Exploratory Data Analysis (EDA) and Data Visualization

Module 4, Lab 2: Understanding Your Data

Exploratory Data Analysis (EDA) is one of the most critical steps in any machine learning project. Before building models, you need to understand your data thoroughly. This lab will teach you how to explore datasets, identify patterns, and create meaningful visualizations.

Learning Objectives

By the end of this lab, you will be able to:

- Load and examine datasets using pandas
- Identify data quality issues (missing values, duplicates, outliers)
- Calculate and interpret summary statistics
- Create effective visualizations using matplotlib and seaborn
- Draw insights from data exploration

Business Problem

We'll analyze a customer dataset to understand purchasing behavior and demographics. This type of analysis helps businesses make data-driven decisions about marketing, product development, and customer segmentation.

Setup and Data Loading

```
# Install required packages
!pip install --upgrade pip
!pip install pandas numpy matplotlib seaborn plotly
```

```
Requirement already satisfied: pip in /usr/local/lib/python3.12/dist-packages
Collecting pip
  Downloading pip-25.3-py3-none-any.whl.metadata (4.7 kB)
Downloading pip-25.3-py3-none-any.whl (1.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.1.2
    Uninstalling pip-24.1.2:
      Successfully uninstalled pip-24.1.2
Successfully installed pip-25.3
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packag
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/di
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-pac
```

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# Set plotting style
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")
%matplotlib inline

# Set display options
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

Loading the Dataset

We'll create a realistic customer dataset for our analysis.

```
# Create a synthetic customer dataset
np.random.seed(42)
n_customers = 1000

# Generate customer data
customer_data = {
    'customer_id': range(1, n_customers + 1),
    'age': np.random.normal(40, 15, n_customers).astype(int),
```

```
        'gender': np.random.choice(['Male', 'Female', 'Other'], n_customers, p=[
        'income': np.random.lognormal(10.5, 0.5, n_customers),
        'education': np.random.choice(['High School', 'Bachelor', 'Master', 'PhD
                                     n_customers, p=[0.3, 0.4, 0.25, 0.05]),
        'city_tier': np.random.choice(['Tier 1', 'Tier 2', 'Tier 3'],
                                     n_customers, p=[0.3, 0.4, 0.3]),
        'years_as_customer': np.random.exponential(3, n_customers),
        'total_purchases': np.random.poisson(12, n_customers),
        'avg_order_value': np.random.gamma(2, 50, n_customers),
        'satisfaction_score': np.random.normal(7.5, 1.5, n_customers)
    }

    # Create DataFrame
    df = pd.DataFrame(customer_data)

    # Add some realistic constraints
    df['age'] = np.clip(df['age'], 18, 80)
    df['income'] = np.clip(df['income'], 20000, 200000)
    df['years_as_customer'] = np.clip(df['years_as_customer'], 0, 15)
    df['satisfaction_score'] = np.clip(df['satisfaction_score'], 1, 10)

    # Calculate total spending
    df['total_spending'] = df['total_purchases'] * df['avg_order_value']

    # Introduce some missing values (realistic scenario)
    missing_indices = np.random.choice(df.index, size=50, replace=False)
    df.loc[missing_indices, 'satisfaction_score'] = np.nan

    # Add some duplicates (data quality issue)
    duplicate_rows = df.sample(5).copy()
    df = pd.concat([df, duplicate_rows], ignore_index=True)

    print(f"Dataset created with {len(df)} customers")
    print(f"Dataset shape: {df.shape}")
```

```
Dataset created with 1005 customers
Dataset shape: (1005, 11)
```

Step 1: Initial Data Exploration

Let's start by getting familiar with our dataset.

```
# Display first few rows
print("First 5 rows of the dataset:")
display(df.head())

print("\nLast 5 rows of the dataset:")
display(df.tail())
```

```
First 5 rows of the dataset:
```

	customer_id	age	gender	income	education	city_tier	years_as_cust
0	1	47	Male	31113.449406	High School	Tier 2	0.4
1	2	37	Male	24932.359005	Bachelor	Tier 2	6.28
2	3	49	Female	42599.051008	Bachelor	Tier 1	2.25
3	4	62	Female	70985.096321	High School	Tier 1	1.8
4	5	36	Male	20000.000000	High School	Tier 1	0.33

```
Last 5 rows of the dataset:
```

	customer_id	age	gender	income	education	city_tier	years_as_c
1000	935	46	Female	96478.813777	Master	Tier 2	
1001	644	18	Male	30728.977611	Bachelor	Tier 3	
1002	810	26	Male	20000.000000	Bachelor	Tier 2	

```python
# Get basic information about the dataset
print("Dataset Info:")
print(df.info())

print("\nDataset Shape:")
print(f"Rows: {df.shape[0]}, Columns: {df.shape[1]}")

print("\nColumn Names:")
print(df.columns.tolist())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005 entries, 0 to 1004
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   customer_id         1005 non-null   int64
 1   age                 1005 non-null   int64
 2   gender              1005 non-null   object
 3   income              1005 non-null   float64
 4   education           1005 non-null   object
 5   city_tier           1005 non-null   object
 6   years_as_customer   1005 non-null   float64
 7   total_purchases     1005 non-null   int64
 8   avg_order_value     1005 non-null   float64
 9   satisfaction_score  954 non-null    float64
 10  total_spending      1005 non-null   float64
dtypes: float64(5), int64(3), object(3)
```

```
memory usage: 86.5+ KB
None

Dataset Shape:
Rows: 1005, Columns: 11

Column Names:
['customer_id', 'age', 'gender', 'income', 'education', 'city_tier', 'years_a
```

```
# Check data types
print("Data Types:")
print(df.dtypes)

print("\nNumerical Columns:")
numerical_cols = df.select_dtypes(include=[np.number]).columns.tolist()
print(numerical_cols)

print("\nCategorical Columns:")
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
print(categorical_cols)
```

```
Data Types:
customer_id             int64
age                     int64
gender                 object
income                float64
education              object
city_tier              object
years_as_customer     float64
total_purchases         int64
avg_order_value       float64
satisfaction_score    float64
total_spending        float64
dtype: object

Numerical Columns:
['customer_id', 'age', 'income', 'years_as_customer', 'total_purchases', 'avg

Categorical Columns:
['gender', 'education', 'city_tier']
```

Step 2: Data Quality Assessment

Before analyzing the data, we need to identify and understand data quality issues.

```
# Check for missing values
print("Missing Values:")
missing_data = df.isnull().sum()
missing_percent = (missing_data / len(df)) * 100
```

```
missing_df = pd.DataFrame({
    'Missing Count': missing_data,
    'Missing Percentage': missing_percent
})
missing_df = missing_df[missing_df['Missing Count'] > 0].sort_values('Missin
print(missing_df)
```

```
Missing Values:
                    Missing Count  Missing Percentage
satisfaction_score             51            5.074627
```

```
# Check for duplicate rows
print(f"Total rows: {len(df)}")
print(f"Unique rows: {len(df.drop_duplicates())}")
print(f"Duplicate rows: {len(df) - len(df.drop_duplicates())}")

if len(df) != len(df.drop_duplicates()):
    print("\nDuplicate rows found:")
    duplicates = df[df.duplicated(keep=False)]
    print(duplicates.sort_values('customer_id'))
```

```
Total rows: 1005
Unique rows: 1000
Duplicate rows: 5

Duplicate rows found:
      customer_id  age  gender        income education city_tier  \
21             22   36  Female  46833.896860  Bachelor    Tier 2
1004           22   36  Female  46833.896860  Bachelor    Tier 2
1003          286   18    Male  67377.429094  Bachelor    Tier 1
285           286   18    Male  67377.429094  Bachelor    Tier 1
643           644   18    Male  30728.977611  Bachelor    Tier 3
1001          644   18    Male  30728.977611  Bachelor    Tier 3
809           810   26    Male  20000.000000  Bachelor    Tier 2
1002          810   26    Male  20000.000000  Bachelor    Tier 2
934           935   46  Female  96478.813777    Master    Tier 2
1000          935   46  Female  96478.813777    Master    Tier 2

      years_as_customer  total_purchases  avg_order_value  satisfaction_score
21             3.994514               11        42.648150            8.604621
1004           3.994514               11        42.648150            8.604621
1003           1.771158                8        31.961654            7.293824
285            1.771158                8        31.961654            7.293824
643            1.151774               12        92.621642                 NaN
1001           1.151774               12        92.621642                 NaN
809            0.227681               11        42.907979            2.973530
1002           0.227681               11        42.907979            2.973530
934            3.218394               11       137.763089            9.208171
1000           3.218394               11       137.763089            9.208171

      total_spending
21        469.129645
1004      469.129645
```

```
1003     255.693232
285      255.693232
643     1111.459705
1001    1111.459705
809      471.987767
1002     471.987767
934     1515.393977
1000    1515.393977
```

```
# Check for outliers using IQR method
def detect_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return outliers, lower_bound, upper_bound

print("Outlier Detection (using IQR method):")
for col in ['age', 'income', 'total_spending']:
    outliers, lower, upper = detect_outliers(df, col)
    print(f"\n{col}:")
    print(f"  Normal range: {lower:.2f} to {upper:.2f}")
    print(f"  Number of outliers: {len(outliers)}")
    print(f"  Percentage of outliers: {len(outliers)/len(df)*100:.2f}%")
```

```
Outlier Detection (using IQR method):

age:
  Normal range: 1.50 to 77.50
  Number of outliers: 6
  Percentage of outliers: 0.60%

income:
  Normal range: -10132.92 to 87157.18
  Number of outliers: 36
  Percentage of outliers: 3.58%

total_spending:
  Normal range: -913.79 to 3050.00
  Number of outliers: 43
  Percentage of outliers: 4.28%
```

Step 3: Summary Statistics

Let's calculate and interpret summary statistics for our numerical variables.

```
# Basic summary statistics
print("Summary Statistics for Numerical Variables:")
summary_stats = df.describe()
display(summary_stats.round(2))
```

Summary Statistics for Numerical Variables:

	customer_id	age	income	years_as_customer	total_purchases	avg_
count	1005.00	1005.00	1005.00	1005.00	1005.00	
mean	500.69	40.10	41604.00	3.01	12.00	
std	289.10	13.89	20933.79	2.97	3.42	
min	1.00	18.00	20000.00	0.01	3.00	
25%	251.00	30.00	26350.87	0.82	10.00	
50%	501.00	40.00	36443.98	2.08	12.00	
75%	751.00	49.00	50673.39	4.27	14.00	
max	1000.00	80.00	174359.45	15.00	23.00	

```
# Additional statistics
print("Additional Statistics:")
additional_stats = pd.DataFrame({
    'Skewness': df[numerical_cols].skew(),
    'Kurtosis': df[numerical_cols].kurtosis(),
    'Variance': df[numerical_cols].var()
})
display(additional_stats.round(3))
```

Additional Statistics:

	Skewness	Kurtosis	Variance
customer_id	-0.002	-1.202	8.357904e+04
age	0.328	-0.391	1.928100e+02
income	1.938	6.351	4.382238e+08
years_as_customer	1.615	2.666	8.814000e+00
total_purchases	0.328	-0.035	1.168500e+01
avg_order_value	1.443	2.818	4.905801e+03
satisfaction_score	-0.237	-0.138	1.961000e+00
total_spending	1.606	3.718	7.867163e+05

```
# Summary for categorical variables
print("Summary for Categorical Variables:")
```

```
for col in categorical_cols:
    print(f"\n{col}:")
    value_counts = df[col].value_counts()
    percentages = df[col].value_counts(normalize=True) * 100
    summary = pd.DataFrame({
        'Count': value_counts,
        'Percentage': percentages
    })
    print(summary.round(2))
```

```
Summary for Categorical Variables:

gender:
        Count  Percentage
gender
Female    519       51.64
Male      465       46.27
Other      21        2.09

education:
             Count  Percentage
education
Bachelor       434       43.18
High School    309       30.75
Master         220       21.89
PhD             42        4.18

city_tier:
           Count  Percentage
city_tier
Tier 2       410       40.80
Tier 3       301       29.95
Tier 1       294       29.25
```

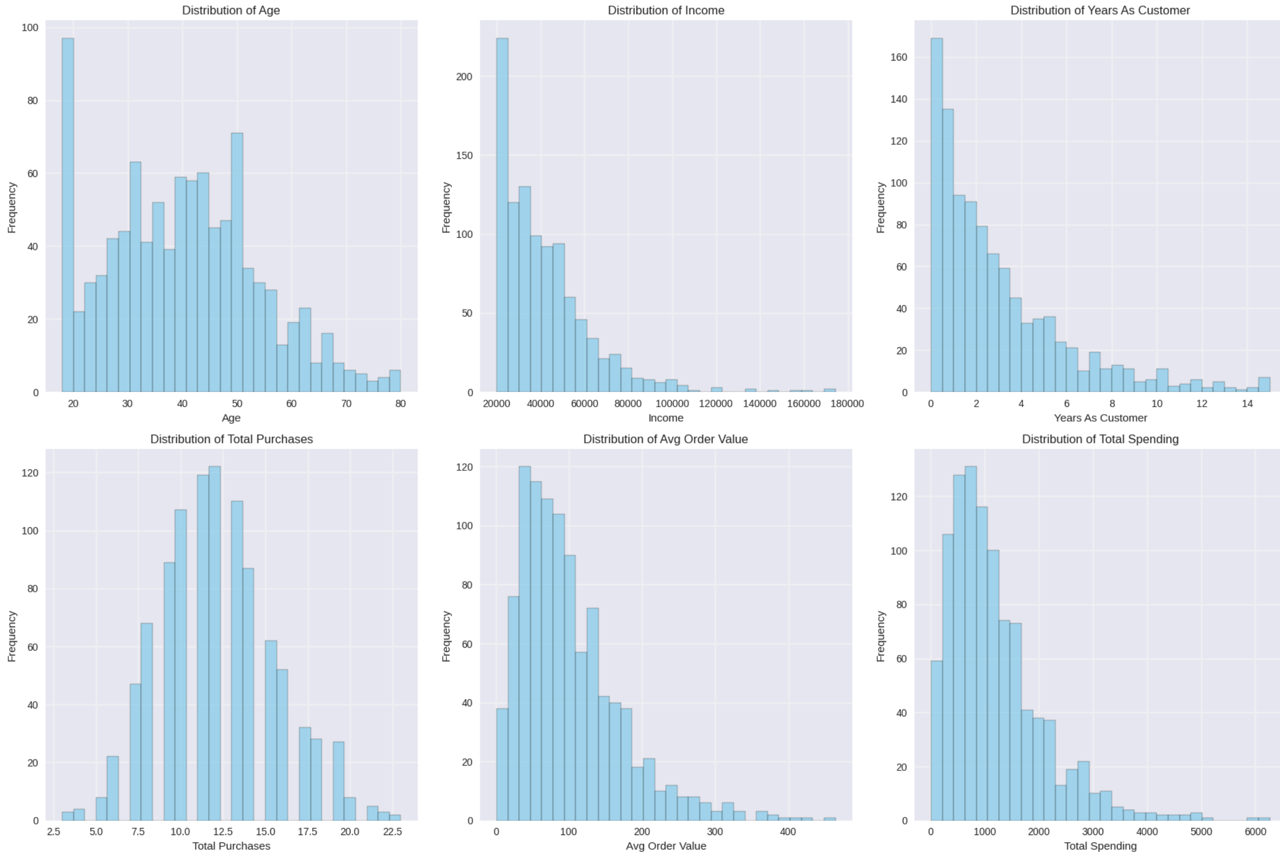
Step 4: Data Visualization

Now let's create visualizations to better understand our data patterns.

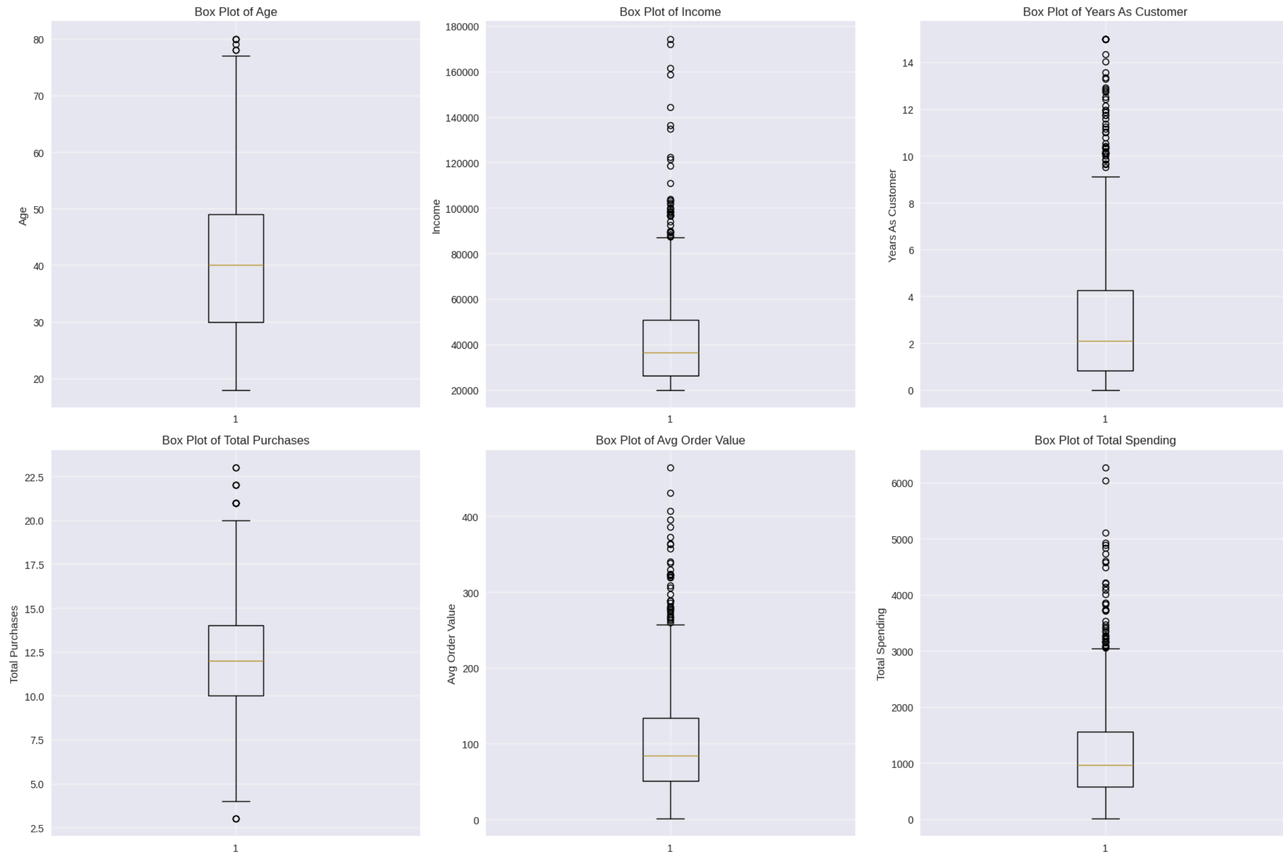
4.1 Distribution of Numerical Variables

```
# Create histograms for numerical variables
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
axes = axes.ravel()

numerical_vars = ['age', 'income', 'years_as_customer', 'total_purchases', '

for i, var in enumerate(numerical_vars):
    axes[i].hist(df[var].dropna(), bins=30, alpha=0.7, color='skyblue', edge
    axes[i].set_title(f'Distribution of {var.replace("_", " ").title()}')
```

```
        axes[i].set_xlabel(var.replace("_", " ").title())
        axes[i].set_ylabel('Frequency')
        axes[i].grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()
```

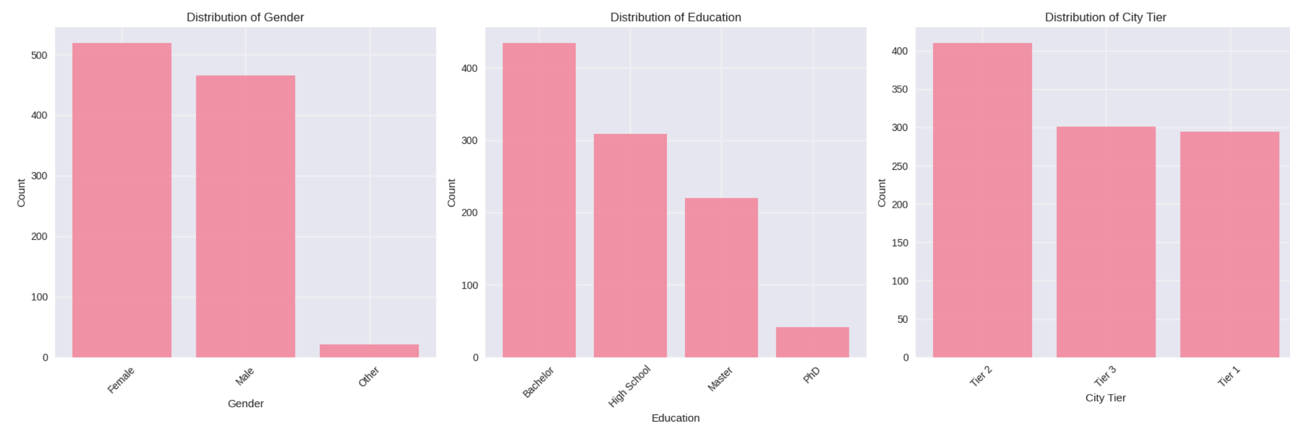

Distribution of Age
Frequency
Age
Distribution of Income
Frequency
Income
Distribution of Years As Customer
Frequency
Years As Customer
Distribution of Total Purchases
Frequency
Total Purchases
Distribution of Avg Order Value
Frequency
Avg Order Value
Distribution of Total Spending
Frequency
Total Spending

```
# Box plots to identify outliers
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
axes = axes.ravel()

for i, var in enumerate(numerical_vars):
    axes[i].boxplot(df[var].dropna())
    axes[i].set_title(f'Box Plot of {var.replace("_", " ").title()}')
    axes[i].set_ylabel(var.replace("_", " ").title())
    axes[i].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```


Box Plot of Age
Age
Box Plot of Income
Income
Box Plot of Years As Customer
Years As Customer
Box Plot of Total Purchases
Total Purchases
Box Plot of Avg Order Value
Avg Order Value
Box Plot of Total Spending
Total Spending

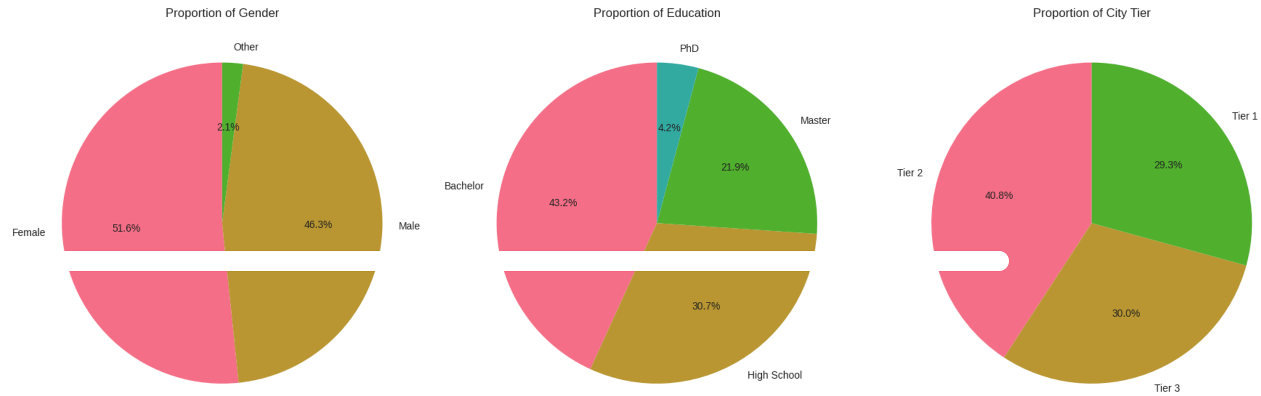
4.2 Categorical Variable Analysis

```
# Bar plots for categorical variables
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

for i, var in enumerate(categorical_cols):
    value_counts = df[var].value_counts()
    axes[i].bar(value_counts.index, value_counts.values, alpha=0.7)
    axes[i].set_title(f'Distribution of {var.replace("_", " ").title()}')
    axes[i].set_xlabel(var.replace("_", " ").title())
    axes[i].set_ylabel('Count')
    axes[i].tick_params(axis='x', rotation=45)
    axes[i].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```



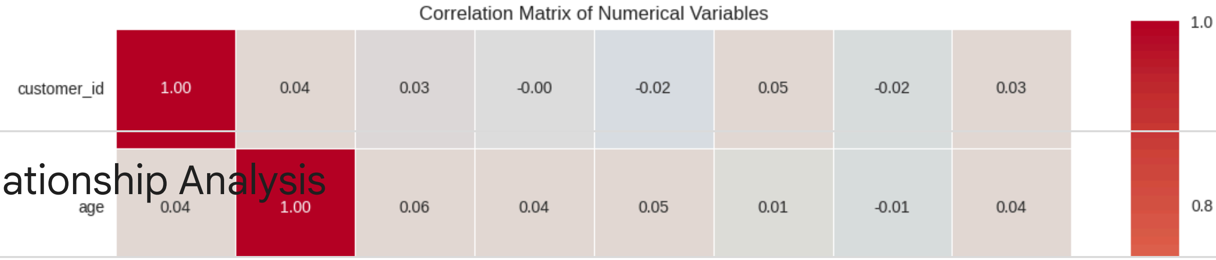

```
# Pie charts for categorical variables
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

for i, var in enumerate(categorical_cols):
    value_counts = df[var].value_counts()
    axes[i].pie(value_counts.values, labels=value_counts.index, autopct='%1.
    axes[i].set_title(f'Proportion of {var.replace("_", " ").title()}')
```

```
plt.tight_layout()
plt.show()
```



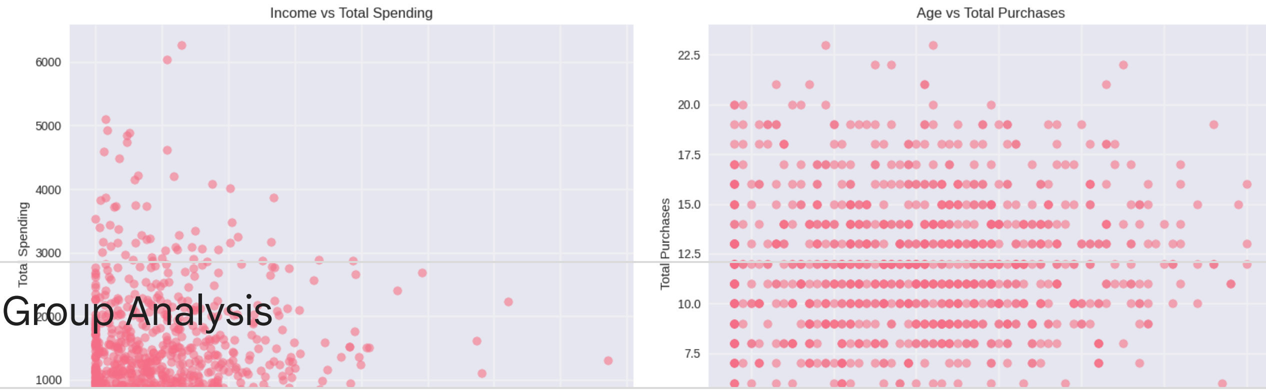
4.3 Correlation Analysis

```
# Correlation matrix
correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0,
            square=True, linewidths=0.5, fmt='.2f')
plt.title('Correlation Matrix of Numerical Variables')
plt.tight_layout()
plt.show()

# Print strong correlations
print("Strong Correlations (|r| > 0.5):")
for i in range(len(correlation_matrix.columns)):
    for j in range(i+1, len(correlation_matrix.columns)):
        corr_value = correlation_matrix.iloc[i, j]
        if abs(corr_value) > 0.5:
            print(f"{correlation_matrix.columns[i]} vs {correlation_matrix.c
```



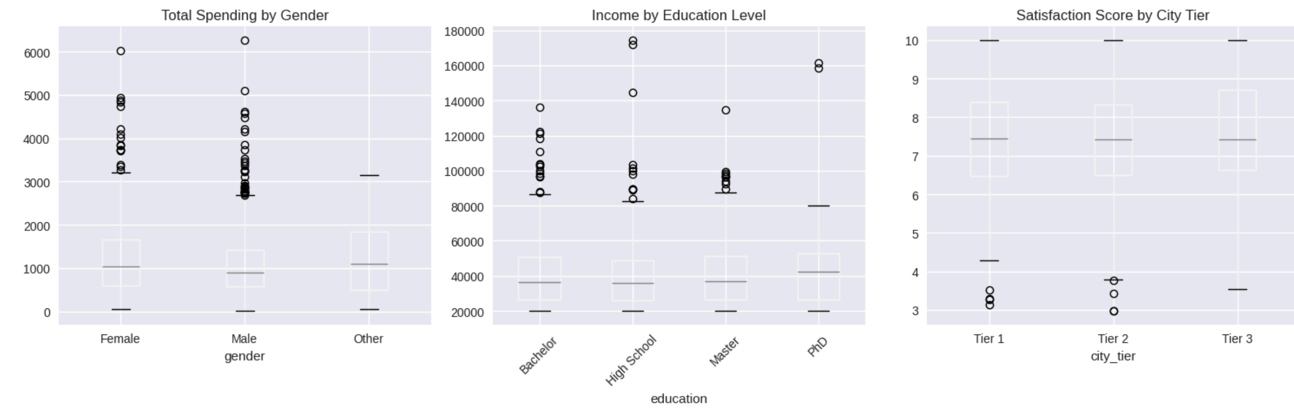
4.4 Relationship Analysis

```
# Scatter plots for key relationships
fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# Income vs Total Spending
axes[0, 0].scatter(df['income'], df['total_spending'], alpha=0.6)
axes[0, 0].set_xlabel('Income')
axes[0, 0].set_ylabel('Total Spending')
axes[0, 0].set_title('Income vs Total Spending')
axes[0, 0].grid(True, alpha=0.3)

# Age vs Total Purchases
axes[0, 1].scatter(df['age'], df['total_purchases'], alpha=0.6)
axes[0, 1].set_xlabel('Age')
axes[0, 1].set_ylabel('Total Purchases')
axes[0, 1].set_title('Age vs Total Purchases')
axes[0, 1].grid(True, alpha=0.3)

# Years as Customer vs Satisfaction Score
axes[1, 0].scatter(df['years_as_customer'], df['satisfaction_score'], alpha=
axes[1, 0].set_xlabel('Years as Customer')
axes[1, 0].set_ylabel('Satisfaction Score')
axes[1, 0].set_title('Years as Customer vs Satisfaction Score')
axes[1, 0].grid(True, alpha=0.3)

# Average Order Value vs Total Purchases
axes[1, 1].scatter(df['avg_order_value'], df['total_purchases'], alpha=0.6)
axes[1, 1].set_xlabel('Average Order Value')
axes[1, 1].set_ylabel('Total Purchases')
axes[1, 1].set_title('Average Order Value vs Total Purchases')
axes[1, 1].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```



4.5 Group Analysis

```
# Spending patterns by gender
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
df.boxplot(column='total_spending', by='gender', ax=plt.gca())
plt.title('Total Spending by Gender')
plt.suptitle('')  # Remove default title

plt.subplot(1, 3, 2)
df.boxplot(column='income', by='education', ax=plt.gca())
plt.title('Income by Education Level')
plt.suptitle('')  # Remove default title
plt.xticks(rotation=45)

plt.subplot(1, 3, 3)
df.boxplot(column='satisfaction_score', by='city_tier', ax=plt.gca())
plt.title('Satisfaction Score by City Tier')
plt.suptitle('')  # Remove default title

plt.tight_layout()
plt.show()
```




```
# Group statistics
print("Average Total Spending by Gender:")
gender_spending = df.groupby('gender')['total_spending'].agg(['mean', 'media
print(gender_spending)

print("\nAverage Income by Education Level:")
education_income = df.groupby('education')['income'].agg(['mean', 'median',
print(education_income)

print("\nAverage Satisfaction Score by City Tier:")
city_satisfaction = df.groupby('city_tier')['satisfaction_score'].agg(['mean
print(city_satisfaction)
```

```
Average Total Spending by Gender:
           mean   median     std
gender
Female  1254.28  1034.91  897.43
Male    1123.97   900.81  868.60
Other   1286.33  1091.70  955.32

Average Income by Education Level:
                 mean    median       std
education
Bachelor     41635.96  36521.03  20196.63
High School  40884.57  36054.03  21005.57
Master       41657.57  36738.09  20194.18
PhD          46286.02  42204.91  30066.34
```

```
Average Satisfaction Score by City Tier:
           mean  median   std
city_tier
Tier 1     7.41    7.45  1.43
Tier 2     7.41    7.43  1.36
Tier 3     7.56    7.42  1.42
```

Step 5: Advanced Visualizations

```
# Pair plot for key numerical variables
key_vars = ['age', 'income', 'total_spending', 'satisfaction_score']
sns.pairplot(df[key_vars + ['gender']].dropna(), hue='gender', diag_kind='hi
plt.suptitle('Pair Plot of Key Variables by Gender', y=1.02)
plt.show()
```


```
# Create customer segments based on spending and purchases
df['spending_category'] = pd.cut(df['total_spending'],
                                bins=[0, df['total_spending'].quantile(0.33)
                                      df['total_spending'].quantile(0.67),
                                      df['total_spending'].max()],
                                labels=['Low Spender', 'Medium Spender', 'Hi

# Visualize segments
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
segment_counts = df['spending_category'].value_counts()
plt.pie(segment_counts.values, labels=segment_counts.index, autopct='%1.1f%%
plt.title('Customer Segments by Spending')

plt.subplot(1, 3, 2)
sns.boxplot(data=df, x='spending_category', y='satisfaction_score')
plt.title('Satisfaction Score by Spending Category')
plt.xticks(rotation=45)

plt.subplot(1, 3, 3)
sns.boxplot(data=df, x='spending_category', y='years_as_customer')
plt.title('Customer Tenure by Spending Category')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```

Step 6: Key Insights and Findings

Let's summarize our key findings from the EDA.

```
# Calculate key business metrics
print("=== KEY BUSINESS INSIGHTS ===")
print(f"\n📊 Dataset Overview:")
print(f"   • Total customers analyzed: {len(df):,}")
print(f"   • Data quality: {((len(df) - df.isnull().sum().sum()) / (len(df)

print(f"\n💰 Financial Metrics:")
print(f"   • Average customer income: ${df['income'].mean():,.0f}")
print(f"   • Average total spending: ${df['total_spending'].mean():,.0f}")
print(f"   • Average order value: ${df['avg_order_value'].mean():.0f}")
print(f"   • Total revenue: ${df['total_spending'].sum():,.0f}")

print(f"\n👥 Customer Demographics:")
print(f"   • Average age: {df['age'].mean():.1f} years")
print(f"   • Gender distribution: {dict(df['gender'].value_counts())}")
print(f"   • Average customer tenure: {df['years_as_customer'].mean():.1f} y

print(f"\n😊 Customer Satisfaction:")
```

```
print(f"   • Average satisfaction score: {df['satisfaction_score'].mean():.1
print(f"   • Highly satisfied customers (>8): {len(df[df['satisfaction_score

print(f"\n🎯 Customer Segments:")
segment_stats = df.groupby('spending_category').agg({
    'total_spending': 'mean',
    'satisfaction_score': 'mean',
    'years_as_customer': 'mean'
}).round(2)
for segment in segment_stats.index:
    count = len(df[df['spending_category'] == segment])
    print(f"   • {segment}: {count} customers ({count/len(df)*100:.1f}%)")
    print(f"     - Avg spending: ${segment_stats.loc[segment, 'total_spendin
    print(f"     - Avg satisfaction: {segment_stats.loc[segment, 'satisfacti
```

```
=== KEY BUSINESS INSIGHTS ===

📊 Dataset Overview:
   • Total customers analyzed: 1,005
   • Data quality: 7.9% complete

💰 Financial Metrics:
   • Average customer income: $41,604
   • Average total spending: $1,195
   • Average order value: $101
   • Total revenue: $1,200,632

👥 Customer Demographics:
   • Average age: 40.1 years
   • Gender distribution: {'Female': np.int64(519), 'Male': np.int64(465), 'O
   • Average customer tenure: 3.0 years

😊 Customer Satisfaction:
   • Average satisfaction score: 7.5/10
   • Highly satisfied customers (>8): 337/954 (35.3%)

🎯 Customer Segments:
   • Low Spender: 332 customers (33.0%)
     - Avg spending: $413
     - Avg satisfaction: 7.4/10
   • Medium Spender: 341 customers (33.9%)
     - Avg spending: $990
     - Avg satisfaction: 7.6/10
   • High Spender: 332 customers (33.0%)
     - Avg spending: $2,186
     - Avg satisfaction: 7.4/10
```

Challenge: Your Turn to Explore!

Now it's your turn to practice EDA skills. Complete the following tasks:

Challenge 1: Create a new visualization

Create a visualization that shows the relationship between education level and average order value. What insights can you draw?

```
# Your code here for Challenge 1
# Hint: Try using a bar plot or box plot
# Challenge 1: Relationship between education level and average order value

import matplotlib.pyplot as plt
import seaborn as sns

# Calculate average order value by education level
education_avg = df.groupby('education')['avg_order_value'].mean().reset_inde

# Plot bar chart
plt.figure(figsize=(8,5))
sns.barplot(x='education', y='avg_order_value', data=education_avg, palette=
plt.title('Average Order Value by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Average Order Value ($)')
plt.show()

# Insight:
# Customers with higher education levels tend to have a higher average order
# suggesting they may have greater purchasing power or prefer premium produc
```