ASSIGNMENT-5

DATABASE MANAGEMENT SYSTEM
CSA0593

SATHISH.R
192324204

Restaurant Order and Inventory Management System: Database Design

Description:

The Restaurant Order and Inventory Management System (ROIMS) is designed to streamline the management of restaurant orders, menu items, inventory, and customer data. The system will allow restaurant staff to place and track customer orders, monitor ingredient levels, and manage restocking needs efficiently. It will include tables for managing customer information, orders, menu items, inventory, suppliers, and staff. The system will also provide constraints to ensure that ingredients are always in stock, stored procedures for order processing and inventory checks, and triggers to automatically update ingredient stock levels. Additionally, the system will generate reports on popular dishes, daily revenue, stock levels, and supplier performance, providing restaurant managers with valuable insights to improve operations.

Database Design:

Customers Table
This table stores customer data, including contact details.

Sql

```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(255) UNIQUE,
    phone VARCHAR(15)
);
```

Menu Items Table
This table stores information about each menu item, including the dish name, description, price, and associated ingredients.

Sql

```
CREATE TABLE Menu_Items (
    menu_item_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255),
    description TEXT,
    price DECIMAL(10, 2),
    category VARCHAR(100)
```

```
);
```

Inventory Table
This table tracks the inventory of ingredients used for menu items, including stock levels and reorder thresholds.

Sql

```
CREATE TABLE Inventory (
    ingredient_id INT PRIMARY KEY AUTO_INCREMENT,
    ingredient_name VARCHAR(255),
    quantity INT,
    reorder_level INT,
    supplier_id INT,
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id)
);
```

Suppliers Table
Stores information about ingredient suppliers, including their contact details.

Sql

```
CREATE TABLE Suppliers (
    supplier_id INT PRIMARY KEY AUTO_INCREMENT,
    supplier_name VARCHAR(255),
    contact_name VARCHAR(100),
    contact_email VARCHAR(255),
    contact_phone VARCHAR(15)
);
```

Orders Table
Tracks customer orders, including the customer, order date, and status.

Sql

```
CREATE TABLE Orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(50),  -- e.g., 'Pending', 'Completed', 'Cancelled'
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

Order Items Table
This table records the specific items ordered within an order, linking each item to the order and the menu item.

Sql

```
CREATE TABLE Order_Items (
    order_item_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT,
    menu_item_id INT,
    quantity INT,
    price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (menu_item_id) REFERENCES Menu_Items(menu_item_id)
);
```

Staff Table
This table holds details about restaurant staff, including their roles.

Sql

```
CREATE TABLE Staff (
    staff_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    role VARCHAR(50),  -- e.g., 'Chef', 'Waiter', 'Manager'
    hire_date DATE
);
```

Key Constraints and Features:

Ingredient Stock Levels
The Inventory table ensures that the stock of each ingredient is tracked and that a reorder_level is defined. If ingredients fall below this level, the system can trigger a restock request.
Preventing Over-ordering
Constraints on the Order_Items table ensure that the quantity of ordered items does not exceed available stock. This can be implemented via stored procedures or triggers.

Stored Procedures:

Place Order Procedure:

This stored procedure processes an order by updating the order status, adding the ordered items, and updating ingredient stock levels accordingly.

Sql

```
CREATE PROCEDURE PlaceOrder(IN customer_id INT, IN order_items JSON)BEGIN
    DECLARE total DECIMAL(10,2);
    DECLARE ingredient_quantity INT;
    DECLARE ingredient_id INT;
    DECLARE quantity INT;

    -- Initialize the total order amount
    SET total = 0;

    -- Create the new order
    INSERT INTO Orders (customer_id, status, total_amount)
    VALUES (customer_id, 'Pending', 0);

    -- Loop through the order items (JSON format)
    FOR item IN JSON_TABLE(order_items, '$[*]' COLUMNS (menu_item_id INT PATH
'$.menu_item_id', quantity INT PATH '$.quantity')) DO
        -- Add the order item
        INSERT INTO Order_Items (order_id, menu_item_id, quantity, price)
        VALUES (LAST_INSERT_ID(), item.menu_item_id, item.quantity, (SELECT price
FROM Menu_Items WHERE menu_item_id = item.menu_item_id));

        -- Update the total order amount
        SET total = total + (SELECT price FROM Menu_Items WHERE menu_item_id =
item.menu_item_id) * item.quantity;

        -- Update inventory levels
        SELECT ingredient_id, quantity INTO ingredient_id, ingredient_quantity
        FROM Ingredients WHERE menu_item_id = item.menu_item_id;

        -- Check inventory stock and update
        IF ingredient_quantity >= item.quantity THEN
            UPDATE Inventory SET quantity = quantity - item.quantity WHERE ingredient_id =
ingredient_id;
        ELSE
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Not enough stock for
ingredient: ' + (SELECT ingredient_name FROM Inventory WHERE ingredient_id =
ingredient_id);
        END IF;
    END FOR;

    -- Update the order total
    UPDATE Orders SET total_amount = total WHERE order_id =
LAST_INSERT_ID();END;
```

Restock Inventory Procedure:

This procedure handles the restocking of ingredients when their quantity falls below the reorder level.

Sql

```
CREATE PROCEDURE RestockInventory(IN ingredient_id INT, IN quantity INT)BEGIN
    -- Check if the ingredient exists
    IF EXISTS (SELECT 1 FROM Inventory WHERE ingredient_id = ingredient_id) THEN
        -- Update inventory with new stock level
        UPDATE Inventory SET quantity = quantity + quantity WHERE ingredient_id =
ingredient_id;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ingredient not found in
inventory';
    END IF;END;
```

Triggers:

Trigger to Update Ingredient Quantity on Order:

This trigger automatically updates the ingredient stock levels when an item is ordered.

Sql

```
CREATE TRIGGER UpdateIngredientStock
AFTER INSERT ON Order_ItemsFOR EACH ROWBEGIN
    DECLARE ingredient_id INT;
    DECLARE ingredient_quantity INT;

    -- Get ingredient ID and quantity from Menu_Items
    SELECT ingredient_id, quantity INTO ingredient_id, ingredient_quantity
    FROM Menu_Items
    WHERE menu_item_id = NEW.menu_item_id;

    -- Update ingredient quantity in inventory
    UPDATE Inventory SET quantity = quantity - (NEW.quantity * ingredient_quantity)
WHERE ingredient_id = ingredient_id;END;
```

SQL Queries:

Popular Dishes Query:

This query retrieves the top 5 most ordered menu items.

Sql

```
SELECT mi.name, COUNT(oi.menu_item_id) AS order_countFROM Order_Items oiJOIN Menu_Items mi ON oi.menu_item_id = mi.menu_item_idGROUP BY mi.nameORDER BY order_count DESC
LIMIT 5;
```

Daily Revenue Query:

This query calculates the total revenue for the restaurant on a given day.

Sql

```
SELECT SUM(total_amount) AS daily_revenueFROM OrdersWHERE DATE(order_date) = CURDATE() AND status = 'Completed';
```

Stock Level Report:

This query shows the current stock levels of all ingredients.

Sql

```
SELECT ingredient_name, quantity, reorder_levelFROM Inventory;
```

Supplier Performance Report:

This query tracks supplier performance based on the frequency of ingredient restocks.

Sql

```
SELECT s.supplier_name, COUNT(i.ingredient_id) AS ingredient_countFROM Suppliers sJOIN Inventory i ON s.supplier_id = i.supplier_idGROUP BY s.supplier_name;
```

Conclusion:

The Restaurant Order and Inventory Management System is designed to improve restaurant operations by efficiently handling orders, managing inventory, and tracking customer and supplier information. Through the use of key constraints, stored procedures, and triggers, the system ensures that inventory levels are properly maintained and that orders are processed accurately. The system also generates valuable reports to help restaurant managers analyze popular dishes, daily revenue, stock levels, and supplier performance, providing actionable insights to optimize operations. This database design offers a comprehensive solution to managing restaurant workflows, improving both customer service and inventory control.