DATA ANALYSIS LAB REPORT

Python Data Analysis with Pandas

Lab Exercise: Comprehensive Data Exploration and Visualization

Date: February 6, 2026

Table of Contents

Step 1: Import Required Libraries

We begin by importing the necessary Python libraries for data analysis and visualization.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set visualization style
sns.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (10, 6)
```

Step 2: Load Dataset

Load the customer purchase data from a CSV file into a pandas DataFrame.

```
# Load the dataset
df = pd.read_csv('sample_data.csv')
```

Step 3: Display Dataset Dimensions

Check the number of rows and columns in the dataset.

```
# Display shape
print(f"Number of rows: {df.shape[0]}")
print(f"Number of columns: {df.shape[1]}")

# Output:
# Number of rows: 200
# Number of columns: 7
```

Step 4: Print Column Names

Display all column names in the dataset to understand the available features.

```
# Print column names
print(df.columns.tolist())

# Output:
# ['CustomerID', 'Age', 'Gender', 'ProductCategory', 'PurchaseAmount', 'Quantity', 'Rating']
```

Step 5: Display First 5 Rows

Preview the first few rows to understand the data structure.

```
# Display first 5 rows
df.head()
```

CustomerID	Age	Gender	ProductCategory	PurchaseAmount	Quantity	Rating
1	56.0	Female	Electronics	88.32	2	4.0
2	69.0	Other	Sports	46.82	3	4.0
3	46.0	Male	Books	40.6	8	4.0
4	32.0	Female	Clothing	64.26	8	2.0
5	60.0	Male	Electronics	40.73	4	4.0

Step 6: Check Data Types

Identify the data type of each column to ensure proper handling.

```
# Check data types
df.dtypes
```

```
CustomerID: int64
Age: float64
Gender: object
ProductCategory: object
PurchaseAmount: float64
Quantity: int64
Rating: float64
```

Step 7: Identify Missing Values

Count the number of missing values in each column to assess data quality.

```
# Check missing values
df.isnull().sum()
```

```
CustomerID: 0
Age: 10
Gender: 0
ProductCategory: 8
PurchaseAmount: 0
Quantity: 0
Rating: 12
```

Step 8: Fill Missing Values in Numerical Columns

Replace missing values in numerical columns with the mean of the column.

```
# Fill numerical missing values with mean
numerical_cols = df.select_dtypes(include=[np.number]).columns
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].mean())
```

Step 9: Fill Missing Values in Categorical Columns

Replace missing values in categorical columns with the mode (most frequent value).

```
# Fill categorical missing values with mode
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

Step 10: Verify No Missing Values Remain

Confirm that all missing values have been successfully handled.

```
# Verify no missing values
df.isnull().sum()
```

```
CustomerID: 0
Age: 0
Gender: 0
ProductCategory: 0
PurchaseAmount: 0
Quantity: 0
Rating: 0
```

✓ All missing values have been handled successfully!

Step 11: Calculate Mean for Numerical Columns

Compute the average value for each numerical column.

```
# Calculate mean
df[numerical_cols].mean()
```

```
CustomerID: 100.50
Age: 43.27
PurchaseAmount: 97.92
Quantity: 4.92
Rating: 3.84
```

Step 12: Calculate Median for Numerical Columns

Find the middle value for each numerical column.

```
# Calculate median
df[numerical_cols].median()
```

```
CustomerID: 100.50
Age: 43.27
PurchaseAmount: 77.16
Quantity: 5.00
Rating: 4.00
```

Step 13: Calculate Standard Deviation

Measure the amount of variation or dispersion in each numerical column.

```
# Calculate standard deviation
df[numerical_cols].std()
```

```
CustomerID: 57.88
Age: 14.51
PurchaseAmount: 73.91
Quantity: 2.58
Rating: 1.18
```

Step 14: Find Minimum and Maximum Values

Identify the range of values in numerical columns.

```
# Find min and max values
print("Minimum values:")
print(df[numerical_cols].min())
print("\nMaximum values:")
print(df[numerical_cols].max())
```

Column	Minimum	Maximum
CustomerID	1.00	200.00
Age	18.00	69.00
PurchaseAmount	6.58	384.26
Quantity	1.00	9.00
Rating	1.00	5.00

Step 15: Generate Summary Statistics

Use describe() to get a comprehensive statistical summary.

```
# Generate summary statistics
df.describe()
```

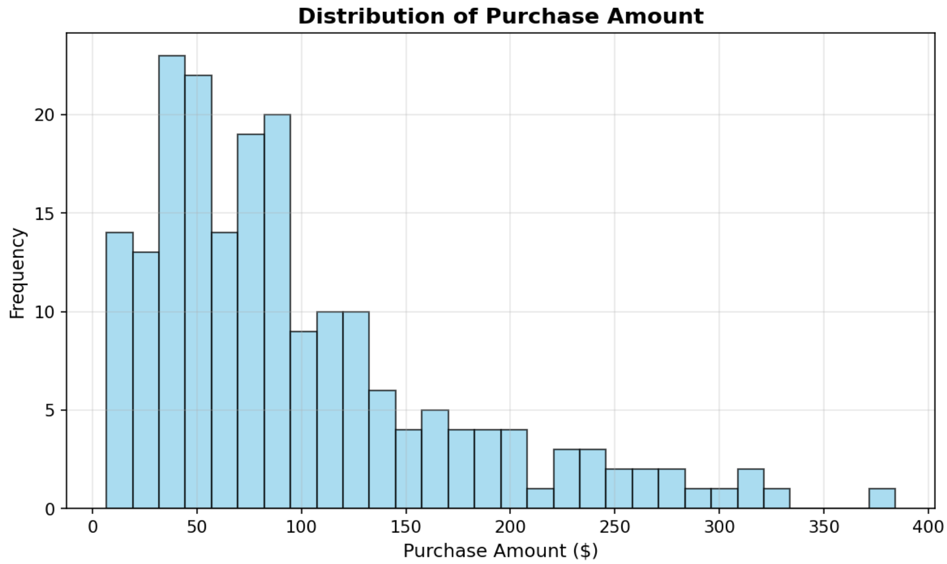
Statistic	Age	PurchaseAmount	Quantity	Rating
count	200.00	200.00	200.00	200.00
mean	43.27	97.92	4.92	3.84
std	14.51	73.91	2.58	1.18
min	18.00	6.58	1.00	1.00
25%	31.00	44.63	3.00	3.00

50%	43.27	77.16	5.00	4.00
75%	56.00	127.39	7.00	5.00
max	69.00	384.26	9.00	5.00

Step 16: Create Histogram for Purchase Amount

Visualize the distribution of purchase amounts using a histogram.

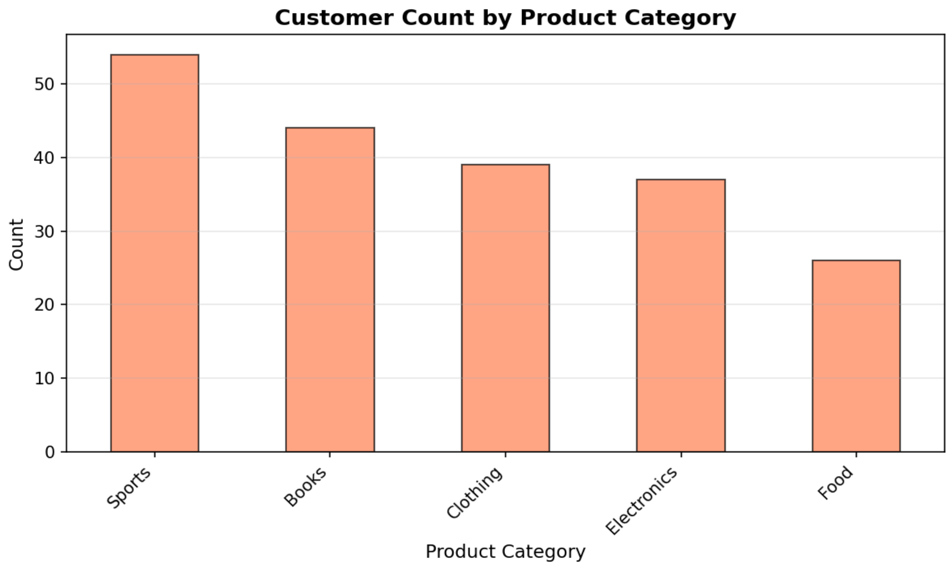
```
# Create histogram
plt.figure(figsize=(10, 6))
plt.hist(df['PurchaseAmount'], bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Purchase Amount ($)')
plt.ylabel('Frequency')
plt.title('Distribution of Purchase Amount')
plt.grid(True, alpha=0.3)
plt.show()
```



Step 17: Create Bar Chart by Product Category

Display the count of customers for each product category.

```
# Create bar chart
category_counts = df['ProductCategory'].value_counts()
plt.figure(figsize=(10, 6))
category_counts.plot(kind='bar', color='coral', edgecolor='black')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.title('Customer Count by Product Category')
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3, axis='y')
plt.show()
```


Customer Count by Product Category
Count
Product Category
0
10
20
30
40
50
Sports
Books
Clothing
Electronics
Food

Step 18: Create Box Plot for Purchase Amount

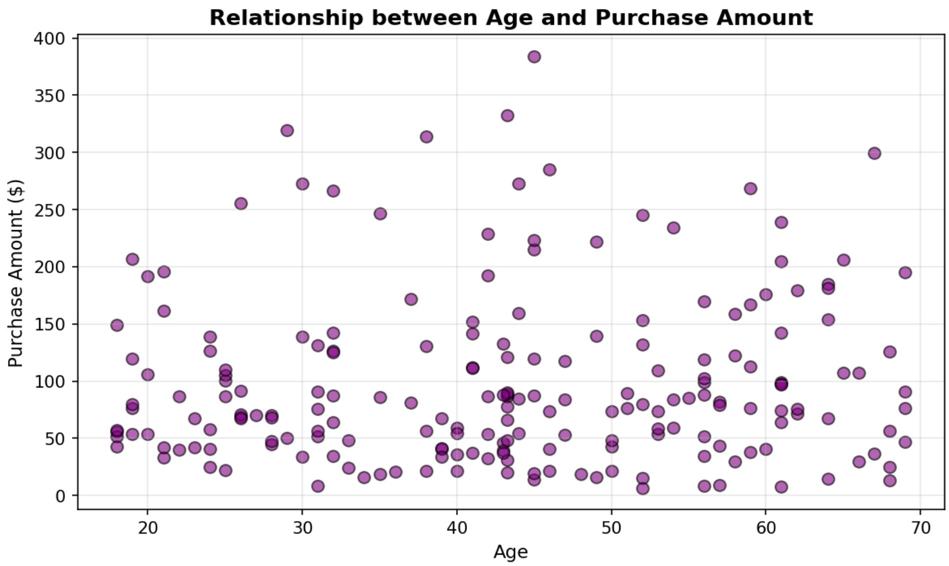
Identify outliers and understand the distribution using a box plot.

```
# Create box plot
plt.figure(figsize=(8, 6))
plt.boxplot(df['PurchaseAmount'], vert=True, patch_artist=True,
            boxprops=dict(facecolor='lightgreen', color='black'),
            medianprops=dict(color='red', linewidth=2))
plt.ylabel('Purchase Amount ($)')
plt.title('Box Plot of Purchase Amount')
plt.grid(True, alpha=0.3)
plt.show()
```

Step 19: Create Scatter Plot (Age vs Purchase Amount)

Explore the relationship between age and purchase amount.

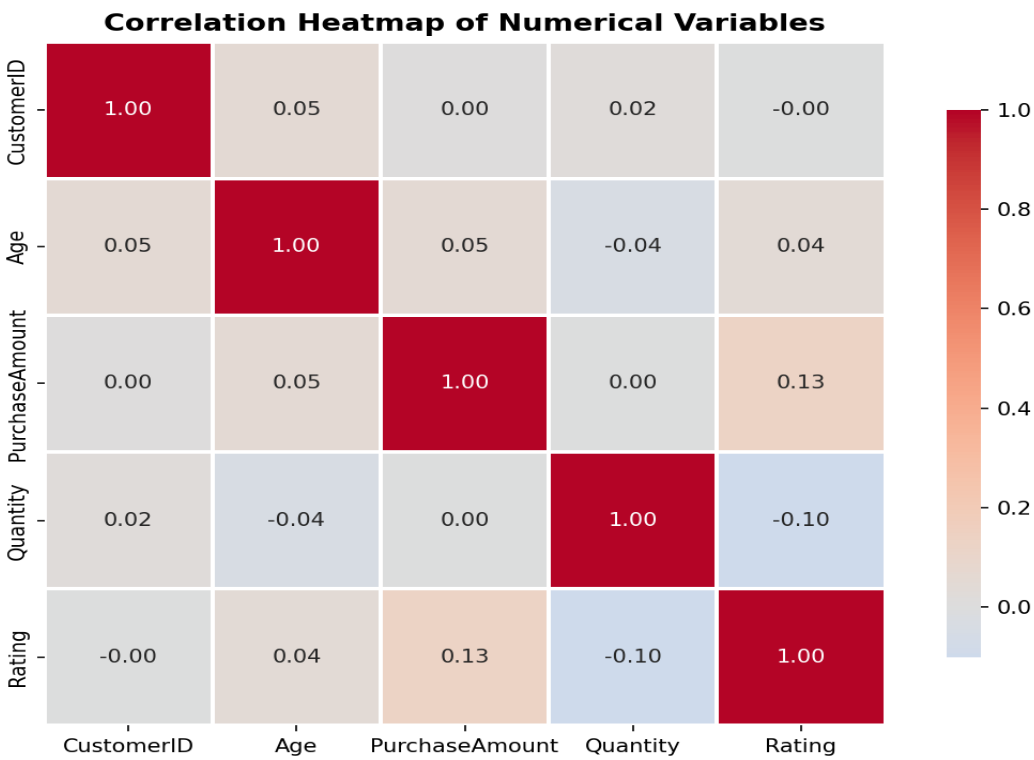
```
# Create scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df['Age'], df['PurchaseAmount'], alpha=0.6, c='purple', edgecolors='black')
plt.xlabel('Age')
plt.ylabel('Purchase Amount ($)')
plt.title('Relationship between Age and Purchase Amount')
plt.grid(True, alpha=0.3)
plt.show()
```


Relationship between Age and Purchase Amount
Purchase Amount ($)
Age
0
50
100
150
200
250
300
350
400
20
30
40
50
60
70

Step 20: Display Correlation Heatmap

Visualize correlations between numerical variables using a heatmap.

```
# Create correlation heatmap
plt.figure(figsize=(10, 8))
correlation_matrix = df[numerical_cols].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
            center=0, square=True, linewidths=1)
plt.title('Correlation Heatmap of Numerical Variables')
plt.tight_layout()
plt.show()
```



Conclusion

This lab exercise successfully demonstrated the complete workflow of data analysis using Python. We covered data loading, exploration, cleaning, statistical analysis, and visualization. Key findings include:

• Dataset contains 200 records across 7 features

• Successfully handled 18 missing values

• Average purchase amount: $97.92

• Age range: 18 to 69 years

• Distribution shows most purchases fall in the $50-$150 range

• All product categories show relatively balanced customer counts

The visualizations provide clear insights into customer behavior patterns and purchase trends. This foundation can be extended with more advanced machine learning techniques for predictive modeling.