TASK ONE: EXPLORATORY DATA ANALYSIS (EDA)

Start with a dataset of your choice and perform EDA using libraries like pandas, numpy, and matplotlib or seaborn. Explore the data's characteristics, distributions, correlations, and outliers. Visualize your findings with histograms, scatter plots, and heatmaps to gain insights into the data.

Import the Required Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the Dataset

In [2]:
```
data = pd.read_csv('loan_borowwer_data.csv')
```

In [3]:
```
data
```

Out[3]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not.fully.paid
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	0
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	0
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	0
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	0
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9573	0	all_other	0.1461	344.76	12.180755	10.39	672	10474.000000	215372	82.1	2	0	0	1
9574	0	all_other	0.1253	257.70	11.141862	0.21	722	4380.000000	184	1.1	5	0	0	1
9575	0	debt_consolidation	0.1071	97.81	10.596635	13.09	687	3450.041667	10036	82.9	8	0	0	1
9576	0	home_improvement	0.1600	351.58	10.819778	19.18	692	1800.000000	0	3.2	5	0	0	1
9577	0	debt_consolidation	0.1392	853.43	11.264464	16.28	732	4740.000000	37879	57.0	6	0	0	1

9578 rows × 14 columns

Understand the Dataset

In [7]:
```
# Display the first few rows
print(data.head())
```

```
   credit.policy             purpose  int.rate  installment  log.annual.inc  \
0              1  debt_consolidation    0.1189       829.10       11.350407
1              1         credit_card    0.1071       228.22       11.082143
2              1  debt_consolidation    0.1357       366.86       10.373491
3              1  debt_consolidation    0.1008       162.34       11.350407
4              1         credit_card    0.1426       102.92       11.299732

     dti  fico  days.with.cr.line  revol.bal  revol.util  inq.last.6mths  \
0  19.48   737        5639.958333      28854        52.1               0
1  14.29   707        2760.000000      33623        76.7               0
2  11.63   682        4710.000000       3511        25.6               1
3   8.10   712        2699.958333      33667        73.2               1
4  14.97   667        4066.000000       4740        39.5               0

   delinq.2yrs  pub.rec  not.fully.paid
0            0        0               0
1            0        0               0
2            0        0               0
3            0        0               0
4            1        0               0
```

In [8]:
```
# Get a summary of the dataset
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   credit.policy      9578 non-null   int64
 1   purpose            9578 non-null   object
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64
 11  delinq.2yrs        9578 non-null   int64
 12  pub.rec            9578 non-null   int64
 13  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
None
```

In [9]:
```
# Summary statistics
print(data.describe())
```

```
       credit.policy     int.rate  installment  log.annual.inc          dti  \
count    9578.000000  9578.000000  9578.000000     9578.000000  9578.000000
mean        0.804970     0.122640   319.089413       10.932117    12.606679
std         0.396245     0.026847   207.071301        0.614813     6.883970
min         0.000000     0.060000    15.670000        7.547502     0.000000
25%         1.000000     0.103900   163.770000       10.558414     7.212500
50%         1.000000     0.122100   268.950000       10.928884    12.665000
75%         1.000000     0.140700   432.762500       11.291293    17.950000
max         1.000000     0.216400   940.140000       14.528354    29.960000

              fico  days.with.cr.line     revol.bal   revol.util  \
count  9578.000000        9578.000000  9.578000e+03  9578.000000
mean    710.846314        4560.767197  1.691396e+04    46.799236
std      37.970537        2496.930377  3.375619e+04    29.014417
min     612.000000         178.958333  0.000000e+00     0.000000
25%     682.000000        2820.000000  3.187000e+03    22.600000
50%     707.000000        4139.958333  8.596000e+03    46.300000
75%     737.000000        5730.000000  1.824950e+04    70.900000
max     827.000000       17639.958330  1.207359e+06   119.000000

       inq.last.6mths  delinq.2yrs      pub.rec  not.fully.paid
count     9578.000000  9578.000000  9578.000000     9578.000000
mean         1.577469     0.163708     0.062122        0.160054
std          2.200245     0.546215     0.262126        0.366676
min          0.000000     0.000000     0.000000        0.000000
25%          0.000000     0.000000     0.000000        0.000000
50%          1.000000     0.000000     0.000000        0.000000
75%          2.000000     0.000000     0.000000        0.000000
max         33.000000    13.000000     5.000000        1.000000
```

In [10]:
```
# Check for missing values
print(data.isnull().sum())
```

```
credit.policy        0
purpose              0
int.rate             0
installment          0
log.annual.inc       0
dti                  0
fico                 0
days.with.cr.line    0
revol.bal            0
revol.util           0
inq.last.6mths       0
delinq.2yrs          0
pub.rec              0
not.fully.paid       0
dtype: int64
```

In [11]:
```
# Get the shape of the dataset
print(data.shape)
```

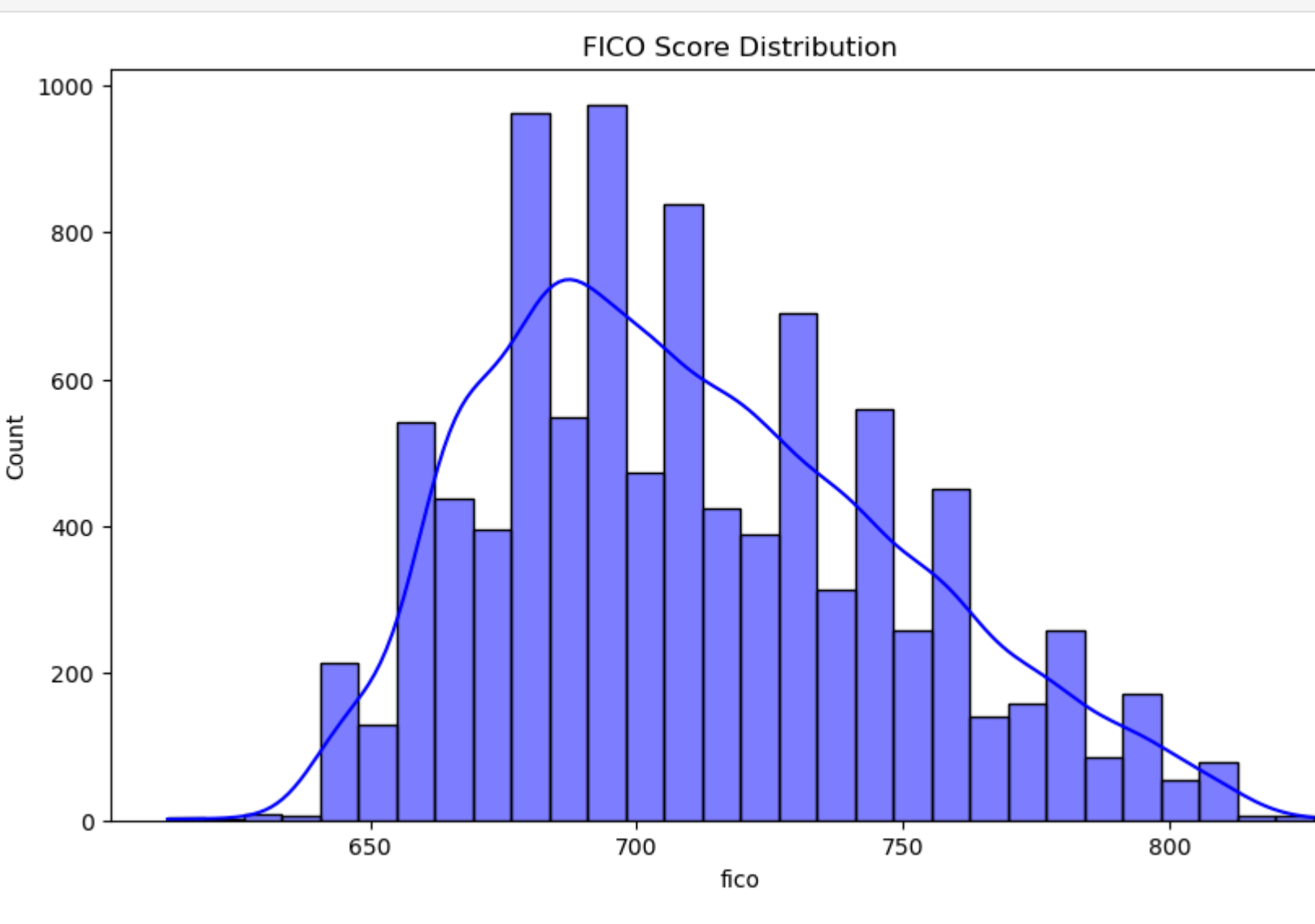
```
(9578, 14)
```

Examine the distribution of individual columns.

Visualizing Distribution

Use histograms to visualize the distribution of key numeric features like 'fico','int.rate', and 'installment'.

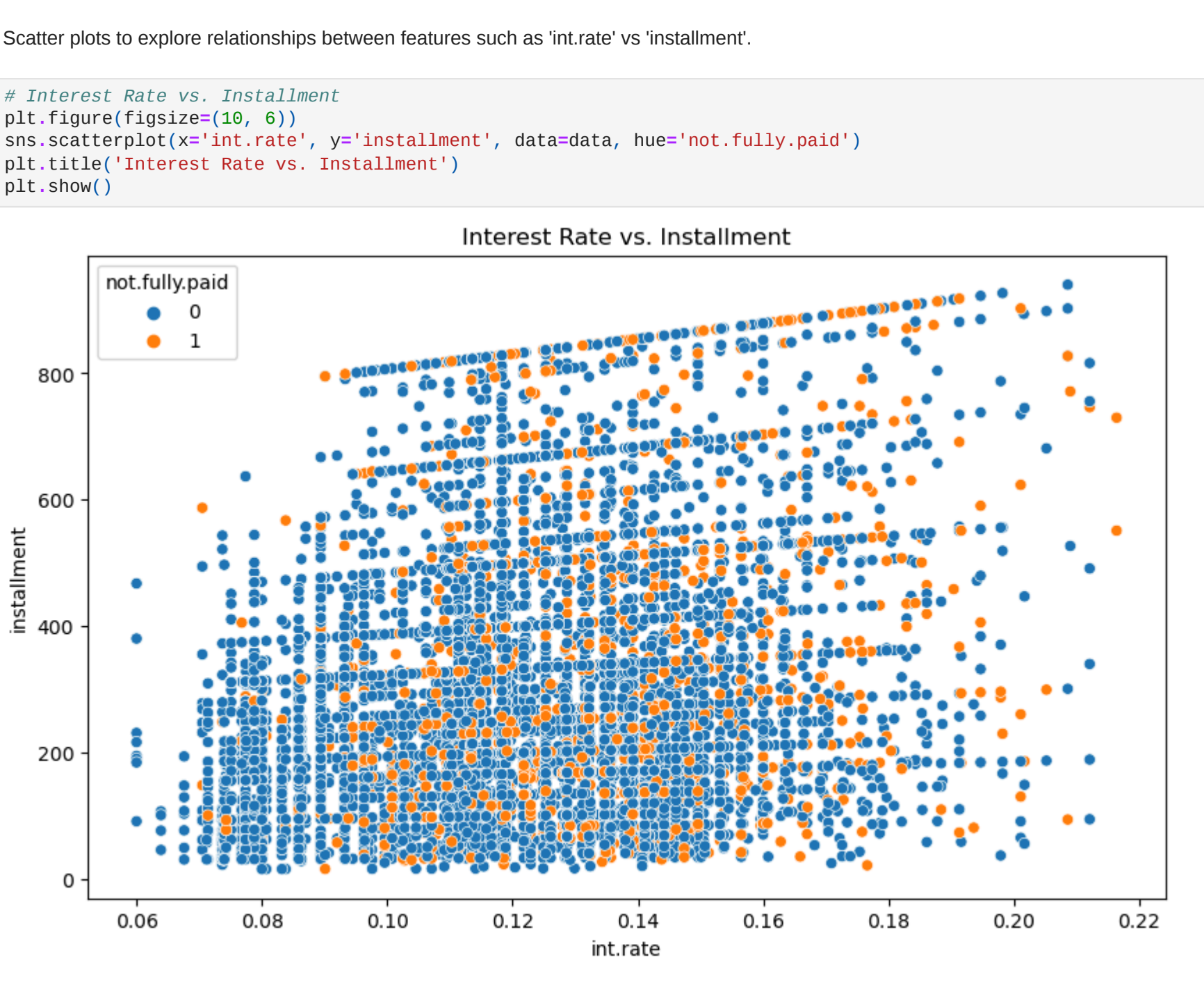
In [12]:
```
import matplotlib.pyplot as plt

# FICO Score Distribution
plt.figure(figsize=(10, 6))
sns.histplot(data['fico'], bins=30, kde=True, color='blue')
plt.title('FICO Score Distribution')
plt.show()
```



Visualizing Relationships

Scatter plots to explore relationships between features such as 'int.rate' vs 'installment'.

In [13]:
```
# Interest Rate vs. Installment
plt.figure(figsize=(10, 6))
sns.scatterplot(x='int.rate', y='installment', data=data, hue='not.fully.paid')
plt.title('Interest Rate vs. Installment')
plt.show()
```

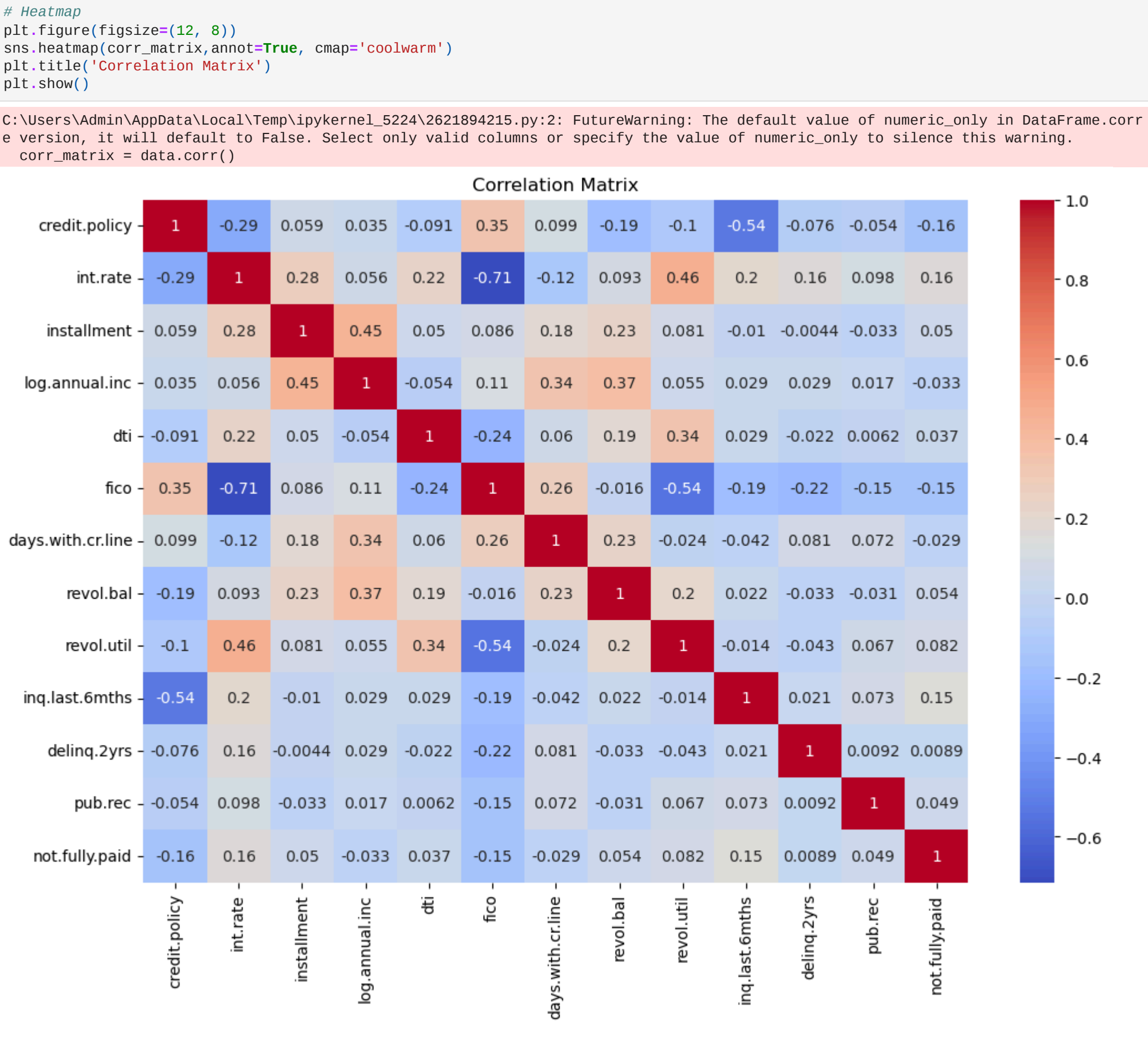


Correlation Analysis

Create a correlation matrix and visualize it using a heatmap to identify relationships between numeric features.

In [17]:
```
# Correlation matrix
corr_matrix = data.corr()

# Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix,annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

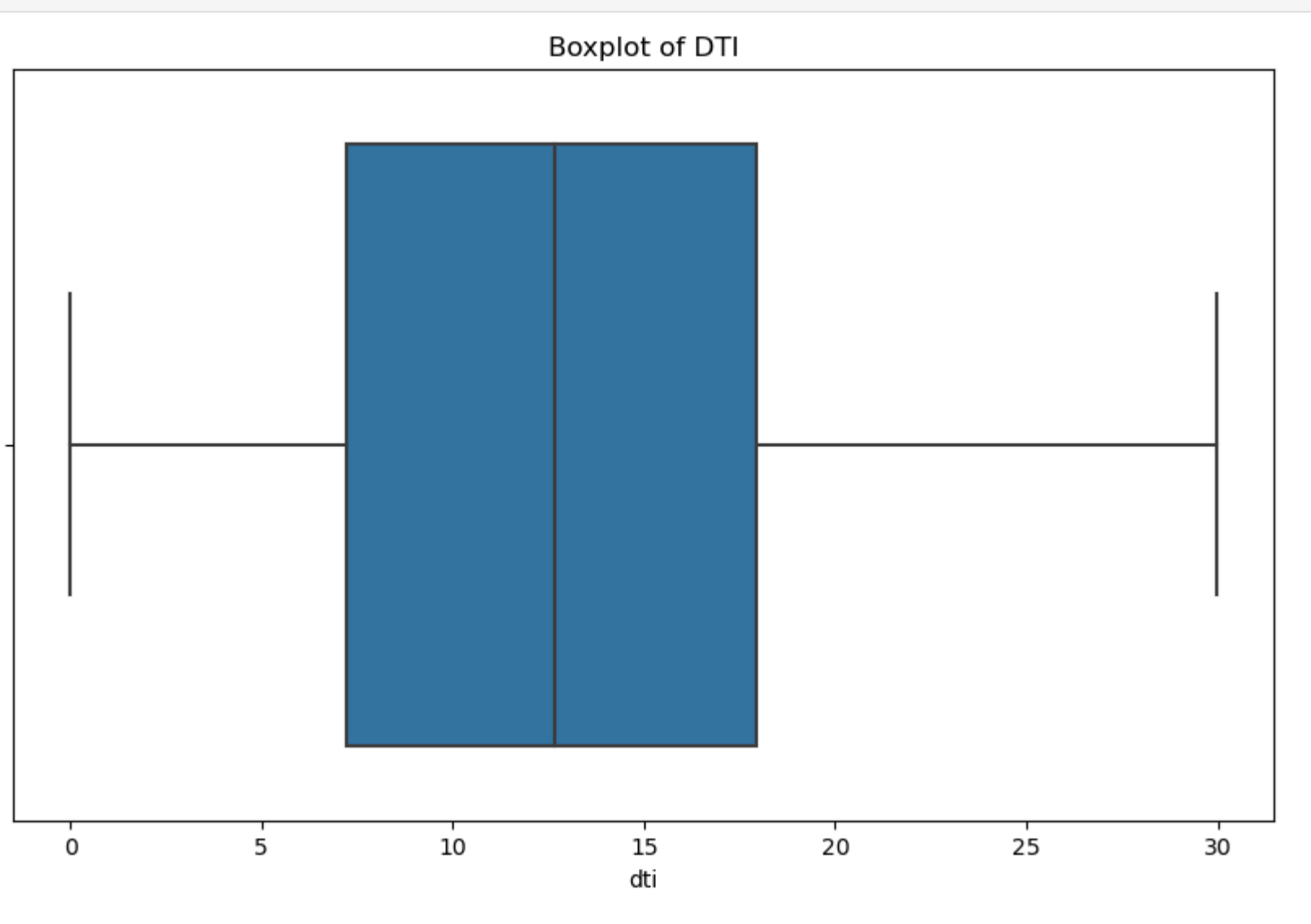
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_5224\2621894215.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a futur
e version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr_matrix = data.corr()
```



Identifying Outliers

Use box plots to detect outliers in the dataset, focusing on features like 'dti' and 'revol.util'

In [18]:
```
# Boxplot for Debt-to-Income Ratio (DTI)
plt.figure(figsize=(10, 6))
sns.boxplot(x='dti', data=data)
plt.title('Boxplot of DTI')
plt.show()
```



Data Cleaning and Preprocessing

In [22]:
```
# Check for missing values
missing_values = data.isnull().sum()

# Drop rows with missing values or fill them
data_cleaned = data.dropna()  # Alternatively, use data.fillna(value) to fill missing data
```

In [23]:
```
data_cleaned
```

Out[23]:

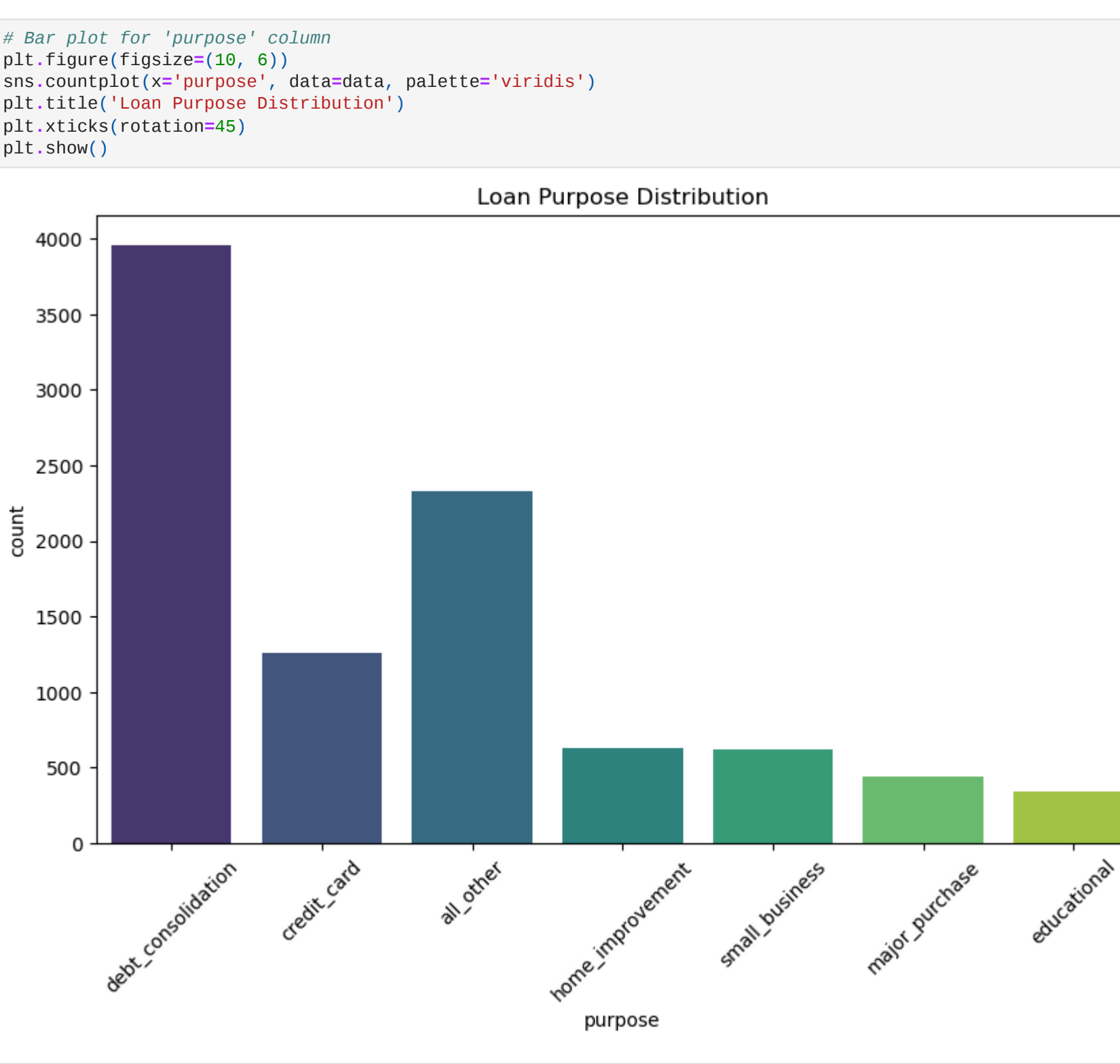
	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not.fully.paid
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	0
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	0
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	0
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	0
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9573	0	all_other	0.1461	344.76	12.180755	10.39	672	10474.000000	215372	82.1	2	0	0	1
9574	0	all_other	0.1253	257.70	11.141862	0.21	722	4380.000000	184	1.1	5	0	0	1
9575	0	debt_consolidation	0.1071	97.81	10.596635	13.09	687	3450.041667	10036	82.9	8	0	0	1
9576	0	home_improvement	0.1600	351.58	10.819778	19.18	692	1800.000000	0	3.2	5	0	0	1
9577	0	debt_consolidation	0.1392	853.43	11.264464	16.28	732	4740.000000	37879	57.0	6	0	0	1

9578 rows × 14 columns

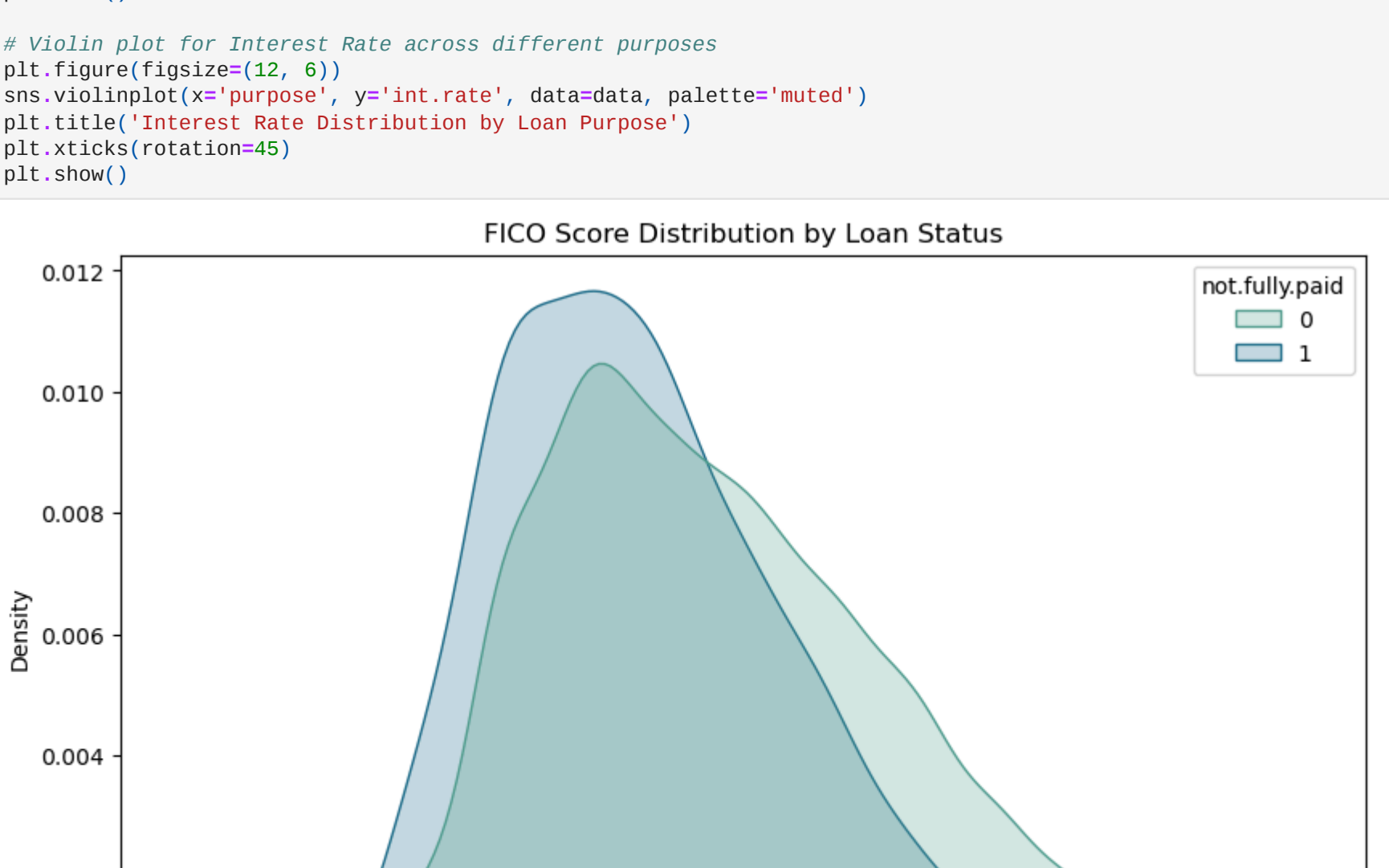
Distribution of Categorical Features

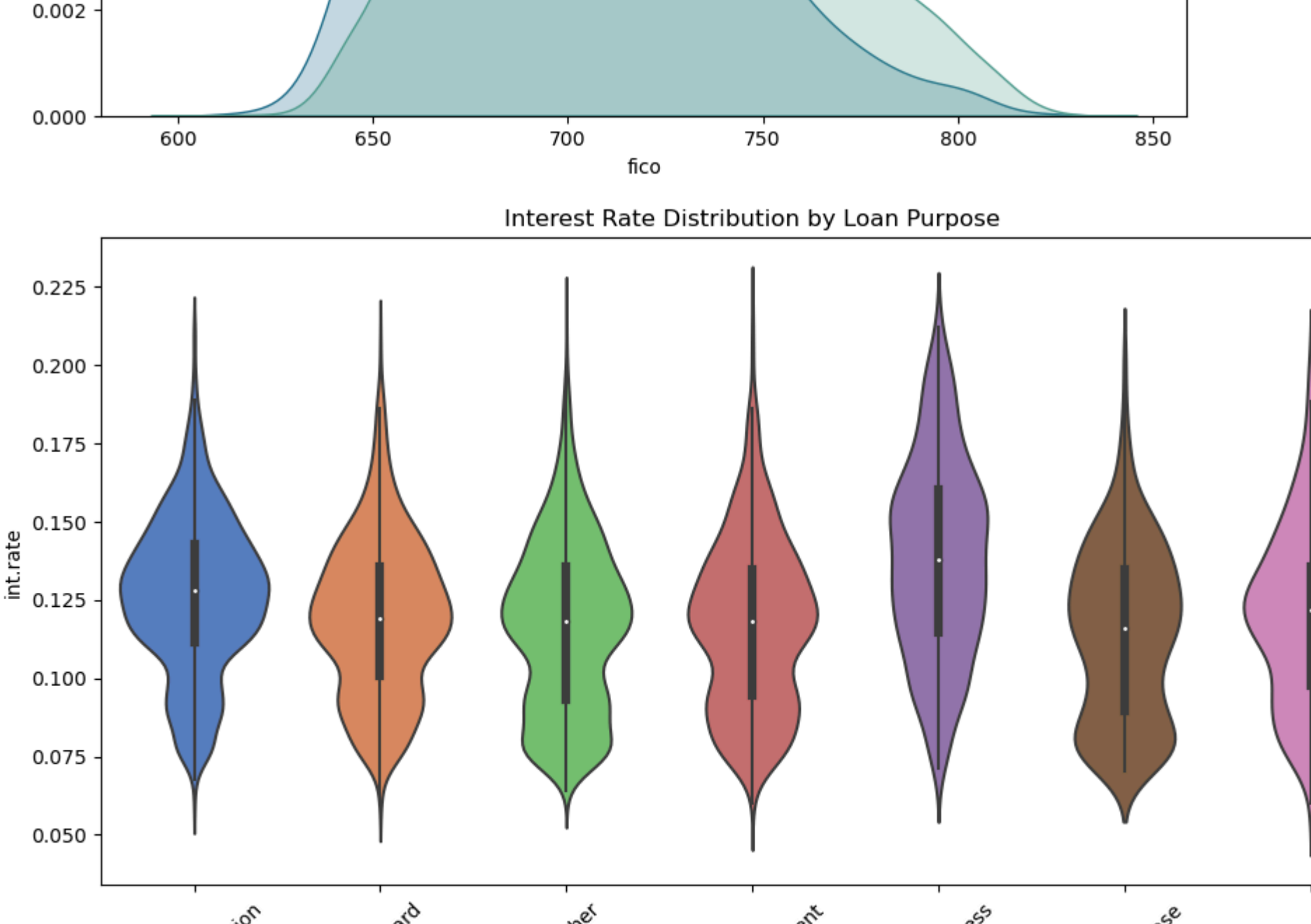
Use bar plots to visualize the distribution of categorical features such as purpose.

In [24]:
```
# Bar plot for 'purpose' column
plt.figure(figsize=(10, 6))
sns.countplot(x='purpose', data=data, palette='viridis')
plt.title('Loan Purpose Distribution')
plt.xticks(rotation=45)
plt.show()
```



In [26]:
```
# KDE plot of FICO scores by loan status
plt.figure(figsize=(10, 6))
sns.kdeplot(data=data, x='fico', hue='not.fully.paid', fill=True, palette='crest', common_norm=False)
plt.title('FICO Score Distribution by Loan Status')
plt.show()

# Violin plot for Interest Rate across different purposes
plt.figure(figsize=(12, 6))
sns.violinplot(x='purpose', y='int.rate', data=data, palette='muted')
plt.title('Interest Rate Distribution by Loan Purpose')
plt.xticks(rotation=45)
plt.show()
```





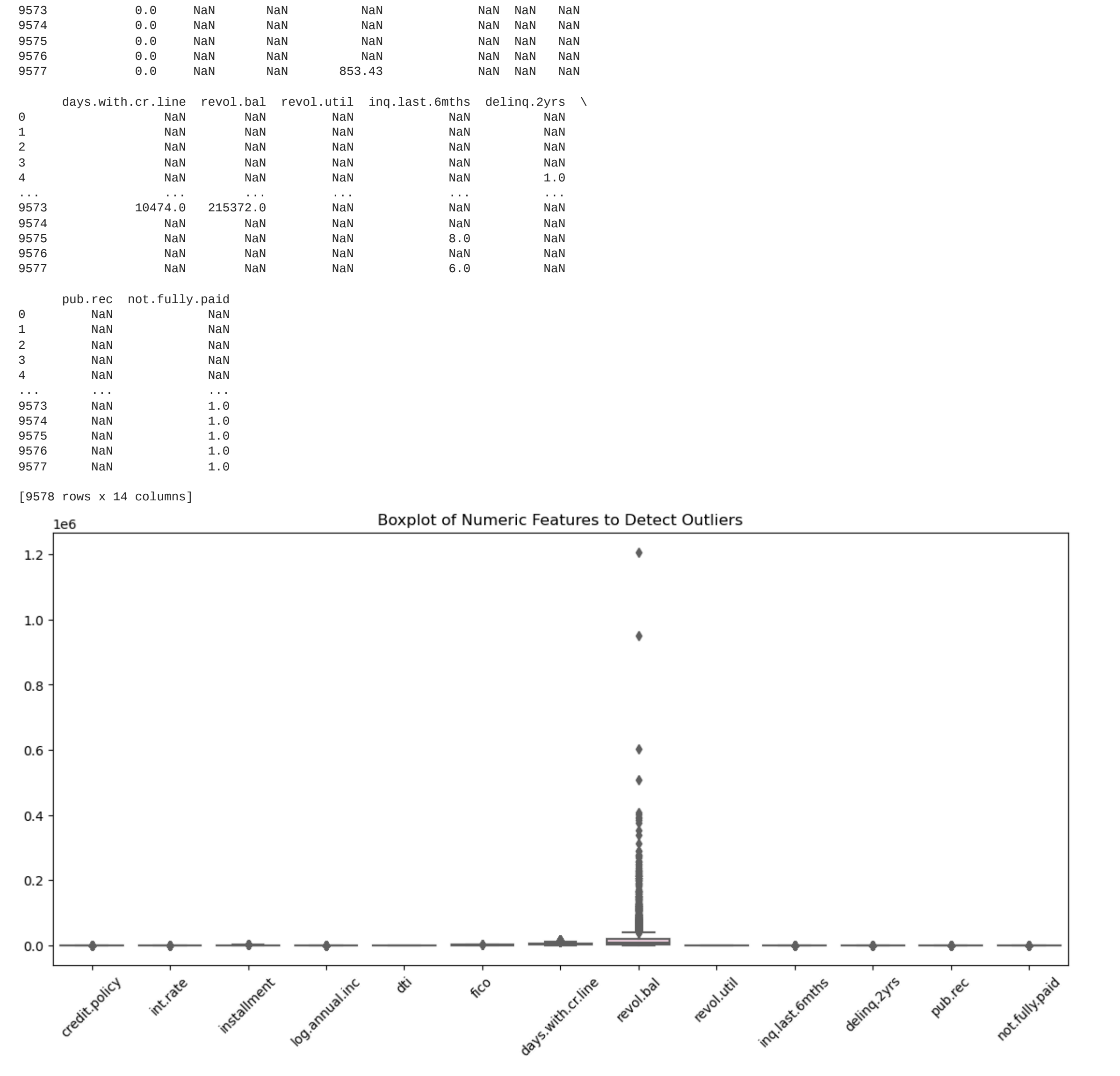
Outlier Detection

Z-Score and IQR methods for detecting outliers in numeric columns.

Visualize outliers using box plots.

In [34]:
```
from scipy import stats

# Define numeric columns
numeric_columns = data.select_dtypes(include=[np.number]).columns.tolist()


# Calculate Z-Scores
z_scores = np.abs(stats.zscore(data[numeric_columns]))
# Identify outliers (Z-Score > 3)
outliers_z = np.where(z_scores > 3)
print("Outliers based on Z-Score method:", outliers_z)


# Calculate IQR
Q1 = data[numeric_columns].quantile(0.25)
Q3 = data[numeric_columns].quantile(0.75)
IQR = Q3 - Q1
# Identify outliers using IQR
outliers_iqr = data[(data[numeric_columns] < (Q1 - 1.5 * IQR)) | (data[numeric_columns] > (Q3 + 1.5 * IQR))]
print("Outliers based on IQR method:")
print(outliers_iqr)



# Boxplot for Outlier Visualization
plt.figure(figsize=(14, 6))
sns.boxplot(data=data[numeric_columns], palette='Set3')
plt.title('Boxplot of Numeric Features to Detect Outliers')
plt.xticks(rotation=45)
plt.show()
```

```
Outliers based on Z-Score method: (array([   6,   10,   24, ..., 9568, 9571, 9573], dtype=int64), array([11, 11, 11, ..., 11, 11,  7], dtype=int64))
Outliers based on IQR method:
      credit.policy purpose  int.rate  installment  log.annual.inc  dti  fico  \
0               NaN     NaN       NaN          NaN             NaN  NaN   NaN
1               NaN     NaN       NaN          NaN             NaN  NaN   NaN
2               NaN     NaN       NaN          NaN             NaN  NaN   NaN
3               NaN     NaN       NaN          NaN             NaN  NaN   NaN
4               NaN     NaN       NaN          NaN             NaN  NaN   NaN
...             ...     ...       ...          ...             ...  ...   ...
9573            0.0     NaN       NaN          NaN             NaN  NaN   NaN
9574            0.0     NaN       NaN          NaN             NaN  NaN   NaN
9575            0.0     NaN       NaN          NaN             NaN  NaN   NaN
9576            0.0     NaN       NaN          NaN             NaN  NaN   NaN
9577            0.0     NaN       NaN       853.43             NaN  NaN   NaN

      days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.2yrs  \
0                   NaN        NaN         NaN             NaN          NaN
1                   NaN        NaN         NaN             NaN          NaN
2                   NaN        NaN         NaN             NaN          NaN
3                   NaN        NaN         NaN             NaN          NaN
4                   NaN        NaN         NaN             NaN          1.0
...                 ...        ...         ...             ...          ...
9573            10474.0   215372.0         NaN             NaN          NaN
9574                NaN        NaN         NaN             NaN          NaN
9575                NaN        NaN         NaN             8.0          NaN
9576                NaN        NaN         NaN             NaN          NaN
9577                NaN        NaN         NaN             6.0          NaN

      pub.rec  not.fully.paid
0         NaN             NaN
1         NaN             NaN
2         NaN             NaN
3         NaN             NaN
4         NaN             NaN
...       ...             ...
9573      NaN             1.0
9574      NaN             1.0
9575      NaN             1.0
9576      NaN             1.0
9577      NaN             1.0

[9578 rows x 14 columns]
```



In [ ]: