# ASSIGNMENT - 4

# DATABASE MANAGEMENT SYSTEM

**SIDDARTH.P**

**192324264**

# Restaurant Order and Inventory Management System

## Description

The **Restaurant Order and Inventory Management System** (ROIMS) database is designed to manage customer orders, menu items, inventory tracking, and supplier data. This system ensures smooth operations by maintaining accurate ingredient stock levels, enabling real-time updates to inventory upon order placement, and generating insightful reports on popular dishes, revenue, stock levels, and supplier performance. The system includes constraints to prevent over-ordering, triggers for inventory updates, and stored procedures for managing orders and inventory restocking.

---

## Database Tables Design

### 1. Customers

Stores customer details for order processing and communication.

| Column | Data Type | Description |
|---|---|---|
| customer_id | INT (PK) | Unique identifier for each customer |
| first_name | VARCHAR(50) | Customer's first name |
| last_name | VARCHAR(50) | Customer's last name |
| phone_number | VARCHAR(15) | Customer's contact number |
| email | VARCHAR(100) | Customer's email address |

**2. Orders**

Records details of each order placed by customers.

| Column | Data Type | Description |
|---|---|---|
| order_id | INT (PK) | Unique identifier for each order |
| customer_id | INT (FK) | References Customers(customer_id) |
| order_date | DATETIME | Date and time of order |
| total_amount | DECIMAL(10,2) | Total order amount |
| status | VARCHAR(20) | Order status (e.g., Pending, Completed) |

**3. Menu_Items**

Tracks details of items available on the menu.

| Column | Data Type | Description |
|---|---|---|
| menu_item_id | INT (PK) | Unique identifier for each menu item |
| name | VARCHAR(100) | Name of the menu item |
| price | DECIMAL(10,2) | Price of the menu item |
| description | TEXT | Description of the menu item |

**4. Inventory**

Keeps track of ingredients and their quantities.

| Column | Data Type | Description |
|---|---|---|
| ingredient_id | INT (PK) | Unique identifier for each ingredient |
| name | VARCHAR(100) | Name of the ingredient |
| quantity | DECIMAL(10,2) | Current stock level |
| unit_price | DECIMAL(10,2) | Price per unit of ingredient |
| supplier_id | INT (FK) | References Suppliers(supplier_id) |

**5. Staff**

Stores details of restaurant staff.

| Column | Data Type | Description |
|---|---|---|
| staff_id | INT (PK) | Unique identifier for each staff |
| first_name | VARCHAR(50) | Staff's first name |
| last_name | VARCHAR(50) | Staff's last name |
| position | VARCHAR(50) | Staff's position (e.g., Waiter, Chef) |
| hire_date | DATETIME | Date the staff member was hired |

**6. Suppliers**

Tracks suppliers for ingredients.

| Column | Data Type | Description |
|---|---|---|
| supplier_id | INT (PK) | Unique identifier for each supplier |
| name | VARCHAR(100) | Name of the supplier |
| contact_name | VARCHAR(50) | Contact person at the supplier |
| phone_number | VARCHAR(15) | Supplier's contact number |
| email | VARCHAR(100) | Supplier's email address |

**Constraints for Referential Integrity**

- **Foreign Keys:**
    - customer_id in **Orders** references **Customers(customer_id)**.
    - supplier_id in **Inventory** references **Suppliers(supplier_id)**.
- **Primary Keys:**
    - Each table includes a primary key for unique record identification.
- **Check Constraints:**
    - Ensure positive ingredient quantities and valid order amounts to prevent inconsistencies.
    - Prevent over-ordering by limiting order quantity based on available stock in **Inventory**.

**Stored Procedures**

**a. Place Order**

Registers an order and updates inventory levels.

```
CREATE PROCEDURE PlaceOrder(
  IN customerId INT,
  IN menuItemId INT,
  IN quantity INT
)
BEGIN
  DECLARE price DECIMAL(10,2);
  DECLARE ingredientRequired DECIMAL(10,2);

  -- Get the price of the menu item
  SELECT price INTO price FROM Menu_Items WHERE menu_item_id = menuItemId;
```

```
  -- Calculate total amount for the order
  INSERT INTO Orders (customer_id, order_date, total_amount, status)
  VALUES (customerId, NOW(), price * quantity, 'Pending');

  -- Update inventory for ingredients used in the menu item
  SELECT required_ingredients INTO ingredientRequired
  FROM Menu_Items WHERE menu_item_id = menuItemId;

  -- Check if sufficient ingredients are available in inventory
  IF ingredientRequired <= (SELECT quantity FROM Inventory WHERE ingredient_id =
ingredientRequired) THEN
    UPDATE Inventory
    SET quantity = quantity - ingredientRequired
    WHERE ingredient_id = ingredientRequired;
  ELSE
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Not enough ingredients in stock';
  END IF;
END;
```

**b. Check Ingredient Levels**

Confirms if there is enough stock for the menu item ingredients.

```
CREATE PROCEDURE CheckIngredientLevels(IN menuItemId INT)
BEGIN
  DECLARE ingredientRequired DECIMAL(10,2);
  DECLARE availableStock DECIMAL(10,2);

  -- Get required ingredient and available stock
```

```
  SELECT required_ingredients INTO ingredientRequired FROM Menu_Items WHERE menu_item_id
= menuItemId;
  SELECT quantity INTO availableStock FROM Inventory WHERE ingredient_id =
ingredientRequired;

  IF availableStock < ingredientRequired THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Insufficient ingredient stock';
  END IF;
END;
```

**c. Restock Inventory**

Restocks ingredients when low stock is detected.

```
CREATE PROCEDURE RestockInventory(IN ingredientId INT, IN quantity INT)
BEGIN
  UPDATE Inventory
  SET quantity = quantity + quantity
  WHERE ingredient_id = ingredientId;
END;
```

**Triggers**

**a. Update Inventory on Order**

Automatically updates inventory when an order is placed.

```
CREATE TRIGGER AfterOrderInsert
AFTER INSERT ON Orders
FOR EACH ROW
BEGIN
```

```
    DECLARE required_ingredients INT;

    -- Loop through all ordered items and update inventory
    SELECT required_ingredients INTO required_ingredients
    FROM Menu_Items WHERE menu_item_id = NEW.menu_item_id;

    UPDATE Inventory
    SET quantity = quantity - required_ingredients
    WHERE ingredient_id = required_ingredients;
END;
```

**b. Update Inventory on Restock**

Automatically updates inventory when ingredients are restocked.

```
CREATE TRIGGER AfterRestockInsert
AFTER INSERT ON Inventory
FOR EACH ROW
BEGIN
    UPDATE Inventory
    SET quantity = quantity + NEW.quantity
    WHERE ingredient_id = NEW.ingredient_id;
END;
```

**SQL Queries for Reports**

**a. Popular Dishes Report**

Generates a report of the most frequently ordered menu items.

```
SELECT menu_item_id, COUNT(order_id) AS total_orders
FROM Orders
GROUP BY menu_item_id
ORDER BY total_orders DESC;
```

**b. Daily Revenue**

Generates daily revenue report.

```
SELECT DATE(order_date) AS date, SUM(total_amount) AS daily_revenue
FROM Orders
GROUP BY DATE(order_date)
ORDER BY date DESC;
```

**c. Stock Levels**

Checks the current stock levels of all ingredients.

```
SELECT ingredient_id, name, quantity
FROM Inventory;
```

**d. Supplier Performance**

Assesses supplier performance based on ingredient stock levels and restocks.

```
SELECT s.supplier_id, s.name, SUM(i.quantity) AS total_stock
FROM Suppliers s
JOIN Inventory i ON s.supplier_id = i.supplier_id
GROUP BY s.supplier_id;
```

## Conclusion

The **Restaurant Order and Inventory Management System** database is a comprehensive solution that integrates order processing, inventory management, and supplier tracking. The system ensures efficient operations by automating inventory updates through triggers and stored procedures, preventing stock shortages, and enabling real-time reporting on sales, stock levels, and supplier performance. This design helps restaurant owners optimize resources, track ingredient usage, and maintain profitability while ensuring customer satisfaction through accurate order fulfillment and timely restocking.