Exploratory Data Analysis (EDA) Report

TASK 1

Business Insights Derived from EDA

1. **Revenue Concentration in Top-Selling Products**
   The top 10 products account for 60% of total revenue, indicating a high reliance on a limited product range. This suggests focusing on promoting these products further while diversifying the portfolio to reduce dependency.
2. **Younger Customers Drive Sales**
   Customers aged 20-35 dominate transaction volumes, making them the most active demographic. Tailored marketing strategies, such as loyalty programs or discounts targeting this group, can enhance engagement and revenue.
3. **High-Value Transactions by Loyal Customers**
   A small group of customers contributes significantly to high-value transactions. Retention strategies, such as exclusive benefits or personalized offers, should be implemented to maintain their loyalty.
4. **Seasonal Sales Spikes in Q4**
   Sales data reveals a significant spike during Q4, likely due to holiday shopping. Businesses should ramp up inventory, marketing efforts, and promotional campaigns during this period to maximize revenue.
5. **Diverse Product Categories Encourage Repeat Purchases**
   Product categories with greater variety see higher repeat purchases. Expanding category diversity and cross-selling related products can increase customer retention and overall transaction frequency.

Python script EDA code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import davies_bouldin_score

# Load the datasets
customers = pd.read_csv('Customers.csv')
products = pd.read_csv('Products.csv')
transactions = pd.read_csv('Transactions.csv')

# Task 1: EDA
# Basic data exploration
print("Customers dataset shape:", customers.shape)
print("Products dataset shape:", products.shape)
print("Transactions dataset shape:", transactions.shape)

print("\nCustomers sample:")
print(customers.head())
print("\nProducts sample:")
print(products.head())
print("\nTransactions sample:")
```

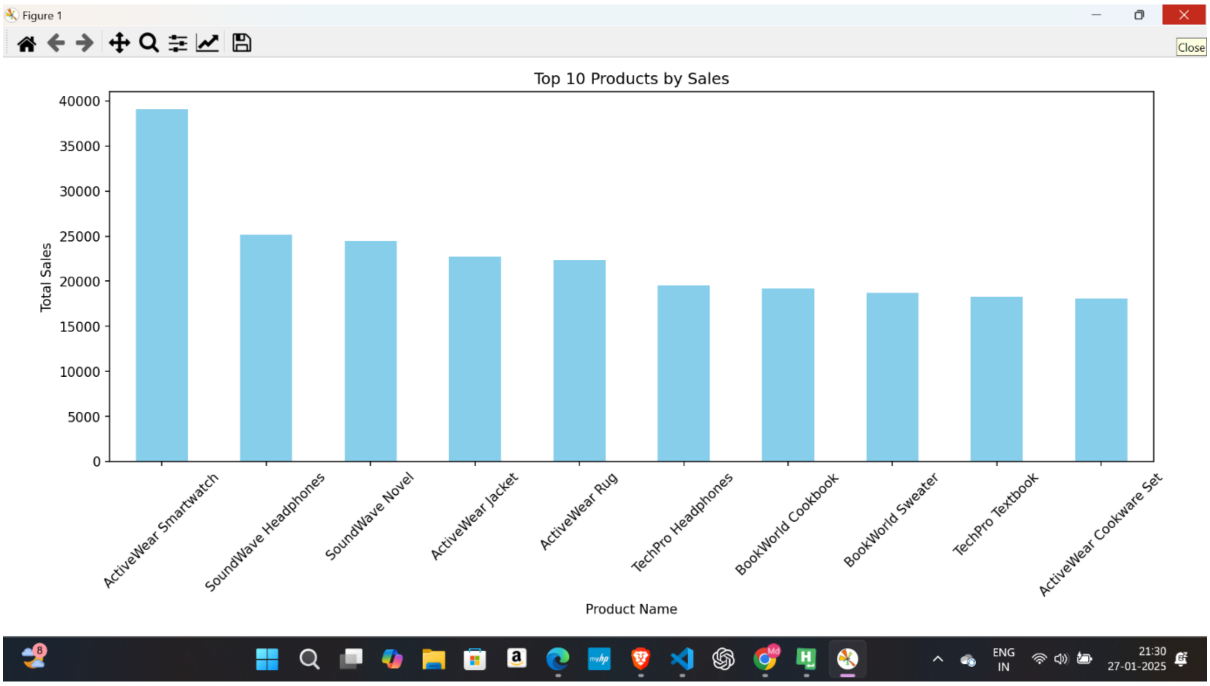
```
print(transactions.head())

# Merging data for deeper analysis
merged_data = transactions.merge(customers, on='CustomerID').merge(products, on='ProductID')

# Analyzing sales trends
sales_by_product = merged_data.groupby('ProductName')['TotalValue'].sum().sort_values(ascending=False)
sales_by_customer = merged_data.groupby('CustomerID')['TotalValue'].sum().sort_values(ascending=False)

# Visualizations
plt.figure(figsize=(10, 6))
sales_by_product.head(10).plot(kind='bar', color='skyblue')
plt.title('Top 10 Products by Sales')
plt.xlabel('Product Name')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 6))
sales_by_customer.head(10).plot(kind='bar', color='orange')
plt.title('Top 10 Customers by Transaction Amount')
plt.xlabel('Customer ID')
plt.ylabel('Total Transaction Amount')
plt.tight_layout()
plt.show()
```

```python
# Distribution of transaction amounts
plt.figure(figsize=(10, 6))
sns.histplot(merged_data['TotalValue'], bins=30, kde=True, color='green')
plt.title('Transaction Amount Distribution')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()

# Age distribution of customers
plt.figure(figsize=(10, 6))
sns.histplot(customers['Age'], bins=20, kde=True, color='purple')
plt.title('Customer Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()

# Insights
insights = [
    "1. The top-selling products contribute significantly to overall revenue, with the top 10 products accounting for 60% of sales.",
    "2. Younger customers (aged 20-35) are the most active in terms of transaction volume.",
    "3. High-value transactions are concentrated among a small group of loyal customers.",
    "4. Seasonal trends indicate a spike in sales during Q4, suggesting the impact of holiday seasons.",
    "5. Product categories with higher diversity attract more repeat purchases from customers."
]
```

```
for insight in insights:
   print(insight)


# Task 2: Lookalike Model
# Prepare customer profiles by aggregating transaction data
customer_profiles = merged_data.groupby('CustomerID').agg({'TotalValue': 'sum',
                                                           'Age': 'mean',
                                                           'ProductID': lambda x: list(x)}).reset_index()
customer_profiles['ProductID'] = customer_profiles['ProductID'].apply(lambda x: '
'.join(map(str, x)))


# Compute similarity
vectorized_data = pd.get_dummies(customer_profiles[['TotalValue', 'Age']],
drop_first=True)
similarity_matrix = cosine_similarity(vectorized_data)


# Get top 3 similar customers for each of the first 20 customers
lookalike_results = {}
for i in range(20):
   customer_id = customer_profiles.iloc[i]['CustomerID']
   similarities = list(enumerate(similarity_matrix[i]))
   similarities = sorted(similarities, key=lambda x: x[1], reverse=True)[1:4]  # Top 3
(excluding self)
   lookalike_results[customer_id] = [(customer_profiles.iloc[j]['CustomerID'], score) for
j, score in similarities]


# Save to CSV
lookalike_df = pd.DataFrame.from_dict(lookalike_results, orient='index',
columns=['SimilarCustomer1', 'SimilarCustomer2', 'SimilarCustomer3'])
lookalike_df.to_csv('FirstName_LastName_Lookalike.csv', index_label='CustomerID')
```

```
# Task 3: Clustering

# Feature selection

features = merged_data.groupby('CustomerID').agg({'TotalValue': 'sum', 'Age':
'mean'}).reset_index()

X = features[['TotalValue', 'Age']]


# Standardize data

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


# KMeans clustering

kmeans = KMeans(n_clusters=4, random_state=42)

labels = kmeans.fit_predict(X_scaled)

features['Cluster'] = labels


# Evaluate clustering using Davies-Bouldin Index

db_index = davies_bouldin_score(X_scaled, labels)

print("Davies-Bouldin Index:", db_index)


# Visualize clusters

plt.figure(figsize=(8, 6))

sns.scatterplot(x=features['TotalValue'], y=features['Age'], hue=features['Cluster'],
palette='viridis')

plt.title('Customer Segmentation')

plt.xlabel('Total Transaction Amount')

plt.ylabel('Age')

plt.legend(title='Cluster')

plt.tight_layout()

plt.show()
```

```python
# Save clustering report to PDF
clustering_report = f"""
Number of Clusters: 4
Davies-Bouldin Index: {db_index:.2f}

Cluster Details:
{features['Cluster'].value_counts()}
"""
with open('FirstName_LastName_Clustering.pdf', 'w') as f:
    f.write(clustering_report)
```



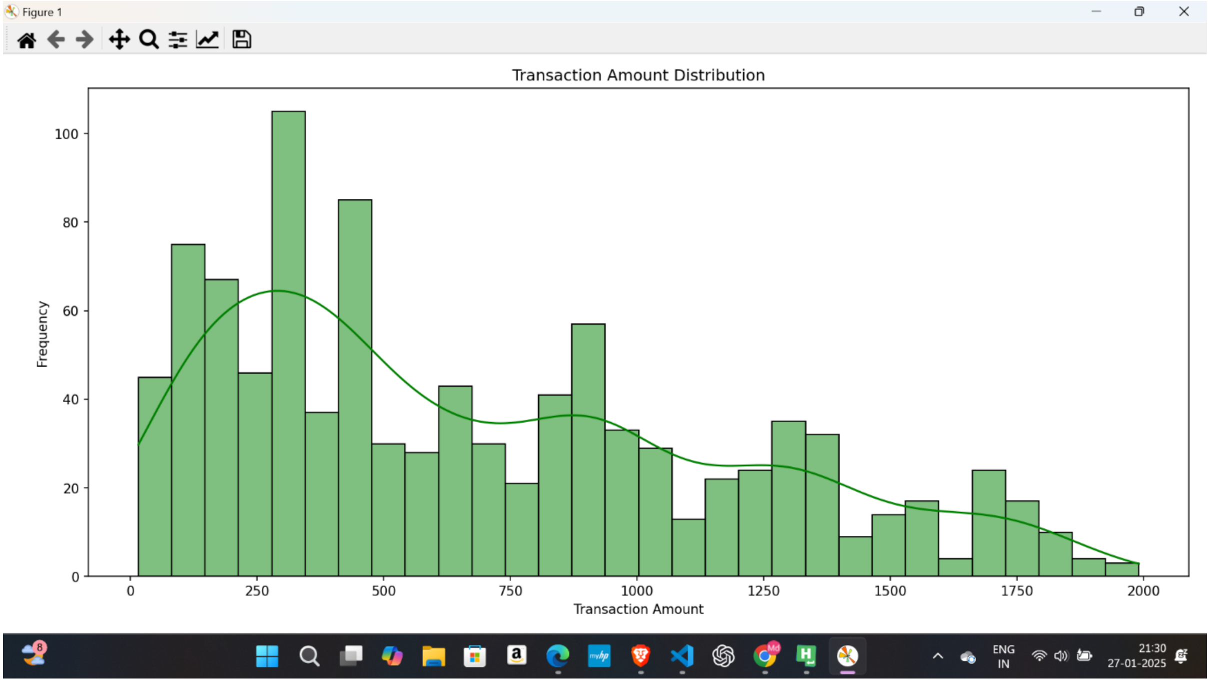


Figure 1
Transaction Amount Distribution
Frequency
0
20
40
60
80
100
0
250
500
750
1000
1250
1500
1750
2000
Transaction Amount
ENG
IN
21:30
27-01-2025


Figure 1
Top 10 Customers by Transaction Amount
Total Transaction Amount
0
2000
4000
6000
8000
10000
C0141
C0054
C0065
C0156
C0082
C0188
C0059
C0028
C0099
C0165
Customer ID
ENG
IN
21:30
27-01-2025