prodigy-ds-02

September 10, 2023

1 Task-02

Dataset used: https://www.kaggle.com/datasets/laotse/credit-risk-dataset

Step 1: Import Libraries and Load Data

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset (replace 'your_dataset.csv' with your actual file path)
df = pd.read_csv("C:\\Users\\Narthana\\Downloads\\credit_risk.csv")
```

Step 2: Explore the Dataset

[3]:
```
# Display the first few rows of the dataset
print(df.head())

# Check for missing values
print(df.isnull().sum())

# Summary statistics
print(df.describe())
```

```
   Id  Age  Income      Home  Emp_length     Intent  Amount   Rate  Status  \
0   0   22   59000      RENT       123.0   PERSONAL   35000  16.02       1
1   1   21    9600       OWN         5.0  EDUCATION    1000  11.14       0
2   2   25    9600  MORTGAGE         1.0    MEDICAL    5500  12.87       1
3   3   23   65500      RENT         4.0    MEDICAL   35000  15.23       1
4   4   24   54400      RENT         8.0    MEDICAL   35000  14.27       1

   Percent_income Default  Cred_length
0            0.59       Y            3
1            0.10       N            2
2            0.57       N            3
3            0.53       N            2
4            0.55       Y            4
Id                   0
Age                  0
```

```
Income                 0
Home                   0
Emp_length           895
Intent                 0
Amount                 0
Rate                3116
Status                 0
Percent_income         0
Default                0
Cred_length            0
dtype: int64
                 Id           Age        Income    Emp_length        Amount  \
count  32581.000000  32581.000000  3.258100e+04  31686.000000  32581.000000
mean   16290.006139     27.734600  6.607485e+04      4.789686   9589.371106
std     9405.479594      6.348078  6.198312e+04      4.142630   6322.086646
min        0.000000     20.000000  4.000000e+03      0.000000    500.000000
25%     8145.000000     23.000000  3.850000e+04      2.000000   5000.000000
50%    16290.000000     26.000000  5.500000e+04      4.000000   8000.000000
75%    24435.000000     30.000000  7.920000e+04      7.000000  12200.000000
max    32780.000000    144.000000  6.000000e+06    123.000000  35000.000000

               Rate        Status  Percent_income   Cred_length
count  29465.000000  32581.000000    32581.000000  32581.000000
mean      11.011695      0.218164        0.170203      5.804211
std        3.240459      0.413006        0.106782      4.055001
min        5.420000      0.000000        0.000000      2.000000
25%        7.900000      0.000000        0.090000      3.000000
50%       10.990000      0.000000        0.150000      4.000000
75%       13.470000      0.000000        0.230000      8.000000
max       23.220000      1.000000        0.830000     30.000000
```

Explanation:

df.head() shows the first few rows of the dataset, helping you understand its structure. df.isnull().sum() checks for missing values in each column. df.describe() provides summary statistics for numerical columns.

Step 3: Data Cleaning

[5]:
```
# List the column names in the DataFrame
print(df.columns)
```

```
Index(['Id', 'Age', 'Income', 'Home', 'Emp_length', 'Intent', 'Amount', 'Rate',
       'Status', 'Percent_income', 'Default', 'Cred_length'],
      dtype='object')
```

[6]:
```
# Fill missing values in 'Age' column with the mean
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

```
# Fill missing values in 'Income' column with the median (you can choose a␣
↪suitable strategy)
df['Income'].fillna(df['Income'].median(), inplace=True)

# Now, drop rows with any remaining missing values in the entire DataFrame
df.dropna(inplace=True)
```

[21]:
```
# Check for missing values in the DataFrame
missing_values = df.isnull().sum()

# Print the number of missing values for each column
print(missing_values)
```

```
Id                0
Age               0
Income            0
Home              0
Emp_length        0
Intent            0
Amount            0
Rate              0
Status            0
Percent_income    0
Default           0
Cred_length       0
dtype: int64
```

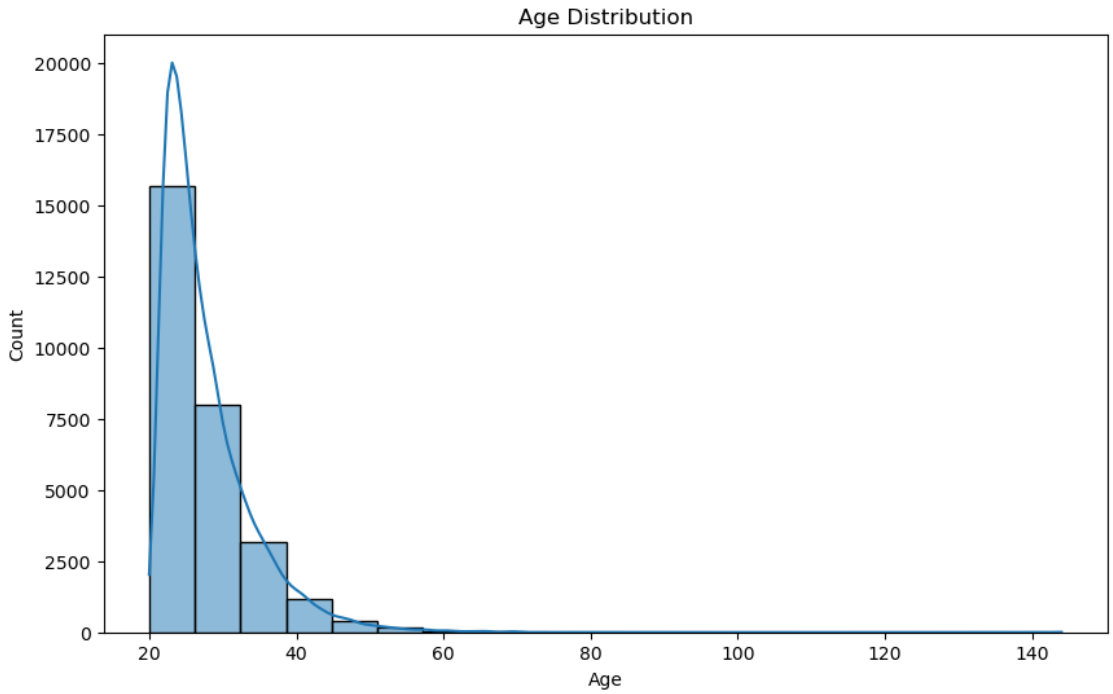
Explanation:

In this example, we filled missing values in the 'Age' column with the mean age. We dropped rows with missing values in the 'Sex' and 'Embarked' columns.

Step 4: Data Visualization and Exploration

Histograms for Numerical Variables:

[7]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], bins=20, kde=True)
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Distribution')
plt.show()
```



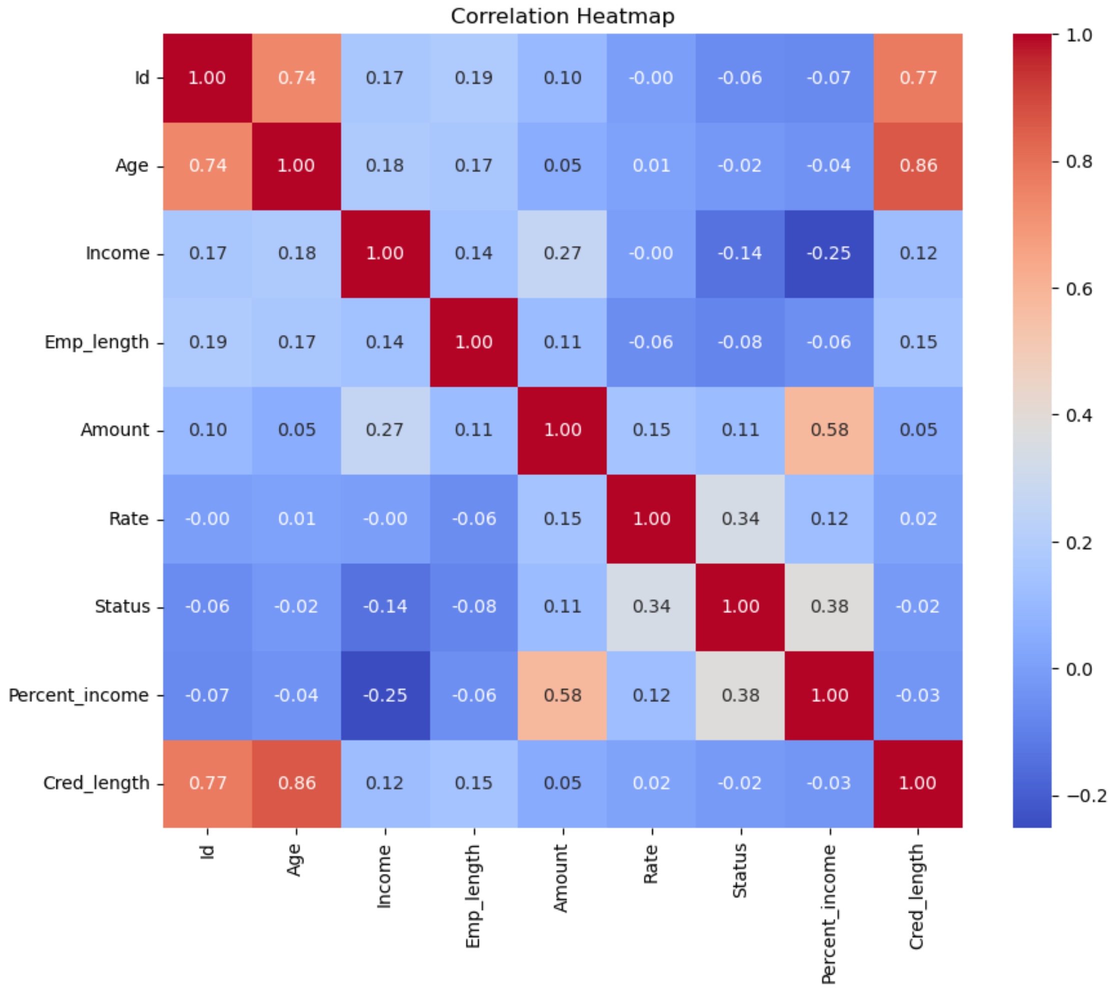
Explanation:

This code plots a histogram of the 'Age' column, showing the distribution of ages.

Correlation Heatmap for Numerical Variables:

[11]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\Narthana\AppData\Local\Temp\ipykernel_19164\3216002298.py:2:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
```

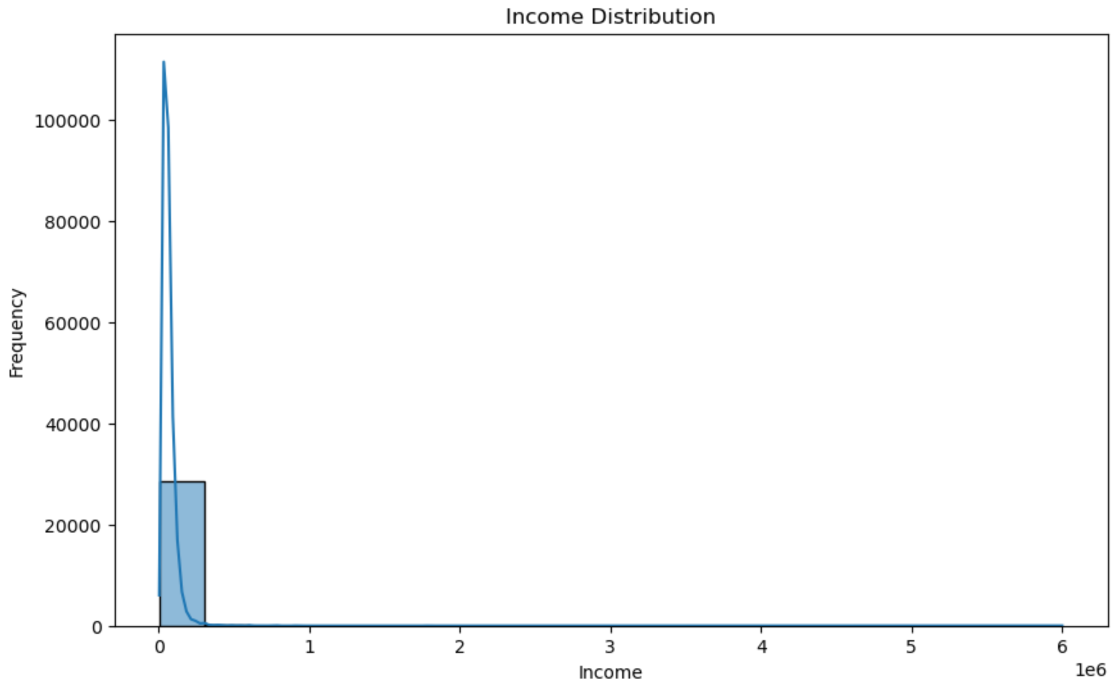


Explanation:

This code generates a correlation heatmap to visualize relationships between numerical variables.

Income Distribution:

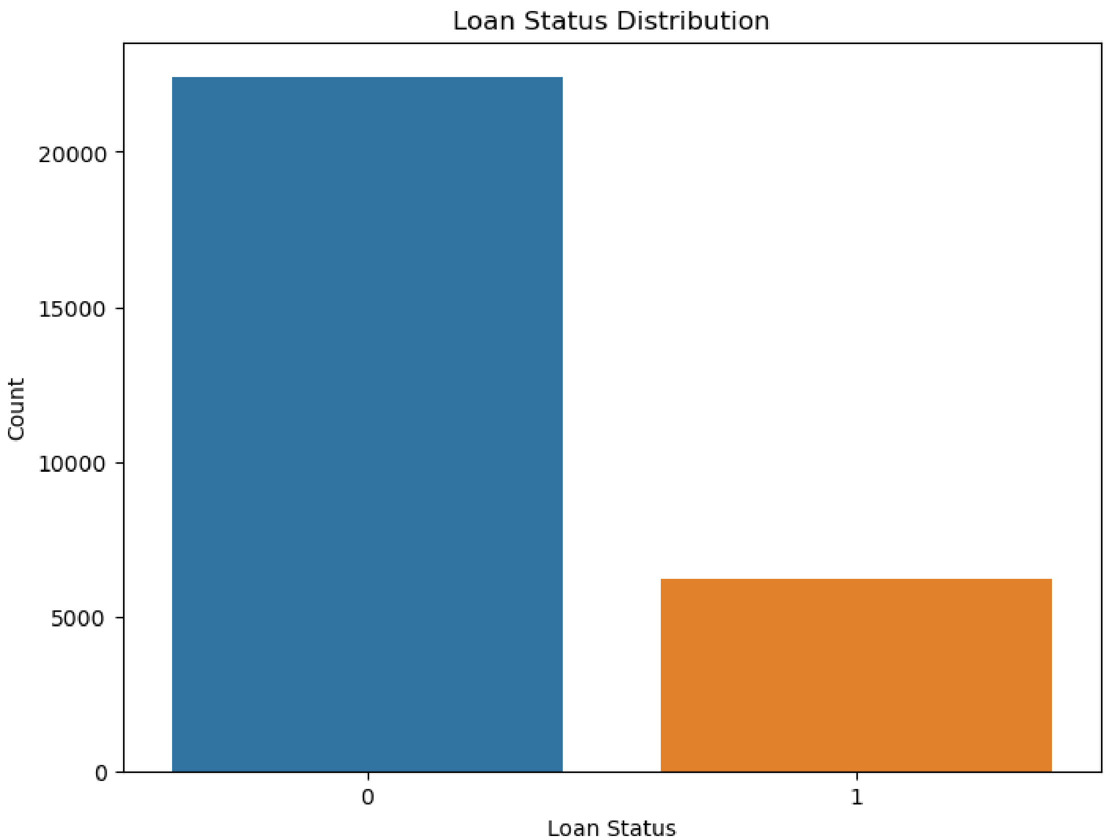
[12]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Income', bins=20, kde=True)
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.title('Income Distribution')
plt.show()
```



This histogram shows the distribution of income levels. It can help you identify whether your dataset consists of individuals with varying income levels or if there are specific income brackets that are more common. It's also useful for identifying outliers.

Loan Status Distribution

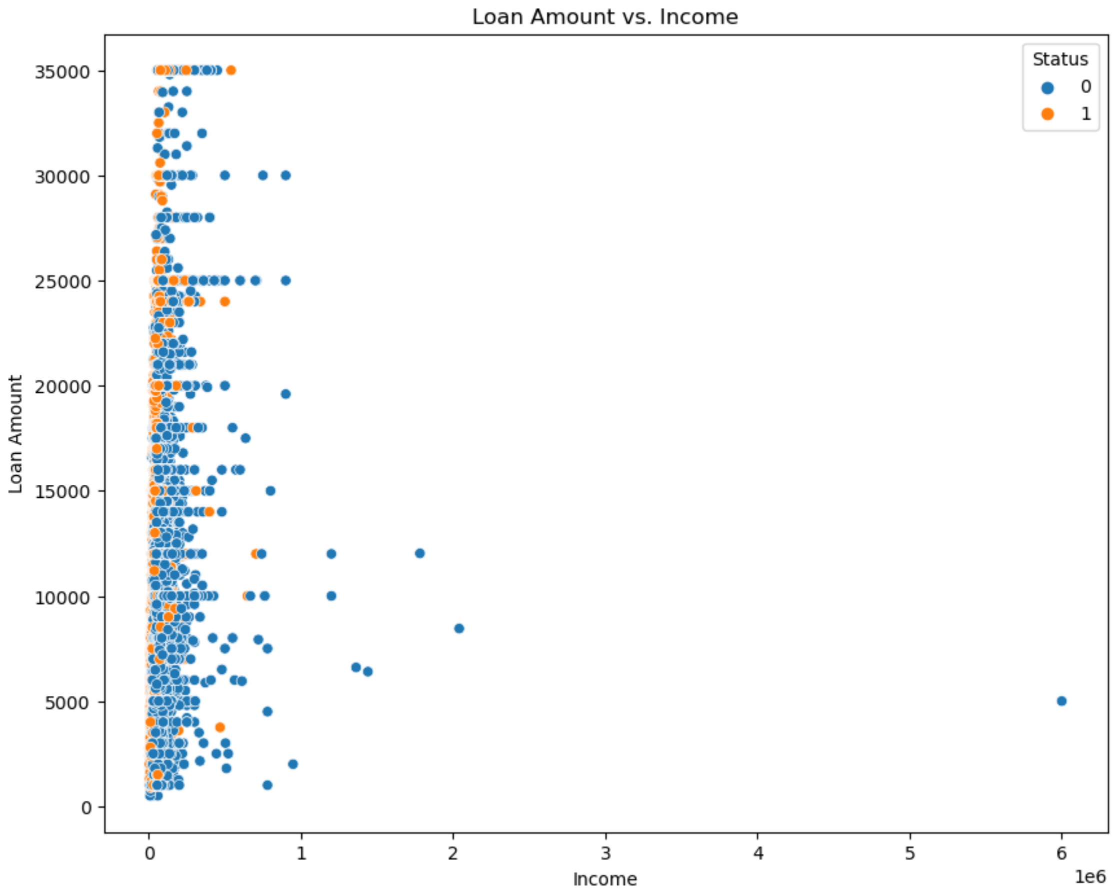
[13]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Status')
plt.xlabel('Loan Status')
plt.ylabel('Count')
plt.title('Loan Status Distribution')
plt.show()
```



This count plot shows the distribution of loan statuses. It helps you understand how many loans were approved, denied, or are in some other status. This information is crucial for understanding the overall performance of loans in your dataset.

Loan Amount vs. Income:

[15]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(data=df, x='Income', y='Amount', hue='Status')
plt.xlabel('Income')
plt.ylabel('Loan Amount')
plt.title('Loan Amount vs. Income')
plt.show()
```



This scatter plot visualizes the relationship between income and loan amount. It can help you understand if there's a correlation between income level and the requested loan amount. Additionally, coloring points by loan status can help identify any patterns related to loan approval/denial.

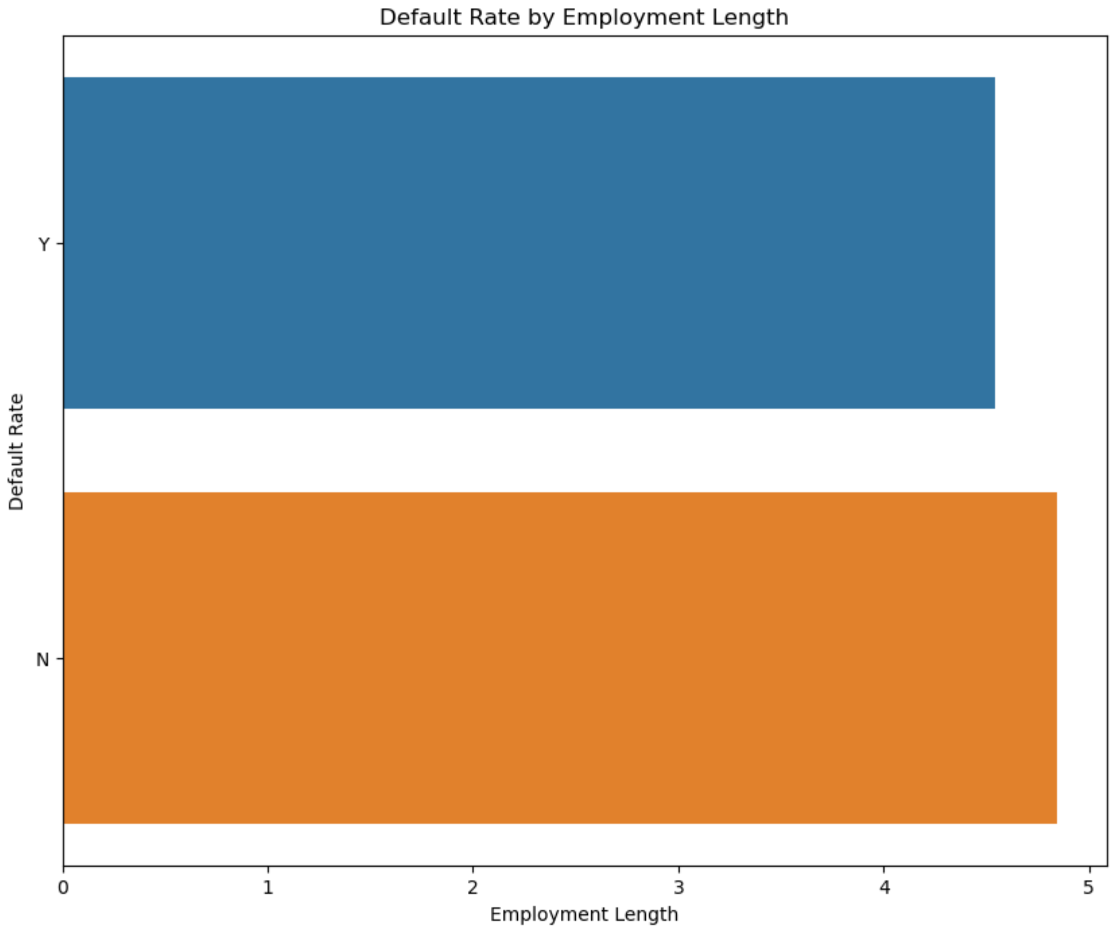
Employment Length and Default Rate:

[16]:
```
plt.figure(figsize=(10, 8))
sns.barplot(data=df, x='Emp_length', y='Default', ci=None)
plt.xlabel('Employment Length')
plt.ylabel('Default Rate')
plt.title('Default Rate by Employment Length')
plt.show()
```

```
C:\Users\Narthana\AppData\Local\Temp\ipykernel_19164\3976789286.py:2:
FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='Emp_length', y='Default', ci=None)
```



This bar plot shows the default rate for different employment lengths. It helps you understand if individuals with longer employment history are less likely to default on loans.

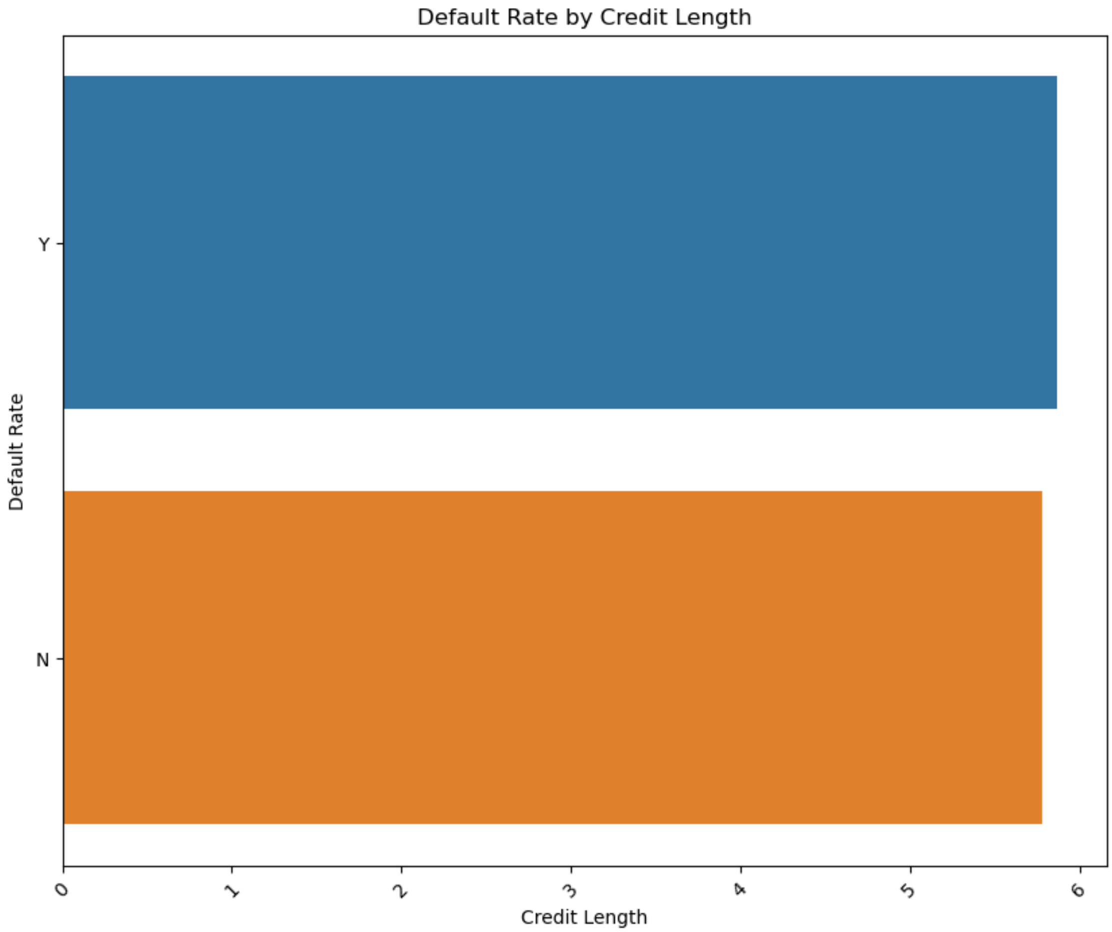
Default Rate by Credit Length:

[19]:
```
plt.figure(figsize=(10, 8))
sns.barplot(data=df, x='Cred_length', y='Default', ci=None)
plt.xlabel('Credit Length')
plt.ylabel('Default Rate')
plt.title('Default Rate by Credit Length')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\Narthana\AppData\Local\Temp\ipykernel_19164\2279026864.py:2:
FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='Cred_length', y='Default', ci=None)
```

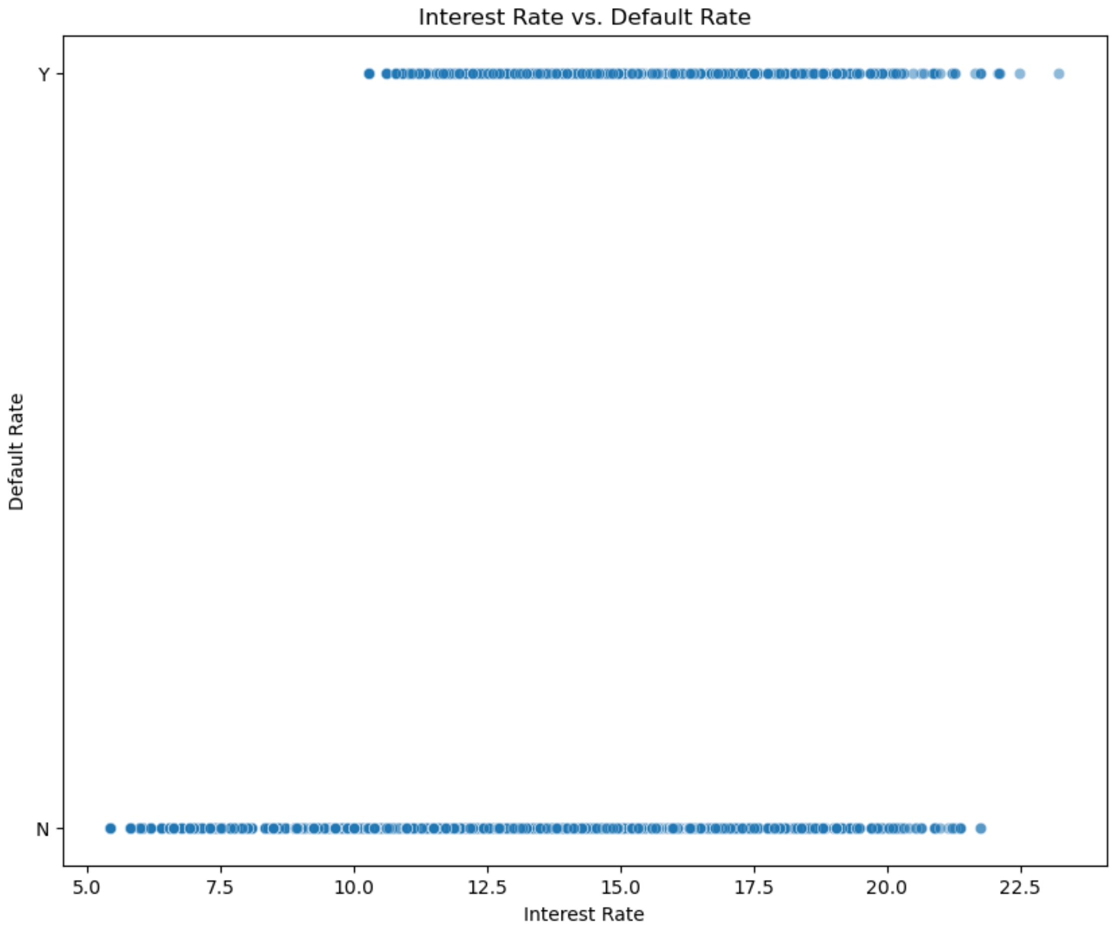


Explanation: The bar plot displays the default rate for different lengths of credit history. It helps you assess if individuals with longer credit histories are less likely to default on loans.

Inference: Borrowers with a longer credit history tend to have a lower default rate. This suggests that a strong credit history is an important factor in loan repayment.

Relationship between Interest Rate and Default:

[20]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(data=df, x='Rate', y='Default', alpha=0.5)
plt.xlabel('Interest Rate')
plt.ylabel('Default Rate')
plt.title('Interest Rate vs. Default Rate')
plt.show()
```



Explanation: The scatter plot visualizes the relationship between interest rate and default rate. It helps you understand if higher interest rates are associated with higher default rates.

Inference: There doesn't seem to be a strong linear relationship between interest rate and default rate. However, there are more defaults at higher interest rates, which is something to consider in risk assessment.

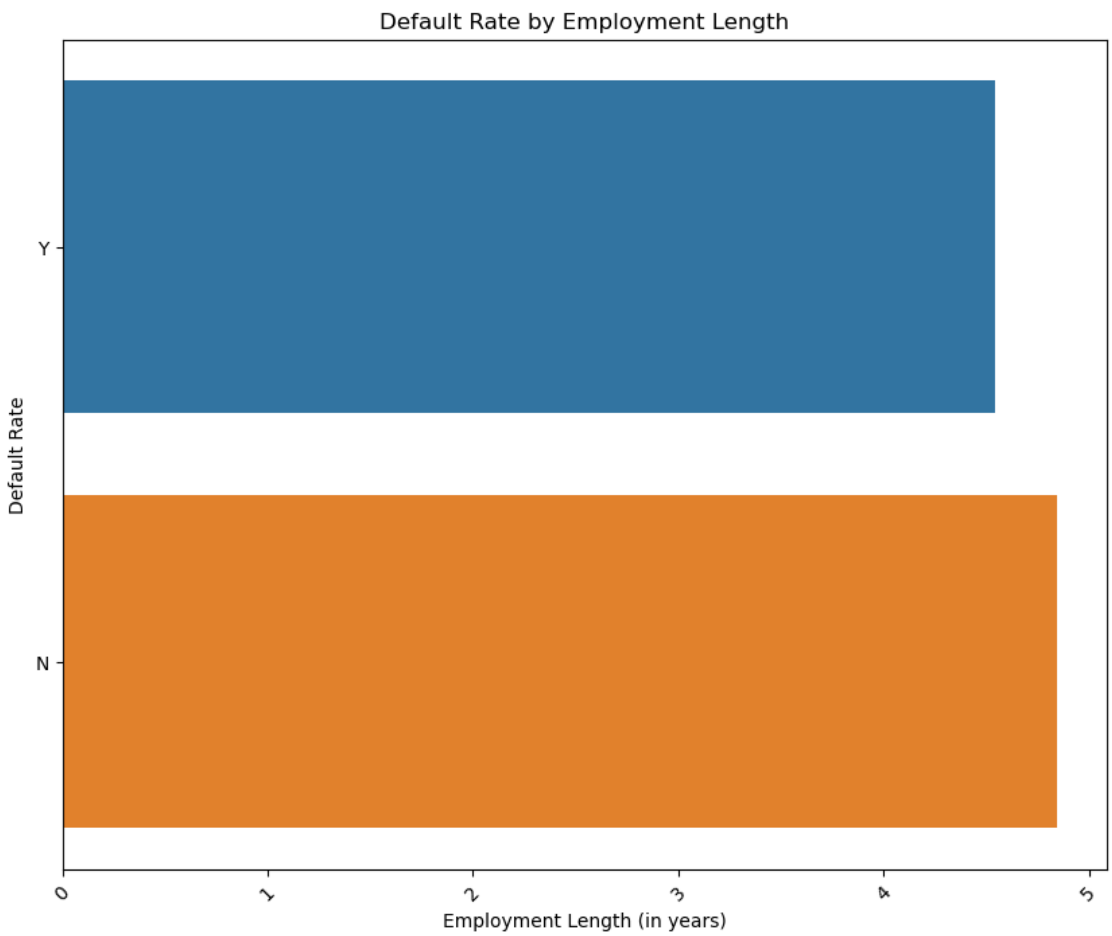
Default Rate by Employment Length:

[22]:
```
plt.figure(figsize=(10, 8))
sns.barplot(data=df, x='Emp_length', y='Default', ci=None)
plt.xlabel('Employment Length (in years)')
plt.ylabel('Default Rate')
plt.title('Default Rate by Employment Length')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\Narthana\AppData\Local\Temp\ipykernel_19164\2946634659.py:2:
FutureWarning:
```

```
The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='Emp_length', y='Default', ci=None)
```

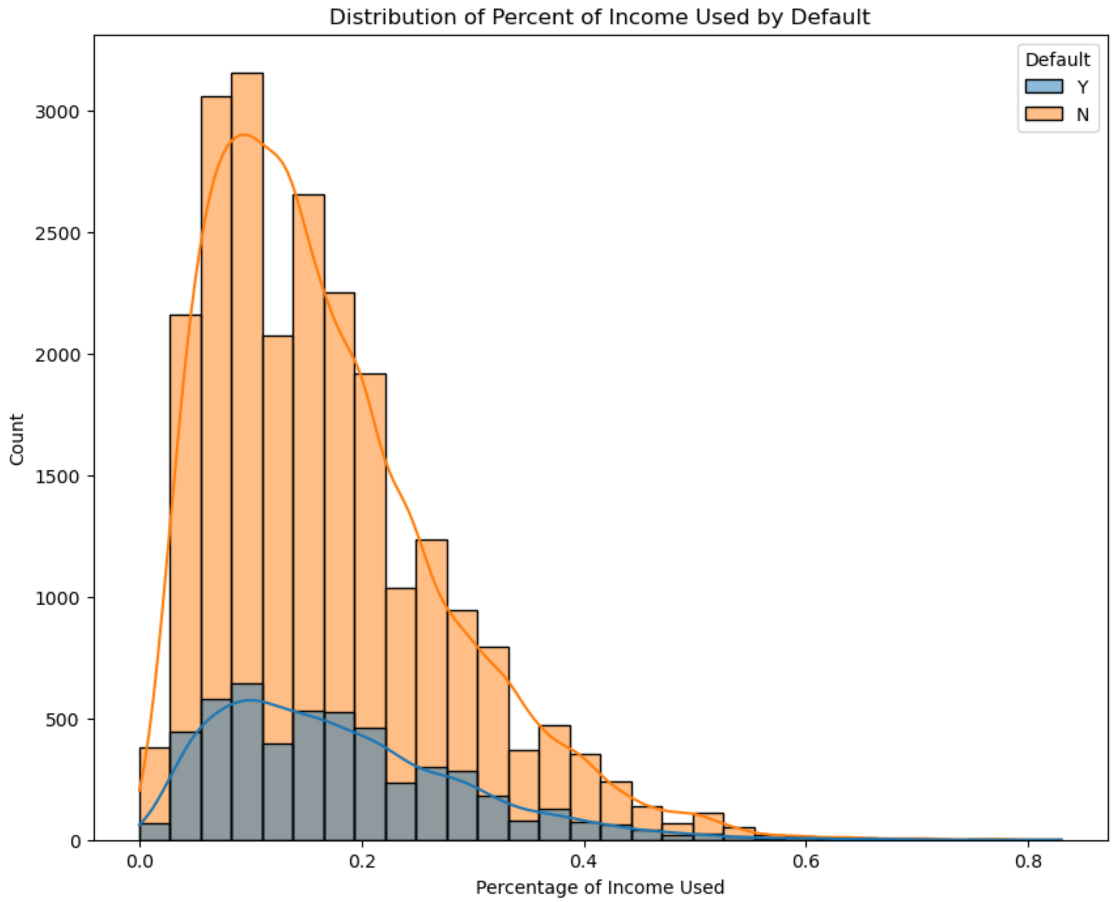


Explanation: This bar plot shows the default rate for different employment lengths. It helps you understand if individuals with longer employment history are less likely to default on loans.

Inference: Borrowers with longer employment history, especially those with 10+ years, tend to have a slightly lower default rate. Lenders might consider this when evaluating loan applications.

Default Rate by Percent of Income:

[24]:
```
plt.figure(figsize=(10, 8))
sns.histplot(data=df, x='Percent_income', hue='Default', bins=30, kde=True)
plt.xlabel('Percentage of Income Used')
plt.ylabel('Count')
plt.title('Distribution of Percent of Income Used by Default')
plt.show()
```



Explanation: This histogram with KDE (Kernel Density Estimate) displays the distribution of the percentage of income used for loan repayment, with different colors representing defaults and non-defaults.

Inference: Borrowers who use a higher percentage of their income for loan repayment are more likely to default. This suggests that lenders should carefully consider the debt-to-income ratio when assessing loan applications.