Case Study: Online Food Ordering System

Introduction

The online food ordering industry has revolutionized the way people dine by connecting customers with restaurants through digital platforms. Customers can browse menus, place orders, make payments, and review their experiences—all from the convenience of their devices.
Restaurants, on the other hand, gain access to a larger customer base and can analyze sales trends, menu popularity, and customer satisfaction to enhance performance.

To ensure efficiency, data-driven decision-making is critical. This case study focuses on designing a relational database schema for an online food ordering system and using SQL to extract business insights.

Database Schema

Customers

- CustomerID (INT, Primary Key)
- FirstName (VARCHAR)
- LastName (VARCHAR)
- Email (VARCHAR)
- Phone (VARCHAR)
- RegistrationDate (DATE)

Restaurants

- RestaurantID (INT, Primary Key)
- RestaurantName (VARCHAR)
- Address (VARCHAR)

- CuisineType (VARCHAR)

MenuItems

- MenuItemID (INT, Primary Key)
- RestaurantID (INT, Foreign Key)
- ItemName (VARCHAR)
- Price (DECIMAL)
- Description (TEXT)

Orders

- OrderID (INT, Primary Key)
- CustomerID (INT, Foreign Key)
- RestaurantID (INT, Foreign Key)
- OrderDate (DATE)
- TotalAmount (DECIMAL)

OrderDetails

- OrderDetailID (INT, Primary Key)
- OrderID (INT, Foreign Key)
- MenuItemID (INT, Foreign Key)
- Quantity (INT)
- ItemPrice (DECIMAL)

Reviews

- ReviewID (INT, Primary Key)

- RestaurantID (INT, Foreign Key)
- CustomerID (INT, Foreign Key)
- ReviewDate (DATE)
- Rating (DECIMAL)
- Comments (TEXT)

Objective

The company aims to enhance customer experience, optimize restaurant performance, and increase profitability by analyzing data across various dimensions.
The main objectives are to:

- Understand customer ordering patterns and preferences.
- Identify top-performing restaurants and popular menu items.
- Evaluate customer satisfaction and feedback.
- Analyze revenue trends and operational bottlenecks.
- Support decision-making through actionable insights derived from SQL queries.

Problem Statement

Customer Behavior & Engagement

1. Identify customers who have placed the highest number of orders. What does this indicate about their loyalty or engagement?
2. Find customers who haven't placed any orders in the last 3 months. How can the platform re-engage them?
3. Determine the average order value per customer. What does this reveal about spending habits?

Restaurant Performance Analysis

1. Identify the top 5 restaurants with the highest average ratings. What factors might contribute to their success?

2. Find restaurants with the lowest number of orders despite being active. What could be causing low performance?

3. Determine which cuisine types receive the most and least orders. How can marketing focus be adjusted accordingly?

Revenue & Business Metrics

1. Calculate the total revenue generated in the last quarter. How does it compare to the previous one?

2. Identify the top 3 most ordered menu items across all restaurants. Should these items be promoted further?

3. Analyze the revenue contribution of each restaurant. Which ones are the most profitable?

Operational Efficiency

1. Determine the average number of items per order. How can order processing be optimized?

2. Identify orders where the total amount was less than a certain threshold (e.g., ₹200). Should minimum order charges be introduced?

3. Find restaurants that receive the most reviews but have average ratings below 3.0. What corrective actions could be taken?

Customer Satisfaction & Feedback

1. Analyze the distribution of customer ratings. How many reviews are 4 stars or above?

2. Identify customers who frequently leave low ratings. Could this indicate dissatisfaction or unrealistic expectations?

3. Find patterns in review comments to detect recurring issues (e.g., “cold food,” “late delivery”).

Comparative & Predictive Analysis

1. Compare total revenue from vegetarian vs non-vegetarian menu items. Should the company invest more in one category?

2. Predict which customers are likely to become inactive based on order frequency and last order date.

3. Analyze order trends between weekends and weekdays. Should dynamic pricing or promotions be introduced on low-order days?

Tech Stack

- **Database:** MySQL / PostgreSQL