Cash, Card, UPI), amount paid, and payment date. This helps in tracking the financials of the restaurant.

- **Reservations**:
  Inputs include customer ID, reservation time, number of people, and table

ENTITY-RELATIONSHIP DIAGRAM



The Entity-Relationship (ER) diagram outlines the structure and relationships among different entities of the restaurant. It forms the blueprint for the actual database schema.

Each entity has clearly defined attributes and is connected using appropriate relationships like one-to-many and many-to-one, ensuring normalization and avoiding data redundancy.

RELATIONSHIP BETWEEN TABLES

These relationships ensure that the relational database mirrors real-world interactions within Restaurant.

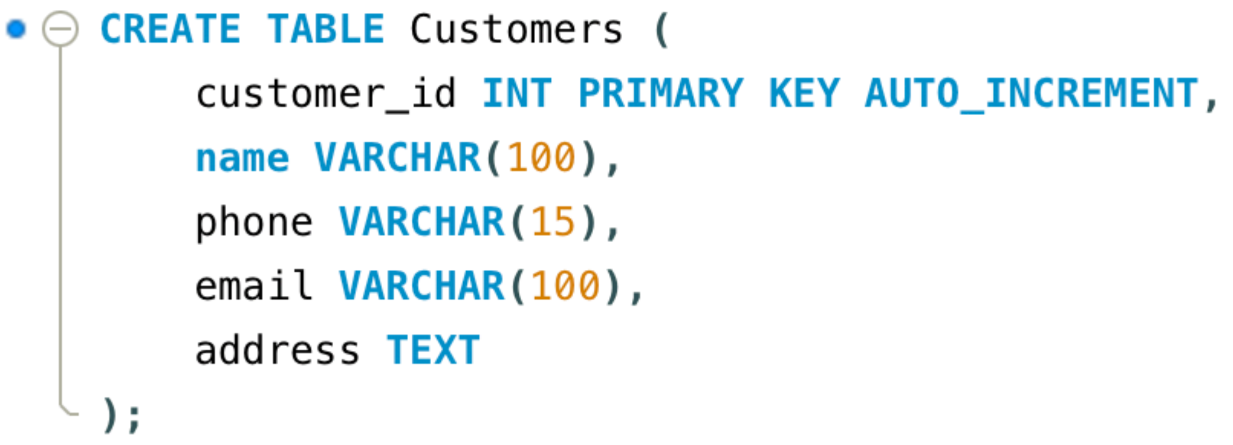
No.	Relationship Type	Parent Table	Child Table	Foreign key	Description
1	One-to-many	Customers	Orders	CustomerID	Customer can place multiple orders
2	One-to-many	Employees	Orders	EmployeeID	One employee can handle multiple orders
3	One-to-many	Menu	OrderDetails	ItemID	One item can appear multiple order details
4	One-to-many	Orders	OrderDetails	OrderID	One order can contain multiple items
5	One-to-many	Orders	Payments	OrderID	One order have one payment with it
6	One-to-many	Customers	Reservations	CustomerID	One customer make multiple reservations

TABULAR FORMAT (SCHEMA)

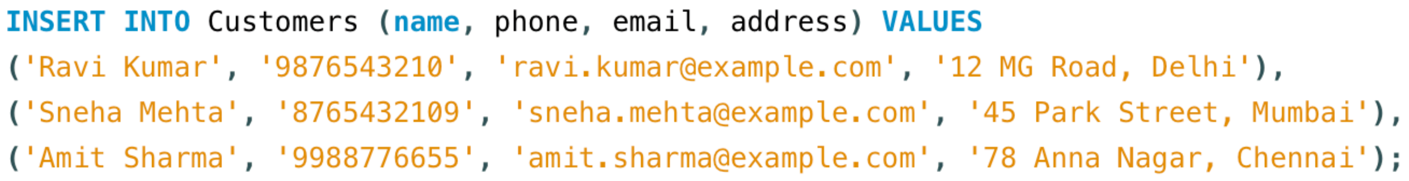
Table Name	Primary Key	Foreign Key	Description
Customers	CustomerID	—	Store Customer info
Employees	EmployeeID	—	Record of Employee
Menu	ItemID	—	Contain menu items
Orders	OrderID	CustomerID, EmployeeID	Records order details
OrderDetails	detailsID	OrderID,ItemID	Store specific item
Payment	PaymentID	OrderID	Store paymentdetail
Reservations	ReservationID	CustomerID	Stores table booking

TABLE CREATION

1. Customers Table:

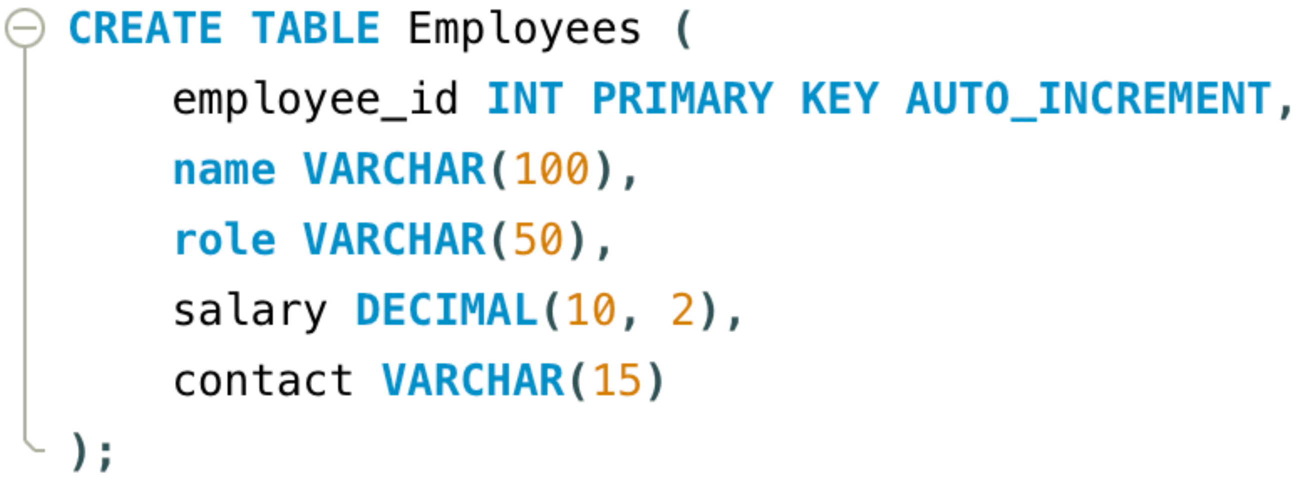


```
CREATE TABLE Customers (
        customer_id INT PRIMARY KEY AUTO_INCREMENT,
        name VARCHAR(100),
        phone VARCHAR(15),
        email VARCHAR(100),
        address TEXT
);
```

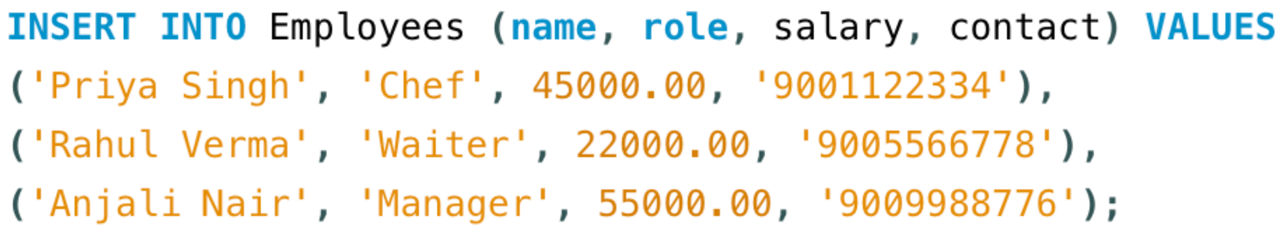
- 

```
INSERT INTO Customers (name, phone, email, address) VALUES
('Ravi Kumar', '9876543210', 'ravi.kumar@example.com', '12 MG Road, Delhi'),
('Sneha Mehta', '8765432109', 'sneha.mehta@example.com', '45 Park Street, Mumbai'),
('Amit Sharma', '9988776655', 'amit.sharma@example.com', '78 Anna Nagar, Chennai');
```

2. Employees Table:

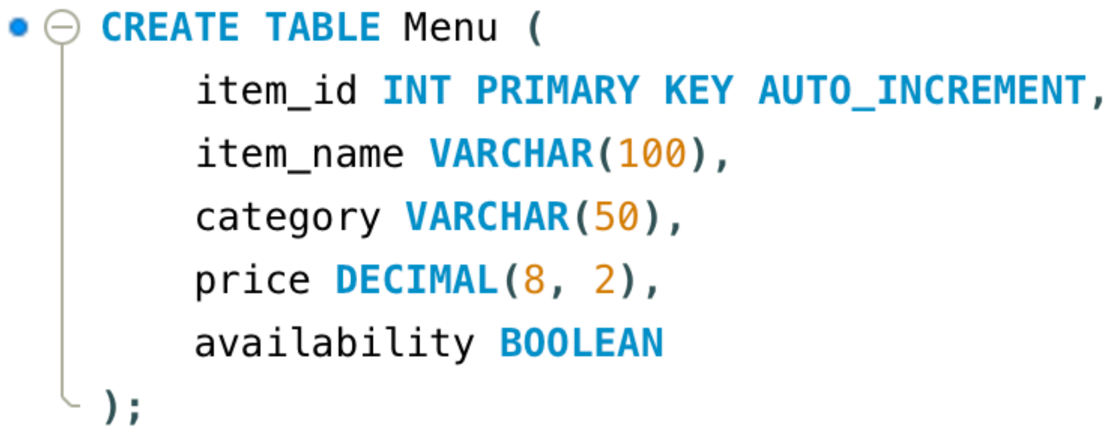
- 

```
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    role VARCHAR(50),
    salary DECIMAL(10, 2),
    contact VARCHAR(15)
);
```

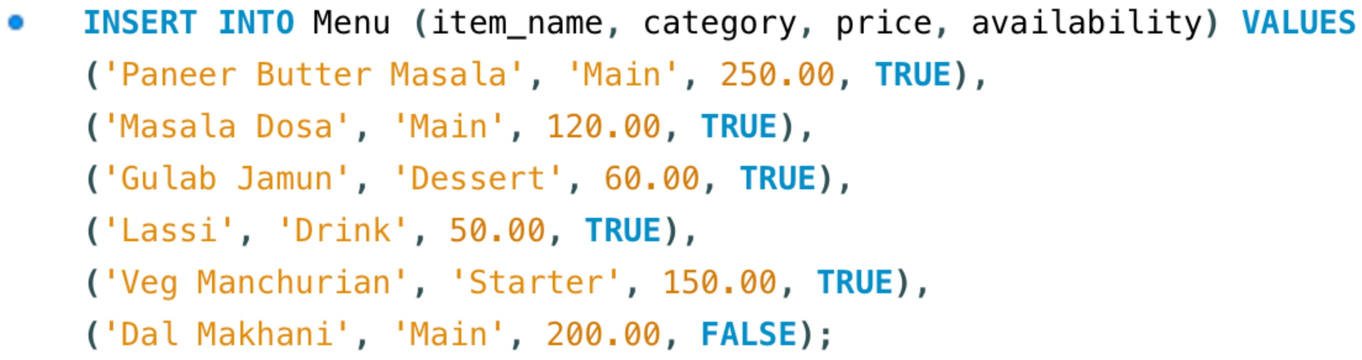
- 

```
INSERT INTO Employees (name, role, salary, contact) VALUES
('Priya Singh', 'Chef', 45000.00, '9001122334'),
('Rahul Verma', 'Waiter', 22000.00, '9005566778'),
('Anjali Nair', 'Manager', 55000.00, '9009988776');
```

3. Menu Table

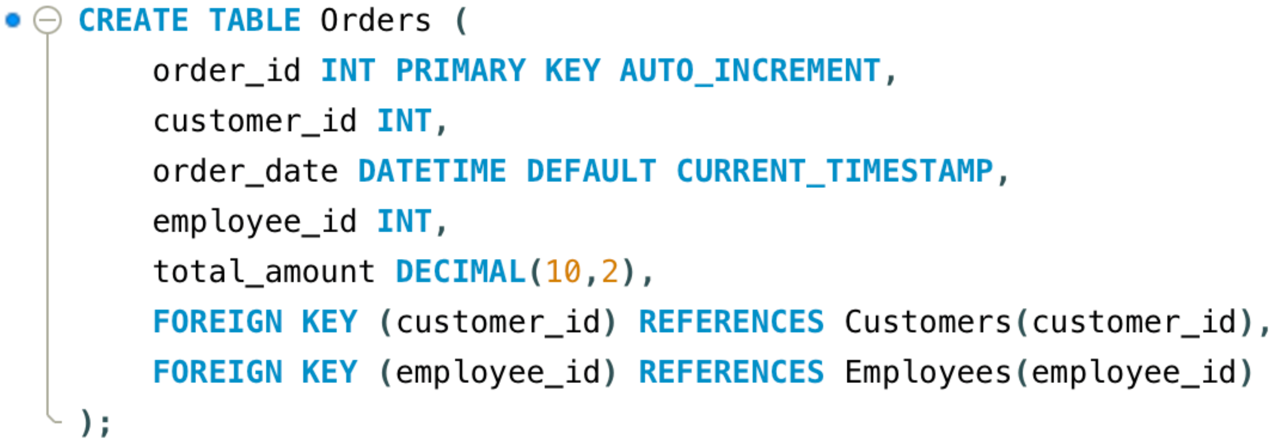


```
CREATE TABLE Menu (
    item_id INT PRIMARY KEY AUTO_INCREMENT,
    item_name VARCHAR(100),
    category VARCHAR(50),
    price DECIMAL(8, 2),
    availability BOOLEAN
);
```

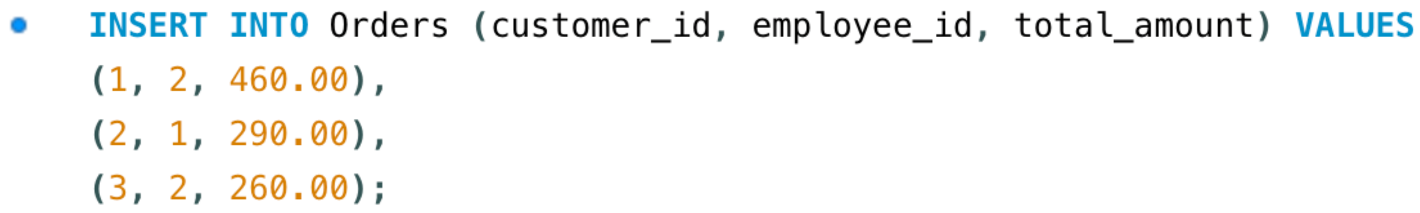


```
INSERT INTO Menu (item_name, category, price, availability) VALUES
('Paneer Butter Masala', 'Main', 250.00, TRUE),
('Masala Dosa', 'Main', 120.00, TRUE),
('Gulab Jamun', 'Dessert', 60.00, TRUE),
('Lassi', 'Drink', 50.00, TRUE),
('Veg Manchurian', 'Starter', 150.00, TRUE),
('Dal Makhani', 'Main', 200.00, FALSE);
```

4. Orders Table

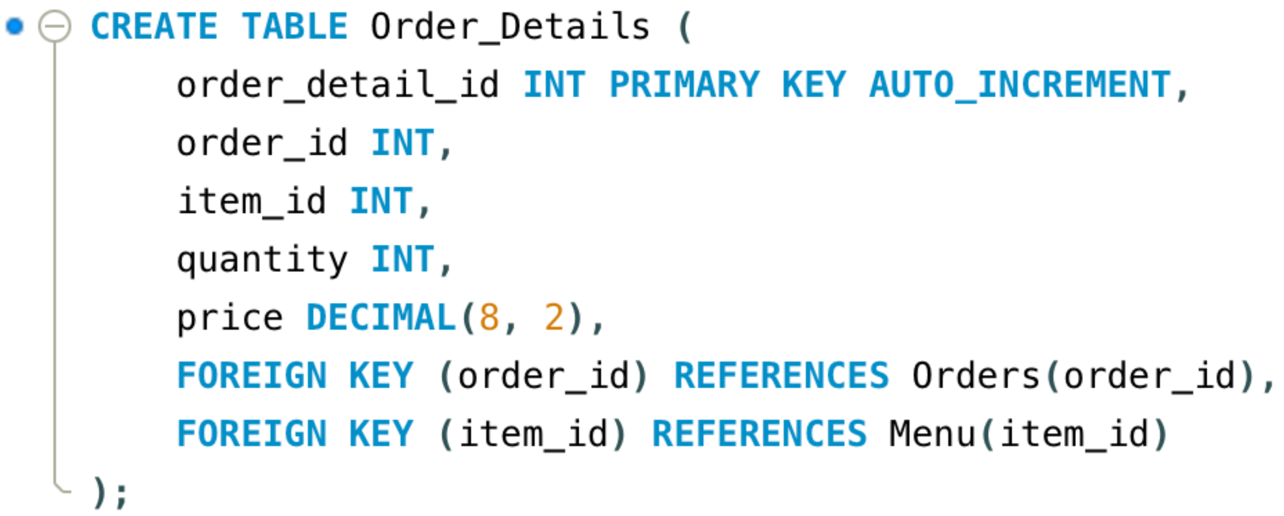


```
CREATE TABLE Orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    employee_id INT,
    total_amount DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
```

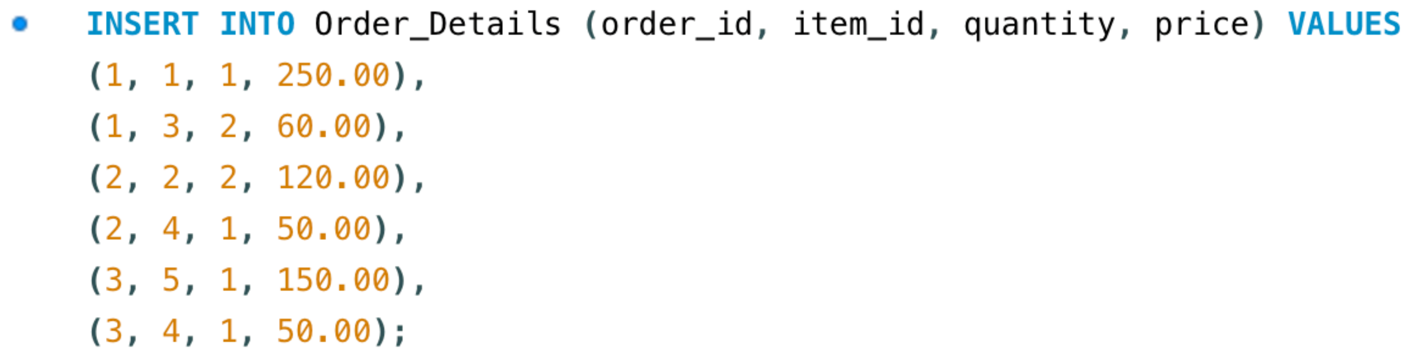


```
INSERT INTO Orders (customer_id, employee_id, total_amount) VALUES
(1, 2, 460.00),
(2, 1, 290.00),
(3, 2, 260.00);
```

5. Order Details Table

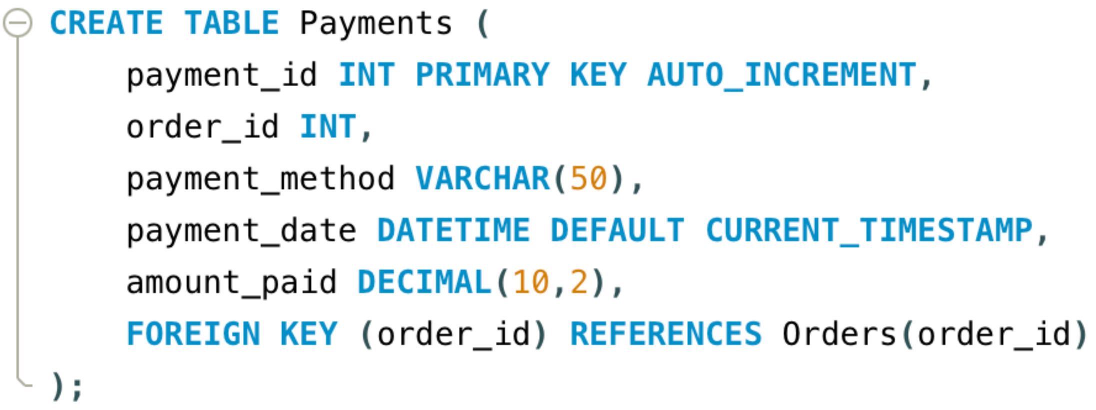


```
CREATE TABLE Order_Details (
    order_detail_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT,
    item_id INT,
    quantity INT,
    price DECIMAL(8, 2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (item_id) REFERENCES Menu(item_id)
);
```

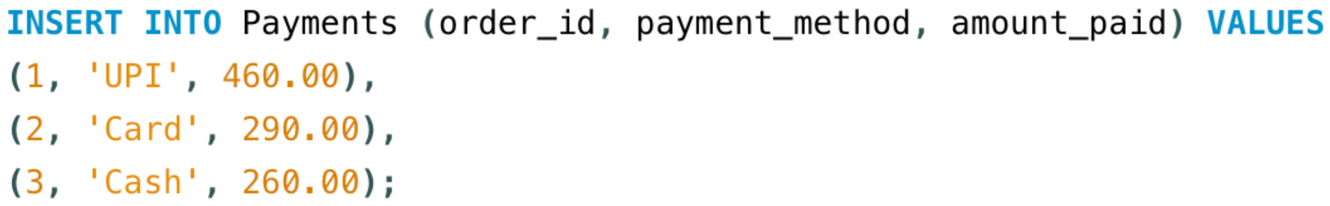


```
INSERT INTO Order_Details (order_id, item_id, quantity, price) VALUES
(1, 1, 1, 250.00),
(1, 3, 2, 60.00),
(2, 2, 2, 120.00),
(2, 4, 1, 50.00),
(3, 5, 1, 150.00),
(3, 4, 1, 50.00);
```

6. Payment Table

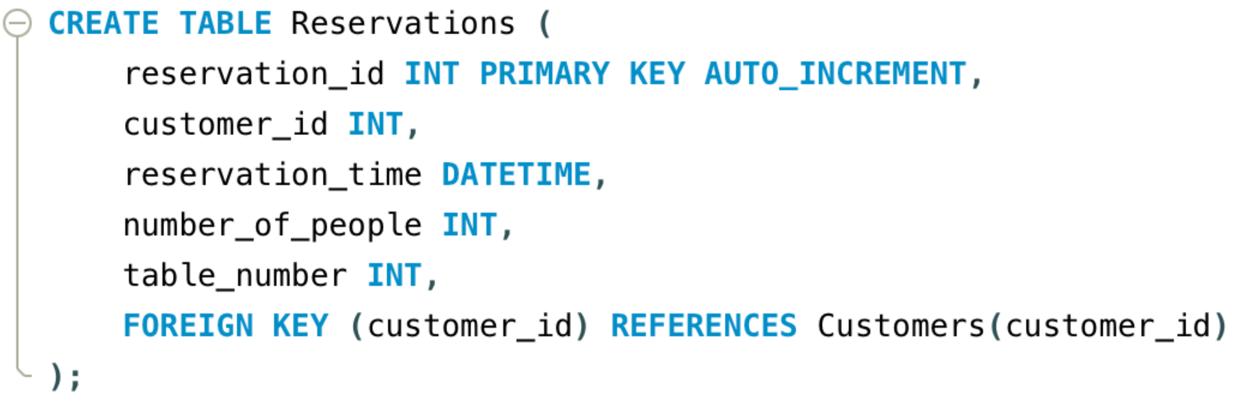


```
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT,
    payment_method VARCHAR(50),
    payment_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    amount_paid DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
```

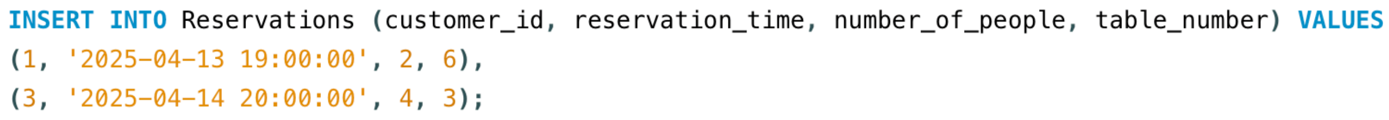


```
INSERT INTO Payments (order_id, payment_method, amount_paid) VALUES
(1, 'UPI', 460.00),
(2, 'Card', 290.00),
(3, 'Cash', 260.00);
```

7. Reservation Table

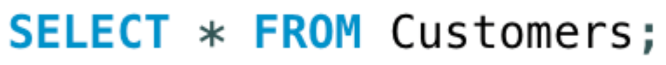


```
CREATE TABLE Reservations (
    reservation_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    reservation_time DATETIME,
    number_of_people INT,
    table_number INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

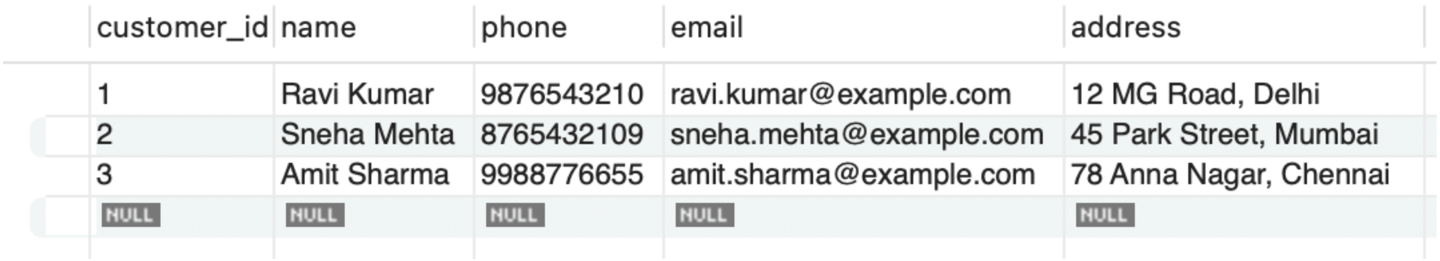


```
INSERT INTO Reservations (customer_id, reservation_time, number_of_people, table_number) VALUES
(1, '2025-04-13 19:00:00', 2, 6),
(3, '2025-04-14 20:00:00', 4, 3);
```

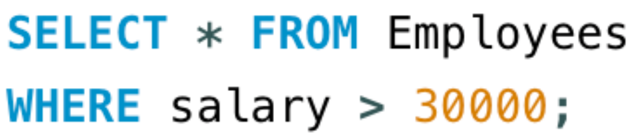
SQL QUERIES (10 Queries)



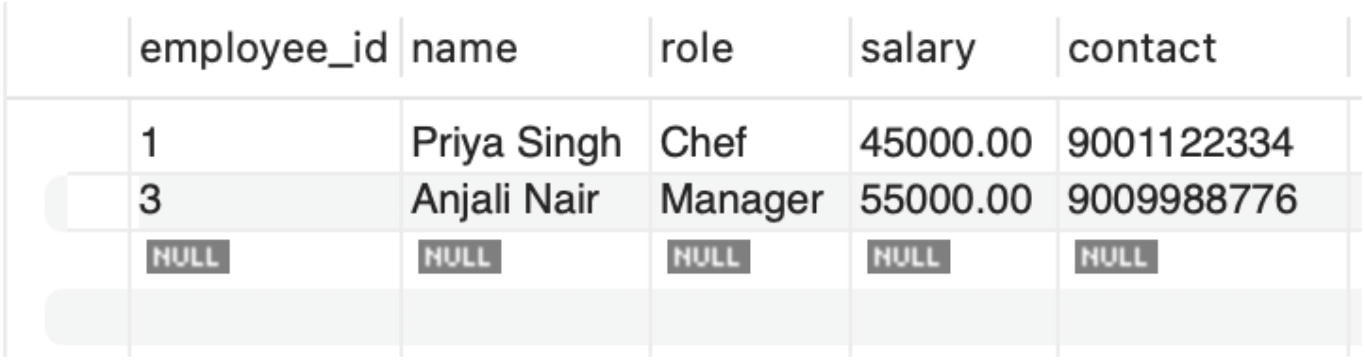
```
SELECT * FROM Customers;
```



customer_id	name	phone	email	address
1	Ravi Kumar	9876543210	ravi.kumar@example.com	12 MG Road, Delhi
2	Sneha Mehta	8765432109	sneha.mehta@example.com	45 Park Street, Mumbai
3	Amit Sharma	9988776655	amit.sharma@example.com	78 Anna Nagar, Chennai
NULL	NULL	NULL	NULL	NULL

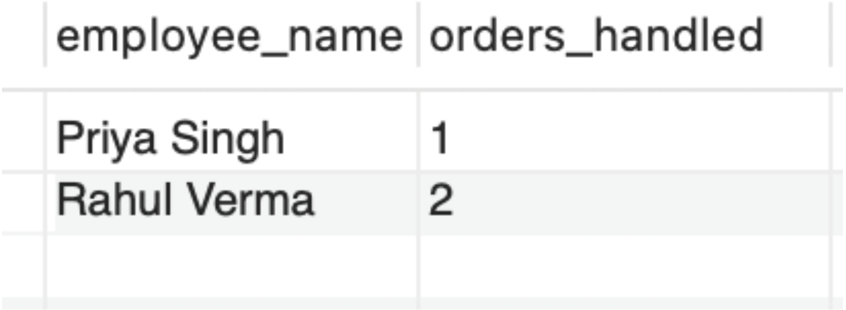


```
SELECT * FROM Employees
WHERE salary > 30000;
```



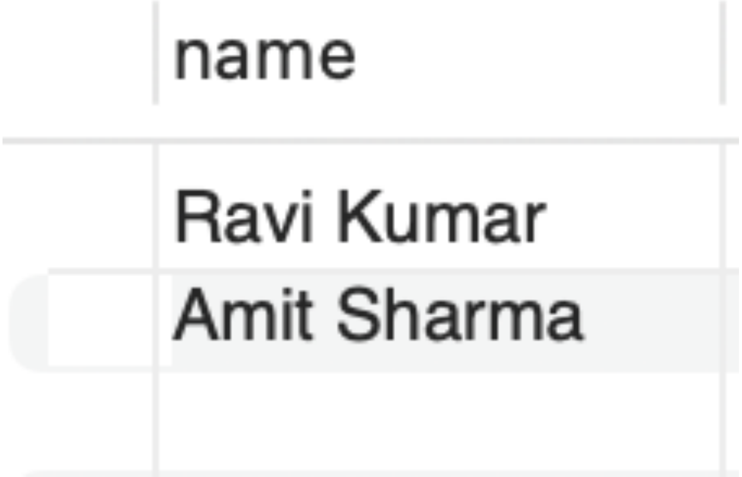
employee_id	name	role	salary	contact
1	Priya Singh	Chef	45000.00	9001122334
3	Anjali Nair	Manager	55000.00	9009988776
NULL	NULL	NULL	NULL	NULL

-
  ```
  SELECT e.name AS employee_name, COUNT(o.order_id) AS orders_handled
  FROM Employees e
  JOIN Orders o ON e.employee_id = o.employee_id
  GROUP BY e.name;
  ```

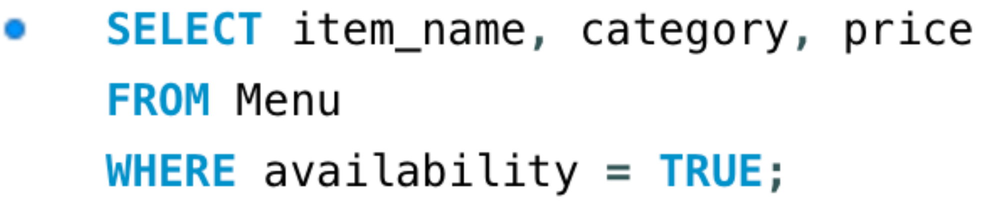


employee_name	orders_handled
Priya Singh	1
Rahul Verma	2

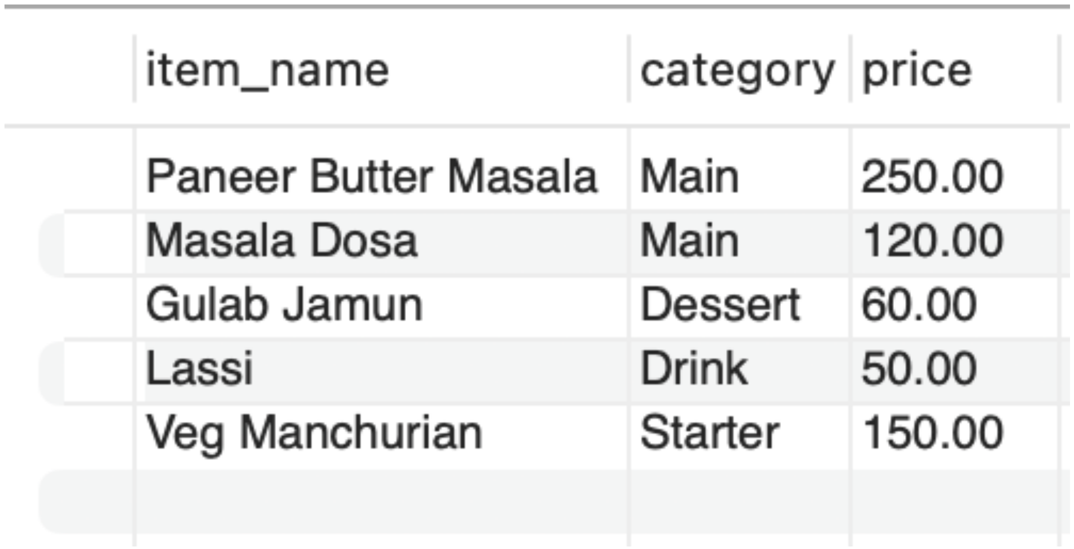
-
  ```
  SELECT DISTINCT c.name
  FROM Customers c
  JOIN Orders o ON c.customer_id = o.customer_id
  JOIN Reservations r ON c.customer_id = r.customer_id;
  ```



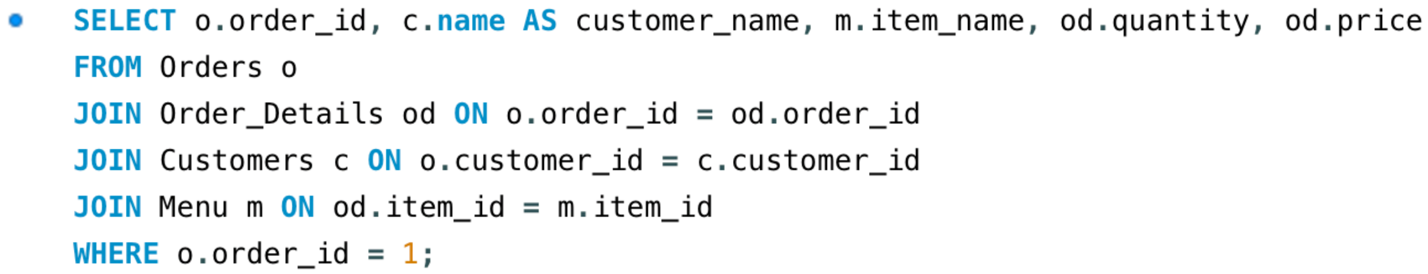
name
Ravi Kumar
Amit Sharma



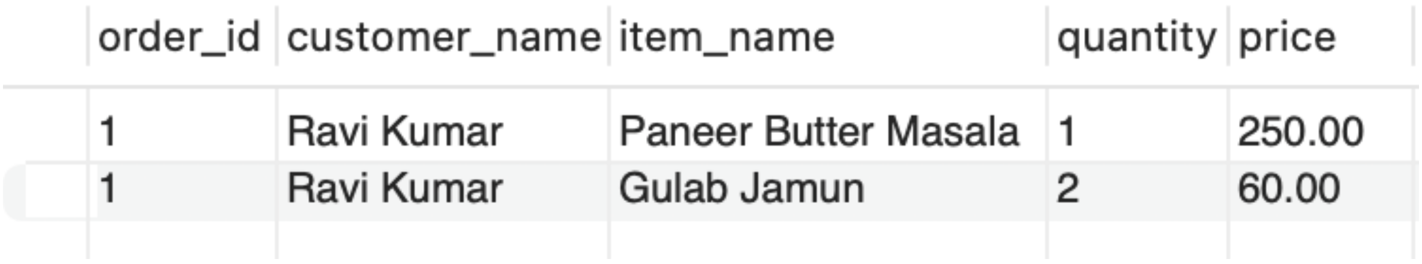
```
SELECT item_name, category, price
FROM Menu
WHERE availability = TRUE;
```



item_name	category	price
Paneer Butter Masala	Main	250.00
Masala Dosa	Main	120.00
Gulab Jamun	Dessert	60.00
Lassi	Drink	50.00
Veg Manchurian	Starter	150.00

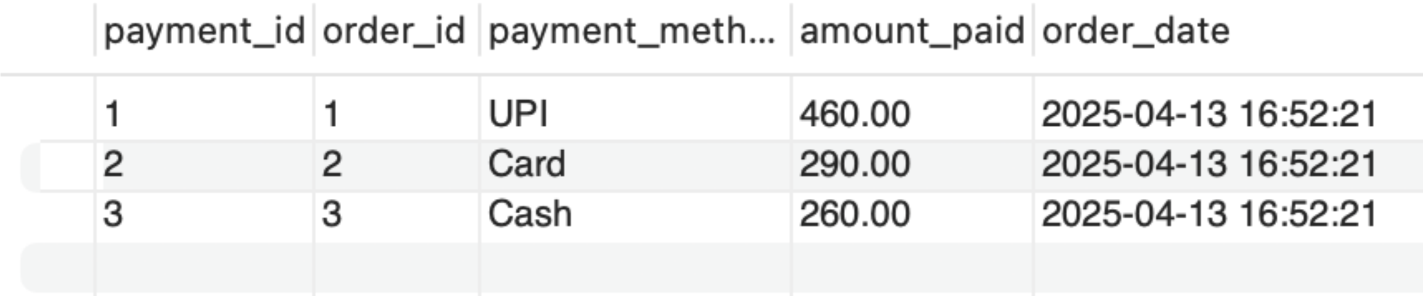


```
SELECT o.order_id, c.name AS customer_name, m.item_name, od.quantity, od.price
FROM Orders o
JOIN Order_Details od ON o.order_id = od.order_id
JOIN Customers c ON o.customer_id = c.customer_id
JOIN Menu m ON od.item_id = m.item_id
WHERE o.order_id = 1;
```



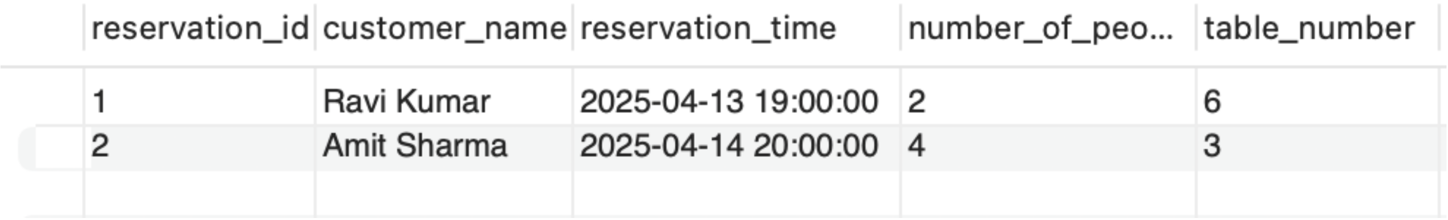
order_id	customer_name	item_name	quantity	price
1	Ravi Kumar	Paneer Butter Masala	1	250.00
1	Ravi Kumar	Gulab Jamun	2	60.00

-
  ```
  SELECT p.payment_id, p.order_id, p.payment_method, p.amount_paid, o.order_date
  FROM Payments p
  JOIN Orders o ON p.order_id = o.order_id;
  ```

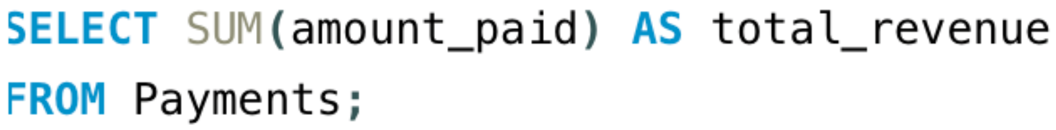


payment_id	order_id	payment_meth...	amount_paid	order_date
1	1	UPI	460.00	2025-04-13 16:52:21
2	2	Card	290.00	2025-04-13 16:52:21
3	3	Cash	260.00	2025-04-13 16:52:21

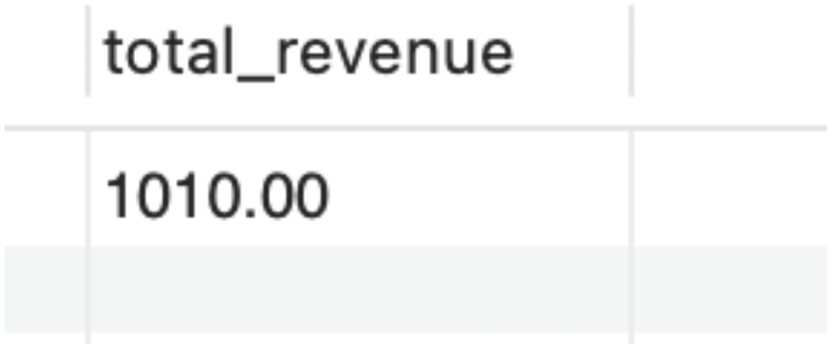
-
  ```
  SELECT r.reservation_id, c.name AS customer_name, r.reservation_time, r.number_of_people, r.table_number
  FROM Reservations r
  JOIN Customers c ON r.customer_id = c.customer_id;
  ```



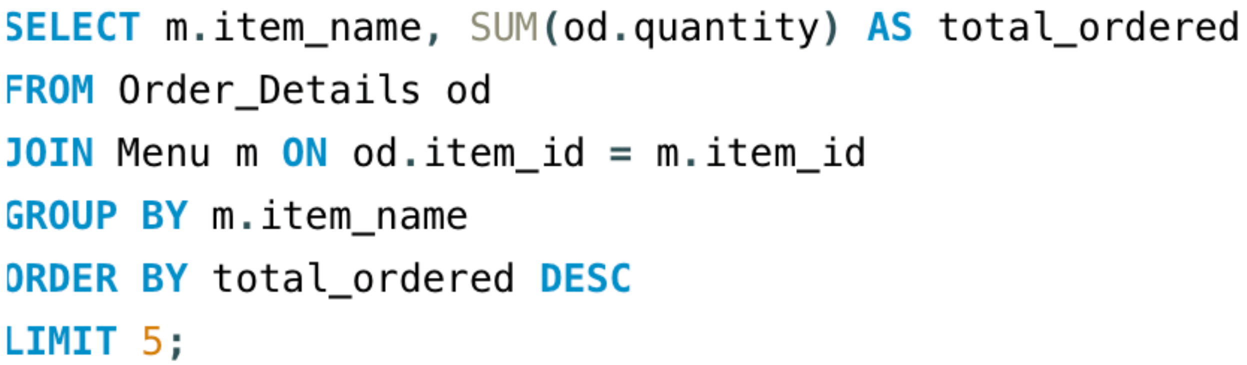
reservation_id	customer_name	reservation_time	number_of_peo...	table_number
1	Ravi Kumar	2025-04-13 19:00:00	2	6
2	Amit Sharma	2025-04-14 20:00:00	4	3

- 

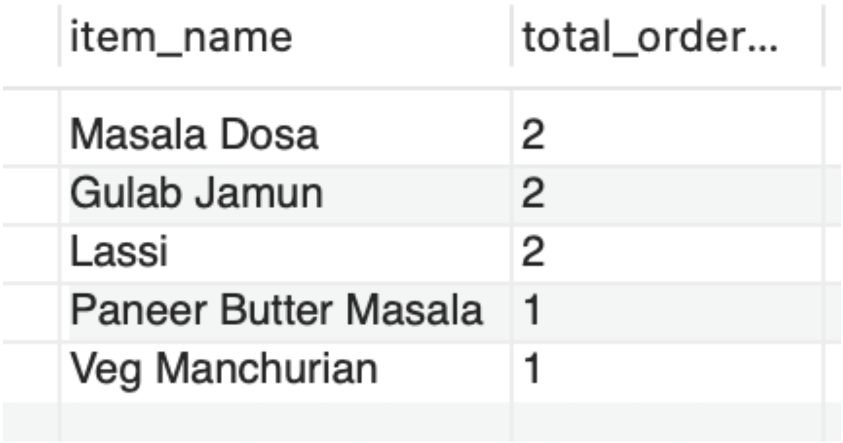
```
SELECT SUM(amount_paid) AS total_revenue
FROM Payments;
```



total_revenue
1010.00

- 

```
SELECT m.item_name, SUM(od.quantity) AS total_ordered
FROM Order_Details od
JOIN Menu m ON od.item_id = m.item_id
GROUP BY m.item_name
ORDER BY total_ordered DESC
LIMIT 5;
```



item_name	total_order...
Masala Dosa	2
Gulab Jamun	2
Lassi	2
Paneer Butter Masala	1
Veg Manchurian	1

SUMMARY

The **Restaurant Management System Database** is designed to efficiently handle and manage all core operations of a restaurant, including customer management, employee management, menu listings, order processing, payments, and reservations.

The system comprises **seven primary tables**:

1. **Customers** – Maintains detailed records of customers such as name, contact number, email, and address. Each customer can place multiple orders and make reservations.
2. **Employees** – Stores employee data including their name, role, contact, and salary. Employees are responsible for handling various orders.
3. **Menu** – Contains the list of food and beverage items offered by the restaurant. Each item has a name, category (e.g., main, dessert, drink), price, and availability status.
4. **Orders** – Keeps track of customer orders, linking each order to the respective customer and the employee who managed it. It also records the total amount of the order.
5. **Order_Details** – Provides itemized information of each order. It connects orders to specific menu items along with quantity and individual item prices.
6. **Payments** – Records payment transactions for each order, including the payment method (e.g., UPI, Card, Cash), the amount paid, and the date of payment.
7. **Reservations** – Manages table reservations by customers, storing data like reservation time, number of people, and allocated table number.

These interconnected tables ensure a smooth flow of information and support all major functionalities required in restaurant operations — from placing and tracking orders to managing staff and payments. The relationships between the tables enforce data integrity and enable robust reporting and analysis.

CONCLUSION

Observations

- The database is well-structured and scalable, which allows for easy additions such as new menu items, customers, or employees.
- The use of foreign keys ensures data consistency and integrity across tables. For example, the connection between `Orders`, `Order_Details`, and `Payments` guarantees the accuracy of transaction data.
- The relationships between tables, such as `Customers` with `Orders` and `Reservations`, ensure all aspects of customer interaction are covered.
- The system provides essential reporting capabilities like tracking total revenue, most ordered menu items, and employee performance.

Limitations

- **Lack of Real-Time Integration**: The system does not handle real-time updates (e.g., live inventory tracking or dynamic menu updates), which may be a limitation in a high-volume, fast-paced restaurant environment.
- **Limited User Roles**: The current system only stores basic employee roles. More granular roles (e.g., kitchen staff, cashier, delivery staff) and access control could enhance functionality.
- **Data Redundancy**: There is potential for redundant data entry, especially in terms of customer and employee details, if not properly maintained or managed.
- **Scalability Concerns**: As the restaurant grows or diversifies (e.g., adding online ordering), the system may require modifications to handle new features like delivery or multi-location operations.

Future Scope

- **Real-Time Inventory Management**: The system can be expanded to include inventory management to track the availability of ingredients in real-time and update menu availability accordingly.
- **Online Ordering and Delivery Integration**: Integrating an online ordering system could allow customers to place orders directly through a website or mobile app, which would sync with the database for order management and payment processing.
- **Dynamic Pricing and Discounts**: Implementing dynamic pricing models, promotional discounts, and loyalty programs can enhance the customer experience and improve sales.
- **Advanced Reporting and Analytics**: More sophisticated analytics features could provide detailed insights into sales trends, customer preferences, and operational efficiency.
- **Multiple Location Support**: Expanding the system to support multiple restaurant locations, allowing centralized management and data consolidation across various branches.