Step 1: Import Necessary Libraries

In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (classification_report, accuracy_score, confusion_ma
                             precision_score, recall_score, f1_score, roc_auc_sc
                             mean_squared_error, r2_score)
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Step 2: Load and Clean the Data

In [3]:
```
# Load the data
data = pd.read_csv('loanno_reg1.csv')

# Data cleaning: Drop missing values
data.dropna(inplace=True)  # For simplicity, drop missing values
```

Step 3: Encode Categorical Variables

In [4]:
```
# Encoding categorical variables
data['Gender'] = data['Gender'].map({'M': 1, 'F': 0})  # Map M to 1 and F to 0
data['Has Active Credit Card'] = data['Has Active Credit Card'].map({'Unpossesse
```

Step 4: Convert Target Variable to Binary

In [5]:
```
# Convert loan_approved to binary if it's continuous
data['loan_approved'] = (data['loan_approved'] > 0).astype(int)
```

Step 5: Outlier Detection

In [6]:
```
# Outlier Detection Function
def detect_outliers_iqr(data, feature):
    Q1 = data[feature].quantile(0.25)
    Q3 = data[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data[feature] < lower_bound) | (data[feature] > upper_bound)]

# Detect outliers for each feature
outliers = {}
for feature in ['Age', 'Income (USD)', 'Current Loan Expenses (USD)', 'Credit Sc
    outliers[feature] = detect_outliers_iqr(data, feature)
```

```python
# Print outliers found for each feature
for feature, outlier_data in outliers.items():
    print(f"Outliers in {feature}:\n{outlier_data}\n")
```

```
Outliers in Age:
Empty DataFrame
Columns: [Customer ID, Name, Gender, Age, Income (USD), Profession, Location, Loa
n Amount Request (USD), Current Loan Expenses (USD), Credit Score, Has Active Cre
dit Card, Property ID, Property Age, Property Location, Property Price, loan_appr
oved]
Index: []

Outliers in Income (USD):
   Customer ID             Name  Gender  Age  Income (USD)  \
39     C-24451  Margareta Wind       1   18       7885.56

              Profession Location  Loan Amount Request (USD)  \
39  Commercial associate    Urban                   121963.5

    Current Loan Expenses (USD)  Credit Score  Has Active Credit Card  \
39                        587.8        721.18                     0.0

    Property ID  Property Age Property Location  Property Price  loan_approved
39          530       7885.56             Urban       142645.25              1

Outliers in Current Loan Expenses (USD):
Empty DataFrame
Columns: [Customer ID, Name, Gender, Age, Income (USD), Profession, Location, Loa
n Amount Request (USD), Current Loan Expenses (USD), Credit Score, Has Active Cre
dit Card, Property ID, Property Age, Property Location, Property Price, loan_appr
oved]
Index: []

Outliers in Credit Score:
Empty DataFrame
Columns: [Customer ID, Name, Gender, Age, Income (USD), Profession, Location, Loa
n Amount Request (USD), Current Loan Expenses (USD), Credit Score, Has Active Cre
dit Card, Property ID, Property Age, Property Location, Property Price, loan_appr
oved]
Index: []

Outliers in Loan Amount Request (USD):
   Customer ID               Name  Gender  Age  Income (USD) Profession  \
29     C-39879  Thea Rodenberger       1   29       3880.49    Working

   Location  Loan Amount Request (USD)  Current Loan Expenses (USD)  \
29    Urban                  234770.77                       636.65

    Credit Score  Has Active Credit Card  Property ID  Property Age  \
29        767.85                     NaN          514       3880.49

   Property Location  Property Price  loan_approved
29        Semi-Urban       326886.67              1
```

Step 6: Select Features and Target Variable

In [7]:
```
# Select features and target variable
features = ['Age', 'Income (USD)', 'Current Loan Expenses (USD)', 'Credit Score'
target = 'loan_approved'

X = data[features]
y = data[target]
```

Step 7: Train-Test Split

In [8]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

Step 8: Scale the Features

In [9]:
```
# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 9: Create and Fit the Logistic Regression Model

In [10]:
```
# Create and fit the Logistic Regression model
model = LogisticRegression(max_iter=200)  # Increased max_iter to 200
model.fit(X_train, y_train)
```

Out[10]:
```
▾ LogisticRegression
LogisticRegression(max_iter=200)
```

Step 10: Make Predictions

In [11]:
```
# Predictions
predictions = model.predict(X_test)
predicted_probabilities = model.predict_proba(X_test)[:, 1]  # Get predicted pro
```

Step 11: Evaluate the Model

In [12]:
```
# Evaluation Metrics
accuracy = accuracy_score(y_test, predictions)
```

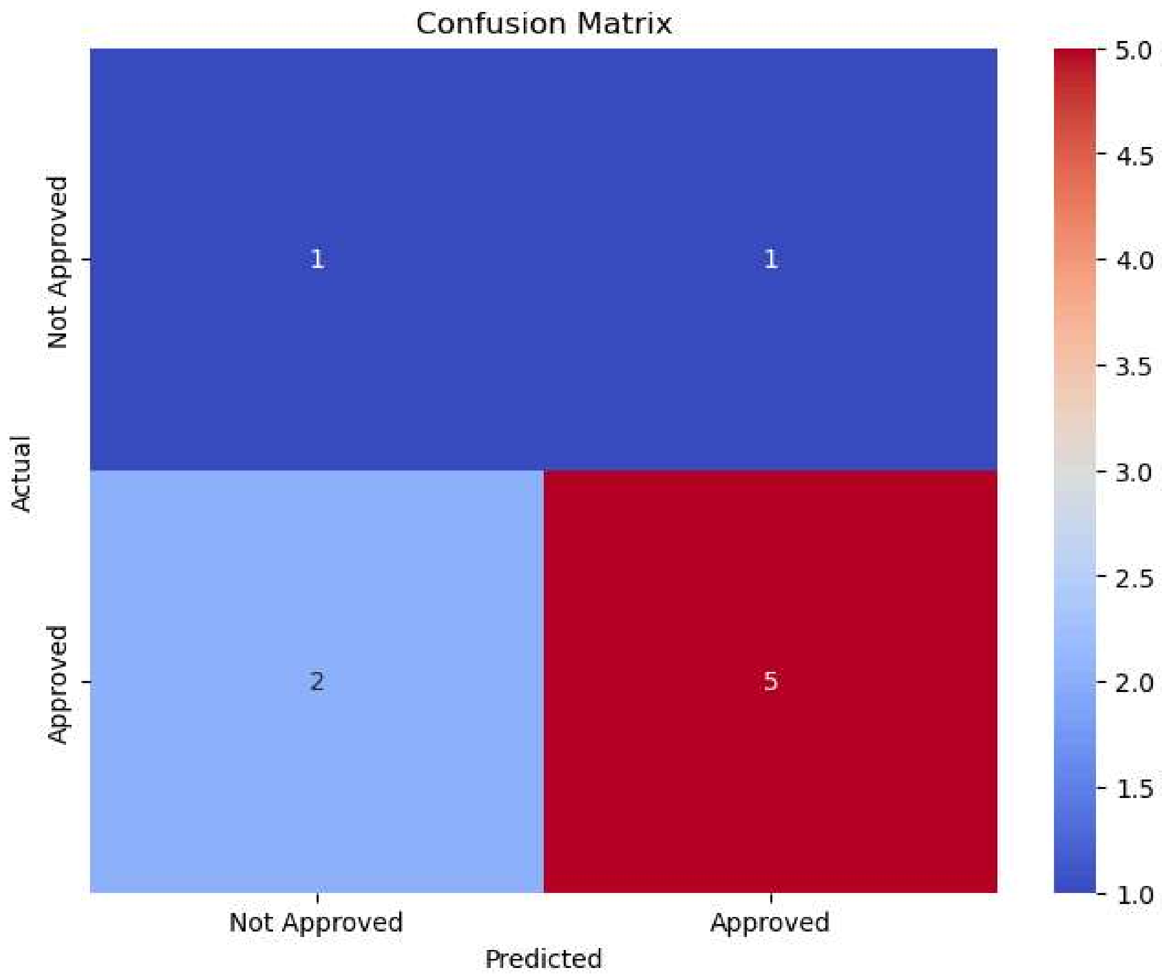
```python
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)
roc_auc = roc_auc_score(y_test, predictions)

# MSE and R² score (applicable in regression context)
mse = mean_squared_error(y_test, predicted_probabilities)
r2 = r2_score(y_test, predicted_probabilities)

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC AUC Score:", roc_auc)
print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)
print("\nClassification Report:\n", classification_report(y_test, predictions))
```

```
Accuracy: 0.6666666666666666
Precision: 0.8333333333333334
Recall: 0.7142857142857143
F1 Score: 0.7692307692307693
ROC AUC Score: 0.6071428571428572
Mean Squared Error (MSE): 0.14981190110969453
R² Score: 0.1332311435796245

Classification Report:
               precision    recall  f1-score   support

           0       0.33      0.50      0.40         2
           1       0.83      0.71      0.77         7

    accuracy                           0.67         9
   macro avg       0.58      0.61      0.58         9
weighted avg       0.72      0.67      0.69         9
```

Step 12: Confusion Matrix Visualization

In [15]:

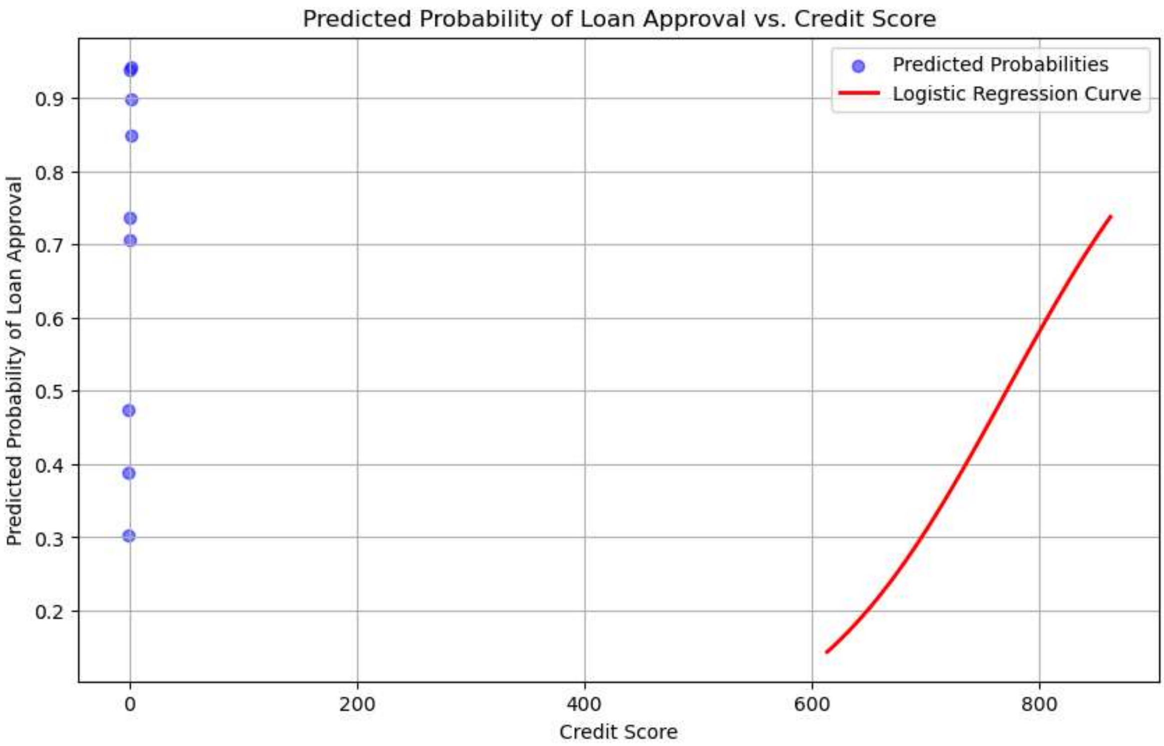
```python
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)

# Plotting Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='coolwarm', xticklabels=['Not
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()
```



Step 13: Logistic Regression Curve Visualization

In [16]:
```
# Logistic Regression Curve
plt.figure(figsize=(10, 6))

# Scatter plot of Credit Score vs. Predicted Probabilities
plt.scatter(X_test[:, 3], predicted_probabilities, color='blue', label='Predicte

# Create a range of Credit Score values for the plot
credit_score_range = np.linspace(X['Credit Score'].min(), X['Credit Score'].max(

# Create a DataFrame for dummy features, only using 'Credit Score'
dummy_features = pd.DataFrame(0, index=np.arange(100), columns=features)  # Crea
dummy_features['Credit Score'] = credit_score_range.flatten()  # Set the Credit

# Scale the dummy features
dummy_features_scaled = scaler.transform(dummy_features)

# Get predicted probabilities for the logistic fit
predicted_probabilities_line = model.predict_proba(dummy_features_scaled)[:, 1]

# Plot the logistic regression curve
plt.plot(credit_score_range, predicted_probabilities_line, color='red', label='L

plt.xlabel('Credit Score')
plt.ylabel('Predicted Probability of Loan Approval')
```

```python
plt.title('Predicted Probability of Loan Approval vs. Credit Score')
plt.legend()
plt.grid()
plt.show()
```



In [ ]: