Initial Steps and Data Analysis

1.Loaded csv file using read csv method and then viewed file content using head() method

```
In [3]: #Displaying the content of the loan file.

        import pandas as pd
        LoanData = pd.read_csv("C:/Users/bhavy/OneDrive/Desktop/loan.csv")
        pd.set_option('display.max_columns', None)
        LoanData.head(2)
```

```
C:\Users\bhavy\AppData\Local\Temp\ipykernel_12888\2002972083.py:4: DtypeWarning: Columns (47) have mixed types. Specify dtype o
ption on import or set low_memory=False.
  LoanData = pd.read_csv("C:/Users/bhavy/OneDrive/Desktop/loan.csv")
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	a
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	NaN	10+ years	RENT	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	Ryder	< 1 year	RENT	

2. Displayed all the columns in the data set file.

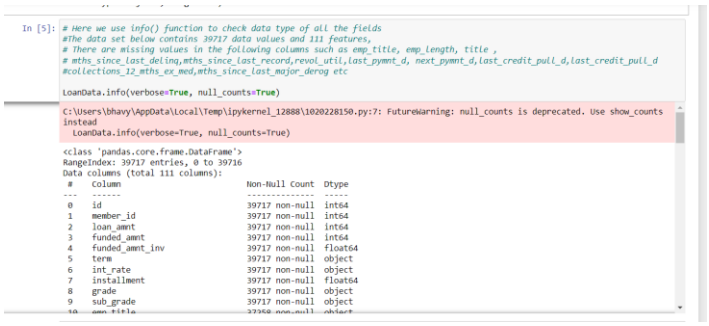
```
In [4]: # Displaying all the columns in the csv file, As displayed below,
        LoanData.columns
```

```
Out[4]: Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
               'term', 'int_rate', 'installment', 'grade', 'sub_grade',
               ...
               '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'],
              dtype='object', length=111)
```

Data Cleaning:

1.To understand the column data types, we are using info() function:

Here in the data set ,we have empty columns which are not useful for EDA analysis hence can be removed:



```
In [5]: # Here we use info() function to check data type of all the fields
        #The data set below contains 39717 data values and 111 features,
        # There are missing values in the following columns such as emp_title, emp_length, title ,
        # mths_since_last_deling,mths_since_last_record,revol_util,last_pymnt_d, next_pymnt_d,last_credit_pull_d,last_credit_pull_d
        #collections_12_mths_ex_med,mths_since_last_major_derog etc

        LoanData.info(verbose=True, null_counts=True)
```

```
C:\Users\bhavy\AppData\Local\Temp\ipykernel_12888\1020228150.py:7: FutureWarning: null_counts is deprecated. Use show_counts
instead
  LoanData.info(verbose=True, null_counts=True)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 111 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          39717 non-null  int64
 1   member_id                   39717 non-null  int64
 2   loan_amnt                   39717 non-null  int64
 3   funded_amnt                 39717 non-null  int64
 4   funded_amnt_inv             39717 non-null  float64
 5   term                        39717 non-null  object
 6   int_rate                    39717 non-null  object
 7   installment                 39717 non-null  float64
 8   grade                       39717 non-null  object
 9   sub_grade                   39717 non-null  object
```

2. Here we are using drop function to drop all the columns where there are 0 values:

```
In [6]: #Here in the data set, few of the columns has NaN values which are 100%, we need to drop those columns.


LoanData.drop(["verification_status_joint", "tot_coll_amt", "tot_cur_bal"], axis = 1, inplace = True)
LoanData.drop(["open_acc_6m", "open_il_6m", "open_il_12m", "open_il_24m", "mths_since_rcnt_il", "total_bal_il", "il_util", "open_
LoanData.drop(["open_rv_24m", "max_bal_bc" ,"all_util"  ,"total_rev_hi_lim","acc_open_past_24mths"], axis = 1, inplace = True)
LoanData.drop(["avg_cur_bal","bc_open_to_buy" ,"bc_util", "mo_sin_old_il_acct" ,"mo_sin_old_rev_tl_op", "mo_sin_rcnt_rev_tl_op","
LoanData.drop(["mths_since_recent_bc" ,"mths_since_recent_bc_dlq", "mths_since_recent_inq", "mths_since_recent_revol_delinq" ,"nu
LoanData.drop(["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
LoanData.drop([ "num_sats","num_tl_120dpd_2m","num_tl_30dpd","num_tl_90g_dpd_24m","num_tl_op_past_12m","pct_tl_nvr_dlq" ,"percent
LoanData.drop(["total_bal_ex_mort", "total_bc_limit","total_il_high_credit_limit"], axis = 1, inplace = True)
LoanData.drop(["inq_fi", "inq_last_12m"], axis = 1, inplace = True)
LoanData.drop(["mths_since_last_major_derog", "annual_inc_joint","dti_joint", "total_cu_tl"], axis = 1, inplace = True)
```

3. Here we are again using info() and shape() functions to display all the columns data to make sure the empty columns are deleted:

```
In [8]: #Rows and column counts
        LoanData.shape

Out[8]: (39717, 57)
```

```
In [7]: LoanData.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 57 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          39717 non-null  int64
 1   member_id                   39717 non-null  int64
 2   loan_amnt                   39717 non-null  int64
 3   funded_amnt                 39717 non-null  int64
 4   funded_amnt_inv             39717 non-null  float64
 5   term                        39717 non-null  object
 6   int_rate                    39717 non-null  object
 7   installment                 39717 non-null  float64
 8   grade                       39717 non-null  object
 9   sub_grade                   39717 non-null  object
 10  emp_title                   37258 non-null  object
 11  emp_length                  38642 non-null  object
 12  home_ownership              39717 non-null  object
 13  annual_inc                  39717 non-null  float64
 14  verification_status         39717 non-null  object
 15  issue_d                     39717 non-null  object
 16  loan_status                 39717 non-null  object
 17  pymnt_plan                  39717 non-null  object
 18  url                         39717 non-null  object
 19  desc                        26777 non-null  object
 20  purpose                     39717 non-null  object
 21  title                       39706 non-null  object
```

4. We are then checking missing values in each of the columns:

```
In [9]: # Here we are checking missing values in each column as shown in following block of code
        import pandas as pd
        #LoanData = pd.read_csv("C:/Users/bhavy/OneDrive/Desktop/loan.csv")
        Missing_data = pd.DataFrame({'total_missing': LoanData.isnull().sum(),'percentage_missing':(LoanData.isnull().sum()/39717)*100})
        Missing_data
```

Out[9]:

	total_missing	percentage_missing
id	0	0.000000
member_id	0	0.000000
loan_amnt	0	0.000000
funded_amnt	0	0.000000
funded_amnt_inv	0	0.000000
term	0	0.000000
int_rate	0	0.000000
installment	0	0.000000
grade	0	0.000000
sub_grade	0	0.000000
emp_title	2459	6.191303
emp_length	1075	2.706650

Univariate Analysis:

5. Here we are performing analysis of each of the columns through univariate analysis using function describe() function which provides information about mean, median, mode and percentile data of all the columns.

```
LoanData.describe()
```

Out[10]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	dti	delinq_2yrs	inq_last_6mths	mtl
count	3.971700e+04	3.971700e+04	39717.000000	39717.000000	39717.000000	39717.000000	3.971700e+04	39717.000000	39717.000000	39717.000000	
mean	6.831319e+05	8.504636e+05	11219.443815	10947.713196	10397.448868	324.561922	6.896893e+04	13.315130	0.146512	0.869200	
std	2.106941e+05	2.656783e+05	7456.670694	7187.238670	7128.450439	208.874874	6.379377e+04	6.678594	0.491812	1.070219	
min	5.473400e+04	7.069900e+04	500.000000	500.000000	0.000000	15.690000	4.000000e+03	0.000000	0.000000	0.000000	
25%	5.162210e+05	6.667800e+05	5500.000000	5400.000000	5000.000000	167.020000	4.040400e+04	8.170000	0.000000	0.000000	
50%	6.656650e+05	8.508120e+05	10000.000000	9600.000000	8975.000000	280.220000	5.900000e+04	13.400000	0.000000	1.000000	
75%	8.377550e+05	1.047339e+06	15000.000000	15000.000000	14400.000000	430.780000	8.230000e+04	18.600000	0.000000	1.000000	
max	1.077501e+06	1.314167e+06	35000.000000	35000.000000	35000.000000	1305.190000	6.000000e+06	29.990000	11.000000	8.000000	

Bivariate Analysis:

6. Here we are performing analysis of each of the columns through Bivariate analysis using function corr() function which provides which compares each of the columns.

```
In [11]: LoanData.corr()
```

Out[11]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	dti	delinq_2yrs	inq_last_6mths
id	1.000000	0.993650	0.141919	0.152286	0.249547	0.086587	0.008731	0.095983	-0.008644	-0.042378
member_id	0.993650	1.000000	0.140710	0.150322	0.257887	0.081025	0.009380	0.096963	-0.008119	-0.047086
loan_amnt	0.141919	0.140710	1.000000	0.981578	0.940034	0.930288	0.271149	0.066439	-0.031864	0.009229
funded_amnt	0.152286	0.150322	0.981578	1.000000	0.958422	0.956159	0.266965	0.066283	-0.032355	0.009259
funded_amnt_inv	0.249547	0.257887	0.940034	0.958422	1.000000	0.905039	0.254375	0.074689	-0.038501	-0.005712
installment	0.086587	0.081025	0.930288	0.956159	0.905039	1.000000	0.270874	0.054186	-0.019657	0.009722
annual_inc	0.008731	0.009380	0.271149	0.266965	0.254375	0.270874	1.000000	-0.122732	0.023083	0.033908
dti	0.095983	0.096963	0.066439	0.066283	0.074689	0.054186	-0.122732	1.000000	-0.034452	0.001405
delinq_2yrs	-0.008644	-0.008119	-0.031864	-0.032355	-0.038501	-0.019657	0.023083	-0.034452	1.000000	0.008091
inq_last_6mths	-0.042378	-0.047086	0.009229	0.009259	-0.005712	0.009722	0.033908	0.001405	0.008091	1.000000
mths_since_last_delinq	0.117193	0.122963	0.014871	0.016359	0.071924	0.000047	-0.008822	0.068815	-0.569438	-0.003851
mths_since_last_record	0.702130	0.719538	0.004968	-0.006059	0.437906	-0.062171	-0.031866	0.187175	-0.036628	-0.027878
open_acc	0.020388	0.017928	0.177168	0.175530	0.163027	0.172812	0.158200	0.288045	0.011656	0.091713
pub_rec	-0.019440	-0.018721	-0.051236	-0.052169	-0.053214	-0.046532	-0.018689	-0.004621	0.007463	0.024802
revol_bal	0.008763	0.005113	0.317597	0.310392	0.290797	0.312679	0.279961	0.228743	-0.055125	-0.022381
total_acc	0.042149	0.044374	0.256442	0.250589	0.242854	0.230824	0.235771	0.229881	0.067892	0.111499
out_prncp	0.180619	0.170743	0.192937	0.194941	0.203688	0.125082	0.033573	0.036095	-0.003008	-0.012106
out_prncp_inv	0.180591	0.170706	0.192623	0.194675	0.203693	0.124932	0.033472	0.036012	-0.003203	-0.011780
total_pymnt	0.140045	0.139975	0.886613	0.903160	0.881228	0.856928	0.257980	0.064766	-0.022695	-0.010559
total_pymnt_inv	0.220838	0.229307	0.854243	0.870799	0.913257	0.817416	0.247119	0.071647	-0.028976	-0.020277

7. Then we perform analysis on categorial columns to understand categories of data through value count() function and then we try to fill empty values through mean data using fillna function.

```
In [12]: LoanData['initial_list_status'].value_counts()
```

Out[12]:
```
f    39717
Name: initial_list_status, dtype: int64
```

```
In [13]: LoanData.head(2)
```

Out[13]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	a
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	NaN	10+ years	RENT	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	Ryder	< 1 year	RENT	

```
In [14]: LoanData.fillna(LoanData.mean(numeric_only=True).round(1), inplace=True)
```

8. We then perform univariate analysis by converting string type of the columns to number type(int/float) as below:

```
In [40]: # Here converting the term column value to integer value to perform univariate analysis

         LoanData['term']=LoanData['term'].str.extract('(\d+)').astype(int)
```

```
In [70]: LoanData.head(3)
```

Out[70]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	anr
0	1077501	1296599	5000	5000	4975.0	36	10.65	162.87	2	2.2	NaN	10	1	
1	1077430	1314167	2500	2500	2500.0	60	15.27	59.83	3	3.4	Ryder	1	1	
2	1077175	1313524	2400	2400	2400.0	36	15.96	84.33	3	3.5	NaN	10	1	

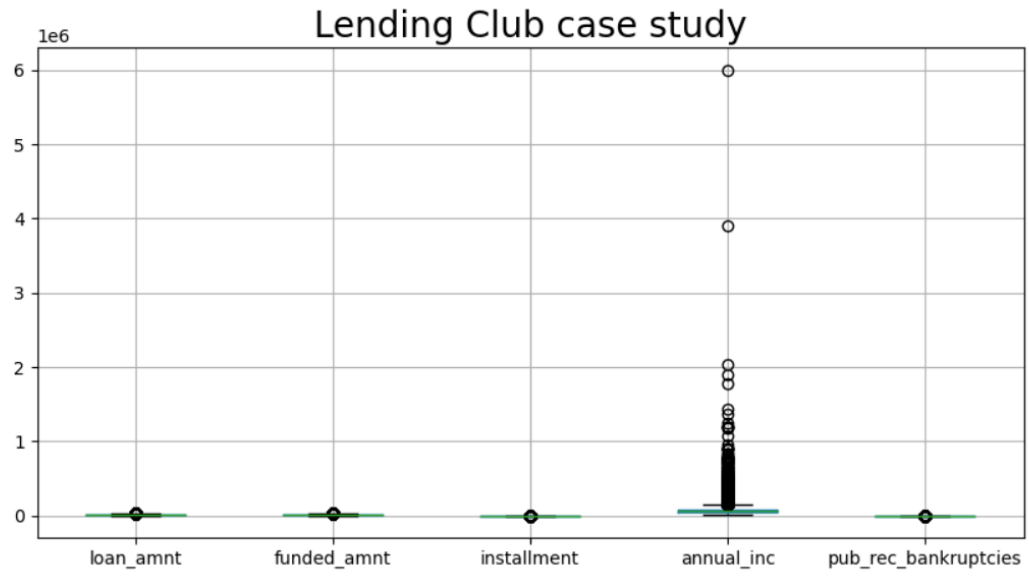
```
In [16]: # Here converting the grade column Categorial value to integer value to perform univariate analysis
         LoanData['grade'].replace(['A','B','C','D','E','F','G'],
         [1,2,3,4,5,6,7], inplace=True)
```

Note- for other remaining columns please refer to python file.

9. To identify outliner we are using following graph where we can see there is an outliner value identified in “annual_inc” column:

```
import matplotlib.pyplot as plt

num_cols = ['loan_amnt', 'funded_amnt', 'installment','annual_inc','pub_rec_bankruptcies']
plt.figure(figsize=(10,5))
LoanData[num_cols].boxplot()
plt.title("Lending Club case study", fontsize=20)
plt.show()
```



10. Also to display information about defaulters we are using following graph:

In [150]:

```
from matplotlib import pyplot as plt

LoanData['loan_status'].value_counts().head(10).plot.bar()

plt.title('Defaulter in the loan re-payment')
plt.xlabel('Categories', fontsize=15)
plt.ylabel('Defaulter', fontsize=15)
plt.show()
```