```
#import the dependencies
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from google.colab import drive
drive.mount('../content/drive')
```

```
Drive already mounted at ../content/drive; to attempt to forcibly
remount, call drive.mount("../content/drive", force_remount=True).
```

```
##%%shell
!jupyter nbconvert --to html /content/drive/Himamshu_AML_LDP.ipynb
```

```
[NbConvertApp] WARNING | pattern
'/content/sample_data/Himamshu_AML_LDP.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE
RELEASES.

Options
=======
The options below are convenience aliases to configurable class-
options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
```

```
--allow-errors
    Continue notebook execution even if one of the cells throws an
error and include the error message in the cell output (the default
behaviour is to abort conversion). This flag is only relevant if '--
execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting
notebook with default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --
NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --
NbConvertApp.export_format=notebook --FilesWriter.build_directory= --
ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --
TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --
TemplateExporter.exclude_input=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN',
'ERROR', 'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'rst', 'script', 'slides']
            or a dotted object name that represents the import path
for an
            `Exporter` class
```

```
    Default: 'html'
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template file to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_file]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a
time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                    to output to the directory of each
notebook. To recover
                                    previous default behaviour
(outputting to the current
                                    working directory) use . as the flag
value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url
pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to
a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of
the
            current directory (from which the server is run).
            See the usage documentation

(https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-
slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
```

```
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb

            which will convert mynotebook.ipynb to the default format
(probably HTML).

            You can specify the export format with `--to`.
            Options include ['asciidoc', 'custom', 'html', 'latex',
'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates.
LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic' and
'full'. You
            can specify the flavor of the format used.

            > jupyter nbconvert --to html --template basic
mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post
serve

            Multiple notebooks can be given at the command line in a
couple of
            different ways:
```

```
            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file,
containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.


loanData = pd.read_csv("/content/lending_club_loans.csv")

/usr/local/lib/python3.8/dist-packages/IPython/core/
interactiveshell.py:3326: DtypeWarning: Columns (49) have mixed
types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)

from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

New Section

Delete 1st row of loan dataset

```
#loanData = loanData.tail(loanData.shape[0] -1)

display(loanData)
#42542 rows and 115 columns

            id  member_id  loan_amnt  funded_amnt  funded_amnt_inv  \
0      1077501    1296599       5000         5000           4975.0
1      1077430    1314167       2500         2500           2500.0
2      1077175    1313524       2400         2400           2400.0
3      1076863    1277178      10000        10000          10000.0
4      1075358    1311748       3000         3000           3000.0
...        ...        ...        ...          ...              ...
42530    73582      73096       3500         3500            225.0
42531    72998      72992       1000         1000              0.0
42532    72176      70868       2525         2525            225.0
42533    71623      70735       6500         6500              0.0
42534    70686      70681       5000         5000              0.0

             term int_rate  installment grade sub_grade  ...  \
0       36 months   10.65%       162.87     B        B2  ...
```

```
1       60 months   15.27%        59.83     C        C4  ...
2       36 months   15.96%        84.33     C        C5  ...
3       36 months   13.49%       339.31     C        C1  ...
4       60 months   12.69%        67.79     B        B5  ...
...           ...      ...          ...   ...       ...  ...
42530   36 months   10.28%       113.39     C        C1  ...
42531   36 months    9.64%        32.11     B        B4  ...
42532   36 months    9.33%        80.69     B        B3  ...
42533   36 months    8.38%       204.84     A        A5  ...
42534   36 months    7.75%       156.11     A        A3  ...

      num_tl_90g_dpd_24m num_tl_op_past_12m pct_tl_nvr_dlq
percent_bc_gt_75  \
0                    NaN                NaN            NaN
NaN
1                    NaN                NaN            NaN
NaN
2                    NaN                NaN            NaN
NaN
3                    NaN                NaN            NaN
NaN
4                    NaN                NaN            NaN
NaN
...                  ...                ...            ...
...
42530                NaN                NaN            NaN
NaN
42531                NaN                NaN            NaN
NaN
42532                NaN                NaN            NaN
NaN
42533                NaN                NaN            NaN
NaN
42534                NaN                NaN            NaN
NaN

      pub_rec_bankruptcies tax_liens tot_hi_cred_lim total_bal_ex_mort
\
0                      0.0       0.0             NaN               NaN

1                      0.0       0.0             NaN               NaN

2                      0.0       0.0             NaN               NaN

3                      0.0       0.0             NaN               NaN

4                      0.0       0.0             NaN               NaN

...                    ...       ...             ...               ...
```

```
42530                  NaN       NaN             NaN                NaN

42531                  NaN       NaN             NaN                NaN

42532                  NaN       NaN             NaN                NaN

42533                  NaN       NaN             NaN                NaN

42534                  NaN       NaN             NaN                NaN


      total_bc_limit total_il_high_credit_limit
0                NaN                       NaN
1                NaN                       NaN
2                NaN                       NaN
3                NaN                       NaN
4                NaN                       NaN
...              ...                       ...
42530            NaN                       NaN
42531            NaN                       NaN
42532            NaN                       NaN
42533            NaN                       NaN
42534            NaN                       NaN

[42535 rows x 115 columns]

list(loanData.columns)

['id',
 'member_id',
 'loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'term',
 'int_rate',
 'installment',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'annual_inc',
 'verification_status',
 'issue_d',
 'loan_status',
 'pymnt_plan',
 'url',
 'desc',
 'purpose',
```

```
'title',
'zip_code',
'addr_state',
'dti',
'delinq_2yrs',
'earliest_cr_line',
'fico_range_low',
'fico_range_high',
'inq_last_6mths',
'mths_since_last_delinq',
'mths_since_last_record',
'open_acc',
'pub_rec',
'revol_bal',
'revol_util',
'total_acc',
'initial_list_status',
'out_prncp',
'out_prncp_inv',
'total_pymnt',
'total_pymnt_inv',
'total_rec_prncp',
'total_rec_int',
'total_rec_late_fee',
'recoveries',
'collection_recovery_fee',
'last_pymnt_d',
'last_pymnt_amnt',
'next_pymnt_d',
'last_credit_pull_d',
'last_fico_range_high',
'last_fico_range_low',
'collections_12_mths_ex_med',
'mths_since_last_major_derog',
'policy_code',
'application_type',
'annual_inc_joint',
'dti_joint',
'verification_status_joint',
'acc_now_delinq',
'tot_coll_amt',
'tot_cur_bal',
'open_acc_6m',
'open_il_6m',
'open_il_12m',
'open_il_24m',
'mths_since_rcnt_il',
'total_bal_il',
'il_util',
'open_rv_12m',
```

```
 'open_rv_24m',
 'max_bal_bc',
 'all_util',
 'total_rev_hi_lim',
 'inq_fi',
 'total_cu_tl',
 'inq_last_12m',
 'acc_open_past_24mths',
 'avg_cur_bal',
 'bc_open_to_buy',
 'bc_util',
 'chargeoff_within_12_mths',
 'delinq_amnt',
 'mo_sin_old_il_acct',
 'mo_sin_old_rev_tl_op',
 'mo_sin_rcnt_rev_tl_op',
 'mo_sin_rcnt_tl',
 'mort_acc',
 'mths_since_recent_bc',
 'mths_since_recent_bc_dlq',
 'mths_since_recent_inq',
 'mths_since_recent_revol_delinq',
 'num_accts_ever_120_pd',
 'num_actv_bc_tl',
 'num_actv_rev_tl',
 'num_bc_sats',
 'num_bc_tl',
 'num_il_tl',
 'num_op_rev_tl',
 'num_rev_accts',
 'num_rev_tl_bal_gt_0',
 'num_sats',
 'num_tl_120dpd_2m',
 'num_tl_30dpd',
 'num_tl_90g_dpd_24m',
 'num_tl_op_past_12m',
 'pct_tl_nvr_dlq',
 'percent_bc_gt_75',
 'pub_rec_bankruptcies',
 'tax_liens',
 'tot_hi_cred_lim',
 'total_bal_ex_mort',
 'total_bc_limit',
 'total_il_high_credit_limit']
```

First 10 ['id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length']

loanData datafarame is created.

```
for col in loanData.columns:
  if loanData[col].isnull().sum()==len(loanData):
    loanData=loanData.drop(col,axis=1)


# Check for null values
for col in loanData.columns:
    print("{} : {}".format(col, loanData[col].isnull().sum()))
```

```
id : 0
member_id : 0
loan_amnt : 0
funded_amnt : 0
funded_amnt_inv : 0
term : 0
int_rate : 0
installment : 0
grade : 0
sub_grade : 0
emp_title : 2626
emp_length : 1112
home_ownership : 0
annual_inc : 4
verification_status : 0
issue_d : 0
loan_status : 0
pymnt_plan : 0
url : 0
desc : 13293
purpose : 0
title : 13
zip_code : 0
addr_state : 0
dti : 0
delinq_2yrs : 29
earliest_cr_line : 29
fico_range_low : 0
fico_range_high : 0
inq_last_6mths : 29
mths_since_last_delinq : 26926
mths_since_last_record : 38884
open_acc : 29
pub_rec : 29
revol_bal : 0
revol_util : 90
total_acc : 29
initial_list_status : 0
out_prncp : 0
out_prncp_inv : 0
total_pymnt : 0
```

```
total_pymnt_inv : 0
total_rec_prncp : 0
total_rec_int : 0
total_rec_late_fee : 0
recoveries : 0
collection_recovery_fee : 0
last_pymnt_d : 83
last_pymnt_amnt : 0
next_pymnt_d : 39239
last_credit_pull_d : 4
last_fico_range_high : 0
last_fico_range_low : 0
collections_12_mths_ex_med : 145
policy_code : 0
application_type : 0
acc_now_delinq : 29
chargeoff_within_12_mths : 145
delinq_amnt : 29
pub_rec_bankruptcies : 1365
tax_liens : 105
```

```
loanData.loan_status.value_counts()
```

```
Fully Paid                                             33586
Charged Off                                             5653
Does not meet the credit policy. Status:Fully Paid      1988
Does not meet the credit policy. Status:Charged Off      761
Current                                                  513
In Grace Period                                           16
Late (31-120 days)                                        12
Late (16-30 days)                                          5
Default                                                    1
Name: loan_status, dtype: int64
```

```
# This is formatted as code
```

#Charged off - 5653 people

```
loanData.shape
#reduced to 61 columns
```

```
(42535, 61)
```

```
loanData.dtypes
```

```
id                            int64
member_id                     int64
loan_amnt                     int64
funded_amnt                   int64
funded_amnt_inv             float64
                             ...
acc_now_delinq              float64
```

```
chargeoff_within_12_mths    float64
delinq_amnt                 float64
pub_rec_bankruptcies        float64
tax_liens                   float64
Length: 61, dtype: object
```

```
# iterating the columns
for col in loanData.columns:
    print(col)
```

```
id
member_id
loan_amnt
funded_amnt
funded_amnt_inv
term
int_rate
installment
grade
sub_grade
emp_title
emp_length
home_ownership
annual_inc
verification_status
issue_d
loan_status
pymnt_plan
url
desc
purpose
title
zip_code
addr_state
dti
delinq_2yrs
earliest_cr_line
fico_range_low
fico_range_high
inq_last_6mths
mths_since_last_delinq
mths_since_last_record
open_acc
pub_rec
revol_bal
revol_util
total_acc
initial_list_status
out_prncp
out_prncp_inv
total_pymnt
```

```
total_pymnt_inv
total_rec_prncp
total_rec_int
total_rec_late_fee
recoveries
collection_recovery_fee
last_pymnt_d
last_pymnt_amnt
next_pymnt_d
last_credit_pull_d
last_fico_range_high
last_fico_range_low
collections_12_mths_ex_med
policy_code
application_type
acc_now_delinq
chargeoff_within_12_mths
delinq_amnt
pub_rec_bankruptcies
tax_liens
```

Loan_amnt ✔

- integer type
- NO NA

```
loanData["loan_amnt"].dtypes
```

```
dtype('int64')
```

```
loanData["loan_amnt"].describe
```

```
<bound method NDFrame.describe of 0         5000
1         2500
2         2400
3        10000
4         3000
         ...
42530     3500
42531     1000
42532     2525
42533     6500
42534     5000
Name: loan_amnt, Length: 42535, dtype: int64>
```

```
loanData["loan_amnt"].isna().sum()
```

```
0
```

Funded amount ✔

funded_amnt

- integer type
- NO NA

```
loanData["funded_amnt"].dtypes
```

```
dtype('int64')
```

```
loanData["funded_amnt"].describe
```

```
<bound method NDFrame.describe of 0         5000
1         2500
2         2400
3        10000
4         3000
         ...
42530     3500
42531     1000
42532     2525
42533     6500
42534     5000
Name: funded_amnt, Length: 42535, dtype: int64>
```

```
loanData["funded_amnt"].isna().sum()
```

```
0
```

Funded_amnt_inv ✓

- integer type
- NO NA

```
loanData["funded_amnt_inv"].isna().sum()
```

```
0
```

```
loanData["funded_amnt_inv"].dtypes
```

```
dtype('float64')
```

```
loanData["funded_amnt"].describe
```

```
<bound method NDFrame.describe of 0         5000
1         2500
2         2400
3        10000
4         3000
         ...
42530     3500
42531     1000
42532     2525
42533     6500
42534     5000
Name: funded_amnt, Length: 42535, dtype: int64>
```

Term_months ✓

- integer type
- NO NA

```
loanData.rename(columns={'term': 'term_months'}, inplace=True)
```

```
loanData["term_months"].dtypes
```

```
dtype('O')
```

Replaced months string and converted Term months to number of months in int

```
loanData["term_months"].dtypes
```

```
dtype('O')
```

```
loanData[loanData.term_months.isna()]
```

```
Empty DataFrame
Columns: [id, member_id, loan_amnt, funded_amnt, funded_amnt_inv,
term_months, int_rate, installment, grade, sub_grade, emp_title,
emp_length, home_ownership, annual_inc, verification_status, issue_d,
loan_status, pymnt_plan, url, desc, purpose, title, zip_code,
addr_state, dti, delinq_2yrs, earliest_cr_line, fico_range_low,
fico_range_high, inq_last_6mths, mths_since_last_delinq,
mths_since_last_record, open_acc, pub_rec, revol_bal, revol_util,
total_acc, initial_list_status, out_prncp, out_prncp_inv, total_pymnt,
total_pymnt_inv, total_rec_prncp, total_rec_int, total_rec_late_fee,
recoveries, collection_recovery_fee, last_pymnt_d, last_pymnt_amnt,
next_pymnt_d, last_credit_pull_d, last_fico_range_high,
last_fico_range_low, collections_12_mths_ex_med, policy_code,
application_type, acc_now_delinq, chargeoff_within_12_mths,
delinq_amnt, pub_rec_bankruptcies, tax_liens]
Index: []

[0 rows x 61 columns]
```

```
loanData = loanData.drop(loanData.term_months.isna().index)
```

```
loanData.term_months.isna().sum()
```

```
0
```

```
loanData.term_months =
loanData.term_months.str.replace("months","").astype(int)
```

Installment ✓

- Float type
- NO NA

```
loanData.installment.dtypes
```

```
dtype('float64')
```

```
loanData.installment.isna().sum()
```

```
0
```

Grade ✅

- String type
- NO NA

```
loanData.grade = loanData.grade.astype(str)
```

```
loanData.grade.isna().sum()
```

```
0
```

Convert dtype[0] object to string

```
loanData['grade'] = loanData['grade'].astype('|S') # which will by
default set the length to the max len it encounters
#loanData['grade'] = loanData['grade'].astype(str)
```

```
loanData.grade.dtypes
```

```
dtype('S1')
```

Sub grade ✅

String type

No NA

```
loanData.sub_grade.isna().sum()
```

```
0
```

Convert dtype[0] object to string

```
loanData.sub_grade = loanData.sub_grade.astype('|S')
```

Employee Title ⛔

```
loanData.emp_title.isna().sum()
```

```
0
```

6892 null values present in employee title - its ok since its description. No operation done.

```
#Throwing error
#loanData["emp_title"] = loanData["emp_title"].astype('|S')
```

Employee Length ⛔

```
loanData.emp_title.isna().sum()
```

```
0
```

```
loanData.emp_title = loanData.emp_title.astype(str)

loanData.emp_title.dtypes

dtype('O')
```

Interest Rate

```
loanData["int_rate"].describe

<bound method NDFrame.describe of Series([], Name: int_rate, dtype:
object)>

loanData['int_rate'] = loanData["int_rate"].apply(lambda x:
str(x).strip('%'))

len([rate for rate in loanData['int_rate'] if '%' in str(rate)])

0

loanData['int_rate'] = loanData['int_rate'].astype(float)

print(" \n Description of each column with missing values \n",
loanData.isnull().sum())



#Total number of missing values NaN at each column in a DataFrame

print(" \n Total number of missing values NaN in the DataFrame : \n\
n", loanData.isnull().sum().sum())



#Total number of columns with missing values with axis = 0 for column
wise operation

print(" \n Total number of columns with missing values : \n\n",
loanData.isnull().any(axis=0).sum())



#Total number of records with missing values axis = 1 for row wise
operation

print(" \n Total number of records with missing values : \n\n",
loanData.isnull().any(axis=1).sum() )


 Description of each column with missing values
 id                          0.0
```

```
member_id                   0.0
loan_amnt                   0.0
funded_amnt                 0.0
funded_amnt_inv             0.0
                           ...
acc_now_delinq              0.0
chargeoff_within_12_mths    0.0
delinq_amnt                 0.0
pub_rec_bankruptcies        0.0
tax_liens                   0.0
Length: 61, dtype: float64

 Total number of missing values NaN in the DataFrame :

 0.0

 Total number of columns with missing values :

 0

 Total number of records with missing values :

 0
```

```
loanData.int_rate.isna().sum()
```

```
0
```

ID

```
loanData.id.isna().sum()
```

```
0
```

Member id

```
loanData.member_id.isna().sum()
```

```
0
```

```
#import the dependencies
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from google.colab import drive
drive.mount('../content/drive')
```

```
Drive already mounted at ../content/drive; to attempt to forcibly
remount, call drive.mount("../content/drive", force_remount=True).
```

```
##%%shell
##jupyter nbconvert --to html ../content/LoanStats_2017Q1 2.csv
```

```
loanData = pd.read_csv("/content/lending_club_loans.csv")
```

```
/usr/local/lib/python3.8/dist-packages/IPython/core/
interactiveshell.py:3326: DtypeWarning: Columns (49) have mixed
types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

Data Cleaning

```
display(loanData)
#42542 rows and 115 columns
```

```
            id  member_id  loan_amnt  funded_amnt  funded_amnt_inv  \
0      1077501    1296599       5000         5000           4975.0
1      1077430    1314167       2500         2500           2500.0
2      1077175    1313524       2400         2400           2400.0
3      1076863    1277178      10000        10000          10000.0
4      1075358    1311748       3000         3000           3000.0
...        ...        ...        ...          ...              ...
42530    73582      73096       3500         3500            225.0
42531    72998      72992       1000         1000              0.0
42532    72176      70868       2525         2525            225.0
42533    71623      70735       6500         6500              0.0
42534    70686      70681       5000         5000              0.0

             term int_rate  installment grade sub_grade  ...  \
0       36 months   10.65%       162.87     B        B2  ...
1       60 months   15.27%        59.83     C        C4  ...
2       36 months   15.96%        84.33     C        C5  ...
3       36 months   13.49%       339.31     C        C1  ...
4       60 months   12.69%        67.79     B        B5  ...
...           ...      ...          ...   ...       ...  ...
42530   36 months   10.28%       113.39     C        C1  ...
42531   36 months    9.64%        32.11     B        B4  ...
42532   36 months    9.33%        80.69     B        B3  ...
42533   36 months    8.38%       204.84     A        A5  ...
42534   36 months    7.75%       156.11     A        A3  ...

      num_tl_90g_dpd_24m num_tl_op_past_12m pct_tl_nvr_dlq
```

```
percent_bc_gt_75  \
0                    NaN                NaN            NaN
NaN
1                    NaN                NaN            NaN
NaN
2                    NaN                NaN            NaN
NaN
3                    NaN                NaN            NaN
NaN
4                    NaN                NaN            NaN
NaN
...                  ...                ...            ...
...
42530                NaN                NaN            NaN
NaN
42531                NaN                NaN            NaN
NaN
42532                NaN                NaN            NaN
NaN
42533                NaN                NaN            NaN
NaN
42534                NaN                NaN            NaN
NaN

      pub_rec_bankruptcies tax_liens tot_hi_cred_lim total_bal_ex_mort
\
0                      0.0       0.0             NaN               NaN

1                      0.0       0.0             NaN               NaN

2                      0.0       0.0             NaN               NaN

3                      0.0       0.0             NaN               NaN

4                      0.0       0.0             NaN               NaN

...                    ...       ...             ...               ...

42530                  NaN       NaN             NaN               NaN

42531                  NaN       NaN             NaN               NaN

42532                  NaN       NaN             NaN               NaN

42533                  NaN       NaN             NaN               NaN

42534                  NaN       NaN             NaN               NaN
```

```
      total_bc_limit total_il_high_credit_limit
0                NaN                       NaN
1                NaN                       NaN
2                NaN                       NaN
3                NaN                       NaN
4                NaN                       NaN
...              ...                       ...
42530            NaN                       NaN
42531            NaN                       NaN
42532            NaN                       NaN
42533            NaN                       NaN
42534            NaN                       NaN

[42535 rows x 115 columns]
```

```
list(loanData.columns)
```

```
['id',
 'member_id',
 'loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'term',
 'int_rate',
 'installment',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'annual_inc',
 'verification_status',
 'issue_d',
 'loan_status',
 'pymnt_plan',
 'url',
 'desc',
 'purpose',
 'title',
 'zip_code',
 'addr_state',
 'dti',
 'delinq_2yrs',
 'earliest_cr_line',
 'fico_range_low',
 'fico_range_high',
 'inq_last_6mths',
 'mths_since_last_delinq',
 'mths_since_last_record',
 'open_acc',
```

```
'pub_rec',
'revol_bal',
'revol_util',
'total_acc',
'initial_list_status',
'out_prncp',
'out_prncp_inv',
'total_pymnt',
'total_pymnt_inv',
'total_rec_prncp',
'total_rec_int',
'total_rec_late_fee',
'recoveries',
'collection_recovery_fee',
'last_pymnt_d',
'last_pymnt_amnt',
'next_pymnt_d',
'last_credit_pull_d',
'last_fico_range_high',
'last_fico_range_low',
'collections_12_mths_ex_med',
'mths_since_last_major_derog',
'policy_code',
'application_type',
'annual_inc_joint',
'dti_joint',
'verification_status_joint',
'acc_now_delinq',
'tot_coll_amt',
'tot_cur_bal',
'open_acc_6m',
'open_il_6m',
'open_il_12m',
'open_il_24m',
'mths_since_rcnt_il',
'total_bal_il',
'il_util',
'open_rv_12m',
'open_rv_24m',
'max_bal_bc',
'all_util',
'total_rev_hi_lim',
'inq_fi',
'total_cu_tl',
'inq_last_12m',
'acc_open_past_24mths',
'avg_cur_bal',
'bc_open_to_buy',
'bc_util',
'chargeoff_within_12_mths',
```

```
 'delinq_amnt',
 'mo_sin_old_il_acct',
 'mo_sin_old_rev_tl_op',
 'mo_sin_rcnt_rev_tl_op',
 'mo_sin_rcnt_tl',
 'mort_acc',
 'mths_since_recent_bc',
 'mths_since_recent_bc_dlq',
 'mths_since_recent_inq',
 'mths_since_recent_revol_delinq',
 'num_accts_ever_120_pd',
 'num_actv_bc_tl',
 'num_actv_rev_tl',
 'num_bc_sats',
 'num_bc_tl',
 'num_il_tl',
 'num_op_rev_tl',
 'num_rev_accts',
 'num_rev_tl_bal_gt_0',
 'num_sats',
 'num_tl_120dpd_2m',
 'num_tl_30dpd',
 'num_tl_90g_dpd_24m',
 'num_tl_op_past_12m',
 'pct_tl_nvr_dlq',
 'percent_bc_gt_75',
 'pub_rec_bankruptcies',
 'tax_liens',
 'tot_hi_cred_lim',
 'total_bal_ex_mort',
 'total_bc_limit',
 'total_il_high_credit_limit']
```

First 10 ['id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length']

loanData datafarame is created.

```
for col in loanData.columns:
  if loanData[col].isnull().sum()==len(loanData):
    loanData=loanData.drop(col,axis=1)


# Check for null values
for col in loanData.columns:
    print("{} : {}".format(col, loanData[col].isnull().sum()))
```

```
id : 0
member_id : 0
loan_amnt : 0
funded_amnt : 0
```

```
funded_amnt_inv : 0
term : 0
int_rate : 0
installment : 0
grade : 0
sub_grade : 0
emp_title : 2626
emp_length : 1112
home_ownership : 0
annual_inc : 4
verification_status : 0
issue_d : 0
loan_status : 0
pymnt_plan : 0
url : 0
desc : 13293
purpose : 0
title : 13
zip_code : 0
addr_state : 0
dti : 0
delinq_2yrs : 29
earliest_cr_line : 29
fico_range_low : 0
fico_range_high : 0
inq_last_6mths : 29
mths_since_last_delinq : 26926
mths_since_last_record : 38884
open_acc : 29
pub_rec : 29
revol_bal : 0
revol_util : 90
total_acc : 29
initial_list_status : 0
out_prncp : 0
out_prncp_inv : 0
total_pymnt : 0
total_pymnt_inv : 0
total_rec_prncp : 0
total_rec_int : 0
total_rec_late_fee : 0
recoveries : 0
collection_recovery_fee : 0
last_pymnt_d : 83
last_pymnt_amnt : 0
next_pymnt_d : 39239
last_credit_pull_d : 4
last_fico_range_high : 0
last_fico_range_low : 0
collections_12_mths_ex_med : 145
```

```
policy_code : 0
application_type : 0
acc_now_delinq : 29
chargeoff_within_12_mths : 145
delinq_amnt : 29
pub_rec_bankruptcies : 1365
tax_liens : 105
```

```
loanData.loan_status.value_counts()
```

```
Fully Paid                                             33586
Charged Off                                             5653
Does not meet the credit policy. Status:Fully Paid      1988
Does not meet the credit policy. Status:Charged Off      761
Current                                                  513
In Grace Period                                           16
Late (31-120 days)                                        12
Late (16-30 days)                                          5
Default                                                    1
Name: loan_status, dtype: int64
```

```
# This is formatted as code
```

#Charged off - 5653 people

```
loanData.shape
#reduced to 61 columns
```

```
(42535, 61)
```

```
loanData.dtypes
```

```
id                            int64
member_id                     int64
loan_amnt                     int64
funded_amnt                   int64
funded_amnt_inv             float64
                             ...
acc_now_delinq              float64
chargeoff_within_12_mths    float64
delinq_amnt                 float64
pub_rec_bankruptcies        float64
tax_liens                   float64
Length: 61, dtype: object
```

```
# iterating the columns
for col in loanData.columns:
    print(col)
```

```
id
member_id
loan_amnt
```

funded_amnt
funded_amnt_inv
term
int_rate
installment
grade
sub_grade
emp_title
emp_length
home_ownership
annual_inc
verification_status
issue_d
loan_status
pymnt_plan
url
desc
purpose
title
zip_code
addr_state
dti
delinq_2yrs
earliest_cr_line
fico_range_low
fico_range_high
inq_last_6mths
mths_since_last_delinq
mths_since_last_record
open_acc
pub_rec
revol_bal
revol_util
total_acc
initial_list_status
out_prncp
out_prncp_inv
total_pymnt
total_pymnt_inv
total_rec_prncp
total_rec_int
total_rec_late_fee
recoveries
collection_recovery_fee
last_pymnt_d
last_pymnt_amnt
next_pymnt_d
last_credit_pull_d
last_fico_range_high
last_fico_range_low

```
collections_12_mths_ex_med
policy_code
application_type
acc_now_delinq
chargeoff_within_12_mths
delinq_amnt
pub_rec_bankruptcies
tax_liens
```

Loan_amnt ✓

- integer type
- 7 NA

```
loanData["loan_amnt"].dtypes
```

```
dtype('int64')
```

```
loanData["loan_amnt"].describe
```

```
<bound method NDFrame.describe of 0         5000
1         2500
2         2400
3        10000
4         3000
         ...
42530     3500
42531     1000
42532     2525
42533     6500
42534     5000
Name: loan_amnt, Length: 42535, dtype: int64>
```

```
loanData["loan_amnt"].isna().sum()
```

```
0
```

Funded amount ✓

funded_amnt

- integer type
- 7 NA

```
loanData["funded_amnt"].dtypes
```

```
dtype('int64')
```

```
loanData["funded_amnt"].describe
```

```
<bound method NDFrame.describe of 0         5000
1         2500
2         2400
3        10000
```

```
4         3000
         ...
42530     3500
42531     1000
42532     2525
42533     6500
42534     5000
Name: funded_amnt, Length: 42535, dtype: int64>
```

```
loanData["funded_amnt"].isna().sum()
```

```
0
```

Funded_amnt_inv ✓

- integer type
- 7 NA

```
loanData["funded_amnt_inv"].isna().sum()
```

```
0
```

```
loanData["funded_amnt_inv"].dtypes
```

```
dtype('float64')
```

```
loanData["funded_amnt"].describe
```

```
<bound method NDFrame.describe of 0         5000
1         2500
2         2400
3        10000
4         3000
         ...
42530     3500
42531     1000
42532     2525
42533     6500
42534     5000
Name: funded_amnt, Length: 42535, dtype: int64>
```

Term_months ✓

- integer type
- NO NA

```
loanData.rename(columns={'term': 'term_months'}, inplace=True)
```

```
loanData["term_months"].dtypes
```

```
dtype('O')
```

Replaced months string and converted Term months to number of months in int

```
loanData["term_months"].dtypes
```

```
dtype('O')
```

```
loanData[loanData.term_months.isna()]
```

```
Empty DataFrame
Columns: [id, member_id, loan_amnt, funded_amnt, funded_amnt_inv,
term_months, int_rate, installment, grade, sub_grade, emp_title,
emp_length, home_ownership, annual_inc, verification_status, issue_d,
loan_status, pymnt_plan, url, desc, purpose, title, zip_code,
addr_state, dti, delinq_2yrs, earliest_cr_line, fico_range_low,
fico_range_high, inq_last_6mths, mths_since_last_delinq,
mths_since_last_record, open_acc, pub_rec, revol_bal, revol_util,
total_acc, initial_list_status, out_prncp, out_prncp_inv, total_pymnt,
total_pymnt_inv, total_rec_prncp, total_rec_int, total_rec_late_fee,
recoveries, collection_recovery_fee, last_pymnt_d, last_pymnt_amnt,
next_pymnt_d, last_credit_pull_d, last_fico_range_high,
last_fico_range_low, collections_12_mths_ex_med, policy_code,
application_type, acc_now_delinq, chargeoff_within_12_mths,
delinq_amnt, pub_rec_bankruptcies, tax_liens]
Index: []

[0 rows x 61 columns]
```

```
#Drop rows where term months have NA
loanData = loanData.drop(loanData[loanData.term_months.isna()].index)
```

```
loanData.term_months.isna().sum()
```

```
0
```

```
loanData.term_months =
loanData.term_months.str.replace("months","").astype(int)
```

Installment ✅

- Float type
- NO NA

```
loanData.installment.dtypes
```

```
dtype('float64')
```

```
loanData.installment.isna().sum()
```

```
0
```

Grade ✅

- String type
- NO NA

```
loanData.grade = loanData.grade.astype(str)
```

```
loanData.grade.isna().sum()
```

```
0
```

Convert dtype[0] object to string

```
loanData['grade'] = loanData['grade'].astype('|S') # which will by
default set the length to the max len it encounters
#loanData['grade'] = loanData['grade'].astype(str)
```

```
loanData.grade.dtypes
```

```
dtype('S1')
```

Sub grade ✅

- String type
- No NA

```
loanData.sub_grade.isna().sum()
```

```
0
```

Convert dtype[0] object to string

```
loanData.sub_grade = loanData.sub_grade.astype('|S')
```

Employee Title ⛔

```
loanData.emp_title.isna().sum()
```

```
2626
```

6892 null values present in employee title - its ok since its description. No operation done.

```
#Throwing error
#loanData["emp_title"] = loanData["emp_title"].astype('|S')
```

Employee Length ⛔

```
loanData.emp_title.isna().sum()
```

```
2626
```

```
loanData.emp_title = loanData.emp_title.astype(str)
```

```
loanData.emp_title.dtypes
```

```
dtype('O')
```

Interest Rate

```
loanData["int_rate"].describe
```

```
<bound method NDFrame.describe of 0        10.65%
1        15.27%
2        15.96%
```

```
3        13.49%
4        12.69%
          ...
42530    10.28%
42531     9.64%
42532     9.33%
42533     8.38%
42534     7.75%
Name: int_rate, Length: 42535, dtype: object>
```

```
loanData['int_rate'] = loanData["int_rate"].apply(lambda x:
str(x).strip('%'))
```

```
len([rate for rate in loanData['int_rate'] if '%' in str(rate)])
```

```
0
```

```
loanData['int_rate'] = loanData['int_rate'].astype(float)
```

```
print(" \n Description of each column with missing values \n",
loanData.isnull().sum())
```

```
#Total number of missing values NaN at each column in a DataFrame
```

```
print(" \n Total number of missing values NaN in the DataFrame : \n\
n", loanData.isnull().sum().sum())
```

```
#Total number of columns with missing values with axis = 0 for column
wise operation
```

```
print(" \n Total number of columns with missing values : \n\n",
loanData.isnull().any(axis=0).sum())
```

```
#Total number of records with missing values axis = 1 for row wise
operation
```

```
print(" \n Total number of records with missing values : \n\n",
loanData.isnull().any(axis=1).sum() )
```

```
 Description of each column with missing values
 id                              0
member_id                       0
loan_amnt                       0
```

```
funded_amnt                   0
funded_amnt_inv               0
                            ...
acc_now_delinq               29
chargeoff_within_12_mths    145
delinq_amnt                  29
pub_rec_bankruptcies       1365
tax_liens                   105
Length: 61, dtype: int64
```

```
 Total number of missing values NaN in the DataFrame :

 121640

 Total number of columns with missing values :

 22

 Total number of records with missing values :

 42421
```

```
loanData.int_rate.isna().sum()
```

```
0
```

ID

```
loanData.id.isna().sum()
```

```
0
```

Member id

```
loanData.member_id.isna().sum()
```

```
0
```

Home Ownership

```
loanData.home_ownership.dtypes
```

```
dtype('O')
```

```
loanData.home_ownership.isna().sum()
```

```
0
```

Annual Income

```
loanData.annual_inc.dtypes
```

```
dtype('float64')
```

```
loanData.annual_inc.isna().sum()
```

```
4
```

Verification Status

```
loanData.verification_status.dtypes
```

```
dtype('O')
```

```
loanData.verification_status.isna().sum()
```

```
0
```

Issued Date

```
loanData.issue_d.dtypes
```

```
dtype('O')
```

```
loanData.rename(columns={'issue_d': 'issue_date'}, inplace=True)
```

```
from datetime import datetime

for i in range(len(loanData['issue_date'])):
  date_time_str = loanData['issue_date'][i]
  #date_time_converted = datetime.strptime(date_time_str, '%m-
%b').strftime('%m-%Y')
  #date_time_converted = datetime.strptime(date_time_str, '%m-/%y')
  #loanData['issue_date'][i]= date_time_converted
  #date_time_obj = datetime.datetime.strptime(date_time_str, '%b %d
%Y')
print(loanData['issue_date'])
```

```
0        Dec-11
1        Dec-11
2        Dec-11
3        Dec-11
4        Dec-11
         ...
42530    Jun-07
42531    Jun-07
42532    Jun-07
42533    Jun-07
42534    Jun-07
Name: issue_date, Length: 42535, dtype: object
```

```
loanData['issue_date']
```

```
0        Dec-11
1        Dec-11
2        Dec-11
3        Dec-11
4        Dec-11
          ...
42530    Jun-07
42531    Jun-07
42532    Jun-07
42533    Jun-07
42534    Jun-07
Name: issue_date, Length: 42535, dtype: object
```

Loan Status

```
loanData.loan_status.dtypes
```

```
dtype('O')
```

```
loanData.loan_status.isna().sum()
```

```
0
```

Payment Plan - Removed

```
loanData.rename(columns={'pymnt_plan': 'payment_plan'}, inplace=True)
```

```
loanData.payment_plan.dtypes
```

```
dtype('O')
```

```
loanData.payment_plan.isna().sum()
```

```
0
```

```
loanData["payment_plan"].unique()
```

```
array(['n', 'y'], dtype=object)
```

```
loanData = loanData.drop('payment_plan', axis=1)
```

URL & desc - Removed

Purpose

```
loanData["purpose"].unique()
```

```
array(['credit_card', 'car', 'small_business', 'other', 'wedding',
       'debt_consolidation', 'home_improvement', 'major_purchase',
       'medical', 'moving', 'vacation', 'house', 'renewable_energy',
       'educational'], dtype=object)
```

```
loanData.purpose.isna().sum()
```

```
0
```

Title

```
loanData.title.isna().sum()
```

```
13
```

```
loanData["title"].unique()
```

```
array(['Computer', 'bike', 'real estate business', ..., 'delight',
       'Car repair bill', 'Aroundthehouse'], dtype=object)
```

COLUMNS AF to AO

Revolving Balance

```
loanData.rename(columns={'revol_bal': 'revolve_balance'},
inplace=True)
```

```
loanData['revolve_balance'].describe
```

```
<bound method NDFrame.describe of 0        13648
1         1687
2         2956
3         5598
4        27783
         ...
42530        0
42531        0
42532        0
42533        0
42534        0
Name: revolve_balance, Length: 42535, dtype: int64>
```

```
loanData.revolve_balance.isna().sum()
```

```
0
```

Total Account

```
loanData.rename(columns={'total_acc': 'total_account'}, inplace=True)
```

```
loanData.total_account.isna().sum()
```

```
29
```

Initial List Status

```
loanData['initial_list_status'].describe
```

```
<bound method NDFrame.describe of 0        f
1        f
2        f
```

```
3        f
4        f
        ..
42530    f
42531    f
42532    f
42533    f
42534    f
Name: initial_list_status, Length: 42535, dtype: object>
```

```
loanData.initial_list_status.isna().sum()
```

```
0
```

Outstanding Principal

```
loanData.rename(columns={'out_prncp': 'outstanding_principal'},
inplace=True)
```

```
loanData['outstanding_principal'].describe
```

```
<bound method NDFrame.describe of 0          0.00
1          0.00
2          0.00
3          0.00
4        270.78
          ...
42530      0.00
42531      0.00
42532      0.00
42533      0.00
42534      0.00
Name: outstanding_principal, Length: 42535, dtype: float64>
```

```
loanData.outstanding_principal.isna().sum()
```

```
0
```

Outstanding Principal by Investors

```
loanData.rename(columns={'out_prncp_inv':
'outstanding_principal_investors'}, inplace=True)
```

```
loanData['outstanding_principal_investors'].describe
```

```
<bound method NDFrame.describe of 0          0.00
1          0.00
2          0.00
3          0.00
4        270.78
          ...
42530      0.00
42531      0.00
```

```
42532      0.00
42533      0.00
42534      0.00
Name: outstanding_principal_investors, Length: 42535, dtype: float64>
```

```
loanData.outstanding_principal_investors.isna().sum()
```

```
0
```

Total Payment

```
loanData.rename(columns={'total_pymnt': 'total_payment'},
inplace=True)
```

```
loanData['total_payment'].describe
```

```
<bound method NDFrame.describe of 0         5863.155187
1         1008.710000
2         3005.666844
3        12231.890000
4         3784.490000
             ...
42530     3719.431070
42531     1155.600899
42532     2904.498829
42533     7373.904962
42534     5619.762090
Name: total_payment, Length: 42535, dtype: float64>
```

```
loanData.total_payment.isna().sum()
```

```
0
```

Total Payment by Investors

```
loanData.rename(columns={'total_pymnt_inv':
'total_payment_investors'}, inplace=True)
```

```
loanData['total_payment_investors'].describe
```

```
<bound method NDFrame.describe of 0         5833.84
1         1008.71
2         3005.67
3        12231.89
4         3784.49
             ...
42530      239.11
42531        0.00
42532      258.82
42533        0.00
42534        0.00
Name: total_payment_investors, Length: 42535, dtype: float64>
```

```
loanData.total_payment_investors.isna().sum()
```

```
0
```

Total Principal Recieved

```
loanData.rename(columns={'total_rec_prncp':
'total_principal_recieved'}, inplace=True)
```

```
loanData['total_principal_recieved'].describe
```

```
<bound method NDFrame.describe of 0         5000.00
1          456.46
2         2400.00
3        10000.00
4         2729.22
           ...
42530     3500.00
42531     1000.00
42532     2525.00
42533     6500.00
42534     5000.00
Name: total_principal_recieved, Length: 42535, dtype: float64>
```

```
loanData.total_principal_recieved.isna().sum()
```

```
0
```

Total Interest Recieved

```
loanData.rename(columns={'total_rec_int': 'total_interest_recieved'},
inplace=True)
```

```
loanData['total_interest_recieved'].describe
```

```
<bound method NDFrame.describe of 0         863.16
1         435.17
2         605.67
3        2214.92
4        1055.27
           ...
42530     219.43
42531     155.60
42532     379.50
42533     873.90
42534     619.76
Name: total_interest_recieved, Length: 42535, dtype: float64>
```

```
loanData.total_interest_recieved.isna().sum()
```

```
0
```

Columns AZ to BI

Application Type ⛔

```
loanData['application_type'].dtypes
```

```
dtype('O')
```

```
#loanData.application_type = loanData.application_type.astype('|S')
```

```
loanData.application_type.dtypes
```

```
dtype('O')
```

```
loanData.shape
```

```
(42535, 60)
```

```
loanData = loanData.iloc[: , :-5]
```

```
from google.colab import files
loanData.to_csv('outputLoanInfo.csv')
files.download('outputLoanInfo.csv')
```

```
<IPython.core.display.Javascript object>
```

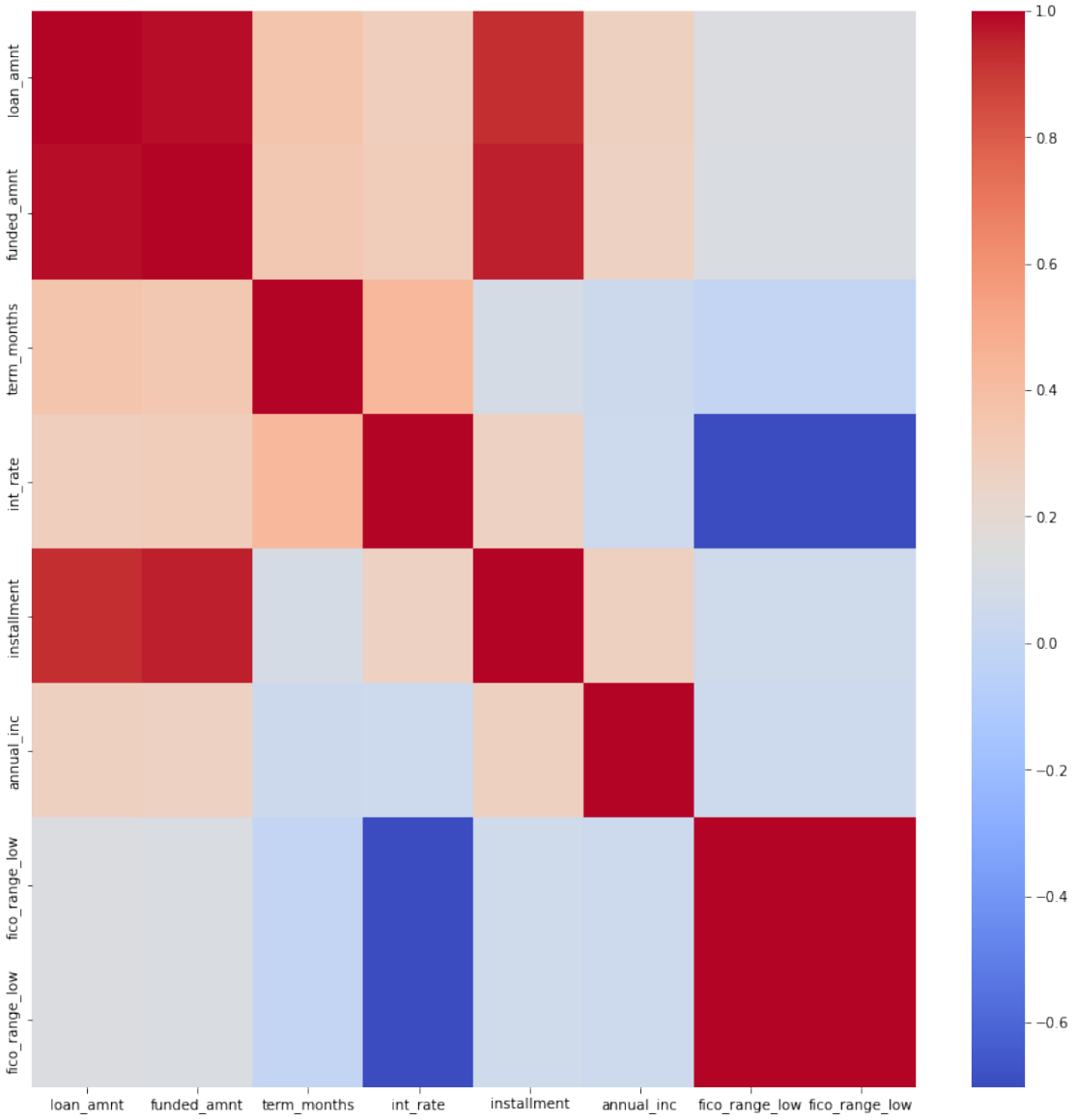
```
<IPython.core.display.Javascript object>
```

```
# Set correlation variable
corrLoanData =
loanData[['loan_amnt','funded_amnt','term_months','int_rate','installm
ent','annual_inc',"fico_range_low","fico_range_low","emp_length","revo
l_util"]]
```

```
corr = corrLoanData.corr()
```

```
# Plot the heatmap
corrplt=plt.figure(figsize=(14,14))
sns.heatmap(corr,
        xticklabels=corr.columns,
        yticklabels=corr.columns,
        cmap='coolwarm')
```

```
corrplt.savefig("correlation.pdf", bbox_inches='tight')
```

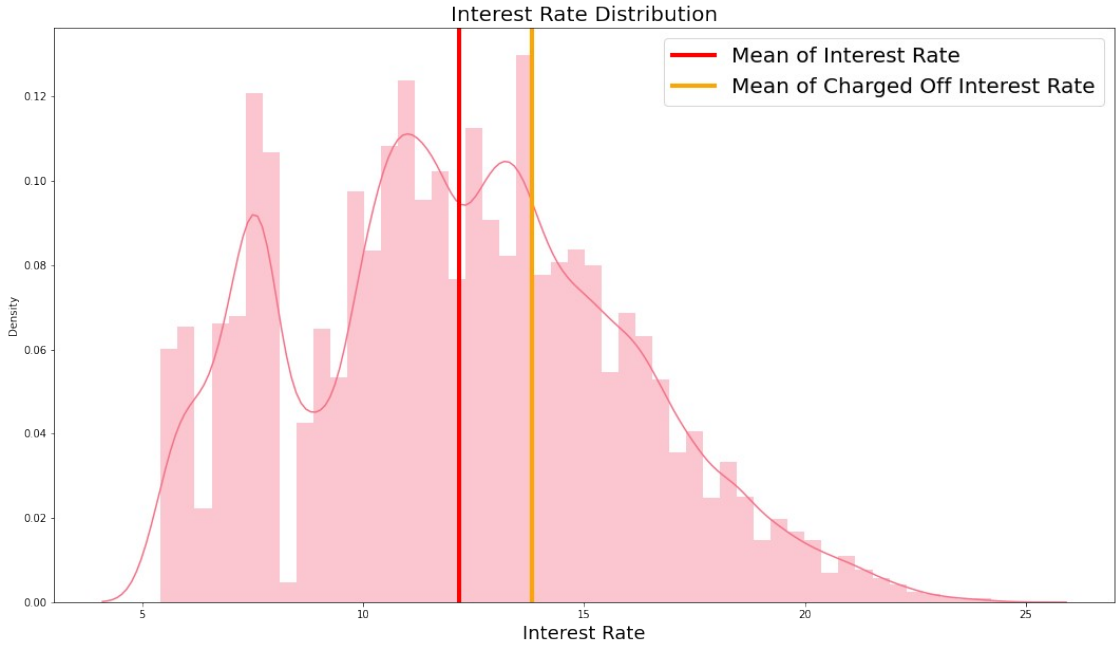



```
# Set new variable name to include only loans that are defaulted
loanDataChargedOff = loanData[loanData.loan_status == 'Charged Off']

# Distribution of interest rates
sns.set_palette("husl")
f=plt.figure(figsize=(18,10))
sns.distplot(loanData['int_rate'], hist='density')
plt.axvline(x=loanData.int_rate.mean(), color='red', linestyle='-',
lw=4, label='Mean of Interest Rate')
plt.axvline(x=loanDataChargedOff.int_rate.mean(), color='orange',
linestyle='-', lw=4, label='Mean of Charged Off Interest Rate')
plt.title('Interest Rate Distribution', fontsize=20)
plt.xlabel('Interest Rate', fontsize=18)
plt.legend(fontsize=20)
```

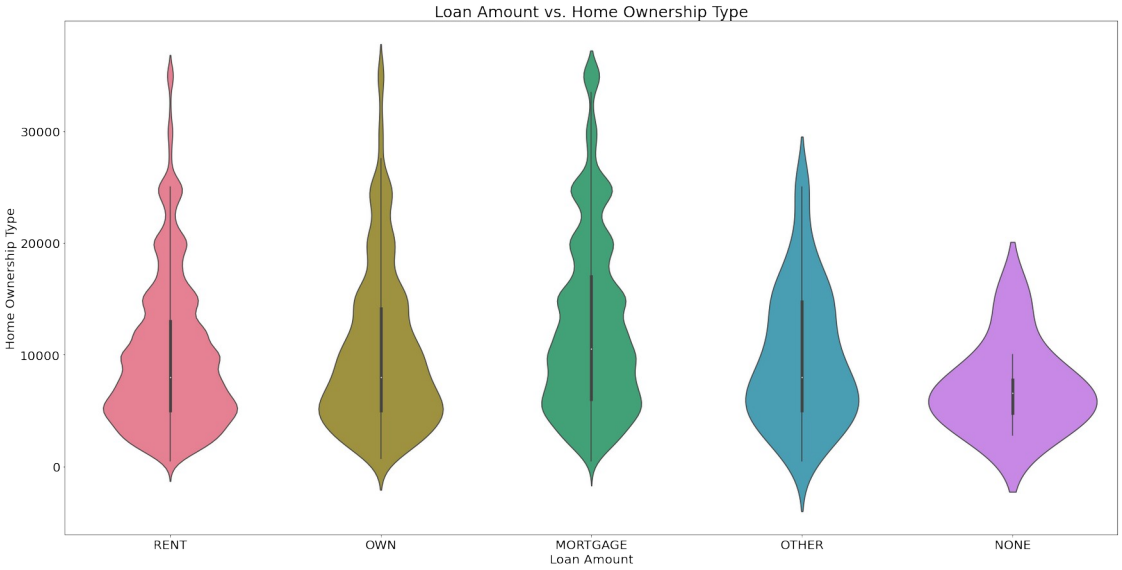
```
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
#f.savefig("int_rate_distribution.pdf", bbox_inches='tight')

# Loan Amount vs. Home Ownership Type
plt.figure(figsize=(30,15))
sns.violinplot(x="home_ownership", y="loan_amnt", data=loanData,
palette="husl")
plt.title('Loan Amount vs. Home Ownership Type', fontsize=25)
plt.xlabel('Loan Amount', fontsize=20)
plt.ylabel('Home Ownership Type', fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```




```
# Debt to income ratio vs Load grade
plt.figure(figsize=(30,12))
sns.boxplot(x="grade", y="dti", data=loanData, palette="husl")
plt.title('Box Plot of Interest Rate vs. Loan Grade', fontsize=25)
plt.xlabel('Installment', fontsize=15)
plt.ylabel('Loan Grade', fontsize=15)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```