Dineflow - User Needs Analysis

Project Overview

DineFlow DBMS is a database management system designed to streamline restaurant operations using a *payment-first model*. In this model, customers browse the menu, select their items, proceed to the cashier, and make the payment before the order is created. Once the payment is successfully processed, the system automatically generates the order record. This model is ideal for **fast-casual restaurants, cafeterias, food courts, and quick-service dining environments**, where efficiency and transaction speed are key.

Business Model

- **Payment-First Flow:** Orders are only created after payment confirmation.
- **Customer Journey:** Browse Menu → Select Items → Pay at Cashier → Order Created.
- **Not Traditional Dine-In:** Unlike conventional restaurants, orders are not made before payment.

Scope

The system manages:

- Employee management with multiple roles (lattice structure).
- Categorized menu item catalog.
- Customer registration and tracking.
- Payment processing (cash and card).
- Order creation and fulfillment.
- Basic reporting and analytics.

Development Note

This project explores the design and implementation of a restaurant ordering system built around a payment-first model. While I used AI tools to assist with structuring the EER diagram and refining the database logic, the process itself gave me a solid understanding of how a complete DBMS comes together — from identifying user needs and modeling data relationships to building, testing, and optimizing a functional database system.

2. USER ROLES & NEEDS

2.1 Cashier (Primary User)

Role Description: Processes customer payments and creates orders at point-of-sale

User Needs:

- UN-CASH-01: Quickly enter/scan menu items customer wants to purchase
- UN-CASH-02: Calculate order total (subtotal + tax) automatically
- UN-CASH-03: Process cash payments with automatic change calculation
- UN-CASH-04: Process card payments with authorization
- UN-CASH-05: Print receipt for customer
- UN-CASH-06: Create order record at moment of payment
- UN-CASH-07: Assign order number for kitchen/pickup
- UN-CASH-08: Handle special instructions for menu items
- UN-CASH-09: Void/cancel transactions if needed (with manager approval)
- UN-CASH-10: View menu items with current prices and availability

Pain Points to Solve:

- Manual calculation errors
- Slow payment processing
- Difficulty tracking which orders were paid
- Can't handle rush hour efficiently

2.2 Waiter (For Dine-In Service)

Role Description: Serves dine-in customers at tables (optional role for table service)

User Needs:

- UN-WAIT-01: View assigned orders for specific tables
- UN-WAIT-02: Track order status (pending, ready, delivered)
- UN-WAIT-03: Communicate special requests to kitchen
- UN-WAIT-04: View tips earned
- UN-WAIT-05: Know which tables need service
- UN-WAIT-06: Receive notifications when orders are ready

Pain Points to Solve:

- Confusion about which orders belong to which tables
- Not knowing when food is ready
- Tracking tips across multiple orders

2.3 Chef (Kitchen Staff)

Role Description: Prepares food items based on orders received

User Needs:

- UN-CHEF-01: View incoming orders in queue
- UN-CHEF-02: See order details (items, quantities, special instructions)
- UN-CHEF-03: Mark items as "in preparation"
- UN-CHEF-04: Mark items as "ready"
- UN-CHEF-05: Prioritize orders by time/urgency
- UN-CHEF-06: View menu items they are certified to prepare
- UN-CHEF-07: Alert system when running low on items (86'd items)

Pain Points to Solve:

- Lost paper tickets
- Can't see order queue clearly
- Miscommunication about special instructions
- Don't know preparation sequence

2.4 Manager

Role Description: Oversees operations, manages employees, analyzes business performance

User Needs:

- UN-MGR-01: View daily/weekly/monthly sales reports
- UN-MGR-02: Track employee performance (orders processed, tips earned)
- UN-MGR-03: Manage employee records and roles
- UN-MGR-04: Manage menu items (add, update, remove, pricing)
- UN-MGR-05: Approve refunds and cancellations
- UN-MGR-06: View real-time operational dashboard
- UN-MGR-07: Generate financial reports (revenue, payment methods breakdown)
- UN-MGR-08: Analyze popular menu items
- UN-MGR-09: Track peak hours and customer flow
- UN-MGR-10: Set menu item availability (mark items unavailable)

Pain Points to Solve:

- Manual report compilation takes hours
- Can't track performance in real-time
- Difficulty identifying best-selling items
- No visibility into employee efficiency

2.5 Customer

Role Description: Browses menu, makes selections, pays for food

User Needs:

- UN-CUST-01: Browse available menu items with descriptions and prices
- UN-CUST-02: See dietary information (vegetarian, vegan, gluten-free, etc.)
- UN-CUST-03: Know preparation time estimates
- UN-CUST-04: Make special requests/modifications
- UN-CUST-05: Pay quickly with preferred method (cash or card)
- UN-CUST-06: Receive order number/receipt
- UN-CUST-07: Know when order will be ready (for takeout)
- UN-CUST-08: Know table assignment (for dine-in)
- UN-CUST-09: View order history (for registered customers)

Pain Points to Solve:

- Long wait times at cashier
- Unclear menu item descriptions
- Not knowing dietary information
- Confusion about order status

3. DETAILED USER REQUIREMENTS

3.1 Employee Management Requirements

UR-EM-01: Basic Employee Information

- The system SHALL store employee information:
  - Employee ID (unique, auto-generated primary key)
  - First name (required)
  - Last name (required)
  - Phone number (required, format: XXX-XXX-XXXX)
  - Email address (required, valid email format)
  - Hire date (required, cannot be future date)
  - Hourly wage (required, must be ≥ minimum wage)
  - Employment status (required: Active, Inactive, On Leave)

UR-EM-02: Employee Role Specializations

- The system SHALL support multiple employee role types with specific attributes:

  Cashier Role:

  - Register number (which register/terminal assigned)
  - Cash handling certification status (Yes/No)
  - Date certified
- **Waiter Role**:
  - Assigned section (Section A, B, C, Patio, etc.)
  - Total tips earned (running cumulative total)
  - Tables served count (lifetime count)
- **Chef Role**:
  - Culinary specialty (Italian, French, Asian, American, Mexican, Pastry, etc.)
  - Certification level (Junior Chef, Senior Chef, Sous Chef, Head Chef)
  - Years of experience (numeric)
- **Manager Role**:
  - Department managed (Front-of-House, Kitchen, General)
  - Management level (Shift Manager, Assistant Manager, General Manager)
  - Office extension number

UR-EM-03: Multi-Role Employee Support (CRITICAL - LATTICE STRUCTURE)

- The system MUST support employees holding MULTIPLE roles simultaneously
- An employee can be:
  - Just a Cashier
  - Just a Waiter
  - Just a Chef
  - Just a Manager
  - Cashier AND Waiter (works register and serves tables)
  - Waiter AND Manager (shift supervisor who also serves)
  - Chef AND Manager (head chef)
  - Any other valid combination
- When an employee has multiple roles, ALL role-specific attributes MUST be maintained
- This creates an **OVERLAPPING SPECIALIZATION** (lattice structure)
- The system must allow querying: "Show me all employees who are both Waiters AND Managers"

UR-EM-04: Employee Work Assignment

- The system SHALL track which employee (cashier) processed which payment
- The system SHALL track which employee (waiter) served which order (for dine-in)
- The system SHALL track which employees (chefs) can prepare which menu items

UR-EM-05: Employee Status Management

- Managers SHALL be able to activate/deactivate employee accounts

- The system SHALL maintain hire date and cannot be modified after initial entry
- Only active employees can be assigned to payments/orders

3.2 Customer Management Requirements

UR-CM-01: Basic Customer Information

- The system SHALL store customer data:
    - Customer ID (unique, auto-generated primary key)
    - First name (required)
    - Last name (required)
    - Phone number (required, unique, format: XXX-XXX-XXXX)
    - Email address (optional, if provided must be valid format)
    - Registration date (auto-generated, timestamp of first registration)

UR-CM-02: Customer Registration

- The system SHALL allow walk-in customers to make purchases without registration
- The system SHALL allow customers to optionally register for tracking purposes
- Registered customers can view their order history
- Phone number is used as unique identifier for returning customers

UR-CM-03: Customer Privacy

- The system SHALL protect customer personal information
- Only managers and cashiers can view customer details
- Customer data cannot be shared with third parties

3.3 Menu Management Requirements

UR-MM-01: Basic Menu Item Information

- The system SHALL store menu items with:
    - Item ID (unique, auto-generated primary key)
    - Item name (required, max 100 characters)
    - Description (required, max 500 characters)
    - Price (required, must be > 0, up to 2 decimal places)
    - Preparation time (required, in minutes, must be > 0)
    - Availability status (required: Available, Unavailable, Seasonal)
    - Calories (optional, if provided must be > 0)
    - Image URL/path (optional)

UR-MM-02: Menu Item Categories (DISJOINT SPECIALIZATION)

- The system SHALL categorize every menu item into exactly ONE category:

Appetizer Category:

  - Serving size (required: Small, Medium, Large, Shareable)
  - Spice level (required: None, Mild, Medium, Hot, Extra Hot)
- **Main Course Category**:
  - Protein type (required: Beef, Chicken, Fish, Pork, Vegetarian, Vegan)
  - Includes side dish (required: Yes/No)
  - Portion size (required: Regular, Large)
- **Beverage Category**:
  - Volume in oz (required, must be > 0)
  - Is alcoholic (required: Yes/No)
  - Is refillable (required: Yes/No)
- A menu item CANNOT belong to multiple categories (disjoint)
- Every menu item MUST belong to one category (total participation)

UR-MM-03: Dietary Information (MULTI-VALUED ATTRIBUTE)

- The system SHALL support multiple dietary tags per menu item:
  - Vegetarian
  - Vegan
  - Gluten-Free
  - Dairy-Free
  - Nut-Free
  - Halal
  - Kosher
  - Low-Calorie
  - Spicy
- A single menu item can have zero or multiple dietary tags
- Example: An item can be both "Vegetarian" AND "Gluten-Free" AND "Dairy-Free"

UR-MM-04: Menu Item Availability Management

- Managers SHALL be able to mark items as Available/Unavailable
- Chefs SHALL be able to temporarily mark items unavailable (86'd)
- Unavailable items SHALL NOT appear in cashier's ordering interface
- The system SHALL prevent ordering unavailable items

UR-MM-05: Pricing Management

- Only managers can update menu item prices
- Price changes take effect immediately for new orders
- Historical orders maintain the price at time of order (stored in CONTAINS relationship)

3.4 Order Management Requirements

UR-OM-01: Order Creation Logic (CRITICAL BUSINESS RULE)

- Orders ARE CREATED ONLY when payment is successfully processed
- Order CANNOT exist without a corresponding payment
- The creation sequence is:
    1. Customer selects menu items (not yet an order)
    2. Cashier processes payment
    3. Payment is successful → Order is automatically created
    4. Order is sent to kitchen (if food items)
    5. Order is fulfilled
- This is a **1:1 relationship** between Payment and Order

UR-OM-02: Basic Order Information

- The system SHALL store order data:
    - Order ID (unique, auto-generated primary key)
    - Order datetime (required, auto-generated timestamp)
    - Order type (required: DineIn, Takeout) ← Simple attribute, not specialization
    - Table number (optional, required only for DineIn, 1-50)
    - Number of guests (optional, relevant for DineIn)
    - Pickup time (optional, relevant for Takeout)
    - Subtotal (required, auto-calculated, sum of line totals)
    - Tax amount (required, auto-calculated, subtotal × 0.08)
    - Total amount (required, auto-calculated, subtotal + tax)
    - Order status (required: Pending, In-Progress, Ready, Completed, Cancelled)
    - Special notes (optional, max 500 characters)

UR-OM-03: Order-MenuItem Relationship (MANY-TO-MANY)

- An order CONTAINS multiple menu items

- A menu item appears in multiple orders

- This is a **MANY-TO-MANY relationship** with attributes stored on the relationship:

  CONTAINS Relationship Attributes:

    - Quantity (required, must be > 0)
    - Special instructions (optional, e.g., "No onions", "Extra sauce", "Well done")
    - Unit price at order time (required, captures price at moment of order)
    - Line total (required, auto-calculated: quantity × unit_price_at_order_time)

- Since there is NO separate OrderItems entity, these attributes exist on the relationship itself

- This is implemented as a junction/associative table in relational model

UR-OM-04: Order Status Workflow

- Order status must follow logical progression:
    - Pending → In-Progress → Ready → Completed
    - Can jump to Cancelled from any status
- Status transitions:
    - **Pending**: Order created, sent to kitchen
    - **In-Progress**: Chef has started preparing
    - **Ready**: Food is ready for pickup/service
    - **Completed**: Customer received order
    - **Cancelled**: Order was cancelled (with reason)

UR-OM-05: Order Type Handling

- **DineIn Orders**:
    - MUST have table number assigned
    - SHOULD have number of guests recorded
    - Waiter may be assigned to serve order
    - Order delivered to table when ready
- **Takeout Orders**:
    - MUST have pickup time specified
    - Table number is NULL
    - Customer called when order ready
    - Customer picks up at counter

UR-OM-06: Order Calculations (AUTO-CALCULATED)

- Subtotal = SUM of all line_total from CONTAINS relationship
- Tax Amount = Subtotal × 0.08 (8% tax rate)
- Total Amount = Subtotal + Tax Amount
- All calculations must round to 2 decimal places
- Calculations triggered automatically when items added/removed

UR-OM-07: Order Modification Rules

- Orders can be modified ONLY while in "Pending" status
- Once "In-Progress", items cannot be added/removed
- Cancelled orders cannot be modified
- Modifications update subtotal, tax, and total automatically

UR-OM-08: Order Assignment

- Every order MUST have a cashier (who processed the payment)
- DineIn orders MAY have a waiter assigned (optional)
- System tracks these via relationships

3.5 Payment Processing Requirements

UR-PP-01: Payment Creates Order (CRITICAL RELATIONSHIP)

- Payment MUST be processed BEFORE order is created
- One Payment creates exactly ONE Order (1:1 relationship)
- Payment and Order are created in same transaction (atomic operation)
- If payment fails, no order is created
- If payment succeeds, order MUST be created

UR-PP-02: Basic Payment Information

- The system SHALL store payment data:
  - Payment ID (unique, auto-generated primary key)
  - Payment datetime (required, auto-generated timestamp)
  - Amount paid (required, must be ≥ order total)
  - Tip amount (required, can be 0, must be ≥ 0)
  - Payment status (required: Pending, Completed, Failed, Refunded)
  - Cashier ID (required, FK to Employee/Cashier who processed it)
  - Customer ID (optional, FK to Customer if registered customer)

UR-PP-03: Payment Method Types (DISJOINT SPECIALIZATION)

- Every payment is processed using exactly ONE payment method:

  Cash Payment:

  - Amount tendered (required, must be ≥ amount_paid)
  - Change given (required, auto-calculated: tendered - amount_paid)
- **Card Payment**:
  - Card type (required: Visa, Mastercard, American Express, Discover)
  - Last four digits (required, exactly 4 digits)
  - Authorization code (required, 6-8 alphanumeric characters)
  - Transaction ID (required, unique)
- A payment is EITHER cash OR card, not both (disjoint)

- Every payment MUST be one of these types (total participation)

UR-PP-04: Payment Validation Rules

- Amount paid MUST be ≥ Order total
- For cash: Amount tendered MUST be ≥ Amount paid
- For card: Must receive valid authorization code
- Tip amount is separate from order total
- Total collected = Amount paid + Tip amount

UR-PP-05: Payment-Cashier Relationship (PROCESSES)

- Every payment MUST be processed by a Cashier
- One Cashier processes many payments over time (1:M relationship)
- System must record which cashier handled transaction
- Used for accountability and performance tracking

UR-PP-06: Payment Refund Handling

- Only managers can process refunds
- Refunded payments must record:
    - Refund reason (required)
    - Refund amount (may be partial or full)
    - Manager who approved (required)
    - Refund datetime
- Original payment status changes to "Refunded"
- Associated order status changes to "Cancelled"

3.6 Employee-Order Relationships

UR-EO-01: Waiter SERVES Order (1:M Relationship)

- A Waiter can serve multiple orders
- Each order is served by at most one waiter
- This relationship is OPTIONAL (partial participation):
    - Takeout orders don't have waiters
    - Only DineIn orders may have waiters assigned
- Relationship tracks which waiter is responsible for table service
- Used for tip distribution and performance tracking

UR-EO-02: Waiter Assignment Logic

- Waiter can be assigned when order is created (for DineIn)
- Assignment based on section and table number
- System should suggest waiter based on assigned section
- Manager can manually reassign waiter if needed

UR-EO-03: Waiter Performance Tracking

- System tracks number of orders served per waiter
- System tracks total tips earned per waiter
- Tips from payments are attributed to assigned waiter
- Reports available showing waiter performance metrics

4. SYSTEM FUNCTIONAL REQUIREMENTS

4.1 Cashier Interface Requirements

SR-CASH-01: Order Entry Interface

- System SHALL provide touch-screen friendly interface for item selection
- Menu items displayed by category with images
- Quick access to frequently ordered items
- Search functionality to find items by name
- Display current price and availability status

SR-CASH-02: Order Building

- Cashier can add multiple items to cart
- Specify quantity for each item
- Add special instructions per item
- Modify or remove items before payment
- View running subtotal, tax, and total

SR-CASH-03: Payment Processing Interface

- Select payment method (Cash or Card)
- For cash: Enter amount tendered, display change
- For card: Integrate with card terminal, capture authorization
- Add tip amount (optional)
- Process payment with single click/tap

SR-CASH-04: Receipt Generation

- Automatically print receipt upon successful payment
- Receipt includes: order number, datetime, items, quantities, prices, subtotal, tax, total, payment method, cashier name
- Display order number prominently
- For takeout: include estimated pickup time

SR-CASH-05: Error Handling

- Display clear error messages for failed payments
- Allow retry without losing order details

- Void transaction capability (with manager approval)

4.2 Kitchen/Chef Interface Requirements

SR-CHEF-01: Order Queue Display

- Display all pending and in-progress orders
- Sort by order time (oldest first)
- Highlight urgent/delayed orders
- Show order number, items, quantities, special instructions
- Different colors for dine-in vs takeout

SR-CHEF-02: Order Status Updates

- Chef can mark order as "In-Progress" when starting
- Chef can mark order as "Ready" when complete
- Updates reflected immediately in waiter interface
- Notification sent to waiter when order ready (for dine-in)

SR-CHEF-03: Item Availability Management

- Chef can mark menu items as temporarily unavailable (86'd)
- Affects cashier's ordering interface immediately
- Manager can restore availability

4.3 Waiter Interface Requirements

SR-WAIT-01: Order Tracking

- View all orders assigned to waiter
- Filter by table number
- See order status in real-time
- Receive alerts when orders ready for pickup from kitchen

SR-WAIT-02: Table Management

- View list of tables with current status
- See which orders are assigned to which tables
- View order details for specific table

SR-WAIT-03: Tip Tracking

- View tips earned today
- View tips earned this week/month
- View total lifetime tips

4.4 Manager Interface Requirements

SR-MGR-01: Dashboard

- Real-time sales summary (today's revenue)
- Number of orders today
- Average order value
- Payment method breakdown
- Current employee on duty list
- Low availability menu items alert

SR-MGR-02: Employee Management

- Add/edit/deactivate employees
- Assign/modify employee roles
- View employee performance metrics
- Approve refunds and voids

SR-MGR-03: Menu Management

- Add/edit/remove menu items
- Update prices
- Set availability status
- Manage categories and dietary tags

SR-MGR-04: Reporting

- Daily sales report
- Weekly/monthly revenue trends
- Popular menu items
- Payment method analysis
- Employee performance report
- Peak hours analysis

5. DATABASE REQUIREMENTS

5.1 Entity Requirements

DR-ENT-01: Core Entities The system SHALL implement the following entities:

1. Employee (superclass)
2. Cashier (subclass of Employee)
3. Waiter (subclass of Employee)
4. Chef (subclass of Employee)
5. Manager (subclass of Employee)

6. Customer
7. MenuItem (superclass)
8. Appetizer (subclass of MenuItem)
9. MainCourse (subclass of MenuItem)
10. Beverage (subclass of MenuItem)
11. Order
12. Payment (superclass)
13. CashPayment (subclass of Payment)
14. CardPayment (subclass of Payment)

5.2 Relationship Requirements

DR-REL-01: Customer-Payment (1:M)

- One customer can make many payments
- Each payment made by one customer
- Relationship: MAKES

DR-REL-02: Payment-Order (1:1) - CRITICAL

- One payment creates exactly one order
- One order created by exactly one payment
- Relationship: CREATES
- Must be enforced as 1:1 in database

DR-REL-03: Cashier-Payment (1:M)

- One cashier processes many payments
- Each payment processed by one cashier
- Relationship: PROCESSES

DR-REL-04: Order-MenuItem (M:N)

- One order contains many menu items
- One menu item appears in many orders
- Relationship: CONTAINS
- Relationship attributes: quantity, special_instructions, unit_price_at_order_time, line_total

DR-REL-05: Waiter-Order (1:M, Optional)

- One waiter serves many orders
- Each order served by at most one waiter (can be null for takeout)
- Relationship: SERVES
- Partial participation from Order side

5.3 Specialization Requirements

DR-SPEC-01: Employee Hierarchy (OVERLAPPING)

- Specialization type: Overlapping, Total
- Constraint: Every employee must be at least one subtype
- Constraint: Employee can be multiple subtypes simultaneously
- Creates lattice structure

DR-SPEC-02: MenuItem Hierarchy (DISJOINT)

- Specialization type: Disjoint, Total
- Constraint: Every menu item is exactly one category
- Constraint: Cannot belong to multiple categories

DR-SPEC-03: Payment Hierarchy (DISJOINT)

- Specialization type: Disjoint, Total
- Constraint: Every payment is exactly one payment type
- Constraint: Cannot be both cash and card

5.4 Integrity Constraints

DR-INT-01: Primary Key Constraints

- Every entity must have a unique, non-null primary key
- Primary keys should be auto-generated integers where possible

DR-INT-02: Foreign Key Constraints

- All foreign keys must reference existing primary keys
- Referential integrity must be enforced
- Define appropriate CASCADE, SET NULL, or RESTRICT rules

DR-INT-03: Domain Constraints

- Prices, amounts, quantities must be positive
- Email must match valid email pattern
- Phone numbers must match format XXX-XXX-XXXX
- Dates cannot be in future (except pickup_time)
- Status fields must be from predefined enumeration

DR-INT-04: Business Rule Constraints

- Order total must equal subtotal + tax
- Payment amount must be ≥ order total

- DineIn orders must have table number
- Takeout orders must have pickup time
- Cash tendered must be ≥ amount paid
- Quantity must be > 0

5.5 Trigger Requirements

DR-TRIG-01: Auto-Calculate Order Totals

- Trigger fires when items added/removed from order
- Calculates subtotal, tax, total automatically
- Updates Order entity

DR-TRIG-02: Create Order on Payment Success

- Trigger fires when payment status becomes "Completed"
- Automatically creates corresponding Order entity
- Ensures 1:1 relationship between Payment and Order

DR-TRIG-03: Update Waiter Tips

- Trigger fires when payment is completed with tip amount
- Updates total_tips in Waiter entity
- Only if waiter is assigned to order

DR-TRIG-04: Validate Order Status Transitions

- Trigger prevents invalid status changes
- Enforces workflow: Pending → In-Progress → Ready → Completed
- Allows Cancelled from any status

DR-TRIG-05: Prevent Ordering Unavailable Items

- Trigger fires on insert to CONTAINS relationship
- Checks if menu item is Available
- Raises error if unavailable

5.6 Stored Procedure Requirements

DR-PROC-01: Process_Payment_And_Create_Order

- Input: customer_id, cashier_id, payment_method, items_list, order_type, table_number
- Logic:
    1. Calculate order totals
    2. Validate payment amount
    3. Create Payment record

4. Create Order record
5. Create CONTAINS relationships for each item
6. Return order_id and payment_id

- Must be atomic transaction (all or nothing)

DR-PROC-02: Generate_Daily_Sales_Report

- Input: report_date
- Output: total_revenue, order_count, avg_order_value, payment_method_breakdown, top_selling_items
- Aggregates data from orders and payments for specified date

DR-PROC-03: Assign_Waiter_To_Order

- Input: order_id, waiter_id
- Validates waiter exists and is active
- Creates SERVES relationship
- Updates order with waiter assignment

DR-PROC-04: Update_Menu_Item_Availability

- Input: item_id, new_status, employee_id
- Validates employee has permission (chef or manager)
- Updates menu item availability status
- Logs change with timestamp and employee

5.7 View Requirements

DR-VIEW-01: Available_Menu_Items

- Shows only available menu items
- Includes: item_id, name, category, price, dietary_tags, preparation_time
- Used by cashier interface

DR-VIEW-02: Waiter_Active_Orders

- Input: waiter_id
- Shows all pending/in-progress orders for specific waiter
- Includes: order_id, table_number, order_time, status, items_count

DR-VIEW-03: Daily_Sales_Summary

- Shows aggregated sales for current day
- Includes: total_orders, total_revenue, avg_order_value, cash_vs_card_breakdown

DR-VIEW-04: Kitchen_Order_Queue

- Shows all orders with status Pending or In-Progress
- Sorted by order time (oldest first)
- Includes: order_id, order_time, items_with_quantities, special_instructions, order_type

5.8 Index Requirements

DR-IDX-01: Performance Indexes

- Create indexes on:
  - Customer(phone_number) - for quick customer lookup
  - Order(order_datetime) - for date-range queries
  - Order(order_status) - for filtering by status
  - Payment(payment_datetime) - for date-range queries
  - MenuItem(availability_status) - for filtering available items
  - Employee(employee_id, role) - for role-based queries (if using single table inheritance)