```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

df = pd.read_csv('lending_club_loan_two.csv')

df.head()
```

```
   loan_amnt        term  int_rate  installment grade sub_grade  \
0    10000.0   36 months     11.44       329.48     B        B4
1     8000.0   36 months     11.99       265.68     B        B5
2    15600.0   36 months     10.49       506.97     B        B3
3     7200.0   36 months      6.49       220.65     A        A2
4    24375.0   60 months     17.27       609.33     C        C5

                  emp_title emp_length home_ownership  annual_inc  ...
\
0                 Marketing  10+ years           RENT    117000.0  ...

1           Credit analyst     4 years       MORTGAGE     65000.0  ...

2              Statistician   < 1 year           RENT     43057.0  ...

3           Client Advocate    6 years           RENT     54000.0  ...

4  Destiny Management Inc.    9 years       MORTGAGE     55000.0  ...


  open_acc pub_rec revol_bal revol_util total_acc  initial_list_status
\
0     16.0     0.0   36369.0       41.8      25.0                    w

1     17.0     0.0   20131.0       53.3      27.0                    f

2     13.0     0.0   11987.0       92.2      26.0                    f

3      6.0     0.0    5472.0       21.5      13.0                    f

4     13.0     0.0   24584.0       69.8      43.0                    f


  application_type  mort_acc  pub_rec_bankruptcies  \
0       INDIVIDUAL       0.0                   0.0
1       INDIVIDUAL       3.0                   0.0
2       INDIVIDUAL       0.0                   0.0
3       INDIVIDUAL       0.0                   0.0
4       INDIVIDUAL       1.0                   0.0
```

```
                                                address
0     0174 Michelle Gateway\nMendozaberg, OK 22690
1  1076 Carney Fort Apt. 347\nLoganmouth, SD 05113
2  87025 Mark Dale Apt. 269\nNew Sabrina, WV 05113
3            823 Reid Ford\nDelacruzside, MA 00813
4             679 Luna Roads\nGreggshire, VA 11650

[5 rows x 27 columns]
```

```
df.info()
```

```
<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
```

```
df.describe()
```

```
           loan_amnt       int_rate    installment    annual_inc  \
count  396030.000000  396030.000000  396030.000000  3.960300e+05
```

```
mean    14113.888089      13.639400     431.849698  7.420318e+04
std      8357.441341       4.472157     250.727790  6.163762e+04
min       500.000000       5.320000      16.080000  0.000000e+00
25%      8000.000000      10.490000     250.330000  4.500000e+04
50%     12000.000000      13.330000     375.430000  6.400000e+04
75%     20000.000000      16.490000     567.300000  9.000000e+04
max     40000.000000      30.990000    1533.810000  8.706582e+06

                dti       open_acc        pub_rec     revol_bal  \
count  396030.000000  396030.000000  396030.000000  3.960300e+05
mean       17.379514      11.311153       0.178191  1.584454e+04
std        18.019092       5.137649       0.530671  2.059184e+04
min         0.000000       0.000000       0.000000  0.000000e+00
25%        11.280000       8.000000       0.000000  6.025000e+03
50%        16.910000      10.000000       0.000000  1.118100e+04
75%        22.980000      14.000000       0.000000  1.962000e+04
max      9999.000000      90.000000      86.000000  1.743266e+06

          revol_util      total_acc       mort_acc
pub_rec_bankruptcies
count  395754.000000  396030.000000  358235.000000
395495.000000
mean       53.791749      25.414744       1.813991
0.121648
std        24.452193      11.886991       2.147930
0.356174
min         0.000000       2.000000       0.000000
0.000000
25%        35.800000      17.000000       0.000000
0.000000
50%        54.800000      24.000000       1.000000
0.000000
75%        72.900000      32.000000       3.000000
0.000000
max       892.300000     151.000000      34.000000
8.000000
```

```
df['purpose'].value_counts()
```

```
purpose
debt_consolidation    234507
credit_card            83019
home_improvement       24030
other                  21185
major_purchase          8790
small_business          5701
car                     4697
medical                 4196
moving                  2854
vacation                2452
```

```
house                   2201
wedding                 1812
renewable_energy         329
educational              257
Name: count, dtype: int64
```

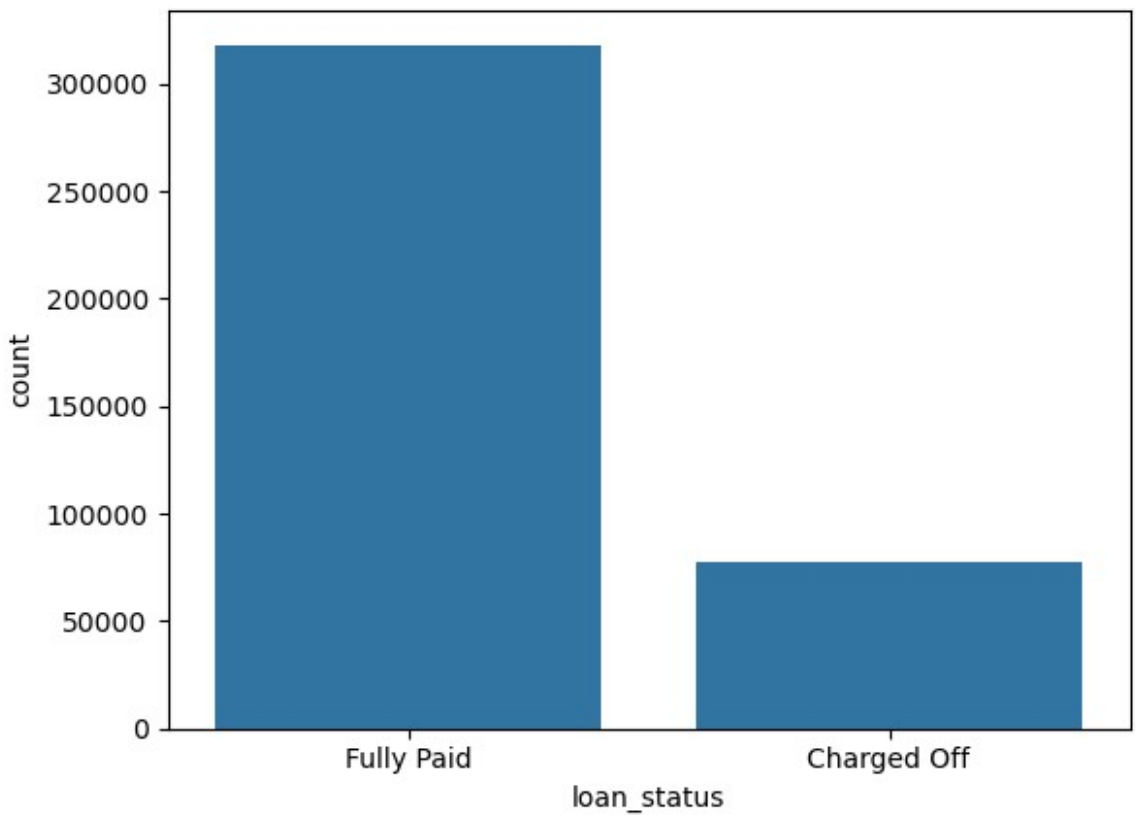
```
df['loan_status'].value_counts()
```

```
loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64
```

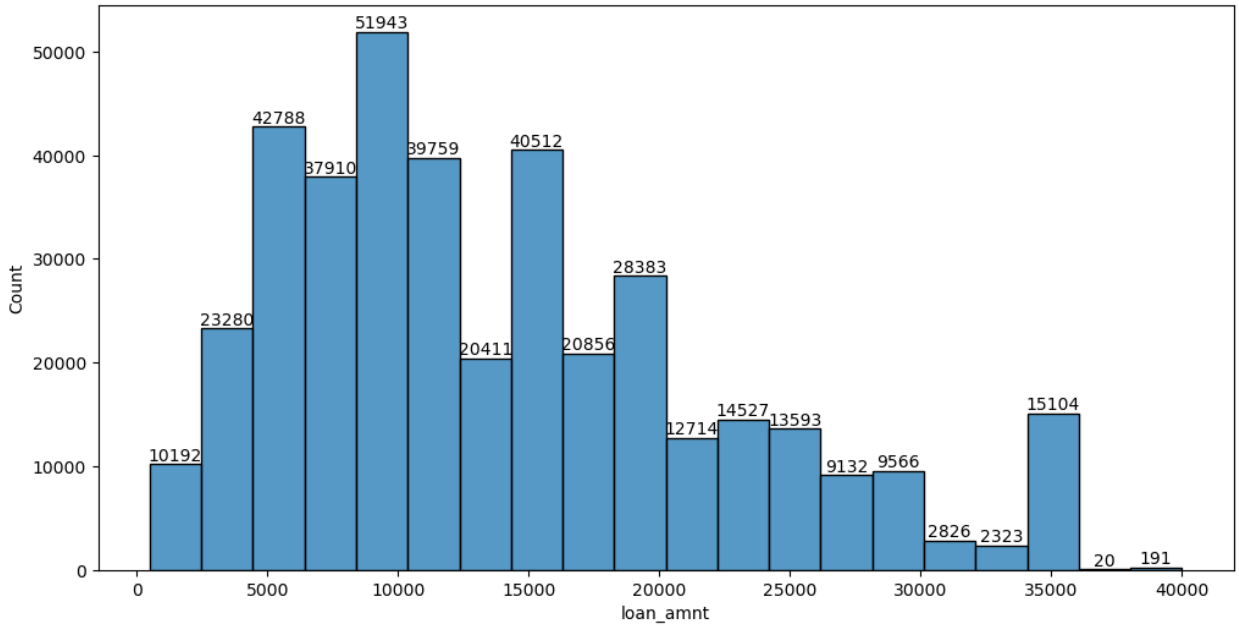
```
df['annual_inc'].sort_values()
```

```
285674          0.0
350865        600.0
7011         2500.0
72405        4000.0
127390       4000.0
             ...
100946    7000000.0
376306    7141778.0
100370    7446395.0
318255    7600000.0
308700    8706582.0
Name: annual_inc, Length: 396030, dtype: float64
```

```
sns.countplot(x='loan_status', data = df,stat='count')
plt.show()
```




```
plt.figure(figsize=(12, 6))
ax = sns.histplot(df['loan_amnt'], bins=20, kde=False)

# Annotate each bin with its count
for p in ax.patches:
    height = p.get_height()  # Get the height of the bin (count)
    if height > 0:  # Only annotate if the count is greater than 0
        ax.text(
            p.get_x() + p.get_width() / 2.,  # X position (center of
the bin)
            height + 5,  # Y position (slightly above the bin)
            f'{int(height)}',  # Text to display (count)
            ha='center',  # Horizontal alignment
            va='bottom'  # Vertical alignment
        )

plt.show()
```




```
df['home_ownership'].value_counts()
```

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

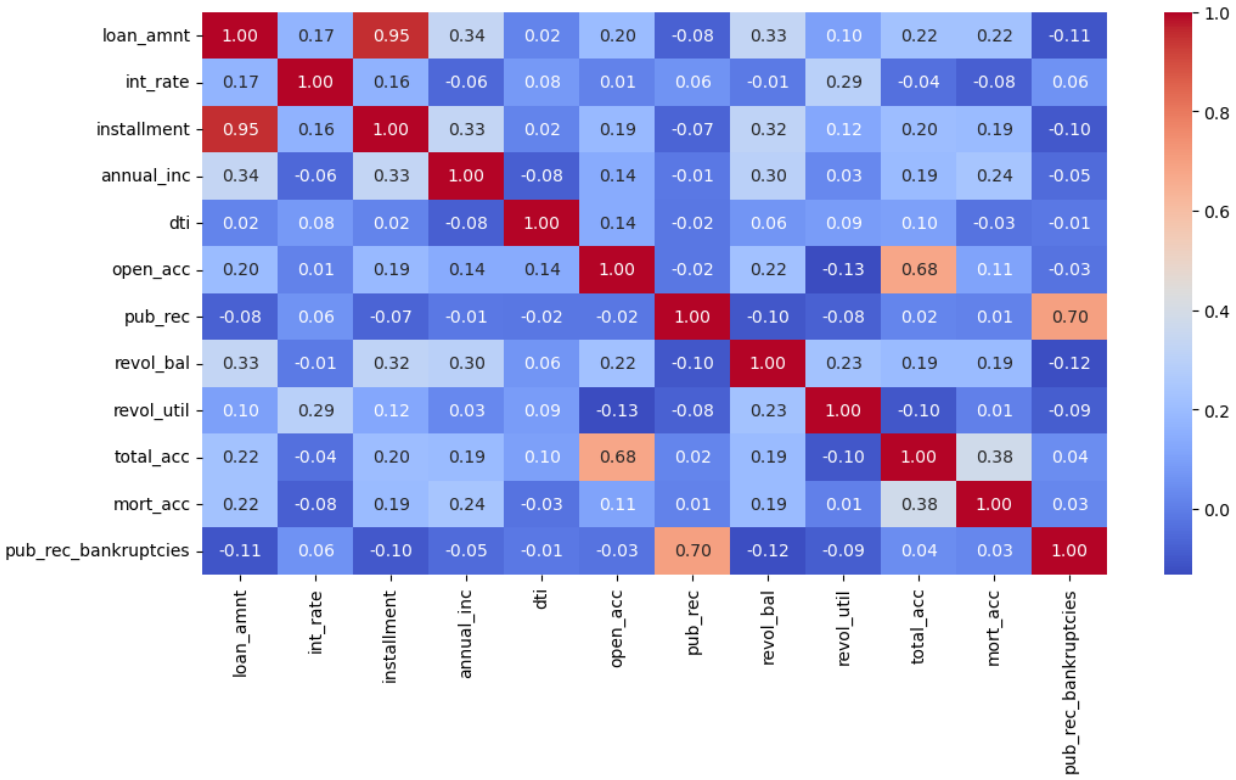
```
df['term'] = df['term'].apply(lambda x: x.split()[0])

numeric_df = df.select_dtypes(include=['float64', 'int64'])
numeric_df.corr()
```

```
                     loan_amnt  int_rate  installment  annual_inc
dti  \
loan_amnt             1.000000  0.168921     0.953929    0.336887
0.016636
int_rate              0.168921  1.000000     0.162758   -0.056771
0.079038
installment           0.953929  0.162758     1.000000    0.330381
0.015786
annual_inc            0.336887 -0.056771     0.330381    1.000000 -
0.081685
dti                   0.016636  0.079038     0.015786   -0.081685
1.000000
open_acc              0.198556  0.011649     0.188973    0.136150
0.136181
pub_rec              -0.077779  0.060986    -0.067892   -0.013720 -
```

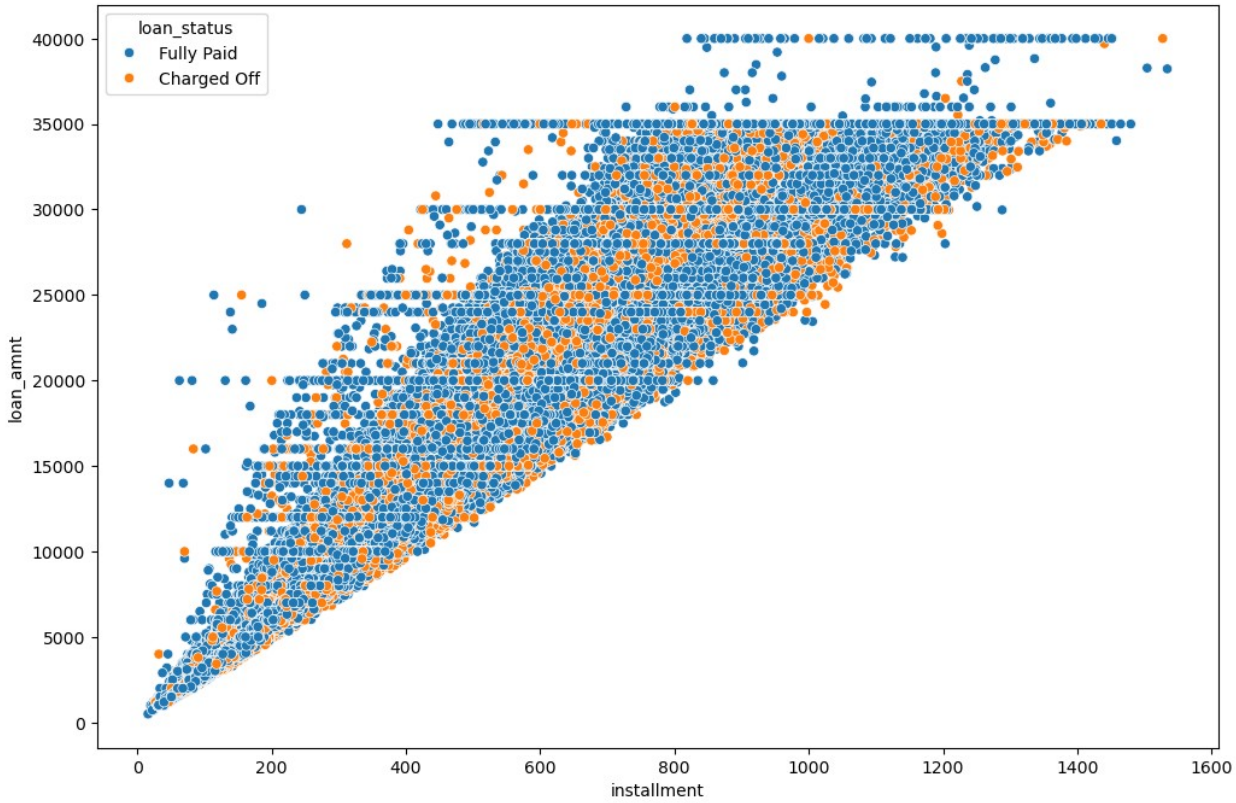
```
0.017639
revol_bal              0.328320 -0.011280     0.316455    0.299773
0.063571
revol_util             0.099911  0.293659     0.123915    0.027871
0.088375
total_acc              0.223886 -0.036404     0.202430    0.193023
0.102128
mort_acc               0.222315 -0.082583     0.193694    0.236320 -
0.025439
pub_rec_bankruptcies  -0.106539  0.057450    -0.098628   -0.050162 -
0.014558

                      open_acc   pub_rec  revol_bal  revol_util
total_acc  \
loan_amnt             0.198556 -0.077779   0.328320    0.099911
0.223886
int_rate              0.011649  0.060986  -0.011280    0.293659  -
0.036404
installment           0.188973 -0.067892   0.316455    0.123915
0.202430
annual_inc            0.136150 -0.013720   0.299773    0.027871
0.193023
dti                   0.136181 -0.017639   0.063571    0.088375
0.102128
open_acc              1.000000 -0.018392   0.221192   -0.131420
0.680728
pub_rec              -0.018392  1.000000  -0.101664   -0.075910
0.019723
revol_bal             0.221192 -0.101664   1.000000    0.226346
0.191616
revol_util           -0.131420 -0.075910   0.226346    1.000000  -
0.104273
total_acc             0.680728  0.019723   0.191616   -0.104273
1.000000
mort_acc              0.109205  0.011552   0.194925    0.007514
0.381072
pub_rec_bankruptcies -0.027732  0.699408  -0.124532   -0.086751
0.042035

                      mort_acc  pub_rec_bankruptcies
loan_amnt             0.222315             -0.106539
int_rate             -0.082583              0.057450
installment           0.193694             -0.098628
annual_inc            0.236320             -0.050162
dti                  -0.025439             -0.014558
open_acc              0.109205             -0.027732
pub_rec               0.011552              0.699408
revol_bal             0.194925             -0.124532
revol_util            0.007514             -0.086751
```

```
total_acc             0.381072              0.042035
mort_acc              1.000000              0.027239
pub_rec_bankruptcies  0.027239              1.000000
```

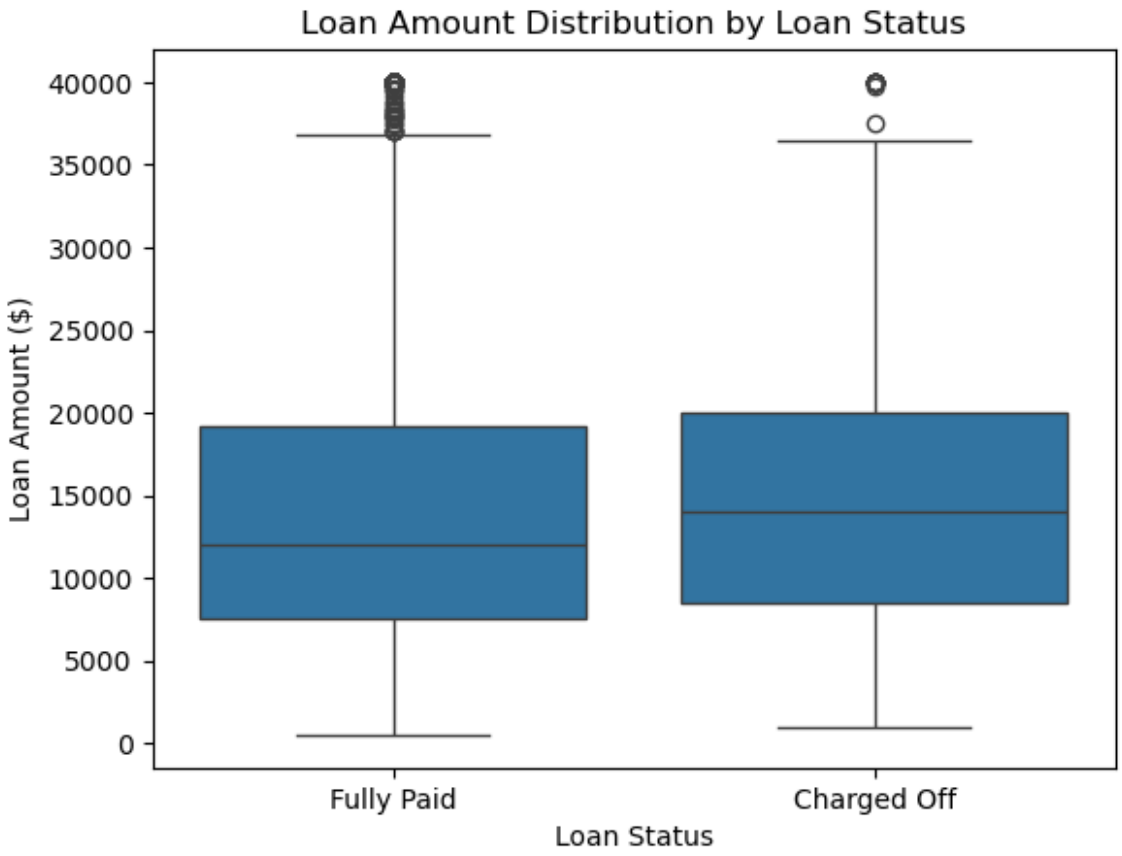
```
plt.figure(figsize=(12,6))
sns.heatmap(numeric_df.corr(),annot=True,cmap='coolwarm',fmt='.2f')
plt.show()
```




```
plt.figure(figsize=(12,8))
sns.scatterplot(y='loan_amnt',x='installment',data=df,hue='loan_status
')
plt.show()
```




```
sns.boxplot(x='loan_status',y='loan_amnt',data=df)
plt.title('Loan Amount Distribution by Loan Status')
plt.xlabel('Loan Status')
plt.ylabel('Loan Amount ($)')
plt.show()
```




```
df.groupby('loan_status')['loan_amnt'].describe()
```

```
                count          mean           std     min     25%
50%  \
loan_status

Charged Off   77673.0  15126.300967  8505.090557  1000.0  8525.0
14000.0
Fully Paid   318357.0  13866.878771  8302.319699   500.0  7500.0
12000.0

                 75%      max
loan_status
Charged Off  20000.0  40000.0
Fully Paid   19225.0  40000.0
```

```
sorted(df['grade'].unique())
```

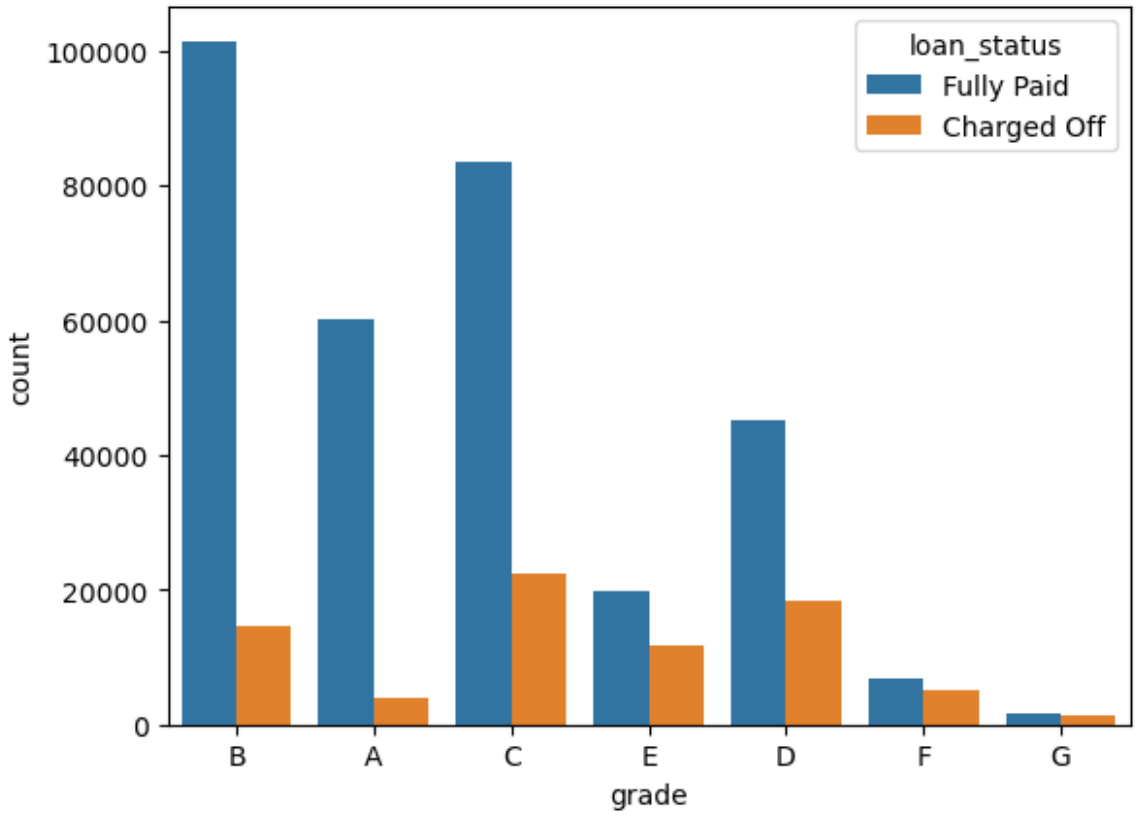
```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

```
sorted(df['sub_grade'].unique())
```

```
['A1',
 'A2',
```

```
 'A3',
 'A4',
 'A5',
 'B1',
 'B2',
 'B3',
 'B4',
 'B5',
 'C1',
 'C2',
 'C3',
 'C4',
 'C5',
 'D1',
 'D2',
 'D3',
 'D4',
 'D5',
 'E1',
 'E2',
 'E3',
 'E4',
 'E5',
 'F1',
 'F2',
 'F3',
 'F4',
 'F5',
 'G1',
 'G2',
 'G3',
 'G4',
 'G5']
```

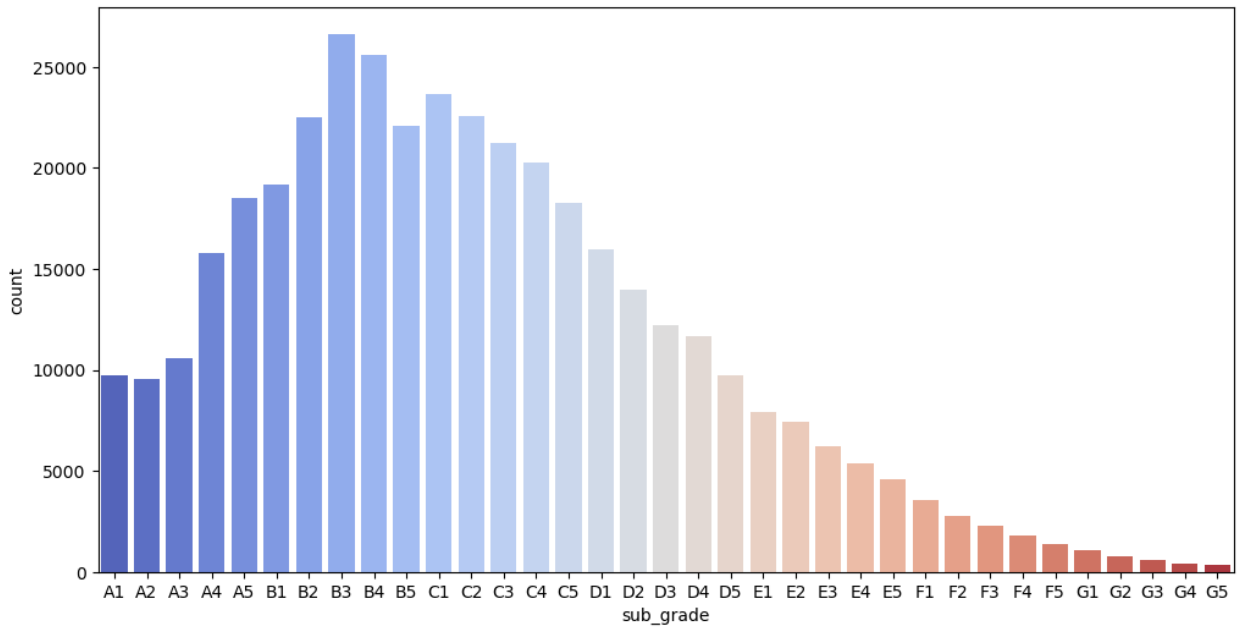
```
sns.countplot(x='grade',data=df,hue='loan_status')
plt.show()
```



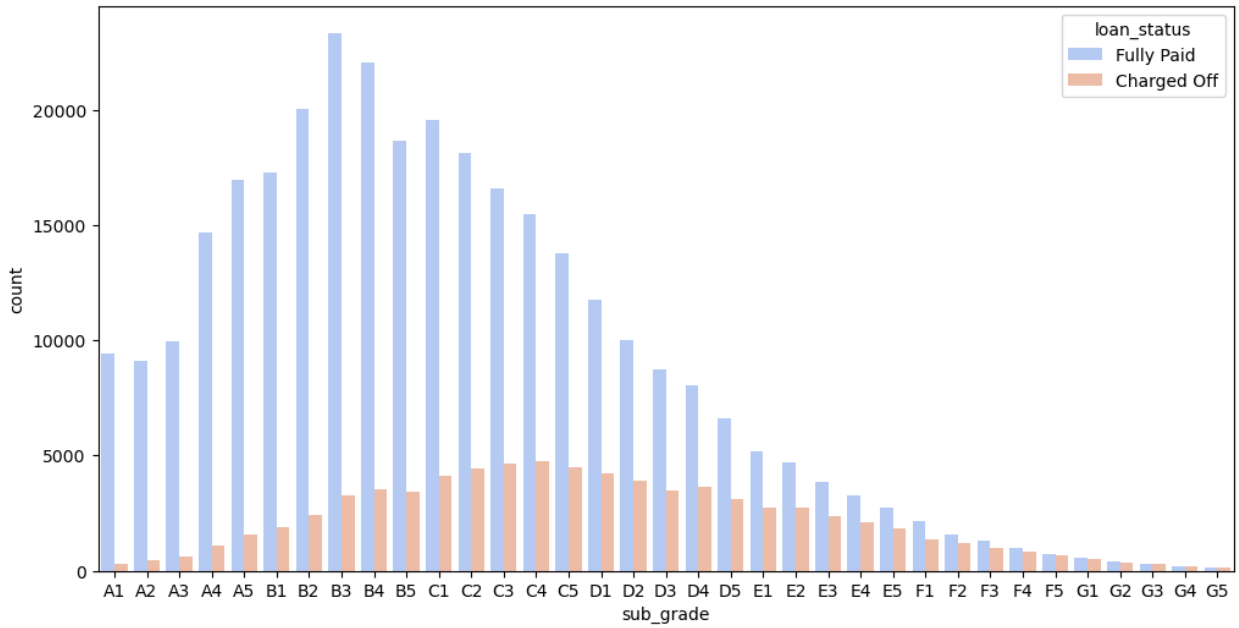

```
plt.figure(figsize=(12,6))
sns.countplot(x='sub_grade',data=df,order=sorted(df['sub_grade'].uniqu
e()),palette='coolwarm')
plt.show()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_5000\1691869283.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
sns.countplot(x='sub_grade',data=df,order=sorted(df['sub_grade'].uniqu
e()),palette='coolwarm')
```

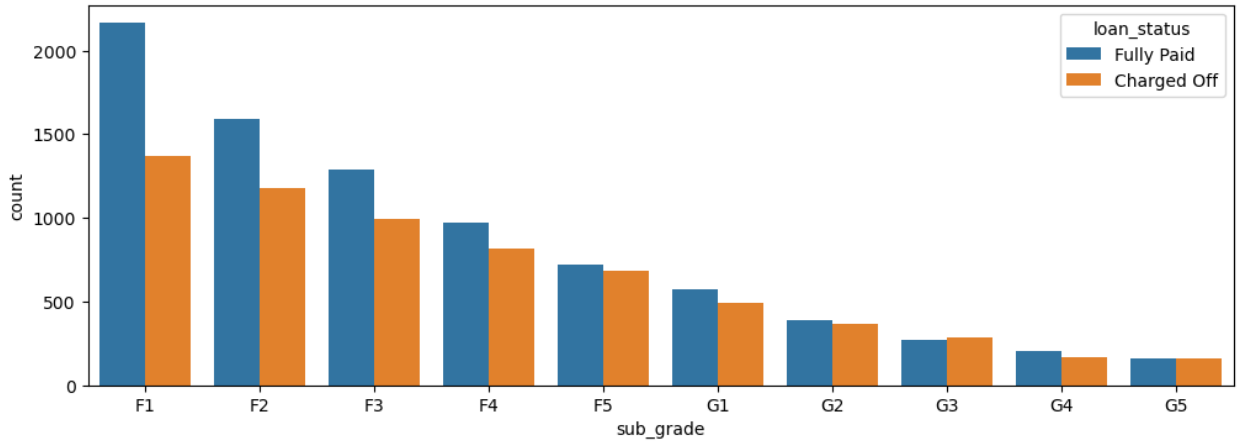



```
plt.figure(figsize=(12,6))
sns.countplot(x='sub_grade',data=df,order=sorted(df['sub_grade'].uniqu
e()),palette='coolwarm',hue='loan_status')
plt.show()
```




```
f_and_g = df[(df['grade']=='G') | (df['grade']=='F')]

plt.figure(figsize=(12,4))
subgrade_order = sorted(f_and_g['sub_grade'].unique())
sns.countplot(x='sub_grade',data=f_and_g,order =
```

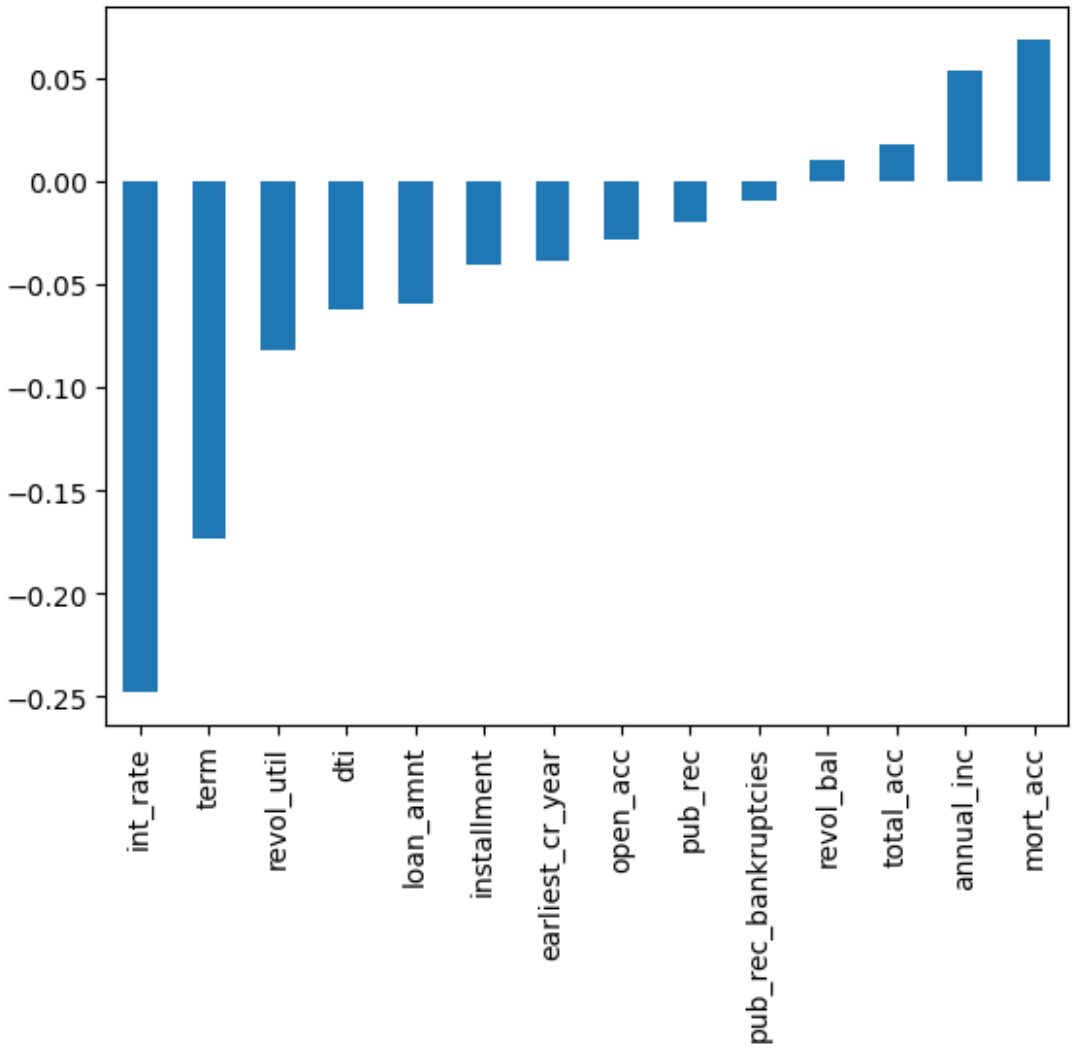
```
subgrade_order,hue='loan_status')
plt.show()
```




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

df[['loan_repaid','loan_status']]
```

```
        loan_repaid  loan_status
0                 1   Fully Paid
1                 1   Fully Paid
2                 1   Fully Paid
3                 1   Fully Paid
4                 0  Charged Off
...             ...          ...
396025            1   Fully Paid
396026            1   Fully Paid
396027            1   Fully Paid
396028            1   Fully Paid
396029            1   Fully Paid

[395219 rows x 2 columns]
```

```
numeric_df = df.select_dtypes(include=['float64', 'int64'])
numeric_df.corr()['loan_repaid'][:-1].sort_values().plot(kind='bar')
plt.show()
```




```
df.head()
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	\
0	10000.0	36	11.44	329.48	B	B4	
1	8000.0	36	11.99	265.68	B	B5	
2	15600.0	36	10.49	506.97	B	B3	
3	7200.0	36	6.49	220.65	A	A2	
4	24375.0	60	17.27	609.33	C	C5	

	emp_title	emp_length	home_ownership	annual_inc	...
pub_rec	\				
0	Marketing	10+ years	RENT	117000.0	...
0.0					
1	Credit analyst	4 years	MORTGAGE	65000.0	...
0.0					
2	Statistician	< 1 year	RENT	43057.0	...
0.0					
3	Client Advocate	6 years	RENT	54000.0	...

```
0.0
4  Destiny Management Inc.    9 years       MORTGAGE     55000.0  ...
0.0

  revol_bal revol_util total_acc initial_list_status  application_type
\
0   36369.0       41.8      25.0                   w        INDIVIDUAL

1   20131.0       53.3      27.0                   f        INDIVIDUAL

2   11987.0       92.2      26.0                   f        INDIVIDUAL

3    5472.0       21.5      13.0                   f        INDIVIDUAL

4   24584.0       69.8      43.0                   f        INDIVIDUAL


  mort_acc  pub_rec_bankruptcies  \
0      0.0                   0.0
1      3.0                   0.0
2      0.0                   0.0
3      0.0                   0.0
4      1.0                   0.0

                                             address  loan_repaid
0     0174 Michelle Gateway\nMendozaberg, OK 22690            1
1  1076 Carney Fort Apt. 347\nLoganmouth, SD 05113            1
2  87025 Mark Dale Apt. 269\nNew Sabrina, WV 05113            1
3            823 Reid Ford\nDelacruzside, MA 00813            1
4             679 Luna Roads\nGreggshire, VA 11650            0

[5 rows x 28 columns]
```

```
len(df)
```

```
396030
```

```
df.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
```

```
loan_status                 0
purpose                     0
title                    1756
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                37795
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

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

```
loan_amnt               0.000000
term                    0.000000
int_rate                0.000000
installment             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
```

```
df['emp_title'].nunique()
```

```
173105
```

```
df['emp_title'].value_counts()
```

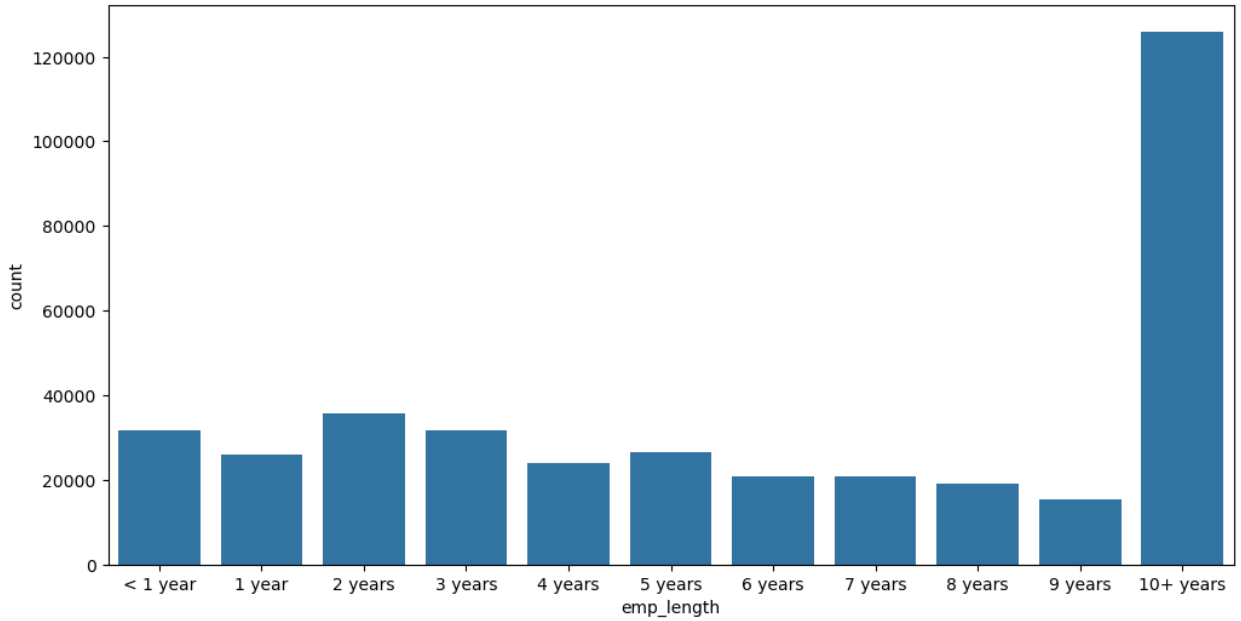
```
emp_title
Teacher                     4389
Manager                     4250
Registered Nurse            1856
RN                          1846
Supervisor                  1830
                            ...
Postman                        1
McCarthy & Holthus, LLC        1
jp flooring                    1
Histology Technologist         1
Gracon Services, Inc           1
Name: count, Length: 173105, dtype: int64
```

```
df= df.drop('emp_title',axis=1)
```

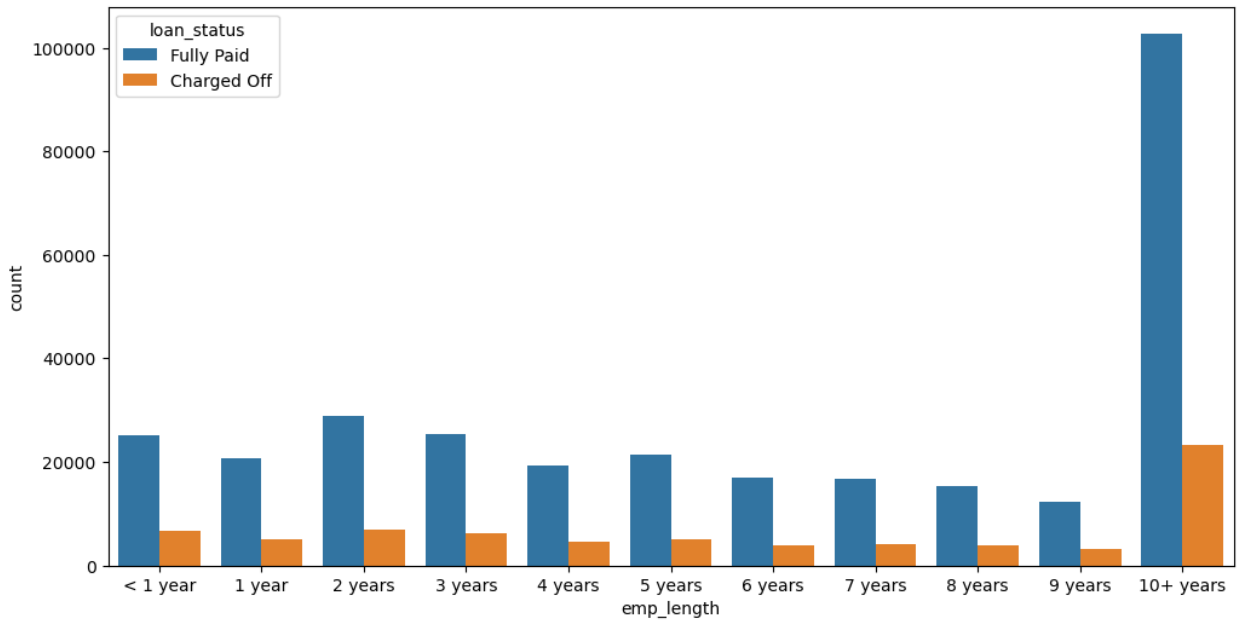
```
sorted(df['emp_length'].dropna().unique())
```

```
['1 year',
 '10+ years',
 '2 years',
 '3 years',
 '4 years',
 '5 years',
 '6 years',
 '7 years',
 '8 years',
 '9 years',
 '< 1 year']
```

```
emp_length_order = [ '< 1 year',
                      '1 year',
                     '2 years',
                     '3 years',
                     '4 years',
                     '5 years',
                     '6 years',
                     '7 years',
                     '8 years',
                     '9 years',
                     '10+ years']

plt.figure(figsize=(12,6))
sns.countplot(x='emp_length',data=df,order=emp_length_order)
plt.show()
```

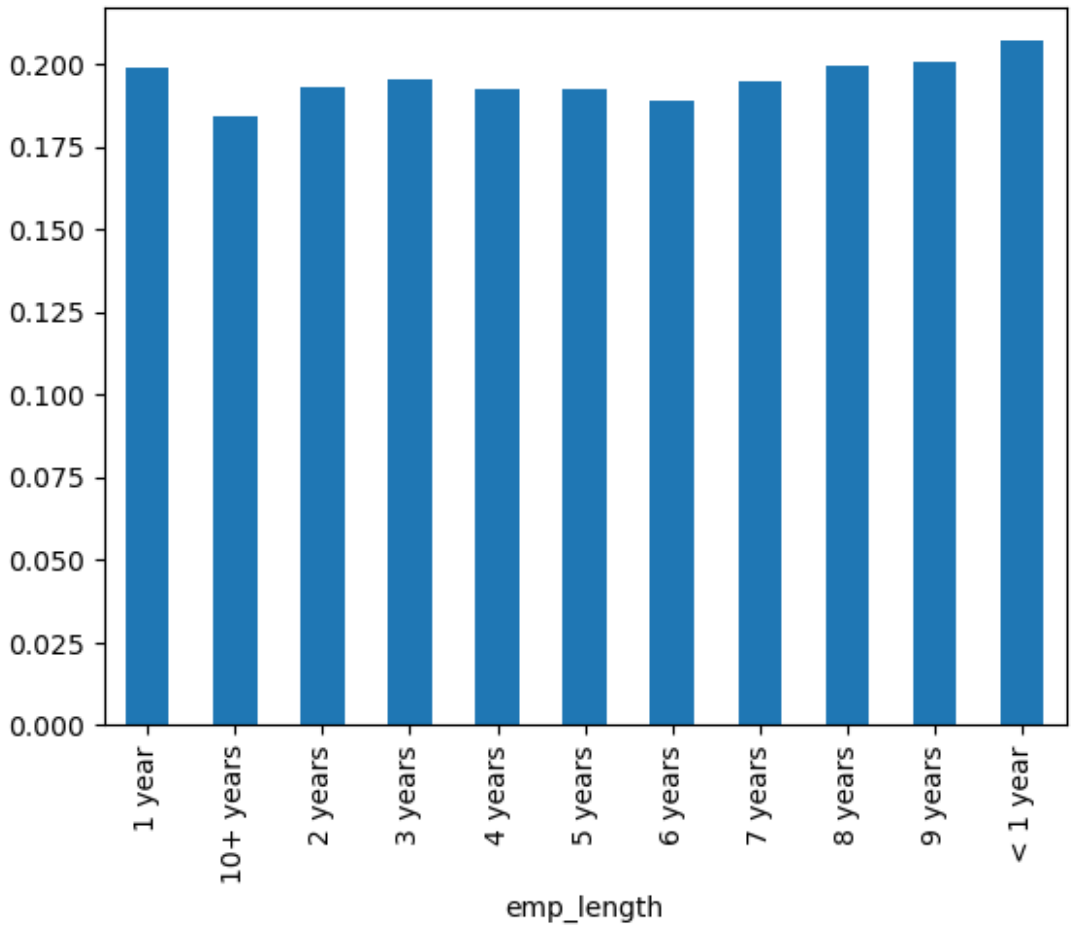



```
plt.figure(figsize=(12,6))
sns.countplot(x='emp_length',data=df,order=emp_length_order,hue='loan_
status')
plt.show()
```




```
emp_co = df[df['loan_status'] == 'Charged
Off'].groupby('emp_length').size()

emp_fp = df[df['loan_status'] == 'Fully
Paid'].groupby('emp_length').size()
```

```
emp_co/emp_fp

emp_length
1 year       0.248649
10+ years    0.225770
2 years      0.239560
3 years      0.242593
4 years      0.238213
5 years      0.237911
6 years      0.233341
7 years      0.241887
8 years      0.249625
9 years      0.250735
< 1 year     0.260830
dtype: float64

emp_len = emp_co/(emp_co + emp_fp)

emp_len.plot(kind='bar')
plt.show()
```



```
df= df.drop('emp_length',axis=1)
```

```
df.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1756
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                37795
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

```
df['title'].nunique()
```

```
48816
```

```
df = df.drop('title',axis=1)
```

```
df['mort_acc'].unique()
```

```
array([ 0.,  3.,  1.,  4.,  2.,  6.,  5., nan, 10.,  7., 12., 11.,
8.,
        9., 13., 14., 22., 34., 15., 25., 19., 16., 17., 32., 18.,
24.,
       21., 20., 31., 28., 30., 23., 26., 27.])
```

```
df['mort_acc'].value_counts()
```

```
mort_acc
0.0     139777
1.0      60416
2.0      49948
```

```
3.0      38049
4.0      27887
5.0      18194
6.0      11069
7.0       6052
8.0       3121
9.0       1656
10.0       865
11.0       479
12.0       264
13.0       146
14.0       107
15.0        61
16.0        37
17.0        22
18.0        18
19.0        15
20.0        13
24.0        10
22.0         7
21.0         4
25.0         4
27.0         3
32.0         2
31.0         2
23.0         2
26.0         2
28.0         1
30.0         1
34.0         1
Name: count, dtype: int64
```

```
numeric_df = df.select_dtypes(include=['float64', 'int64'])
numeric_df .corr()['mort_acc'].sort_values()
```

```
int_rate               -0.082583
dti                    -0.025439
revol_util              0.007514
pub_rec                 0.011552
pub_rec_bankruptcies    0.027239
open_acc                0.109205
installment             0.193694
revol_bal               0.194925
loan_amnt               0.222315
annual_inc              0.236320
total_acc               0.381072
mort_acc                1.000000
Name: mort_acc, dtype: float64
```

```
numeric_df = df.select_dtypes(include=['float64', 'int64'])
total_acc_avg = numeric_df.groupby('total_acc').mean()['mort_acc']
```

```
numeric_df.groupby('total_acc').mean()['mort_acc']
```

```
total_acc
2.0      0.000000
3.0      0.052023
4.0      0.066743
5.0      0.103289
6.0      0.151293
           ...
124.0    1.000000
129.0    1.000000
135.0    3.000000
150.0    2.000000
151.0    0.000000
Name: mort_acc, Length: 118, dtype: float64
```

```
def fill_mort_acc(total_acc,mort_acc):

    if np.isnan(mort_acc):
        return total_acc_avg[total_acc]
    else:
        return mort_acc

df['mort_acc'] = df.apply(lambda x:
fill_mort_acc(x['total_acc'],x['mort_acc']),axis=1)
```

```
df.isnull().sum()
```

```
loan_amnt               0
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
home_ownership          0
annual_inc              0
verification_status     0
issue_d                 0
loan_status             0
purpose                 0
dti                     0
earliest_cr_line        0
open_acc                0
pub_rec                 0
revol_bal               0
revol_util              0
total_acc               0
initial_list_status     0
```

```
application_type        0
mort_acc                0
pub_rec_bankruptcies    0
address                 0
dtype: int64
```

```
df = df.dropna()
```

```
df.select_dtypes(['object']).columns
```

```
Index(['loan_status', 'address', 'zip_code'], dtype='object')
```

```
df['term'] = df['term'].apply(lambda term: int(term[:3]))
```

```
df['term'].value_counts()
```

```
term
36    301247
60     93972
Name: count, dtype: int64
```

```
df = df.drop('grade',axis=1)
```

```
dummies = pd.get_dummies(df['sub_grade'],drop_first=True)
```

```
df = pd.concat([df.drop('sub_grade',axis=1),dummies],axis=1)
```

```
df.columns
```

```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'annual_inc',
       'loan_status', 'dti', 'open_acc', 'pub_rec', 'revol_bal',
'revol_util',
       'total_acc', 'mort_acc', 'pub_rec_bankruptcies', 'address',
'A2', 'A3',
       'A4', 'A5', 'B1', 'B2', 'B3', 'B4', 'B5', 'C1', 'C2', 'C3',
'C4', 'C5',
       'D1', 'D2', 'D3', 'D4', 'D5', 'E1', 'E2', 'E3', 'E4', 'E5',
'F1', 'F2',
       'F3', 'F4', 'F5', 'G1', 'G2', 'G3', 'G4', 'G5',
       'verification_status_Source Verified',
'verification_status_Verified',
       'application_type_INDIVIDUAL', 'application_type_JOINT',
       'initial_list_status_w', 'purpose_credit_card',
       'purpose_debt_consolidation', 'purpose_educational',
       'purpose_home_improvement', 'purpose_house',
'purpose_major_purchase',
       'purpose_medical', 'purpose_moving', 'purpose_other',
       'purpose_renewable_energy', 'purpose_small_business',
       'purpose_vacation', 'purpose_wedding', 'OTHER', 'OWN', 'RENT',
       'zip_code', '05113', '11650', '22690', '29597', '30723',
'48052',
       '70466', '86630', '93700', '05113', '11650', '22690', '29597',
```

```
'30723',
       '48052', '70466', '86630', '93700', 'earliest_cr_year',
'OTHER', 'OWN',
       'RENT', 'OTHER', 'OWN', 'RENT'],
      dtype='object')
```

```
dummies = pd.get_dummies(df[['verification_status',
'application_type','initial_list_status','purpose' ]],drop_first=True)
df = df.drop(['verification_status',
'application_type','initial_list_status','purpose'],axis=1)
df = pd.concat([df,dummies],axis=1)
```

```
df['home_ownership'].value_counts()
```

```
home_ownership
MORTGAGE    198022
RENT        159395
OWN          37660
OTHER          142
Name: count, dtype: int64
```

```
df['home_ownership'] = df['home_ownership'].replace(['NONE','ANY'],
'OTHER')
```

```
dummies = pd.get_dummies(df['home_ownership'],drop_first = True)
df = pd.concat([df,dummies],axis=1)
```

```
df = df.drop('home_ownership',axis=1)
```

```
df['zip_code'] = df['address'].apply(lambda address:address[-5:])
```

```
df['zip_code'].value_counts()
```

```
zip_code
70466    56880
22690    56413
30723    56402
48052    55811
00813    45725
29597    45393
05113    45300
11650    11210
93700    11126
86630    10959
Name: count, dtype: int64
```

```
dummies = pd.get_dummies(df['zip_code'],drop_first = True)
df = df.drop('zip_code',axis=1)
df = df.drop('address',axis=1)
df = pd.concat([df,dummies],axis=1)
```

```
df.head(2)
```

```
   loan_amnt  term  int_rate  installment  annual_inc loan_status
dti  \
0    10000.0    36     11.44       329.48    117000.0  Fully Paid
26.24
1     8000.0    36     11.99       265.68     65000.0  Fully Paid
22.05

   open_acc  pub_rec  revol_bal  ...   RENT  05113  11650  22690
29597  \
0      16.0      0.0    36369.0  ...   True  False  False   True
False
1      17.0      0.0    20131.0  ...  False   True  False  False
False

   30723  48052  70466  86630  93700
0  False  False  False  False  False
1  False  False  False  False  False

[2 rows x 103 columns]
```

```
df = df.drop('issue_d',axis=1)
```

```
df['earliest_cr_year'] = df['earliest_cr_line'].apply(lambda
date:int(date[-4:]))
df = df.drop('earliest_cr_line',axis=1)
```

```
df.select_dtypes(['object']).columns
```

```
Index(['loan_status'], dtype='object')
```

```
df[['loan_repaid','loan_status']]
```

```
        loan_repaid  loan_status
0                 1   Fully Paid
1                 1   Fully Paid
2                 1   Fully Paid
3                 1   Fully Paid
4                 0  Charged Off
...             ...          ...
396025            1   Fully Paid
396026            1   Fully Paid
396027            1   Fully Paid
396028            1   Fully Paid
396029            1   Fully Paid

[395219 rows x 2 columns]
```

```
from sklearn.model_selection import train_test_split
```

```
df = df.drop('loan_status',axis=1)
```

```
X =df.drop('loan_repaid',axis=1).values
y=df['loan_repaid'].values

#X
#y
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

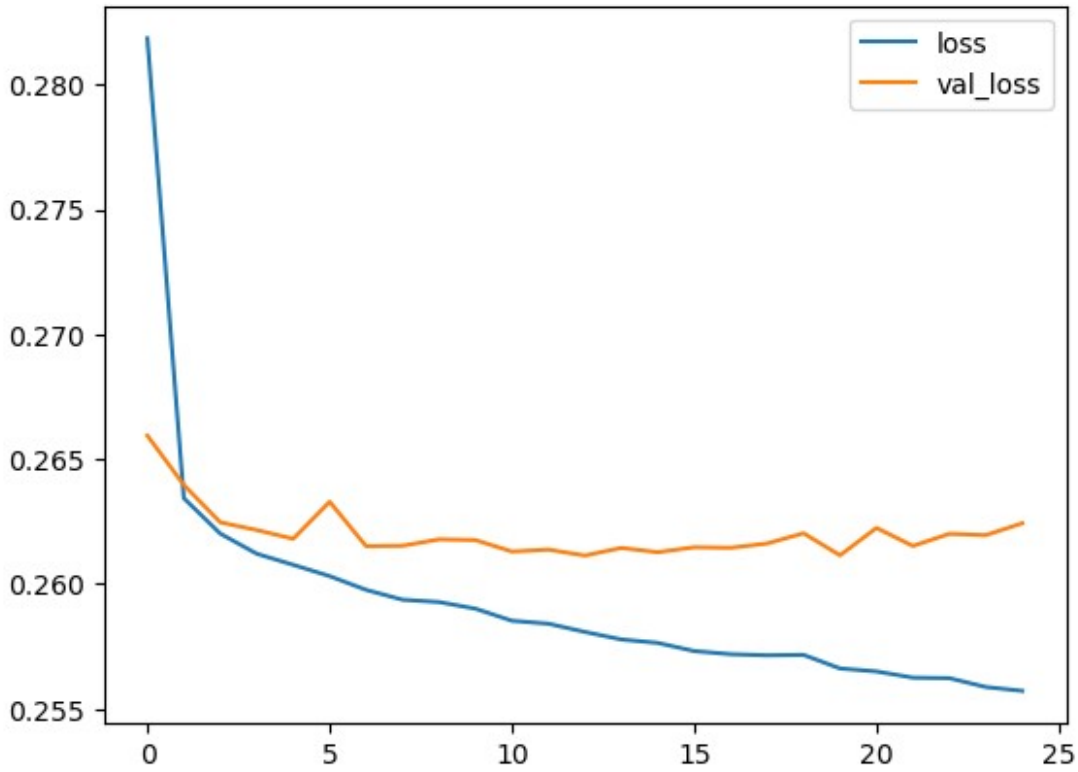
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=101)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_train =scaler.fit_transform(X_train)

X_test =scaler.transform(X_test)

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation,Dropout
from tensorflow.keras.constraints import max_norm

X_train.shape
```

```
(316175, 102)
```

```
model = Sequential()

model.add(Dense(102,activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(51,activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(26,activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(units=1,activation = 'sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics
=['accuracy'])

model.fit(x=X_train,
          y=y_train,
          epochs =25,
          batch_size =256,
          validation_data =(X_test,y_test),
          verbose =1,
         )
```

```
Epoch 1/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8683 - loss:
0.3209 - val_accuracy: 0.8869 - val_loss: 0.2659
Epoch 2/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8881 - loss:
0.2639 - val_accuracy: 0.8869 - val_loss: 0.2640
Epoch 3/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8887 - loss:
0.2618 - val_accuracy: 0.8872 - val_loss: 0.2625
Epoch 4/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8879 - loss:
0.2621 - val_accuracy: 0.8871 - val_loss: 0.2622
Epoch 5/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8873 - loss:
0.2627 - val_accuracy: 0.8871 - val_loss: 0.2618
Epoch 6/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8891 - loss:
0.2587 - val_accuracy: 0.8871 - val_loss: 0.2633
Epoch 7/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8888 - loss:
0.2588 - val_accuracy: 0.8871 - val_loss: 0.2615
Epoch 8/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8887 - loss:
0.2587 - val_accuracy: 0.8872 - val_loss: 0.2615
Epoch 9/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8885 - loss:
0.2592 - val_accuracy: 0.8871 - val_loss: 0.2618
Epoch 10/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8896 - loss:
0.2577 - val_accuracy: 0.8871 - val_loss: 0.2618
Epoch 11/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8893 - loss:
0.2575 - val_accuracy: 0.8873 - val_loss: 0.2613
Epoch 12/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8892 - loss:
0.2579 - val_accuracy: 0.8871 - val_loss: 0.2614
Epoch 13/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8889 - loss:
0.2579 - val_accuracy: 0.8872 - val_loss: 0.2611
Epoch 14/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8886 - loss:
0.2585 - val_accuracy: 0.8872 - val_loss: 0.2614
Epoch 15/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8891 - loss:
0.2563 - val_accuracy: 0.8872 - val_loss: 0.2613
Epoch 16/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8896 - loss:
0.2565 - val_accuracy: 0.8873 - val_loss: 0.2615
Epoch 17/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8895 - loss:
```

```
0.2558 - val_accuracy: 0.8874 - val_loss: 0.2615
Epoch 18/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8888 - loss:
0.2572 - val_accuracy: 0.8871 - val_loss: 0.2616
Epoch 19/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8887 - loss:
0.2577 - val_accuracy: 0.8872 - val_loss: 0.2620
Epoch 20/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8892 - loss:
0.2571 - val_accuracy: 0.8875 - val_loss: 0.2612
Epoch 21/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8882 - loss:
0.2578 - val_accuracy: 0.8874 - val_loss: 0.2623
Epoch 22/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8897 - loss:
0.2555 - val_accuracy: 0.8879 - val_loss: 0.2615
Epoch 23/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8898 - loss:
0.2551 - val_accuracy: 0.8871 - val_loss: 0.2620
Epoch 24/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8896 - loss:
0.2553 - val_accuracy: 0.8878 - val_loss: 0.2620
Epoch 25/25
1236/1236 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8889 - loss:
0.2559 - val_accuracy: 0.8876 - val_loss: 0.2624

<keras.src.callbacks.history.History at 0x22a2c8f2c60>

from tensorflow.keras.models import load_model

model.save('lendingmodel.h5')

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

table = pd.DataFrame(model.history.history)

losses = pd.DataFrame(model.history.history)
losses =losses.drop('accuracy',axis=1)
losses =losses.drop('val_accuracy',axis=1)

losses.plot()
plt.show()
```




```
from  sklearn.metrics import classification_report,confusion_matrix

predictions = model.predict(X_test)
predictions = (predictions > 0.5).astype(int)
predictions= predictions.flatten()
```

```
2471/2471 ━━━━━━━━━━━━━━━━━━━━ 2s 614us/step
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.97      0.45      0.61     15658
           1       0.88      1.00      0.93     63386

    accuracy                           0.89     79044
   macro avg       0.93      0.72      0.77     79044
weighted avg       0.90      0.89      0.87     79044
```

```
confusion_matrix(y_test,predictions)
```

```
array([[ 6977,  8681],
       [  205, 63181]], dtype=int64)
```

```
# Check for new Customer
```

```
import random
random.seed(101)
random_ind = random.randint(0,len(df))

new_customer = df.drop('loan_repaid',axis=1).iloc[random_ind]
new_customer
```

```
loan_amnt      25000.0
term              60
int_rate         18.24
installment     638.11
annual_inc     61665.0
                ...
30723             True
48052            False
70466            False
86630            False
93700            False
Name: 305323, Length: 102, dtype: object
```

```
new_customer = scaler.transform(new_customer.values.reshape(1,102))

model.predict(new_customer)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step

array([[0.57112336]], dtype=float32)

df.iloc[random_ind]['loan_repaid']

1

$ export PATH=/Library/TeX/texbin:$PATH

  Cell In[4], line 1
    $ export PATH=/Library/TeX/texbin:$PATH
    ^
SyntaxError: invalid syntax
```