SMART RESTAURANT ORDER & KITCHEN MANAGEMENT SYSTEM

PL/SQL Practicum Project

Student Name: Tricia Nshuti
Student ID: 27253
Course: PL/SQL
Date: November 13, 2025

1 PROJECT OVERVIEW

This project develops a complete restaurant management system that handles everything from customer orders to kitchen operations and inventory control. The system connects waiters, kitchen staff, and managers in real-time, making restaurant operations smoother and reducing errors that happen with manual order taking.

2 PROBLEM STATEMENT

From visiting restaurants in Kigali, I've noticed several problems with how orders are managed:

- Waiters write orders on paper which the kitchen sometimes can't read clearly
- Kitchen doesn't know which orders to prepare first during busy hours
- Ingredients run out suddenly because there's no tracking system
- Customers wait too long and sometimes their orders get forgotten
- Wrong items are sometimes delivered to tables
- It's difficult to know which menu items are most profitable
- No way to send order ready notifications to waiters

These issues lead to customer complaints, wasted food, lost revenue, and stressed staff. I want to solve this using a database system with smart automation.

3 MY PROPOSED SOLUTION

I will build an intelligent restaurant system with these features:

Core Features:

- **Digital Order Taking** - waiters enter orders directly into the system
- **Smart Kitchen Queue** - automatically prioritizes orders based on preparation time and order time
- **Auto Stock Deduction** - ingredients are automatically reduced when orders are placed
- **Order Ready Notifications** - sends SMS alerts to waiters when food is ready (simulated)
- **Customer Table Management** - tracks which tables are occupied and their current orders

- **Low Stock Alerts** - warns when ingredients are running out
- **Daily Sales Reports** - shows revenue, popular items, and kitchen performance
- **Bill Splitting** - handles split bills for groups at same table

Key Innovations:

1. **Intelligent Kitchen Queue Algorithm** - orders are prioritized by prep time and waiting duration
2. **Real-time SMS Notifications** - waiters get alerts when orders are ready to serve (simulated)
3. **Automatic Ingredient Validation** - blocks orders if ingredients are insufficient
4. **Recipe-Based Stock Calculation** - system knows exactly how much of each ingredient each dish needs
5. **Customer Wait Time Tracking** - monitors and alerts if orders take too long
6. **Peak Hours Analysis** - identifies busy times to optimize staff scheduling

4 DATABASE DESIGN

I will create 10 interconnected tables:

Main Tables:

CUSTOMERS - customer information: customer_id (PK), first_name, last_name, phone_number, email, loyalty_points, registration_date, total_visits

TABLES - restaurant tables: table_id (PK), table_number, seating_capacity, status (AVAILABLE/OCCUPIED/RESERVED), current_customer_id (FK), last_occupied_time

MENU_ITEMS - menu details: menu_id (PK), item_name, category (APPETIZER/MAIN/DESSERT/BEVERAGE), price, preparation_time_minutes, is_available, description

INGREDIENTS - ingredient inventory: ingredient_id (PK), ingredient_name, unit_of_measure, stock_quantity, reorder_level, unit_cost, expiry_date, supplier_name

RECIPES - ingredient requirements: recipe_id (PK), menu_id (FK), ingredient_id (FK), quantity_needed, notes

ORDERS - customer orders: order_id (PK), table_id (FK), customer_id (FK), waiter_name, order_date, order_time, status (PENDING/PREPARING/READY/SERVED/PAID), total_amount

ORDER_DETAILS - specific items ordered: detail_id (PK), order_id (FK), menu_id (FK), quantity, special_instructions, item_status, price_at_order

KITCHEN_QUEUE - preparation queue: queue_id (PK), order_id (FK), menu_id (FK), quantity, priority_score, received_time, started_time, completed_time, assigned_chef

SMS_NOTIFICATIONS - order ready alerts: notification_id (PK), order_id (FK), waiter_phone, message_text, notification_time, sent_status (PENDING/SENT), order_status

TRANSACTIONS - payment records: transaction_id (PK), order_id (FK), payment_method (CASH/CARD/MOBILE), amount_paid, payment_date, change_given

5 PL/SQL COMPONENTS I WILL USE

This project demonstrates PL/SQL concepts :

Collections

- Use VARRAY to store table numbers and their availability status
- Nested table for storing ingredients needed for each menu item
- Associative array for quick menu item lookup by category

Records

- Order summary record combining order info with customer and table details
- Kitchen queue record showing order priority and preparation status
- Daily sales record with revenue breakdown by category

GOTO Statements

- Jump to stock validation section when order is placed
- Navigate to payment processing after order completion
- Jump to error handler when insufficient ingredients detected

Cursors

- Explicit cursor to fetch all orders in kitchen queue sorted by priority
- Cursor loop to display pending orders for specific table
- Cursor to identify ingredients below reorder level
- Cursor to calculate daily revenue by menu category

Procedures

- `SP_PLACE_ORDER` - validates ingredients and creates order
- `SP_ADD_TO_KITCHEN_QUEUE` - adds items to preparation queue with priority
- `SP_COMPLETE_ORDER_ITEM` - marks item as ready and sends SMS notification
- `SP_PROCESS_PAYMENT` - handles bill payment and updates loyalty points
- `SP_GENERATE_DAILY_REPORT` - creates sales and performance report

Triggers

- `TRG_AUTO_DEDUCT_INGREDIENTS` - automatically reduces ingredient stock when order placed (AFTER INSERT)
- `TRG_VALIDATE_STOCK` - checks ingredient availability before accepting order (BEFORE INSERT)

- `TRG_LOW_STOCK_ALERT` - creates alert when ingredients fall below reorder level (AFTER UPDATE)
- `TRG_SEND_ORDER_READY_SMS` - creates SMS notification when order status changes to READY (AFTER UPDATE)
- `TRG_UPDATE_TABLE_STATUS` - marks table as occupied when order placed (AFTER INSERT)

Exception Handling

- Custom exception when insufficient ingredients for ordered item
- Handle table already occupied error
- Raise alert exception when order takes longer than expected
- Catch invalid payment amount errors
- Handle NO_DATA_FOUND when menu item or customer doesn't exist

Packages

- `PKG_ORDER_MANAGEMENT` - groups all order-related procedures
- `PKG_KITCHEN_OPERATIONS` - contains queue management and preparation tracking
- `PKG_INVENTORY_CONTROL` - manages ingredient stock and reordering

6 EXPECTED RESULTS

When I finish this system, it will:

- Eliminate handwritten order errors completely
- Speed up kitchen operations with smart queue prioritization
- Prevent running out of ingredients during service
- Reduce customer wait time by 30% through better coordination
- Alert waiters immediately when food is ready to serve
- Track exactly which menu items are most profitable
- Provide real-time visibility of restaurant operations
- Make end-of-day reporting automatic instead of manual

7 CONCLUSION

This project shows how PL/SQL database features can transform restaurant operations. By using triggers for automatic stock updates, cursors for queue management, and procedures for order processing, the system makes everything faster and more accurate.

The innovation isn't just digitizing the menu - it's adding intelligence that predicts problems (running out of ingredients), optimizes workflows (kitchen queue priority), and improves communication (SMS alerts). This makes the restaurant more efficient and customers happier.