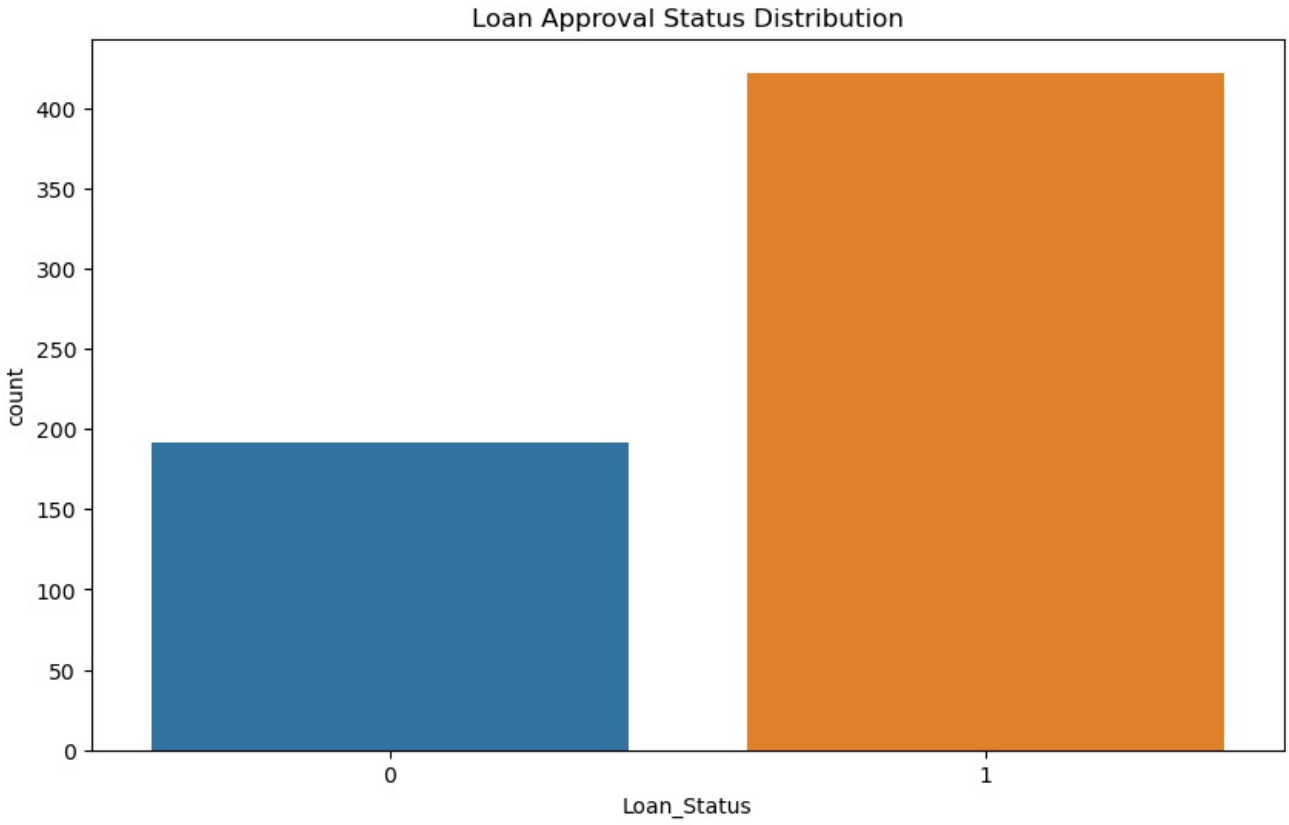
In [1]:
```
pip install pandas numpy scikit-learn matplotlib seaborn plotly
```

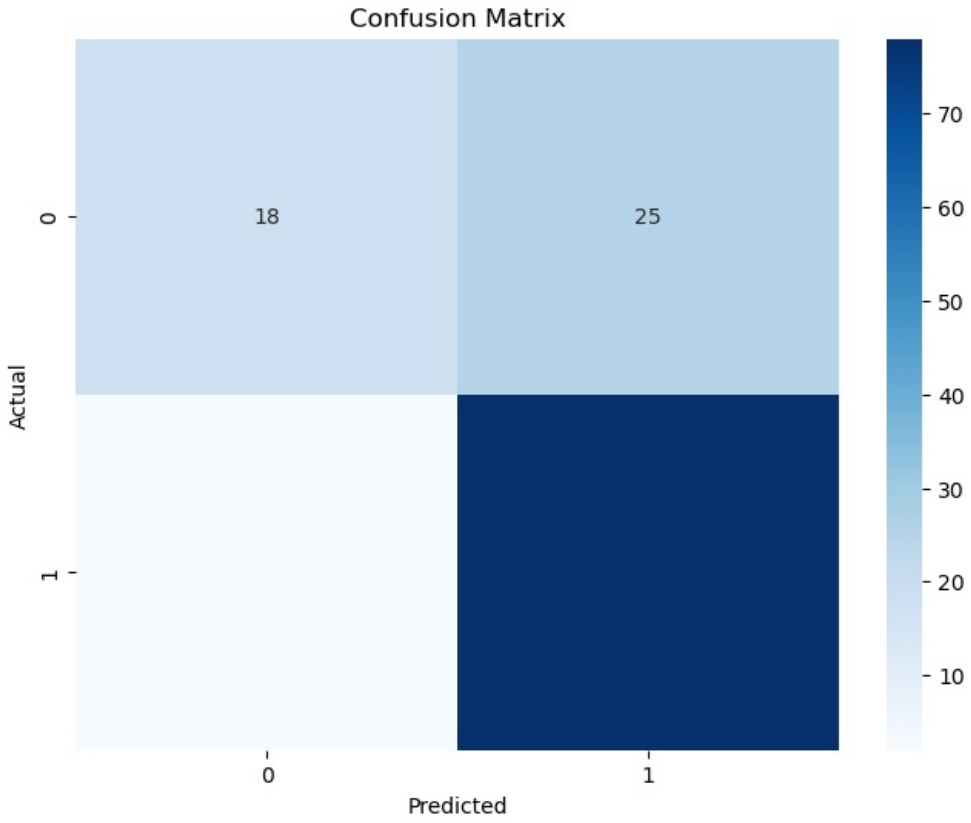
```
Requirement already satisfied: pandas in d:\ml\lib\site-packages (2.1.4)
Requirement already satisfied: numpy in d:\ml\lib\site-packages (1.26.4)
Requirement already satisfied: scikit-learn in d:\ml\lib\site-packages (1.2.2)
Requirement already satisfied: matplotlib in d:\ml\lib\site-packages (3.8.0)
Requirement already satisfied: seaborn in d:\ml\lib\site-packages (0.12.2)
Requirement already satisfied: plotly in d:\ml\lib\site-packages (5.9.0)
Requirement already satisfied: python-dateutil>=2.8.2 in d:\ml\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in d:\ml\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in d:\ml\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.3.2 in d:\ml\lib\site-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in d:\ml\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\ml\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: contourpy>=1.0.1 in d:\ml\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in d:\ml\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in d:\ml\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in d:\ml\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in d:\ml\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in d:\ml\lib\site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in d:\ml\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: tenacity>=6.2.0 in d:\ml\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: six>=1.5 in d:\ml\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0
)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:
```python
# Import necessary libraries
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.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.impute import SimpleImputer
import plotly.express as px

# Load the dataset
# Replace this with your actual dataset path
df = pd.read_csv('loan_prediction.csv')

# Data Preprocessing
# Drop irrelevant columns
df.drop(columns=['Loan_ID'], inplace=True)

# Convert Loan_Status to binary
df['Loan_Status'] = df['Loan_Status'].map({'Y': 1, 'N': 0})

# Exploratory Data Analysis (EDA)
# Visualizing the distribution of Loan Status
plt.figure(figsize=(10, 6))
sns.countplot(x='Loan_Status', data=df)
plt.title('Loan Approval Status Distribution')
plt.show()

# Visualizing other categorical variables
fig = px.histogram(df, x='Gender', color='Loan_Status', barmode='group', title='Loan Status by Gender')
fig.show()

fig = px.histogram(df, x='Married', color='Loan_Status', barmode='group', title='Loan Status by Marital Status'
fig.show()

fig = px.histogram(df, x='Education', color='Loan_Status', barmode='group', title='Loan Status by Education')
fig.show()

# Feature Engineering
# Define features and target variable
X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']

# Identify categorical and numerical columns
categorical_cols = X.select_dtypes(include=['object']).columns
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns

# Create a ColumnTransformer for preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('num', SimpleImputer(strategy='median'), numerical_cols),
        ('cat', Pipeline(steps=[
```

```
            ('imputer', SimpleImputer(strategy='most_frequent')),
            ('onehot', OneHotEncoder(drop='first'))
        ]), categorical_cols)
    ])

# Preprocess the data
X_processed = preprocessor.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_processed, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model Training
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)

# Print classification report and confusion matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Visualize the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



```
              precision    recall  f1-score   support

           0       0.90      0.42      0.57        43
           1       0.76      0.97      0.85        80

    accuracy                           0.78       123
   macro avg       0.83      0.70      0.71       123
weighted avg       0.81      0.78      0.75       123

[[18 25]
 [ 2 78]]
```



In [ ]: