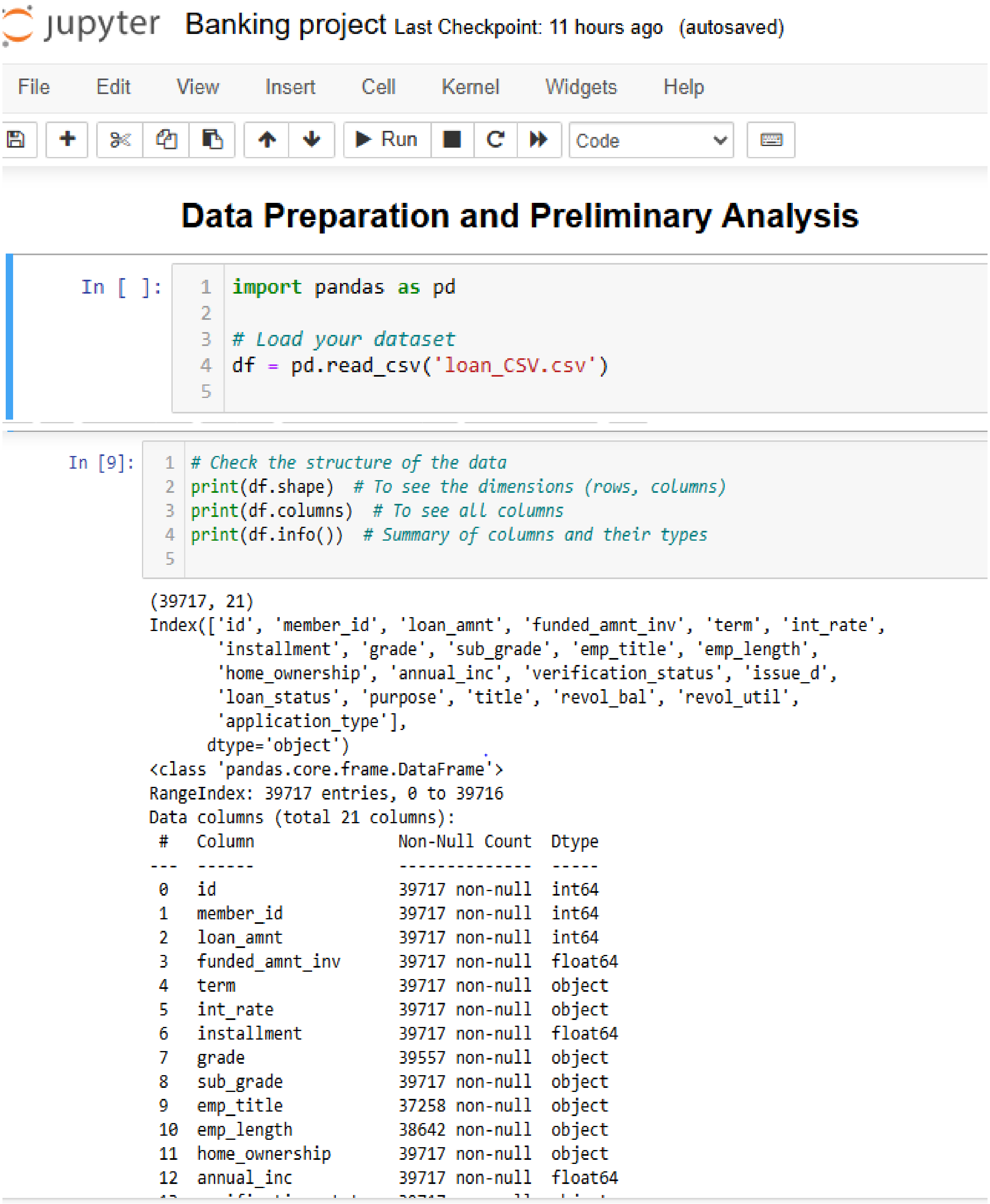
BANKING –CAPSTONE PROJECT



Data Preparation and Preliminary Analysis

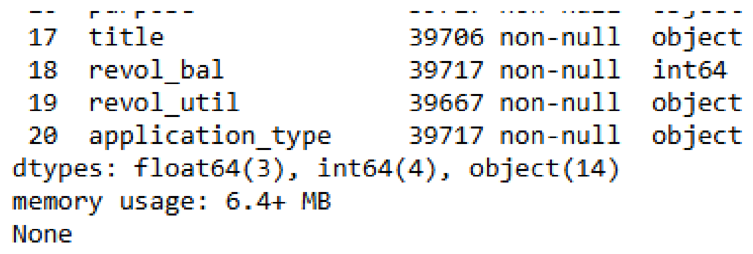
In [ ]:

```
import pandas as pd

# Load your dataset
df = pd.read_csv('loan_CSV.csv')

```

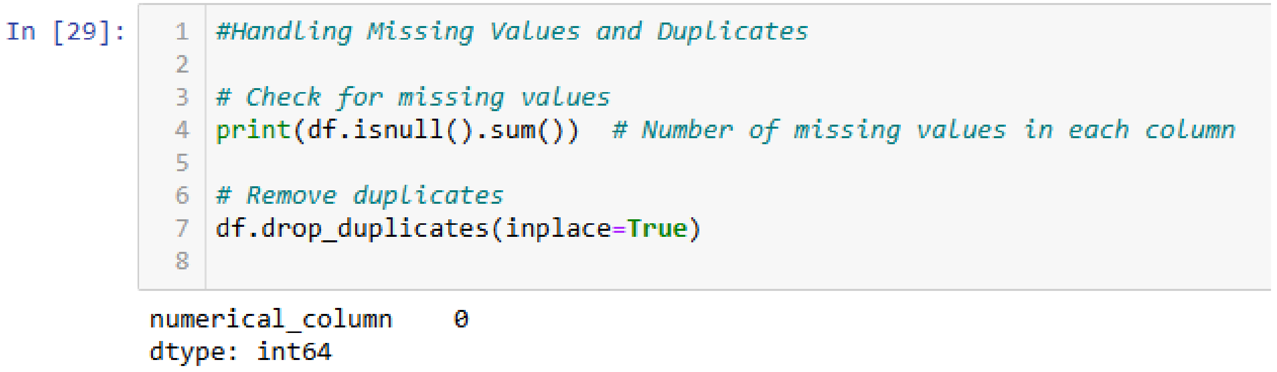
In [9]:

```
# Check the structure of the data
print(df.shape)  # To see the dimensions (rows, columns)
print(df.columns)  # To see all columns
print(df.info())  # Summary of columns and their types

```

```
(39717, 21)
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'home_ownership', 'annual_inc', 'verification_status', 'issue_d',
       'loan_status', 'purpose', 'title', 'revol_bal', 'revol_util',
       'application_type'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 21 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   id                   39717 non-null  int64  
 1   member_id            39717 non-null  int64  
 2   loan_amnt            39717 non-null  int64  
 3   funded_amnt_inv      39717 non-null  float64
 4   term                 39717 non-null  object 
 5   int_rate             39717 non-null  object 
 6   installment          39717 non-null  float64
 7   grade                39557 non-null  object 
 8   sub_grade            39717 non-null  object 
 9   emp_title            37258 non-null  object 
 10  emp_length           38642 non-null  object 
 11  home_ownership       39717 non-null  object 
 12  annual_inc           39717 non-null  float64
 13  verification_status  39717 non-null  object 
```



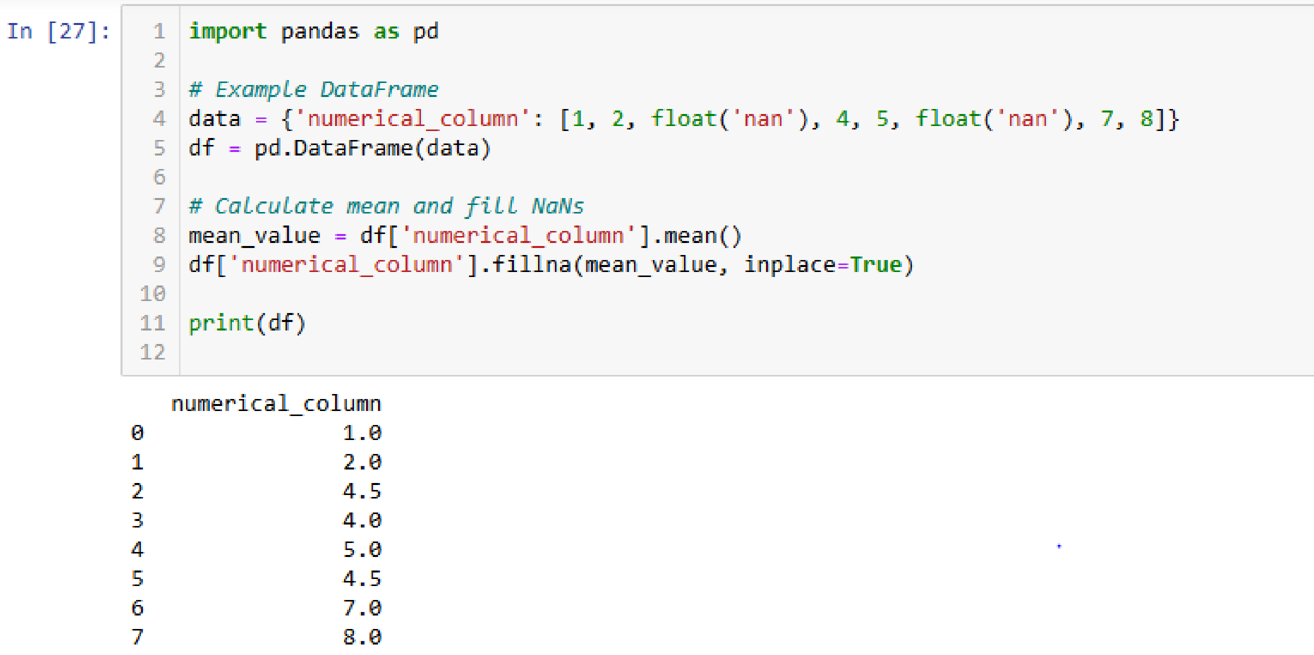
```
 17  title                 39706 non-null  object
 18  revol_bal             39717 non-null  int64
 19  revol_util            39667 non-null  object
 20  application_type      39717 non-null  object
dtypes: float64(3), int64(4), object(14)
memory usage: 6.4+ MB
None
```



In [29]:

```python
#Handling Missing Values and Duplicates

# Check for missing values
print(df.isnull().sum())  # Number of missing values in each column

# Remove duplicates
df.drop_duplicates(inplace=True)

```

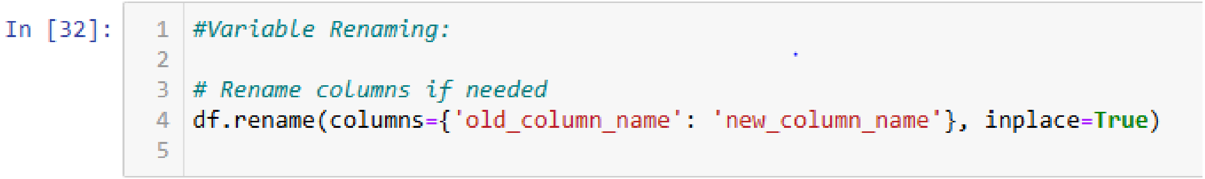
```
numerical_column    0
dtype: int64
```



In [27]:

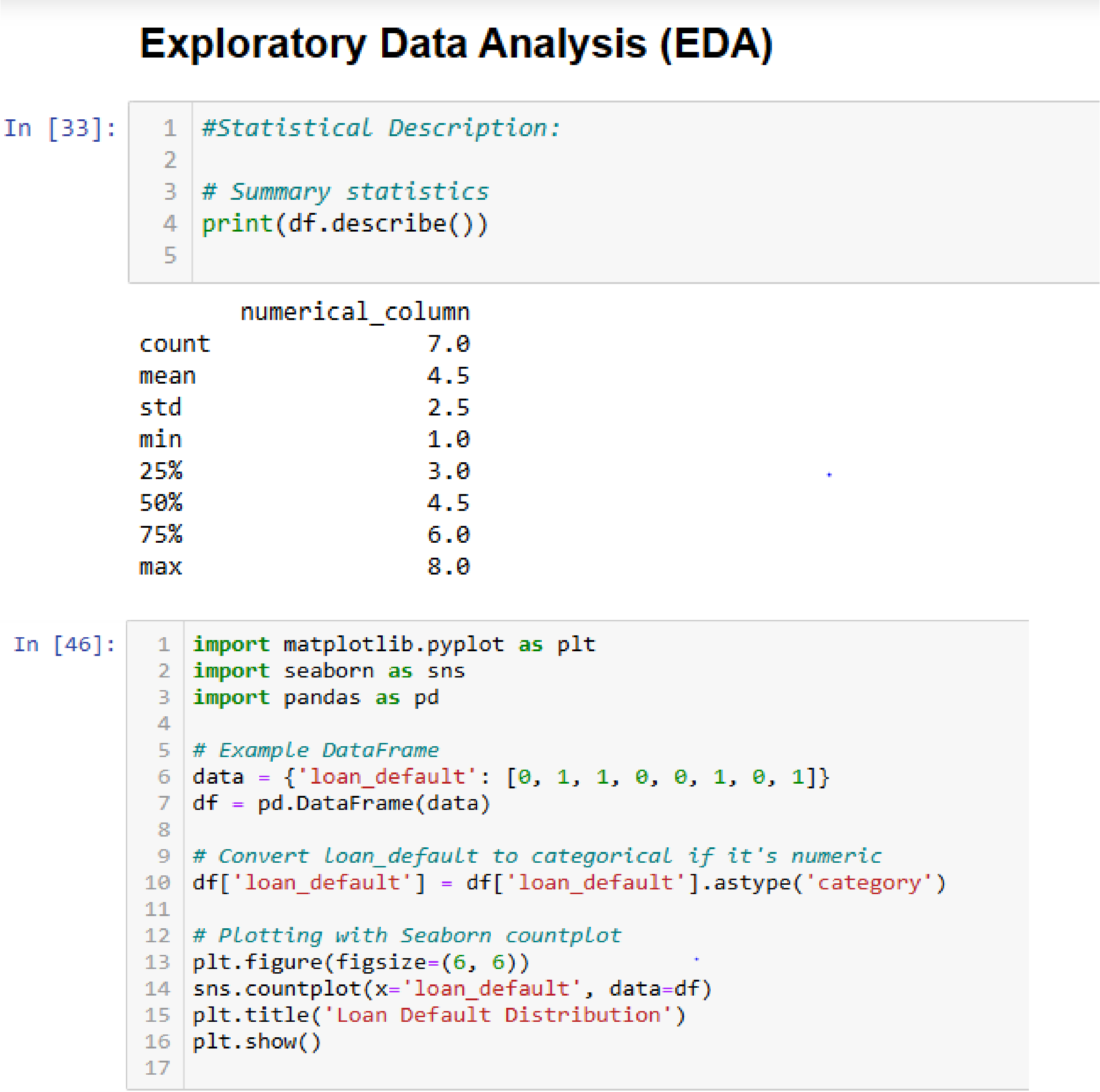
```python
import pandas as pd

# Example DataFrame
data = {'numerical_column': [1, 2, float('nan'), 4, 5, float('nan'), 7, 8]}
df = pd.DataFrame(data)

# Calculate mean and fill NaNs
mean_value = df['numerical_column'].mean()
df['numerical_column'].fillna(mean_value, inplace=True)

print(df)

```

```
   numerical_column
0               1.0
1               2.0
2               4.5
3               4.0
4               5.0
5               4.5
6               7.0
7               8.0
```



In [32]:

```python
#Variable Renaming:

# Rename columns if needed
df.rename(columns={'old_column_name': 'new_column_name'}, inplace=True)

```



Exploratory Data Analysis (EDA)

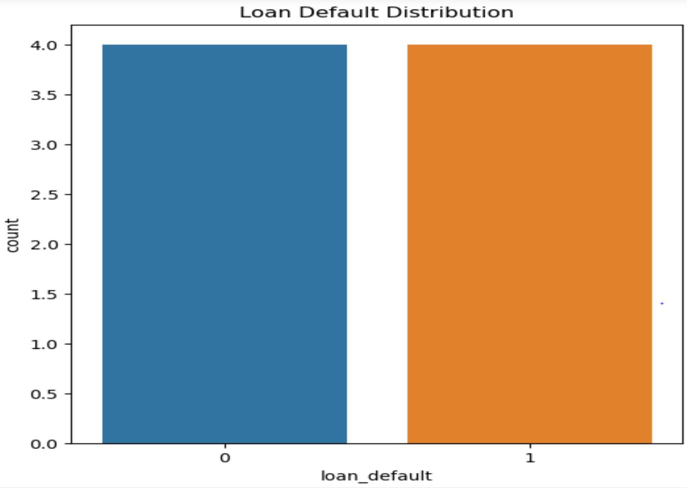
In [33]:

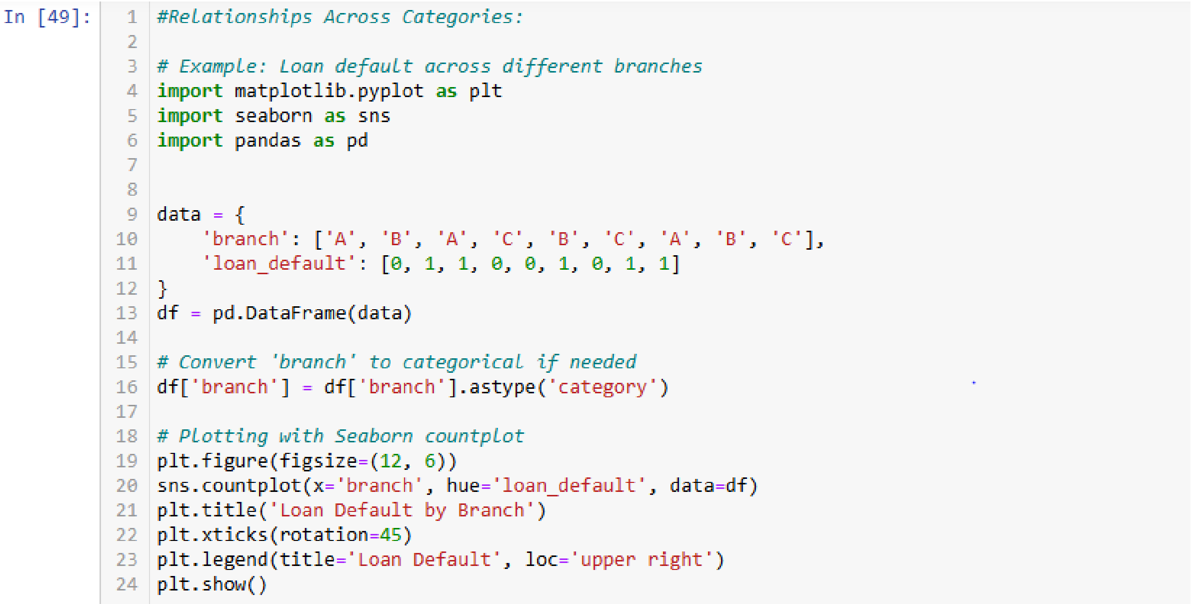
```
#Statistical Description:

# Summary statistics
print(df.describe())

```

```
       numerical_column
count               7.0
mean                4.5
std                 2.5
min                 1.0
25%                 3.0
50%                 4.5
75%                 6.0
max                 8.0
```

In [46]:

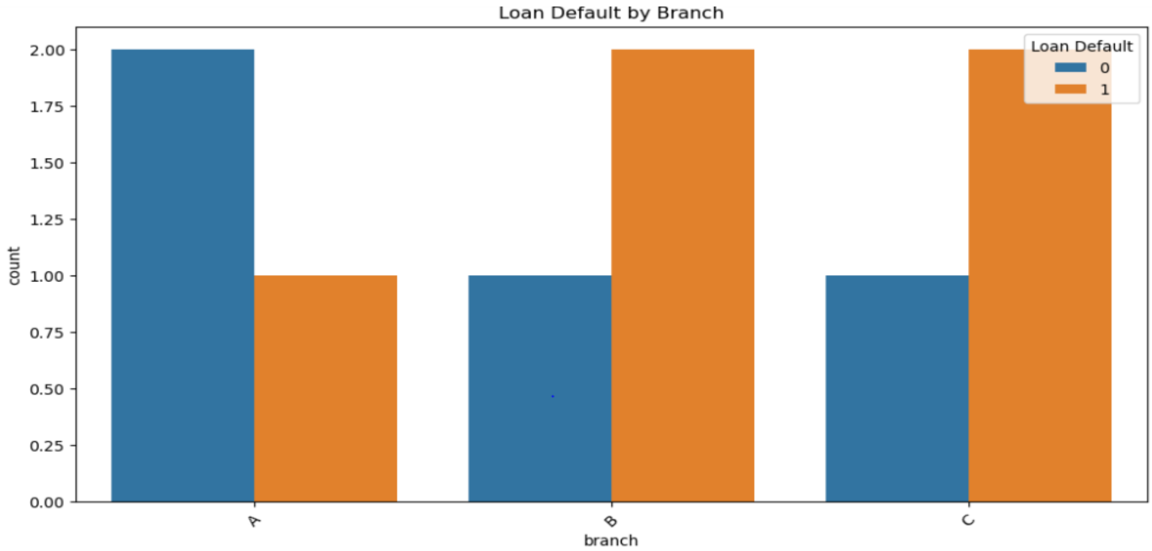
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Example DataFrame
data = {'loan_default': [0, 1, 1, 0, 0, 1, 0, 1]}
df = pd.DataFrame(data)

# Convert loan_default to categorical if it's numeric
df['loan_default'] = df['loan_default'].astype('category')

# Plotting with Seaborn countplot
plt.figure(figsize=(6, 6))
sns.countplot(x='loan_default', data=df)
plt.title('Loan Default Distribution')
plt.show()

```

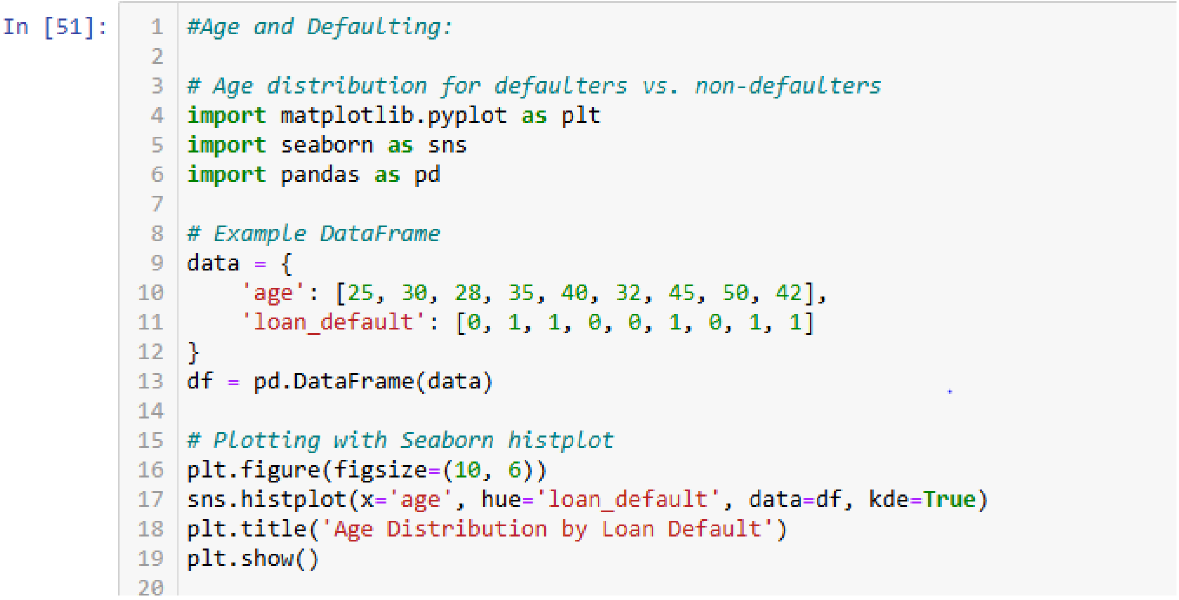




In [49]:

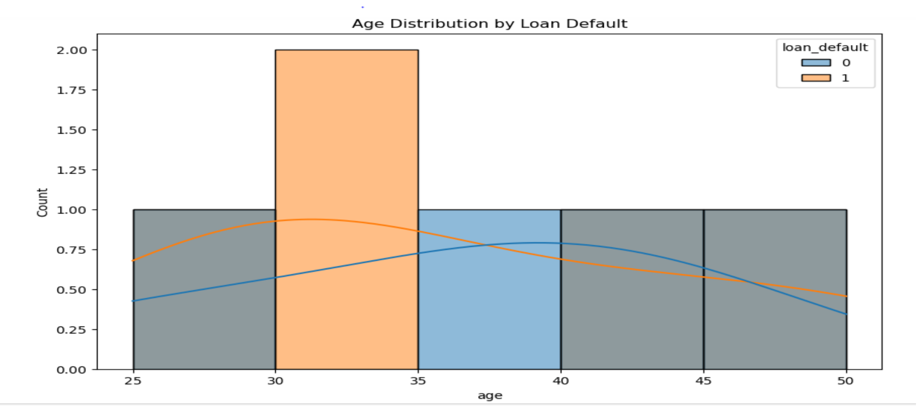
```
#Relationships Across Categories:

# Example: Loan default across different branches
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd


data = {
    'branch': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],
    'loan_default': [0, 1, 1, 0, 0, 1, 0, 1, 1]
}
df = pd.DataFrame(data)

# Convert 'branch' to categorical if needed
df['branch'] = df['branch'].astype('category')

# Plotting with Seaborn countplot
plt.figure(figsize=(12, 6))
sns.countplot(x='branch', hue='loan_default', data=df)
plt.title('Loan Default by Branch')
plt.xticks(rotation=45)
plt.legend(title='Loan Default', loc='upper right')
plt.show()
```

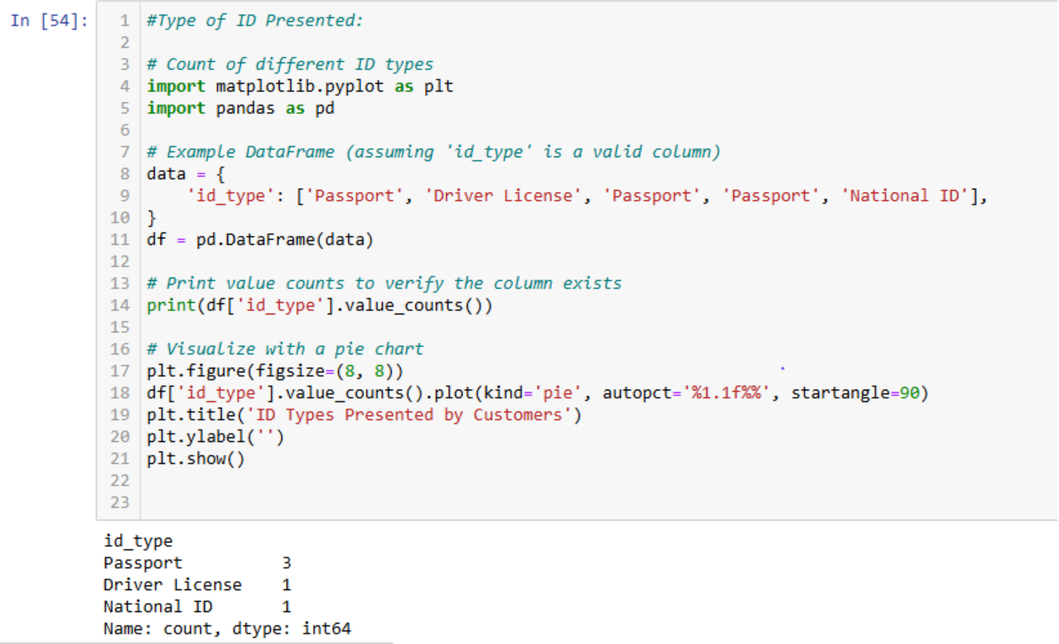




In [51]:

```
#Age and Defaulting:

# Age distribution for defaulters vs. non-defaulters
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Example DataFrame
data = {
    'age': [25, 30, 28, 35, 40, 32, 45, 50, 42],
    'loan_default': [0, 1, 1, 0, 0, 1, 0, 1, 1]
}
df = pd.DataFrame(data)

# Plotting with Seaborn histplot
plt.figure(figsize=(10, 6))
sns.histplot(x='age', hue='loan_default', data=df, kde=True)
plt.title('Age Distribution by Loan Default')
plt.show()

```

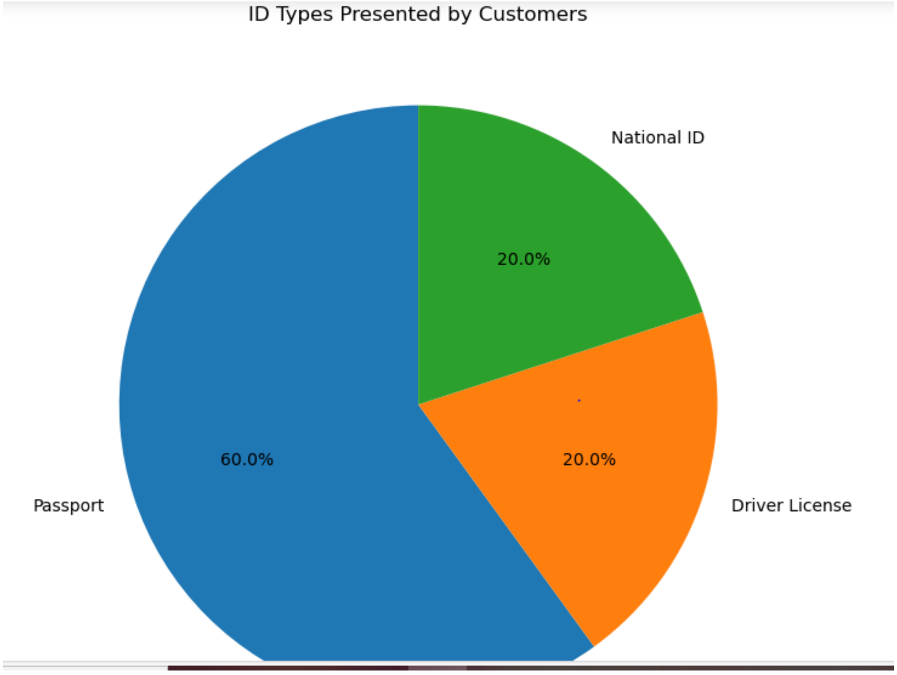


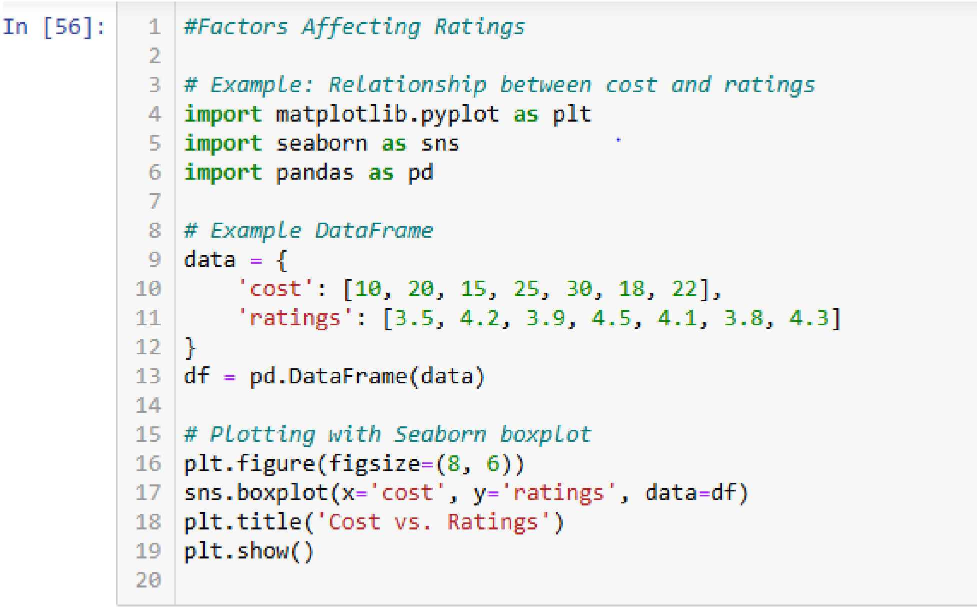


In [54]:

```python
#Type of ID Presented:

# Count of different ID types
import matplotlib.pyplot as plt
import pandas as pd

# Example DataFrame (assuming 'id_type' is a valid column)
data = {
    'id_type': ['Passport', 'Driver License', 'Passport', 'Passport', 'National ID'],
}
df = pd.DataFrame(data)

# Print value counts to verify the column exists
print(df['id_type'].value_counts())

# Visualize with a pie chart
plt.figure(figsize=(8, 8))
df['id_type'].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title('ID Types Presented by Customers')
plt.ylabel('')
plt.show()


```

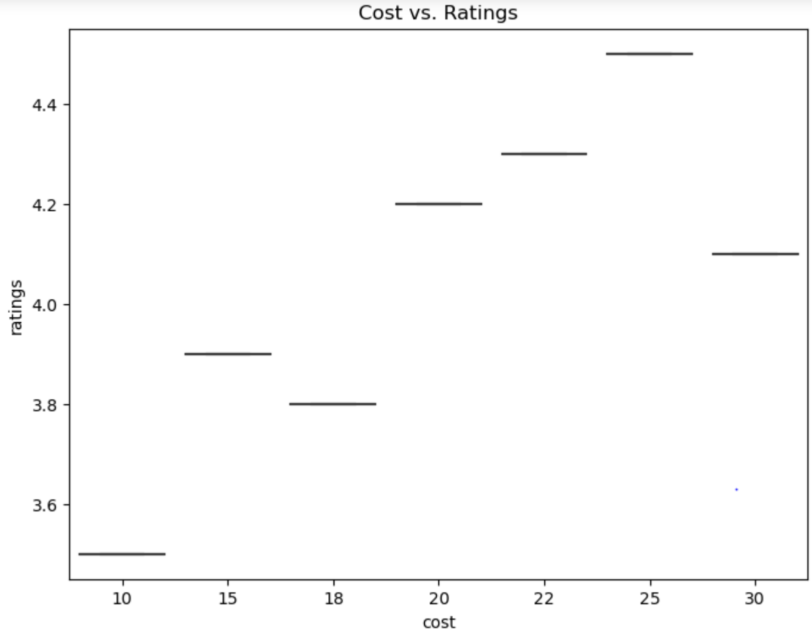
```
id_type
Passport          3
Driver License    1
National ID       1
Name: count, dtype: int64
```

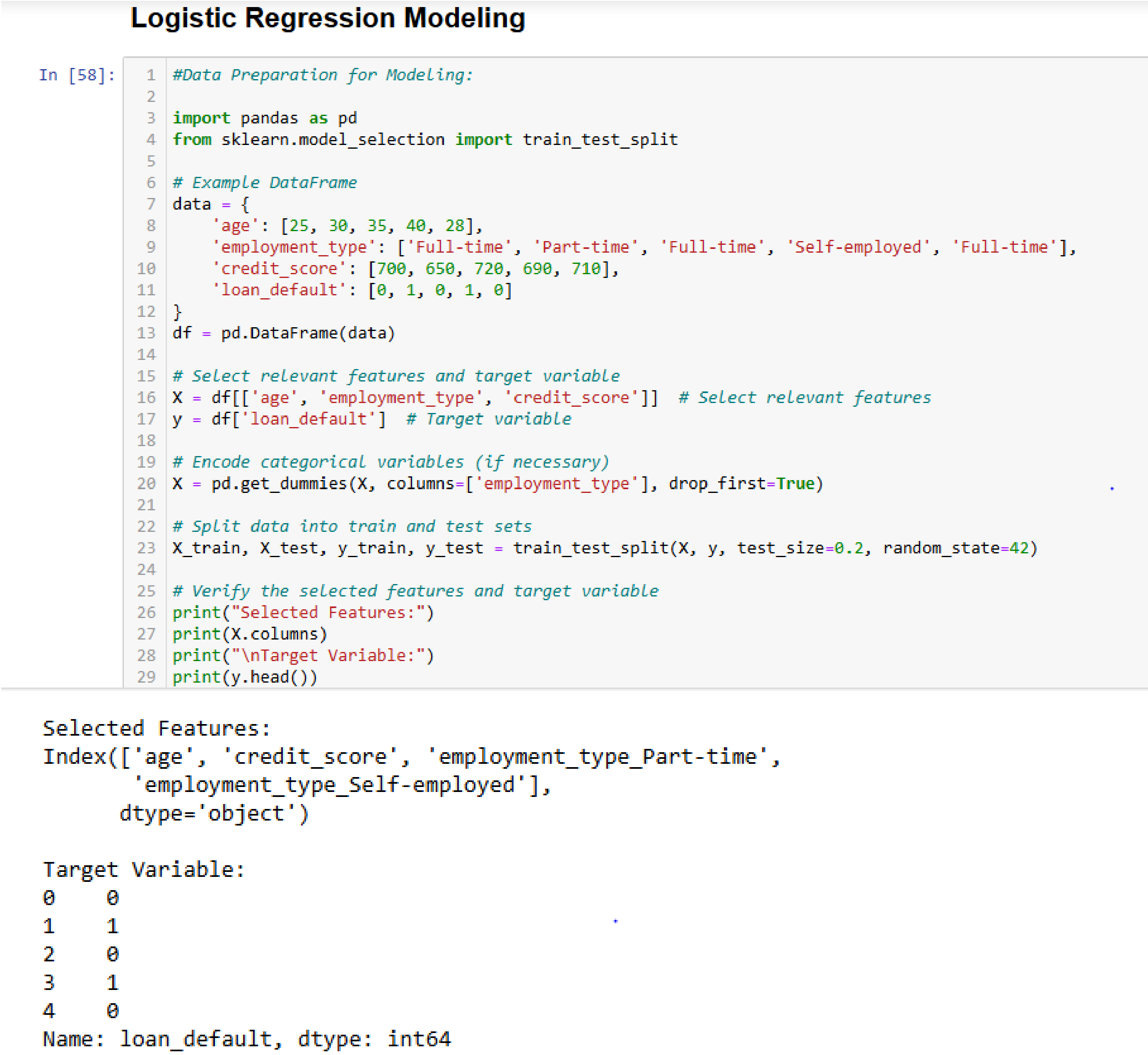




In [56]:

```python
#Factors Affecting Ratings

# Example: Relationship between cost and ratings
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Example DataFrame
data = {
    'cost': [10, 20, 15, 25, 30, 18, 22],
    'ratings': [3.5, 4.2, 3.9, 4.5, 4.1, 3.8, 4.3]
}
df = pd.DataFrame(data)

# Plotting with Seaborn boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x='cost', y='ratings', data=df)
plt.title('Cost vs. Ratings')
plt.show()

```



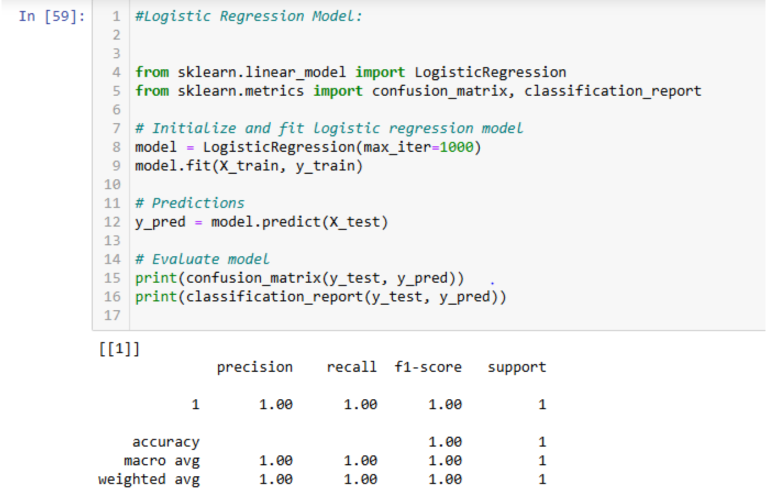


Logistic Regression Modeling

In [58]:

```python
#Data Preparation for Modeling:

import pandas as pd
from sklearn.model_selection import train_test_split

# Example DataFrame
data = {
    'age': [25, 30, 35, 40, 28],
    'employment_type': ['Full-time', 'Part-time', 'Full-time', 'Self-employed', 'Full-time'],
    'credit_score': [700, 650, 720, 690, 710],
    'loan_default': [0, 1, 0, 1, 0]
}
df = pd.DataFrame(data)

# Select relevant features and target variable
X = df[['age', 'employment_type', 'credit_score']]  # Select relevant features
y = df['loan_default']  # Target variable

# Encode categorical variables (if necessary)
X = pd.get_dummies(X, columns=['employment_type'], drop_first=True)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Verify the selected features and target variable
print("Selected Features:")
print(X.columns)
print("\nTarget Variable:")
print(y.head())
```

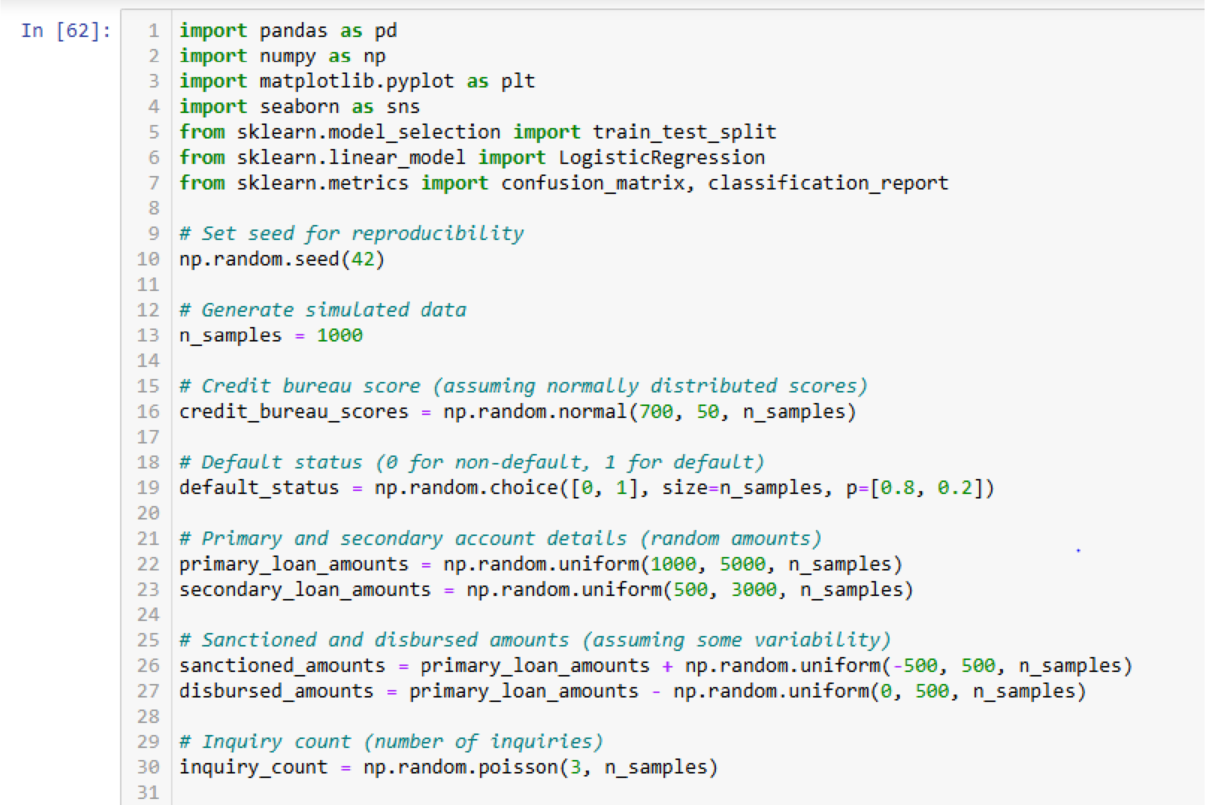
```
Selected Features:
Index(['age', 'credit_score', 'employment_type_Part-time',
       'employment_type_Self-employed'],
      dtype='object')

Target Variable:
0    0
1    1
2    0
3    1
4    0
Name: loan_default, dtype: int64
```



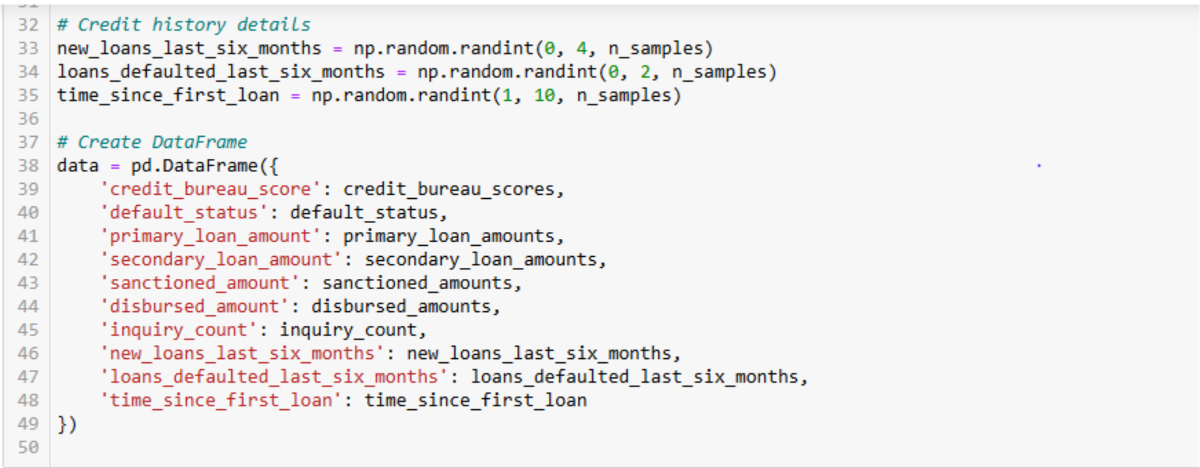
In [59]:

```python
#Logistic Regression Model:


from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report

# Initialize and fit logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate model
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

```

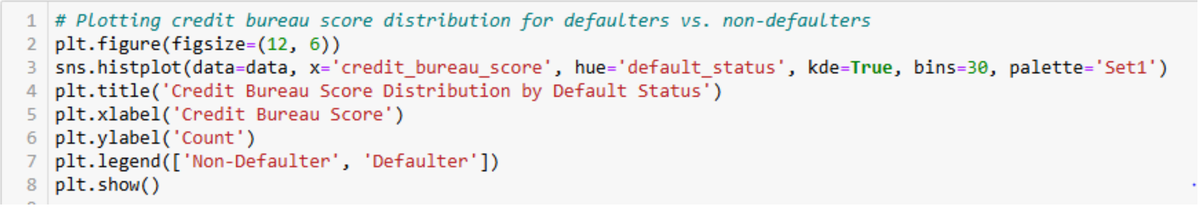
```
[[1]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1
```



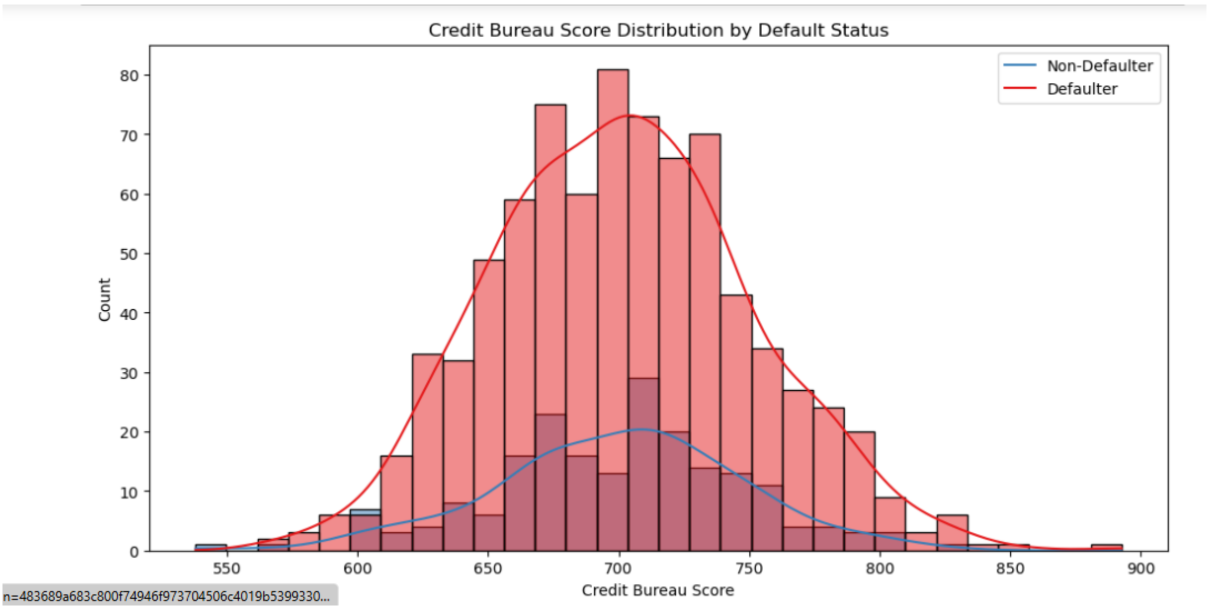
```
In [62]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report

# Set seed for reproducibility
np.random.seed(42)

# Generate simulated data
n_samples = 1000

# Credit bureau score (assuming normally distributed scores)
credit_bureau_scores = np.random.normal(700, 50, n_samples)

# Default status (0 for non-default, 1 for default)
default_status = np.random.choice([0, 1], size=n_samples, p=[0.8, 0.2])

# Primary and secondary account details (random amounts)
primary_loan_amounts = np.random.uniform(1000, 5000, n_samples)
secondary_loan_amounts = np.random.uniform(500, 3000, n_samples)

# Sanctioned and disbursed amounts (assuming some variability)
sanctioned_amounts = primary_loan_amounts + np.random.uniform(-500, 500, n_samples)
disbursed_amounts = primary_loan_amounts - np.random.uniform(0, 500, n_samples)

# Inquiry count (number of inquiries)
inquiry_count = np.random.poisson(3, n_samples)

```

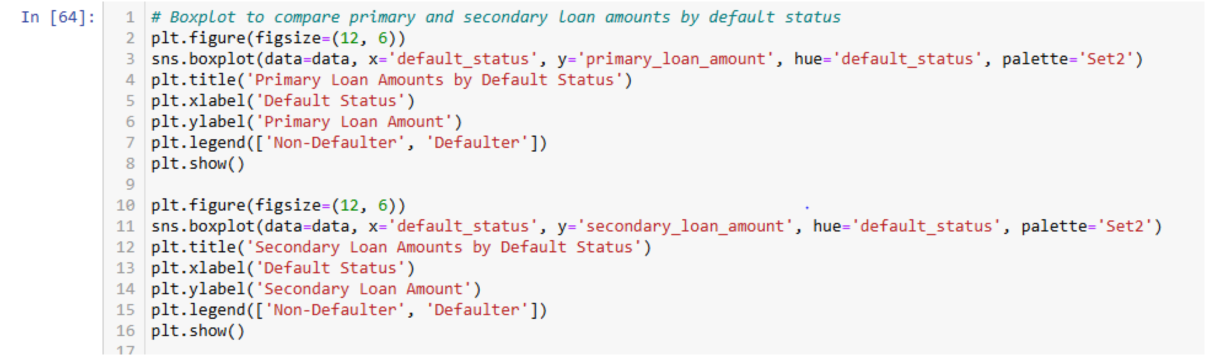


```
# Credit history details
new_loans_last_six_months = np.random.randint(0, 4, n_samples)
loans_defaulted_last_six_months = np.random.randint(0, 2, n_samples)
time_since_first_loan = np.random.randint(1, 10, n_samples)

# Create DataFrame
data = pd.DataFrame({
    'credit_bureau_score': credit_bureau_scores,
    'default_status': default_status,
    'primary_loan_amount': primary_loan_amounts,
    'secondary_loan_amount': secondary_loan_amounts,
    'sanctioned_amount': sanctioned_amounts,
    'disbursed_amount': disbursed_amounts,
    'inquiry_count': inquiry_count,
    'new_loans_last_six_months': new_loans_last_six_months,
    'loans_defaulted_last_six_months': loans_defaulted_last_six_months,
    'time_since_first_loan': time_since_first_loan
})

```



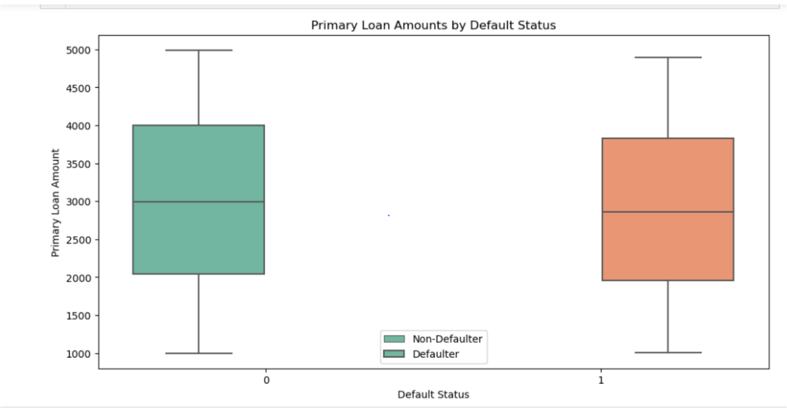
```
# Plotting credit bureau score distribution for defaulters vs. non-defaulters
plt.figure(figsize=(12, 6))
sns.histplot(data=data, x='credit_bureau_score', hue='default_status', kde=True, bins=30, palette='Set1')
plt.title('Credit Bureau Score Distribution by Default Status')
plt.xlabel('Credit Bureau Score')
plt.ylabel('Count')
plt.legend(['Non-Defaulter', 'Defaulter'])
plt.show()
```

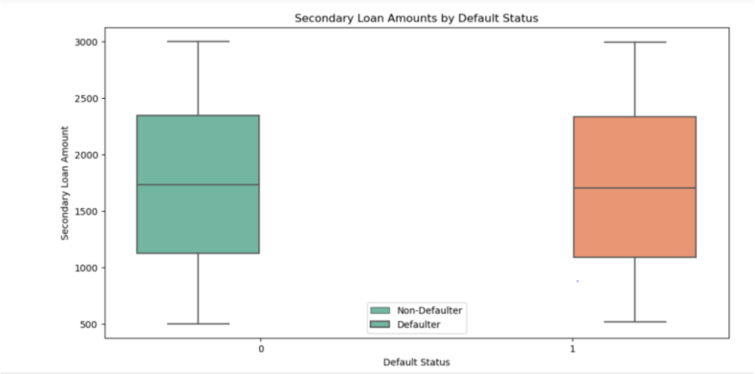


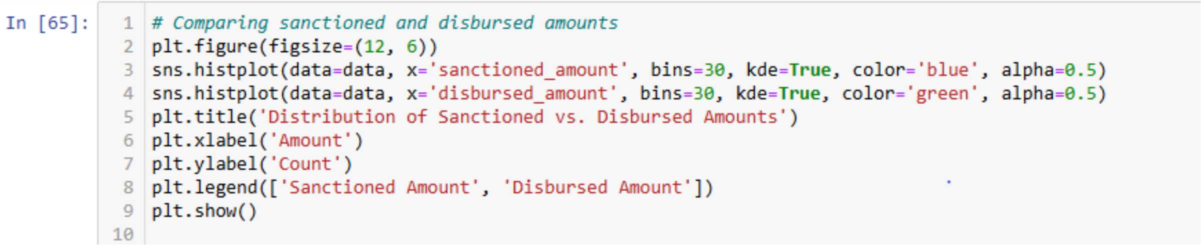


In [64]:

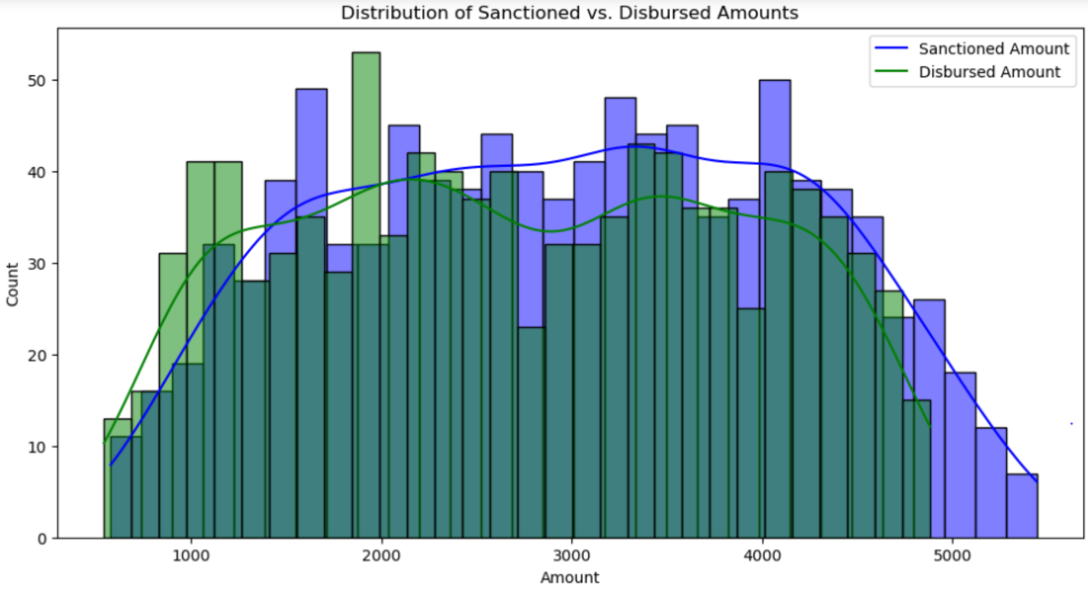
```
# Boxplot to compare primary and secondary loan amounts by default status
plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x='default_status', y='primary_loan_amount', hue='default_status', palette='Set2')
plt.title('Primary Loan Amounts by Default Status')
plt.xlabel('Default Status')
plt.ylabel('Primary Loan Amount')
plt.legend(['Non-Defaulter', 'Defaulter'])
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(data=data, x='default_status', y='secondary_loan_amount', hue='default_status', palette='Set2')
plt.title('Secondary Loan Amounts by Default Status')
plt.xlabel('Default Status')
plt.ylabel('Secondary Loan Amount')
plt.legend(['Non-Defaulter', 'Defaulter'])
plt.show()
```

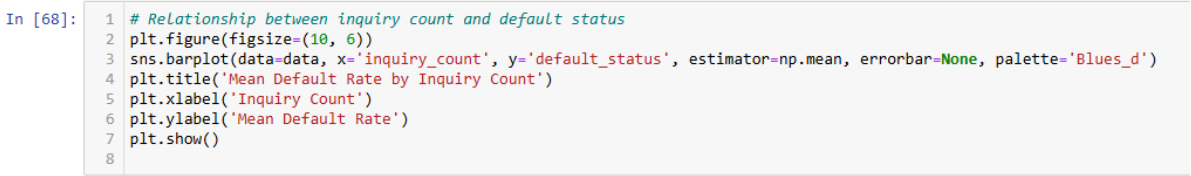




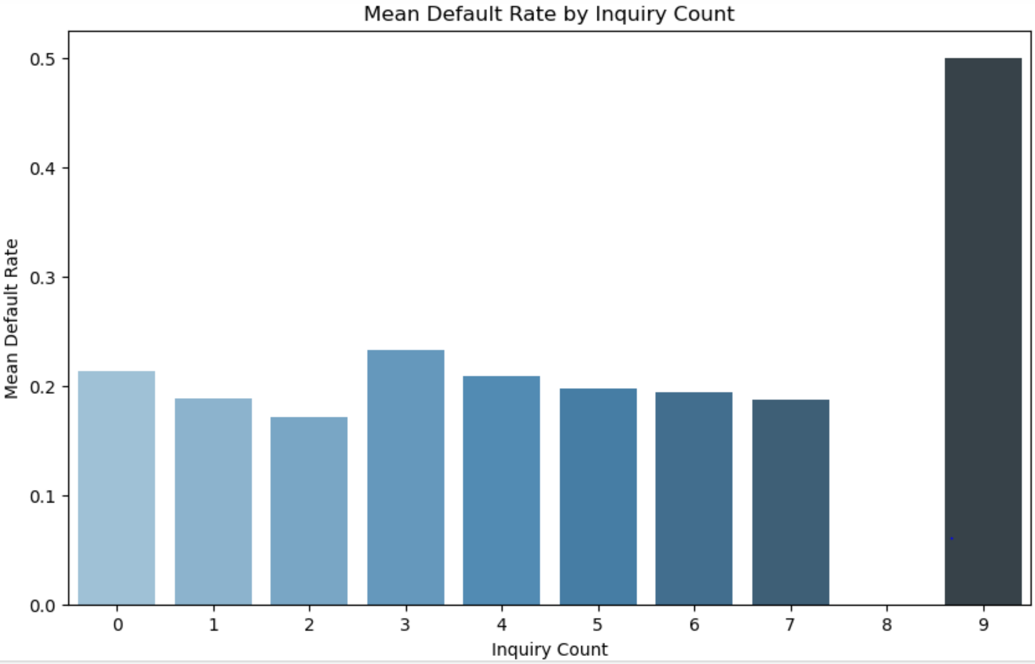


```
In [65]:
# Comparing sanctioned and disbursed amounts
plt.figure(figsize=(12, 6))
sns.histplot(data=data, x='sanctioned_amount', bins=30, kde=True, color='blue', alpha=0.5)
sns.histplot(data=data, x='disbursed_amount', bins=30, kde=True, color='green', alpha=0.5)
plt.title('Distribution of Sanctioned vs. Disbursed Amounts')
plt.xlabel('Amount')
plt.ylabel('Count')
plt.legend(['Sanctioned Amount', 'Disbursed Amount'])
plt.show()
```



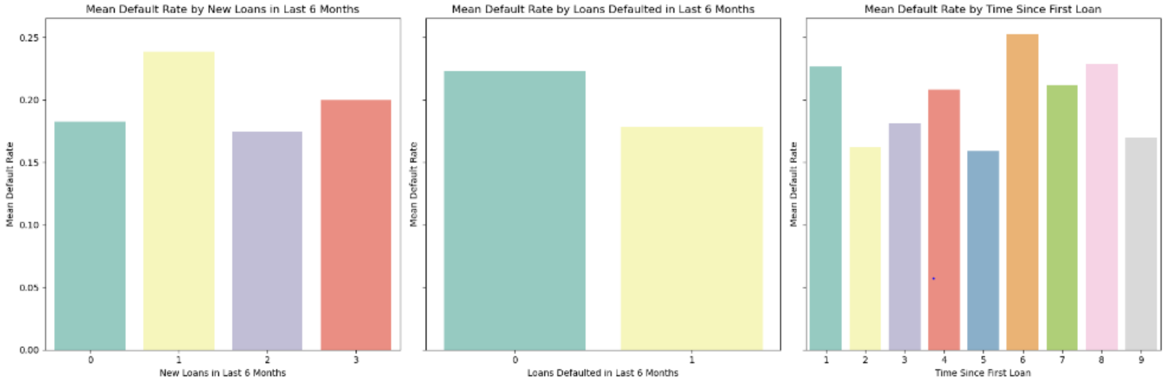


```
In [68]:
# Relationship between inquiry count and default status
plt.figure(figsize=(10, 6))
sns.barplot(data=data, x='inquiry_count', y='default_status', estimator=np.mean, errorbar=None, palette='Blues_d')
plt.title('Mean Default Rate by Inquiry Count')
plt.xlabel('Inquiry Count')
plt.ylabel('Mean Default Rate')
plt.show()
```



In [69]:

```python
# Plotting credit history features by default status
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharey=True)

sns.barplot(data=data, x='new_loans_last_six_months', y='default_status', ax=axes[0], estimator=np.mean,  errorbar=None, pal
axes[0].set_title('Mean Default Rate by New Loans in Last 6 Months')
axes[0].set_xlabel('New Loans in Last 6 Months')
axes[0].set_ylabel('Mean Default Rate')

sns.barplot(data=data, x='loans_defaulted_last_six_months', y='default_status', ax=axes[1], estimator=np.mean,  errorbar=Non
axes[1].set_title('Mean Default Rate by Loans Defaulted in Last 6 Months')
axes[1].set_xlabel('Loans Defaulted in Last 6 Months')
axes[1].set_ylabel('Mean Default Rate')

sns.barplot(data=data, x='time_since_first_loan', y='default_status', ax=axes[2], estimator=np.mean,  errorbar=None, palette
axes[2].set_title('Mean Default Rate by Time Since First Loan')
axes[2].set_xlabel('Time Since First Loan')
axes[2].set_ylabel('Mean Default Rate')

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


Mean Default Rate by New Loans in Last 6 Months
Mean Default Rate
New Loans in Last 6 Months
Mean Default Rate by Loans Defaulted in Last 6 Months
Mean Default Rate
Loans Defaulted in Last 6 Months
Mean Default Rate by Time Since First Loan
Mean Default Rate
Time Since First Loan