Mini-Project [Python]

TITLE : Loan Dataset

Submitted by : Nivedhitha R

Problem Definition

- Dataset : loan.csv

- Problem Statement : To identify variables which indicate if a person is likely to default, which can be used for identifying the risky loan applicants to avoid any financial loss to the company

Data Set Description

- The dataset contains the complete loan data for all loans issued through the time period 2007 to 2011

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

Data Set Description

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
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

Data Set Description

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

Data Set Description

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

Project Flow – Question 1

➢ Import the dataset and understand it

```
df = pd.read_csv('loan.csv')
```

- The above function is used to read and load data from a Comma-Separated Values (CSV) file into the Pandas DataFrame df

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	emp_length	...	p
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	10+ years	...	cr
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	< 1 year	...	ca
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	10+ years	...	sr
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	10+ years	...	ot
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	1 year	...	ot
...	...	...	...	...	...	...	...	...	...	...	...	...
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	4 years	...	ho
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	3 years	...	cr
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	< 1 year	...	de
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	< 1 year	...	ot
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	< 1 year	...	de

39717 rows × 23 columns

Project Flow – Question 2

➢ List down the number of rows and columns

```
df.info()
```

- Info() provides all the information about the DataFrame
- Inference:
- Number of columns = 23
- Number of rows = 39717

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 23 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  float64
 7   installment           39717 non-null  float64
 8   grade                 39717 non-null  object
 9   emp_length            38642 non-null  object
 10  annual_inc            39717 non-null  float64
 11  verification_status   39717 non-null  object
 12  loan_status           39717 non-null  object
 13  purpose               39717 non-null  object
 14  dti                   39717 non-null  float64
 15  total_pymnt           39717 non-null  float64
 16  total_pymnt_inv       39717 non-null  float64
 17  total_rec_prncp       39717 non-null  float64
 18  total_rec_int         39717 non-null  float64
 19  last_pymnt_d          39646 non-null  object
 20  last_pymnt_amnt       39717 non-null  float64
 21  Unnamed: 21           0 non-null      float64
 22  Unnamed: 22           0 non-null      float64
dtypes: float64(12), int64(4), object(7)
memory usage: 7.0+ MB
```

Great Learning
POWER AHEAD

Project Flow – Question 3

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

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

- apply() function is used along with lambda function with a condition to convert ‘int_rate’ column into float datatype
- Inference – ‘int_rate’ converted into float datatype

```
0        10.65
1        15.27
2        15.96
3        13.49
4        12.69
         ...
39712     8.07
39713    10.28
39714     8.07
39715     7.43
39716    13.75
Name: int_rate, Length: 39717, dtype: float64
```

Project Flow – Question 4

➢ Check the datatype of each column

```
df.dtypes
```

- The above code having ‘dtypes’ attribute returns a series with column names as the index and the corresponding data types as the values

```
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
```

Great Learning
POWER AHEAD

Project Flow – Question 5

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

```
df1 = df.dropna(axis = 1, thresh = 1)
```

- Here dropna() drops columns with any missing values (NaN) and stores into a new DataFrame called df1 that contains only the columns without missing values.
- Inference – Two columns having missing values (column number changed from 23 to 21)

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	emp_length	...	v
0	1077501	1296599	5000	5000	4975.0	36 months	10.65	162.87	B	10+ years	...	Ve
1	1077430	1314167	2500	2500	2500.0	60 months	15.27	59.83	C	< 1 year	...	Sc
2	1077175	1313524	2400	2400	2400.0	36 months	15.96	84.33	C	10+ years	...	No
3	1076863	1277178	10000	10000	10000.0	36 months	13.49	339.31	C	10+ years	...	Sc
4	1075358	1311748	3000	3000	3000.0	60 months	12.69	67.79	B	1 year	...	Sc
...	...	...	...	...	...	...	...	...	...	...	...	...
39712	92187	92174	2500	2500	1075.0	36 months	8.07	78.42	A	4 years	...	No
39713	90665	90607	8500	8500	875.0	36 months	10.28	275.38	C	3 years	...	No
39714	90395	90390	5000	5000	1325.0	36 months	8.07	156.84	A	< 1 year	...	No
39715	90376	89243	5000	5000	650.0	36 months	7.43	155.38	A	< 1 year	...	No
39716	87023	86999	7500	7500	800.0	36 months	13.75	255.43	E	< 1 year	...	No

39717 rows × 21 columns

Project Flow – 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

```
df2= df[(df['loan_status']=='Fully Paid') | (df['loan_status']=='Charged Off')]
df2['loan_status']
```

- Created a Boolean Series that indicates whether each element in the 'loan_status' column is equal to 'Fully Paid’ or ‘Charged Off’

- Returns the rows which satisfies the condition, the Solution
- Inference – 38577 entries with ‘loan_status’ either ‘Fully Paid’ or ‘Charged off’

```
            ...
39712     Fully Paid
39713     Fully Paid
39714     Fully Paid
39715     Fully Paid
39716     Fully Paid
Name: loan_status, Length: 38577, dtype: object
```

Project Flow – Question 7

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

```
df["emp_length"]=df["emp_length"].str.extract(r'(\d+)').astype(float)
```

- Above code extracts numerical values from the 'emp_length' column using the regular expression \d+, which matches one or more digits

- Inference – numerical values were extracted from ‘emp_length’ and replaced ‘emp_length’ column with the extracted values

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	emp_length	...	p
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	10.0	...	cr
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	1.0	...	ca
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	10.0	...	sr
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	10.0	...	ot
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	1.0	...	ot
...	...	...	...	...	...	...	...	...	...	...	...	...
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	4.0	...	ho
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	3.0	...	cr
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	1.0	...	de
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	1.0	...	ot
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	1.0	...	de

39717 rows × 23 columns

Project Flow – 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

```
df['term']=df['term'].apply(lambda x : x.strip('months'))
```

- apply() function is used along with lambda function with a strip condition to trim the term ‘months’ from the rows

- Inference – ‘term’ column changed as shown

```
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: object
```

Project Flow – 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’

```
df['risky_loan_applicant']=None
```

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

- apply() function is used along with lambda function with an if else statement as shown to change the values to 0 and 1 accordingly

- Inference – Rows were changed to zeros and ones

```
1  df['risky_loan_applicant'].unique()
```

```
array([0, 1], dtype=int64)
```

Project Flow – Question 10

- Using the bar plot visualize the loan_status column against categorical column grade, term, verification_status. Write the observation from each graph
- Function used to plot barplot – sns.countplot

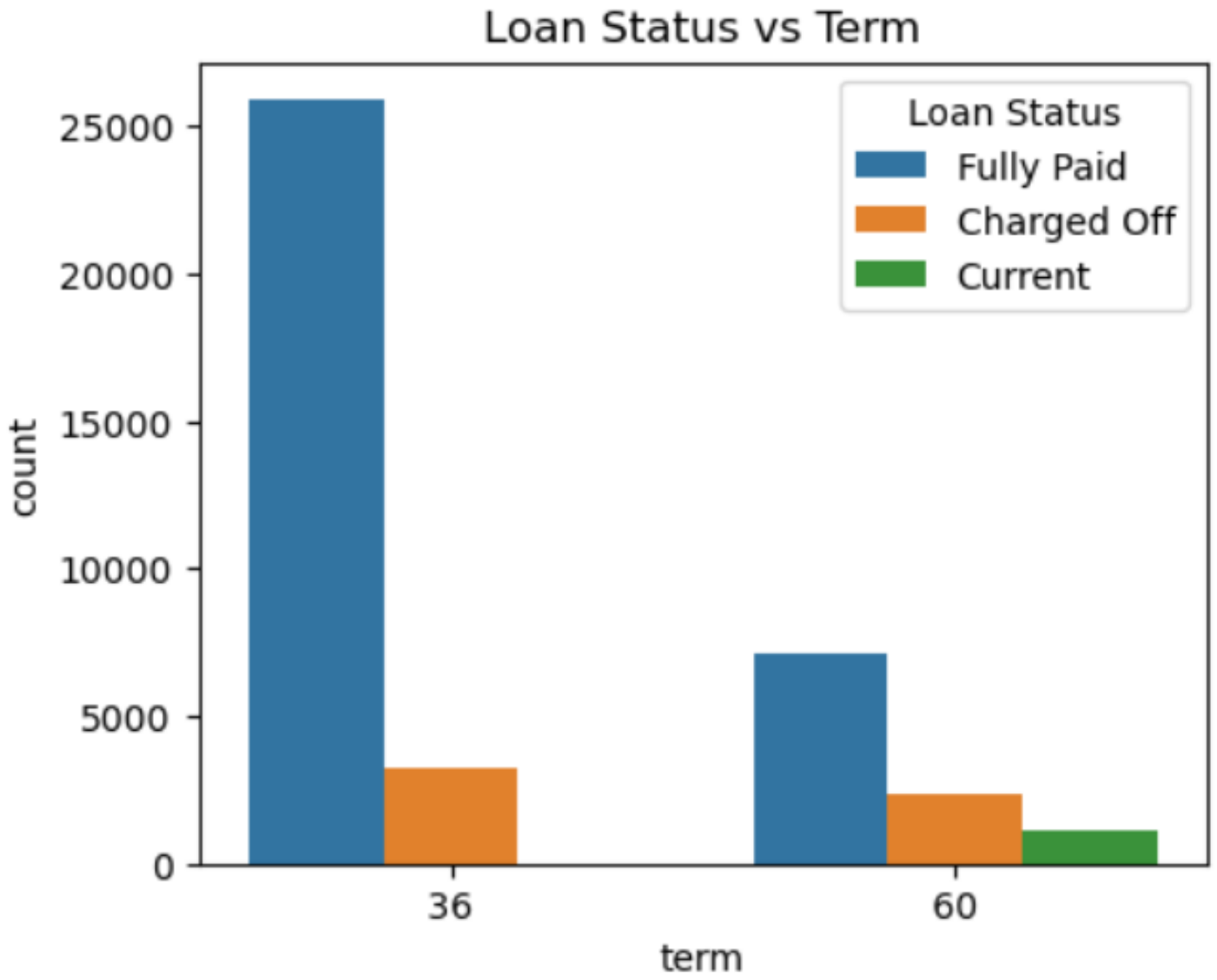
```
# for loan_status and term
plt.figure(figsize=(5, 4))
sns.countplot(data=df, x='term', hue='loan_status')
plt.title('Loan Status vs Term')
plt.legend(title='Loan Status')
plt.show()
```

```
# for loan_status and grade
plt.figure(figsize=(5, 4))
sns.countplot(data=df, x='grade', hue='loan_status')
plt.title('Loan Status vs Grade')
plt.legend(title='Loan Status')
plt.show()
```

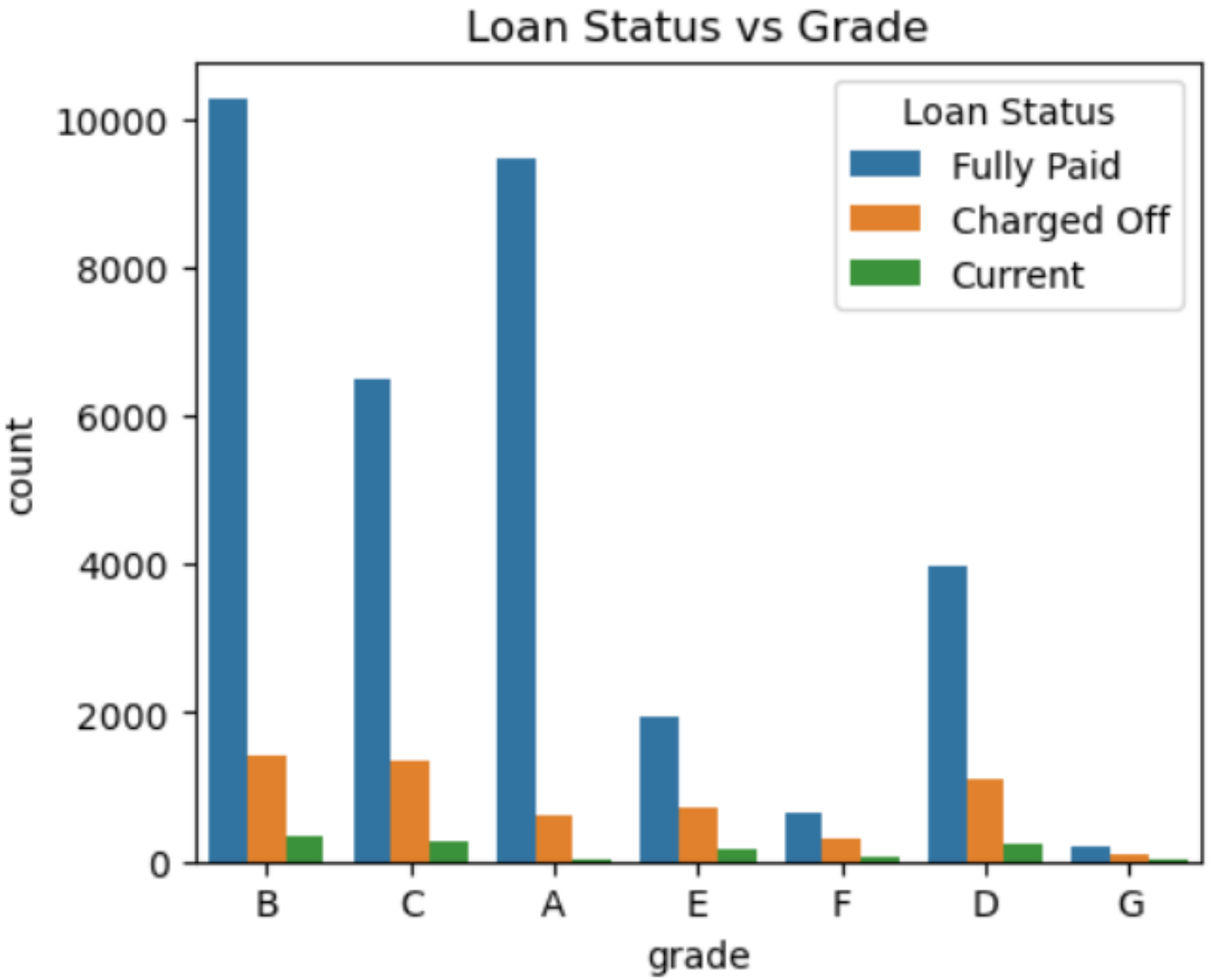
```
# for loan_status and verification_status
plt.figure(figsize=(5, 4))
sns.countplot(data=df, x='verification_status', hue='loan_status')
plt.title('Loan Status vs Term')
plt.legend(title='Loan Status')
plt.show()
```

Great Learning
POWER AHEAD

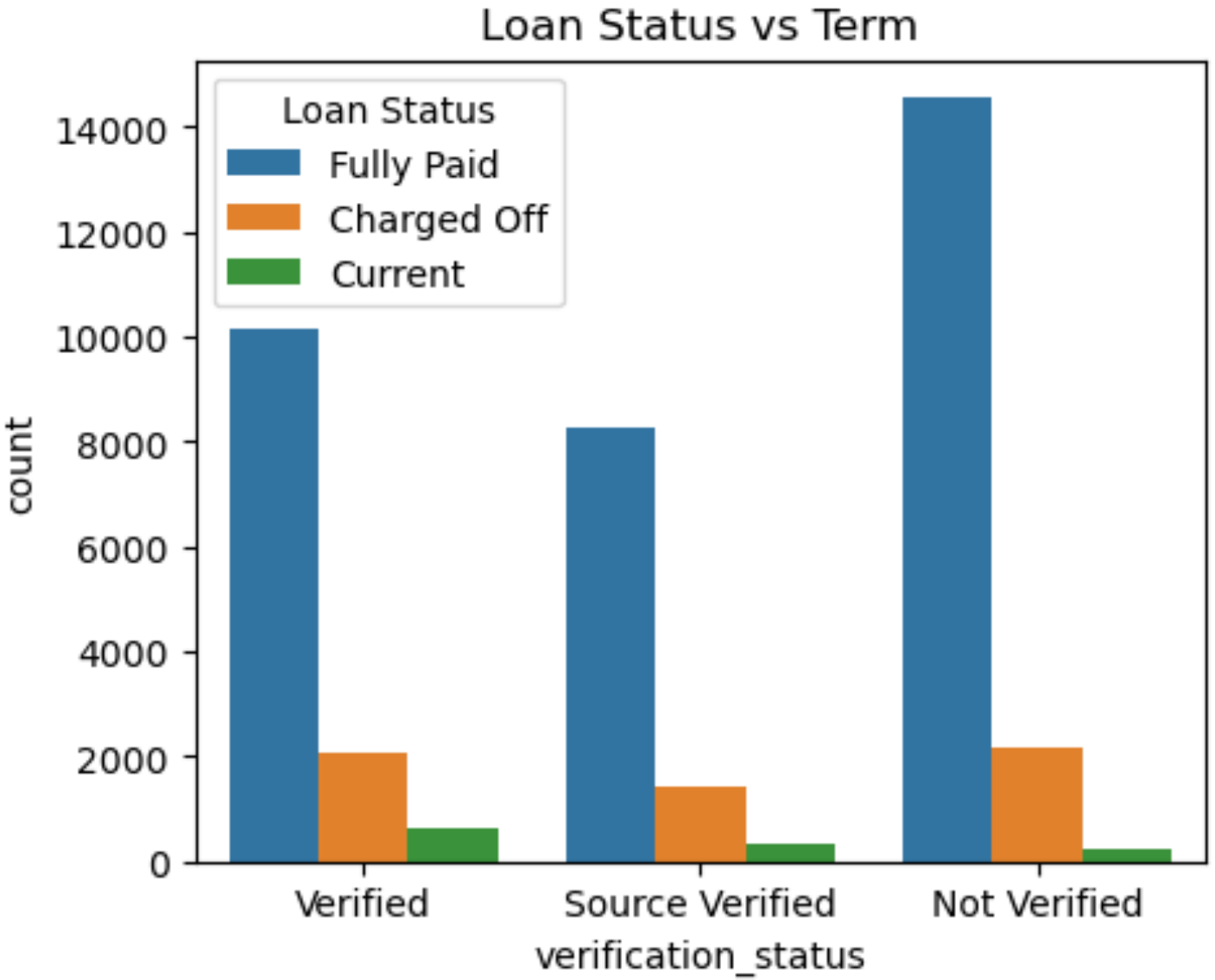
- Loans with a term of '36 months' have a higher proportion of 'Fully paid' loans compared to '60 months'.
- The number of 'Fully Paid' loans is higher for both term options.
- Loans with term '60 months' have loans which are in 'current' condition



- Grade 'B' has the highest number of 'Charged Off' loans compared to other grades.
- Grade 'B' has the highest number of 'Fully Paid' loans folled by Grade 'A'.



- Loans with 'Not Verified' verification status have a higher proportion of 'Fully Paid' loans.
- Loans with 'Verified' verification status have higher 'charged off' and 'current' loan status.
- Loans with 'Not Verified' verification status have lowest 'current' loan status



Project Flow – 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 record as ‘fresher’
- If emp_len is greater than 1 and less than 3 then record as ‘junior’
- If emp_len is greater than 3 and less than 7 then record as ‘senior’
- If emp_len is greater than 7 then record as ‘expert’
- Inference – if available

```python
def cate(emp_length):
    if emp_length <= 1:
        return "fresher"
    elif 1 < emp_length <3:
        return "junior"
    elif 3< emp_length <7:
        return "senior"
    else:
        return "expert"
```

```python
df['emp_length'] = df['emp_length'].apply(cate)
```

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

Project Flow – Question 12

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

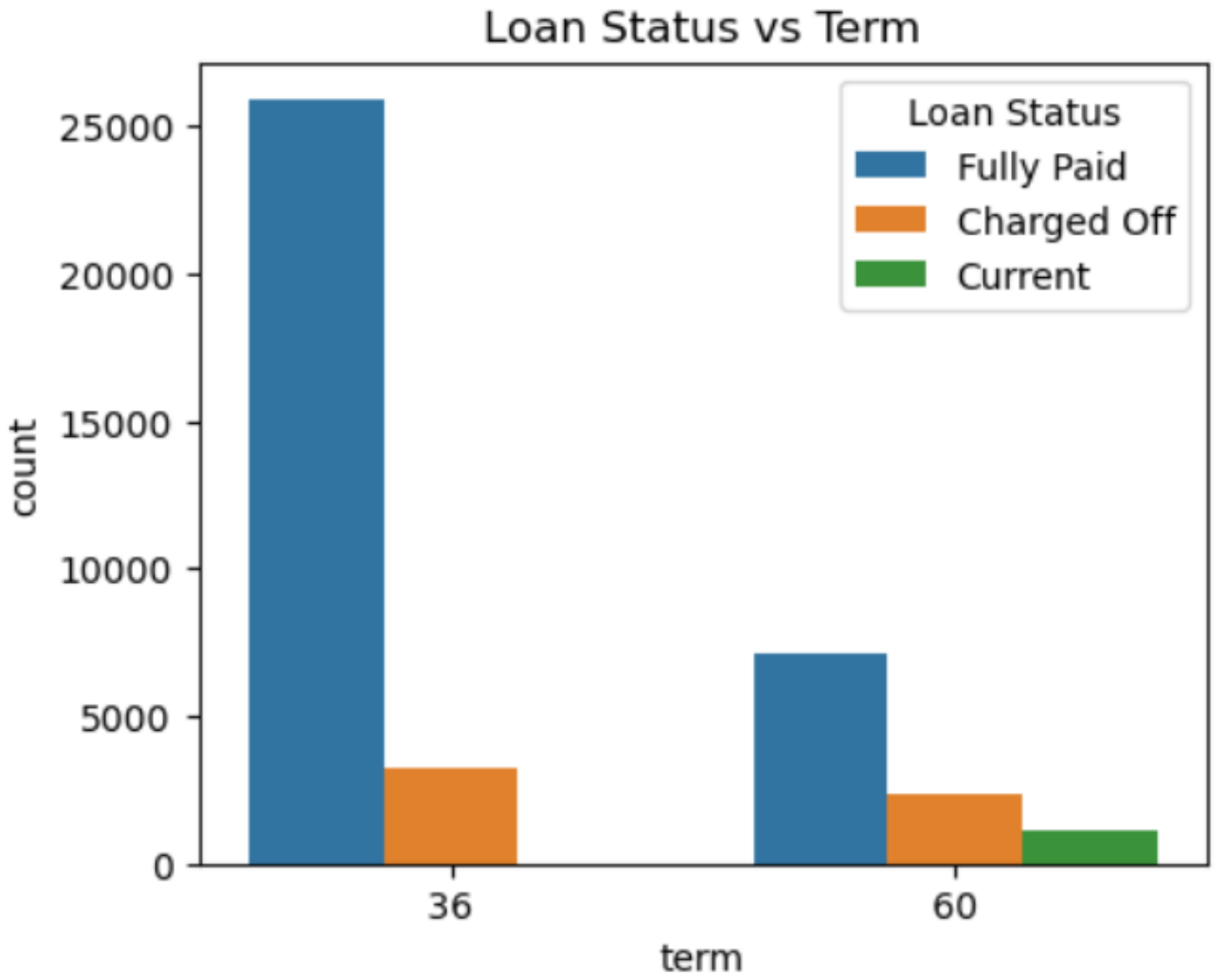
- To finding sum, groupby() function is used to find sum of ‘loan_amt’ with respect to ‘grade’ using the aggregate function sum()

```
s = df.groupby('grade')['loan_amnt'].sum()
```

- Function used to plot pie chart – plt.pie

```
plt.figure(figsize=(8,5))
plt.pie(s,labels=s.index,autopct="%.2f%%")
plt.show()
```

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



Major Challenge – Question 1

➢ What is the relationship between loan amount and interest rate?

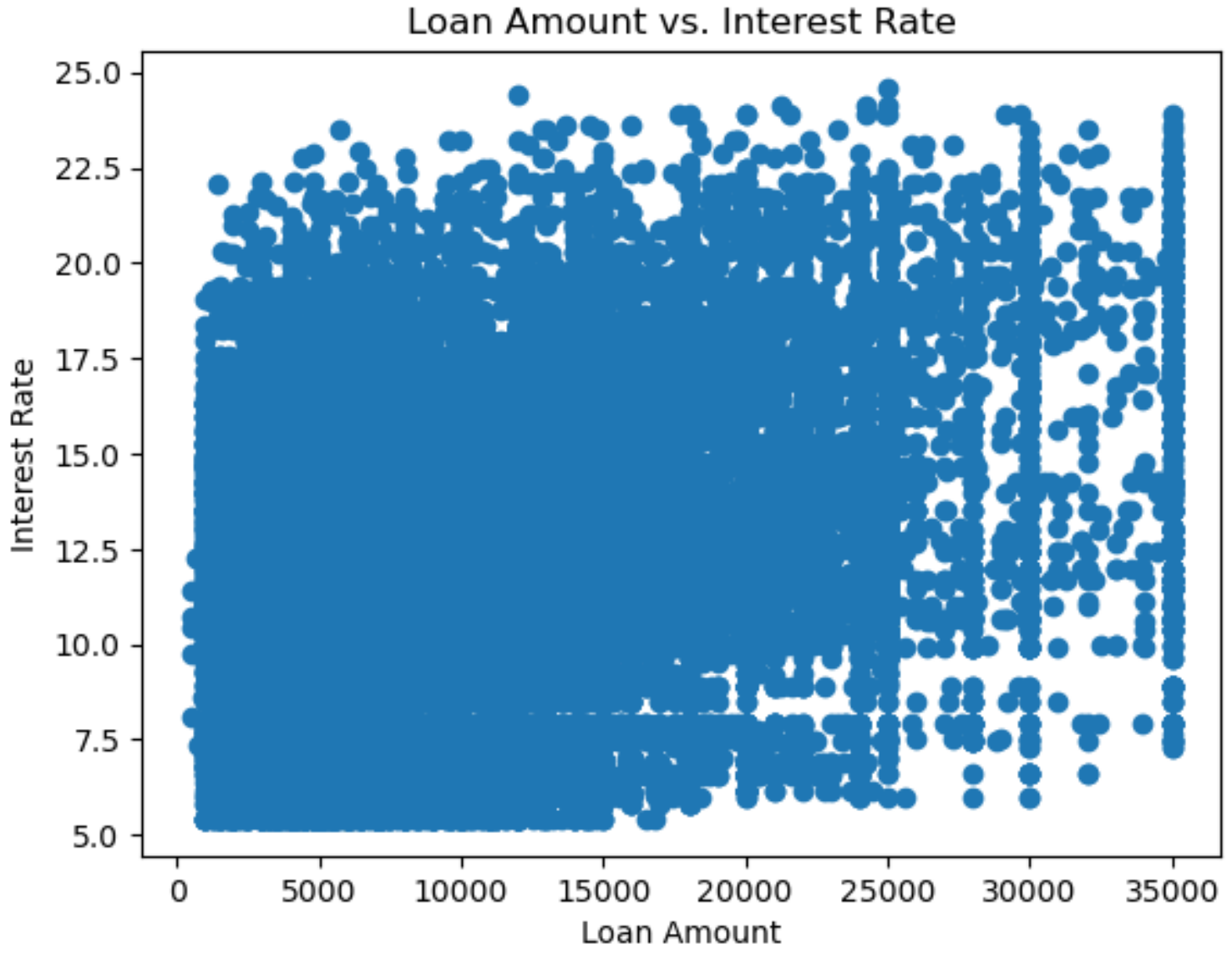
- To find relationship between two numeric data, scatterplot is plotted

```
plt.scatter(df['loan_amnt'], df['int_rate'])
plt.xlabel('Loan Amount')
plt.ylabel('Interest Rate')
plt.title('Loan Amount vs. Interest Rate')
plt.show()
```

Great Learning
POWER AHEAD

Inference – The relationship among

Loan amount and interest rate are dsense till the loan amount of 25,000



Major Challenge – Question 2

➢ What is the highest interest rate in the dataset?

```
interest_rates =df['int_rate']
highest_interest_rate = df['int_rate'].max()
print("Highest Interest Rate:", highest_interest_rate)
```

- The aggregate function max() is used to find maximum of ‘int_rate’

```
Highest Interest Rate: 24.59
```

- Inference – Highest Interest Rate is found to be 24.59%

Major Challenge – Question 3

➢ What is the average loan amount funded across all loans?

```
loan_amounts = df['loan_amnt']
average_loan_amount = np.mean(df['loan_amnt'])
print("Average Loan Amount:", average_loan_amount)
```

- The aggregate function avg() is used to find average of ‘int_rate’

```
Average Loan Amount: 11219.443814991062
```

- Inference – The average loan amount is found to be 11219.44

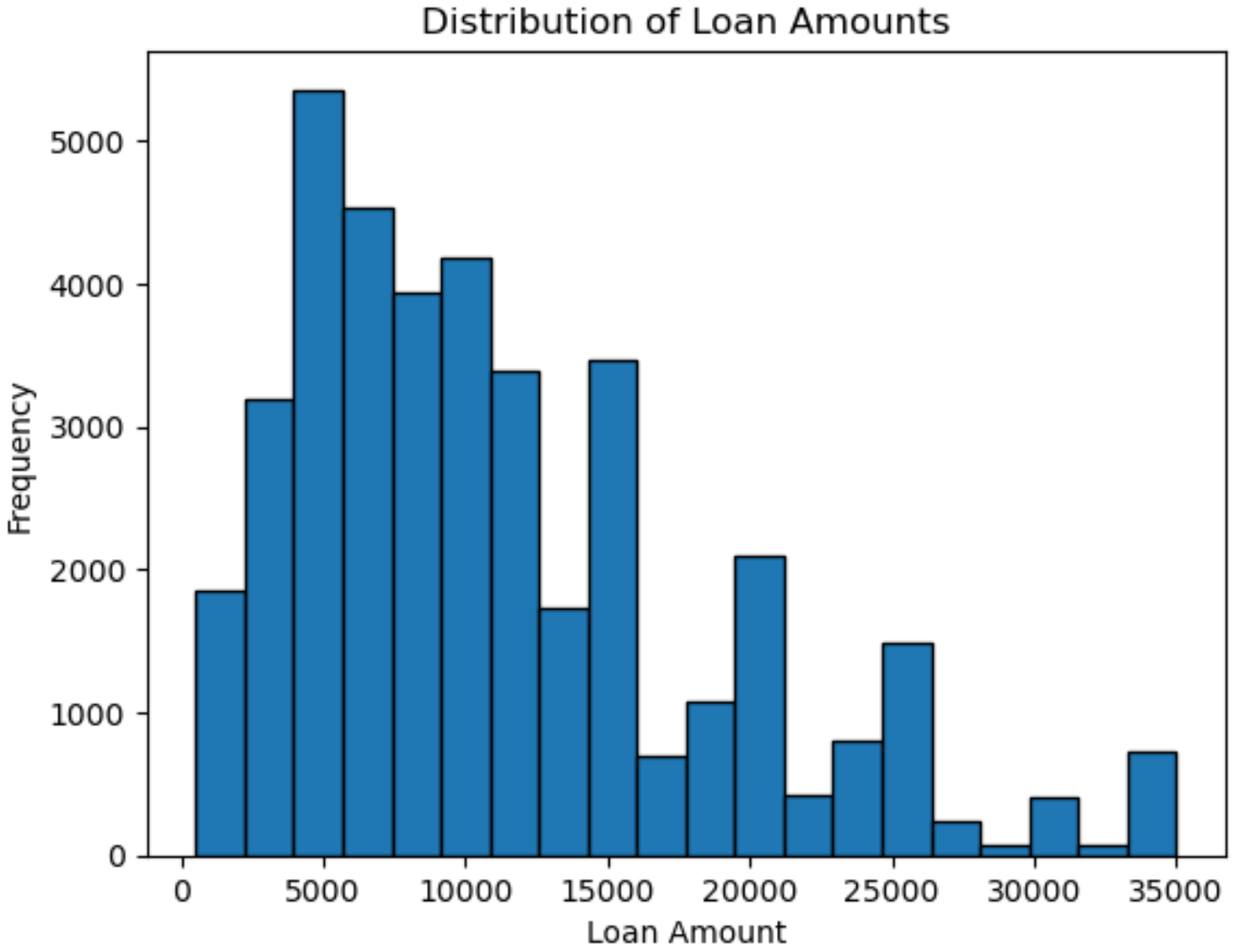
Major Challenge – Question 4

➢ How does the distribution of loan amounts look?

- To find distribution of numeric data, histogram is plotted

```
plt.hist(df['loan_amnt'], bins=20, edgecolor='black')
plt.xlabel('Loan Amount')
plt.ylabel('Frequency')
plt.title('Distribution of Loan Amounts')
plt.show()
```

Inference – The highest distributed loan amount is to be around range of 5000 and the lowest distributed loan amount is to be around range of 30000..



Major Challenge– Question 5

➢ How many loans were taken for different loan purposes?

```
loan_purposes_count = df['purpose'].value_counts()
print("Loan Purposes Count:\n", loan_purposes_count)
```

- Value_counts function is used to count the ‘purpose’ column
- Inference – Counts of each unique values in ‘purpose’ column is generated with ‘debt_consolidation’ having highest count

```
Loan Purposes Count:
 debt_consolidation    18641
credit_card             5130
other                   3993
home_improvement        2976
major_purchase          2187
small_business          1828
car                     1549
wedding                  947
medical                  693
moving                   583
vacation                 381
house                    381
educational              325
renewable_energy         103
Name: purpose, dtype: int64
```

Great Learning
POWER AHEAD

Conclusions

- Write the lessons learned

1. **Data Exploration**: The first step in any data analysis project is to thoroughly explore the dataset. This helps in understanding its structure, variables, and identifying potential issues.
2. **Data Cleaning**: Data cleaning is crucial. Handling missing values, duplicates, and outliers ensures the dataset's integrity and improves the accuracy of subsequent analyses.
3. **Data Visualisation**: Data visualisation plays a vital role in understanding the data more intuitively. Visualisations like histograms and line plots help in identifying trends, patterns, and outliers.

- Skills used

1. Data Cleaning and Preprocessing:
- Handling missing data: Using techniques like imputation or removal of missing values to ensure data completeness.
- Dealing with duplicates: Identifying and removing duplicate records to maintain data integrity.
- Outlier detection: Identifying and addressing outliers that might skew analysis results.

2. Data Exploration:
- Understanding dataset structure: Examining data types, columns, and general structure.
- Data summarization: Using methods like `.info()`, `.describe()`, and `.value_counts()` to gain insights into the data.

3. Data Visualization:
- Matplotlib and Seaborn: Creating visualizations such as histograms, line plots, and bar charts to explore data patterns.
- Visualization libraries: Leveraging Python libraries for data visualization to present findings effectively.

4. Descriptive Statistics:
- Calculating summary statistics like mean, median, and standard deviation to understand data characteristics.

5. Data Handling Libraries:
- Using Pandas for data manipulation and transformation.
- NumPy for numerical computations and array operations.

6. Project Documentation and Reporting:
- Writing clear and concise code comments.
- Creating project documentation to explain the analysis process and results.
- Communicating findings effectively through reports or presentations.

7. Domain Knowledge:
- Understanding the lending industry and loan-related concepts, such as credit scores, interest rates, and loan terms.

- Domain understanding developed

1. Loan Parameters:
- **loan_amnt**: The principal amount of the loan applied for, indicating the initial loan size.
- **funded_amnt**: The actual amount funded to the borrower, which may be lower than the requested amount.
- **funded_amnt_inv**: The total amount committed by investors for that loan.

2. Loan Term:
- **term**: The duration of the loan, typically expressed as the number of months

3. Interest Rate:
- **int_rate**: The annual interest rate for the loan, expressed as a percentage.

4. Loan Installment:
- **installment**: The monthly installment amount to be paid by the borrower, including both principal and interest.

5. Loan Grade:
- **grade**: A risk classification assigned to the loan, often based on the borrower's creditworthiness.

6. Employment Length:
- **emp_length**: The length of time the borrower has been employed, which can impact loan approval and terms.

7. **Borrower’s Income**:
- **annual_inc**: The borrower's annual income, a crucial factor in assessing their ability to repay the loan.

8. **Verification Status**:
- **verification_status**: Indicates whether the borrower's income and other information have been verified by the lending institution.

9. **Loan Status**:
- **loan_status**: Describes the current status of the loan, such as "Approved," "Defaulted," or "Fully Paid."

10. **Loan Purpose**:
- **purpose**: The reason for which the loan is taken, such as debt consolidation, home improvement, or medical expenses.

11. **Debt-to-Income Ratio**:
- **dti**: The borrower's debt-to-income ratio, representing their ability to manage debt based on income.

12. Loan Repayment Metrics:

- **total_pymnt**: The total amount paid by the borrower over the life of the loan.
- **total_pymnt_inv**: The total amount paid by the borrower, considering any investments made by others.
- **total_rec_prncp**: The total principal amount received by the lending institution.
- **total_rec_int**: The total interest amount received by the lending institution.

13. Last Payment Information:

- **last_pymnt_d**: The date of the last payment made by the borrower.
- **last_pymnt_amnt**: The amount of the last payment made by the borrower.

14. Unnamed Columns:

- **Unnamed: 21** and **Unnamed: 22**: These columns appear to be unnamed or contain unspecified data.

Great Learning
POWER AHEAD

Thank You !!!....