Zomato Data Insights

This project simulates an Zomato Data Insights using Python **Faker** library, Python **SQL**, and **Streamlit** to analyze the Food Delivery Data for enhancing operational efficiency and Improve the Customer Satisfaction

Saravanan

31/01/2025

Objective

Zomato Insights

Analyze and visualize restaurant and food delivery trends using **Zomato data** to derive meaningful insights for business and user experience improvement.

Data Processing

Processes and stores data in a **SQL database** for querying and analysis.

Visualization

Develops **a Streamlit app** to visualize Data insights and showcase SQL query outputs.

Business Use Cases

Restaurant Performance Analysis

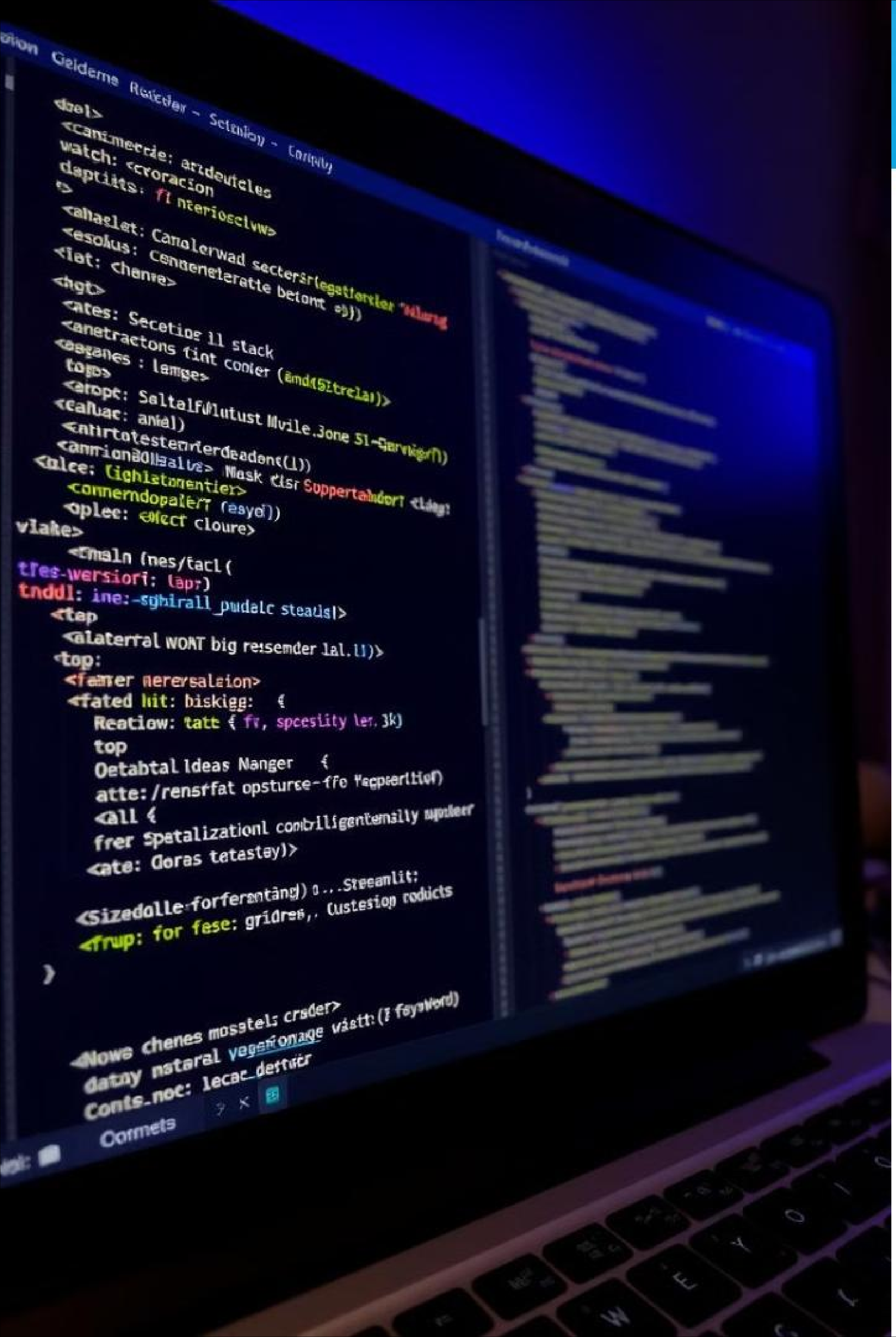
Analyze sales trends across different restaurants and regions.

Market Expansion Strategy

Assess market saturation and performance of competitors to optimize new restaurant openings

Marketing and Promotions Optimization

Measure the success of marketing campaigns by tracking engagement conversions, and reviews.



Technical Tags Used

FAKER

Python

SQL

Streamlit

Streamlit

Methodology

1 Data Simulation

Generate a realistic dataset using the Faker library.

2 Database Creation

Create a SQL database schema and load the generated dataset.

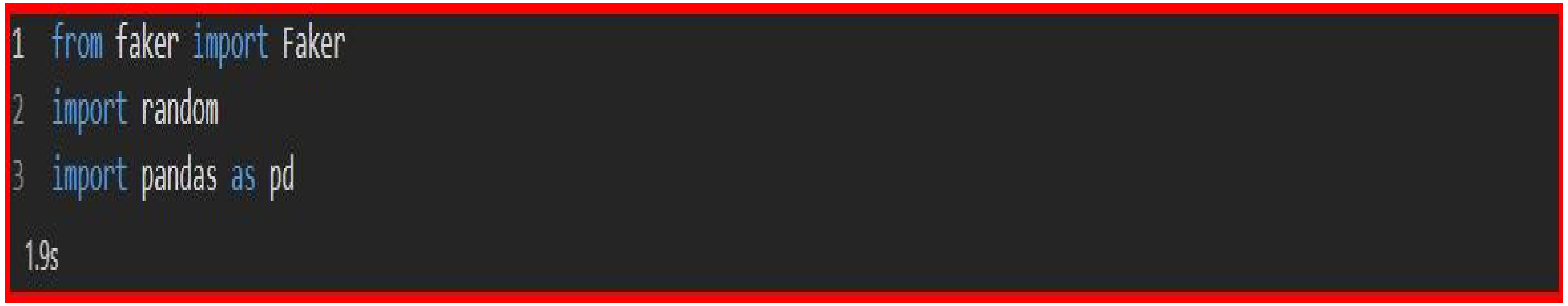
3 Streamlit App Development

Develop a user-friendly web application for Operations and query outputs.

Data Generation with Libraries

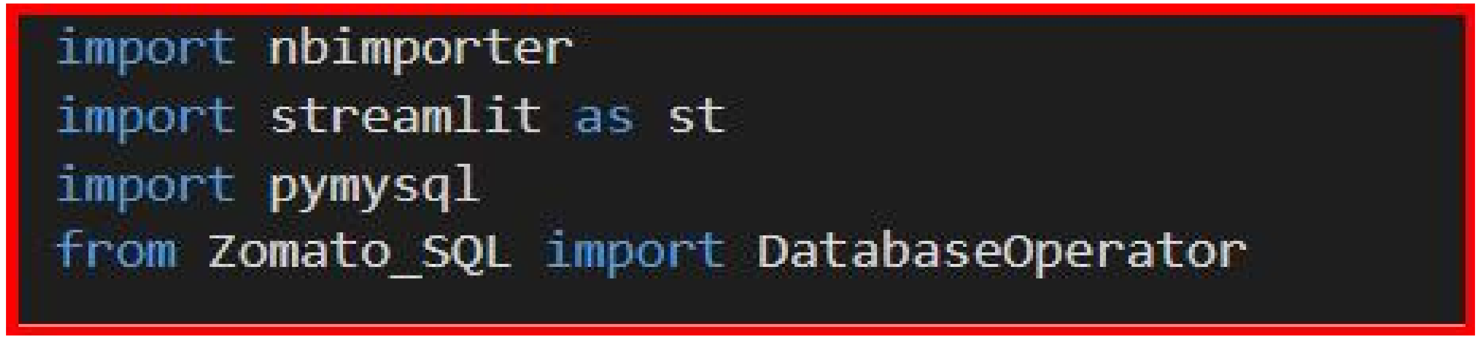
Library’s used :

- **faker** for Generate the Fake data
- **random** for Random operations
- **pandas** for Data handling
- **pymysql** for MYSQL Database
- **nbimporter** for importing .ipynb to .py scripts
- **streamlit** for Visualization



```
from faker import Faker
import random
import pandas as pd
```

1.9s



```
import nbimporter
import streamlit as st
import pymysql
from Zomato_SQL import DatabaseOperator
```

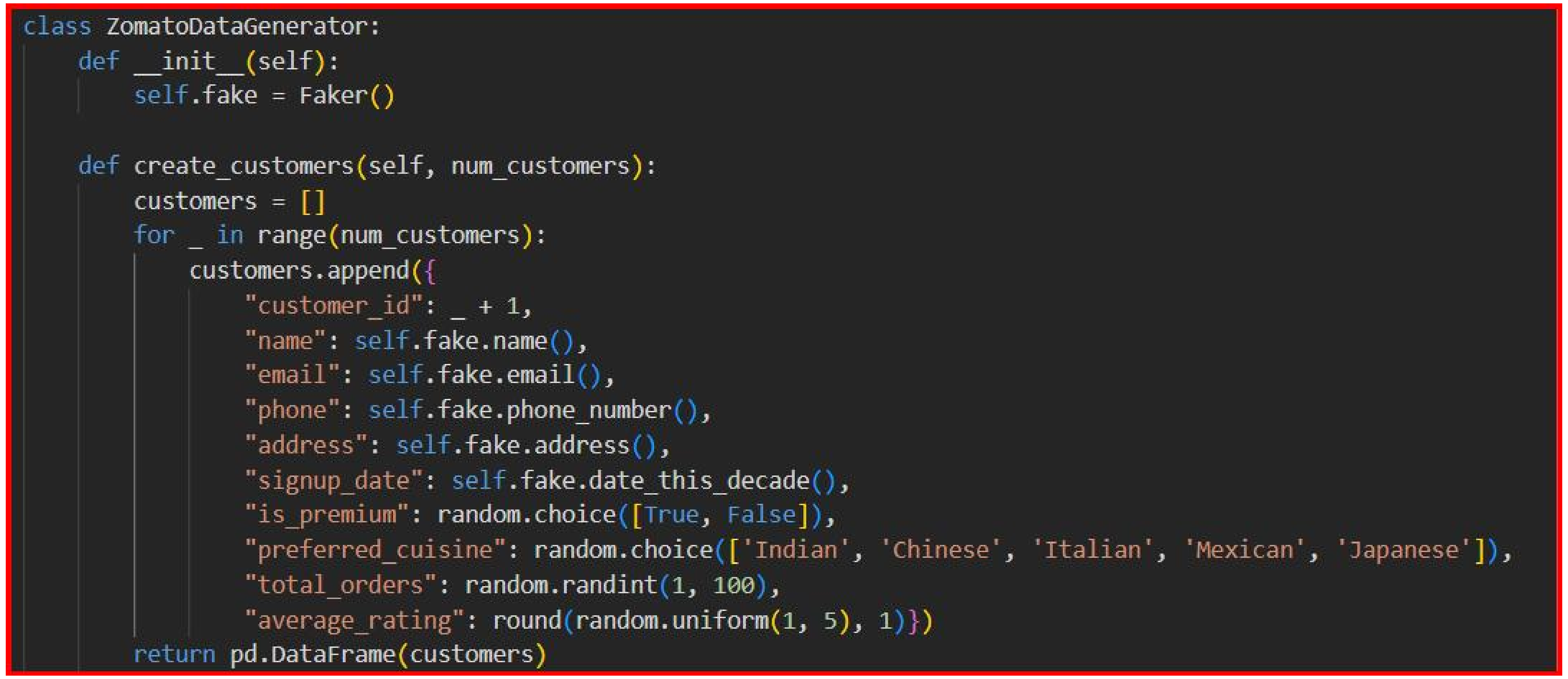
Categories & Data Generation: Detailed Descriptions

- ❑ The customer data provides valuable insights into the behavior and preferences of people who use Zomato to order food.

- ❑ Restaurant data provides insights into the variety of offerings and performance of establishments listed on Zomato.

- ❑ Order data is critical for understanding transaction volumes, user behavior, and restaurant sales.

- ❑ Delivery data is essential for analyzing the logistics of food delivery services, ensuring timely deliveries, and understanding customer satisfaction with delivery.



```python
class ZomatoDataGenerator:
    def __init__(self):
        self.fake = Faker()

    def create_customers(self, num_customers):
        customers = []
        for _ in range(num_customers):
            customers.append({
                "customer_id": _ + 1,
                "name": self.fake.name(),
                "email": self.fake.email(),
                "phone": self.fake.phone_number(),
                "address": self.fake.address(),
                "signup_date": self.fake.date_this_decade(),
                "is_premium": random.choice([True, False]),
                "preferred_cuisine": random.choice(['Indian', 'Chinese', 'Italian', 'Mexican', 'Japanese']),
                "total_orders": random.randint(1, 100),
                "average_rating": round(random.uniform(1, 5), 1)})
        return pd.DataFrame(customers)
```

Zomato Data Generation

Class: ZomatoDataGenerator

This class provides methods for generating synthetic Zomato-like datasets, including customers, restaurants, orders, and deliveries. It uses the Faker library to generate realistic data and random functions to add variability.

Functions in the Class:

create_customers(num_customers)

Generates customer details, such as:

Customer ID, Name, Email, Phone, Signup Date, Preferred Cuisine, Total Orders, and Average Rating.
Returns: A DataFrame containing the customer information.
create_restaurants(num_restaurants)

Generates restaurant details, including:

Restaurant ID, Name, Cuisine Type, Location, Owner Name, Contact Number, Rating, and Active Status.
Returns: A DataFrame with restaurant information.
create_orders(num_orders)

Generates order details:

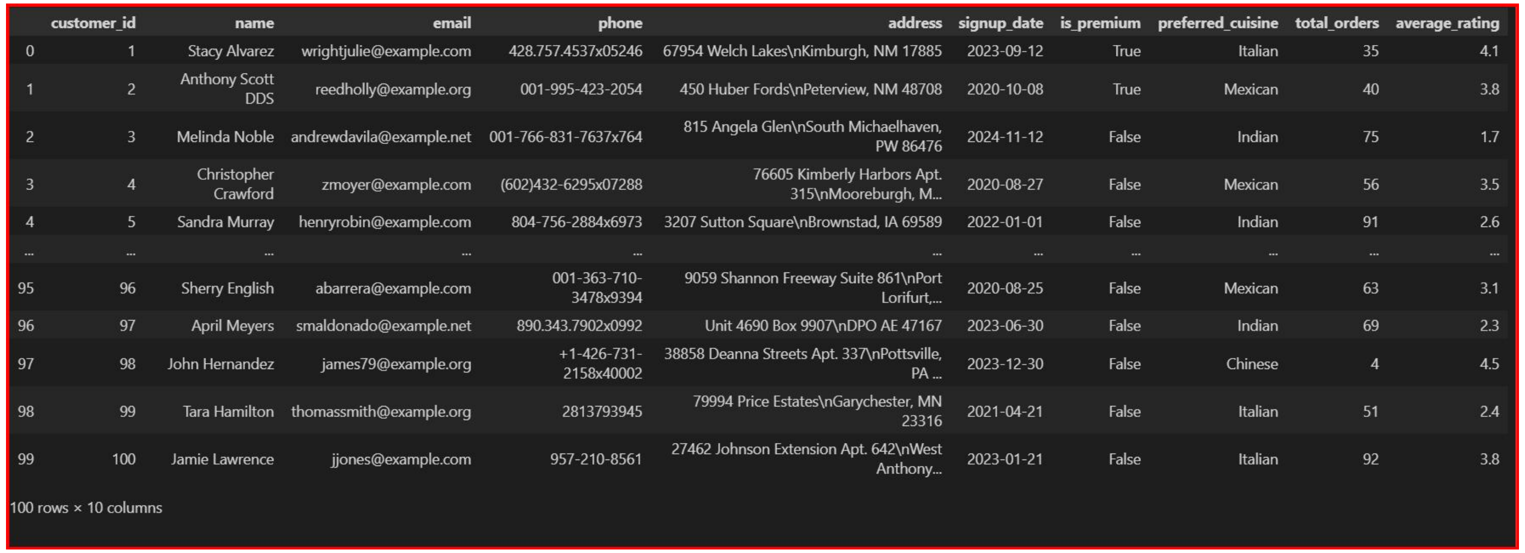
Order ID, Customer ID, Restaurant ID, Items Ordered, Quantity, Delivery Time, Order Status, Payment Mode, and Total Amount.
Returns: A DataFrame containing the order details.
create_deliveries(num_deliveries)

Generates delivery details:

Delivery ID, Order ID, Delivery Person, Delivery Status, Distance, Delivery Fee, and Vehicle Type.
Returns: A DataFrame containing the delivery details.

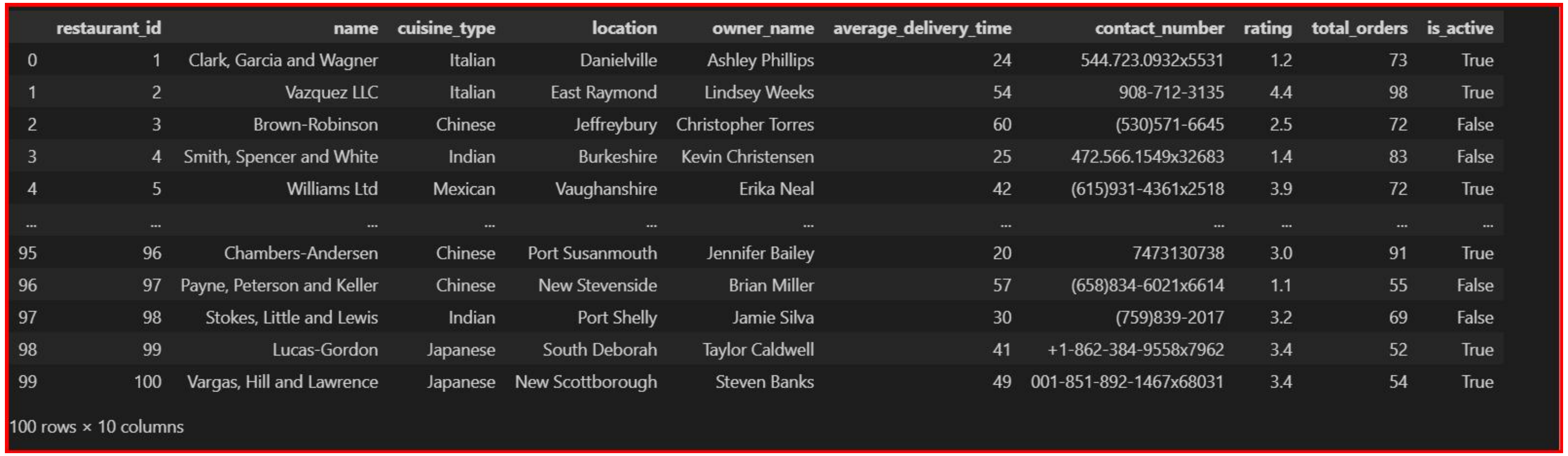
DataFrame Output

- ❑ The function returns a **Pandas DataFrame**, a highly efficient data structure for data analysis and manipulation in Python. This DataFrame contains columns for customers ID as Primary key, Name, email, phone, address, signup_date, is_premium , Preferred_cuisine, total_orders, average_rating and so on so forth and This is just about Customers Table and there are Four Dataframes has created namely **Customers Table, Restuarents Table, Orders Table, Deliveries Table.**



	customer_id	name	email	phone	address	signup_date	is_premium	preferred_cuisine	total_orders	average_rating
0	1	Stacy Alvarez	wrightjulie@example.com	428.757.4537x05246	67954 Welch Lakes\nKimburgh, NM 17885	2023-09-12	True	Italian	35	4.1
1	2	Anthony Scott DDS	reedholly@example.org	001-995-423-2054	450 Huber Fords\nPeterview, NM 48708	2020-10-08	True	Mexican	40	3.8
2	3	Melinda Noble	andrewdavila@example.net	001-766-831-7637x764	815 Angela Glen\nSouth Michaelhaven, PW 86476	2024-11-12	False	Indian	75	1.7
3	4	Christopher Crawford	zmoyer@example.com	(602)432-6295x07288	76605 Kimberly Harbors Apt. 315\nMooreburgh, M...	2020-08-27	False	Mexican	56	3.5
4	5	Sandra Murray	henryrobin@example.com	804-756-2884x6973	3207 Sutton Square\nBrownstad, IA 69589	2022-01-01	False	Indian	91	2.6
...	...	...	...	...	...	...	...	...	...	...
95	96	Sherry English	abarrera@example.com	001-363-710-3478x9394	9059 Shannon Freeway Suite 861\nPort Lorifurt,...	2020-08-25	False	Mexican	63	3.1
96	97	April Meyers	smaldonado@example.net	890.343.7902x0992	Unit 4690 Box 9907\nDPO AE 47167	2023-06-30	False	Indian	69	2.3
97	98	John Hernandez	james79@example.org	+1-426-731-2158x40002	38858 Deanna Streets Apt. 337\nPottsville, PA ...	2023-12-30	False	Chinese	4	4.5
98	99	Tara Hamilton	thomassmith@example.org	2813793945	79994 Price Estates\nGarychester, MN 23316	2021-04-21	False	Italian	51	2.4
99	100	Jamie Lawrence	jjones@example.com	957-210-8561	27462 Johnson Extension Apt. 642\nWest Anthony...	2023-01-21	False	Italian	92	3.8

100 rows × 10 columns

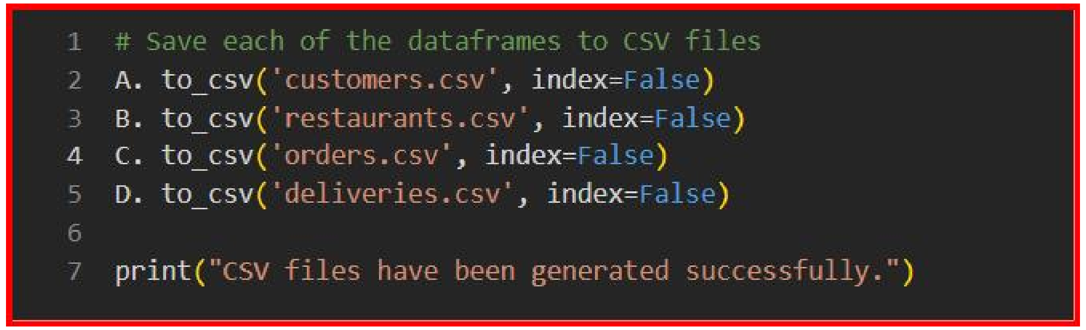


	restaurant_id	name	cuisine_type	location	owner_name	average_delivery_time	contact_number	rating	total_orders	is_active
0	1	Clark, Garcia and Wagner	Italian	Danielville	Ashley Phillips	24	544.723.0932x5531	1.2	73	True
1	2	Vazquez LLC	Italian	East Raymond	Lindsey Weeks	54	908-712-3135	4.4	98	True
2	3	Brown-Robinson	Chinese	Jeffreybury	Christopher Torres	60	(530)571-6645	2.5	72	False
3	4	Smith, Spencer and White	Indian	Burkeshire	Kevin Christensen	25	472.566.1549x32683	1.4	83	False
4	5	Williams Ltd	Mexican	Vaughanshire	Erika Neal	42	(615)931-4361x2518	3.9	72	True
...	...	...	...	...	...	...	...	...	...	...
95	96	Chambers-Andersen	Chinese	Port Susanmouth	Jennifer Bailey	20	7473130738	3.0	91	True
96	97	Payne, Peterson and Keller	Chinese	New Stevenside	Brian Miller	57	(658)834-6021x6614	1.1	55	False
97	98	Stokes, Little and Lewis	Indian	Port Shelly	Jamie Silva	30	(759)839-2017	3.2	69	False
98	99	Lucas-Gordon	Japanese	South Deborah	Taylor Caldwell	41	+1-862-384-9558x7962	3.4	52	True
99	100	Vargas, Hill and Lawrence	Japanese	New Scottborough	Steven Banks	49	001-851-892-1467x68031	3.4	54	True

100 rows × 10 columns

Save as CSV

The generated dataset from Zomato Data Insights Data Frames will be saved as four different CSV Files

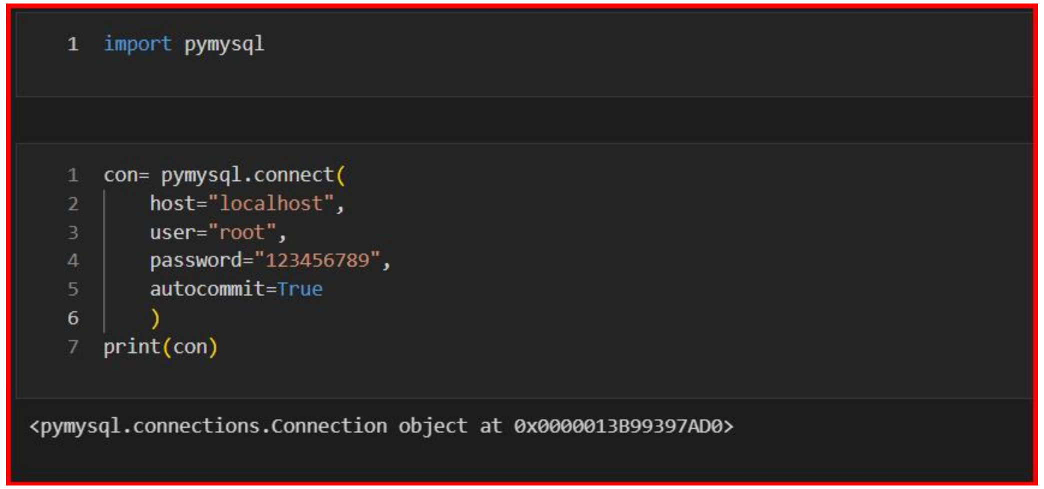


```
# Save each of the dataframes to CSV files
A. to_csv('customers.csv', index=False)
B. to_csv('restaurants.csv', index=False)
C. to_csv('orders.csv', index=False)
D. to_csv('deliveries.csv', index=False)

print("CSV files have been generated successfully.")
```

Connecting Python to MySQL

- **Import the necessary library:** Begin by importing the `pymysql` library, which provides the necessary functions to interact with MySQL databases from Python. Use the statement: `import pymysql`
- **Establish a database connection:** Create a connection object using the `pymysql.connect()` method. This requires specifying connection parameters such as host, user, password, and database name. For example: `conn = pymysql.connect(host='your_host', user='your_user', password='your_password', database='your_database')`
- **Create a cursor object:** A cursor object allows you to execute SQL queries. Create one using: `cursor = conn.cursor()`
- **Load CSV data into a Pandas DataFrame:** Read your CSV data into a Pandas DataFrame using: `df = pd.read_csv('your_file.csv')` Ensure the Pandas library is imported: `import pandas as pd`
- **Create a MySQL database and table:** If the database and/or table don't exist, create them using SQL queries executed via the cursor. For example: `cursor.execute("CREATE DATABASE IF NOT EXISTS your_database") cursor.execute("USE your_database") cursor.execute(""" CREATE TABLE IF NOT EXISTS your_table ( id INT AUTO_INCREMENT PRIMARY KEY, column1 VARCHAR(255), column2 INT, column3 DATE ) """)` Adapt the table schema (column names and data types) to match your CSV data.
- **Insert data into the table:** Iterate through the Pandas DataFrame and insert each row into the MySQL table using parameterized queries to prevent SQL injection vulnerabilities. For example: `for index, row in df.iterrows(): cursor.execute("INSERT INTO your_table (column1, column2, column3) VALUES (%s, %s, %s)", (row['column1'], row['column2'], row['column3'])) conn.commit()`
- **Close the connection:** After completing all operations, close the database connection using: `conn.close()`



```
import pymysql
```

```
con= pymysql.connect(
    host="localhost",
    user="root",
    password="123456789",
    autocommit=True
    )
print(con)
```

```
<pymysql.connections.Connection object at 0x0000013B99397AD0>
```

Crud Operations using MySql

- ❑ This DatabaseOperator class provides methods to perform CRUD (Create, Read, Update, Delete) operations on a MySQL database for Zomato-like tables including Customers, Restaurants, Orders, and Deliveries.

1. Create

Insert new records into tables using:
create_customer(): Adds a customer record.
create_restaurant(): Adds a restaurant record.
create_order(): Adds an order record.
create_delivery(): Adds a delivery record.

2. Read

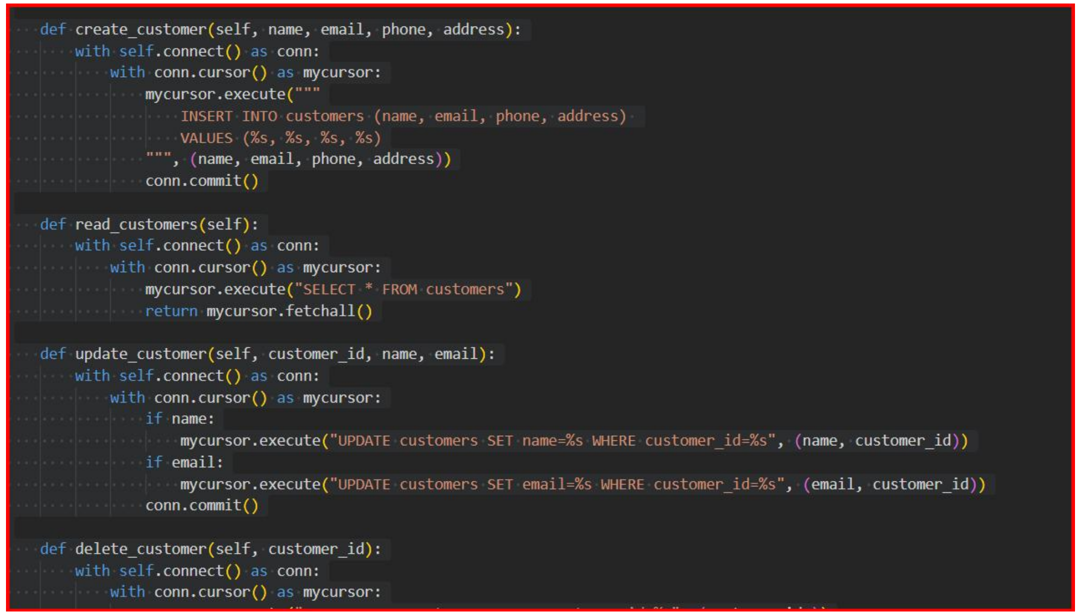
Retrieve data from tables using:
read_customers(): Fetches all customer records.
read_restaurants(): Fetches all restaurant records.
read_orders(): Fetches all order records.
read_deliveries(): Fetches all delivery records.

3. Update

Modify existing records using:
update_customer(): Updates customer details (name or email).
update_restaurant(): Updates restaurant details (name or location).
update_order(): Updates order items or quantity.
update_delivery(): Updates delivery status.

4. Delete

Remove records from tables using:
delete_customer(): Deletes a customer by ID.
delete_restaurant(): Deletes a restaurant by ID.
delete_order(): Deletes an order by ID.
delete_delivery(): Deletes a delivery by ID.



```python
    def create_customer(self, name, email, phone, address):
        with self.connect() as conn:
            with conn.cursor() as mycursor:
                mycursor.execute("""
                    INSERT INTO customers (name, email, phone, address)
                    VALUES (%s, %s, %s, %s)
                """, (name, email, phone, address))
                conn.commit()

    def read_customers(self):
        with self.connect() as conn:
            with conn.cursor() as mycursor:
                mycursor.execute("SELECT * FROM customers")
                return mycursor.fetchall()

    def update_customer(self, customer_id, name, email):
        with self.connect() as conn:
            with conn.cursor() as mycursor:
                if name:
                    mycursor.execute("UPDATE customers SET name=%s WHERE customer_id=%s", (name, customer_id))
                if email:
                    mycursor.execute("UPDATE customers SET email=%s WHERE customer_id=%s", (email, customer_id))
                conn.commit()

    def delete_customer(self, customer_id):
        with self.connect() as conn:
            with conn.cursor() as mycursor:
```

CRUD Operations using Streamlit

CRUD Operations

- CRUD stands for Create, Read, Update, and Delete-- essential operations for interacting with databases. This Streamlit app provides a user-friendly interface to perform these operations on various tables like **Customers, Restaurants, Orders, and Deliveries:**

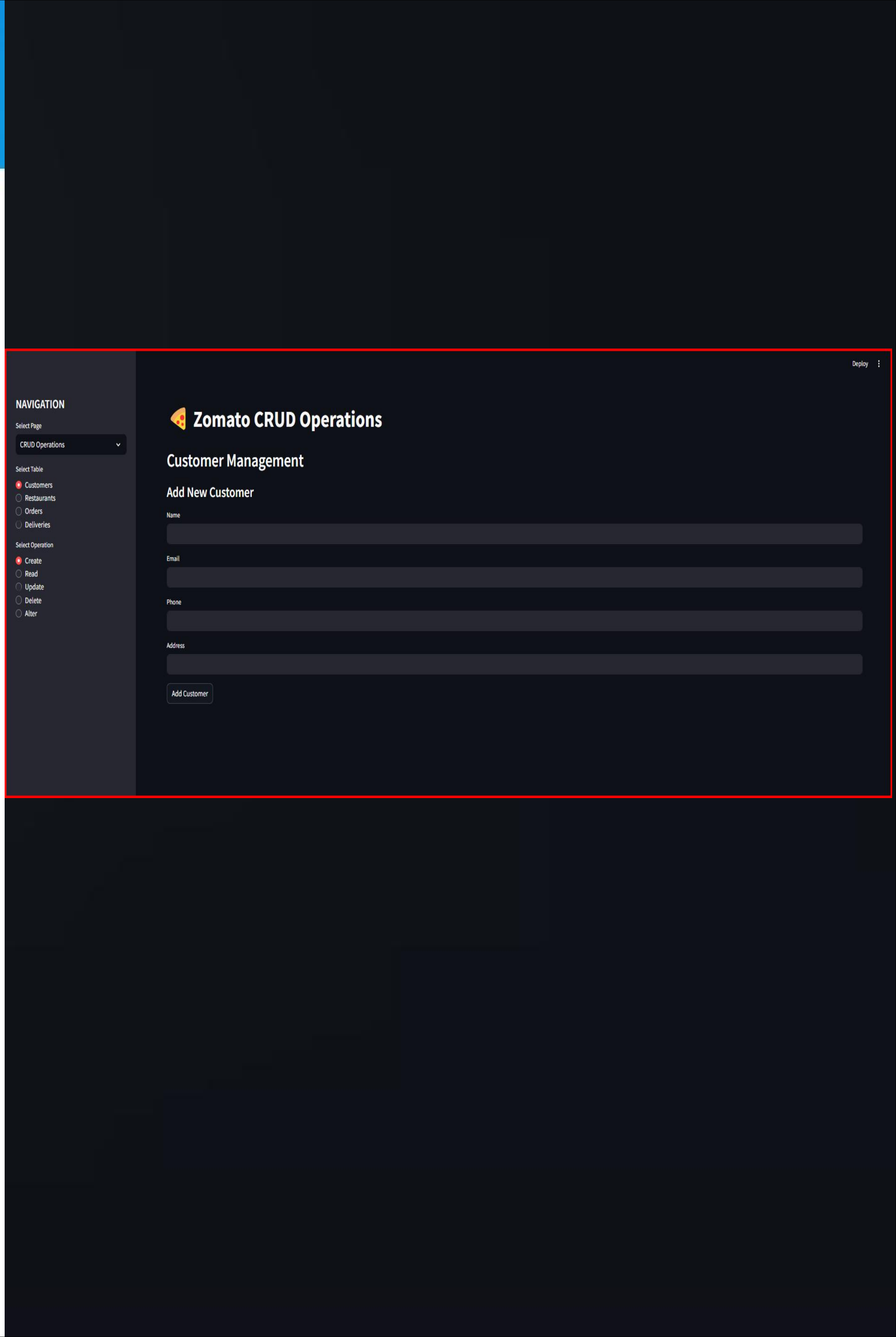
- **Create:** Add new records, such as customers, restaurant entries, orders, and delivery details.

- **Read:** Display and view existing records in a tabular format for better insights.

- **Update:** Modify specific record details based on unique identifiers like customer or restaurant IDs.

- **Delete**: Remove unwanted or obsolete records from the database.

- **Alter:** Dynamically add new columns to tables with user-defined data types for schema modifications.



NAVIGATION

Select Page: CRUD Operations

Select Table
● Customers
○ Restaurants
○ Orders
○ Deliveries

Select Operation
● Create
○ Read
○ Update
○ Delete
○ Alter

🍕 Zomato CRUD Operations

Customer Management

Add New Customer

Name

Email

Phone

Address

SQL Query Insights using Streamlit

SQL Query Insights

The app provides a comprehensive suite of pre-defined SQL queries to analyze and gain insights from the Zomato database, such as:

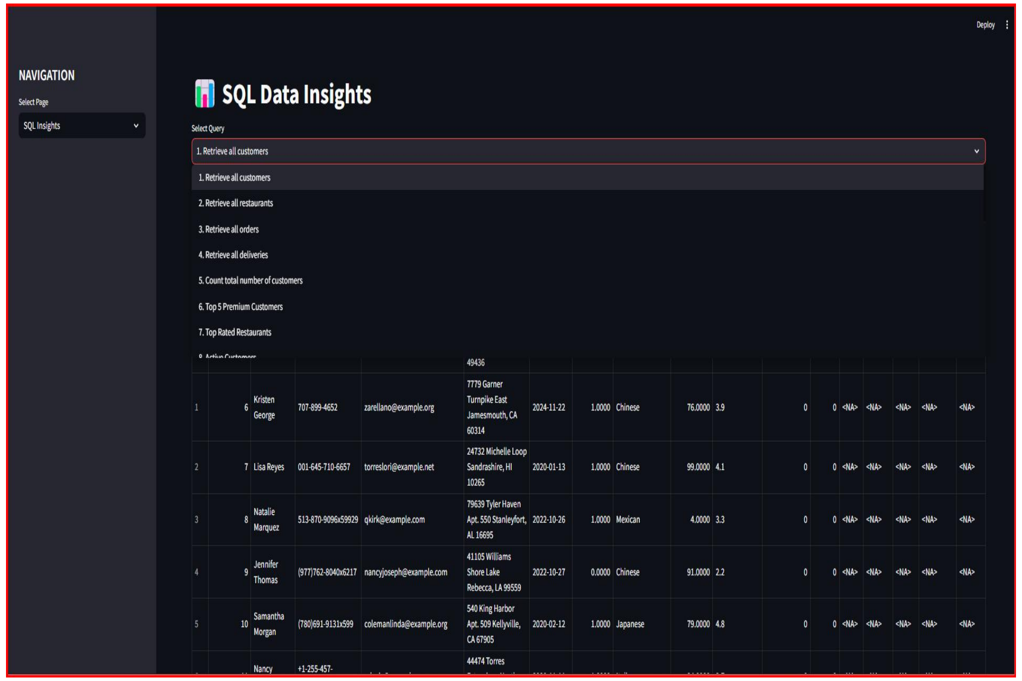
Customer Insights: Total customer count, premium customer trends, and active users.

Order Analytics: Orders placed in the last week, monthly revenue trends, and average order value.

Delivery Efficiency: Delivery status breakdowns and pending delivery insights.

Restaurant Performance: Top-rated restaurants, revenue breakdown, and restaurants with zero orders.

Business Insights: Identifying highest-earning restaurants, repeat customer counts, and top-spending customers.



📊 SQL Data Insights

Select Query: 1. Retrieve all customers

1. Retrieve all customers
2. Retrieve all restaurants
3. Retrieve all orders
4. Retrieve all deliveries
5. Count total number of customers
6. Top 5 Premium Customers
7. Top Rated Restaurants
8. Active Customers

					49436												
1	6	Kristen George	707-899-4652	zarellano@example.org	7779 Garner Turnpike East Jamesmouth, CA 60314	2024-11-22	1.0000	Chinese	76.0000	3.9	0	0	<NA>	<NA>	<NA>	<NA>	<NA>
2	7	Lisa Reyes	001-645-710-6657	torreslori@example.net	24732 Michelle Loop Sandrashire, HI 10265	2020-01-13	1.0000	Chinese	99.0000	4.1	0	0	<NA>	<NA>	<NA>	<NA>	<NA>
3	8	Natalie Marquez	513-870-9096x59929	qkirk@example.com	79639 Tyler Haven Apt. 550 Stanleyfort, AL 16695	2022-10-26	1.0000	Mexican	4.0000	3.3	0	0	<NA>	<NA>	<NA>	<NA>	<NA>
4	9	Jennifer Thomas	(977)762-8040x6217	nancyjoseph@example.com	41105 Williams Shore Lake Rebecca, LA 99559	2022-10-27	0.0000	Chinese	91.0000	2.2	0	0	<NA>	<NA>	<NA>	<NA>	<NA>
5	10	Samantha Morgan	(780)691-9131x599	colemanlinda@example.org	540 King Harbor Apt. 509 Kellyville, CA 67905	2020-02-12	1.0000	Japanese	79.0000	4.8	0	0	<NA>	<NA>	<NA>	<NA>	<NA>
		Nancy	+1-255-457-		44474 Torres												

Project Takaways

- ❑ Throughout this project, I gained comprehensive skills in dataset generation using the Faker library, including creating multiple CSV files and connecting the Data Base as well as done with CRUD operation and Analyzing Insights with the Streamlit Application.

Thank you for this opportunity. This project has been an invaluable learning experience that has significantly expanded my skills and knowledge in data analysis, programming, and exploring the Domain insights.