Great Learning
POWER AHEAD

Mini-Project [ITP+NPV]

TITLE : Loan Dataset Analysis

Submitted by : Muzammil Firdous M A
Manjunath I Kunte
Munir Sheth
Nandish S Pattanashetti
Nandakumar

Problem Definition

- Dataset: Loan Dataset

- Problem Statement :

The company needs information about the previous loans that were made and the status of the customers who received those loans in order to decide whether or not to make another loan to that customer in the future and how much loan money should be given to that person. The company also needs information about the current loans (from 2007 to 2011).

Data Set Description

The dataset includes all available loan information for loans granted from 2007 to 2011.

Data Dictionary -

1.annual_inc - The self-reported annual income provided by the borrower during registration.

2.dti - A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.

3.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.

4.funded_amnt - The total amount committed to that loan at that point in time.

5.funded_amnt_inv -The total amount committed by investors for that loan at that point in time.

6.grade - LC assigned loan grade

7.id - A unique LC assigned ID for the loan listing.

8.installment - The monthly payment owed by the borrower if the loan originates.

9.int_rate - Interest Rate on the loan

10.last_pymnt_amnt-Last total payment amount received

11.last_pymnt_d -Last month payment was received

12.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 value.

13.loan_status - Current status of the loan

14.member_id -A unique LC assigned Id for the borrower member.

15.purpose - A category provided by the borrower for the loan request.

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

17.total_acc -The total number of credit lines currently in the borrower's credit file

18.total_pymnt -Payments received to date for total amount funded

19.total_pymnt_inv -Payments received to date for portion of total amount funded by investors

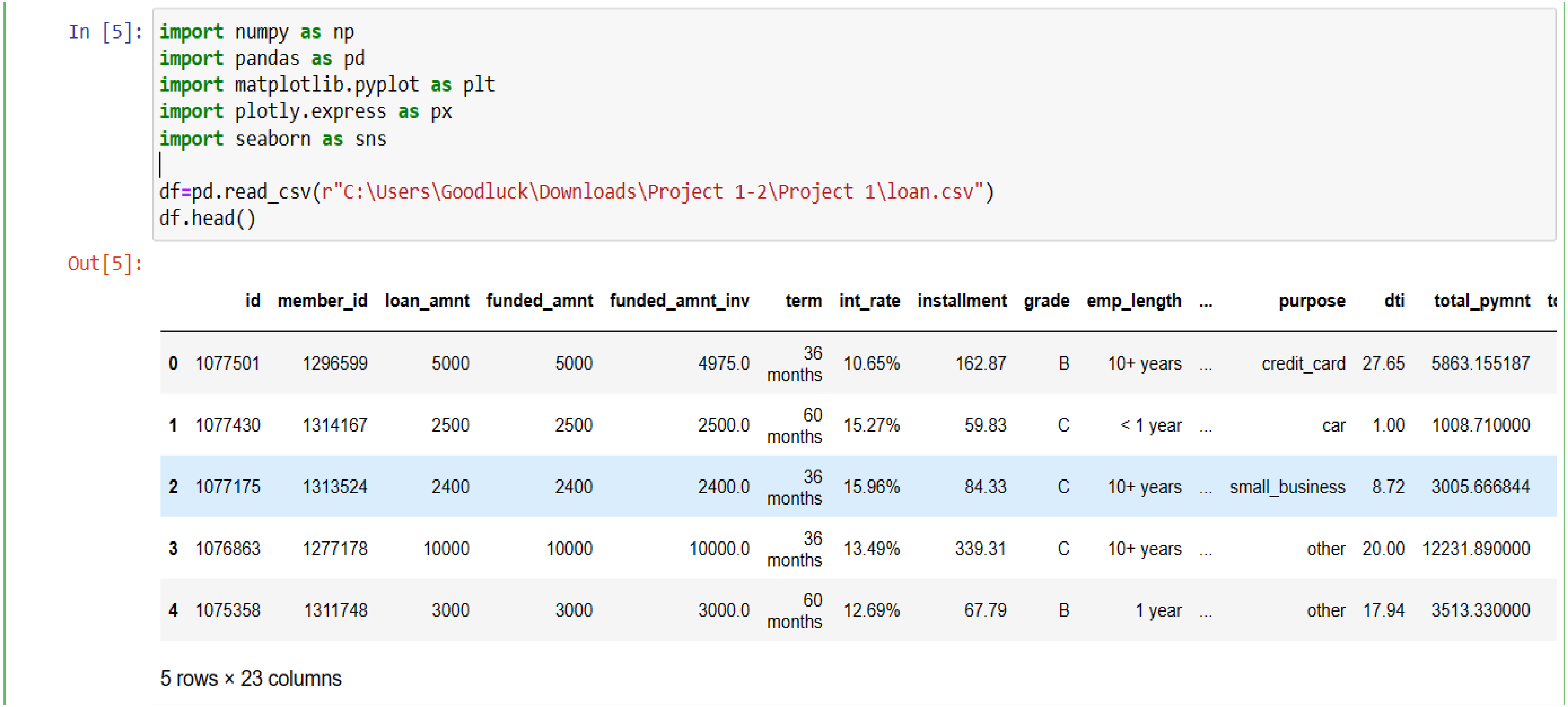
20.total_rec_int -Interest received to date

Business Importance of Problem

Loan analysis helps in ensuring that loans are given to the customers who can and will repay them on time and with the proper terms. In determining the type of analysis needed and the most effective way to meet the need, it is important to consider the nature of the loan, as well as if the company intends to deny the loan due to unresolved past loan accounts.

Question 1

Import the dataset and understand it.



In [5]:

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

df=pd.read_csv(r"C:\Users\Goodluck\Downloads\Project 1-2\Project 1\loan.csv")
df.head()
```

Out[5]:

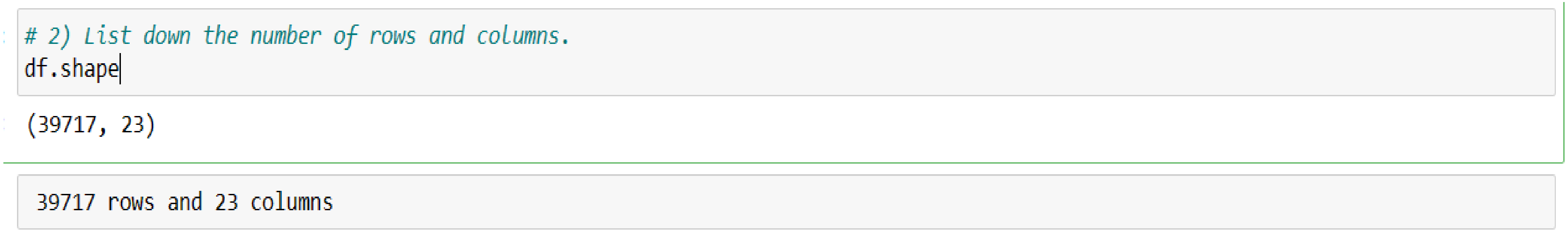
	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	emp_length	...	purpose	dti	total_pymnt	t
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	10+ years	...	credit_card	27.65	5863.155187	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	< 1 year	...	car	1.00	1008.710000	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	10+ years	...	small_business	8.72	3005.666844	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	10+ years	...	other	20.00	12231.890000	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	1 year	...	other	17.94	3513.330000	

5 rows × 23 columns

Question 2

List down the number of rows and columns.

#Finding the number of columns and rows in the given dataset.



```
# 2) List down the number of rows and columns.
df.shape
```

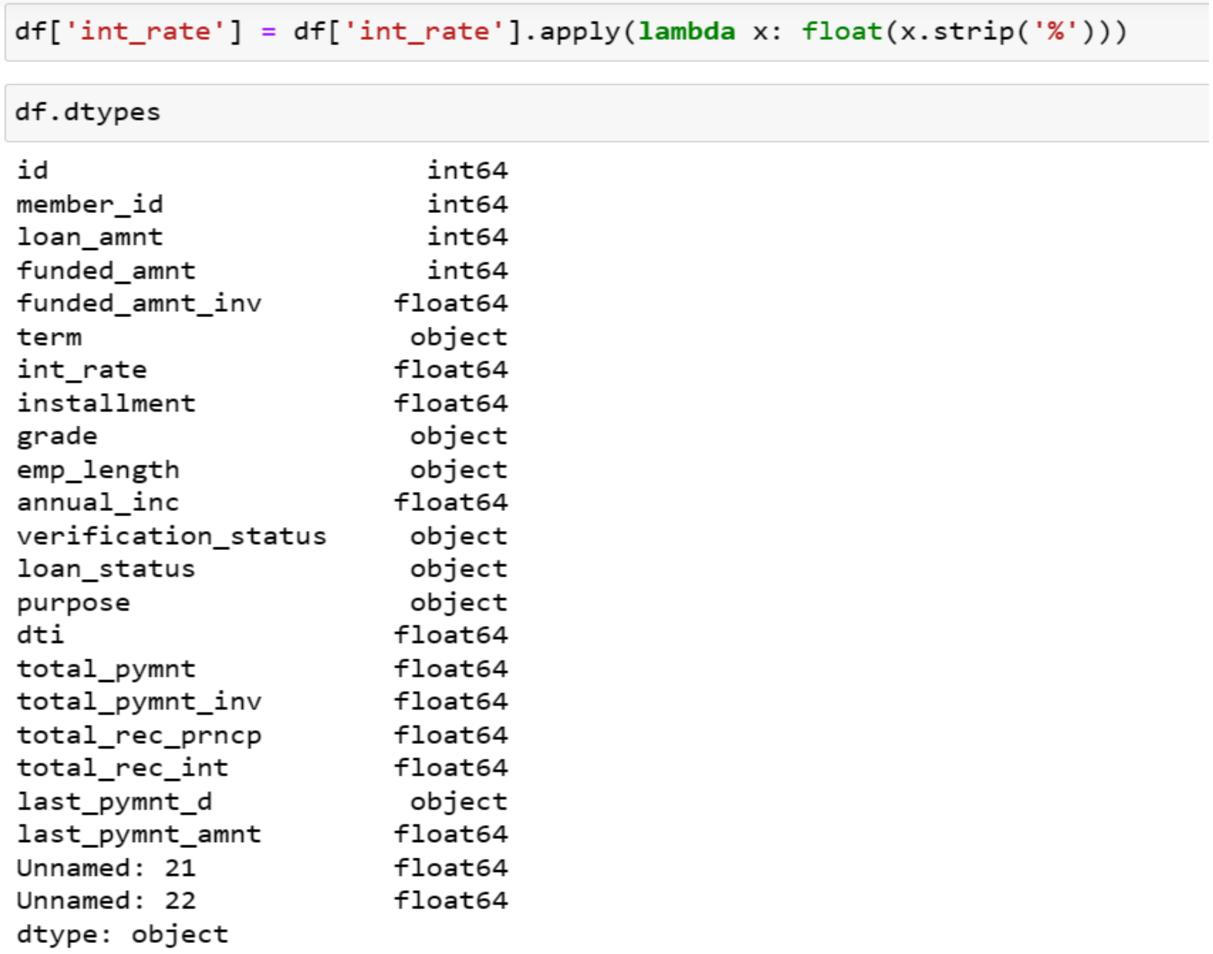
```
(39717, 23)
```

```
39717 rows and 23 columns
```

Question 3

‘Int_rate’ column is character type. With the help of lambda function float type.

#Convert the datatype of the column ‘Int_rate’ from character to float type using lambda function.



```
df['int_rate'] = df['int_rate'].apply(lambda x: float(x.strip('%')))
```

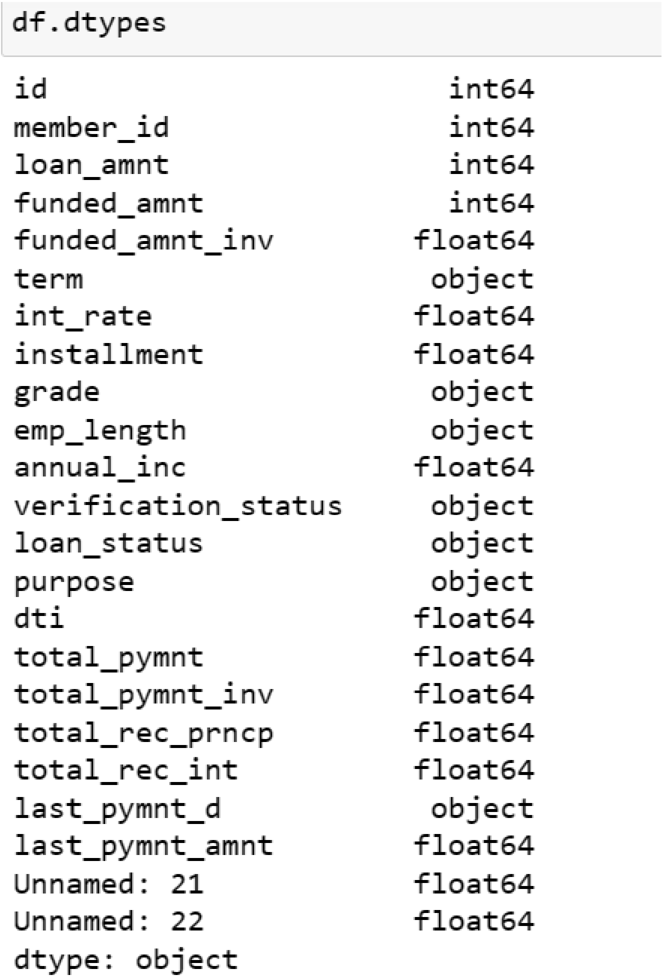
```
df.dtypes
```

```
id                       int64
member_id                int64
loan_amnt                int64
funded_amnt              int64
funded_amnt_inv        float64
term                    object
int_rate               float64
installment            float64
grade                   object
emp_length              object
annual_inc             float64
verification_status     object
loan_status             object
purpose                 object
dti                    float64
total_pymnt            float64
total_pymnt_inv        float64
total_rec_prncp        float64
total_rec_int          float64
last_pymnt_d            object
last_pymnt_amnt        float64
Unnamed: 21            float64
Unnamed: 22            float64
dtype: object
```

Question 4

Check the datatypes of each column.

#Display the datatype of each columns of the dataset.



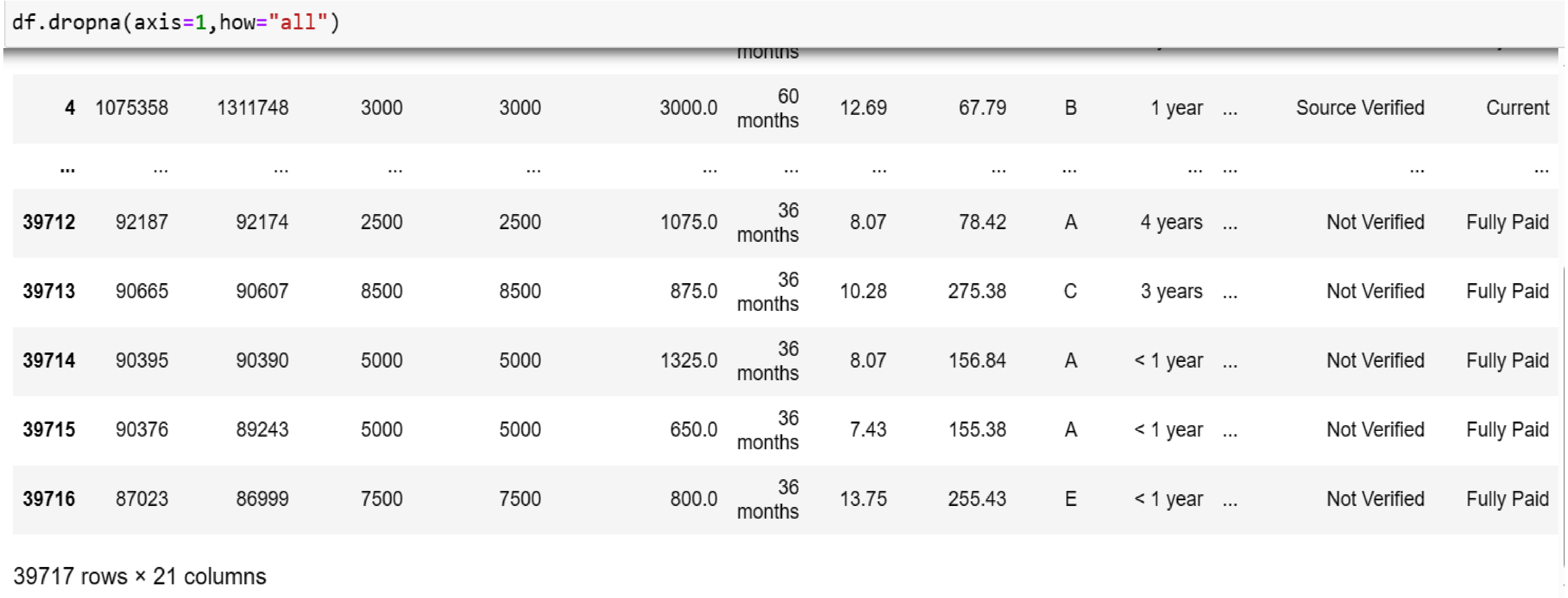
```
df.dtypes
```

```
id                       int64
member_id                int64
loan_amnt                int64
funded_amnt              int64
funded_amnt_inv        float64
term                    object
int_rate               float64
installment            float64
grade                   object
emp_length              object
annual_inc             float64
verification_status     object
loan_status             object
purpose                 object
dti                    float64
total_pymnt            float64
total_pymnt_inv        float64
total_rec_prncp        float64
total_rec_int          float64
last_pymnt_d            object
last_pymnt_amnt        float64
Unnamed: 21            float64
Unnamed: 22            float64
dtype: object
```

Question 5

Cleaning the dataset- Remove the columns having complete NaN value in the entire dataset.

#Drop the columns having all it’s rows as NaN.



```
df.dropna(axis=1,how="all")
```

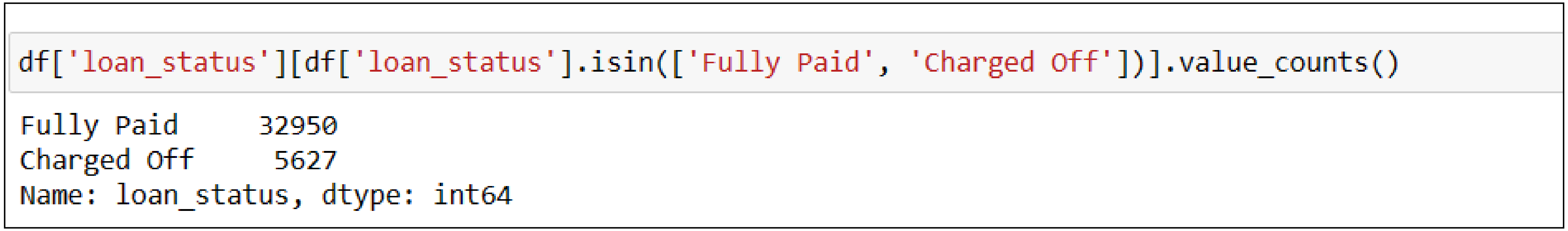
						months							
4	1075358	1311748	3000	3000	3000.0	60 months	12.69	67.79	B	1 year	...	Source Verified	Current
...	...	...	...	...	...	...	...	...	...	...	...	...	...
39712	92187	92174	2500	2500	1075.0	36 months	8.07	78.42	A	4 years	...	Not Verified	Fully Paid
39713	90665	90607	8500	8500	875.0	36 months	10.28	275.38	C	3 years	...	Not Verified	Fully Paid
39714	90395	90390	5000	5000	1325.0	36 months	8.07	156.84	A	< 1 year	...	Not Verified	Fully Paid
39715	90376	89243	5000	5000	650.0	36 months	7.43	155.38	A	< 1 year	...	Not Verified	Fully Paid
39716	87023	86999	7500	7500	800.0	36 months	13.75	255.43	E	< 1 year	...	Not Verified	Fully Paid

39717 rows × 21 columns

Question 6

Write the code to find the value counts of the ‘loan_status’
category column and filter only the ‘fully paid’ and ‘charged off’ categories.

#Obtain the number of occurrence of ‘fully paid’ and ‘charged off’ in the ‘loan_status’ column.



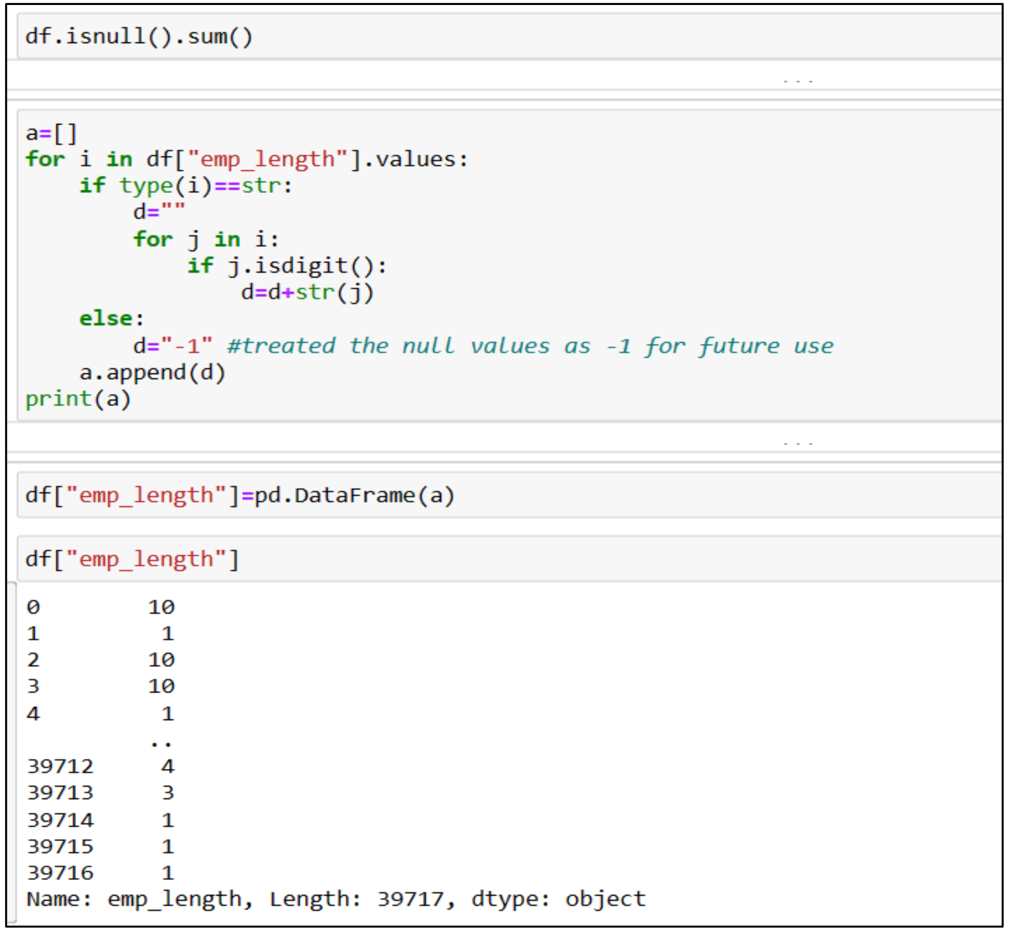
```
df['loan_status'][df['loan_status'].isin(['Fully Paid', 'Charged Off'])].value_counts()
```

```
Fully Paid     32950
Charged Off     5627
Name: loan_status, dtype: int64
```

Question 7

Filter the ‘Emp_Len’ column to extract the numerical value from the string.

#Write a code to extract only the numerical value from the column ‘Emp_len’ like < 1year, 2 years , 3 years as 1 , 2, 3 so on.



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

```
a=[]
for i in df["emp_length"].values:
    if type(i)==str:
        d=""
        for j in i:
            if j.isdigit():
                d=d+str(j)
    else:
        d="-1" #treated the null values as -1 for future use
    a.append(d)
print(a)
```

```
df["emp_length"]=pd.DataFrame(a)
```

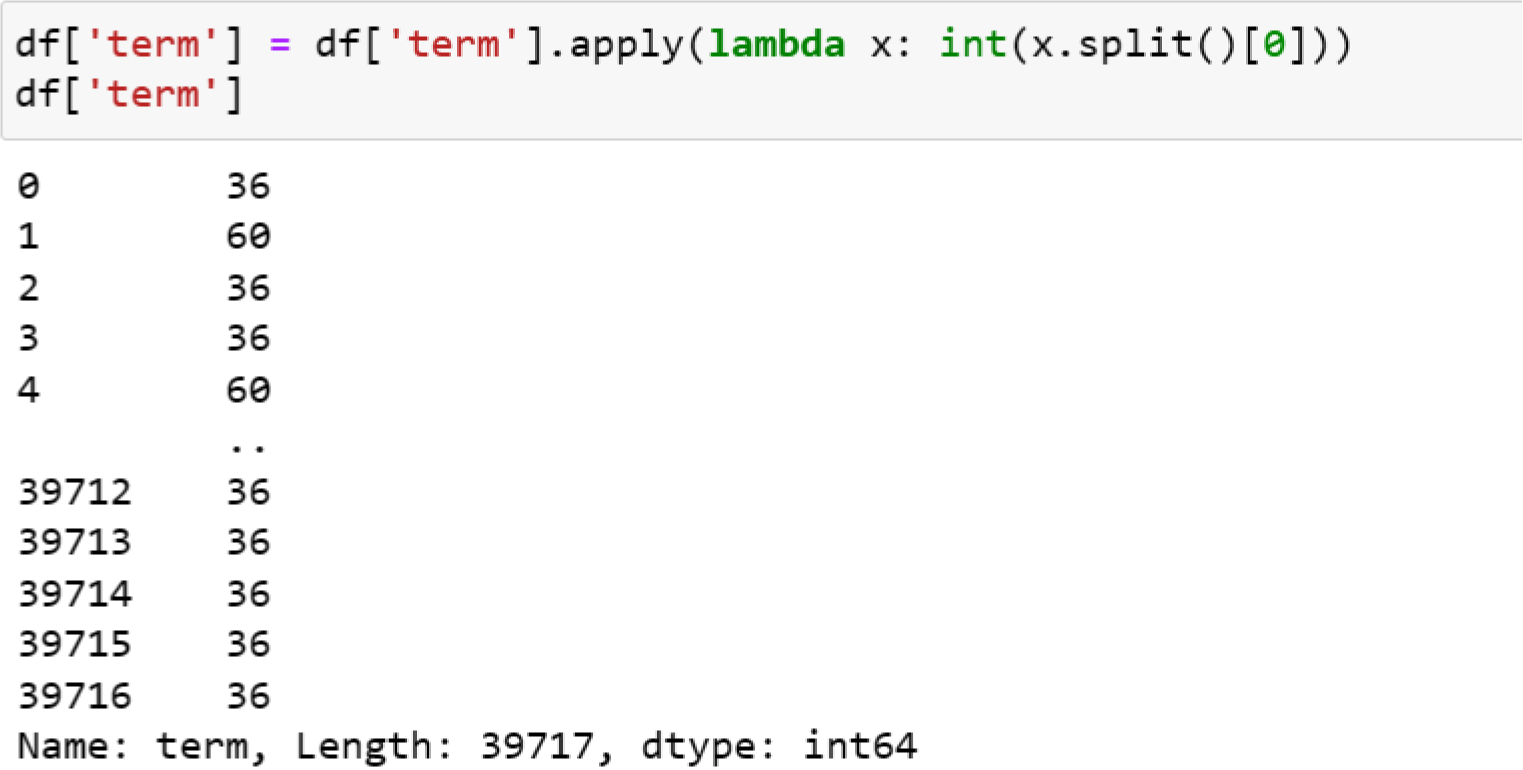
```
df["emp_length"]
```

```
0        10
1         1
2        10
3        10
4         1
         ..
39712     4
39713     3
39714     1
39715     1
39716     1
Name: emp_length, Length: 39717, dtype: object
```

Question 8

Using the Lambda function, remove the month from the ‘term’ column such that ‘36 months’, ‘60 months’ appear as 36 and 60 respectively.

#Write a code to extract only the numerical value from the column ‘term’ for ‘36 months’, ‘60 months’ appear as 36 and 60 respectively.



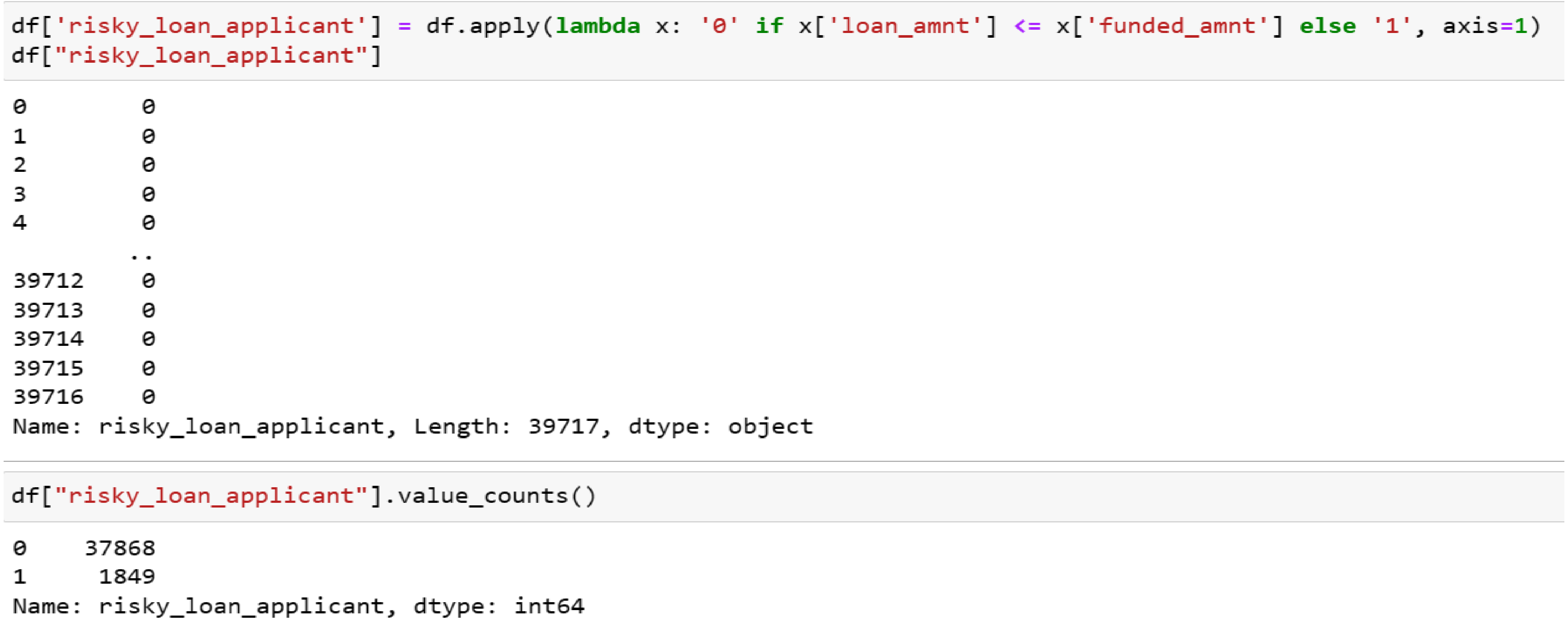
```
df['term'] = df['term'].apply(lambda x: int(x.split()[0]))
df['term']
```

```
0        36
1        60
2        36
3        36
4        60
         ..
39712    36
39713    36
39714    36
39715    36
39716    36
Name: term, Length: 39717, dtype: int64
```

Question 9

Create a new column as risky_loan_applicant by comparing loan_amnt and funded_amnt with the following criteria –
If loan_amnt is less than equals to funded_amnt set it as ‘0’ else set it as ‘1’.

# By comparing loan_amnt and funded_amnt with the following criteria, a new column labeled risky_loan_applicant can be created. If loan_amnt is less than equal to funded_amnt, set the column's value to "0," otherwise set it to "1."



```
df['risky_loan_applicant'] = df.apply(lambda x: '0' if x['loan_amnt'] <= x['funded_amnt'] else '1', axis=1)
df["risky_loan_applicant"]
```

```
0        0
1        0
2        0
3        0
4        0
        ..
39712    0
39713    0
39714    0
39715    0
39716    0
Name: risky_loan_applicant, Length: 39717, dtype: object
```

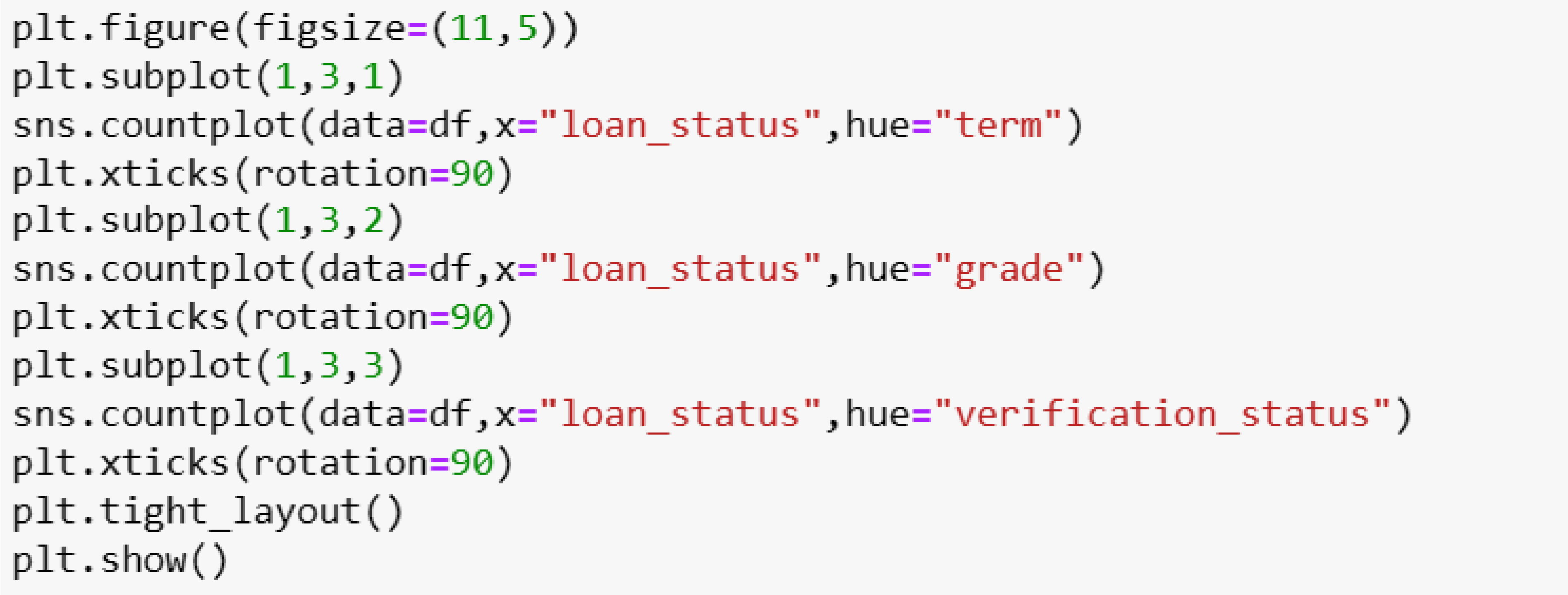
```
df["risky_loan_applicant"].value_counts()
```

```
0    37868
1     1849
Name: risky_loan_applicant, dtype: int64
```

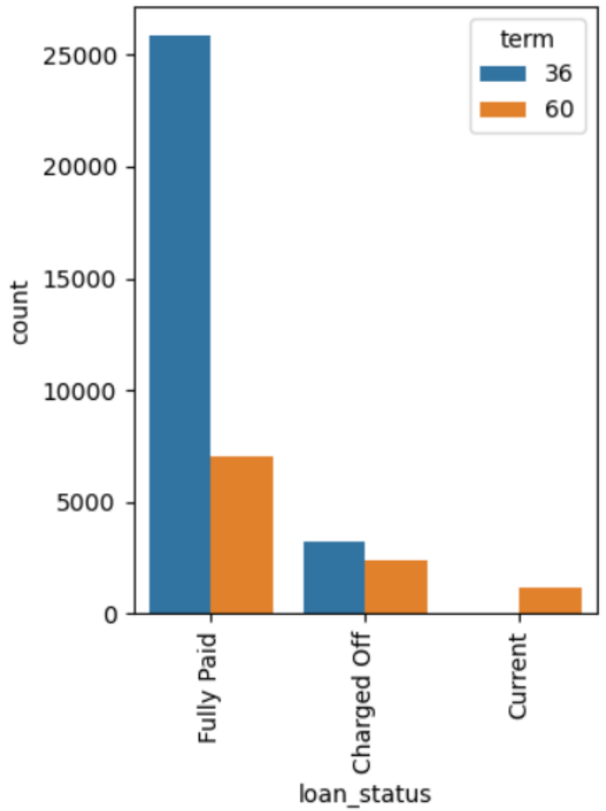
Question 10

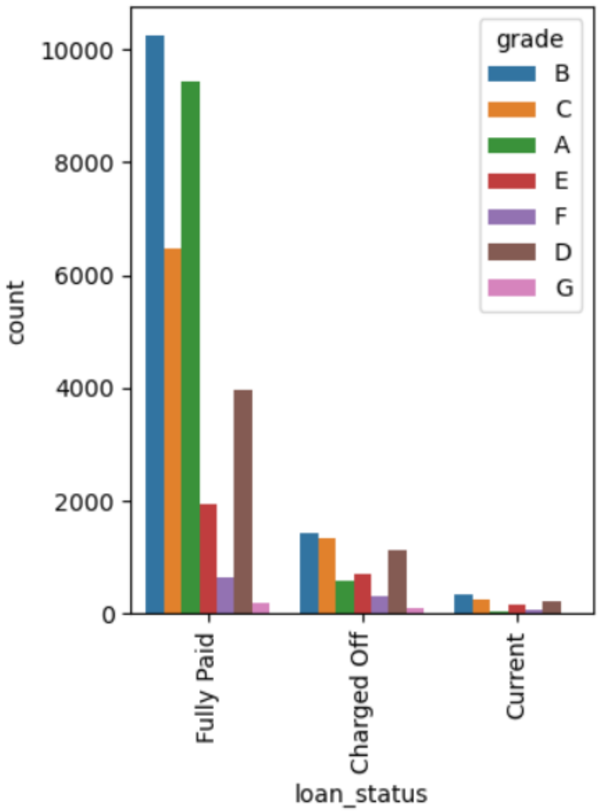
Using the bar plot visualize the loan_status column against categorical column grade, term, verification_status . Write the observation from each graph.

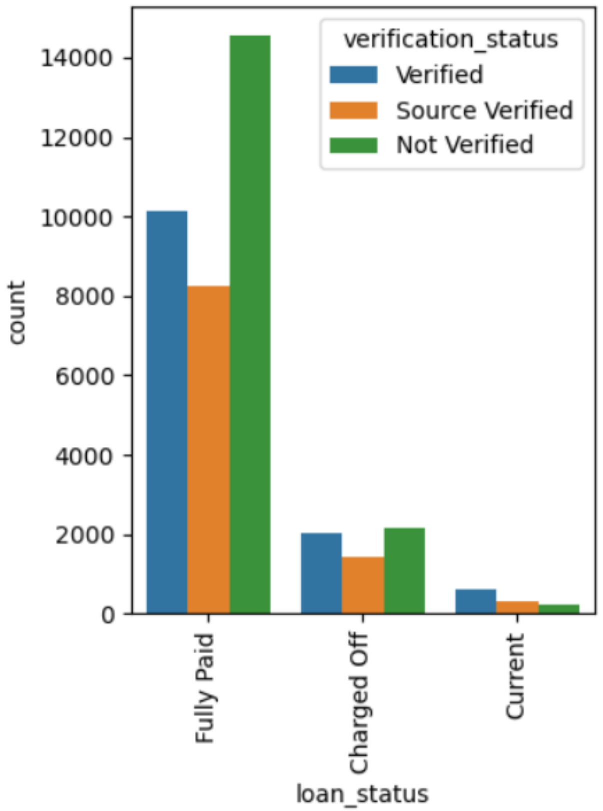
# Visualize the loan_status column against the categorical columns grade, term, and verification_status using a bar plot along with inferences.



```
plt.figure(figsize=(11,5))
plt.subplot(1,3,1)
sns.countplot(data=df,x="loan_status",hue="term")
plt.xticks(rotation=90)
plt.subplot(1,3,2)
sns.countplot(data=df,x="loan_status",hue="grade")
plt.xticks(rotation=90)
plt.subplot(1,3,3)
sns.countplot(data=df,x="loan_status",hue="verification_status")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```







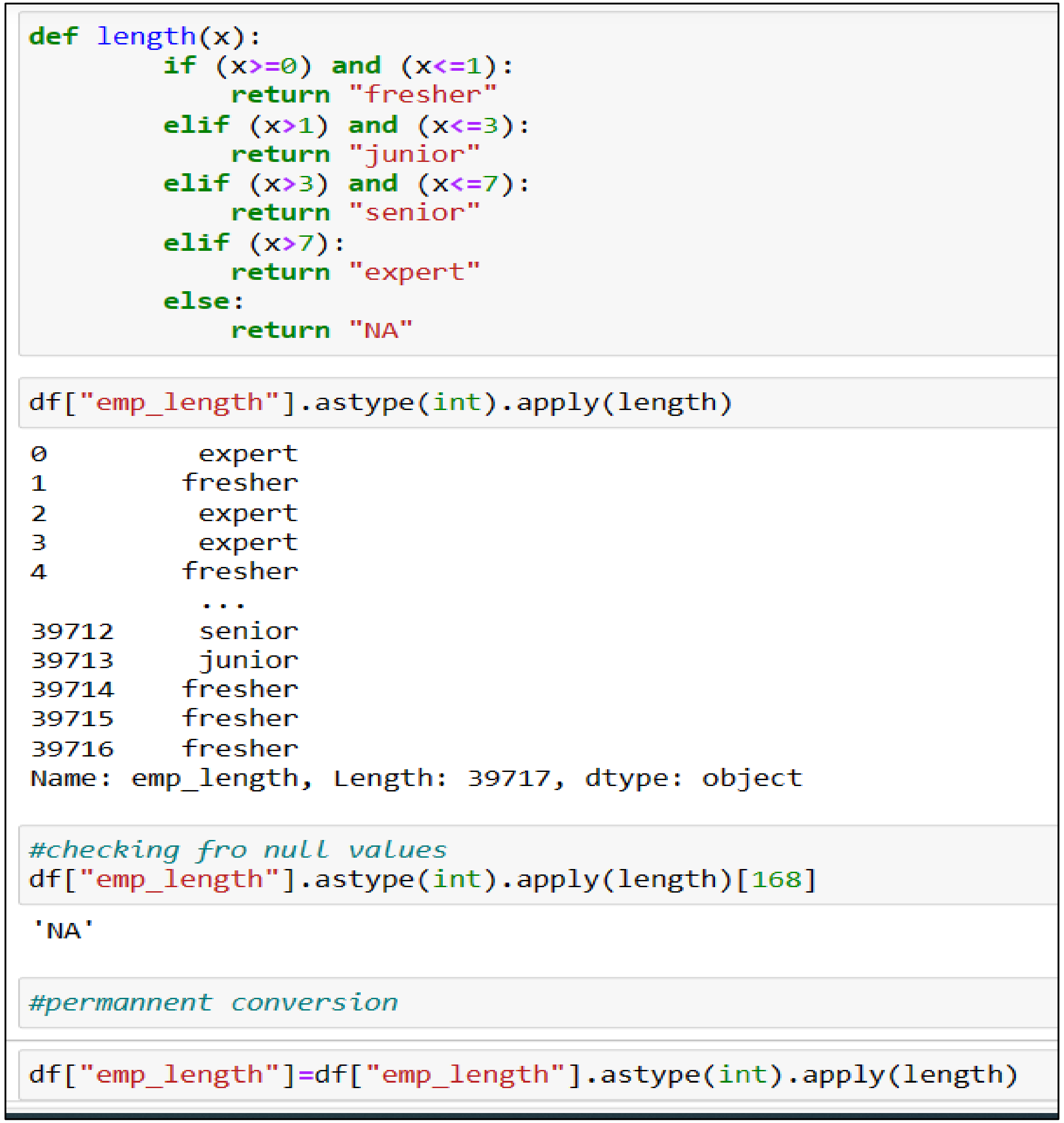
Inferences:

- We can see that number of not verified fully paid loan_status is more.
- It’s almost the same number of not verified and verified charged off loan status are present.
- We can see that verified current loan_status more in number.
- In verification status most of the people not verified but they fully paid.
- For the term 36, mostly the status is fully paid and there is no loan with current status.
- The count of loans with different grades are the least with current status.

Question 11

Using a user defined function convert the ‘emp_len’ column into categorical column as follows - If emp_len is less than equals to 1 then recode as ‘fresher’. If emp_len is greater than 1 and less than 3 then recode as ‘junior’. If emp_len is greater than 3 and less than 7 then recode as ‘senior’ If emp_len is greater than 7 then recode as ‘expert’.

# The "emp_len" column should be converted into a category column using the user-defined function as follows: If emp_len is less than 1, recode it as "fresher". Recode as 'junior' if emp_len is greater than 1 and less than 3, otherwise. Recode as 'senior' if emp_len is higher than 3 and less than 7 'Expert' should be recoded if emp_len is more than 7.



```
def length(x):
        if (x>=0) and (x<=1):
            return "fresher"
        elif (x>1) and (x<=3):
            return "junior"
        elif (x>3) and (x<=7):
            return "senior"
        elif (x>7):
            return "expert"
        else:
            return "NA"
```

```
df["emp_length"].astype(int).apply(length)
```

```
0         expert
1        fresher
2         expert
3         expert
4        fresher
          ...
39712     senior
39713     junior
39714    fresher
39715    fresher
39716    fresher
Name: emp_length, Length: 39717, dtype: object
```

```
#checking fro null values
df["emp_length"].astype(int).apply(length)[168]
```

```
'NA'
```

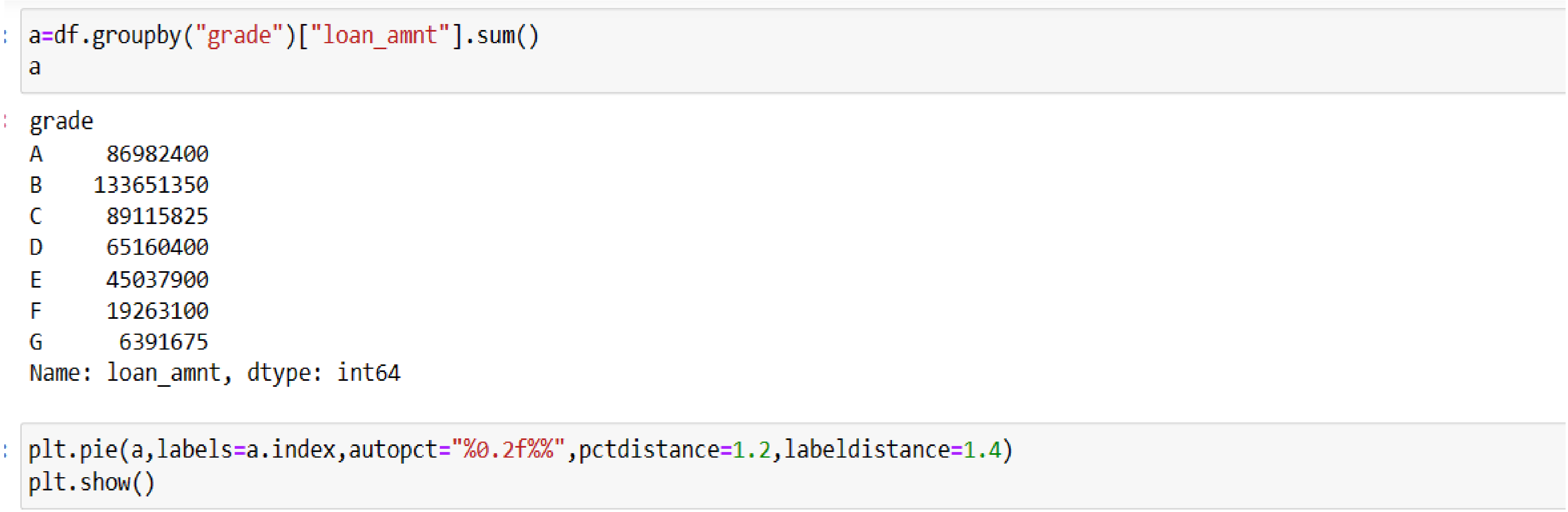
```
#permannent conversion
```

```
df["emp_length"]=df["emp_length"].astype(int).apply(length)
```

Question 12

Find the sum of ‘loan_amnt’ for each grade and display the distribution of ‘loan_amnt’ using a pie plot.

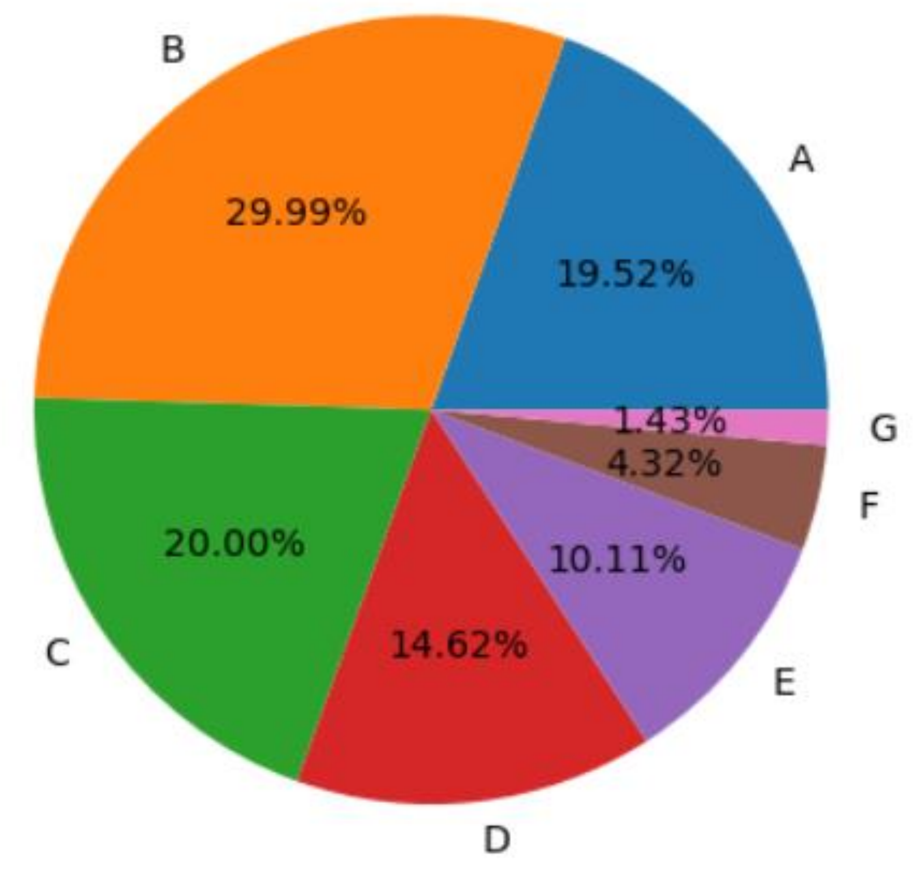
# For each grade, calculate the total amount of "loan_amnt" and use a pie chart to show the distribution of "loan_amnt."



```
a=df.groupby("grade")["loan_amnt"].sum()
a
```

```
grade
A     86982400
B    133651350
C     89115825
D     65160400
E     45037900
F     19263100
G      6391675
Name: loan_amnt, dtype: int64
```

```
plt.pie(a,labels=a.index,autopct="%0.2f%%",pctdistance=1.2,labeldistance=1.4)
plt.show()
```



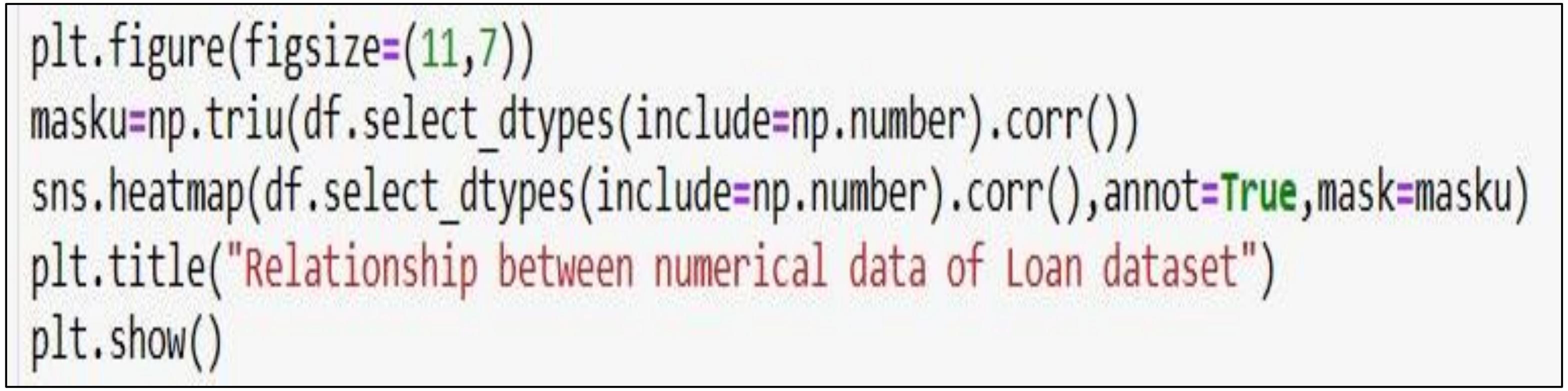
Inferences:

- The total loan amount provided by the company for loan grade B is highest, followed by that of grade C then A.
- The total loan amount for loan grade G is lowest, followed by that of grade F.

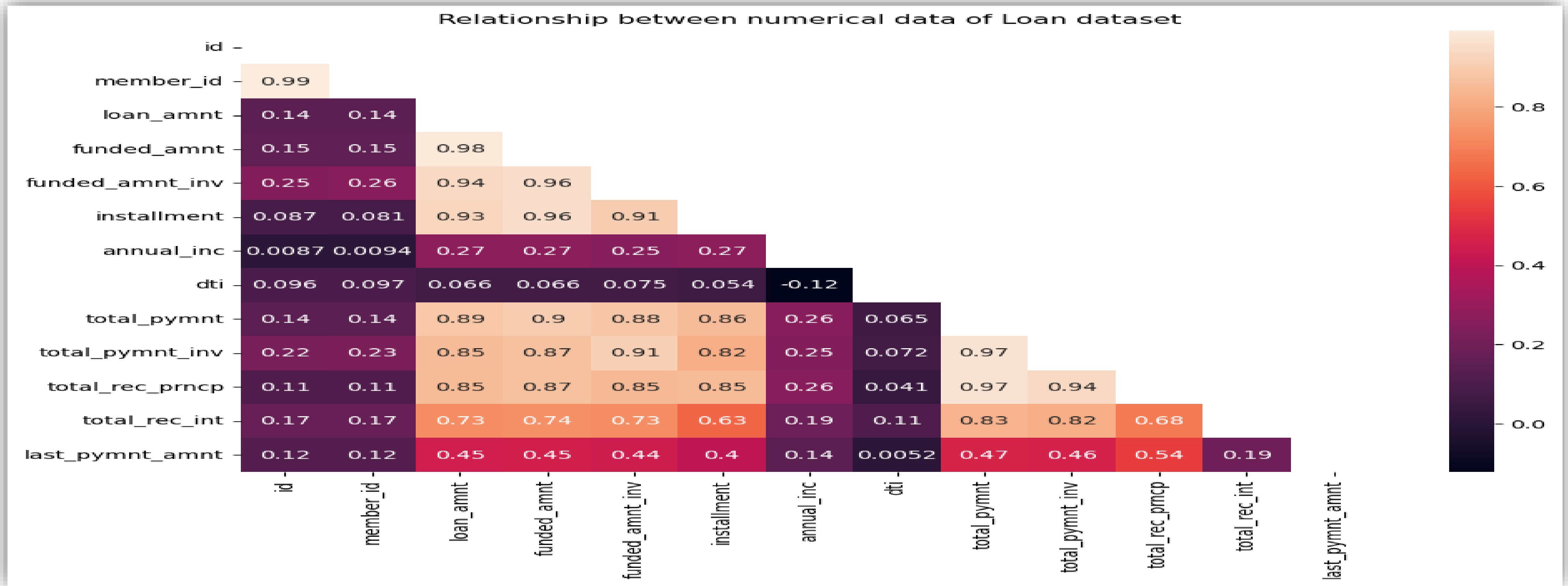
Question 13

Find the relation between the numerical values of the dataset.

- #obtain the correlation of numerical dataset (df.select_dtypes(include=np.number))
- #mask the upper triangle using np.triu
- #use sns.heatmap



```
plt.figure(figsize=(11,7))
masku=np.triu(df.select_dtypes(include=np.number).corr())
sns.heatmap(df.select_dtypes(include=np.number).corr(),annot=True,mask=masku)
plt.title("Relationship between numerical data of Loan dataset")
plt.show()
```



Relationship between numerical data of Loan dataset

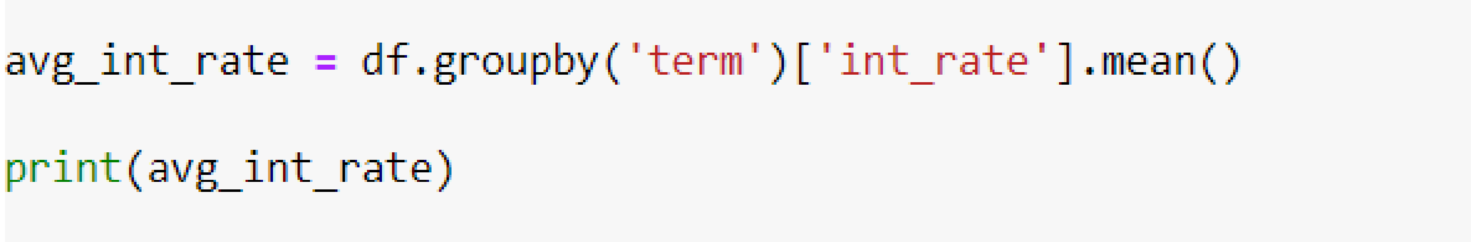
	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	dti	total_pymnt	total_pymnt_inv	total_rec_prncp	total_rec_int	last_pymnt_amnt
id													
member_id	0.99												
loan_amnt	0.14	0.14											
funded_amnt	0.15	0.15	0.98										
funded_amnt_inv	0.25	0.26	0.94	0.96									
installment	0.087	0.081	0.93	0.96	0.91								
annual_inc	0.0087	0.0094	0.27	0.27	0.25	0.27							
dti	0.096	0.097	0.066	0.066	0.075	0.054	-0.12						
total_pymnt	0.14	0.14	0.89	0.9	0.88	0.86	0.26	0.065					
total_pymnt_inv	0.22	0.23	0.85	0.87	0.91	0.82	0.25	0.072	0.97				
total_rec_prncp	0.11	0.11	0.85	0.87	0.85	0.85	0.26	0.041	0.97	0.94			
total_rec_int	0.17	0.17	0.73	0.74	0.73	0.63	0.19	0.11	0.83	0.82	0.68		
last_pymnt_amnt	0.12	0.12	0.45	0.45	0.44	0.4	0.14	0.0052	0.47	0.46	0.54	0.19	

- total_payment_invested has very high positive correlation with total_payment.
- total_rec_prncp has very high positive correlation with total_payment.
- total_funded_amnt has very high positive correlation with invested with loan amnt and funded amnt.
- total_payment has very high positive correlation with funded amnt.
- total rec_prici has very high positive correlation with total payment invested and total payment.
- member_id and id have high positive correlation but does not give inference as its just a number given to customer.
- dti and annual income have weak negative correlation.

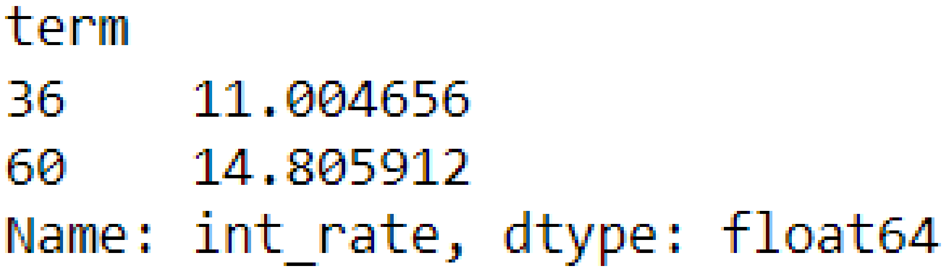
Question 14

How does the term of the loan affect the interest rate? You can compare the average interest rates for 36 months and 60 months terms..

- #The question is asking about the relationship between the term of a loan (how long it takes for the loan to be paid back) and the interest rate of the loan. Specifically, it’s asking whether loans with a term of 36 months have different average interest rates compared to loans with a term of 60 months.



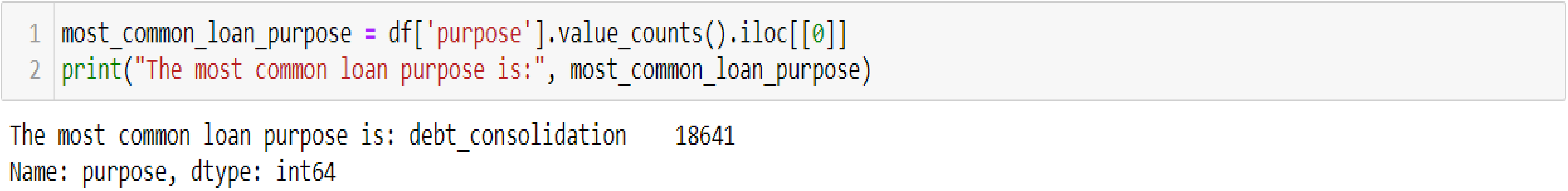
```
avg_int_rate = df.groupby('term')['int_rate'].mean()

print(avg_int_rate)
```



```
term
36    11.004656
60    14.805912
Name: int_rate, dtype: float64
```

Question 15
What is the most common loan purpose?

- #The question is asking for the most common purpose for which loans are taken out, according to the dataset. In the context of the dataset, each loan has a specific purpose (for example, debt consolidation, credit Card, Car, major purchase, etc.). and also find the count of it.

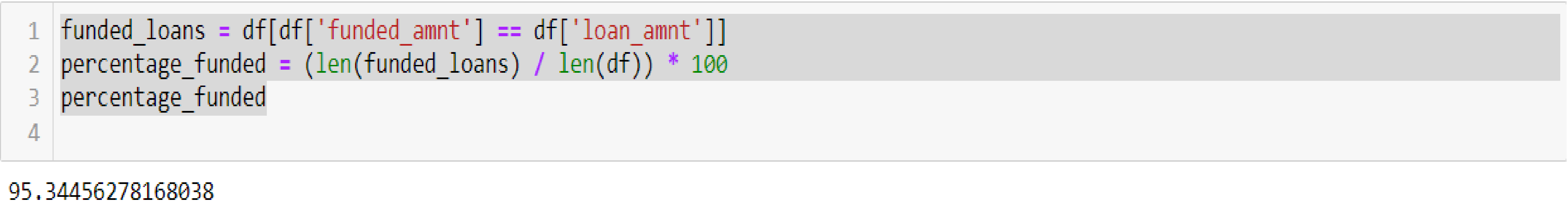


```
most_common_loan_purpose = df['purpose'].value_counts().iloc[[0]]
print("The most common loan purpose is:", most_common_loan_purpose)
```

```
The most common loan purpose is: debt_consolidation    18641
Name: purpose, dtype: int64
```

Question 16
What is the percentage of loans that were funded in full?

- #the proportion of loans where the entire amount requested was provided. This is typically calculated by dividing the number of fully funded loans by the total number of loans, and then multiplying by 100 to get a percentage.

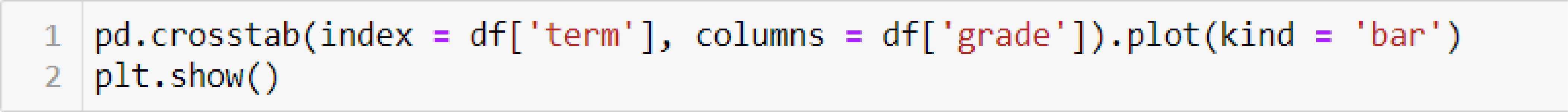


```python
funded_loans = df[df['funded_amnt'] == df['loan_amnt']]
percentage_funded = (len(funded_loans) / len(df)) * 100
percentage_funded
```

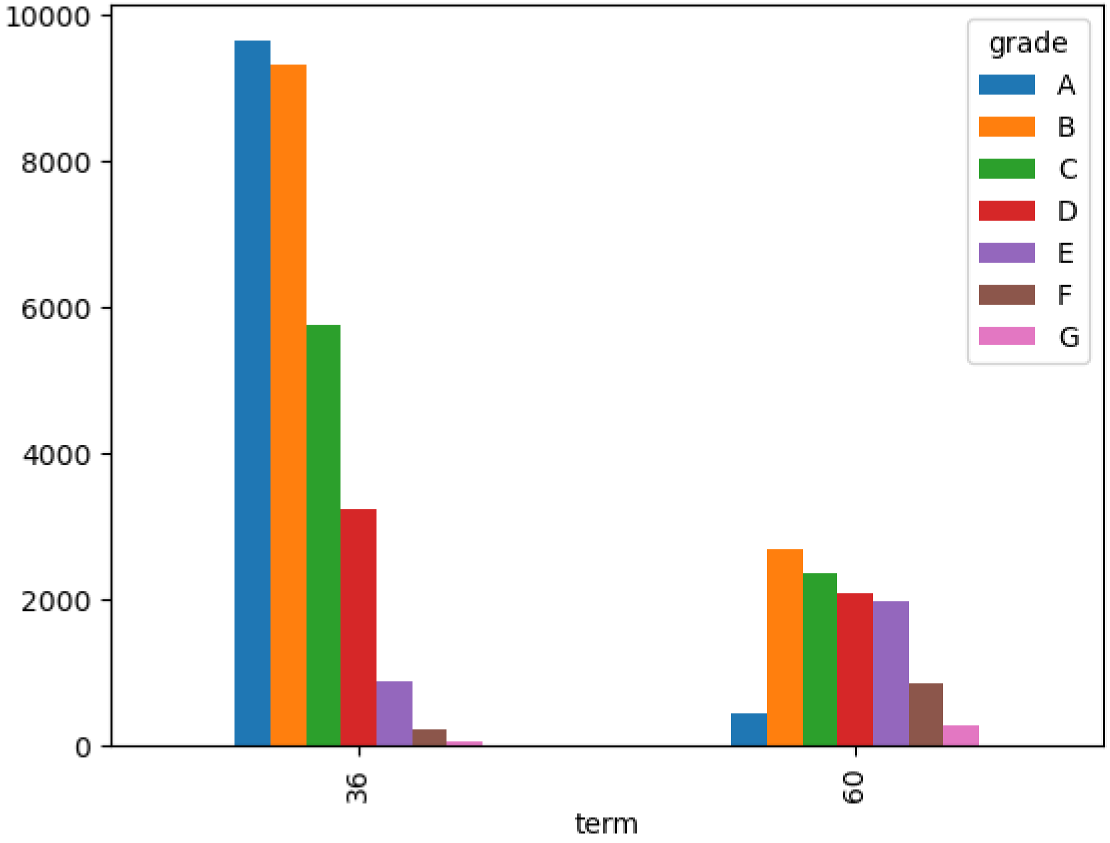
```
95.34456278168038
```

Question 17
In each term which grade is least common?

- #asking for the loan grade that appears the least frequently in each loan term. This could be relevant in a variety of contexts, such as understanding the risk profile of loans of different terms.



```python
pd.crosstab(index = df['term'], columns = df['grade']).plot(kind = 'bar')
plt.show()
```



- Inferences:
- grade G is least common in both the terms.

Conclusions

- **Lessons learned**

❖Importing data from various sources to python

❖Handling of different datatypes

❖Exposure to robust functions using different libraries of pythons

❖Handling of missing values

❖Manipulation of data using different tools of python

❖Detection of type of errors

❖Descriptive and descriptive statistics

Conclusions

- **Skills used**

❖Programming language :Python 3

❖Libraries used : Pandas, Numpy, Matplotlib.pyplot, Plotly.express, seaborn

❖Pandas series, pandas datframes,Slicing ,Filters, lambda functions, User-defined functions, Seaborn plots, Pyplots, numpy.nan, fillna, dropna.

❖Apply, map, replace, for loop,

Conclusions

- **Domain understanding developed**

❖An insight into the segregation of loan applications

❖Using previously captured data to consider new applications

❖Using skills of python to correlate between different attributes of loan applications

❖The impact of previously captured data on the future of the company

❖Risk analysis of the banking domain

THANK YOU