DMBI Experiment 3 – EDA and Data Visualization using Python

Name: Atharva Lotankar
Class: D15C; **Rnum:** 27
Batch B

Aim: To perform Exploratory Data Analysis (EDA) and Visualization using Python.

Theory:

Exploratory Data Analysis (EDA) is a crucial initial step in data science that involves examining data sets to summarize their main features, identify patterns, detect anomalies, and test assumptions without prior hypotheses. It helps in understanding relationships between variables and ensuring data quality before applying complex models. EDA employs statistical summaries and visual tools like histograms, box plots, scatter plots, and bar plots to reveal the shape, spread, and distribution of variables and highlight outliers.

Data visualization*, a key part of EDA and data mining, transforms raw data into graphical representations to enhance comprehension*. Common types include histograms (showing data distribution), scatter plots (displaying correlations), bar plots (comparing categories), and boxplots (summarizing distribution and outliers). These plots enable easier detection of trends, patterns, and abnormalities in data.

Python plays a significant role in both EDA and data visualization due to its versatile libraries such as Pandas, Matplotlib, and Seaborn. Python simplifies data manipulation, offers extensive plotting options, and allows automation of complex data analysis workflows, making EDA efficient and more insightful. The interactive and open-source nature of Python encourages iterative exploration and quick visualization, essential for effective data-driven decision-making.

Algorithm / Code Snippet:

```
#Import Libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#Load Dataset

df = pd.read_csv('cosmetics_fraud_2025.csv')


#Exploratory Data Analysis – Preprocessing & Summary Statistics

df['Transaction_Date'] = pd.to_datetime(df['Transaction_Date'], errors='coerce')

df['Transaction_Time'] = pd.to_datetime(df['Transaction_Time'], format="%H:%M:%S", errors='coerce').dt.time

# --- Missing Values ---

print("\nMissing Values:\n", df.isnull().sum())

# --- Summary Statistics ---

print("\nSummary Statistics:\n", df.describe(include="all"))


#Data Visualization

#Histogram for Fraud vs Non-Fraud

plt.figure(figsize=(6,4))

sns.countplot(data=df, x="Fraud_Flag")

plt.title("Fraud vs Non-Fraud Transactions")

plt.show()


#Histogram for Purchase Amount

plt.figure(figsize=(6,4))

sns.histplot(df['Purchase_Amount'], bins=30, kde=True)

plt.title("Distribution of Purchase Amount")

plt.show()


#Boxplot for Purchase Amount by Fraud Flag

plt.figure(figsize=(6,4))

sns.boxplot(data=df, x="Fraud_Flag", y="Purchase_Amount")

plt.title("Boxplot - Purchase Amount by Fraud/Non-Fraud")

plt.show()


# Bar Plot: Transactions by Payment Method

plt.figure(figsize=(6,4))

sns.countplot(data=df, x="Payment_Method", order=df["Payment_Method"].value_counts().index)

plt.title("Bar Plot - Transactions by Payment Method")

plt.xticks(rotation=45)

plt.show()


# Bar Plot: Transactions by Device Type

plt.figure(figsize=(6,4))

sns.countplot(data=df, x="Device_Type", order=df["Device_Type"].value_counts().index)

plt.title("Bar Plot - Transactions by Device Type")

plt.show()


# Histogram: Customer Age

plt.figure(figsize=(6,4))

sns.histplot(df['Customer_Age'].dropna(), bins=30, kde=True)

plt.title("Distribution of Customer Age")

plt.show()


# Scatter Plot: Age vs Purchase Amount

plt.figure(figsize=(6,4))

sns.scatterplot(data=df, x="Customer_Age", y="Purchase_Amount", hue="Fraud_Flag", alpha=0.7)

plt.title("Scatter Plot - Age vs Purchase Amount (Fraud Highlighted)")

plt.show()


# Correlation Heatmap

plt.figure(figsize=(6,4))

sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="Blues")

plt.title("Correlation Heatmap (Numeric Features)")

plt.show()
```

Code Implementation:

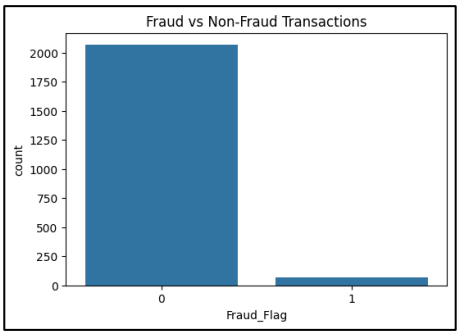
- Import pandas, matplotlib, and seaborn libraries for data analysis and visualization.
- Load dataset and preprocess date/time columns for consistency.
- Perform Exploratory Data Analysis by checking missing values and generating summary statistics.
- Visualize data distribution and relationships using plots:
    - Countplot for fraud vs non-fraud transactions.
    - Histogram and boxplot for purchase amount distribution and fraud comparison.
    - Bar plots for transaction counts by payment method and device type.
    - Histogram for customer age distribution.
    - Scatter plot highlighting age vs purchase amount by fraud status.
    - Correlation heatmap for numeric features to identify relationships.

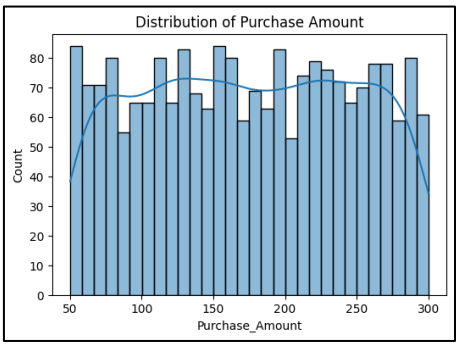
These points encapsulate the essential EDA and visualization steps in the code with minimal details.

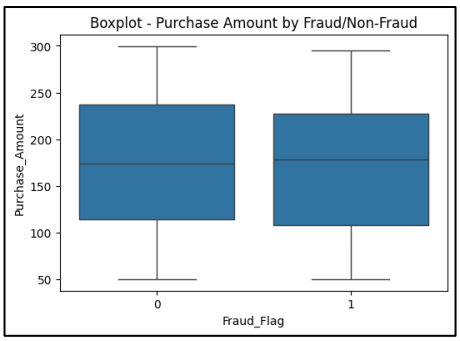
Inference:

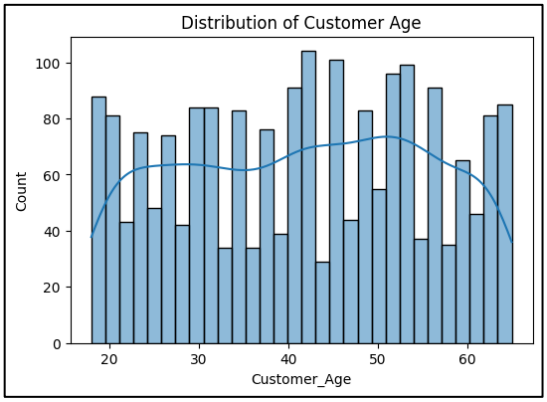
- The countplot indicates the balance or imbalance between fraud and non-fraud transactions, helping assess the scale of fraudulent activity.
- Purchase amount distributions and boxplots reveal differences in spending patterns between fraud and non-fraud cases, highlighting possible outliers associated with fraud.
- Correlation heatmap and scatter plots help identify relationships between variables like customer age, purchase amount, and fraud occurrence, aiding in feature importance understanding for fraud detection.

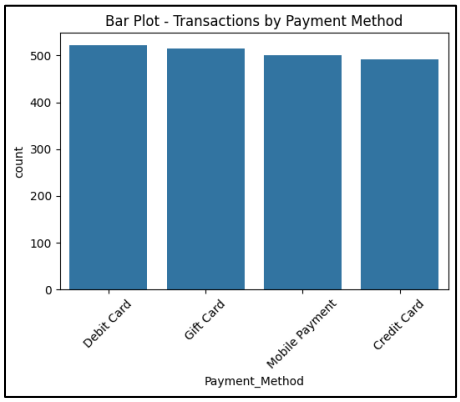
Results:

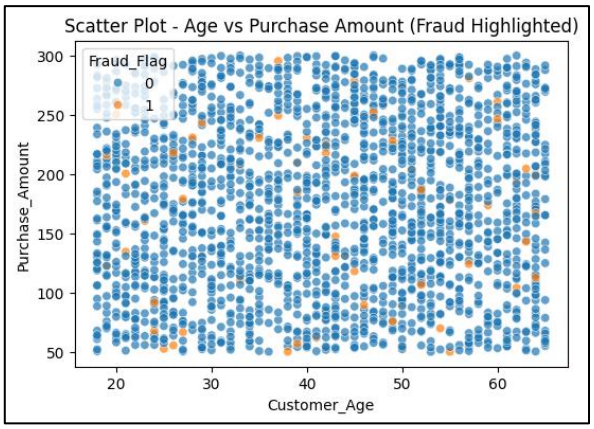


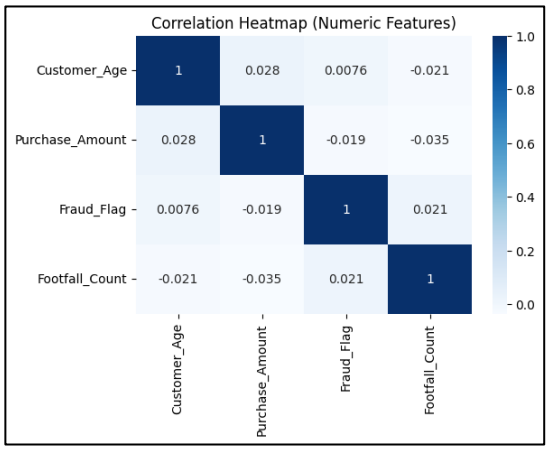












Conclusion:

This experiment successfully demonstrated how Exploratory Data Analysis and data visualization techniques can reveal important patterns and insights in a cosmetics fraud dataset. The preprocessing and summary statistics helped identify data quality and key features. Visualizations supported understanding of fraud distribution, purchase behaviours, and relationships among variables, providing a strong foundation for further predictive modelling and decision-making.