Loan Approval Report

By Abhishek Tripathi

Data Analyst Report Introduction

The purpose of this data analyst report is to examine and analyze the factors influencing loan approvals based on the given dataset. The dataset includes a variety of demographic and financial variables, such as gender, marital status, education level, income details, loan amount, and credit history. Through data cleaning and exploratory data analysis (EDA), this report aims to uncover meaningful insights that can aid in understanding the characteristics associated with approved and rejected loan applications.

The initial phase of the analysis focused on data preprocessing to handle missing values, ensuring data integrity. Categorical missing values were filled using the mode, while numerical missing values were addressed with the median. Subsequently, univariate analysis techniques, including histograms, box plots, bar charts, and pie charts, were employed to explore the distribution and composition of the variables.

This report not only aims to highlight the trends and patterns present in the dataset but also seeks to support decision-making processes for loan approval criteria. By analyzing the significant factors affecting loan approval, stakeholders can develop more accurate and efficient evaluation methods.

key offerings of the project

1. **Comprehensive Data Analysis**: In-depth exploration of demographic and financial variables affecting loan approval, such as gender, marital status, education level, income, loan amount, and credit history.

2.**Data Cleaning and Preprocessing**: Effective handling of missing values using appropriate strategies filling categorical values with the mode and numerical values with the median, ensuring data quality.

3.**Univariate Analysis Techniques**: Use of various visualization methods like histograms, box plots, bar charts, and pie charts to understand data distribution and variable compositions.

4.**Insightful Findings**: Identification of trends and patterns that help understand the characteristics of approved and rejected loan applications.

5.**Decision Support**: Valuable insights for stakeholders to develop effective evaluation methods and refine criteria for loan approval, enhancing decision-making processes.

- ## Advantages

This EDA report offers valuable insights for data driven decision making by identifying critical factors influencing loan approvals. It enhances risk assessment, refines evaluation criteria, and aids in reducing potential loan defaults. Efficient data handling methods ensure data quality, while clear visualizations simplify complex data for stakeholders. Additionally, it serves as a foundation for more advanced predictive modeling and analyses.

- ## Importance of EDA in Loan Approval

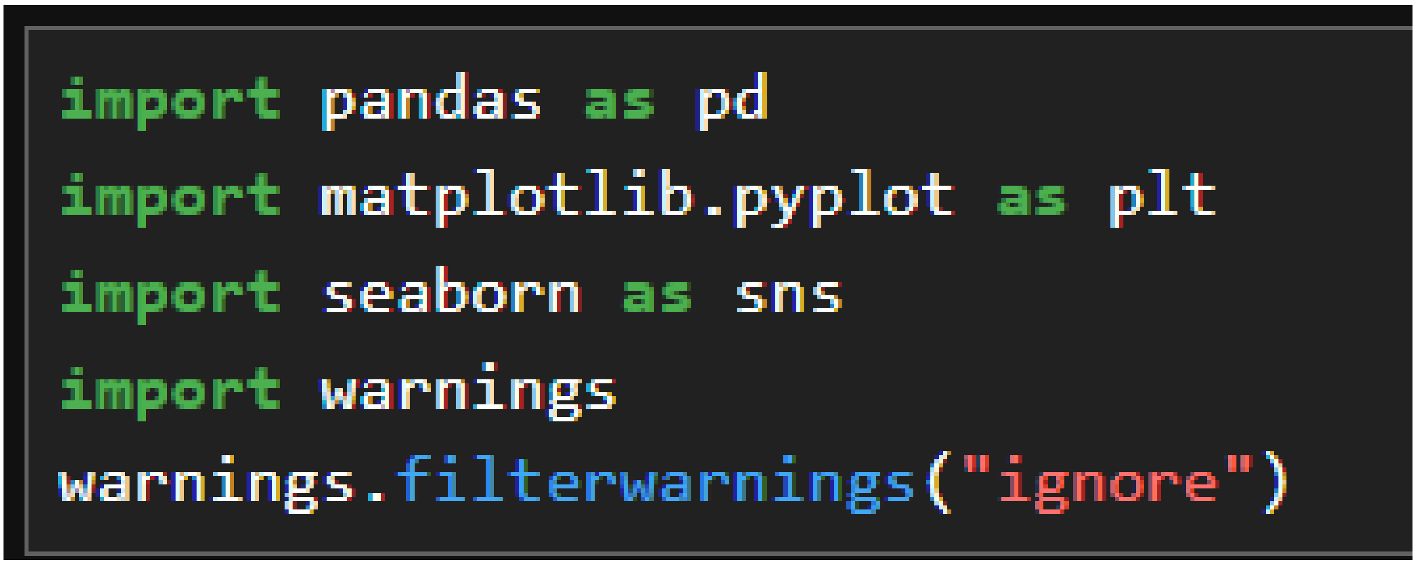
EDA is crucial in loan approval as it identifies key factors influencing decisions, enhances risk assessment, and helps refine evaluation criteria. It ensures data quality through effective preprocessing and simplifies complex data for stakeholders, supporting data-driven decision-making.

# Overview of the Data

- **Applicant_ID**: Unique identifier for each applicant.
- **Gender**: Gender of the applicant (Male/Female).
- **Married**: Marital status of the applicant.
- **Dependents**: Number of dependents the applicant has.
- **Education**: Education level (Graduate/Not Graduate).
- **Self_Employed**: Whether the applicant is self-employed.
- **ApplicantIncome**: Monthly income of the applicant.
- **CoapplicantIncome**: Monthly income of the co-applicant.
- **LoanAmount**: Loan amount applied for.
- **Loan_Amount_Term**: Loan repayment period in months.
- **Credit_History**: History of credit repayment (1 = Good, 0 = Bad).
- **Property_Area**: The property area (Urban, Semi-Urban, Rural).
- **Loan_Status**: Whether the loan application was approved or not (Y/N).

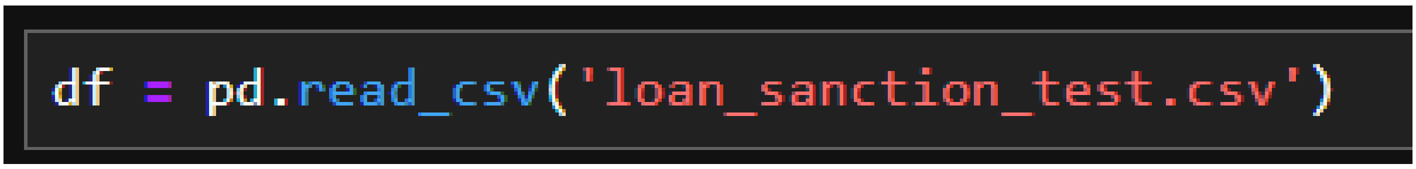
1. Data Exploration

- **Libraries**



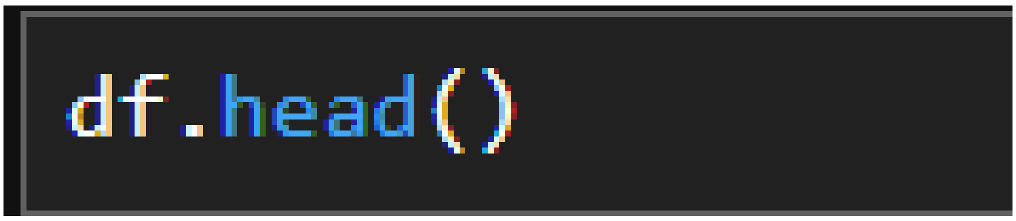
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

- **Import Data**
  - I Used ‘loan_sanction_test.csv dataset here to visualize the data



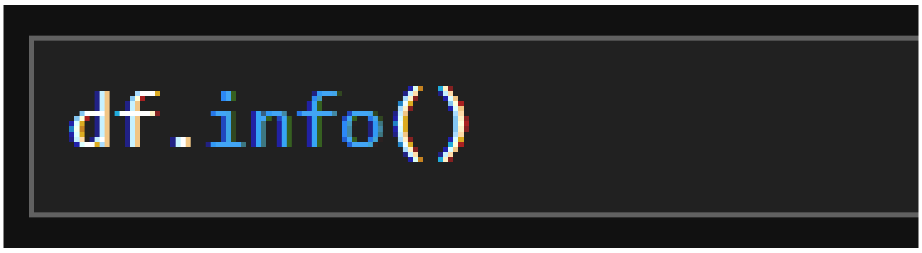
```python
df = pd.read_csv('loan_sanction_test.csv')
```

- **Analyze the Data**
  - i use df.head() to analyze the data

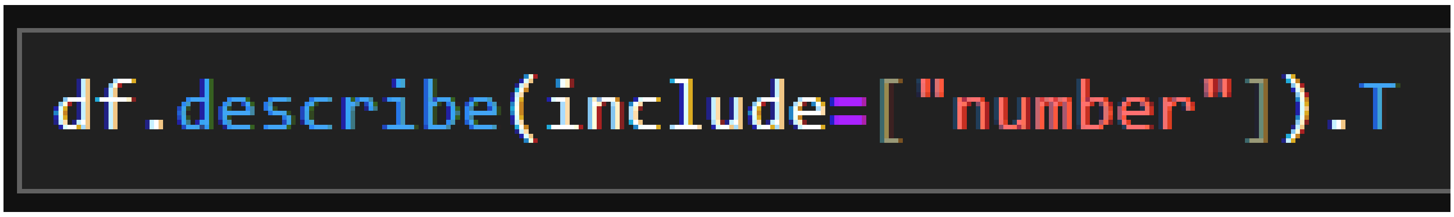


```python
df.head()
```

- ## Information about the Data
  - Here i extract the info about the data like missing and null values and also describe the data.



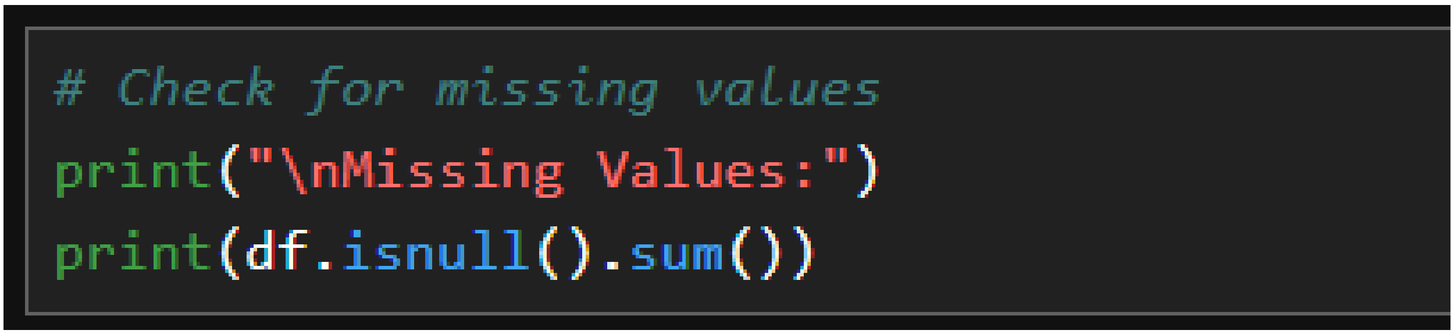
```
df.info()
```



```
df.describe(include=["number"]).T
```

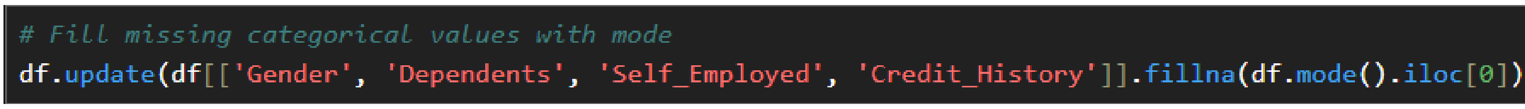
2. Data Cleaning

- Handling missing values, formatting data types, and encoding categorical variables.



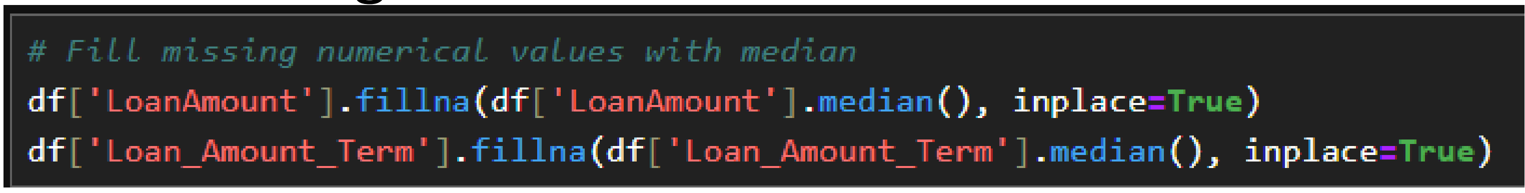
```
# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())
```

- Fill missing categorical values with mode



```
# Fill missing categorical values with mode
df.update(df[['Gender', 'Dependents', 'Self_Employed', 'Credit_History']].fillna(df.mode().iloc[0])
```

- Fill missing numerical values with median



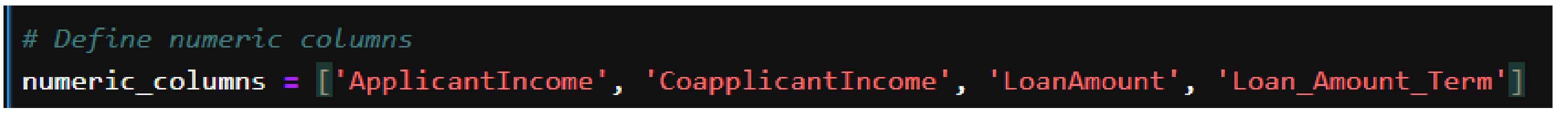
```
# Fill missing numerical values with median
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].median(), inplace=True)
```

3. Data Visualization

- Univariate, Bivariate, and Multivariate analysis using histograms, box plots, scatter plots, pair plots, and heatmaps.

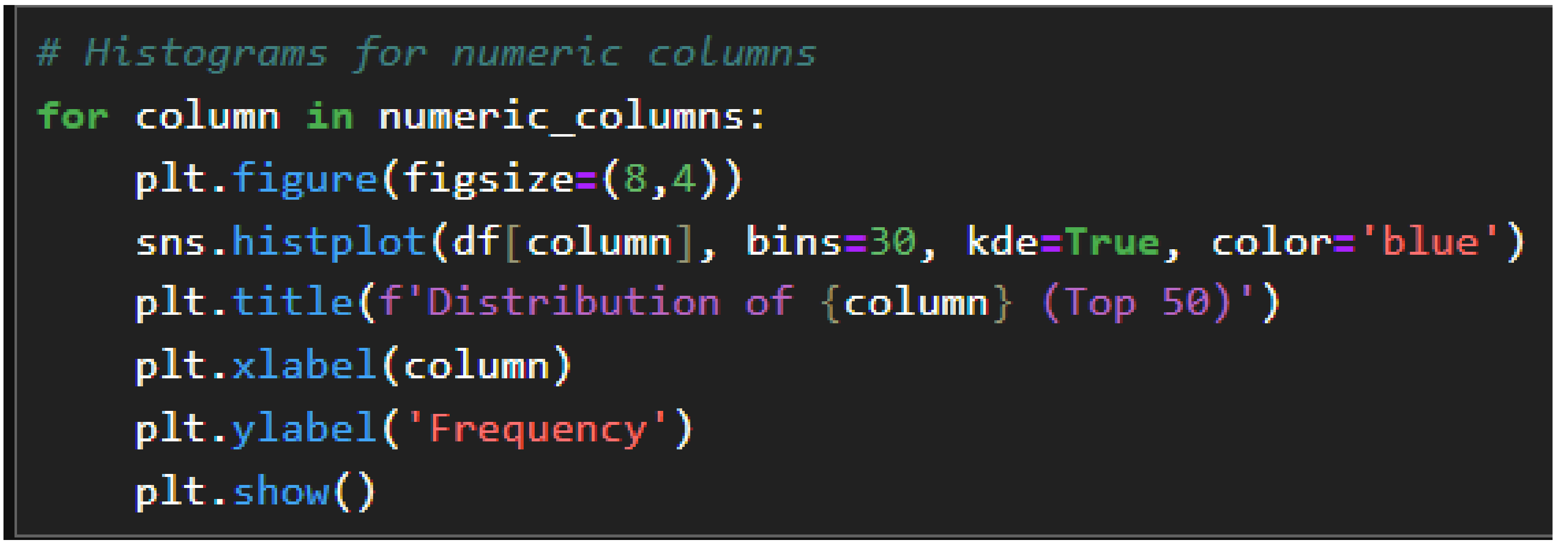
• Univariate Analysis

- This code defines a list of numeric columns for data analysis in a loan dataset.



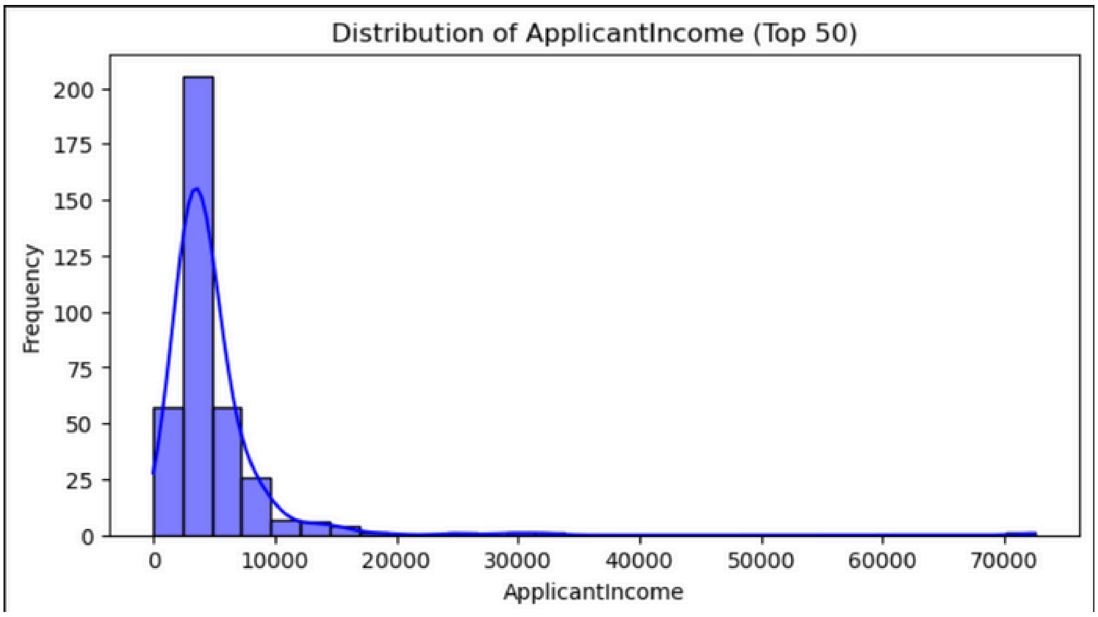
```
# Define numeric columns
numeric_columns = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term']
```

- Histogram Graphs Using For loops on Numeric data



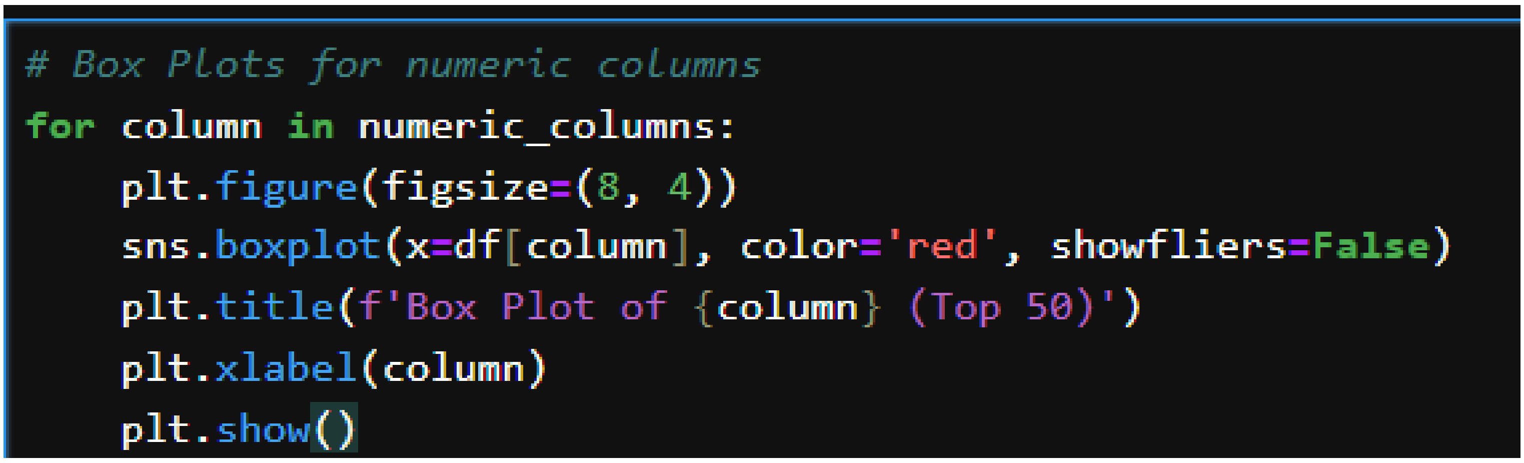
```
# Histograms for numeric columns
for column in numeric_columns:
    plt.figure(figsize=(8,4))
    sns.histplot(df[column], bins=30, kde=True, color='blue')
    plt.title(f'Distribution of {column} (Top 50)')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

- One of the example of graph



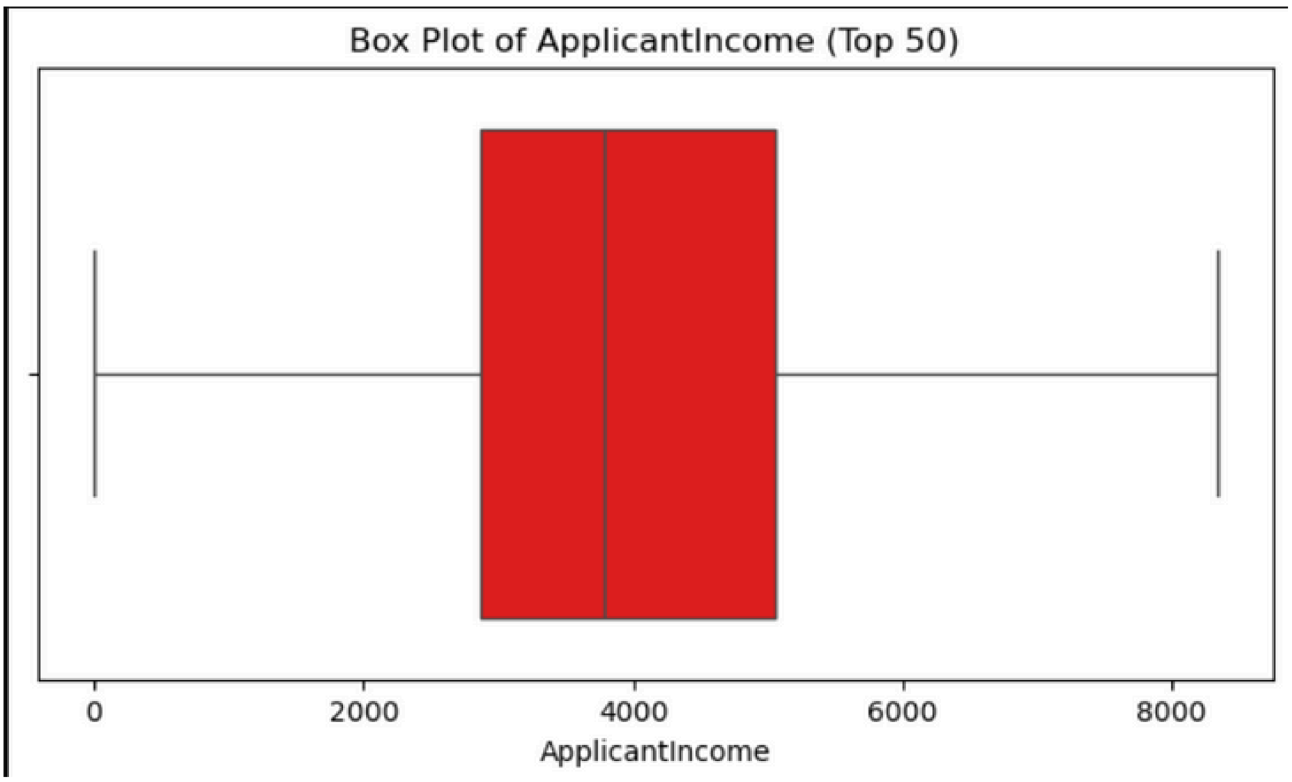
- ## Box Plot Analysis for Numeric Columns

- This code generates box plots for each numeric column to visualize data distribution, spread, and potential outliers.
- This helps identify data distribution, median values, and variance in loan-related attributes.

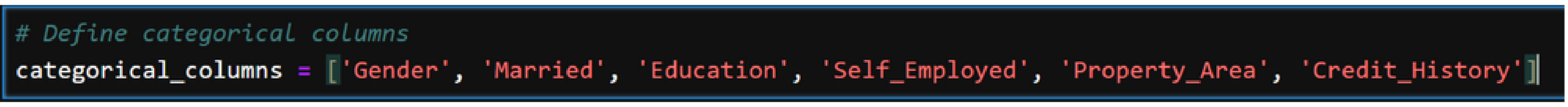


```
# Box Plots for numeric columns
for column in numeric_columns:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=df[column], color='red', showfliers=False)
    plt.title(f'Box Plot of {column} (Top 50)')
    plt.xlabel(column)
    plt.show()
```

- One of the example of Boxplot Graph



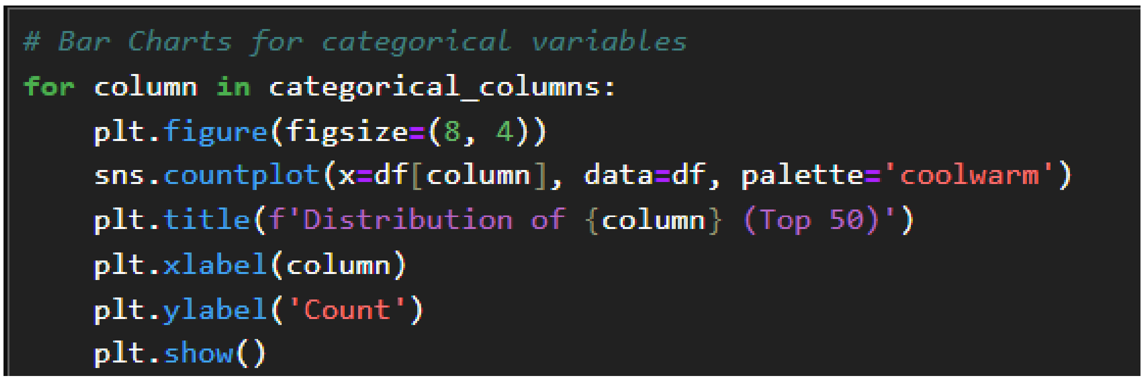
- This code defines categorical columns in the dataset, such as gender, marital status, education level, employment type, property area, and credit history. These variables are analyzed to understand their distribution and potential impact on loan approval decisions.



```
# Define categorical columns
categorical_columns = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area', 'Credit_History']
```

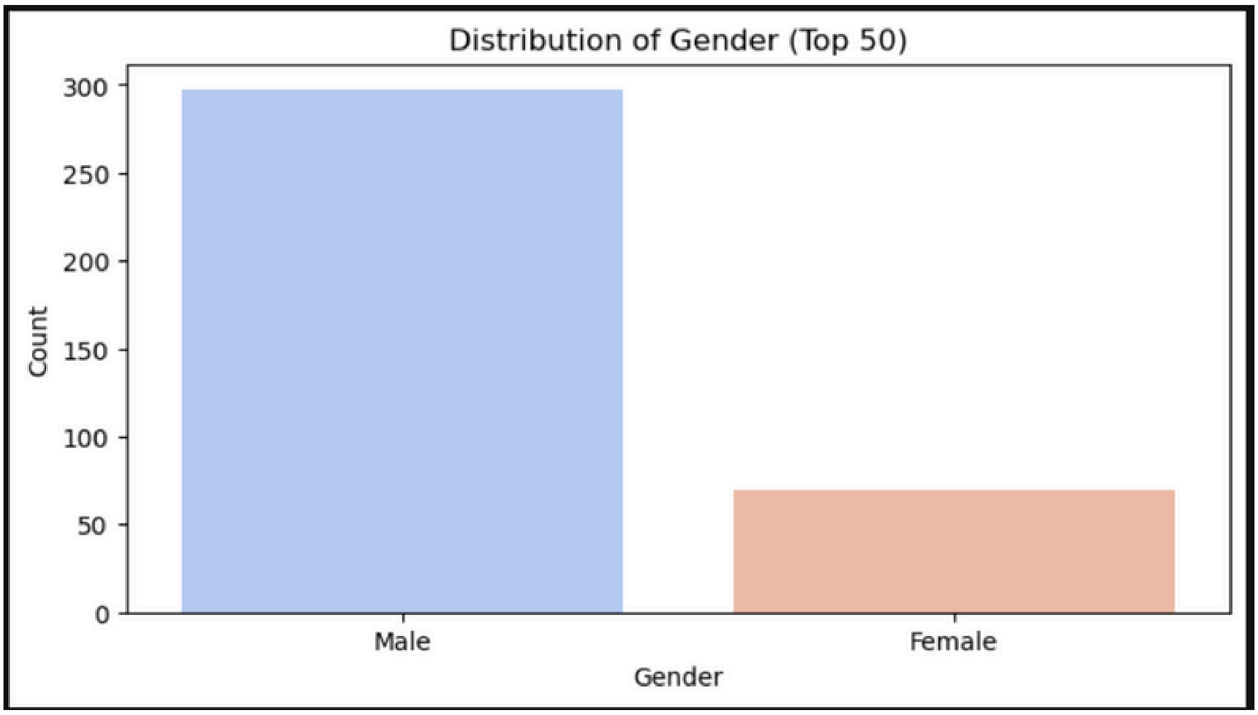
This code generates bar charts to visualize the distribution of each categorical variable. Key points include:

- **Visualization**: Uses count plots to display the frequency of each category, aiding in understanding variable distribution.
- **Aesthetics**: Sets a color palette (coolwarm) and adjusts the plot size for better readability.
- **Labeling**: Adds titles, x-axis, and y-axis labels for clarity.
- **Purpose**: Helps identify dominant categories and potential imbalances in categorical features, aiding data-driven insights.



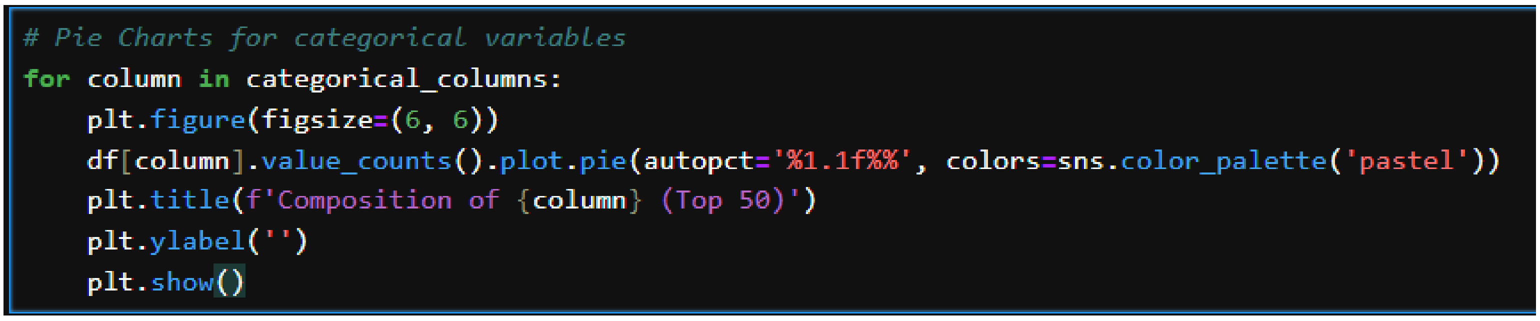
```
# Bar Charts for categorical variables
for column in categorical_columns:
    plt.figure(figsize=(8, 4))
    sns.countplot(x=df[column], data=df, palette='coolwarm')
    plt.title(f'Distribution of {column} (Top 50)')
    plt.xlabel(column)
    plt.ylabel('Count')
    plt.show()
```

One of the Bar chart’s Example



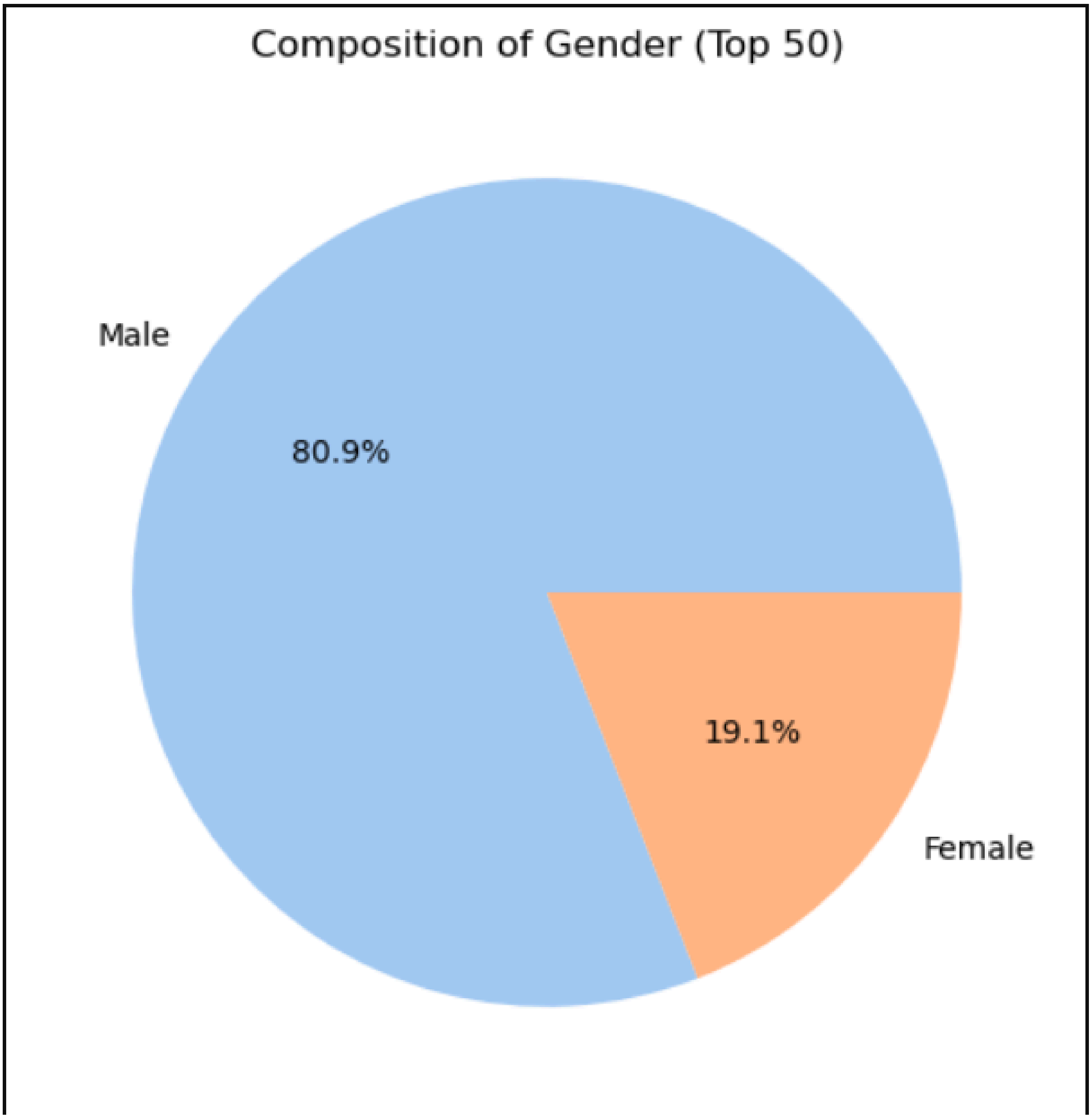
This code generates pie charts for each categorical variable to visualize their composition. Key points include:

- **Visualization**: Displays the percentage distribution of each category, providing a clear view of category proportions.
- **Aesthetics**: Uses a pastel color palette for better visual appeal.
- **Labeling**: Shows percentages (autopct='%1.1f%%') for more precise interpretation while removing y-axis labels for a cleaner look.
- **Purpose**: Helps identify dominant categories, potential imbalances, and distribution patterns, aiding in data-driven insights.



```
# Pie Charts for categorical variables
for column in categorical_columns:
    plt.figure(figsize=(6, 6))
    df[column].value_counts().plot.pie(autopct='%1.1f%%', colors=sns.color_palette('pastel'))
    plt.title(f'Composition of {column} (Top 50)')
    plt.ylabel('')
    plt.show()
```

One of the Pie Chart's Example



Bivariate Analysis

- **Scatter Plots for numeric variables**

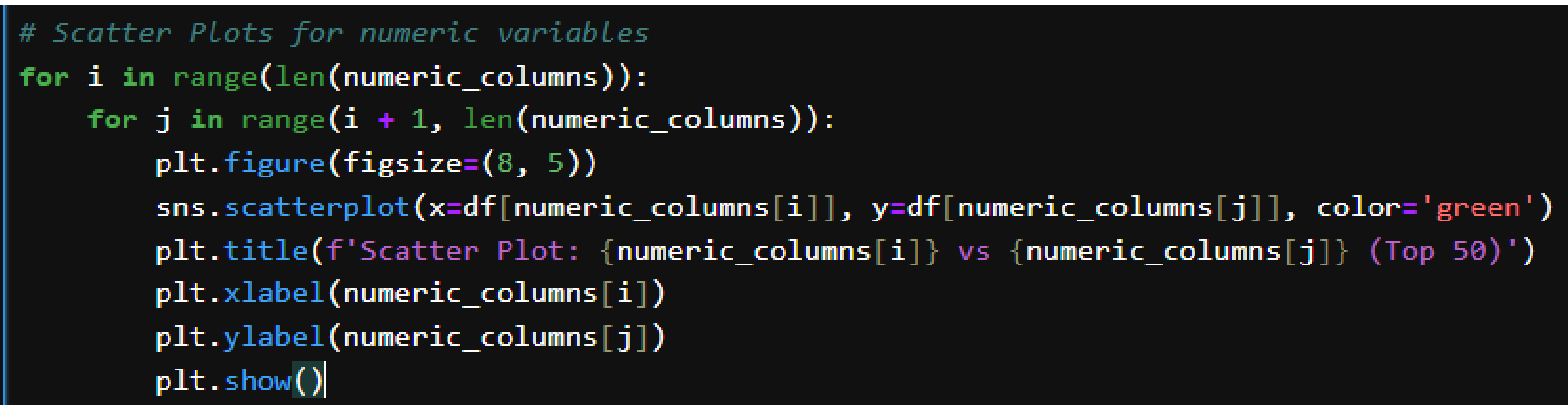
This code generates pairwise scatter plots for each unique combination of numeric columns from a DataFrame, df. The purpose is to visually examine relationships and potential correlations between the selected numeric features. Here’s a breakdown for the report:

- **Iterating Over Numeric Columns**:The outer loop (i) iterates through each numeric column, while the inner loop (j) compares it with the subsequent columns (i + 1 onwards). This avoids redundant plots (like plotting A vs. B and B vs. A).

- **Plotting Scatter Plots**:For each unique column pair, a scatter plot is created using Seaborn’s scatterplot function. The plots are sized at 8x5 inches for readability, and the data points are marked in green.

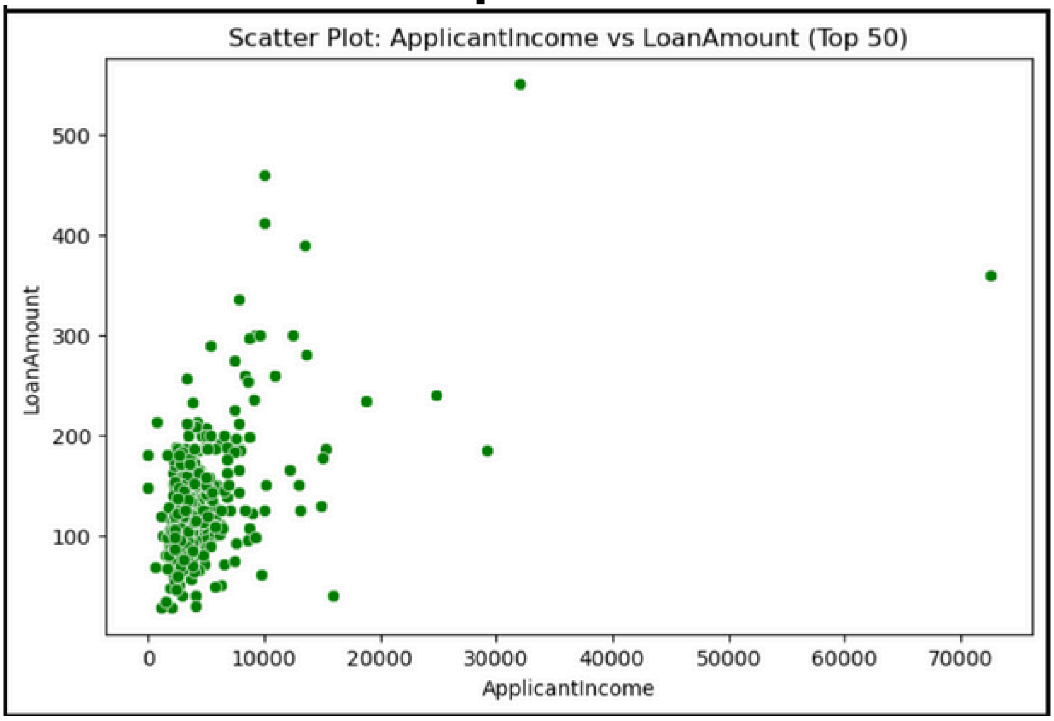
- **Title and Labels**:Each scatter plot is titled with the specific column pair being compared, following the format: "Scatter Plot: Column A vs Column B (Top 50)". The axes are appropriately labeled for clarity.

- **Analysis Purpose**:These scatter plots are valuable for detecting linear or non-linear relationships, potential outliers, and multicollinearity between numeric variables. They can guide further statistical analysis, like correlation calculations or regression modeling.



```python
# Scatter Plots for numeric variables
for i in range(len(numeric_columns)):
    for j in range(i + 1, len(numeric_columns)):
        plt.figure(figsize=(8, 5))
        sns.scatterplot(x=df[numeric_columns[i]], y=df[numeric_columns[j]], color='green')
        plt.title(f'Scatter Plot: {numeric_columns[i]} vs {numeric_columns[j]} (Top 50)')
        plt.xlabel(numeric_columns[i])
        plt.ylabel(numeric_columns[j])
        plt.show()
```

One of the Example of Scatter Plot Graph

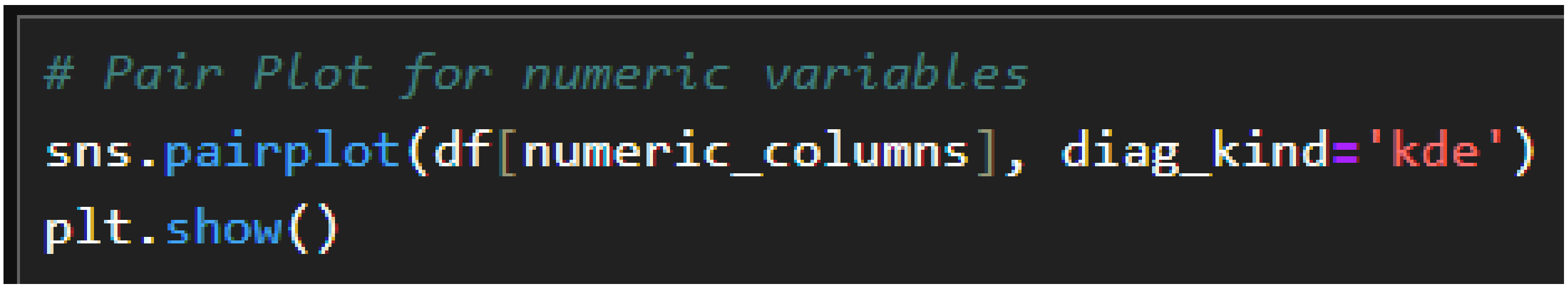


Pair Plot for numeric variables

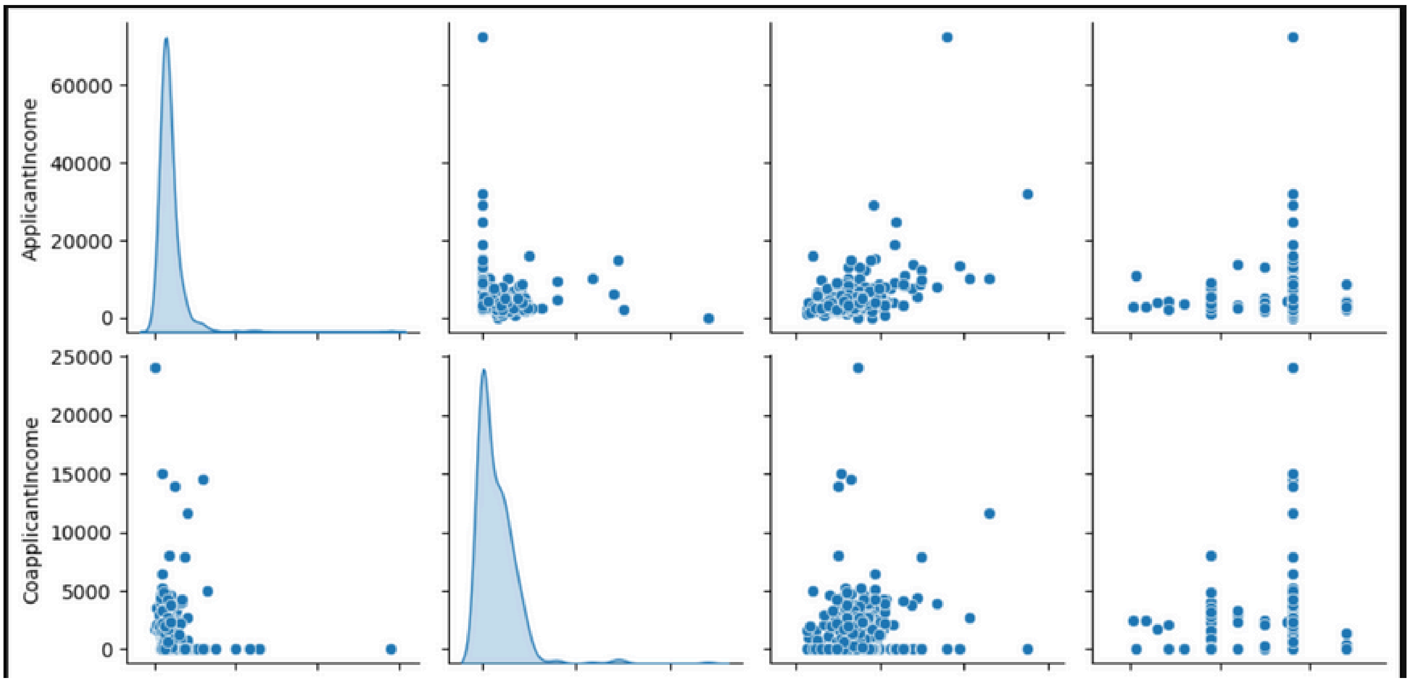
This code creates a pair plot for all numeric columns in the DataFrame df using Seaborn's pairplot() function. The scatter plots show pairwise relationships between variables, while the diagonal displays kernel density estimates (KDE) to illustrate the distribution of each numeric feature.

Purpose:

- **Correlation Analysis**: Identifies relationships between numeric variables.
- **Distribution Insight**: KDE on the diagonal shows each variable's distribution.
- **Outlier Detection**: Helps spot unusual data points.

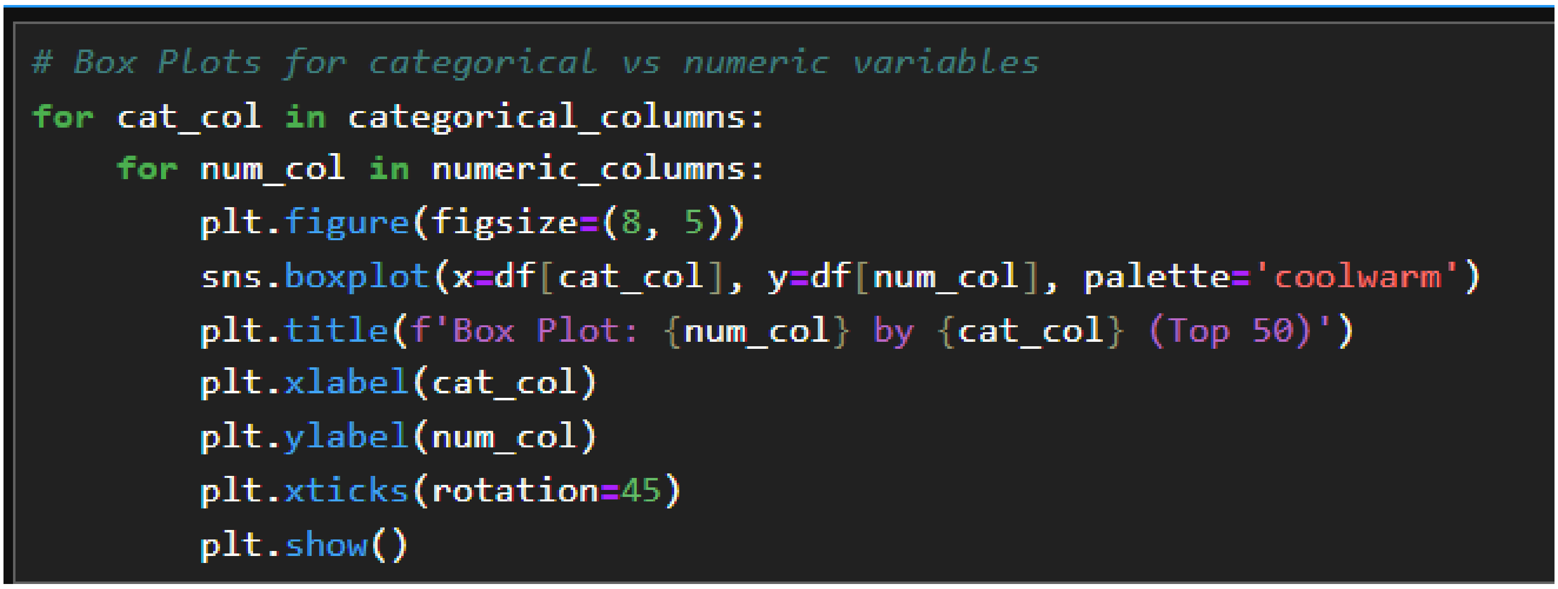


```python
# Pair Plot for numeric variables
sns.pairplot(df[numeric_columns], diag_kind='kde')
plt.show()
```



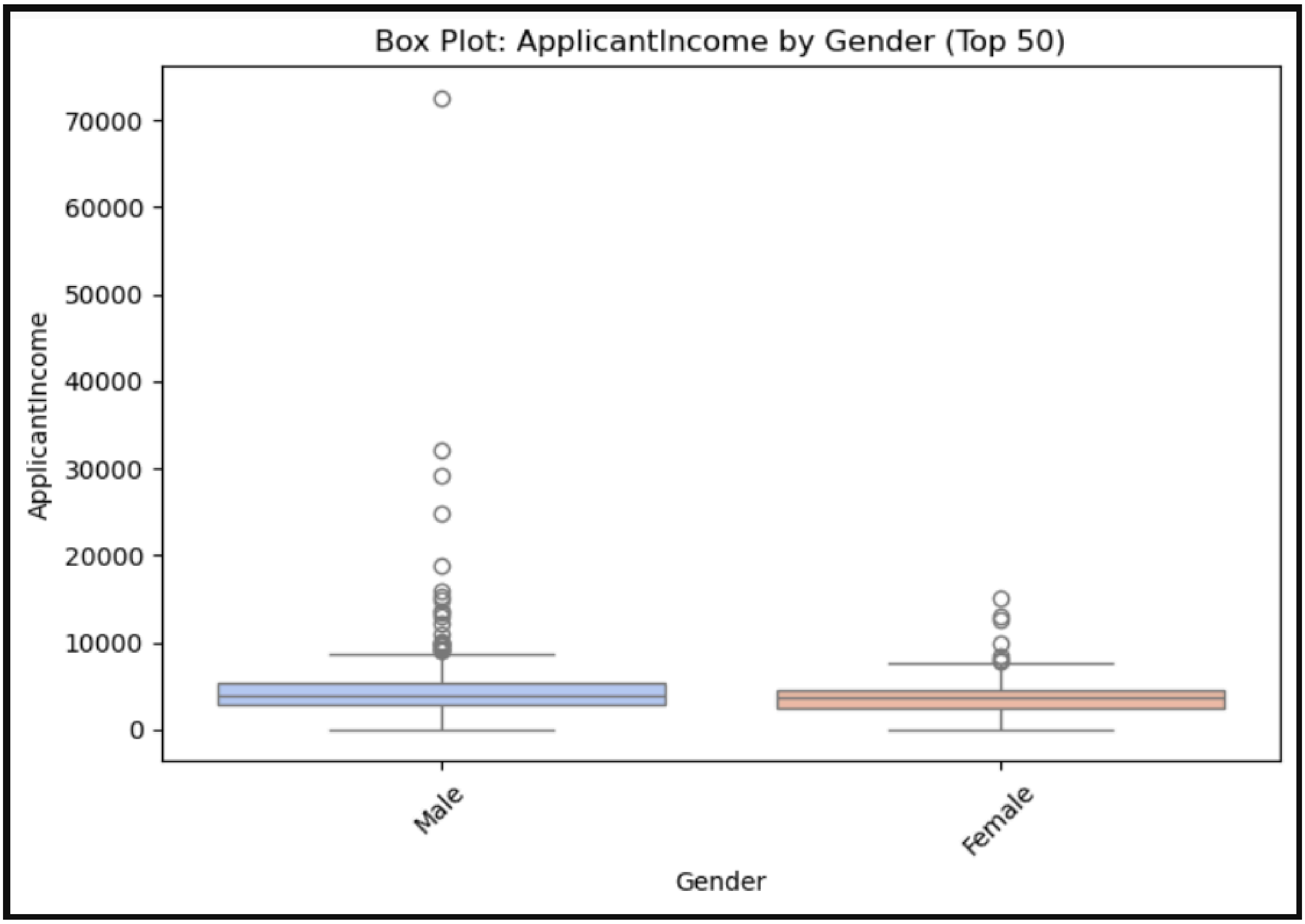
Box Plots for categorical vs numeric variables

This code generates box plots to visually compare the distribution of numeric variables across different categories in the dataset. It iterates through each combination of categorical and numeric columns, creating a separate plot for each pair. The box plots use a 'coolwarm' color palette and display the top 50 data points, providing insights into the spread, central tendency, and potential outliers for each category. The plots are formatted with rotated x-axis labels for better readability.



```python
# Box Plots for categorical vs numeric variables
for cat_col in categorical_columns:
    for num_col in numeric_columns:
        plt.figure(figsize=(8, 5))
        sns.boxplot(x=df[cat_col], y=df[num_col], palette='coolwarm')
        plt.title(f'Box Plot: {num_col} by {cat_col} (Top 50)')
        plt.xlabel(cat_col)
        plt.ylabel(num_col)
        plt.xticks(rotation=45)
        plt.show()
```

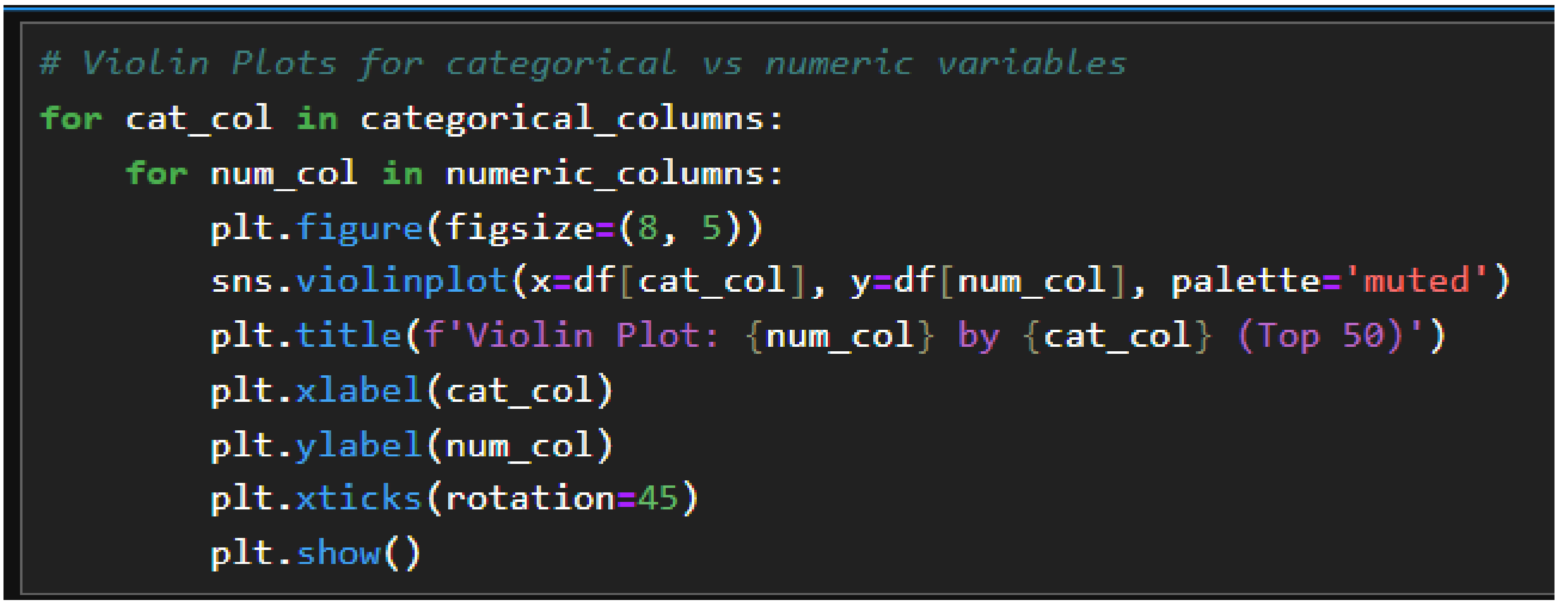
One of the Example of Box Plot



Violin Plots for categorical vs numeric variables

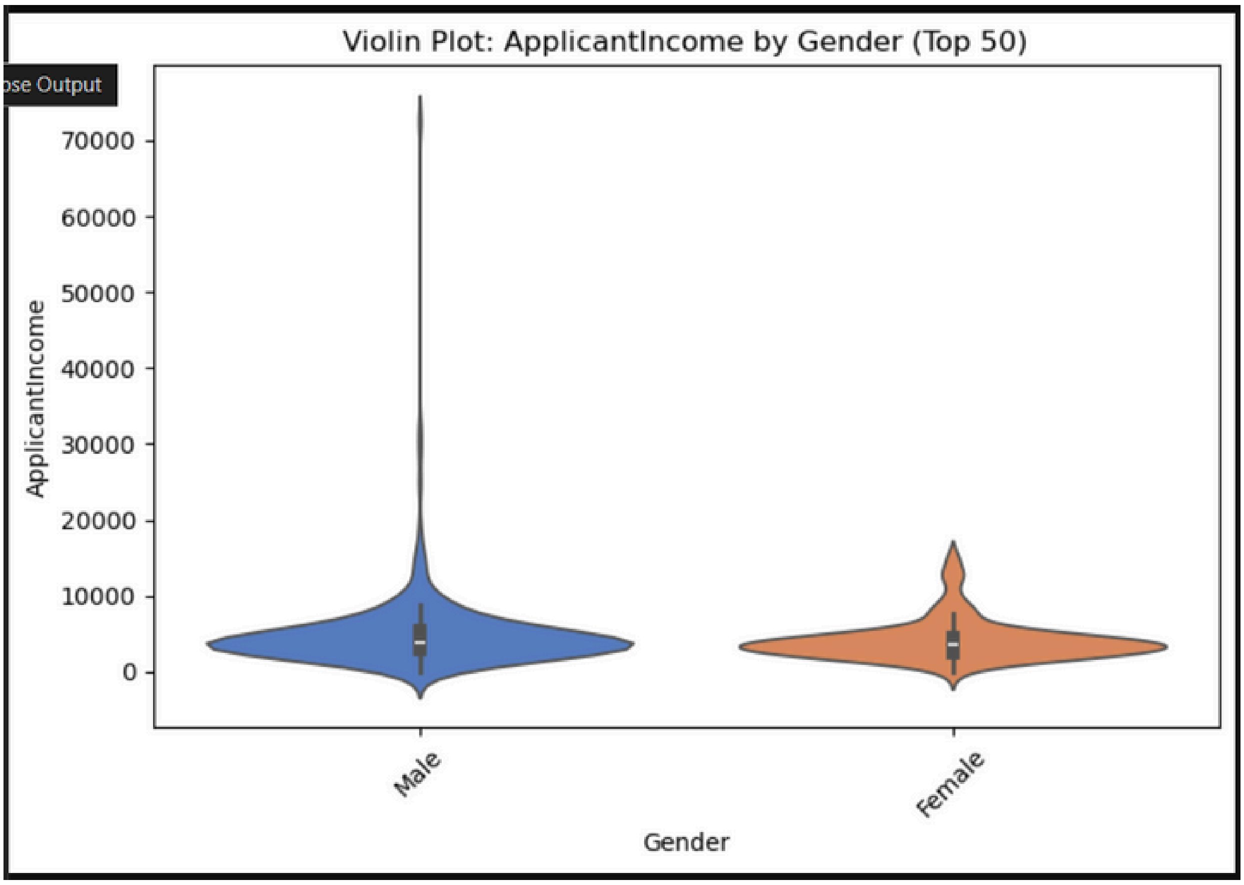
This code generates violin plots to analyze the distribution of numeric variables across different categories in the dataset. It loops through each combination of categorical and numeric columns, creating a plot for each pair

The violin plots use a 'muted' color palette and showcase the top 50 data points, providing insights into the distribution's shape, density, and variability within each category. The x-axis labels are rotated for better readability.



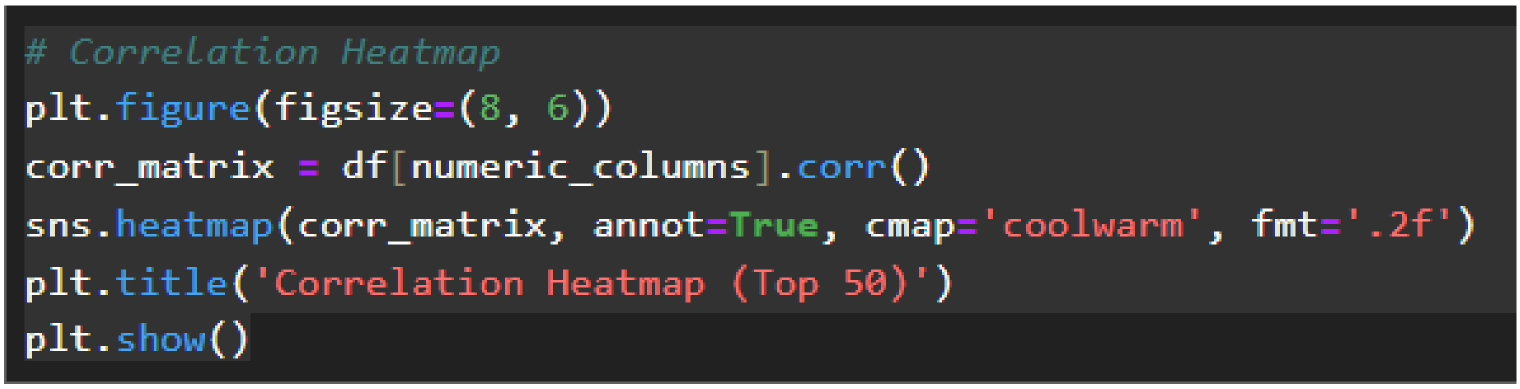
```
# Violin Plots for categorical vs numeric variables
for cat_col in categorical_columns:
    for num_col in numeric_columns:
        plt.figure(figsize=(8, 5))
        sns.violinplot(x=df[cat_col], y=df[num_col], palette='muted')
        plt.title(f'Violin Plot: {num_col} by {cat_col} (Top 50)')
        plt.xlabel(cat_col)
        plt.ylabel(num_col)
        plt.xticks(rotation=45)
        plt.show()
```

One of the example of voilen plot



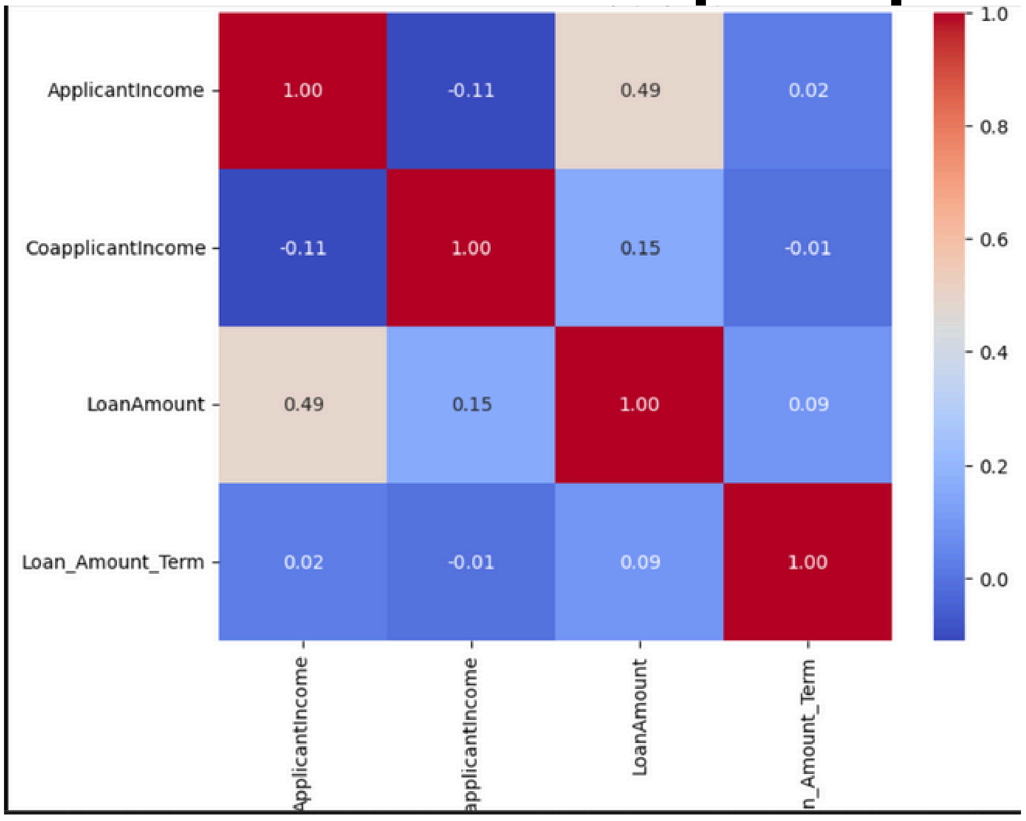
Correlation Heatmap

This code generates a correlation heatmap to analyze the relationships between numeric variables in the dataset. It calculates the correlation matrix for the specified numeric columns and displays it using a 'coolwarm' color palette. The heatmap includes correlation values annotated within the cells (formatted to two decimal places) for clarity. This visualization helps identify strong positive or negative correlations, aiding in understanding the interdependencies between variables.



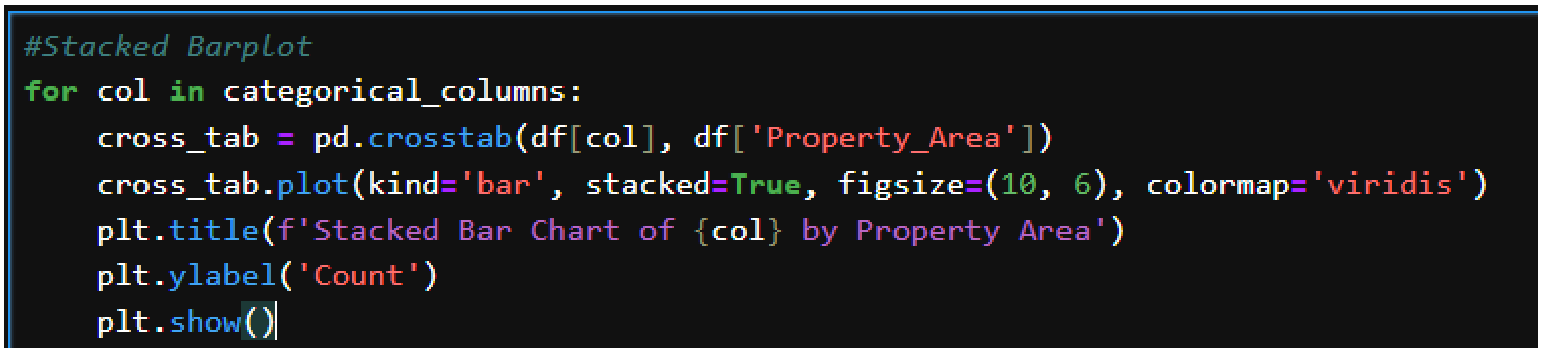
```python
# Correlation Heatmap
plt.figure(figsize=(8, 6))
corr_matrix = df[numeric_columns].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap (Top 50)')
plt.show()
```

Correlation Heatmap Graph



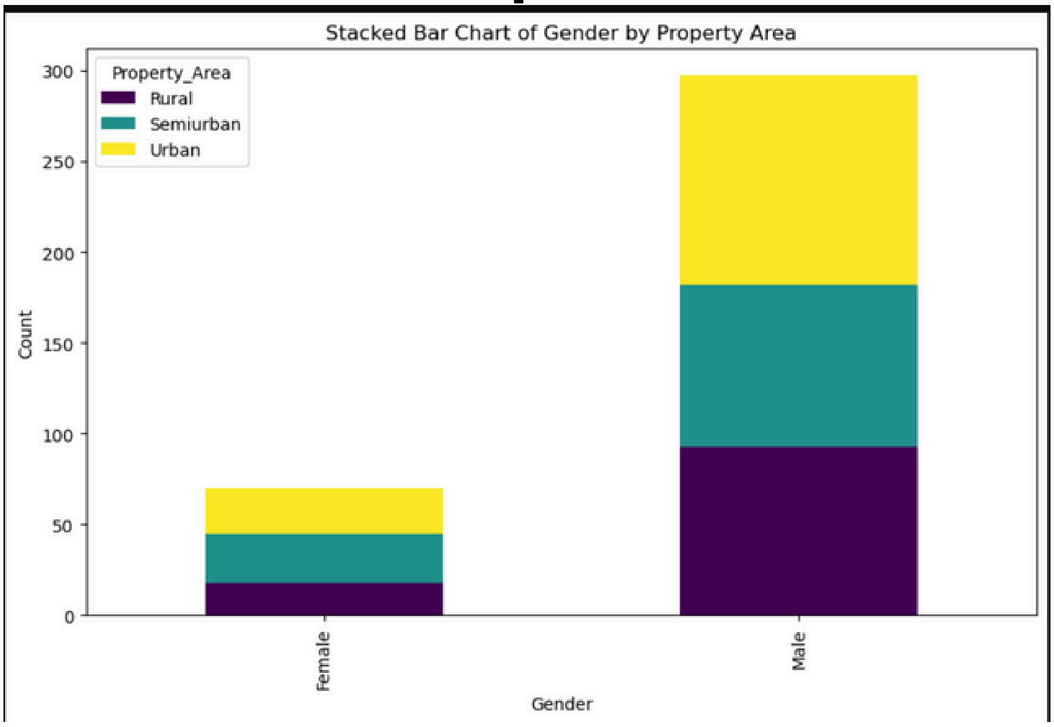
Stacked Barplot

This code generates stacked bar charts to examine the distribution of categorical variables across different property areas. It iterates over each categorical column and creates a cross-tabulation with the "Property_Area" column to count occurrences. The stacked bar charts use a 'viridis' color palette and help visualize the composition of each category within property areas. This approach aids in identifying patterns and relationships between categorical features and the property area.



```
#Stacked Barplot
for col in categorical_columns:
    cross_tab = pd.crosstab(df[col], df['Property_Area'])
    cross_tab.plot(kind='bar', stacked=True, figsize=(10, 6), colormap='viridis')
    plt.title(f'Stacked Bar Chart of {col} by Property Area')
    plt.ylabel('Count')
    plt.show()
```

One of the Exapmle of Stacked Bar Plot



The notebook includes various analyses, such as:

- **Box Plots**: For examining numeric columns and comparing numeric vs. categorical variables.
- **Violin Plots**: To visualize the distribution and density of numeric variables across categorical groups.
- **Correlation Heatmap**: To assess relationships between numeric variables.
- **Stacked Bar Charts**: To analyze categorical variables concerning the "Property_Area."

Summary for the Data Analyst Report:

Purpose: The analysis aims to explore the dataset to understand the distribution of loan-related attributes, identify correlations, and examine relationships between categorical and numeric features. This helps in identifying patterns that may influence loan approval decisions.

Tools Used:

- **Pandas:** Data manipulation and preprocessing.
- **Matplotlib & Seaborn:** Visualization of distributions, correlations, and category comparisons.

Key Insights:

- **Distribution Analysis:** Box and violin plots highlight the distribution, spread, and outliers in numeric data across categories.
- **Correlation Analysis:** The heatmap reveals correlations between variables like applicant income, coapplicant income, and loan amount, aiding in multicollinearity checks.
- **Categorical Analysis:** Stacked bar charts demonstrate the distribution of categorical features like gender, education, and marital status across property areas.
- **Missing Values:** Notable missing data in columns like "Gender," "Self_Employed," and "Credit_History" that may need imputation or handling.