KP FOODS – Detailed Database Project Report

Project Overview

The **KP FOODS Database** is a structured relational database that models an **online food ordering and delivery platform**, similar to real-world applications like Zomato or Swiggy. The system is designed to:

- Store and manage restaurant, customer, delivery partner, and order details.
- Track menu items and their prices.
- Record and monitor food orders and delivery statuses.
- Provide meaningful **business insights** (e.g., revenue, popular dishes, customer trends).

This database is built using **SQL (MySQL)** and demonstrates practical use of **DDL (Data Definition Language)**, **DML (Data Manipulation Language)**, **Constraints**, **Joins**, and **Aggregate functions**.

Objectives

- Create normalized relational tables with proper primary and foreign keys.
- Insert sample realistic data for restaurants, customers, menu, orders, and deliveries.
- Execute queries to extract **business intelligence**.
- Maintain **data integrity** with referential constraints.
- Demonstrate key SQL concepts for academic or professional projects.

Database Schema

Tables and Purpose

1. **KP_Restaurants** – Stores restaurant details (name, location, rating).
2. **KP_Menu** – Menu items offered by restaurants, with prices.
3. **KP_Customers** – Customer personal details and addresses.
4. **KP_DeliveryPartners** – Delivery partner information and ratings.
5. **KP_Orders** – Order details including customer, restaurant, date, and bill amount.
6. **KP_OrderItems** – Tracks specific food items within each order.
7. **KP_Deliveries** – Delivery tracking: which partner delivered, status, and delivery time.

Entity-Relationship (ER) Model

- A **restaurant** can have multiple **menu items**.
- A **customer** can place multiple **orders**.
- Each **order** contains multiple **items** (via KP_OrderItems).
- Each **order** is assigned to one **delivery partner**.
- Delivery details (time, status) are tracked in KP_Deliveries.

Data Dictionary

Table	Primary Key	Important Columns	Description
KP_Restaurants	restaurant_id	name, location, rating	Stores restaurant info
KP_Menu	item_id	item_name, price	Menu items linked to restaurants
KP_Customers	customer_id	name, phone, address	Customer details
KP_DeliveryPartners	partner_id	name, rating	Delivery partner details
KP_Orders	order_id	order_date, total_amount	Tracks customer orders

Table	Primary Key	Important Columns	Description
KP_OrderItems	order_item_id	quantity	Items in each order
KP_Deliveries	delivery_id	status, delivery_time	Delivery tracking

Referential Integrity Rules

- KP_Menu.restaurant_id → references KP_Restaurants(restaurant_id)
- KP_Orders.customer_id → references KP_Customers(customer_id)
- KP_Orders.restaurant_id → references KP_Restaurants(restaurant_id)
- KP_OrderItems.order_id → references KP_Orders(order_id)
- KP_OrderItems.item_id → references KP_Menu(item_id)
- KP_Deliveries.order_id → references KP_Orders(order_id)
- KP_Deliveries.partner_id → references KP_DeliveryPartners(partner_id)

Normalization

The schema follows **3rd Normal Form (3NF)**:

1. **1NF** – No repeating groups; atomic values ensured.
2. **2NF** – All non-key attributes fully dependent on the primary key.
3. **3NF** – No transitive dependencies (e.g., restaurant rating not stored in menu table).

Sample Data

- **Restaurants**: 8 entries (Hyderabad, Bangalore, Chennai, Delhi, Mumbai).
- **Menu**: 14 diverse items across cuisines.
- **Customers**: 4 entries with unique phone numbers.
- **Delivery Partners**: 7 entries with ratings.
- **Orders**: 4 sample orders.
- **Deliveries**: 4 records (3 delivered, 1 pending).

SQL Features Used

- **Primary Keys** → Ensure entity uniqueness.
- **Foreign Keys** → Maintain relationships between tables.
- **Constraints** → Data integrity (e.g., rating as decimal).
- **Joins** → Fetch combined data across multiple tables.
- **Aggregate Functions** → SUM(), AVG(), MAX(), COUNT() used for business insights.
- **DML** → INSERT, UPDATE, DELETE.

Business Queries & Insights

◇ Customers

- List all customers → **Ravi Kumar, Kishore Sharma, Manoj Verma, Sneha Rao**.
- Customers with orders above ₹400 → Ravi (₹550), Sneha (₹540).
-

◇ Restaurants

- Restaurants in Hyderabad → *Spicy Treats*, *BurgerTown*.
- Average restaurant rating → **4.39**.
- Most popular restaurant (by orders) → *Curry Palace*.
- Revenue per restaurant → Highlights *Spicy Treats* and *Curry Palace* as top earners.

◇ Menu

- Items cheaper than ₹250 → Garlic Bread, Veg Burger, Smoothie Bowl, etc.
- Most expensive item → **Salmon Sushi (₹450)**.
- Most popular dish (highest quantity ordered) → **Tandoori Roti (3 orders)**.

◇ Orders & Deliveries

- Total platform revenue → **₹1,560**.
- Pending orders → **1 (Manoj Verma's order)**.
- Average delivery time → Suresh (30 min), Lakshmi (25 min), Sita (35 min).
- Orders with customer + restaurant + delivery → Complete order tracking possible.

Data Quality Checks

- Order 501: Item-wise total = (2 × 250) + (1 × 300) = **800**, but stored as 550 → mismatch.
- Suggestion: Use triggers or stored procedures to auto-calculate total_amount.
- Consistency between delivery time and status ensured (NULL for pending).

Applications of This Project

- Can serve as a **backend database** for a food delivery app.
- Useful for **SQL learning & practice** (Joins, Aggregates, Constraints).
- Extendable to include:
  - Payment tracking
  - Discounts and offers
  - Customer reviews
  - Multi-restaurant orders

Possible Enhancements

- Add **Payment Table** (payment modes, transaction status).
- Add **Reviews & Ratings Table** (customer feedback on restaurants & delivery partners).
- Implement **discounts/coupons** system.
- Use **Indexes** on phone, order_date, restaurant_id for faster queries.
- Add **Views** (e.g., daily sales, active orders).

Conclusion

The **KP FOODS Database** effectively simulates an online food ordering and delivery ecosystem. It ensures:

- **Data integrity** via primary and foreign keys.
- **Efficient queries** for analytics and reporting.
- **Practical business insights** (revenue, trends, performance).

This project demonstrates how **RDBMS concepts** can be applied in real-world scenarios and serves as a strong academic or professional project.