In [230…
```
import pandas as pd
```

In [231…
```
data_info = pd.read_csv('../data/DATA/lending_club_info.csv',index_col='LoanStatNew
```

In [232…
```
print(data_info.loc['revol_util']['Description'])
```

```
Revolving line utilization rate, or the amount of credit the borrower is using relat
ive to all available revolving credit.
```

In [233…
```
def feat_info(col_name):
    print(data_info.loc[col_name]['Description'])
```

In [234…
```
feat_info('mort_acc')
```

```
Number of mortgage accounts.
```

Exploratory Data Analysis

In [235…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [236…
```
df = pd.read_csv('../data/DATA/lending_club_loan_two.csv')
```

In [237…
```
df.info()
```

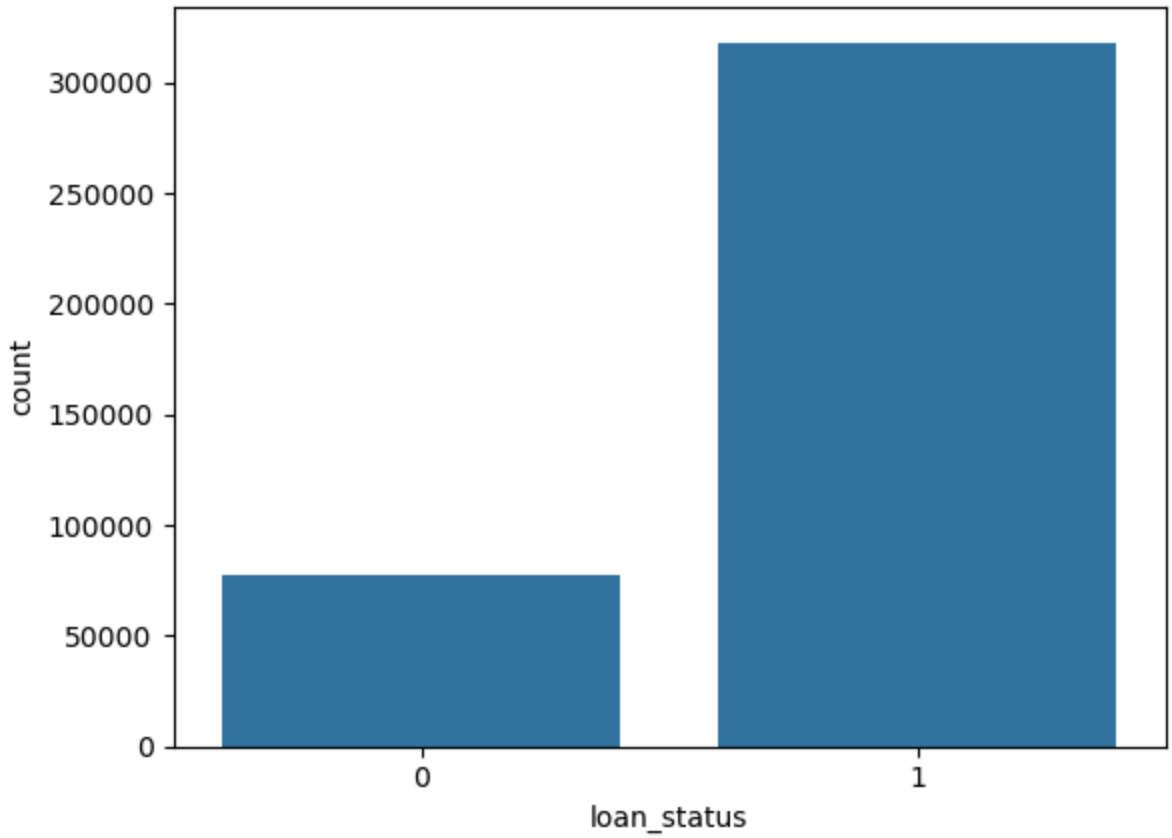
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             396030 non-null  int64
 1   term                  396030 non-null  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  int64
 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  int64
 18  pub_rec               396030 non-null  int64
 19  revol_bal             396030 non-null  int64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  int64
 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(7), int64(6), object(14)
memory usage: 81.6+ MB
```

In [238…

```
sns.countplot(x='loan_status',data=df)
```

Out[238…

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



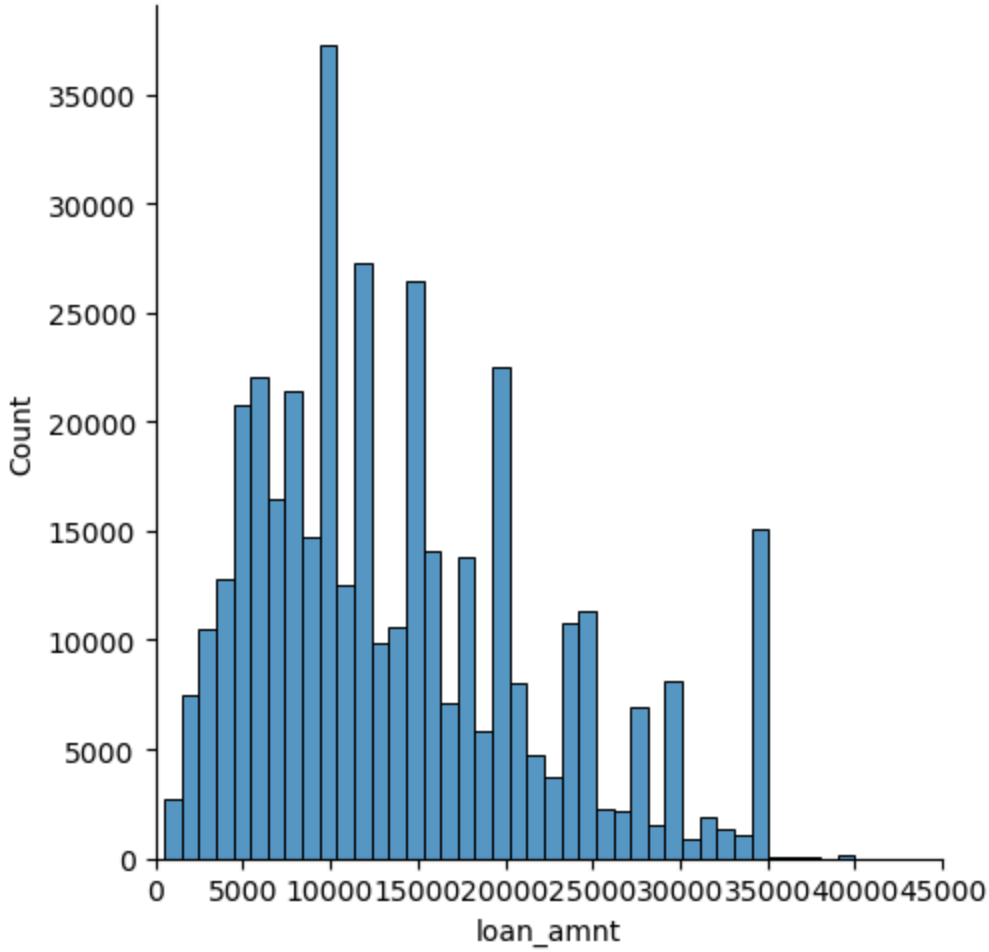
In [239…

```
plt.figure(figsize=(12,4))
sns.displot(df['loan_amnt'],kde=False,bins=40)
plt.xlim(0,45000)
```

```
C:\learnings\envs\deeplearning\lib\site-packages\seaborn\axisgrid.py:123: UserWarnin
g: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[239…

```
(0.0, 45000.0)
<Figure size 1200x400 with 0 Axes>
```



In [240…

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

Out[240…

	loan_amnt	int_rate	installment	annual_inc	loan_status	dti
loan_amnt	1.000000	0.168921	0.953929	0.336887	-0.059836	0.016636
int_rate	0.168921	1.000000	0.162758	-0.056771	-0.247758	0.079038
installment	0.953929	0.162758	1.000000	0.330381	-0.041082	0.015786
annual_inc	0.336887	-0.056771	0.330381	1.000000	0.053432	-0.081685
loan_status	-0.059836	-0.247758	-0.041082	0.053432	1.000000	-0.062413
dti	0.016636	0.079038	0.015786	-0.081685	-0.062413	1.000000
open_acc	0.198556	0.011649	0.188973	0.136150	-0.028012	0.136181
pub_rec	-0.077779	0.060986	-0.067892	-0.013720	-0.019933	-0.017639
revol_bal	0.328320	-0.011280	0.316455	0.299773	0.010892	0.063571
revol_util	0.099911	0.293659	0.123915	0.027871	-0.082373	0.088375
total_acc	0.223886	-0.036404	0.202430	0.193023	0.017893	0.102128
mort_acc	0.222315	-0.082583	0.193694	0.236320	0.073111	-0.025439
pub_rec_bankruptcies	-0.106539	0.057450	-0.098628	-0.050162	-0.009383	-0.014558

In [241…

```
plt.figure(figsize=(12,7))
sns.heatmap(df_numeric.corr(), annot=True, cmap='viridis')
plt.ylim(10,0)
```

Out[241…

```
(10.0, 0.0)
```

In [242…

```
feat_info('installment')
```

```
The monthly payment owed by the borrower if the loan originates.
```

In [243…

```
feat_info('loan_amnt')
```

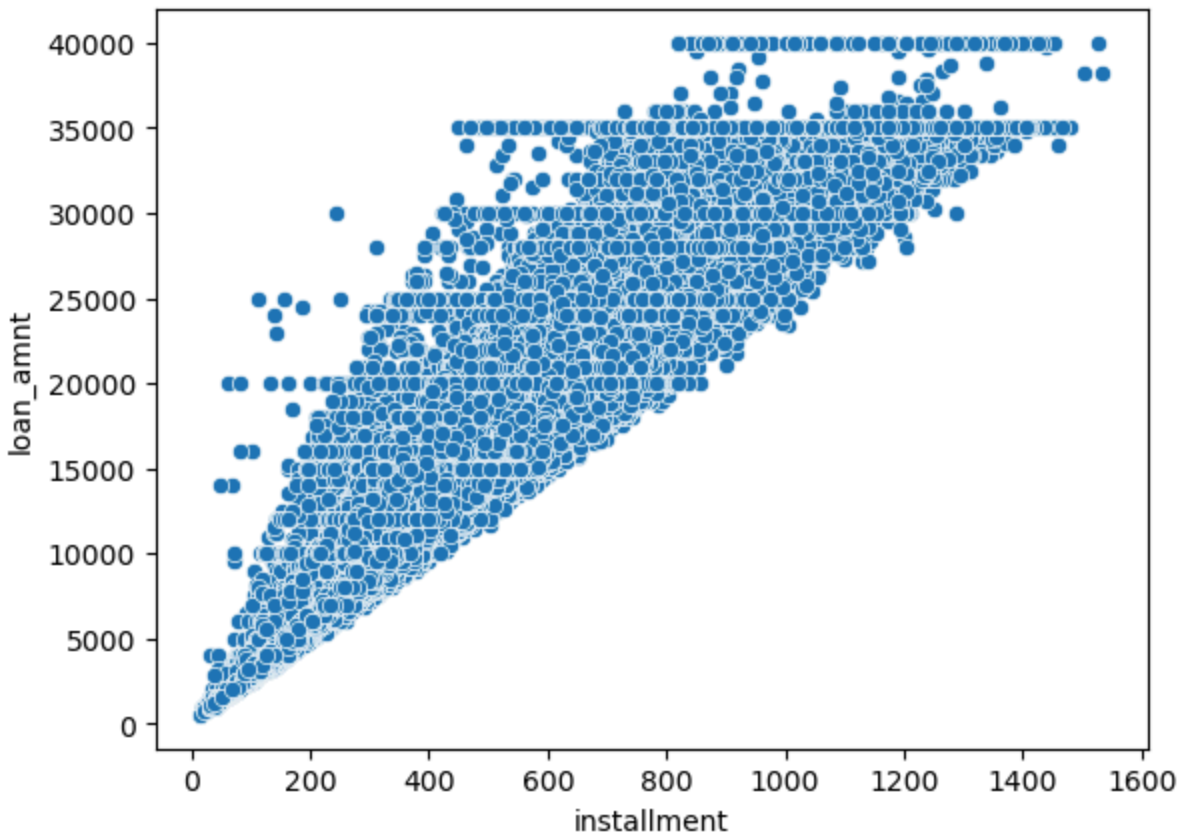
```
The listed amount of the loan applied for by the borrower. If at some point in time,
the credit department reduces the loan amount, then it will be reflected in this val
ue.
```

In [244…

```
sns.scatterplot(x='installment', y='loan_amnt', data=df)
```

Out[244…

```
<Axes: xlabel='installment', ylabel='loan_amnt'>
```

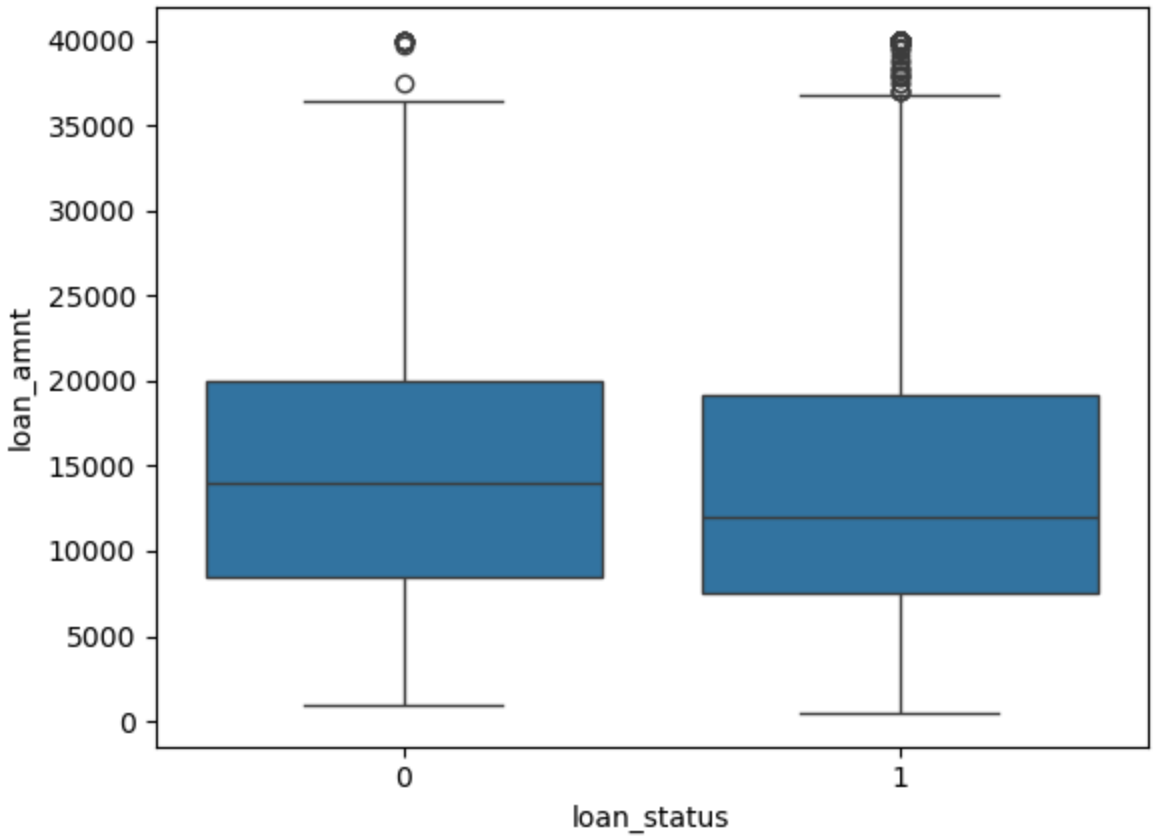


In [245…

```
sns.boxplot(x='loan_status', y='loan_amnt', data=df_numeric)
# Where is 1 is fully paid and 0 is charged off
```

Out[245…

```
<Axes: xlabel='loan_status', ylabel='loan_amnt'>
```



In [246…
```
df_numeric.groupby('loan_status')['loan_amnt'].describe()
```

Out[246…

	count	mean	std	min	25%	50%	75%	max
loan_status								
0	77673.0	15126.300967	8505.090557	1000.0	8525.0	14000.0	20000.0	40000.0
1	318357.0	13866.878771	8302.319699	500.0	7500.0	12000.0	19225.0	40000.0

In [247…
```
df['grade'].unique()
```

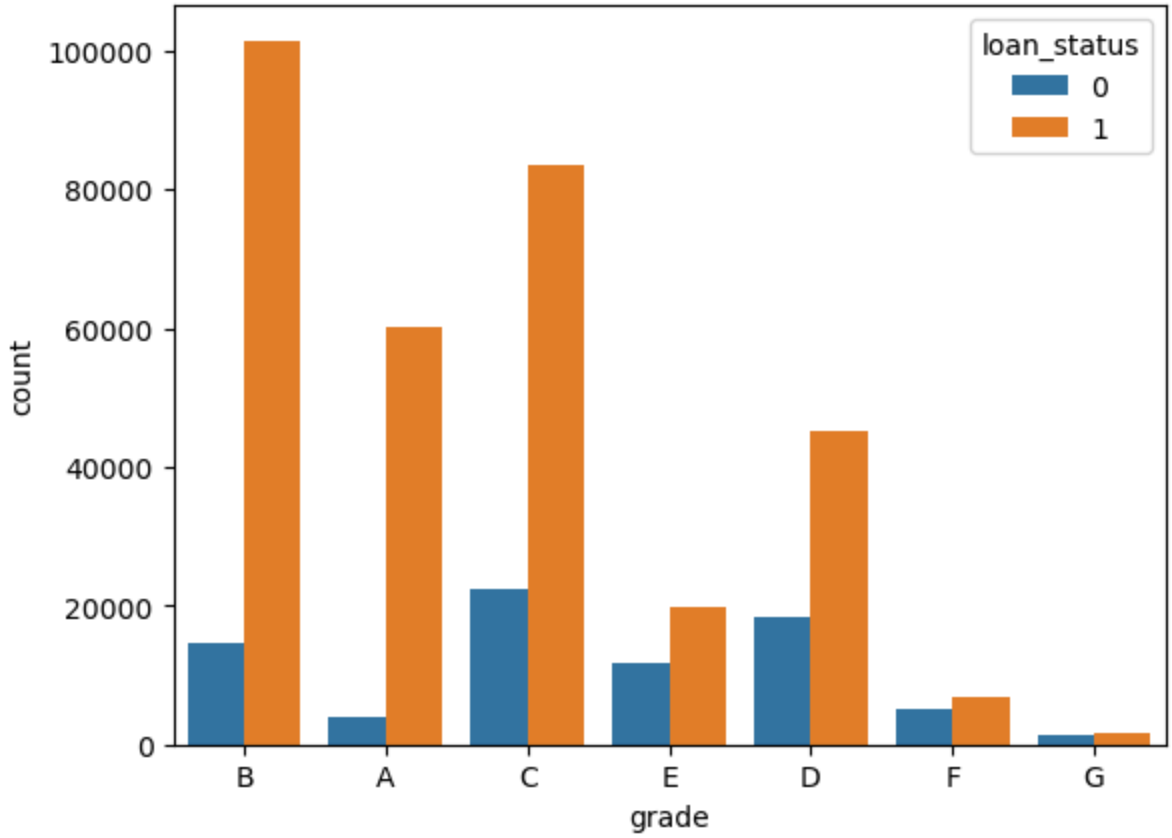
Out[247…
```
array(['B', 'A', 'C', 'E', 'D', 'F', 'G'], dtype=object)
```

In [248…
```
df['sub_grade'].unique()
```

Out[248…
```
array(['B4', 'B5', 'B3', 'A2', 'C5', 'C3', 'A1', 'B2', 'C1', 'A5', 'E4',
       'A4', 'A3', 'D1', 'C2', 'B1', 'D3', 'D5', 'D2', 'E1', 'E2', 'E5',
       'F4', 'E3', 'D4', 'G1', 'F5', 'G2', 'C4', 'F1', 'F3', 'G5', 'G4',
       'F2', 'G3'], dtype=object)
```

In [249…
```
sns.countplot(x='grade', hue='loan_status', data=df)
```

Out[249…
```
<Axes: xlabel='grade', ylabel='count'>
```

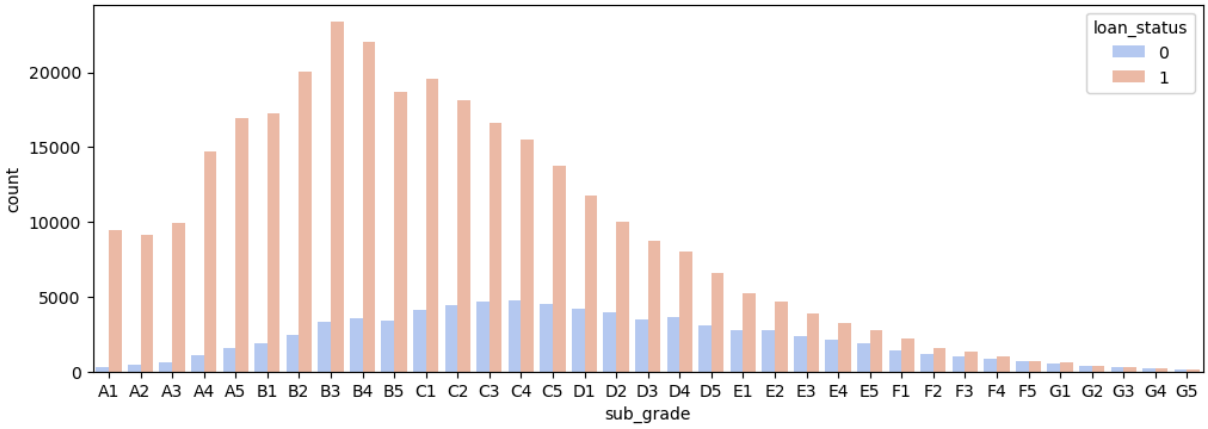


In [250…

```
plt.figure(figsize=(12,4))
subgrade_order = sorted(df['sub_grade'].unique())
sns.countplot(order=subgrade_order, palette='coolwarm', x='sub_grade', hue='loan_st
```

Out[250…

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



In [251…

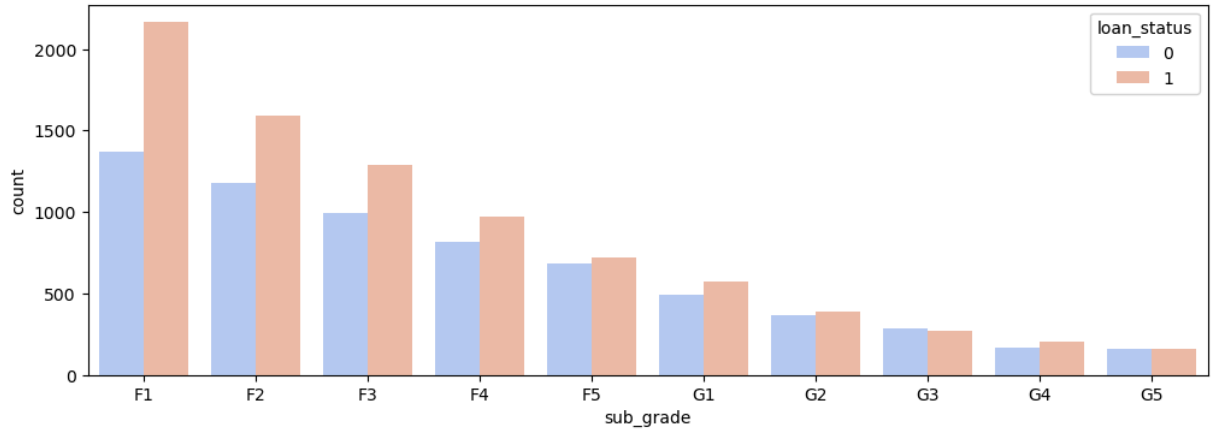
```
#Let take look at F and G grades as their is equal rate of distribution for both Fu
```

In [252…

```
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(order=subgrade_order, palette='coolwarm', x='sub_grade', hue='loan_st
```

Out[252…

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



In [253…

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

In [254…

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

Out[254…

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

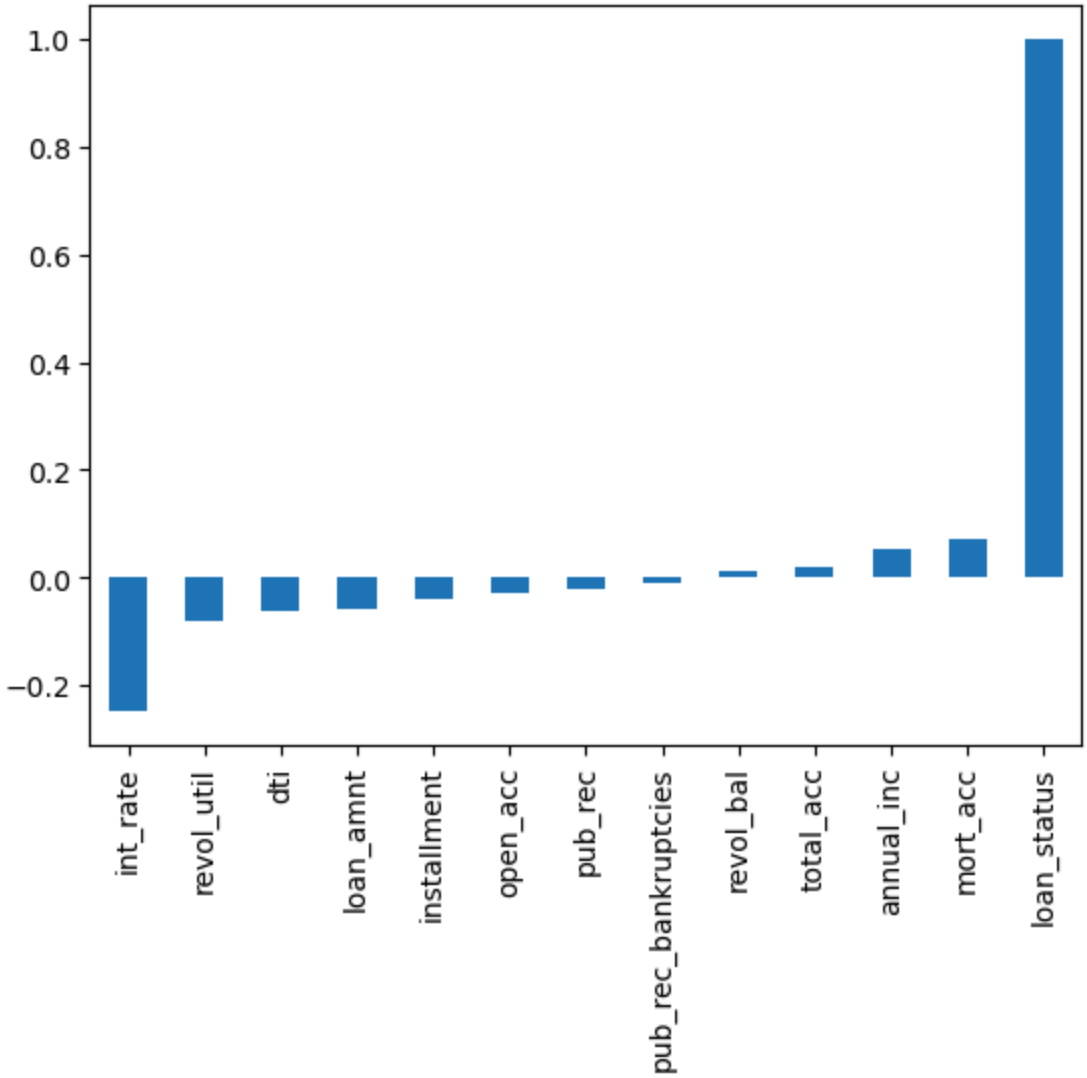
396030 rows × 2 columns

In [255…

```
df_numeric.corr()['loan_status'].sort_values().plot(kind='bar')
```

Out[255…

```
<Axes: >
```



Data PreProcessing

In [256…

```
# we will be removing or filling in an misisng data
```

In [257…

```
df.head()
```

Out[257…

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length
0	10000	36 months	11.44	329.48	B	B4	Marketing	10+ years
1	8000	36 months	11.99	265.68	B	B5	Credit analyst	4 years
2	15600	36 months	10.49	506.97	B	B3	Statistician	< 1 year
3	7200	36 months	6.49	220.65	A	A2	Client Advocate	6 years
4	24375	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years

5 rows × 28 columns

In [258…

```
len(df)
```

Out[258…

```
396030
```

In [259…

```
# Total count of missing values
```

In [260…

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

Out[260…

```
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
loan_repaid                0
dtype: int64
```

In [261…

```
#Check what is the percentage of missing data
100 * df.isnull().sum()/ len(df)
```

Out[261…

```
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
loan_repaid             0.000000
dtype: float64
```

In [262…

```
feat_info('emp_title')
```

```
The job title supplied by the Borrower when applying for the loan.*
```

In [263…

```
feat_info('emp_length')
```

```
Employment length in years. Possible values are between 0 and 10 where 0 means less
than one year and 10 means ten or more years.
```

In [264…

```
len(df['emp_title'].unique())
```

Out[264…

```
173104
```

In [265…

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

In [266…

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

Out[266…

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

In [267…

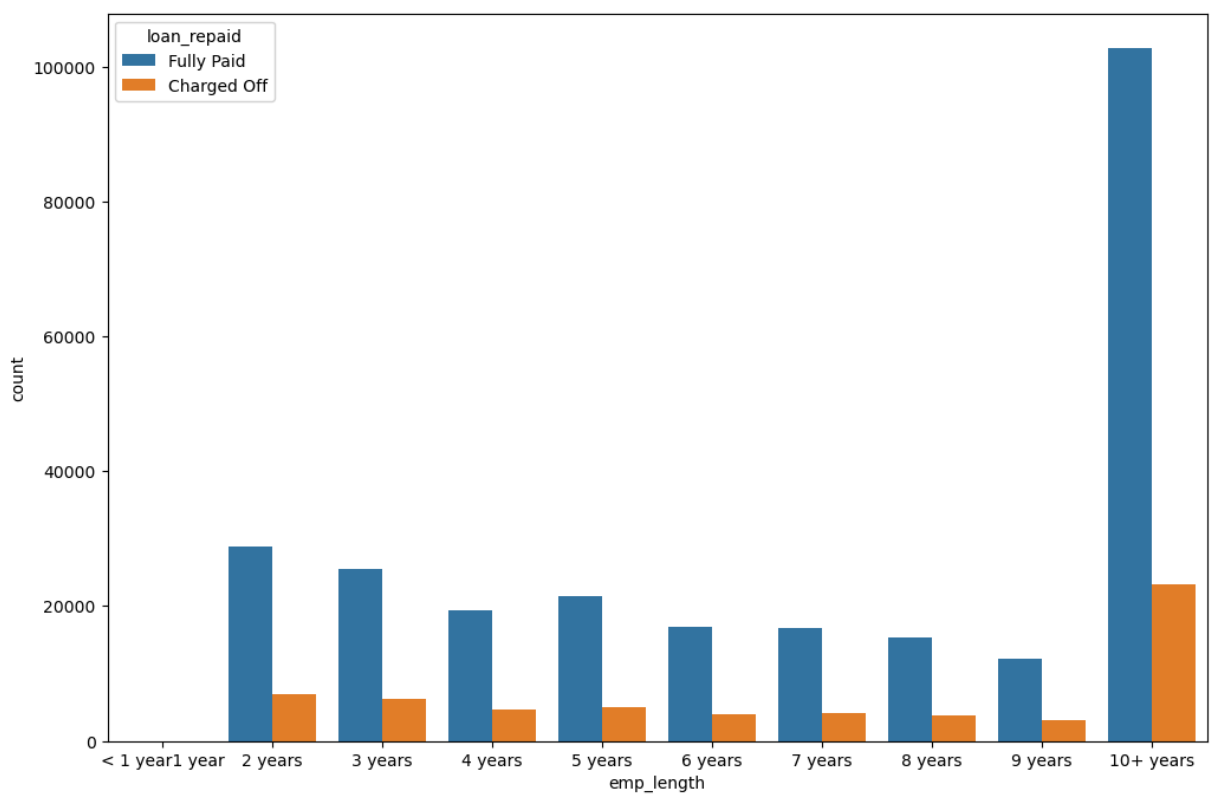
```
emp_length_order = [
 '< 1 year'
 '1 year',
 '2 years',
 '3 years',
 '4 years',
 '5 years',
 '6 years',
 '7 years',
 '8 years',
 '9 years',
 '10+ years'
 ]
```

In [268…

```
plt.figure(figsize=(12,8))
sns.countplot(x='emp_length', data=df, order=emp_length_order, hue='loan_repaid')
```

Out[268…

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



In [269…
```
#To evaluate if emp_length is really a factor for our loan_approval we can check th
```

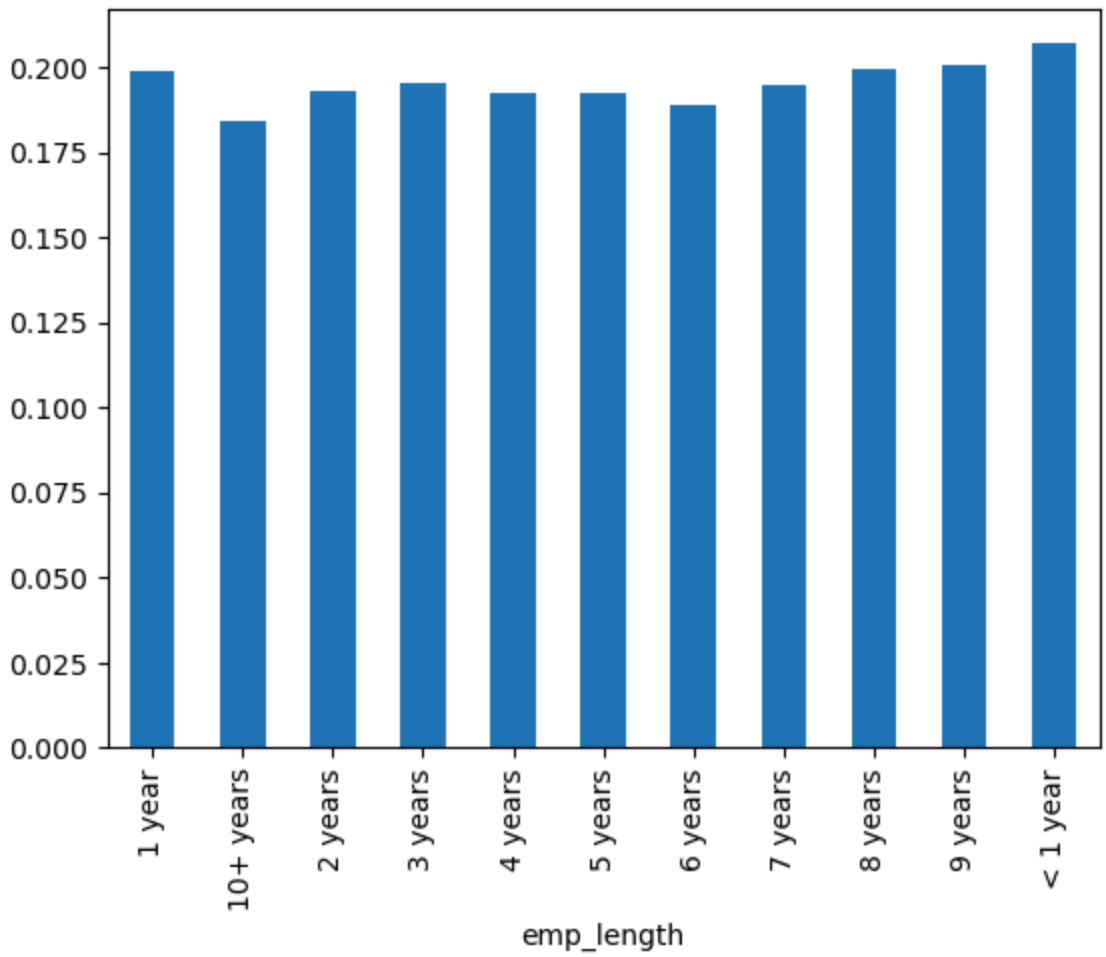
In [270…
```
emp_charged_off = df[df['loan_repaid'] == 'Charged Off'].groupby('emp_length').coun
```

In [271…
```
emp_fully_paid = df[df['loan_repaid'] == 'Fully Paid'].groupby('emp_length').count(
```

In [272…
```
emp_fp = emp_charged_off/(emp_charged_off+ emp_fully_paid)
```

In [273…
```
#Emp length does not have extreme difference to pay off loan; we can exclude this c
emp_fp.plot(kind='bar')
```

Out[273…
```
<Axes: xlabel='emp_length'>
```



In [274…
```
df = df.drop('emp_length', axis=1)
```

In [275…
```
df.isnull().sum()
```

Out[275…

```
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
loan_repaid                 0
dtype: int64
```

In [276…

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

In [277…

```
df.head()
```

Out[277…

	loan_amnt	term	int_rate	installment	grade	sub_grade	home_ownership	annual_in
0	10000	36 months	11.44	329.48	B	B4	RENT	117000.
1	8000	36 months	11.99	265.68	B	B5	MORTGAGE	65000.
2	15600	36 months	10.49	506.97	B	B3	RENT	43057.
3	7200	36 months	6.49	220.65	A	A2	RENT	54000.
4	24375	60 months	17.27	609.33	C	C5	MORTGAGE	55000.

5 rows × 25 columns

In [278…

```
feat_info('mort_acc')
```

```
Number of mortgage accounts.
```

In [279…

```
# LEt's check correlation mort_acc columnn with other column in df
```

In [280…

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

In [281…

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

Out[281…

```
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              276
total_acc                 0
initial_list_status       0
application_type          0
pub_rec_bankruptcies    535
address                   0
loan_repaid               0
dtype: int64
```

In [282…

```
#pub_rec_bankruptcies    revol_util              are very minor percentage of the o
```

In [283…

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

In [284…

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

Out[284…

```
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
pub_rec_bankruptcies    0
address                 0
loan_repaid             0
dtype: int64
```

In [285…

```
#Let's deal with String values
```

In [286…

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

Out[286…

```
Index(['term', 'grade', 'sub_grade', 'home_ownership', 'verification_status',
       'issue_d', 'purpose', 'earliest_cr_line', 'initial_list_status',
       'application_type', 'address', 'loan_repaid'],
      dtype='object')
```

In [287…

```
feat_info('term')
```

```
The number of payments on the loan. Values are in months and can be either 36 or 60.
```

In [288…

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

Out[288…

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

In [289…

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

In [290…

```
df['term']
```

Out[290…
```
0         36
1         36
2         36
3         36
4         60
          ..
396025    60
396026    36
396027    36
396028    60
396029    36
Name: term, Length: 395219, dtype: int64
```

In [291…
```
# Since subgrade also contain grade information; we can drop grade
```

In [292…
```
df = df.drop('grade', axis=1)
```

In [293…
```
dummies = pd.get_dummies(df['sub_grade'], drop_first=True)
df = pd.concat([df.drop('sub_grade', axis=1), dummies], axis=1)
```

In [294…
```
df.columns
```

Out[294…
```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'home_ownership',
       'annual_inc', 'verification_status', 'issue_d', 'loan_status',
       'purpose', 'dti', 'earliest_cr_line', 'open_acc', 'pub_rec',
       'revol_bal', 'revol_util', 'total_acc', 'initial_list_status',
       'application_type', 'pub_rec_bankruptcies', 'address', 'loan_repaid',
       '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'],
      dtype='object')
```

In [295…
```
dummies = pd.get_dummies(df[['verification_status', 'application_type', 'initial_li
df = pd.concat([df.drop(['verification_status', 'application_type', 'initial_list_s
```

In [296…
```
df['home_ownership'].value_counts()
```

Out[296…
```
home_ownership
MORTGAGE    198022
RENT        159395
OWN          37660
OTHER          110
NONE            29
ANY              3
Name: count, dtype: int64
```

In [297…
```
# Most people are MORTAGE, RENT, oR OWN the rest were very miminum compared to the
```

In [298…
```
df['home_ownership'] = df['home_ownership'].replace(['NONE', 'ANY'], 'OTHER')
```

In [299…
```
df['home_ownership'].value_counts()
```

Out[299…
```
home_ownership
MORTGAGE    198022
RENT        159395
OWN          37660
OTHER          142
Name: count, dtype: int64
```

In [300…
```
dummies = pd.get_dummies(df['home_ownership'], drop_first=True)
df = pd.concat([df.drop('home_ownership', axis=1), dummies], axis=1)
```

In [301…
```
df['address']
```

Out[301…
```
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
                                   ...
396025       12951 Williams Crossing\nJohnnyville, DC 30723
396026    0114 Fowler Field Suite 028\nRachelborough, LA...
396027     953 Matthew Points Suite 414\nReedfort, NY 70466
396028    7843 Blake Freeway Apt. 229\nNew Michael, FL 2...
396029          787 Michelle Causeway\nBriannaton, AR 48052
Name: address, Length: 395219, dtype: object
```

In [302…
```
df['zip_code'] = df['address'].apply(lambda address: address[-5:])
```

In [303…
```
df['zip_code'].value_counts()
```

Out[303…
```
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
```

In [304…
```
dummies = pd.get_dummies(df['zip_code'], drop_first=True)
df = pd.concat([df.drop('zip_code', axis=1), dummies], axis=1)
```

In [305…
```
df = df.drop('address', axis=1)
```

In [306…
```
feat_info('issue_d')
```

```
The month which the loan was funded
```

In [307…
```
df = df.drop('issue_d', axis=1)
```

In [308…
```
feat_info('earliest_cr_line')
```

```
The month the borrower's earliest reported credit line was opened
```

In [309…
```
df['earliest_cr_line'] = df['earliest_cr_line'].apply(lambda date: int(date[-2:]))
```

In [310…
```
df['earliest_cr_line']
```

Out[310…
```
0         90
1          4
2          7
3          6
4         99
          ..
396025     4
396026     6
396027    97
396028    90
396029    98
Name: earliest_cr_line, Length: 395219, dtype: int64
```

Preprocessing

In [311…
```
from sklearn.model_selection import train_test_split
```

In [312…
```
df['loan_repaid']
```

Out[312…
```
0          Fully Paid
1          Fully Paid
2          Fully Paid
3          Fully Paid
4         Charged Off
             ...
396025     Fully Paid
396026     Fully Paid
396027     Fully Paid
396028     Fully Paid
396029     Fully Paid
Name: loan_repaid, Length: 395219, dtype: object
```

In [313…
```
df = df.drop('loan_repaid', axis=1)
```

In [314…
```
df.corr()
```

Out[314…

	loan_amnt	term	int_rate	installment	annual_inc	loan_status	dt
loan_amnt	1.000000	0.393731	0.168599	0.953872	0.337364	-0.059731	0.016338
term	0.393731	1.000000	0.434455	0.153040	0.060302	-0.173374	0.036276
int_rate	0.168599	0.434455	1.000000	0.162442	-0.057035	-0.247826	0.078758
installment	0.953872	0.153040	0.162442	1.000000	0.330855	-0.040936	0.015511
annual_inc	0.337364	0.060302	-0.057035	0.330855	1.000000	0.053545	-0.081730
...	...	...	...	...	...	...	...
30723	-0.000204	-0.001271	0.000695	0.000174	0.001218	0.001191	0.003885
48052	0.001001	0.001019	0.000693	0.000858	-0.000309	-0.004072	-0.000971
70466	0.000989	0.003751	0.000941	-0.000004	-0.000332	0.000529	-0.000865
86630	0.019682	0.062450	0.085446	0.012615	-0.019192	-0.341872	0.020679
93700	0.021537	0.055320	0.083114	0.016102	-0.015787	-0.344542	0.020999

78 rows × 78 columns

In [315…
```
X = df.drop('loan_status', axis=1).values
```

In [316…
```
y= df['loan_status']
```

In [317…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [318…
```
from sklearn.preprocessing import MinMaxScaler
```

In [319…
```
scaler = MinMaxScaler()
```

In [320…
```
X_train = scaler.fit_transform(X_train)
```

In [321…
```
X_test = scaler.transform(X_test)
```

Create and training the model

In [322…
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

In [323…
```
#Lets start of Dense layer with our train data shape
```

In [324…
```
X_train.shape
```

Out[324…

```
(316175, 77)
```

In [325…

```
model = Sequential()
model.add(Dense(77, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(36, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(19, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam')
```

In [326…

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

```
Epoch 1/25
1236/1236 [==============================] - 5s 3ms/step - loss: 0.2970 - val_loss:
0.2644
Epoch 2/25
1236/1236 [==============================] - 4s 3ms/step - loss: 0.2655 - val_loss:
0.2646
Epoch 3/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2630 - val_loss:
0.2631
Epoch 4/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2619 - val_loss:
0.2626
Epoch 5/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2613 - val_loss:
0.2624
Epoch 6/25
1236/1236 [==============================] - 4s 4ms/step - loss: 0.2608 - val_loss:
0.2617
Epoch 7/25
1236/1236 [==============================] - 4s 4ms/step - loss: 0.2605 - val_loss:
0.2623
Epoch 8/25
1236/1236 [==============================] - 4s 3ms/step - loss: 0.2601 - val_loss:
0.2613
Epoch 9/25
1236/1236 [==============================] - 4s 3ms/step - loss: 0.2596 - val_loss:
0.2616
Epoch 10/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2593 - val_loss:
0.2615
Epoch 11/25
1236/1236 [==============================] - 4s 4ms/step - loss: 0.2590 - val_loss:
0.2617
Epoch 12/25
1236/1236 [==============================] - 4s 3ms/step - loss: 0.2589 - val_loss:
0.2614
Epoch 13/25
1236/1236 [==============================] - 4s 4ms/step - loss: 0.2586 - val_loss:
0.2613
Epoch 14/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2585 - val_loss:
0.2614
Epoch 15/25
1236/1236 [==============================] - 4s 3ms/step - loss: 0.2584 - val_loss:
0.2612
Epoch 16/25
1236/1236 [==============================] - 4s 3ms/step - loss: 0.2580 - val_loss:
0.2611
Epoch 17/25
1236/1236 [==============================] - 4s 4ms/step - loss: 0.2577 - val_loss:
0.2618
Epoch 18/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2577 - val_loss:
0.2613
Epoch 19/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2575 - val_loss:
```

```
0.2611
Epoch 20/25
1236/1236 [==============================] - 4s 4ms/step - loss: 0.2571 - val_loss:
0.2612
Epoch 21/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2573 - val_loss:
0.2615
Epoch 22/25
1236/1236 [==============================] - 4s 3ms/step - loss: 0.2570 - val_loss:
0.2610
Epoch 23/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2567 - val_loss:
0.2611
Epoch 24/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2566 - val_loss:
0.2608
Epoch 25/25
1236/1236 [==============================] - 5s 4ms/step - loss: 0.2565 - val_loss:
0.2609
```

Out[326…

```
<keras.callbacks.History at 0x1e7640457f0>
```

In [327…

```
#Let's save our model
```

In [328…

```
from tensorflow.keras.models import load_model
```

In [329…

```
model.save('LoanApprovalModel')
```

```
INFO:tensorflow:Assets written to: LoanApprovalModel\assets
```

In [330…

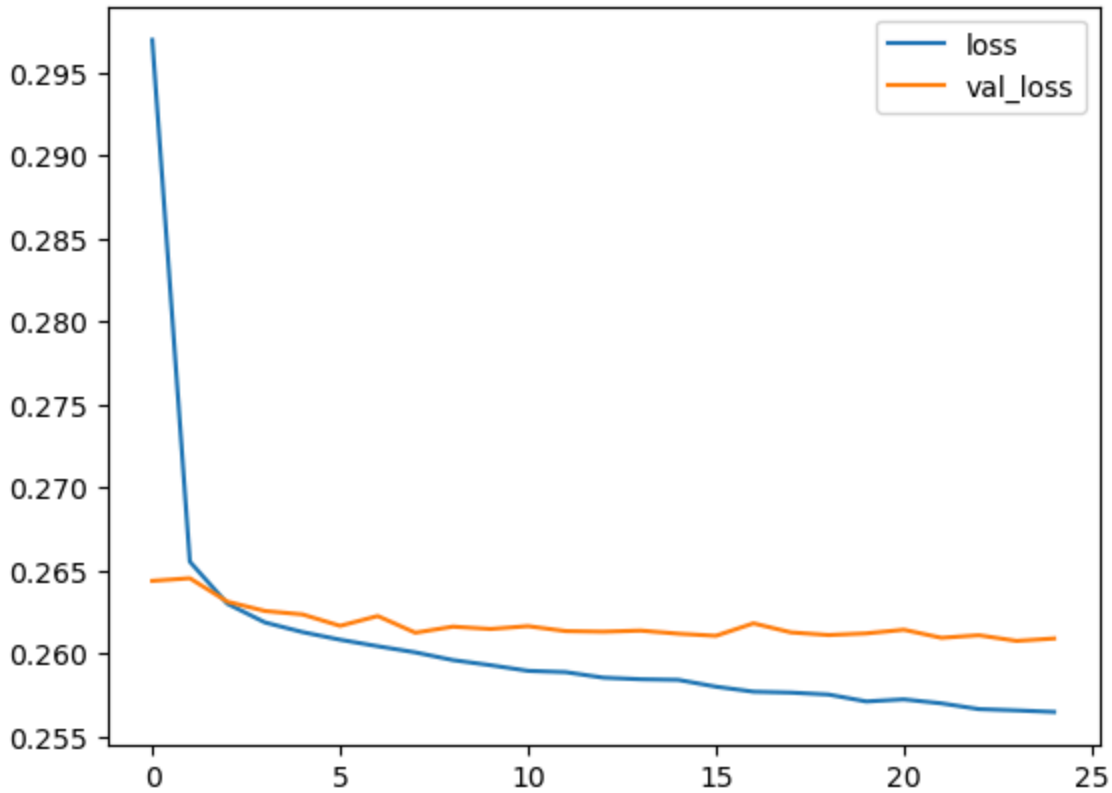
```
losses = pd.DataFrame(model.history.history)
```

In [331…

```
losses.plot()
```

Out[331…

```
<Axes: >
```



In [332…
```
# Create predictions from the X_test set and display classification report and conf
```

In [333…
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [344…
```
predictions = model.predict(X_test)
# Ensure that target values are integers
y_test = y_test.astype(int)
```

```
2471/2471 [==============================] - 6s 2ms/step
```

In [345…
```
df['loan_status'].value_counts()
```

Out[345…
```
loan_status
1    317696
0     77523
Name: count, dtype: int64
```

In [346…
```
#Lets check if the loan are repaid or not
```

In [347…
```
317696/len(df)
```

Out[347…
```
0.8038479931379817
```

In [351…
```
# Get predicted probabilities
predicted_probabilities = model.predict(X_test)

# Convert probabilities to binary labels using a threshold of 0.5
predicted_classes = (predicted_probabilities >= 0.5).astype(int)
```

```
2471/2471 [==============================] - 7s 3ms/step
```

In [352…
```
print(classification_report(y_test, predicted_classes))
```

```
              precision    recall  f1-score   support

           0       0.96      0.45      0.62     15658
           1       0.88      1.00      0.93     63386

    accuracy                           0.89     79044
   macro avg       0.92      0.72      0.78     79044
weighted avg       0.90      0.89      0.87     79044
```

In [353…
```
print(confusion_matrix(y_test, predicted_classes))
```

```
[[ 7122  8536]
 [  310 63076]]
```

In [354…
```
#Let's test if a loan approval is working on random person
```

In [366…
```
import random
random.seed(101)
random_ind = random.randint(0, len(df))
new_customer = df.drop('loan_status', axis=1).iloc[random_ind]
new_customer
```

Out[366…
```
loan_amnt        25000
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: 77, dtype: object
```

In [370…
```
# Reshape the new_customer data into a 2D array (1 sample, n_features)
new_customer_reshaped = new_customer.values.reshape(1, -1)

# Apply the scaler and make the prediction
scaled_new_customer = scaler.transform(new_customer_reshaped)
prediction = model.predict(scaled_new_customer)
# Convert probabilities to binary labels using a threshold of 0.5
predicted_classes = (prediction >= 0.5).astype(int)

print(predicted_classes)
```

```
1/1 [==============================] - 0s 48ms/step
[[1]]
```

In [371…
```
# Reshape the new_customer data into a 2D array (1 sample, n_features)
new_customer_reshaped = new_customer.values.reshape(1, -1)

# Apply the scaler and make the prediction
```

```
scaled_new_customer = scaler.transform(new_customer_reshaped)
prediction = model.predict(scaled_new_customer)
# Convert probabilities to binary labels using a threshold of 0.5
predicted_classes = (prediction >= 0.5).astype(int)

print(predicted_classes)
```

```
1/1 [==============================] - 0s 35ms/step
[[1]]
```

In [372…

```
#Now lets check if this person endup paying loan or not
```

In [376…

```
# Take the record from original dataset and check for the loan status to compare wi
df.iloc[random_ind]['loan_status']
```

Out[376…

```
1
```

Conclusion Our prediction and actual loat status are equal. It means our model is performing good

In [ ]:

```

```