```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import (
    accuracy_score, confusion_matrix, classification_report,
    roc_auc_score, roc_curve, auc,
    ConfusionMatrixDisplay, RocCurveDisplay
)
from statsmodels.stats.outliers_influence import variance_inflation_factor
from imblearn.over_sampling import SMOTE
```

```
data = pd.read_csv('/content/loantap_data.csv')
```

```
print("No. of rows : ",data.shape[0])
print("No. of columns : ",data.shape[1])
```

```
No. of rows :  396030
No. of columns :  27
```

```
data.loan_status.value_counts(normalize=True)*100

```

```
loan_status
Fully Paid     80.387092
Charged Off    19.612908
Name: proportion, dtype: float64
```

```
data.describe(include='all')

```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_ti
count	396030.000000	396030	396030.000000	396030.000000	396030	396030	373
unique	NaN	2	NaN	NaN	7	35	173
top	NaN	36 months	NaN	NaN	B	B3	Teac
freq	NaN	302005	NaN	NaN	116018	26655	4
mean	14113.888089	NaN	13.639400	431.849698	NaN	NaN	N
std	8357.441341	NaN	4.472157	250.727790	NaN	NaN	N
min	500.000000	NaN	5.320000	16.080000	NaN	NaN	N
25%	8000.000000	NaN	10.490000	250.330000	NaN	NaN	N
50%	12000.000000	NaN	13.330000	375.430000	NaN	NaN	N
75%	20000.000000	NaN	16.490000	567.300000	NaN	NaN	N
max	40000.000000	NaN	30.990000	1533.810000	NaN	NaN	N

11 rows × 27 columns

```
1 data.info()
2
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object 
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object 
 5   sub_grade             396030 non-null  object 
 6   emp_title             373103 non-null  object 
 7   emp_length            377729 non-null  object 
 8   home_ownership        396030 non-null  object 
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object 
 11  issue_d               396030 non-null  object 
 12  loan_status           396030 non-null  object 
 13  purpose               396030 non-null  object 
 14  title                 394274 non-null  object 
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object 
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object 
```

```
 23  application_type      396030 non-null  object
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

```
1 import warnings
2 warnings.filterwarnings("ignore")
3 # Correlation Heatmap
4 plt.figure(figsize=(12,8))
5 sns.heatmap(data.corr(method='spearman'),annot=True,cmap='viridis')
6 plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-2c7b73063360> in <cell line: 5>()
      3 # Correlation Heatmap
      4 plt.figure(figsize=(12,8))
----> 5 sns.heatmap(data.corr(method='spearman'),annot=True,cmap='viridis')
      6 plt.show()

                                3 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in
_interleave(self, dtype, na_value)
   1792             else:
   1793                 arr = blk.get_values(dtype)
-> 1794             result[rl.indexer] = arr
   1795             itemmask[rl.indexer] = 1
   1796

ValueError: could not convert string to float: ' 36 months'

<Figure size 1200x800 with 0 Axes>
```

```
1 data.drop(columns=['installment'],axis=1,inplace=True)
2
```

```
1 plt.figure(figsize=(12, 8))
2 sns.heatmap(data.corr(method='spearman'), annot=True, cmap='viridis')
3 plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-f3907261aaff> in <cell line: 2>()
      1 plt.figure(figsize=(12, 8))
----> 2 sns.heatmap(data.corr(method='spearman'), annot=True, cmap='viridis')
      3 plt.show()

                                3 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in
_interleave(self, dtype, na_value)
   1792             else:
   1793                 arr = blk.get_values(dtype)
-> 1794             result[rl.indexer] = arr
   1795             itemmask[rl.indexer] = 1
   1796

ValueError: could not convert string to float: ' 36 months'

<Figure size 1200x800 with 0 Axes>
```

```
1 data.groupby(by='loan_status')['loan_amnt'].describe()
2
```

loan_status	count	mean	std	min	25%	50%	75%	m
Charged Off	77673.0	15126.300967	8505.090557	1000.0	8525.0	14000.0	20000.0	4000
Fully Paid	318357.0	13866.878771	8302.319699	500.0	7500.0	12000.0	19225.0	4000

```
1 data['home_ownership'].value_counts()
2
```

```
home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: count, dtype: int64
```

```
1 data.loc[(data.home_ownership == 'ANY') | (data.home_ownership == 'NONE'), 'home_owner
2 data.home_ownership.value_counts()
```

```
home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          146
Name: count, dtype: int64
```

```
1 data['home_ownership'].value_counts()
```

```
home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          146
Name: count, dtype: int64
```

```
1 data.loc[data['home_ownership']=='OTHER','loan_status'].value_counts()
```

```
loan_status
Fully Paid     123
Charged Off     23
Name: count, dtype: int64
```

```
1 data['issue_d']=pd.to_datetime(data['issue_d'])
2 data['earliest_cr_line']=pd.to_datetime(data['earliest_cr_line'])
```

```
1 data['title'].value_counts()[:20]
```

```
title
Debt consolidation           152472
Credit card refinancing       51487
Home improvement              15264
Other                         12930
Debt Consolidation            11608
Major purchase                 4769
Consolidation                  3852
debt consolidation             3547
Business                       2949
Debt Consolidation Loan        2864
Medical expenses               2742
Car financing                  2139
Credit Card Consolidation      1775
Vacation                       1717
Moving and relocation          1689
consolidation                  1595
Personal Loan                  1591
Consolidation Loan             1299
Home Improvement               1268
Home buying                    1183
Name: count, dtype: int64
```

```
1 data['title']=data.title.str.lower()
```

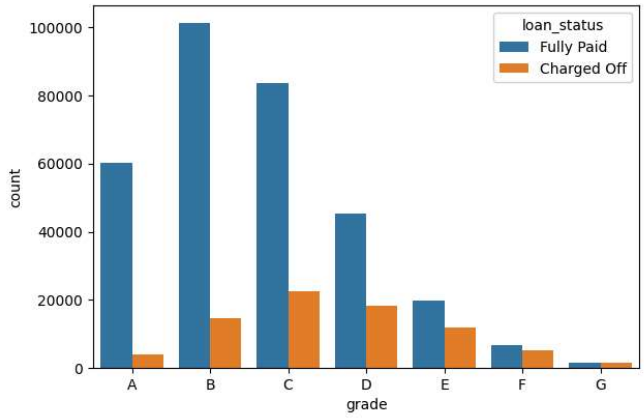
```
1 data['title'].value_counts()[:20]
```

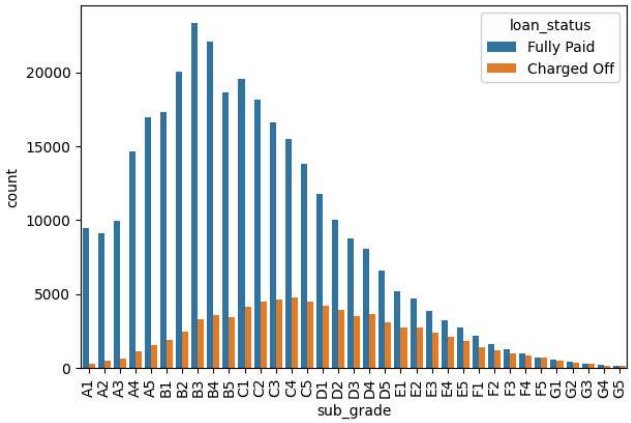
```
title
debt consolidation           168108
credit card refinancing       51781
home improvement              17117
other                         12993
consolidation                  5583
major purchase                 4998
```

```
debt consolidation loan        3513
business                       3017
medical expenses               2820
credit card consolidation      2638
personal loan                  2460
car financing                  2160
credit card payoff             1904
consolidation loan             1887
vacation                       1866
credit card refinance          1832
moving and relocation          1693
consolidate                    1528
personal                       1465
home buying                    1196
Name: count, dtype: int64
```

```
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
grade = sorted(data.grade.unique().tolist())
sns.countplot(x='grade', data=data, hue='loan_status', order=grade)

plt.subplot(2, 2, 2)
sub_grade = sorted(data.sub_grade.unique().tolist())
g = sns.countplot(x='sub_grade', data=data, hue='loan_status', order=sub_grade)
g.set_xticklabels(g.get_xticklabels(), rotation=90)
```

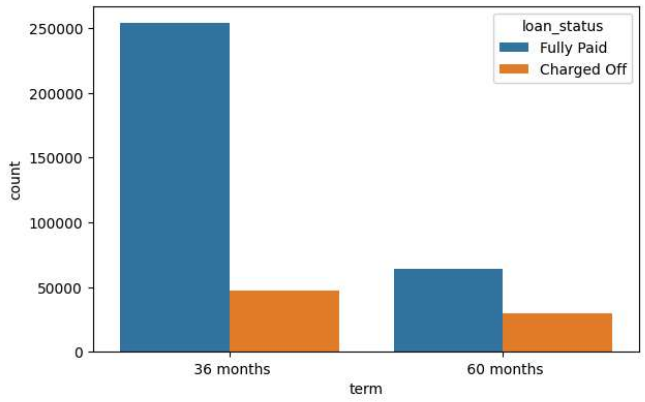
```
[Text(0, 0, 'A1'),
 Text(1, 0, 'A2'),
 Text(2, 0, 'A3'),
 Text(3, 0, 'A4'),
 Text(4, 0, 'A5'),
 Text(5, 0, 'B1'),
 Text(6, 0, 'B2'),
 Text(7, 0, 'B3'),
 Text(8, 0, 'B4'),
 Text(9, 0, 'B5'),
 Text(10, 0, 'C1'),
 Text(11, 0, 'C2'),
 Text(12, 0, 'C3'),
 Text(13, 0, 'C4'),
 Text(14, 0, 'C5'),
 Text(15, 0, 'D1'),
 Text(16, 0, 'D2'),
 Text(17, 0, 'D3'),
 Text(18, 0, 'D4'),
 Text(19, 0, 'D5'),
 Text(20, 0, 'E1'),
 Text(21, 0, 'E2'),
 Text(22, 0, 'E3'),
 Text(23, 0, 'E4'),
 Text(24, 0, 'E5'),
 Text(25, 0, 'F1'),
 Text(26, 0, 'F2'),
 Text(27, 0, 'F3'),
 Text(28, 0, 'F4'),
 Text(29, 0, 'F5'),
 Text(30, 0, 'G1'),
 Text(31, 0, 'G2'),
 Text(32, 0, 'G3'),
 Text(33, 0, 'G4'),
 Text(34, 0, 'G5')]
```

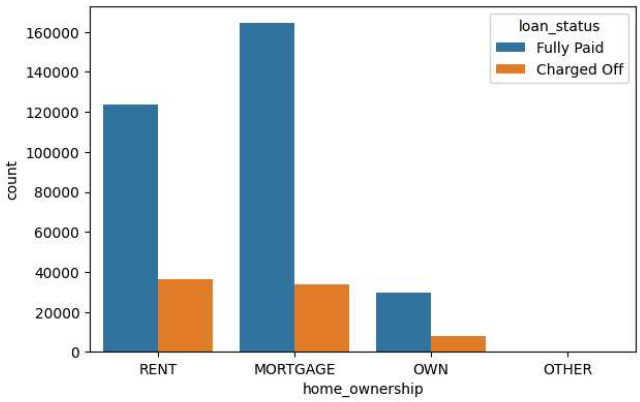


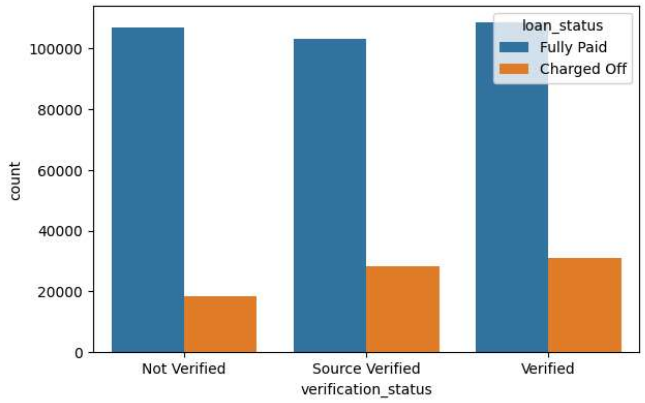


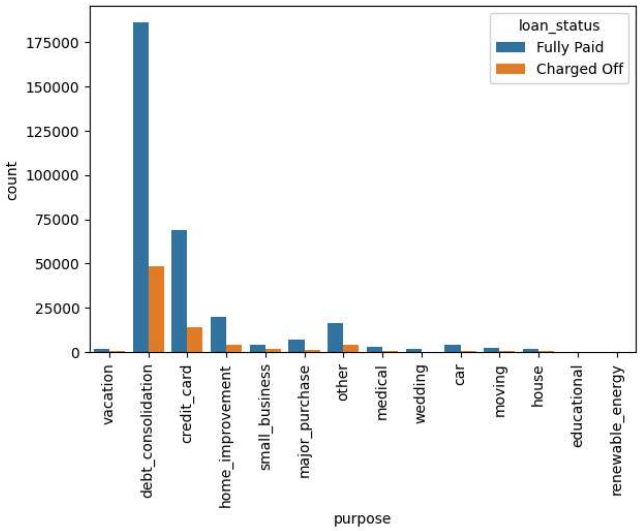
```
plt.figure(figsize=(15,20))

plt.subplot(4,2,1)
sns.countplot(x='term',data=data,hue='loan_status')

plt.subplot(4,2,2)
sns.countplot(x='home_ownership',data=data,hue='loan_status')

plt.subplot(4,2,3)
sns.countplot(x='verification_status',data=data,hue='loan_status')

plt.subplot(4,2,4)
g=sns.countplot(x='purpose',data=data,hue='loan_status')
g.set_xticklabels(g.get_xticklabels(),rotation=90)
```

```
[Text(0, 0, 'vacation'),
 Text(1, 0, 'debt_consolidation'),
 Text(2, 0, 'credit_card'),
 Text(3, 0, 'home_improvement'),
 Text(4, 0, 'small_business'),
 Text(5, 0, 'major_purchase'),
 Text(6, 0, 'other'),
 Text(7, 0, 'medical'),
 Text(8, 0, 'wedding'),
 Text(9, 0, 'car'),
 Text(10, 0, 'moving'),
 Text(11, 0, 'house'),
 Text(12, 0, 'educational'),
 Text(13, 0, 'renewable_energy')]
```

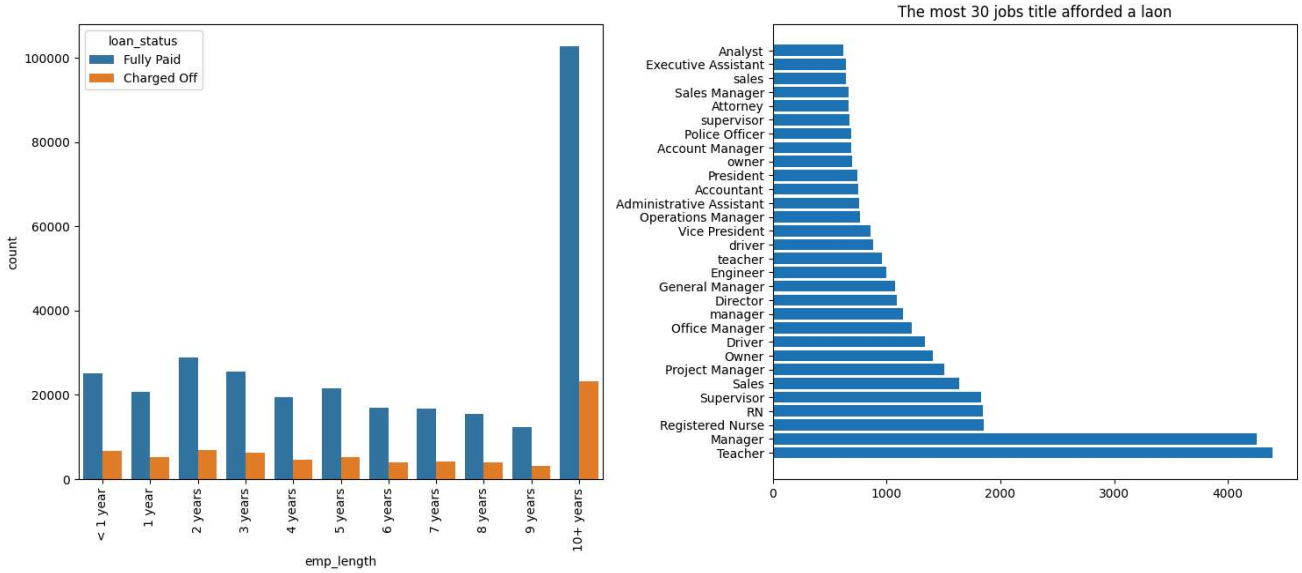








```
plt.figure(figsize=(15,12))

plt.subplot(2,2,1)
order = ['< 1 year', '1 year', '2 years', '3 years', '4 years', '5 years',
         '6 years', '7 years', '8 years', '9 years', '10+ years',]
g=sns.countplot(x='emp_length',data=data,hue='loan_status',order=order)
g.set_xticklabels(g.get_xticklabels(),rotation=90)

plt.subplot(2,2,2)
plt.barh(data.emp_title.value_counts()[:30].index,data.emp_title.value_counts()[:30])
plt.title("The most 30 jobs title afforded a laon")
plt.tight_layout()
```

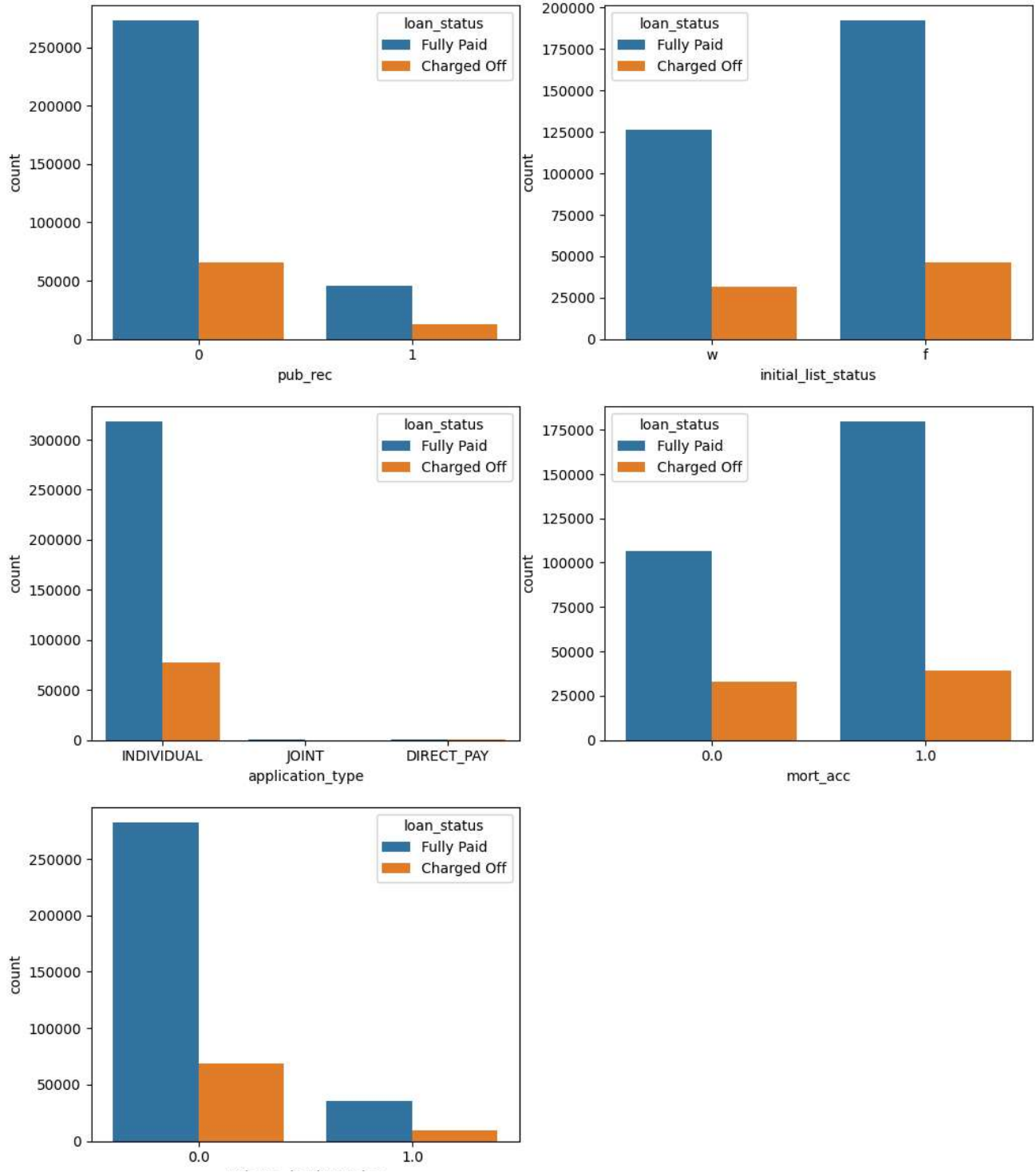



```
def pub_rec(number):
    if number == 0.0:
        return 0
    else:
        return 1

def mort_acc(number):
    if number == 0.0:
        return 0
    elif number >= 1.0:
        return 1
    else:
        return number


def pub_rec_bankruptcies(number):
    if number == 0.0:
        return 0
    elif number >= 1.0:
        return 1
    else:
        return number
```

```
data['pub_rec']=data.pub_rec.apply(pub_rec)
data['mort_acc']=data.mort_acc.apply(mort_acc)
data['pub_rec_bankruptcies']=data.pub_rec_bankruptcies.apply(pub_rec_bankruptcies)
```

```
plt.figure(figsize=(12,30))

plt.subplot(6,2,1)
sns.countplot(x='pub_rec',data=data,hue='loan_status')

plt.subplot(6,2,2)
sns.countplot(x='initial_list_status',data=data,hue='loan_status')

plt.subplot(6,2,3)
sns.countplot(x='application_type',data=data,hue='loan_status')

plt.subplot(6,2,4)
sns.countplot(x='mort_acc',data=data,hue='loan_status')

plt.subplot(6,2,5)
sns.countplot(x='pub_rec_bankruptcies',data=data,hue='loan_status')

```

```
<Axes: xlabel='pub_rec_bankruptcies', ylabel='count'>
```




```
1 data['loan_status']=data.loan_status.map({'Fully Paid':0, 'Charged Off':1})
```

```
1 data.isnull().sum()/len(data)*100
```

```
loan_amnt              0.000000
term                   0.000000
int_rate               0.000000
grade                  0.000000
sub_grade              0.000000
emp_title              5.789208
emp_length             4.621115
home_ownership         0.000000
annual_inc             0.000000
verification_status    0.000000
```

```
issue_d                 0.000000
loan_status             0.000000
purpose                 0.000000
title                   0.443401
dti                     0.000000
earliest_cr_line        0.000000
open_acc                0.000000
pub_rec                 0.000000
revol_bal               0.000000
revol_util              0.069692
total_acc               0.000000
initial_list_status     0.000000
application_type        0.000000
mort_acc                9.543469
pub_rec_bankruptcies    0.135091
address                 0.000000
dtype: float64
```

```
data.groupby(by='total_acc').mean()

```

```
total_acc_avg=data.groupby(by='total_acc').mean().mort_acc
```

```
def fill_mort_acc(total_acc,mort_acc):
    if np.isnan(mort_acc):
        return total_acc_avg[total_acc].round()
    else:
        return mort_acc
```

```
data['mort_acc']=data.apply(lambda x: fill_mort_acc(x['total_acc'],x['mort_acc']),axis
```

```
data.isnull().sum()/len(data)*100
```

```
loan_amnt               0.000000
term                    0.000000
int_rate                0.000000
grade                   0.000000
sub_grade               0.000000
emp_title               5.789208
emp_length              4.621115
home_ownership          0.000000
annual_inc              0.000000
verification_status     0.000000
issue_d                 0.000000
loan_status             0.000000
purpose                 0.000000
title                   0.443401
dti                     0.000000
earliest_cr_line        0.000000
open_acc                0.000000
pub_rec                 0.000000
revol_bal               0.000000
revol_util              0.069692
total_acc               0.000000
initial_list_status     0.000000
```

```
application_type        0.000000
mort_acc                9.543469
pub_rec_bankruptcies    0.135091
address                 0.000000
dtype: float64
```

```
# Current no. of rows
data.shape
```

```
(396030, 26)
```

```
data.dropna(inplace=True)
```

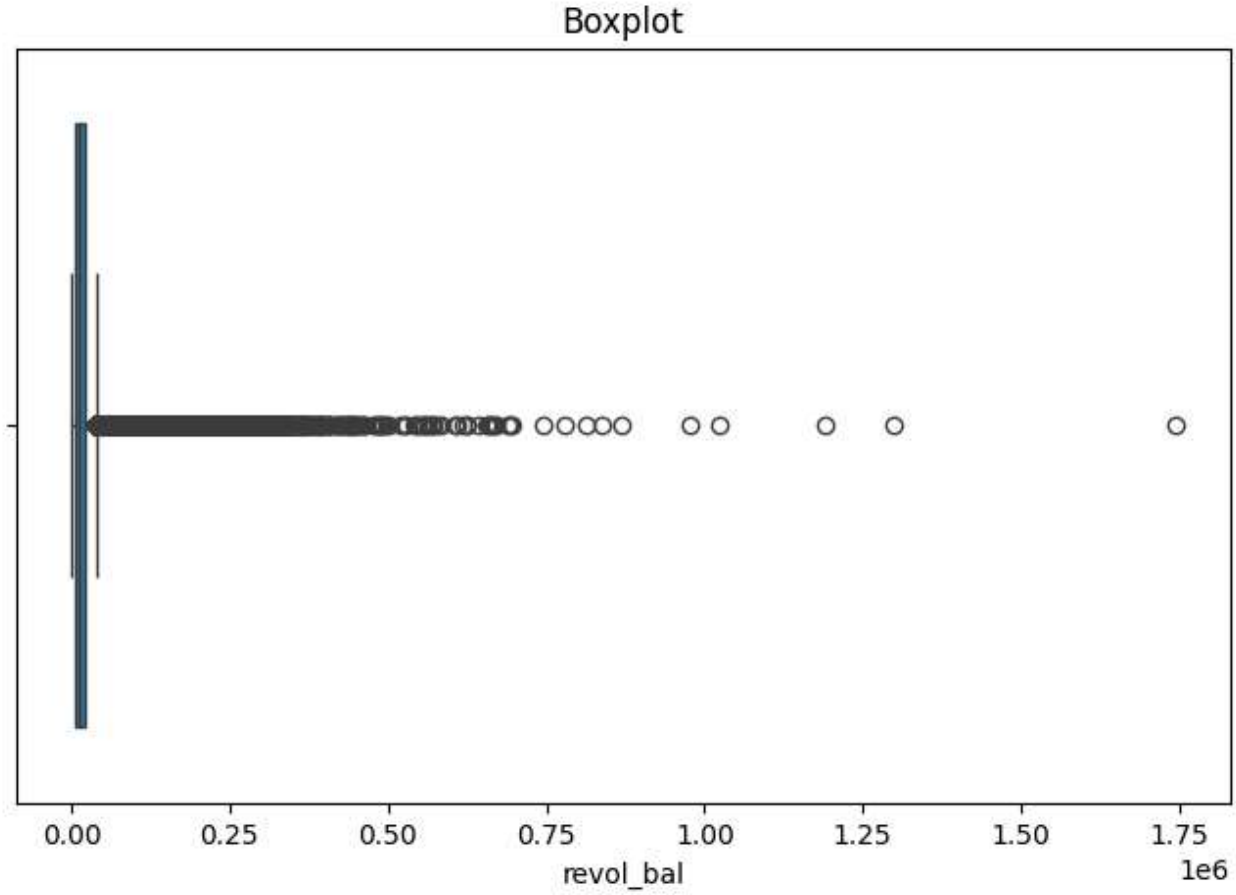
```
data.shape
```

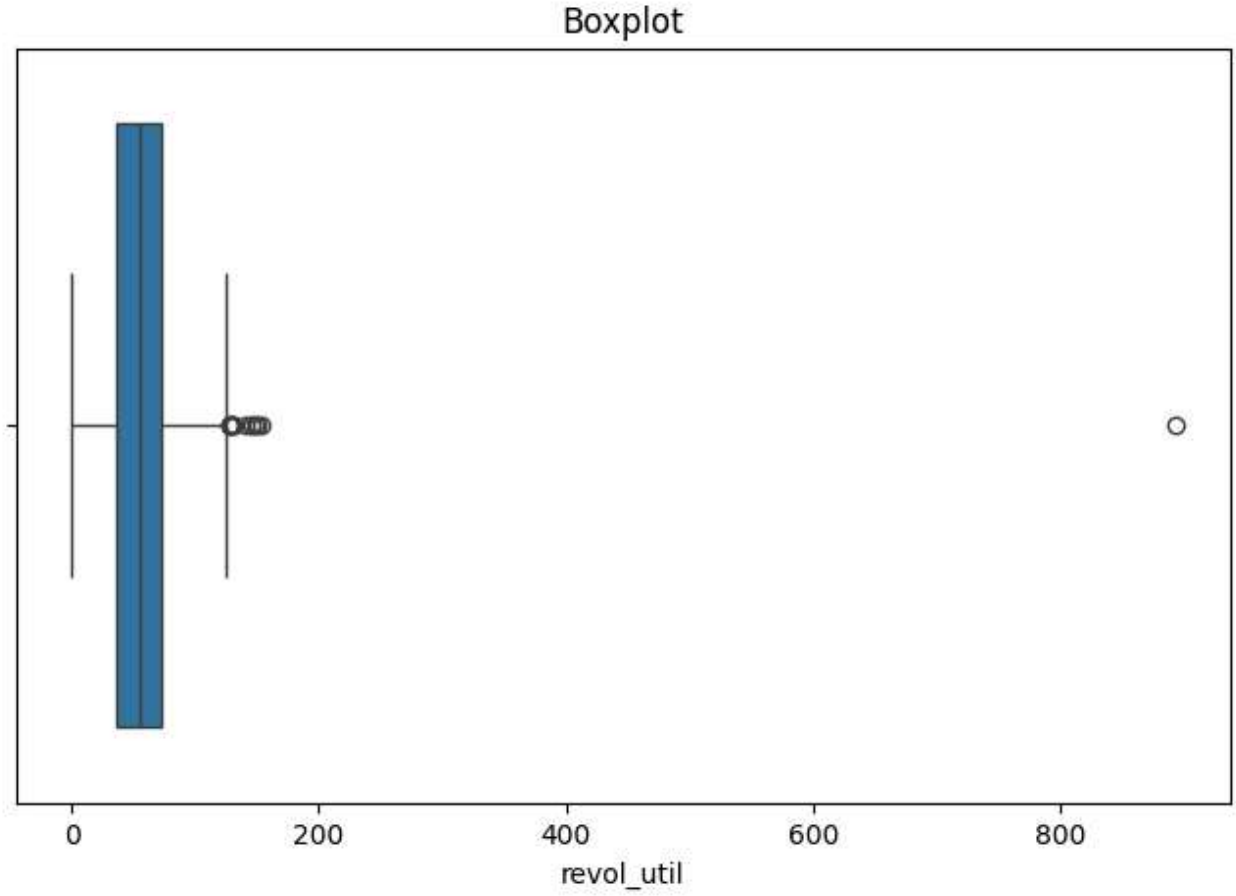
```
(335867, 26)
```

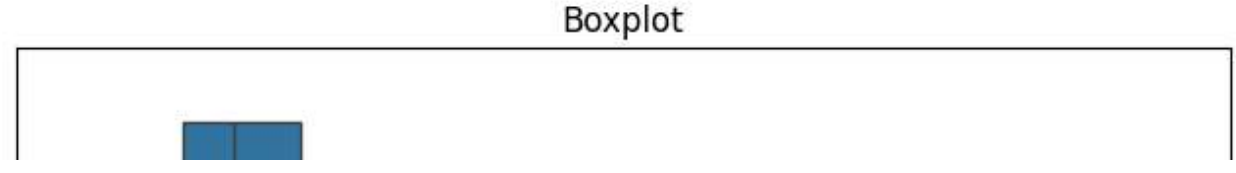
```
numerical_data=data.select_dtypes(include='number')
num_cols=numerical_data.columns
len(num_cols)
```

```
12
```

```
def box_plot(col):
    plt.figure(figsize=(8,5))
    sns.boxplot(x=data[col])
    plt.title('Boxplot')
    plt.show()

for col in num_cols:
    box_plot(col)
```


Boxplot
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
revol_bal
1e6


Boxplot
0
200
400
600
800
revol_util


Boxplot