PRODUCT DISSECTION CASE STUDY

Capstone Project: Data Architecture in D2C Platforms

Platform Focus: LICIOUS

(India's Leading D2C Meat & Seafood Brand)

"Solving the 'Fresh vs. Frozen' dilemma through Hyperlocal Data Modeling"

PROJECT OVERVIEW

This case study dissects the core functionalities of Licious to understand how data architecture solves real-world supply chain challenges. The project includes a detailed analysis of the "Farm-to-Fork" model, followed by a custom SQL-ready Schema Design that supports hyperlocal delivery, inventory traceability, and customer retention.

KEY DELIVERABLES

- **Product Research: Core features and user journey analysis.**
- **Problem Dissection: Addressing hygiene, consistency, and cold-chain logistics.**
- **Case Study: The "Weekend Biryani" User Scenario.**
- **Schema Design: A comprehensive database structure for Inventory & Orders.**
- **ER Diagram: Visual representation of entities and relationships.**
- **Rationale: Strategic reasoning behind the data model.**

Submitted By: Sreekant Bhanu

Date: December 18, 2025

Project: Product Dissection & Schema Design for Licious

Step 1: Choose a Leading Platform

Platform Selected: Licious Domain: D2C (Direct-to-Consumer) Food Tech / Meat & Seafood **Overview:** Licious is India's first D2C unicorn in the meat and seafood sector. It operates on a "farm-to-fork" model, owning the entire cold chain supply backend to deliver fresh (never frozen), hygienic, and chemical-free meat to consumers.

Step 2: Research & Core Features

Licious distinguishes itself through strict quality control and supply chain ownership.

- **Farm-to-Fork Supply Chain:** unlike aggregators (e.g., Swiggy/Zomato), Licious owns the processing canters and delivery logistics.
- **Cold Chain Technology:** Maintains products at 0-4°C from procurement to delivery to ensure freshness without freezing.
- **Specific Cuts & Categorization:** Offers highly specific cuts (e.g., *Chicken Curry Cut - Small*, *Mutton Biryani Cut*) rather than generic weight-based blocks.
- **"Meatopia" Subscription:** A loyalty program offering unlimited free delivery.
- **Ready-to-Cook (RTC):** Marinated meats and spreads that solve the "convenience" problem for busy urban users.
- **Express Delivery:** 90-120 minute delivery promise in specific zones.

Step 3: Product Dissection (Real-World Problems)

The platform solves three major pain points inherent in the traditional Indian meat market:

1. **Hygiene & Safety (The "Wet Market" Problem):**
    - **Problem:** Traditional wet markets are often unhygienic, foul-smelling, and lack quality certification. Consumers worry about antibiotic residue and stale meat.
    - **Solution:** Licious sources from biosecure farms, performs 150+ quality checks, and is FSSAI certified.

2. **Inconsistency & Lack of Variety:**
    - **Problem:** Local butchers often lack specific meats (e.g., Lamb chops, Blue Crab) or cannot cut meat precisely for specific dishes (e.g., boneless cubes for Tikka).
    - **Solution:** Standardized inventory with precise "cuts" described visually on the app.
3. **The "Fresh vs. Frozen" Dilemma:**
    - **Problem:** Supermarket meat is often frozen (hard texture), while butcher meat spoils quickly.
    - **Solution:** The 0-4°C cold chain ensures meat remains "chilled" (fresh texture) but bacteria-free.

Step 4: Case Study - The "Weekend Biryani" Challenge

Scenario:

- **User:** Rahul, a busy data analyst in Bangalore.
- **Challenge:** It is Sunday morning. Rahul wants to cook Mutton Biryani. He visits the local butcher, but the shop is crowded, smells bad, and the butcher runs out of "lean" cuts. Rahul buys what is available but worries the meat is tough.
- **Licious Approach:**
    1. Rahul opens Licious. He searches "Mutton".
    2. He sees **"Goat Biryani Cut"** – specifically curated pieces with the right bone-to-meat ratio for slow cooking.
    3. He sees a "Freshness Guarantee" and the batch source.
    4. He adds a **"Meatopia"** subscription to waive the delivery fee.
    5. **Outcome:** The meat arrives in vacuum-sealed, temperature-controlled packaging within 90 minutes. Rahul avoids the wet market entirely.

Step 5: Schema Design Based on Top Features

To support these features, the data schema must handle **Inventory Freshness (Batches)**, **Subscriptions**, and **Hyperlocal Delivery**.

Core Entities & Attributes

1. **User**
    - user_id (PK), name, phone_number, email, created_at
2. **Address** (Separated from User for multiple delivery locations)
    - address_id (PK), user_id (FK), latitude, longitude, city_id, is_serviceable
3. **Product**
    - product_id (PK), name (e.g., Chicken Curry Cut), category_id, base_price, cut_type (e.g., Boneless, Bone-in), description
4. **Inventory_Batch** (Crucial for "Freshness")
    - batch_id (PK), product_id (FK), hub_id (FK), sourcing_date, expiry_date, temperature_log_status, stock_quantity
5. **Subscription** (Meatopia)
    - subscription_id (PK), user_id (FK), plan_type (1-month, 3-months), start_date, end_date, status (Active/Expired)
6. **Order**
    - order_id (PK), user_id (FK), address_id (FK), total_amount, delivery_fee, order_status (Placed, Packed, Out for Delivery), delivery_slot_type (Express/Scheduled)
7. **Order_Items**
    - item_id (PK), order_id (FK), product_id (FK), batch_id (FK - *Traceability*), quantity, price_at_purchase
8. **Hub** (Supply Chain Node)
    - hub_id (PK), hub_name, city, service_radius_km

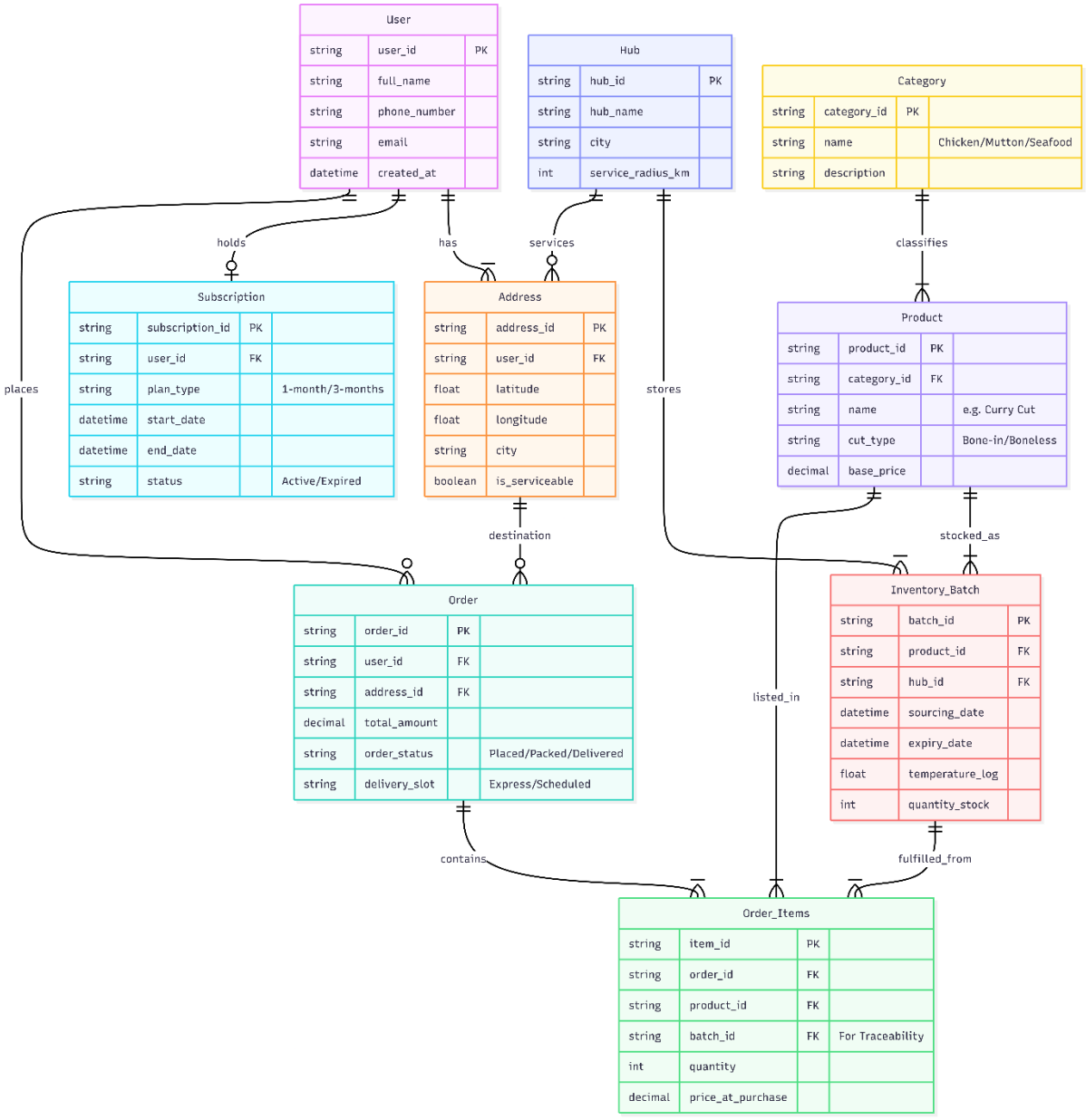
Step 6: Rationale Behind the Design

- **Why Inventory_Batch?** Unlike Amazon (non-perishables), Licious cannot just have a "stock count". They must track *which specific batch* a user gets to ensure it hasn't expired. This table allows the "0-4°C" quality check to be logged against specific meat batches.
- **Why Hub?** Licious is hyperlocal. Availability depends on the user's location relative to the nearest "Processing Hub". The schema links inventory to specific Hubs, not a central warehouse.

- **Why Subscription?** The separate entity manages the logic for waiving delivery fees (Meatopia) easily during the checkout query.

Step 7: ER Diagram

Below is the Entity-Relationship representation.



ER Diagram & Relationships Explanation

The Licious schema is designed to ensure **hyperlocal delivery** and **quality traceability**.

1. Key Entities:

- **User & Address:** Separates the user profile from delivery locations (latitude/longitude) to calculate precise delivery times.

- **Hub:** Represents local processing canters. Unlike a central warehouse, inventory is linked to specific Hubs to determine neighbourhood availability.
- **Inventory_Batch:** The core of the "Freshness" promise. It tracks specific batches (sourcing date, temperature logs, expiry) rather than just product counts.
- **Subscription (Meatopia):** Manages loyalty status to automatically waive delivery fees.

2. Key Relationships:

- **Hub ↔ Inventory_Batch (One-to-Many):** A Hub holds multiple batches of meat. This allows the app to show only the stock available in the user's specific service radius.
- **Product ↔ Inventory_Batch (One-to-Many):** A single product (e.g., "Chicken Curry Cut") is replenished by many batches over time, preventing old stock from being sold.
- **Order_Items ↔ Inventory_Batch (Traceability):** Each item in an order is linked to its specific batch_id. This creates a digital thread, allowing Licious to trace any quality issue back to the exact farm and processing time.

3. Rationale: This structure enforces the **0-4°C cold chain** logic. By querying Inventory_Batch instead of just Product, the system ensures expired items are automatically removed from the app, guaranteeing safety and freshness

---

Step 8: Presentation of Findings

The schema directly impacts Licious's ability to deliver on its brand promise:

1. **Freshness Tracking:** By linking Order Items to Inventory Batch, Licious can trace exactly which farm a specific customer's chicken came from, enabling the "Farm-to-Fork" transparency displayed in the app.
2. **Hyperlocal Availability:** The Hub to Address relationship allows the app to instantly filter out products that are out of stock in the user's specific delivery zone (crucial for 90-min delivery).
3. **Revenue Retention:** The Subscription entity allows for easy implementation of loyalty logic (e.g., IF Subscription.status = 'Active' THEN DeliveryFee = 0), which increases repeat purchase frequency.