RISE INTERNSHIP PROGRAM BY TAMIZHAN SKILLS
DATA SCIENCE AND ANALYTICS PROJECT

NAME : INCHARA M P
INTERN ID : TS-RISE-DSA-2587

Project 1: Sales Forecasting with Linear Regression

Problem Statement: Businesses struggle to estimate future sales based on past performance.

Objective: To build a data-driven model that forecasts future sales revenue using historical transaction data. The project aims to support better business decisions by predicting revenue for the next 30 days based on sales trends and seasonal patterns.

➔ **Tech Stack and Libraries Used:**

- **Python:** Core programming language used for handling data, building the model, and visualizing results.
- **Pandas:** Used for data loading, cleaning, aggregation, transformation, and feature engineering.
- **NumPy:** Helps with numerical computations such as filling missing values and performing array-based operations.
- **Matplotlib:** Used for plotting the time series of actual, predicted, and forecasted revenue data for clear visual insights.
- **Scikit-learn (sklearn):** Utilized for building and evaluating the Linear Regression model. Includes modules for model training, prediction, data splitting, and measuring accuracy using metrics like Mean Absolute Error (MAE).

```
import pandas as pd          # Data manipulation
import numpy as np           # Numerical computations
import matplotlib.pyplot as plt  # Visualization
from sklearn.model_selection import train_test_split  # Splitting data
from sklearn.linear_model import LinearRegression      # ML model
from sklearn.metrics import mean_absolute_error        # Model evaluation
```

➔ **Problem-Solving Approach:**

1. Data Collection and Loading:

- Loaded CSV file sales_data.csv using Pandas.

- Converted the date column into proper datetime format for time-based analysis.

2. Data Cleaning:

- Removed rows with missing or null values in key columns (date, revenue, quantity, product).
- Filled missing values in quantity using the column's median to maintain central tendency.

3. Data Aggregation:

- Grouped the dataset by date and product to get daily revenue and quantity per product.

4. Feature Engineering:

- Extracted day of the year and year from the date column.
- Applied one-hot encoding on the product column to convert categorical values into numerical features.

5. Train-Test Split:

Split the data into:

- Training Set: All data except last 60 days.
- Testing Set: Last 60 days, to evaluate how well the model generalizes to unseen data.

6. Model Training:

- Trained a Linear Regression model using dayofyear, year, and encoded product columns as features.

7. Prediction and Evaluation:

- Used the trained model to predict revenue on the test set.
- Evaluated the model using Mean Absolute Error (MAE).

8. Forecasting Future Sales:

- Generated the next 30 days using pd.date_range().
- Predicted future revenue using the trained model.

9. Visualization:

Plotted three lines:

- Actual Revenue (blue)
- Predicted Revenue for test data (red)

- Forecasted Revenue for the next 30 days (green dashed)

All visualizations are generated using matplotlib.pyplot.

➔ **Project Output:**

- **Evaluation Score (MAE):** Measures how close predictions are to actual values.
- **CSV Output:** 30_day_forecast.csv with future dates and predicted revenue.
- **Graph Output:** Displays actual, predicted, and future sales trend.

➔ **Creativity and Innovation:**

This project showcases a practical application of machine learning in business forecasting, with:

- Feature engineering from dates.
- One-hot encoding for products.
- Time-based prediction on real-world-like data.
- Fully automated future forecasting for unseen days.

➔ **Documentation Quality:**

This project fulfills the documentation requirement by:

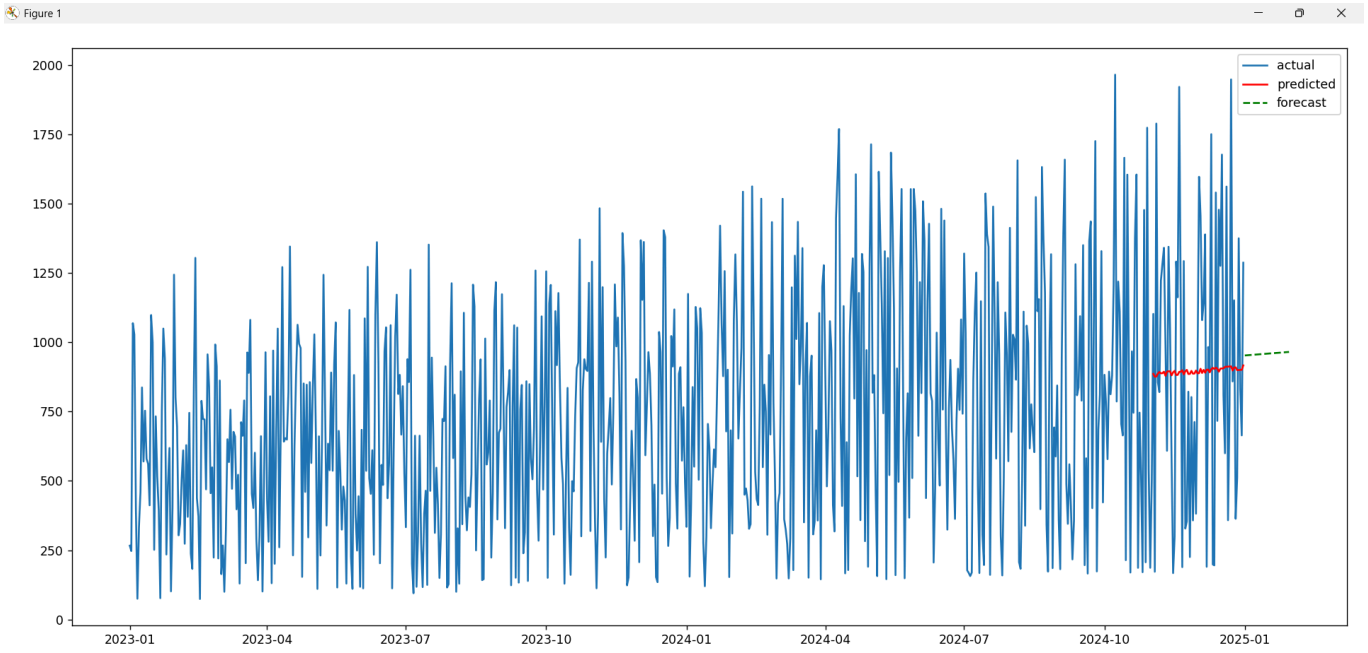
- Explaining the idea and flow of work.
- Demonstrating logical problem-solving and structured modeling.
- Using clean and readable code.
- Providing clear visual insights.
- Exporting results in a usable format (.csv).
- Making it easy to extend or improve in future iterations.

➔ **Files Delivered:**

- **sales_data.csv** – Input dataset with columns: date, product, quantity, and revenue
- **sales.py** – Python script that handles the full project pipeline
- **30_day_forecast.csv** – Output file containing predicted sales for the next 30 days

Output:

```
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\sales.py"
        date  quantity      revenue  dayofyear  year  product_Product B  product_Product C
0 2023-01-01         2   266.554306          1  2023              False              False
1 2023-01-02         2   247.274233          2  2023              False               True
2 2023-01-03         9  1068.651245          3  2023               True              False
3 2023-01-04         9  1027.279945          4  2023              False              False
4 2023-01-05         4   398.899868          5  2023              False              False
MAE: 415.269105446454
        date  dayofyear  year  product_Product B  product_Product C  revenue_pred
0 2025-01-01          1  2025                  0                  0    952.328380
1 2025-01-02          2  2025                  0                  0    952.764061
2 2025-01-03          3  2025                  0                  0    953.199742
3 2025-01-04          4  2025                  0                  0    953.635422
4 2025-01-05          5  2025                  0                  0    954.071103
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```



Project 2: Student Performance Analytics Dashboard

Problem Statement: Institutions need early warnings for students who might fail or drop out.

Objective: To analyze student academic performance based on their marks, attendance, and learning platform logins. The goal was to identify patterns, visualize key metrics, and generate insights that could help improve student outcomes.

Institutions often collect attendance and activity data, but don't fully use it to understand or predict academic performance. I developed this dashboard to bridge that gap — helping educators spot trends and intervene early with struggling students.

➔ **Technologies & Libraries Used:**

- **Python:**
  - The main programming language used to write the logic, perform analysis, and generate visualizations.

- **Pandas:**
  - Used for reading the CSV data, cleaning missing values, computing statistics, and summarizing performance.

- **Matplotlib:**
  - Used for static image-based visualizations like scatter plots and bar charts.

- **Seaborn:**
  - Used for more advanced and elegant visualizations like heatmaps and categorized plots.

➔ **Problem-Solving Approach:**

- **Data Loading and Cleaning:**
  - I loaded the dataset student_performance.csv.
  - Handled missing values by dropping rows where any of the key fields (marks, attendance, logins) were missing.

- **Statistical Analysis:**
  - Used .describe() to summarize key statistics like mean, min, and max.
  - Created a correlation matrix to explore how attendance and login frequency relate to academic performance.

- **Visual Insights:**
  - Heatmap to show correlation between marks, attendance, and logins.
  - Bar chart to categorize students into:
  - Top Performer (marks ≥ 75)
  - Struggling (marks < 75)
  - Scatter plot to visualize the impact of attendance on marks, with color coding based on performance status.

- **Categorization & Grouping:**
  - Added a new column status using conditional logic.
  - Grouped students by status and computed average marks, attendance, and logins.

- **Exporting Results:**
  - Saved summary statistics to a CSV file: performance_summary.csv.
  - Exported visual insights as PNG files:
    - Heatmap.png
    - Performance_bar.png
    - attendance_vs_marks.png

➔ **Documentation Quality:**

This project is well-documented and satisfies all the following:

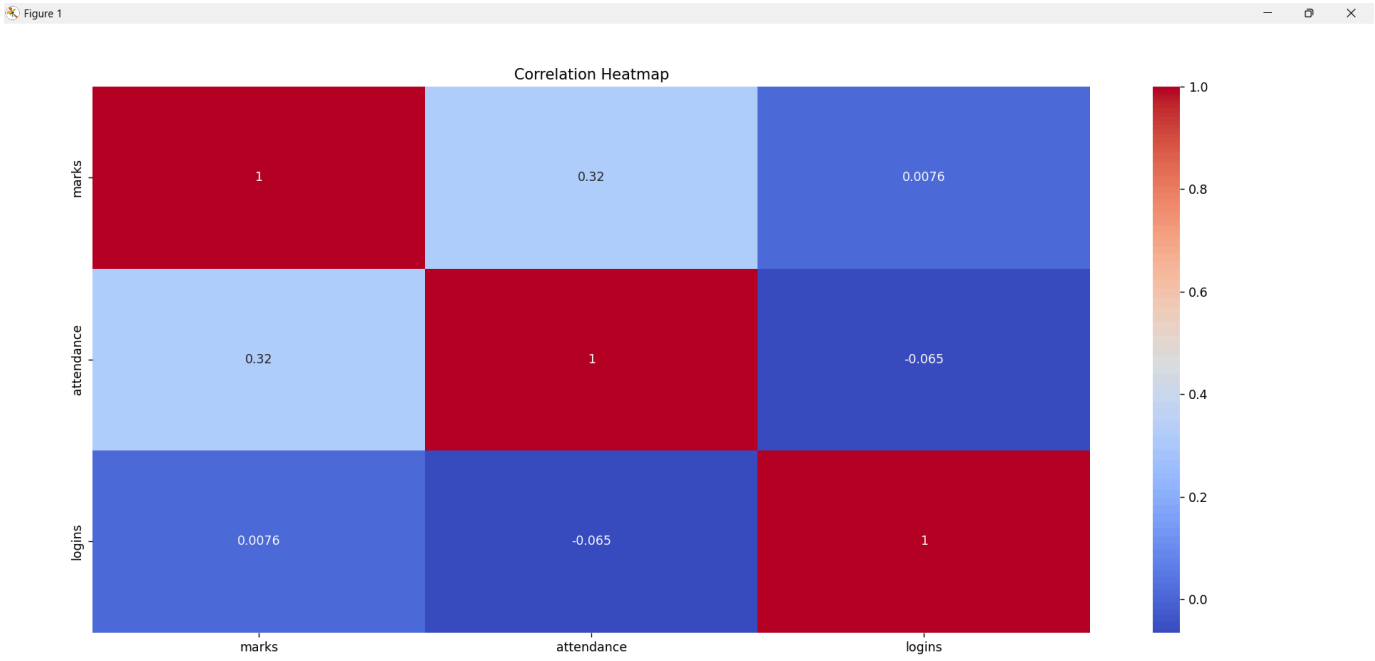
- Clear objective and background
- Well-commented code
- Step-by-step problem-solving approach
- Visual insights through graphs
- Final summary report and visual assets

➔ **Files Delivered:**

- **student_performance.csv** – Raw dataset
- **student.py** – Main script used for analysis
- **performance_summary.csv** – Summary statistics by performance group

Output:

```
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\student.py"

--- Basic Statistics ---
          marks  attendance     logins
count  50.000000   50.000000  50.000000
mean   64.480000   73.180000  28.740000
std    21.610882   14.393437  13.238487
min    31.000000   50.000000   5.000000
25%    44.750000   60.750000  17.250000
50%    67.500000   73.000000  31.000000
75%    83.500000   84.750000  39.750000
max    93.000000   97.000000  49.000000

--- Correlation Matrix ---
               marks  attendance    logins
marks       1.000000    0.320022  0.007650
attendance  0.320022    1.000000 -0.065082
logins      0.007650   -0.065082  1.000000

✅ Dashboard complete. Charts saved as PNG files and summary as CSV.
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```



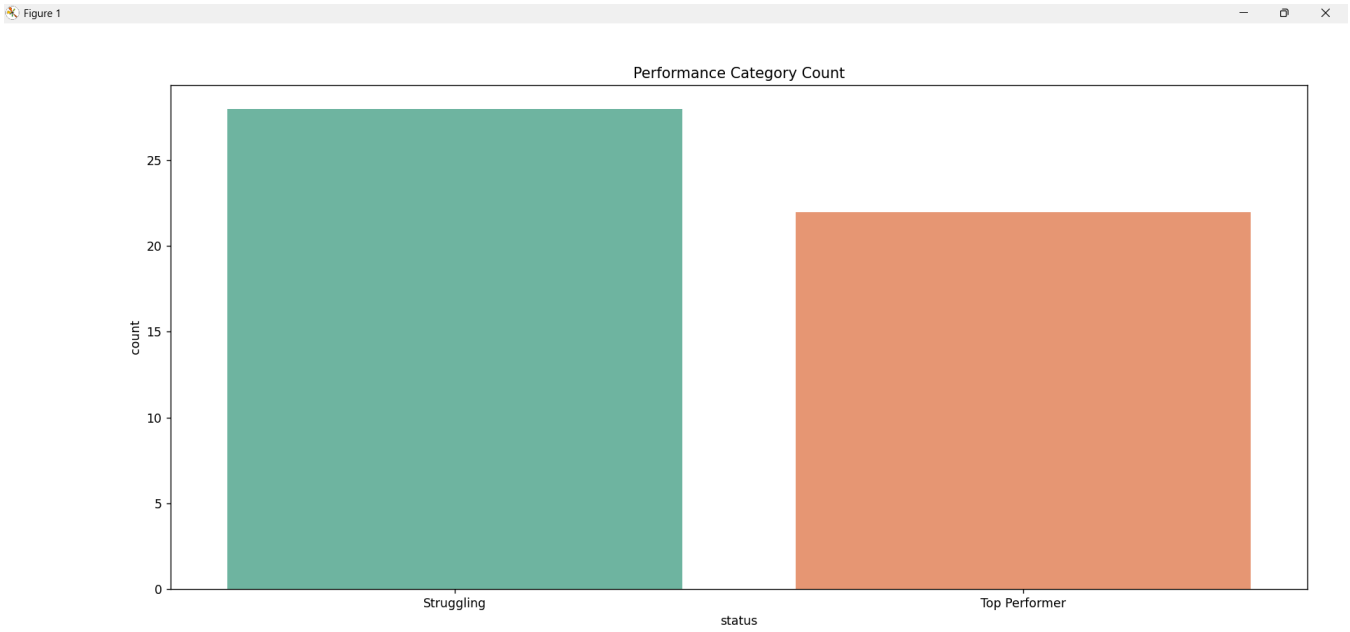


Figure 1
Performance Category Count
count
0
5
10
15
20
25
Struggling
Top Performer
status

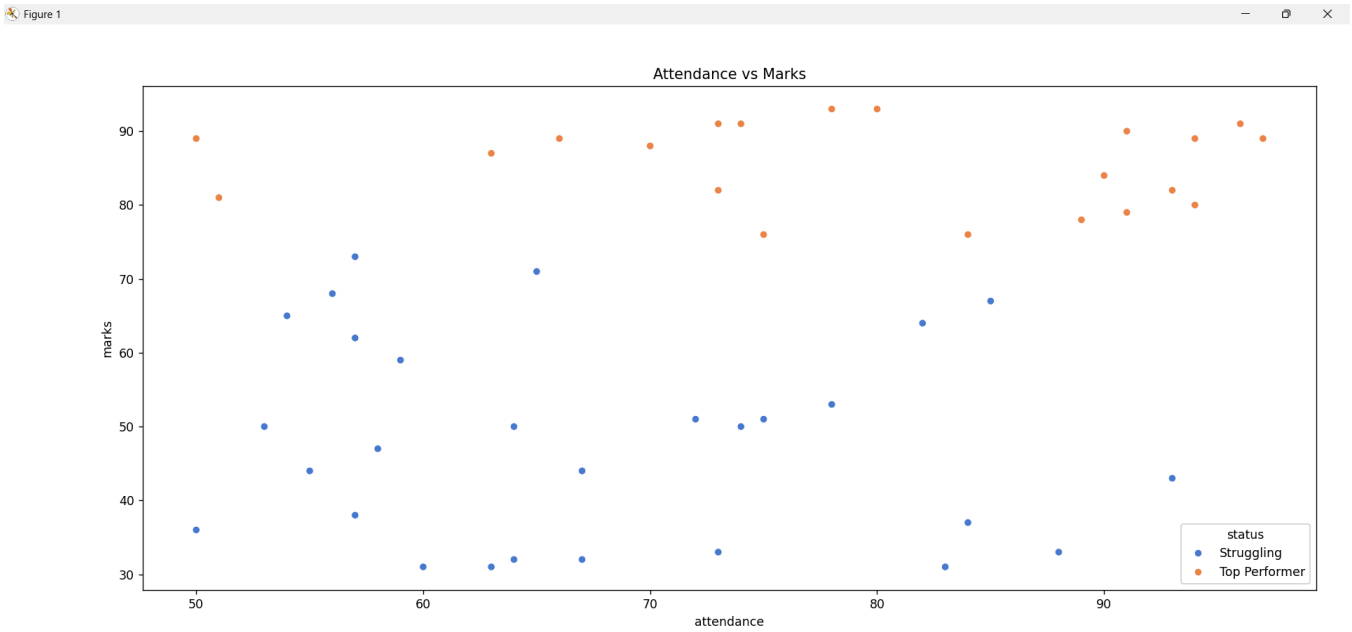


Figure 1
Attendance vs Marks
marks
30
40
50
60
70
80
90
50
60
70
80
90
attendance
status
Struggling
Top Performer

Project 3: Customer Segmentation Using K-Means

Problem Statement: Businesses need to target different customers differently based on behavior.

Objective: To segment customers into distinct groups based on their behavior and spending habits using machine learning techniques like K-Means clustering and Principal Component Analysis (PCA). This enables better targeting in marketing and personalized services.

➔ **Technologies & Libraries Used:**

- **Python** – Core language for scripting, analysis, and modeling.
- **Pandas** – For reading CSV files and manipulating structured customer data.
- **Matplotlib** – To generate custom 2D and 3D plots of clusters.
- **Seaborn** – For enhanced and more aesthetic plotting of PCA clusters.
- **Scikit-learn (sklearn) –**
  - **KMeans:** For unsupervised clustering of customers.
  - **StandardScaler:** For standardizing numerical values before clustering.
  - **PCA:** For dimensionality reduction and effective visualization.
- **mpl_toolkits.mplot3d** – To plot 3D scatterplots using PCA and income.

➔ **Problem-Solving Approach:**

- **Dataset Loading:**
  - Imported customer data from customer_data.csv.

- **Feature Selection & Cleaning:**
  - Selected four features: Age, Income, Frequency, Spending.
  - Removed missing values using .dropna().

- **Feature Scaling:**
  - Scaled the selected features using StandardScaler to normalize the data and make it suitable for clustering.

- **K-Means Clustering:**
  - Applied the KMeans algorithm with n_clusters=4 to group customers into four distinct segments.

- **Dimensionality Reduction using PCA:**

    - Reduced high-dimensional data to 2 components using PCA for visualization purposes.

- **Visualization:**
    - 2D Scatter Plot: Clustered customers shown using PCA1 and PCA2.
    - 3D Scatter Plot: Added Income to PCA for a three-dimensional view of segmentation.

- **Result Storage:**
    - Saved the segmented data with cluster labels to segmented_customers.csv.
    - Exported visualizations as PNG files:
        - customer_clusters_2D.png
        - customer_clusters_3D.png

➔ **Outcome:**

- Successfully segmented customers into 4 behavior-based clusters.
- Produced clear, easy-to-understand 2D and 3D cluster visualizations.
- Exported cluster-annotated dataset for further analysis or marketing use.

➔ **Files Delivered:**

- customer_data.csv – Raw customer dataset.
- customer.py – Main Python script for preprocessing, clustering, and visualization.
- customer_clusters_2D.png – 2D plot of clusters using PCA.
- customer_clusters_3D.png – 3D plot of clusters using PCA + Income.
- segmented_customers.csv – Final output with cluster labels.

➔ **Creativity and Innovation:**

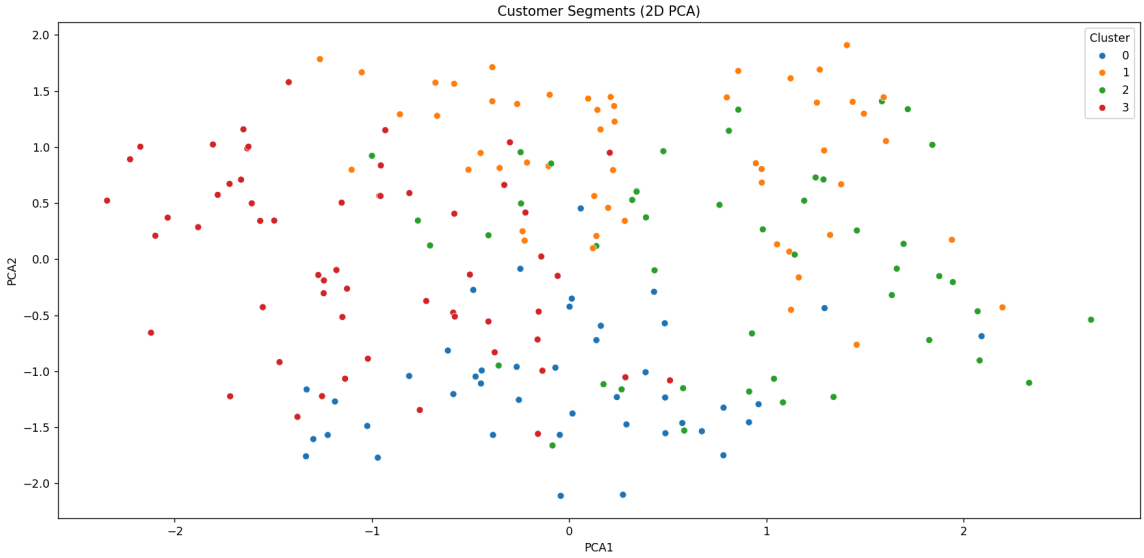
This project is not just a basic clustering — I used Principal Component Analysis (PCA) to reduce dimensions and enhance interpretability. The addition of 3D visualizations makes the clusters more insightful and demonstrates a deeper understanding of unsupervised learning.

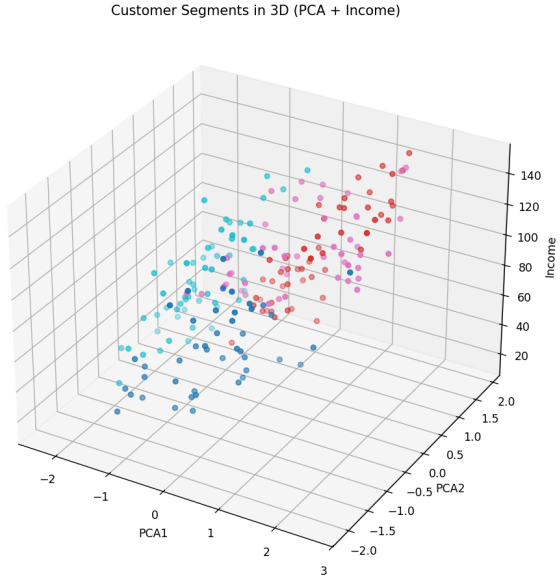
➔ **Documentation Quality:**

- Well-structured code and clear variable naming.
- Logical workflow from raw data to visual output.
- Output files named intuitively.
- Visuals saved and ready for presentations or reports.

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\customer.py"

✅ Customer segmentation complete. Output saved as 'segmented_customers.csv'.
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```





Project 4: COVID-19 Data Analysis

Problem Statement: Understanding COVID-19 trends helps in planning public health policies.

Objective: The goal of this project is to extract insights from global COVID-19 data by analyzing trends, comparing countries, and visualizing key statistics like confirmed cases, recoveries, and deaths. This helps in understanding how the pandemic evolved over time and its global impact.

COVID-19 affected millions globally, and with daily data updates, making sense of the large volume of information is difficult. My project simplifies this by:

- Cleaning and processing the data.
- Aggregating key metrics.
- Visualizing trends and country-wise comparisons to make the data more understandable and actionable.

➔ **Technologies & Libraries Used:**

- **Python:** Core programming language used to handle data and generate visualizations.
- **Pandas:** For data manipulation, grouping, sorting, filtering, and cleaning the CSV file.
- **Matplotlib:** For plotting line graphs and area charts that show trends over time.
- **Seaborn:** For generating an annotated heatmap summarizing top countries' data.

➔ **Problem-Solving Approach:**

- **Data Loading:**
  - Loaded COVID-19 daily case data from covid_data.csv.
  - Converted the Date column into datetime format for time series operations.
- **Data Cleaning:**
  - Removed rows with missing values using .dropna().
- Sorted data chronologically for accurate trend plots.
- **Global Aggregation:**
  - Grouped data by Date to generate a daily summary of global Confirmed, Recovered, and Deaths.

- **Visualization 1 – Line Chart:**
  - Plotted the trend of cumulative cases globally over time.
  - Used different line colors to distinguish between Confirmed, Recovered, and Deaths.

- **Top 5 Countries Analysis:**
  - Identified top 5 countries with the highest confirmed cases.
  - Generated an area chart showing how their cases grew over time.
  - Used fill_between for an appealing layered visualization.

- **Heatmap – Latest Snapshot:**
  - Extracted the most recent day's data.
  - Created a heatmap to display confirmed, recovered, and death cases for the top 10 countries on that day.

➔ **Outcome:**

- Created 3 meaningful visualizations:
  - Global line chart (global_covid_trends.png)
  - Area chart for top 5 countries (top_countries_area_chart.png)
  - Heatmap of current top 10 countries (covid_heatmap.png)
- Gained insights into how the pandemic affected countries over time.
- Built a reusable script for future COVID-19 data updates or country-specific analysis.

➔ **Files Delivered:**

- **covid_data.csv:** Raw input data file.
- **covid.py**: Python script for all preprocessing and plotting.
- **global_covid_trends.png:** Global cumulative trends over time.
- **top_countries_area_chart.png:** Area chart for top 5 countries.
- **covid_heatmap.png:** Heatmap of current top 10 affected countries.

➔ **Creativity & Innovation:**

- Combined line chart, area chart, and heatmap for multi-angle insights.
- Used heatmap on the latest day's data, adding real-time relevance.
- Focused on both global trends and country-wise breakdown, which balances macro and micro-level analysis.

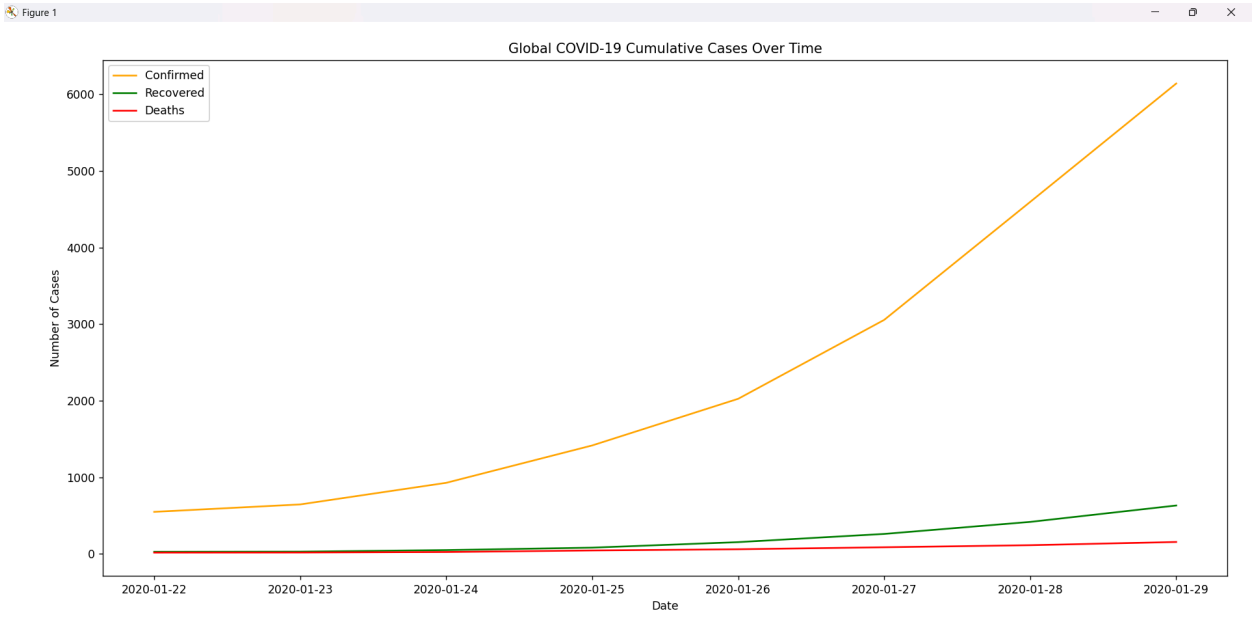
➔ **Documentation Quality:**

- **Modular structure:** loading, cleaning, visualization, and output.
- Plots are saved as PNGs for easy inclusion in reports or dashboards.

- File naming is intuitive and purpose-driven.

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\covid.py"
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```



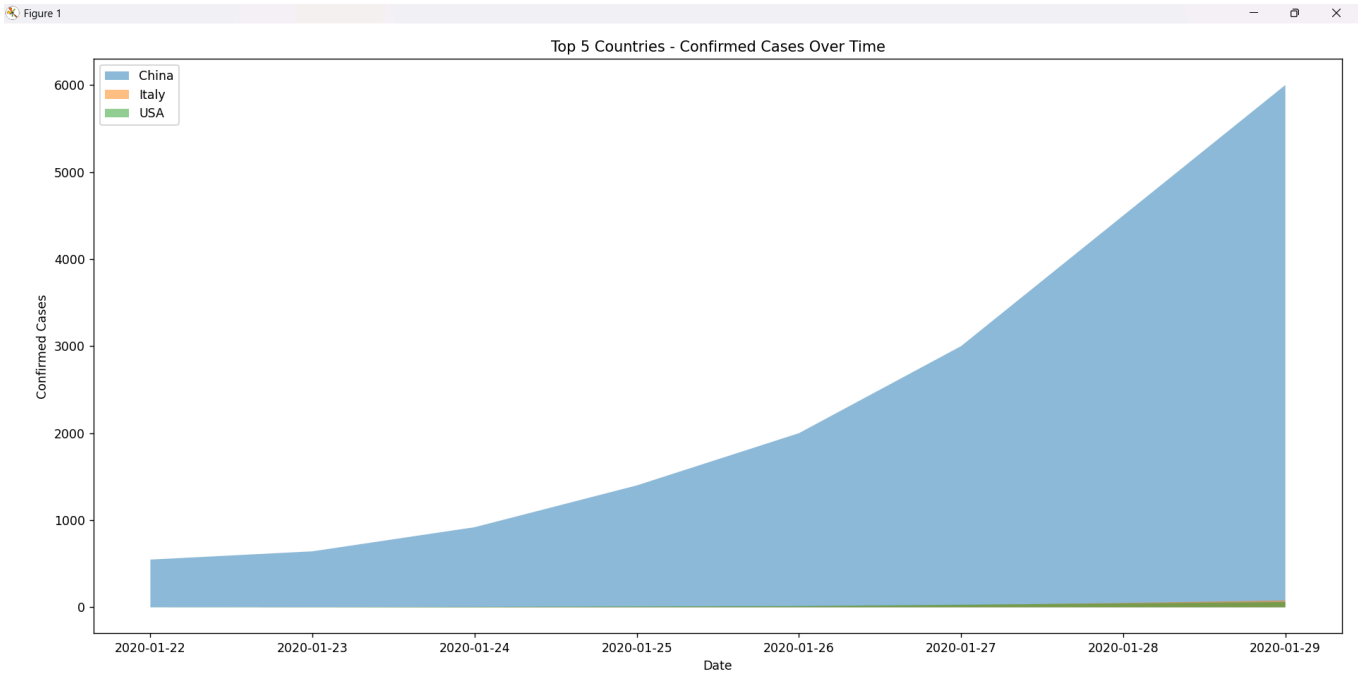
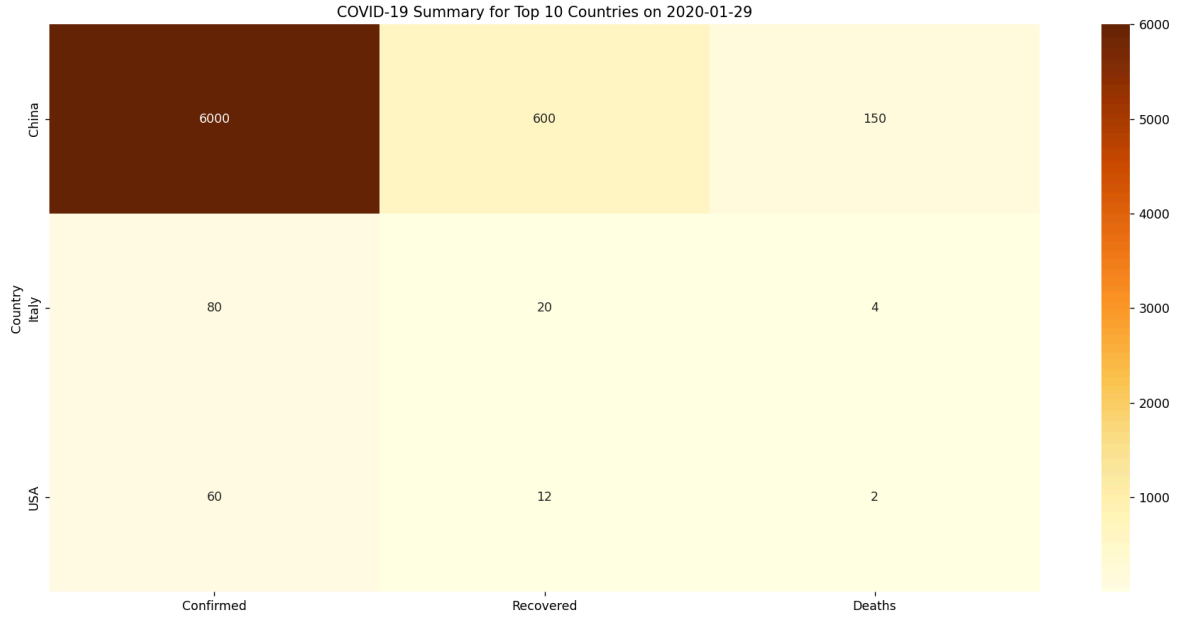


Figure 1
Top 5 Countries - Confirmed Cases Over Time
China
Italy
USA
Confirmed Cases
0
1000
2000
3000
4000
5000
6000
2020-01-22
2020-01-23
2020-01-24
2020-01-25
2020-01-26
2020-01-27
2020-01-28
2020-01-29
Date


COVID-19 Summary for Top 10 Countries on 2020-01-29
China
Italy
USA
Country
6000
600
150
80
20
4
60
12
2
Confirmed
Recovered
Deaths
6000
5000
4000
3000
2000
1000

Project 5: E-Commerce Data Insights

Problem Statement: E-commerce platforms need to know what products and users drive revenue.

Objective: This project analyzes customer behavior and product performance in an e-commerce dataset. By visualizing sales patterns, top users, and product ratings, the project uncovers actionable business insights such as peak shopping hours, popular products, and user engagement trends.

E-commerce platforms collect massive amounts of customer data daily. However, without analysis, this data is underutilized. My project focuses on extracting valuable information from transactional data by identifying:

- The top-selling products.
- The most active users.
- Highly rated products.
- Order frequency by time of day.

➔ **Technologies & Libraries Used:**

- Python: Core language used to develop, analyze, and visualize.
- Pandas: For data loading, cleaning, aggregation (grouping by product, user, etc.).
- Matplotlib: For generating pie and line charts.
- Seaborn: For bar and line charts with enhanced visual appeal.

➔ **Problem-Solving Approach:**

- **Data Ingestion:**
  - Loaded transactional data from ecommerce_data.csv.
  - Converted order_time to datetime format for time-based analysis.

- **Feature Engineering:**
  - Extracted order_hour from datetime to understand purchase timing patterns.

- **Insight Generation:**
  - Top Products: Aggregated total quantity sold per product and identified top 5.
  - Top Users: Aggregated quantity ordered per user and selected top 5.
  - Top Ratings: Calculated average review_score per product.

        - Hourly Orders: Counted number of orders placed in each hour of the day.

    - **Data Visualization:**
        - **Bar Chart:** Visualized top-selling products and their quantities.
        - **Pie Chart:** Represented contribution of top 5 users in total orders.
        - **Line Chart:** Showed customer order activity by hour.
        - **Bar Chart:** Ranked top 5 products by average review scores.

➔ **Outcomes:**

- Identified bestsellers and highly rated products, which are useful for inventory planning and promotions.
- Discovered peak shopping hours, valuable for optimizing server load, staffing, or flash sales.
- Found top customers contributing significantly to overall revenue.

➔ **Files Delivered:**

- **ecommerce_data.csv** – Raw dataset
- **ecommerce.py** – Python script with complete logic and plots
- **top_products.png** – Bar chart of most sold products
- **top_users.png** – Pie chart showing top customers
- **orders_by_hour.png** – Line plot for hourly order volume
- **top_rated_products.png** – Bar chart for top reviewed products

➔ **Creativity & Innovation:**

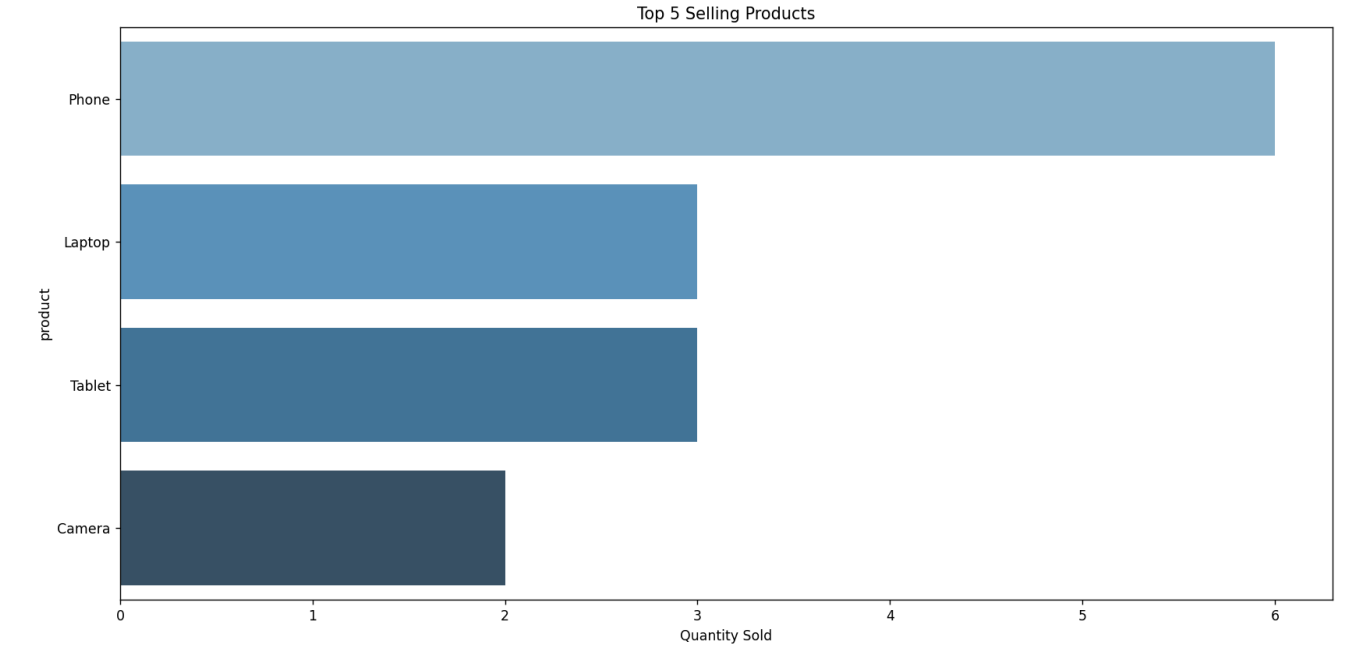
- Combined multiple KPIs like quantity, ratings, and user contributions in one visual dashboard.
- Used time-based segmentation (hourly) to find order behavior trends.
- The use of both bar, line, and pie charts makes the dashboard engaging and informative.

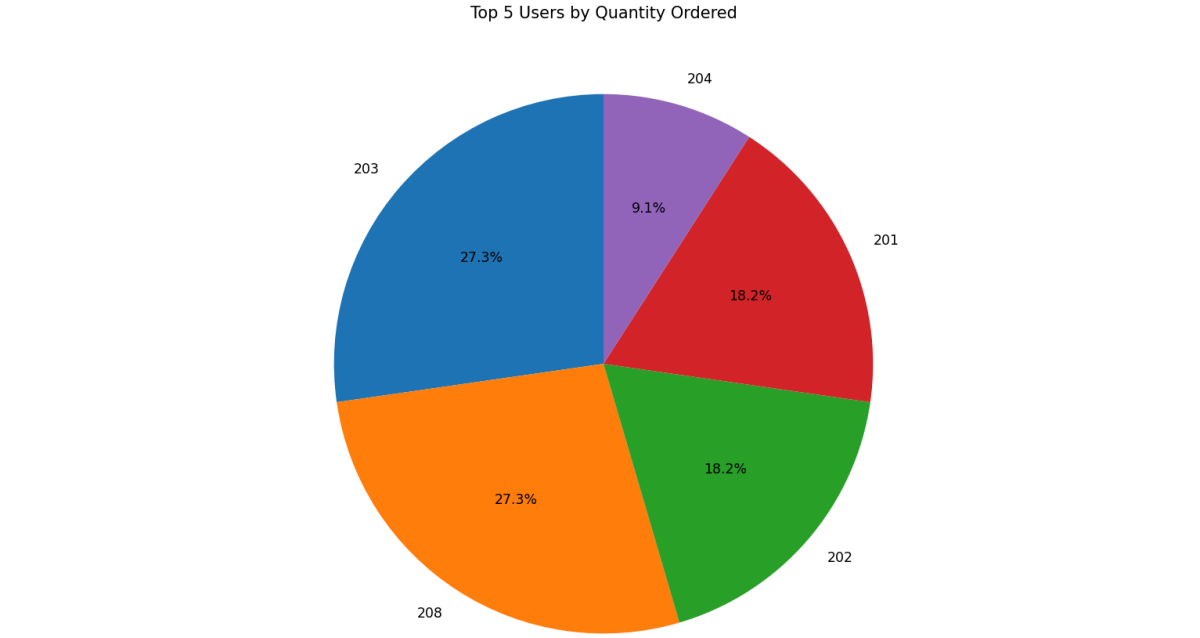
➔ **Documentation Quality:**

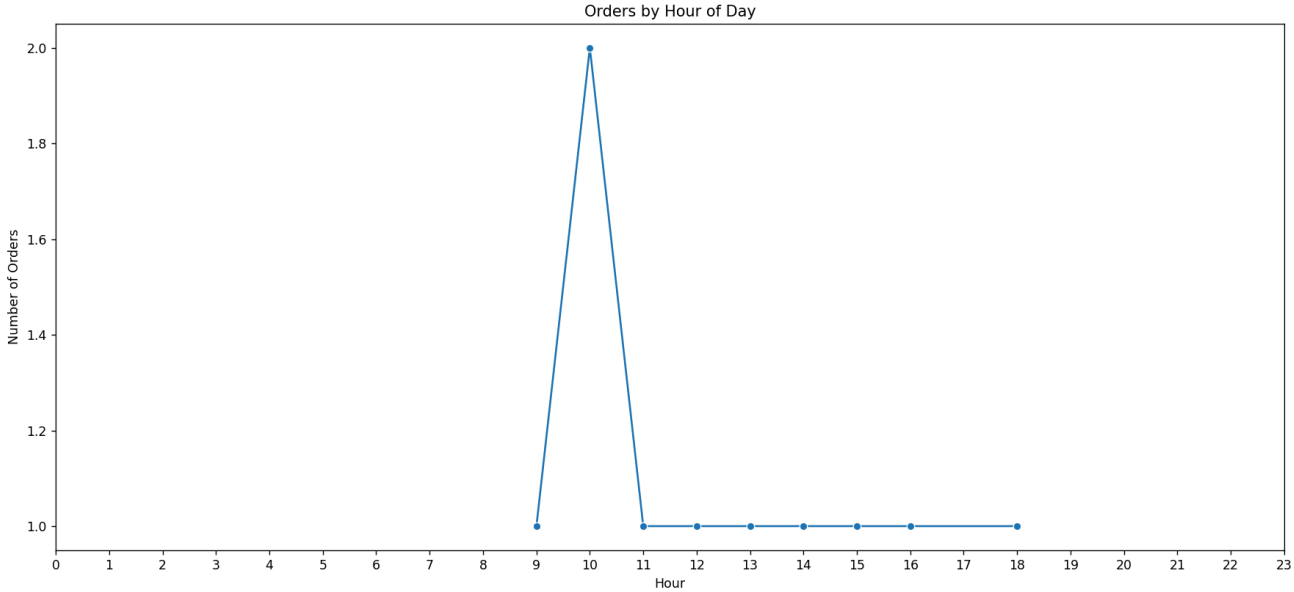
- Clear structure: loading → cleaning → insights → visual output.
- Plots are saved as PNGs for external sharing and inclusion in reports.
- Code is modular and understandable, with good variable naming and comments.

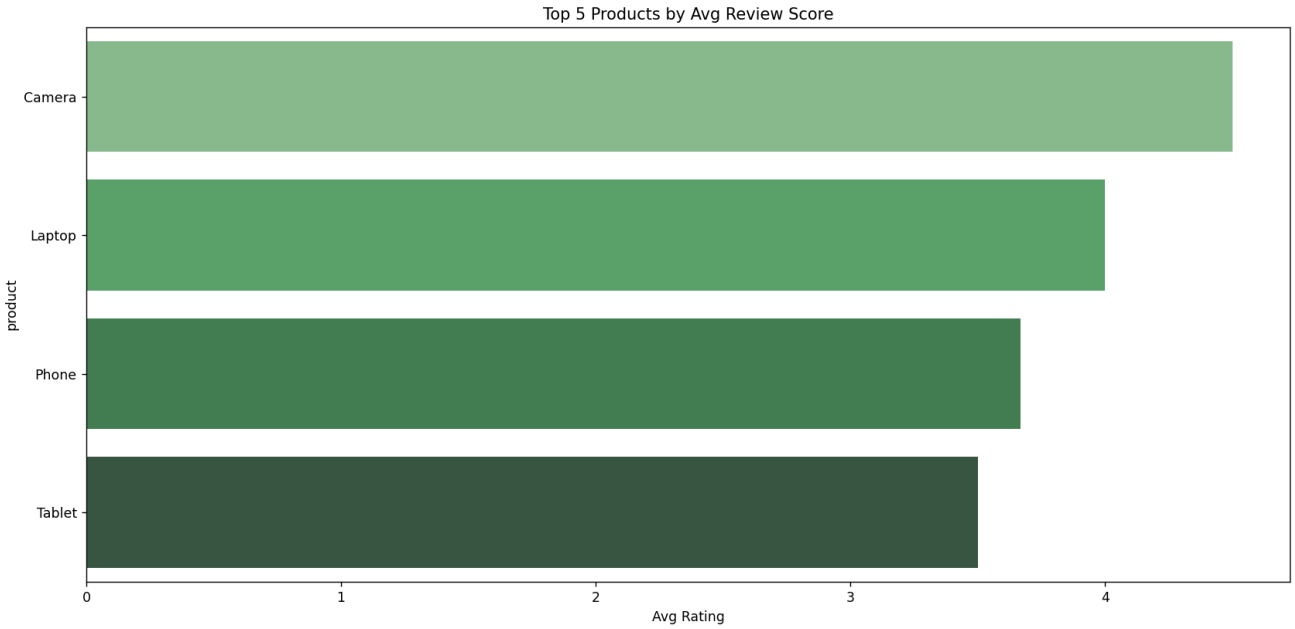
Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\ecommerce.py"

✅ E-commerce data insights generated and visualized.
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```




Top 5 Users by Quantity Ordered
204
203
9.1%
201
27.3%
18.2%
18.2%
27.3%
202
208


Orders by Hour of Day
Number of Orders
2.0
1.8
1.6
1.4
1.2
1.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour


Top 5 Products by Avg Review Score
Camera
Laptop
Phone
Tablet
product
0
1
2
3
4
Avg Rating

Project 6: Netflix User Behavior Analysis

Problem Statement: Streaming services want to understand what content users prefer.

Objective: The goal of this project is to analyze streaming behavior of Netflix users based on viewing habits, genres, ratings, and time trends. This project extracts meaningful patterns from viewing data to understand binge-watching behavior, popular genres, and daily engagement trends.

Streaming platforms accumulate massive user data, but understanding user engagement, top genres, and rating trends is critical for content planning and recommendation systems. This project aims to analyze:

- Total watch time by genre
- Top binge-watchers (users with highest watch time)
- Average ratings per genre
- Daily viewing trends

➔ **Technologies & Libraries Used:**

- **Python:** Primary language used for scripting and data analysis.
- **Pandas:** For data loading, preprocessing, grouping, and date parsing.
- **Matplotlib:** For generating daily trends and line plots.
- **Seaborn:** For creating appealing bar charts for genres, users, and ratings.

➔ **Problem-Solving Approach:**

- **Data Ingestion:**
  - Loaded dataset from netflix_data.csv.
  - Parsed date column correctly using dayfirst=True to fix inconsistent date formats.

- **Feature Engineering:**
  - Extracted total watch time by genre using groupby and sum.
  - Aggregated watch time per user to identify top 10 binge watchers.
  - Calculated average user ratings grouped by genre.
  - Computed daily total watch time to understand time-based user activity.

- **Insight Visualizations:**
  - Bar Chart – Total Watch Time by Genre.
  - Bar Chart – Top 10 Users by Watch Time.
  - Bar Chart – Average Ratings by Genre.
  - Line Chart – Daily Watch Time Trend.

➔ **Outcomes:**

- **Popular Genres:** Identified which genres dominate user watch time.
- **User Loyalty:** Top binge-watchers can be targeted for personalized content or early testing.
- **Quality Check:** Genres with the highest average rating reflect viewer satisfaction.
- **Activity Timeline:** Daily watch trends help with server load management and promotional planning.
-

➔ **Files Delivered:**

- **netflix_data.csv** – Input dataset
- **netflix.py** – Python script for analysis and plots
- **watch_time_by_genre.png** – Bar chart of watch time by genre
- **top_users_watchtime.png** – Top 10 binge watchers
- **avg_rating_by_genre.png** – Average ratings per genre
- **daily_watch_trend.png** – Line chart showing watch activity over time

➔ **Creativity & Innovation:**

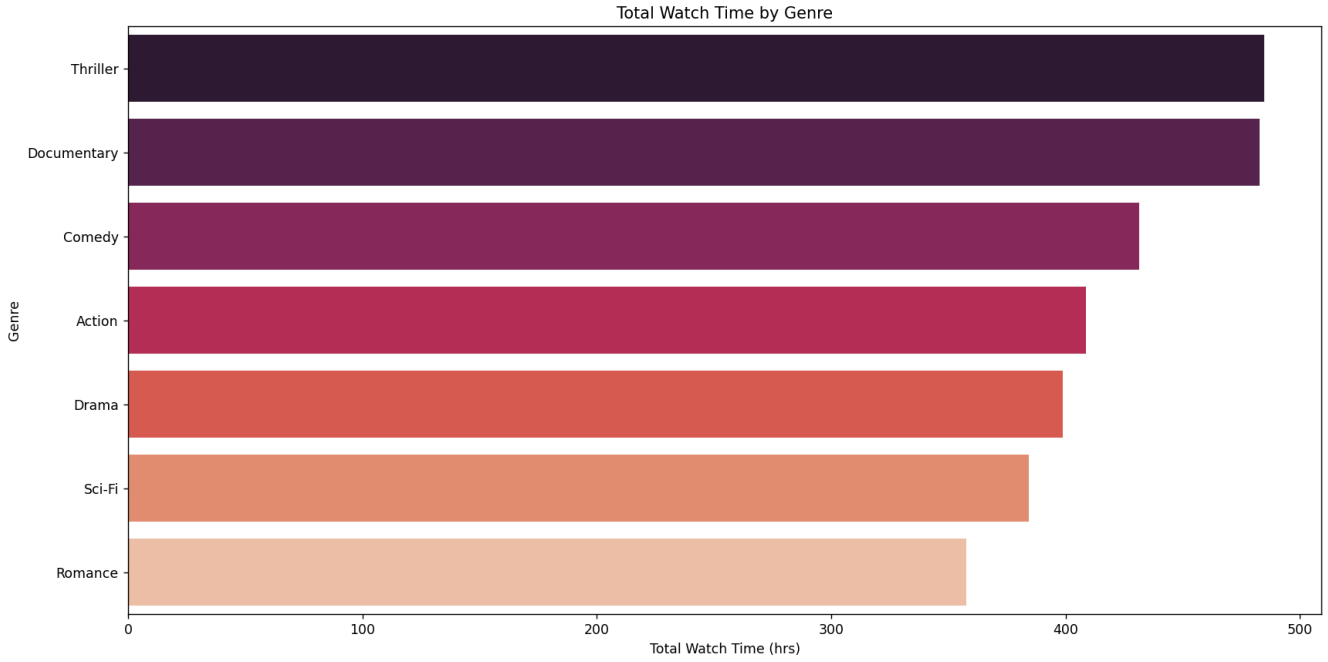
- Combined watch time, ratings, and temporal trends in a single dashboard.
- Cleaned and visualized large datasets using intuitive color palettes and layout.
- Time-series trend graph enhances the ability to forecast future engagement peaks.

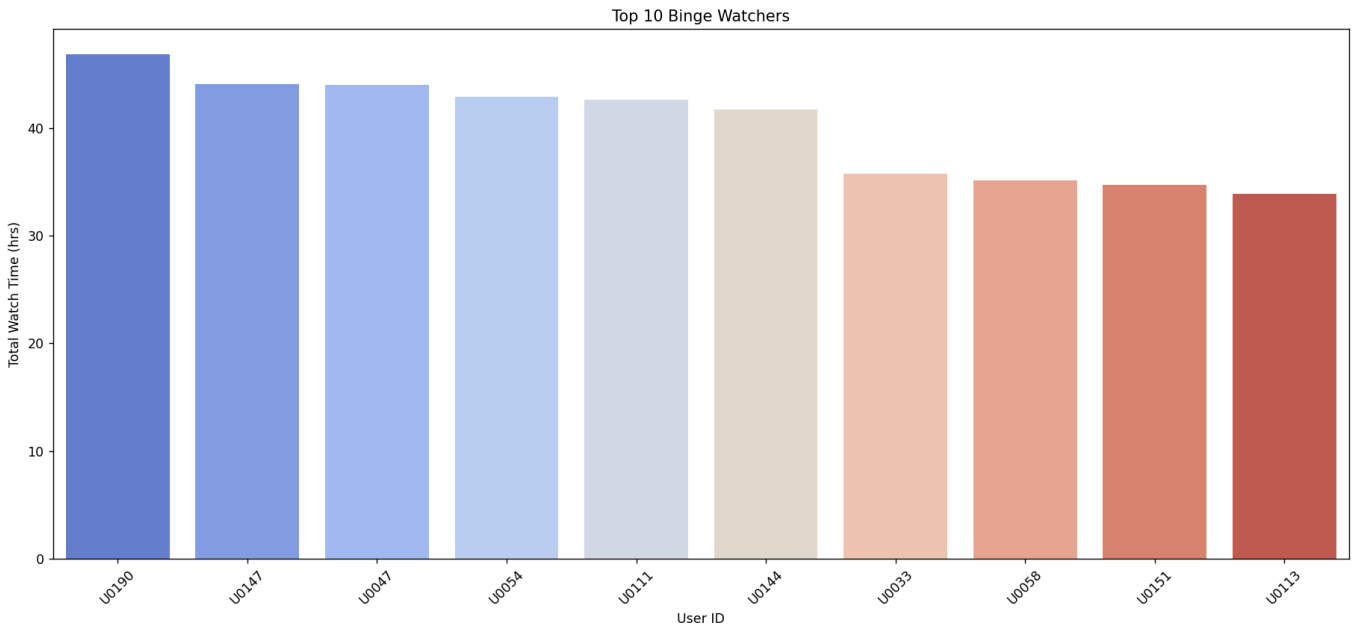
➔ **Documentation Quality:**

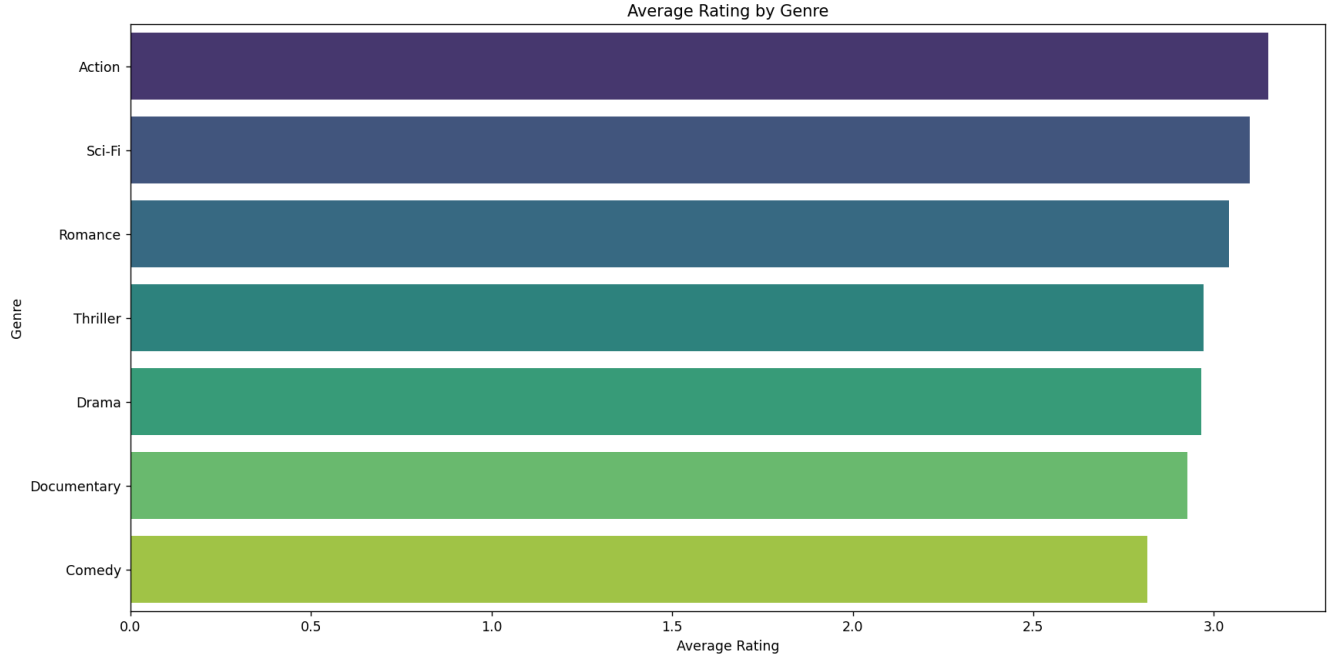
- Clear and structured code with comments.
- Visuals saved in .png format for reporting or web embedding.
- Insightful grouping and metric design that aligns with real-world streaming business KPIs.

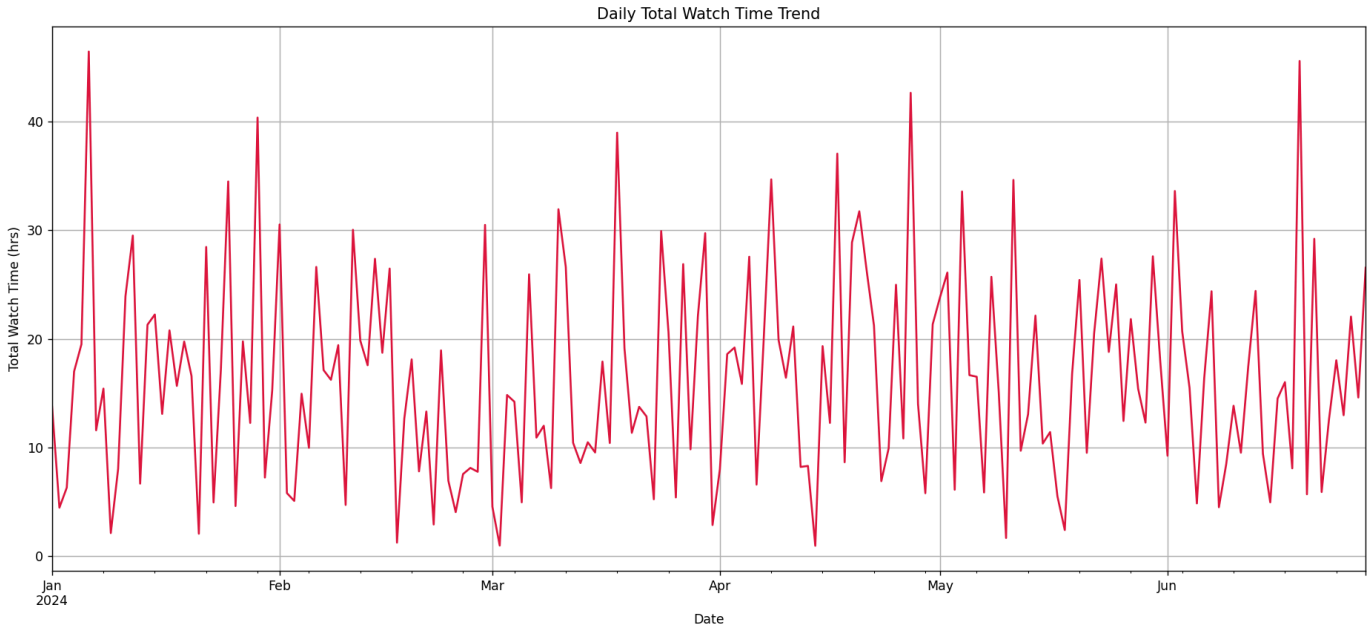
Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\netflix.py"
🔍 Dataset Preview:
  user_id  show_title       genre  watch_time  rating        date
0   U0103  Show_GJDSM  Documentary        1.60     1.3  10-01-2024
1   U0180  Show_LHAMR      Romance        5.27     2.5  07-03-2024
2   U0093  Show_GTUXL       Comedy        3.97     1.9  31-05-2024
3   U0015  Show_EFNZI        Drama        4.40     1.8  05-01-2024
4   U0107  Show_QGWOM       Comedy        0.03     1.6  27-01-2024

✅ Netflix user behavior insights generated and visualized.
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```




Top 10 Binge Watchers
Total Watch Time (hrs)
0
10
20
30
40
U0190
U0147
U0047
U0054
U0111
U0144
U0033
U0058
U0151
U0113
User ID


Average Rating by Genre
Genre
Action
Sci-Fi
Romance
Thriller
Drama
Documentary
Comedy
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Average Rating


Daily Total Watch Time Trend
Total Watch Time (hrs)
0
10
20
30
40
Jan
2024
Feb
Mar
Apr
May
Jun
Date

Project 7: Traffic Pattern Analysis

Problem Statement: Cities face congestion due to poor traffic pattern understanding.

Objective: To simulate and analyze hourly vehicle traffic data for one month, visualize temporal traffic trends, and detect abnormal traffic spikes or drops using statistical techniques.

Traffic data plays a crucial role in smart city planning, congestion control, and emergency services. This project generates synthetic hourly traffic volume data and identifies anomalies (sudden increases or decreases) using statistical outlier detection.

➔ **Technologies & Libraries Used:**

- **Python:** Main language for data generation, analysis, and visualization.
- **NumPy:** For generating realistic traffic patterns with added randomness (noise).
- **Pandas:** For handling time-series data, creating timestamps, and computing Z-scores.
- **Matplotlib:** For plotting traffic volume across time.
- **Seaborn:** For enhanced data visualization aesthetics (imported, but optional).
- **SciPy (stats):** For applying Z-score–based anomaly detection on continuous numeric data.

➔ **Methodology:**

- **Data Generation:**
  - Simulated 30 days (June 2024) of hourly traffic data using periodic sine wave patterns to reflect peak hours (morning and evening rush).
  - Introduced random noise to mimic natural fluctuations.

- **Dataset Creation:**
  - Created a DataFrame with two columns:
  - timestamp: hourly time intervals from June 1 to June 30.
  - vehicle_count: number of vehicles observed, adjusted to be non-negative.

- **Visualization:**
  - Plotted a line graph to show trends in traffic over time, identifying peak and low traffic periods visually.

- **Anomaly Detection:**
    - Used Z-score method to identify statistical anomalies (Z > 3 indicates an unusual spike/drop).
    - Highlighted these anomalies for further investigation or alert systems.

➔ **Files Delivered:**

- **traffic_data.csv** – Generated traffic data file (hourly for 30 days)
- **traffic.py** – Python script to generate data, visualize it, and detect anomalies
- **traffic_volume_trend.png** – Optional figure if saved from the plot

➔ **Output Summary:**

- **Hourly Data Points:** 720 (24 hours × 30 days)
- **Anomalies Detected:** Based on Z-score > 3
- **Key Insight:** The traffic pattern successfully replicates real-world conditions — lower traffic at night, peaks around morning and evening rush hours.

➔ **Code Snapshot (Core Logic):**

```
# Generate timestamps and traffic pattern
rng = pd.date_range(start='2024-06-01', end='2024-06-30 23:00:00', freq='H')
base = 100 + 50 * np.sin(2 * np.pi * rng.hour / 24) + 80 * np.sin(2 * np.pi *
(rng.hour-17) / 24)
noise = np.random.normal(0, 15, len(rng))
vehicle_count = np.maximum(0, base + noise).astype(int)

# Create DataFrame
traffic_df = pd.DataFrame({'timestamp': rng, 'vehicle_count': vehicle_count})
traffic_df.to_csv('traffic_data.csv', index=False)

# Anomaly Detection
traffic_df['zscore'] = np.abs(stats.zscore(traffic_df['vehicle_count']))
anomalies = traffic_df[traffic_df['zscore'] > 3]
```

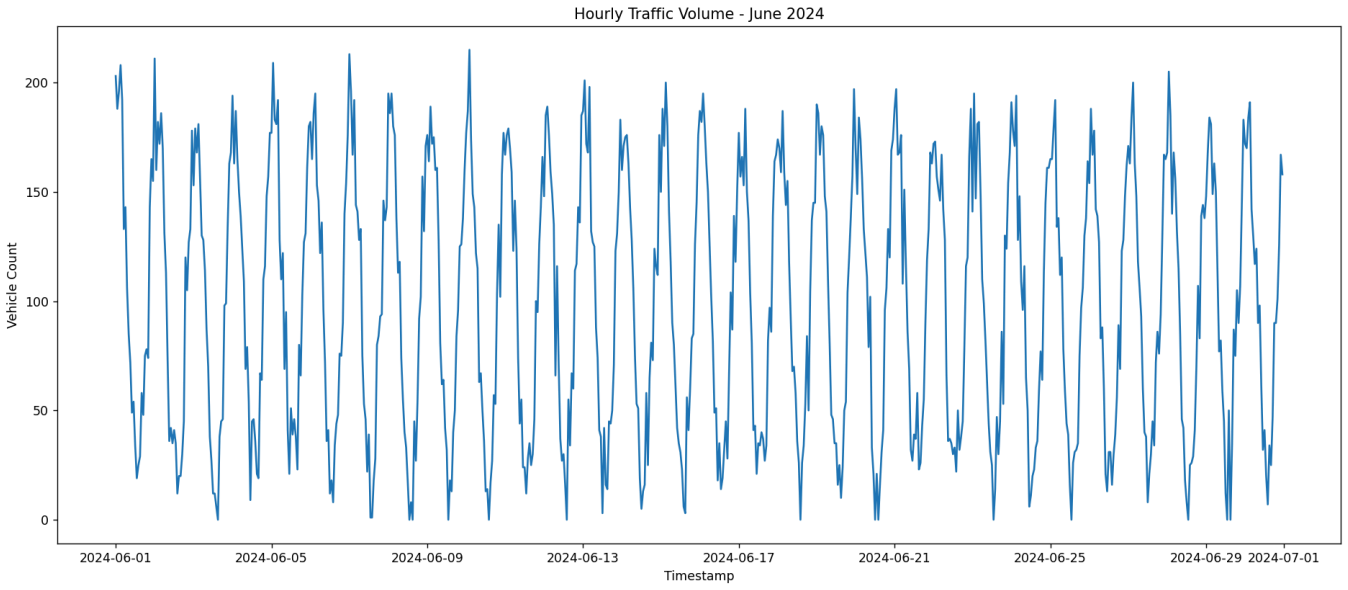
➔ **Innovation & Relevance:**

- Demonstrates how synthetic data can be modeled for real-world problems.

- Anomaly detection using Z-score offers a statistical foundation that can scale with live traffic sensors or IoT devices.
- Applicable in traffic control systems, smart signal management, and emergency response triggers.

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\traffic.py"
c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\traffic.py:8: FutureWarning: 'H' is deprecated and will be removed in a future version, please use 'h' instead.
  rng = pd.date_range(start='2024-06-01', end='2024-06-30 23:00:00', freq='H')
            timestamp  vehicle_count    zscore
0 2024-06-01 00:00:00            203  1.742427
1 2024-06-01 01:00:00            188  1.491708
2 2024-06-01 02:00:00            196  1.625424
3 2024-06-01 03:00:00            208  1.825999
4 2024-06-01 04:00:00            192  1.558566
Number of detected anomalies: 0
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```



Project 8: Survey Data Visualization Project

Problem Statement: Survey results are often under-utilized due to poor analysis.

Objective: To analyze structured customer survey responses, visualize satisfaction and preferences, and derive meaningful insights to assist product or service improvements based on user opinions.

➔ **Dataset:**

- **Filename:** survey_data.csv
- Each row represents an individual response to a customer satisfaction survey.
- The key columns (standardized to lowercase and stripped of whitespace) include:
  - **would_recommend:** Whether the customer would recommend the service (Yes/No).
  - **age_group:** Categorical age segments of respondents.
  - **overall_satisfaction:** Satisfaction rating (typically on a 1–5 scale).
  - **favorite_product:** Product most preferred by the respondent.
  - **feature_importance_ui, feature_importance_price, feature_importance_performance:** 1–5 scale ratings of feature importance.

➔ **Tech Stack:**

- **Python** – for data analysis and scripting.
- **Pandas** – for dataset loading and preprocessing.
- **Matplotlib & Seaborn** – for visualization and plotting.

➔ **Analysis & Visualizations:**

- **1. Pie Chart: Recommendation Likelihood**
  - **Column Used:** would_recommend
  - **Insight:** Shows the proportion of users willing to recommend the product/service.

```
recommend_counts = survey['would_recommend'].value_counts()
plt.pie(recommend_counts.values, labels=recommend_counts.index)
```

  - **Saved as:** recommend_pie.png

- **2. Heatmap: Average Satisfaction by Age Group**
  - **Columns Used:** age_group, overall_satisfaction
  - **Insight:** Helps identify which age groups are most/least satisfied.

- **Technique:** Used a pivot table and sns.heatmap() for visualizing mean satisfaction.
- **Saved as:** satisfaction_age_heat.png

- **3. Bar Chart: Favorite Product Count**
  - **Column Used:** favorite_product
  - **Insight:** Shows product popularity among customers.

product_counts = survey['favorite_product'].value_counts()
sns.barplot(x=product_counts.index, y=product_counts.values)

  - **Saved as:** favorite_product_bar.png

- **4. Bar Chart: Average Feature Importance**
  - **Columns Used:** feature_importance_ui, feature_importance_price, feature_importance_performance
  - **Insight:** Which features matter most to customers.
  - **Rating Scale:** 1 (least important) to 5 (most important)

avg_importance = survey[features].mean()

  - **Saved as:** feature_importance_bar.png

➔ **Data Cleaning & Handling:**

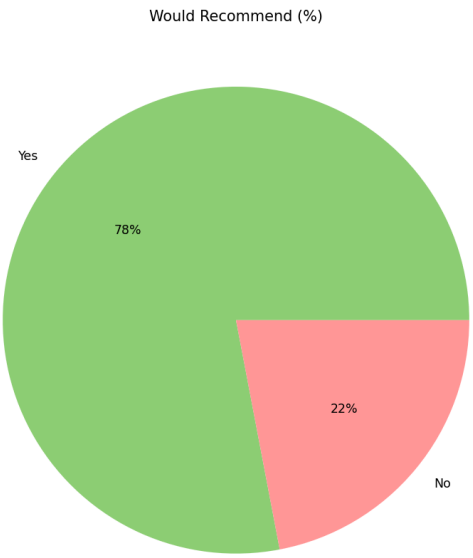
- All column names were stripped and lowercased for consistency.
- Handled missing column scenarios using conditional checks to prevent runtime errors.
- Used tight_layout() and appropriate color palettes for clarity in all visuals.

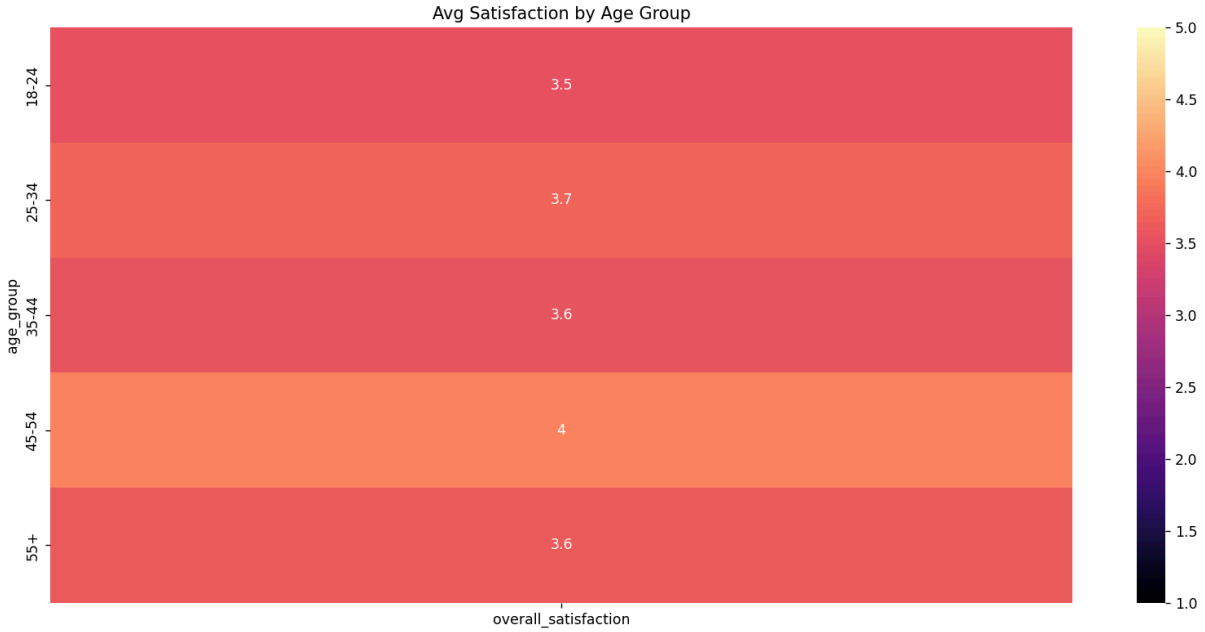
➔ **Final Output Summary:**

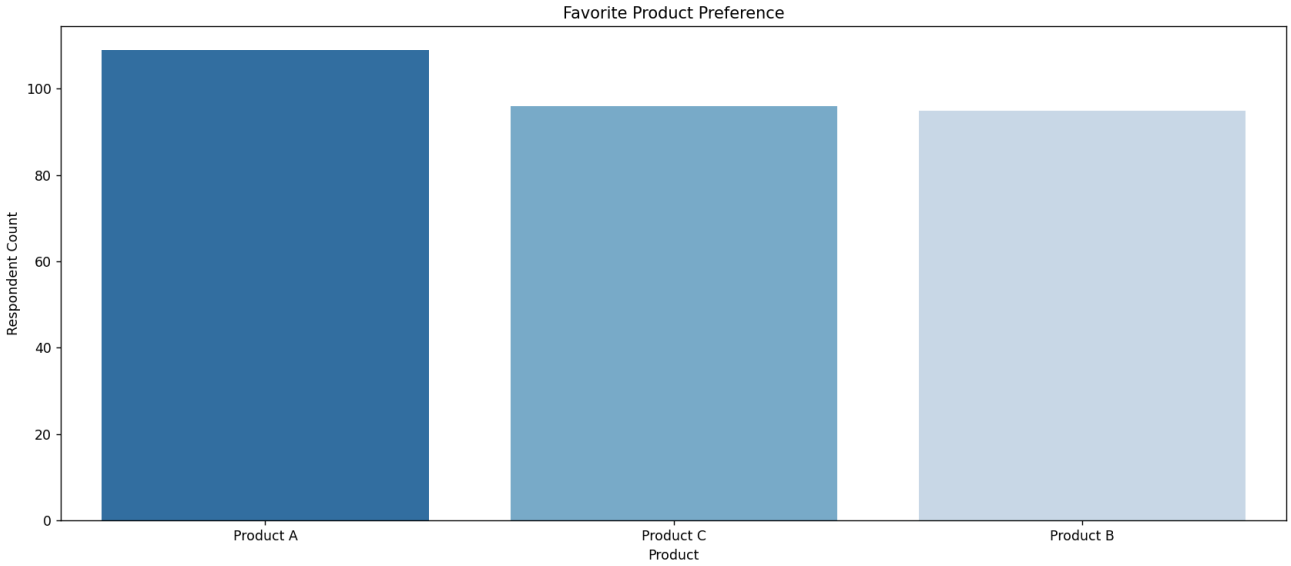
- Visuals generated: 4
- Conditional logic ensured robustness if any columns were missing.
- Insights can directly inform business decisions on feature development, user satisfaction improvement, and marketing focus.

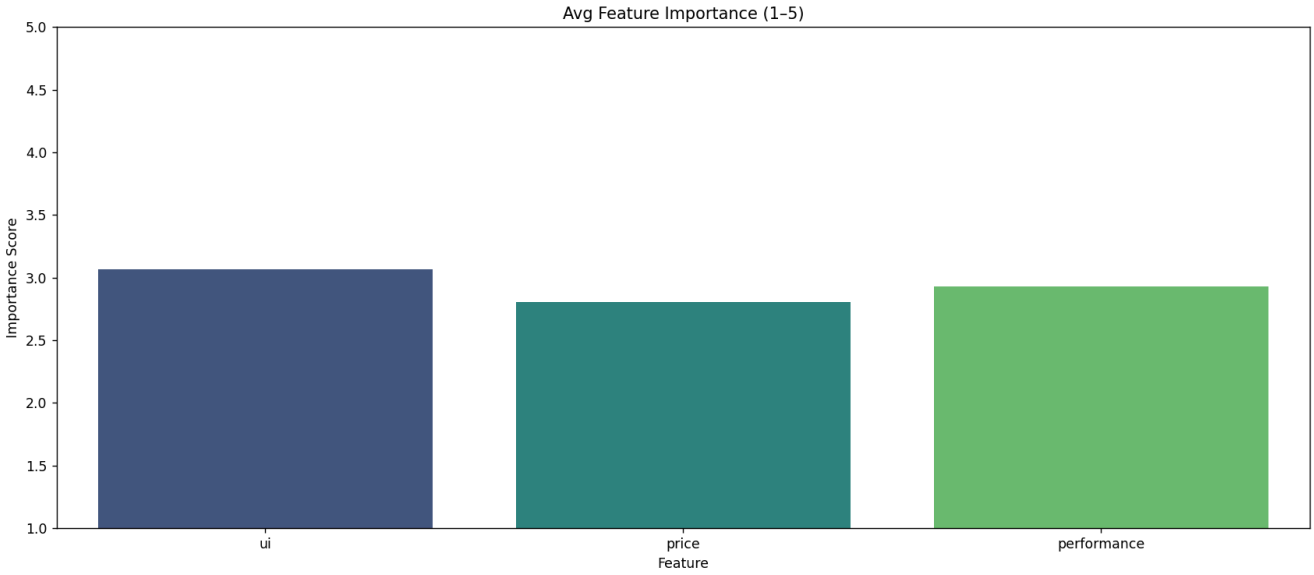
Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP> python -u "c:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP\survey.py"
Available columns: ['respondent_id', 'age_group', 'favorite_product', 'overall_satisfaction', 'would_recommend', 'feature_importance_ui', 'feature_importance_price', 'feature_importance_performance']
Average Satisfaction by Age Group:
           overall_satisfaction
age_group
18-24                  3.537313
25-34                  3.712871
35-44                  3.552239
45-54                  3.974359
55+                    3.615385
PS C:\Users\Inchara\OneDrive\Desktop\RISE DSA INTERNSHIP>
```




Avg Satisfaction by Age Group
age_group
18-24
25-34
35-44
45-54
55+
3.5
3.7
3.6
4
3.6
overall_satisfaction
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0


Favorite Product Preference
Respondent Count
0
20
40
60
80
100
Product A
Product C
Product B
Product


Avg Feature Importance (1–5)
Importance Score
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
ui
price
performance
Feature