

<b>Project Title</b>	<b>Zomata - Food Delivery Data Insights Using Python and SQL</b>
<b>Skills take away From This Project</b>	<b>SQL Python Streamlit Data Engineering Database Management</b>
<b>Domain</b>	<b>Food Delivery and Logistics Analytics</b>

### **Problem Statement:**

Imagine you are working as a data scientist at Zomato. Your goal is to enhance operational efficiency and improve customer satisfaction by analyzing food delivery data. You need to build an interactive Streamlit tool that enables seamless data entry for managing orders, customers, restaurants, and deliveries. The tool should support robust database operations like adding columns or creating new tables dynamically while maintaining compatibility with existing code.

### **Business Use Cases:**

- **Order Management:**
  - Identifying peak ordering times and locations.
  - Tracking delayed and canceled deliveries.
- **Customer Analytics:**
  - Analyzing customer preferences and order patterns.
  - Identifying top customers based on order frequency and value.
- **Delivery Optimization:**
  - Analyzing delivery times and delays to improve logistics.
  - Tracking delivery personnel performance.
- **Restaurant Insights:**
  - Evaluating the most popular restaurants and cuisines.
  - Monitoring order values and frequency by restaurant.

## **Approach:**

### **1) Dataset Creation:**

- Use Python (Faker) to generate synthetic datasets for customers, orders, restaurants, and deliveries.
- Populate the SQL database with these datasets.

### **2) Database Design:**

- Create normalized SQL tables for **Customers, Orders, Restaurants, and Deliveries**.
- Ensure compatibility for dynamic schema changes (e.g., adding columns, creating new tables).

### **3) Data Entry Tool:**

Develop a Streamlit app for:

- Adding, updating, and deleting records in the SQL database.
- Dynamically creating new tables or modifying existing ones.

### **4) Data Insights:**

- Use SQL queries and Python to extract insights like peak times, delayed deliveries, and customer trends.
- Visualize the insights in the Streamlit app.(Add on)

### **5) OOP Implementation:**

- Encapsulate database operations in Python classes.
- Implement robust and reusable methods for CRUD (Create, Read, Update, Delete) operations.

### **6) Order Management:**

- Identifying peak ordering times and locations.
- Tracking delayed and canceled deliveries.

### **7) Customer Analytics:**

- Analyzing customer preferences and order patterns.
- Identifying top customers based on order frequency and value.

## 8) Delivery Optimization:

- Analyzing delivery times and delays to improve logistics.
- Tracking delivery personnel performance.

## 9) Restaurant Insights:

- Evaluating the most popular restaurants and cuisines.
- Monitoring order values and frequency by restaurant.

## Results:

By the end of this project, learners will achieve:

- A fully functional SQL database for managing food delivery data.
- An interactive Streamlit app for data entry and analysis.
- Should write 20 sql queries and do analysis.
- Dynamic compatibility with database schema changes.
- Comprehensive insights into order trends, delivery performance, and customer behavior.

## Project Evaluation metrics:

1. **Database Design:**
  - Proper normalization of tables and relationships between them.
2. **Code Quality:**
  - Use of OOP principles to ensure modularity and scalability.
  - Robust error handling for database operations.
3. **Streamlit App Functionality:**
  - Usability of the interface for data entry and insights.
  - Compatibility with schema changes.
4. **Data Insights:**
  - Use 20 sql queries for data analysis
5. **Documentation:**
  - Clear and comprehensive explanation of the code and approach.

## Technical Tags:

SQL, Python, Streamlit, Data Engineering, Object-Oriented Programming, Relational Databases, Data Analysis.

## Data Set:

- **Source:** Synthetic dataset generated using Python.
- **Format:** CSV or direct insertion into SQL tables.
- **Library:** Use Faker Python to create dummy dataset

### Customers Table

This table stores information about customers.

- **customer\_id** (Primary Key): Unique identifier for each customer.
- **name**: Customer's full name.
- **email**: Contact email address.
- **phone**: Contact phone number.
- **location**: Address or location of the customer.
- **signup\_date**: Date the customer signed up.
- **is\_premium**: Boolean indicating if the customer has a premium membership.
- **preferred\_cuisine**: Customer's preferred cuisine type.
- **total\_orders**: Total number of orders placed by the customer.
- **average\_rating**: Average rating given by the customer to restaurants.

### Restaurants Table

This table manages restaurant information.

- **restaurant\_id** (Primary Key): Unique identifier for each restaurant.
- **name**: Restaurant name.
- **cuisine\_type**: Primary cuisine type served (e.g., **Indian**, **Chinese**).
- **location**: Location of the restaurant.
- **owner\_name**: Name of the restaurant owner.
- **average\_delivery\_time**: Average delivery time for this restaurant (in minutes).
- **contact\_number**: Restaurant's contact number.
- **rating**: Average customer rating of the restaurant (1-5 scale).
- **total\_orders**: Total number of orders received by the restaurant.

- **is\_active**: Boolean indicating if the restaurant is currently active on the platform.

#### Orders Table

This table manages order details.

- **order\_id** (Primary Key): Unique identifier for each order.
- **customer\_id** (Foreign Key): References the **customer\_id** in the Customers table.
- **restaurant\_id** (Foreign Key): References the **restaurant\_id** in the Restaurants table.
- **order\_date**: Date and time when the order was placed.
- **delivery\_time**: Date and time when the order was delivered.
- **status**: Current status of the order (e.g., **Pending**, **Delivered**, **Cancelled**).
- **total\_amount**: Total bill amount for the order.
- **payment\_mode**: Payment mode used (e.g., **Credit Card**, **Cash**, **UPI**).
- **discount\_applied**: Discount amount applied to the order.
- **feedback\_rating**: Rating given by the customer for the order (1-5 scale).

#### Deliveries Table

This table stores information about deliveries.

- **delivery\_id** (Primary Key): Unique identifier for each delivery.
- **order\_id** (Foreign Key): References the **order\_id** in the Orders table.
- **delivery\_person\_id** (Foreign Key): References the **delivery\_person\_id** in a Delivery Persons table (if applicable).
- **delivery\_status**: Current delivery status (e.g., **On the way**, **Delivered**).
- **distance**: Distance of delivery in kilometers.
- **delivery\_time**: Actual time taken for delivery (in minutes).
- **estimated\_time**: Estimated delivery time (in minutes).
- **delivery\_fee**: Delivery fee charged for the order.
- **vehicle\_type**: Type of vehicle used for delivery (e.g., **Bike**, **Car**).

### Delivery Persons Table (*Optional*)

This optional table stores information about delivery personnel.

- **delivery\_person\_id** (Primary Key): Unique identifier for each delivery person.
- **name**: Delivery person's full name.
- **contact\_number**: Contact phone number.
- **vehicle\_type**: Type of vehicle used for delivery.
- **total\_deliveries**: Total number of deliveries completed.
- **average\_rating**: Average rating given by customers to this delivery person.
- **location**: Current base location of the delivery person.

### Justification for Columns:

- **Comprehensive Tracking**: Columns like **order\_status**, **delivery\_status**, and **average\_rating** help track the performance of restaurants and delivery personnel.
- **Customer Insights**: Columns such as **preferred\_cuisine** and **average\_rating** provide insights into customer preferences.
- **Operational Efficiency**: Columns like **distance**, **estimated\_time**, and **delivery\_time** enable performance tracking and optimization.
- **Data Integration**: Foreign keys ensure seamless linkage between tables for relational operations.

### Data Set Explanation:

#### Content:

- Customer profiles with order frequency and locations.
- Order details including date, value, status, and delivery times.
- Restaurant information with cuisines and popularity metrics.
- Delivery personnel performance metrics.

### Preprocessing Steps:

- Ensure no missing or duplicate data.
- Validate relationships between tables (e.g., `customer_id` in Orders matches `customer_id` in Customers).

### Project Deliverables:

1. **Source Code:**
  - Python scripts for dataset generation, database management, and Streamlit app development.
2. **Streamlit App:**
  - An interactive app for data entry and analysis.
  - 20 SQL queries for data analysis
3. **Documentation:**
  - Explanation of the approach, database schema, and instructions to run the project.





### Project Guidelines:

- 1) **Coding Standards:**
  - a) Use proper naming conventions, comments, and modular functions.
  - b) Follow Python OOP principles for database and application logic.
- 2) **Version Control:**
  - a) Use Git to manage the codebase, with clear commit messages.
- 3) **Error Handling:**
  - a) Handle database exceptions and provide meaningful error messages in the Streamlit app.
- 4) **User Experience:**
  - a) Design a user-friendly interface for the Streamlit app.
- 5) **Testing:**
  - a) Ensure all database operations (CRUD) work seamlessly.


### Timeline:

1 week

## References:

<b>Project Live Evaluation</b>	 Project Live Evaluation
<b>EDA Guide</b>	 Exploratory Data Analysis (EDA) Guide
<b>Capstone Explanation Guideline</b>	 Capstone Explanation Guideline
<b>GitHub Reference</b>	 How to Use GitHub.pptx
<b>Project Orientation (English)</b>	<a href="#">Recording Link</a>
<b>Project Orientation (Tamil)</b>	

## REFERENCES:

- Streamlit - Recording([Recording](#)) (TAMIL)
-  Special session for STREAMLIT(11/08/2024) (ENGLISH)
- **STREAMLIT**  
**DOCUMENTATION:**<https://docs.streamlit.io/get-started/installation>

**PROJECT DOUBT CLARIFICATION SESSION ( PROJECT AND CLASS DOUBTS)**



**About Session:** The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.

**Note:** Book the slot at least before 12:00 Pm on the same day

**Timing:** Tuesday, Thursday, Saturday (5:00PM to 7:00PM)

**Booking link :** <https://forms.gle/XC553oSbMJ2Gcfug9>

### **LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)**

**About Session:** The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.

**Note:** This form will Open on Saturday and Sunday Only on Every Week

**Timing:** Monday-Saturday (11:30PM to 12:30PM)

**Booking link :** <https://forms.gle/1m2Gsro41fLtZurRA>