A Minor Project Report

On

Customer Rating on an App or Website

Bachelor of Technology

In

Computer Science and Engineering

2023-2027

By

Name – Dhruv Chirania

Registration Number – 23FE10CSE00654

MANIPAL UNIVERSITY
JAIPUR

Under the supervision of

Mr. JAY SHARMA

School Of Computer Science and Engineering

Manipal University Jaipur, Jaipur, Rajasthan, India JUL-

NOV 2024

Introduction: -

Customer ratings are a fundamental component of understanding user satisfaction and feedback on digital platforms. This project aims to simulate and manage customer ratings for an app or website using Python. By generating random customer data, including names, IDs, ages, ratings, and mobile numbers, the project stores these details in a structured data frame. Furthermore, it filters and identifies customers with an average rating of 3.5 or higher. This tool provides an automated approach to handle customer data and evaluate their ratings effectively.

This report outlines the logic, functionality, and technical details of the Customer Rating on an App or Website, supported by the Python code and its corresponding output.

How the project is Working: -

Objective

- To generate random customer data, including names, IDs, ages, mobile numbers, and ratings (as floating-point numbers between 1 and 5).
- To store the generated data in a panda DataFrame for permanent storage.
- To filter and display customers with an average rating of 3.5 or higher.

Initialization

- The project initializes a string of 50 characters to generate customer names.

- The random module is used to create customer ratings, IDs, ages, and mobile numbers.

User Input Handling

- Random customer ratings are generated in the range of 1 to 5 (as floating-point numbers).
- Names are dynamically created by extracting substrings of varying lengths from the predefined 50-character string.

Data Generation and Storage

- Random data for customer IDs, ages, and mobile numbers is generated using the random module.
- The data is stored in a panda DataFrame with columns representing customer properties and rows representing individual customers.
- The DataFrame is written to a CSV file to ensure permanent storage.

Data Fetching and Processing

- The data is read from the stored CSV file into a panda DataFrame.
- Each customer's data is stored in a Python class object, and the objects are stored in a list.

Completion or Exit

- The list of customer objects is iterated to calculate average ratings.
- Customers with an average rating of 3.5 or higher are identified and displayed.

Code: -

```
import random
import pandas as pd
import string

# Step 1: Generate random data
num_customers = 10
data = {
"CustomerID": [random.randint(1000, 9999) for _ in range(num_customers)],
"Name": [
"".join(random.choices(string.ascii_letters,  k=random.randint(5, 10)))
for _ in range(num_customers)
],
"Age": [random.randint(18, 65) for _ in range(num_customers)],
"Mobile": [random.randint(7000000000, 9999999999) for _ in range(num_customers)],
"Rating": [round(random.uniform(1, 5), 1) for _ in range(num_customers)],
}

# Step 2: Store data in a DataFrame and save to CSV
df = pd.DataFrame(data)
df.to_csv("customer_data.csv", index=False) # Saved locally for persistence

# Step 3: Define the Customer class
class Customer:
def __init__(self, customer_id, name, age, mobile, rating):
self.customer_id  = customer_id
self.name = name
self.age = age
self.mobile = mobile
self.rating = rating

def __str__(self):
return f"{self.name}, Rating: {self.rating}"

# Step 4: Load data from CSV and create Customer objects
df = pd.read_csv("customer_data.csv")
customers = [
Customer(row["CustomerID"], row["Name"], row["Age"], row["Mobile"], row["Rating"])
for _, row in df.iterrows()
]

# Step 5: Find and display customers with an average rating >= 3.5
eligible_customers = [customer for customer in customers if customer.rating >= 3.5]

print("Customers with an average rating >= 3.5:")
for customer in eligible_customers:
print(customer)
```

Output:

Customers with an average rating >= 3.5:

hEXOcAv, Rating: 4.9

ymYNi, Rating: 3.8

ndlrcWjw, Rating: 4.9

GYhehVy, Rating: 3.9

DQrsZmCdIG, Rating: 4.8

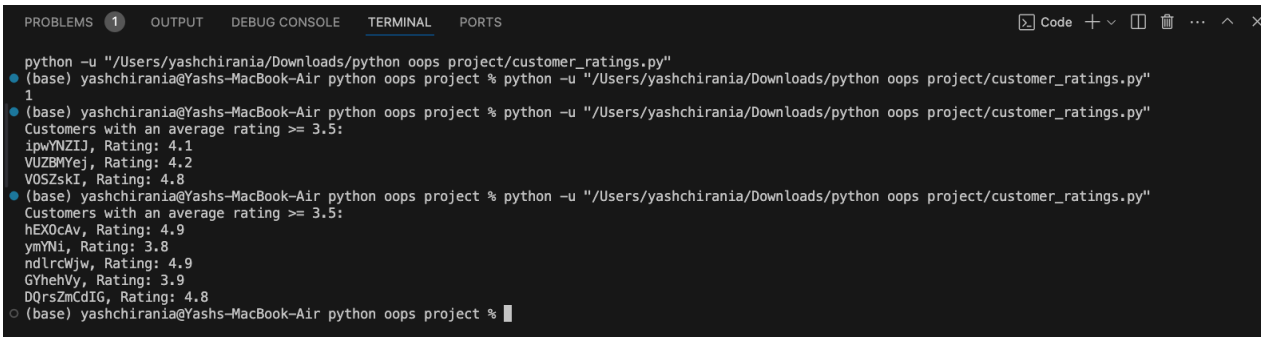
During Execution:

- Generates random customer data and stores it in a CSV file.

Upon Completion:

- Displays customers with an average rating of 3.5 or higher.

Success Screen:


PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Code
python -u "/Users/yashchirania/Downloads/python oops project/customer_ratings.py"
(base) yashchirania@Yashs-MacBook-Air python oops project % python -u "/Users/yashchirania/Downloads/python oops project/customer_ratings.py"
1
(base) yashchirania@Yashs-MacBook-Air python oops project % python -u "/Users/yashchirania/Downloads/python oops project/customer_ratings.py"
Customers with an average rating >= 3.5:
ipwYNZIJ, Rating: 4.1
VUZBMYej, Rating: 4.2
VOSZskI, Rating: 4.8
(base) yashchirania@Yashs-MacBook-Air python oops project % python -u "/Users/yashchirania/Downloads/python oops project/customer_ratings.py"
Customers with an average rating >= 3.5:
hEXOcAv, Rating: 4.9
ymYNi, Rating: 3.8
ndlrcWjw, Rating: 4.9
GYhehVy, Rating: 3.9
DQrsZmCdIG, Rating: 4.8
(base) yashchirania@Yashs-MacBook-Air python oops project %

Detailed Synopsis:

Title: Development of a Customer Rating Management System

1. Background and Motivation

Customer feedback and ratings are indispensable for understanding user satisfaction and improving service quality in the digital ecosystem. Businesses heavily rely on this data to make informed decisions, enhance user experience, and maintain a competitive edge. The manual handling of such data can be cumbersome, time-consuming, and prone to errors.

This project addresses the need for an automated and efficient system to simulate, store, and process customer ratings. By leveraging the power of Python programming and data processing libraries, the project simplifies the management of customer feedback and focuses on identifying high-rated customers for targeted analysis.

2. Objectives

The project aims to design and implement a system capable of:

1. Generating random customer data for testing purposes, including unique attributes like ID, name, age, mobile number, and ratings.
2. Structuring the generated data into a tabular format for efficient processing and exporting it to a CSV file for persistent storage.
3. Utilizing Python's object-oriented programming (OOP) principles to manage individual customer data as objects.
4. Filtering and analyzing the data to identify customers with favorable ratings (≥ 3.5), providing actionable insights.

3. Problem Statement

Handling customer data often involves large datasets, which require systematic approaches to generation, storage, and processing. This project demonstrates a scalable

and automated solution for creating synthetic customer data, persisting it in CSV format, and analyzing it for actionable outcomes.

The proposed system bridges the gap between data generation and data analysis by seamlessly integrating Python-based automation tools.

4. Scope of the Project

The system highlights the following capabilities:

- Random generation of customer details to simulate real-world scenarios.
- Use of pandas for efficient data manipulation and storage.
- Implementation of OOP for clean, modular code and future extensibility.
- Export of data to CSV for external integration or further analysis.
- Basic filtering logic to extract insights from customer ratings.

This foundation can be extended in the future to incorporate advanced data analysis, visualization techniques, or integration with machine learning models for predicting customer satisfaction trends.

5. System Design

5.1 Data Generation

- **Customer ID**: A 4-digit unique identifier is randomly generated.
- **Name**: Random strings of alphabetic characters simulate customer names, with lengths ranging from 5 to 10 characters.
- **Age**: Random values between 18 and 65 represent customer demographics.
- **Mobile Number**: Ten-digit numbers are generated to mimic real-world mobile numbers.
- **Rating**: Floating-point numbers between 1.0 and 5.0 are generated to simulate customer feedback scores.

5.2 Data Storage and Processing

- **DataFrame Structure**: The generated data is organized into a pandas DataFrame, making it suitable for tabular storage and easy export.
- **CSV Export**: Data is saved in a CSV file format to enable persistence and reusability.

5.3 Class Implementation

- **Customer Class**: Each customer is represented as a Python class object, encapsulating their details such as ID, name, age, mobile number, and rating.
- **Object Manipulation**: Python's OOP principles are employed to filter and manage customer data using these objects.

5.4 Analysis and Filtering

- **Filtering Logic**: Customers with a rating of 3.5 or higher are extracted and displayed.
- **Insights Generation**: Provides a basic framework for identifying high-value customers.

6. Tools and Technologies Used

1. **Python**: Primary programming language for implementation.
2. **Pandas**: Library for efficient data handling and manipulation.
3. **CSV**: File format for data storage and persistence.
4. **Random & String Modules**: For generating synthetic data.

7. Expected Outcome

1. A CSV file containing synthetic customer data for further processing or analysis.
2. A Python-based system capable of filtering and displaying customers with a rating of 3.5 or higher.
3. A modular, scalable framework that can be extended for real-world applications.

8. Applications and Future Enhancements

Applications

- Customer feedback analysis for business intelligence.
- Educational demonstrations of Python OOP and data handling concepts.
- Development of test datasets for machine learning or data science projects.

Future Enhancements

- Real-time feedback integration through APIs or web scraping.
- Advanced visualization of customer ratings using tools like Matplotlib or Seaborn.
- Predictive analytics using machine learning to forecast user satisfaction trends.

9. Conclusion

This project provides an efficient, automated system for managing and analyzing customer ratings. It demonstrates the practical application of Python for solving real-world problems and sets the stage for more advanced tools in the future. By focusing on modularity and scalability, the system is poised for further enhancements and practical deployment in diverse industries.