credit-score-calculation

June 27, 2024

1 Credit EDA & Credit Score Calculation

Problem statement:

To conduct a thorough exploratory data analysis (EDA) and deep analysis of a comprehensive dataset containing basic customer details and extensive credit-related information. The aim is to create new, informative features, calculate a hypothetical credit score, and uncover meaningful patterns, anomalies, and insights within the data.

Dataset Link: https://drive.google.com/file/d/1pljm6_3nxcFS9UMIFm124HBsjNZP6ACA/view?usp=sharing

Data Dictionary Link: https://docs.google.com/spreadsheets/d/1ZuK6o1MXFLmnhkFuDEedasDfVqu9ISPV/

Expectations: The project expects a deep dive into bank details and credit data, creating valuable features, a hypothetical credit score, and uncovering hidden patterns. This involves thorough EDA, strategic feature engineering, model-driven score calculation, and insightful analysis that reveals factors influencing creditworthiness and guides potential risk mitigation strategies.

Approach by Suchi Sharma:

Exploratory Data Analysis (EDA): Perform a comprehensive EDA to understand the data's structure, characteristics, distributions, and relationships. Identify and address any missing values, mismatch data types, inconsistencies, or outliers. Utilize appropriate visualizations (e.g., histograms, scatter plots, box plots, correlation matrices) to uncover patterns and insights.

Feature Engineering: Create new features that can be leveraged for the calculation of credit scores based on domain knowledge and insights from EDA. Aggregate the data on the customer level if required Hypothetical Credit Score Calculation:

Develop a methodology to calculate a hypothetical credit score using relevant features(use a minimum of 5 maximum of 10 features). Clearly outline the developed methodology in the notebook, providing a detailed explanation of the reasoning behind it. (use inspiration from FICO scores and try to use relevant features you created) Explore various weighting schemes to assign scores. Provide a score for each individual customer

Analysis and Insights: Add valuable insights from EDA and credit score calculation Can credit score and aggregated features be calculated at different time frames like the last 3 months/last 6 months (recency based metrics)

Remember, your analysis isn't just about dissecting data but uncovering actionable insights. Create a credit score strategy that you think would be the best and mention your justifications for criteria, weightage for the features. Suggestions are just general guidelines for the projects. It is not limited by that but serves as a starter and keeps it open to let you explore more, go into as much depth as you can, and actually make it your own project.

Schema of Data:

1. Import Libraries

[ ]:
```
import numpy as np
import pandas as pd

#Data Visualization

import matplotlib.pyplot as plt
import seaborn as sns
```

2. Import & Read Dataset

[ ]:
```
from google.colab import files
uploaded = files.upload()
```

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

```
Saving Credit_score.csv to Credit_score.csv
```

[ ]:
```
data = pd.read_csv("Credit_score.csv")
data.head()
```

[ ]:
```
       ID Customer_ID     Month           Name   Age          SSN Occupation  \
0  0x1602   CUS_0xd40   January  Aaron Maashoh    23  821-00-0265  Scientist
1  0x1603   CUS_0xd40  February  Aaron Maashoh    23  821-00-0265  Scientist
2  0x1604   CUS_0xd40     March  Aaron Maashoh  -500  821-00-0265  Scientist
3  0x1605   CUS_0xd40     April  Aaron Maashoh    23  821-00-0265  Scientist
4  0x1606   CUS_0xd40       May  Aaron Maashoh    23  821-00-0265  Scientist

  Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  …  \
0      19114.12            1824.843333                  3  …
1      19114.12                    NaN                  3  …
2      19114.12                    NaN                  3  …
3      19114.12                    NaN                  3  …
4      19114.12            1824.843333                  3  …

   Num_Credit_Inquiries  Credit_Mix Outstanding_Debt Credit_Utilization_Ratio  \
0                   4.0           _           809.98                26.822620
1                   4.0        Good           809.98                31.944960
2                   4.0        Good           809.98                28.609352
3                   4.0        Good           809.98                31.377862
4                   4.0        Good           809.98                24.797347
```

```
     Credit_History_Age Payment_of_Min_Amount Total_EMI_per_month  \
0  22 Years and 1 Months                    No           49.574949
1                   NaN                    No           49.574949
2  22 Years and 3 Months                    No           49.574949
3  22 Years and 4 Months                    No           49.574949
4  22 Years and 5 Months                    No           49.574949

  Amount_invested_monthly                 Payment_Behaviour Monthly_Balance
0             80.41529544   High_spent_Small_value_payments     312.4940887
1             118.2802216    Low_spent_Large_value_payments     284.6291625
2             81.69952126   Low_spent_Medium_value_payments     331.2098629
3             199.4580744    Low_spent_Small_value_payments     223.4513097
4             41.42015309  High_spent_Medium_value_payments      341.489231

[5 rows x 27 columns]
```

[ ]:
```
data.tail()
```

[ ]:
```
             ID Customer_ID   Month   Name Age         SSN Occupation  \
99995  0x25fe9  CUS_0x942c   April  Nicks  25  078-73-5990   Mechanic
99996  0x25fea  CUS_0x942c     May  Nicks  25  078-73-5990   Mechanic
99997  0x25feb  CUS_0x942c    June  Nicks  25  078-73-5990   Mechanic
99998  0x25fec  CUS_0x942c    July  Nicks  25  078-73-5990   Mechanic
99999  0x25fed  CUS_0x942c  August  Nicks  25  078-73-5990   Mechanic

      Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  …  \
99995      39628.99            3359.415833                  4  …
99996      39628.99            3359.415833                  4  …
99997      39628.99            3359.415833                  4  …
99998      39628.99            3359.415833                  4  …
99999     39628.99_            3359.415833                  4  …

       Num_Credit_Inquiries  Credit_Mix Outstanding_Debt  \
99995                   3.0           _           502.38
99996                   3.0           _           502.38
99997                   3.0        Good           502.38
99998                   3.0        Good           502.38
99999                   3.0        Good           502.38

      Credit_Utilization_Ratio      Credit_History_Age Payment_of_Min_Amount  \
99995                34.663572   31 Years and 6 Months                    No
99996                40.565631   31 Years and 7 Months                    No
99997                41.255522   31 Years and 8 Months                    No
99998                33.638208   31 Years and 9 Months                    No
99999                34.192463  31 Years and 10 Months                    No
```

```
      Total_EMI_per_month  Amount_invested_monthly  \
99995            35.104023              60.97133256
99996            35.104023              54.18595029
99997            35.104023              24.02847745
99998            35.104023              251.6725822
99999            35.104023              167.1638652

                      Payment_Behaviour Monthly_Balance
99995   High_spent_Large_value_payments      479.866228
99996  High_spent_Medium_value_payments       496.65161
99997   High_spent_Large_value_payments      516.809083
99998    Low_spent_Large_value_payments      319.164979
99999                            !@9#%8      393.673696

[5 rows x 27 columns]
```

3. EDA - Exploratory Data Analysis

EDA refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Here, we will perform EDA on the **categorical columns** of the dataset - and the **numerical columns** of the dataset -

[ ]:
```
data.shape
```

[ ]:
```
(100000, 27)
```

3.1 Datatypes, Missing Data, and Summary Statistics

Shape of the Dataset:

- Columns: 27
- Rows: 100,000

[ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 27 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   ID                        100000 non-null  object
 1   Customer_ID               100000 non-null  object
 2   Month                     100000 non-null  object
 3   Name                      90015 non-null   object
 4   Age                       100000 non-null  object
 5   SSN                       100000 non-null  object
 6   Occupation                100000 non-null  object
```

```
 7   Annual_Income             100000 non-null  object
 8   Monthly_Inhand_Salary     84998 non-null   float64
 9   Num_Bank_Accounts         100000 non-null  int64
 10  Num_Credit_Card           100000 non-null  int64
 11  Interest_Rate             100000 non-null  int64
 12  Num_of_Loan               100000 non-null  object
 13  Type_of_Loan              88592 non-null   object
 14  Delay_from_due_date       100000 non-null  int64
 15  Num_of_Delayed_Payment    92998 non-null   object
 16  Changed_Credit_Limit      100000 non-null  object
 17  Num_Credit_Inquiries      98035 non-null   float64
 18  Credit_Mix                100000 non-null  object
 19  Outstanding_Debt          100000 non-null  object
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        90970 non-null   object
 22  Payment_of_Min_Amount     100000 non-null  object
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   95521 non-null   object
 25  Payment_Behaviour         100000 non-null  object
 26  Monthly_Balance           98800 non-null   object
dtypes: float64(4), int64(4), object(19)
memory usage: 20.6+ MB
```

Observations:

- There are missing values in the dataset
- Dataset has both numerical and string values in columns
- Some data types are incorrectly assigned

[ ]:
```
data.describe().T
```

[ ]:
```
                             count         mean          std         min  \
Monthly_Inhand_Salary      84998.0  4194.170850  3183.686167  303.645417
Num_Bank_Accounts         100000.0    17.091280   117.404834   -1.000000
Num_Credit_Card           100000.0    22.474430   129.057410    0.000000
Interest_Rate             100000.0    72.466040   466.422621    1.000000
Delay_from_due_date       100000.0    21.068780    14.860104   -5.000000
Num_Credit_Inquiries       98035.0    27.754251   193.177339    0.000000
Credit_Utilization_Ratio  100000.0    32.285173     5.116875   20.000000
Total_EMI_per_month       100000.0  1403.118217  8306.041270    0.000000

                                  25%          50%          75%          max
Monthly_Inhand_Salary     1625.568229  3093.745000  5957.448333  15204.63333
Num_Bank_Accounts            3.000000     6.000000     7.000000   1798.00000
Num_Credit_Card              4.000000     5.000000     7.000000   1499.00000
Interest_Rate                8.000000    13.000000    20.000000   5797.00000
Delay_from_due_date         10.000000    18.000000    28.000000     67.00000
Num_Credit_Inquiries         3.000000     6.000000     9.000000   2597.00000
```

```
Credit_Utilization_Ratio    28.052567    32.305784    36.496663     50.00000
Total_EMI_per_month         30.306660    69.249473   161.224249  82331.00000
```

[ ]:
```python
data.describe(exclude = np.number).T
```

[ ]:
```
                          count  unique                            top  \
ID                       100000  100000                         0x1602
Customer_ID              100000   12500                      CUS_0xd40
Month                    100000       8                        January
Name                      90015   10139                         Langep
Age                      100000    1788                             38
SSN                      100000   12501                    #F%$D@*&8
Occupation               100000      16                        _______
Annual_Income            100000   18940                       36585.12
Num_of_Loan              100000     434                              3
Type_of_Loan              88592    6260                  Not Specified
Num_of_Delayed_Payment    92998     749                             19
Changed_Credit_Limit     100000    3635                              _
Credit_Mix               100000       4                       Standard
Outstanding_Debt         100000   13178                        1360.45
Credit_History_Age        90970     404          15 Years and 11 Months
Payment_of_Min_Amount    100000       3                            Yes
Amount_invested_monthly   95521   91049                      __10000__
Payment_Behaviour        100000       7    Low_spent_Small_value_payments
Monthly_Balance           98800   98790  __-333333333333333333333333333__

                          freq
ID                           1
Customer_ID                  8
Month                    12500
Name                        44
Age                       2833
SSN                       5572
Occupation                7062
Annual_Income               16
Num_of_Loan              14386
Type_of_Loan              1408
Num_of_Delayed_Payment    5327
Changed_Credit_Limit      2091
Credit_Mix               36479
Outstanding_Debt            24
Credit_History_Age         446
Payment_of_Min_Amount    52326
Amount_invested_monthly   4305
Payment_Behaviour        25513
Monthly_Balance              9
```

Observations:

- As per Customer_ID column, data has 12500 unique customers.
- Strange but month has only 8 unique values vs. 12. Need to analyze further.
- Age has 1788 unique values which is again illogical as age is generally between 0 - 100.
- SSN has 12501 values vs customers value being 12500. Seems one of the customers has distinct SSNs which is likely to be data quality issue.
- SSN values are some multiple character values which seems to be junk.
- Data clean-up is required to remove unwanted and mis-aligned values in certain columns.

3.2. Data clean-up of missing values, inconsistency, mismatch and outliers

[ ]:
```
data.isna().sum()
```

[ ]:
```
ID                             0
Customer_ID                    0
Month                          0
Name                        9985
Age                            0
SSN                            0
Occupation                     0
Annual_Income                  0
Monthly_Inhand_Salary      15002
Num_Bank_Accounts              0
Num_Credit_Card                0
Interest_Rate                  0
Num_of_Loan                    0
Type_of_Loan               11408
Delay_from_due_date            0
Num_of_Delayed_Payment      7002
Changed_Credit_Limit           0
Num_Credit_Inquiries        1965
Credit_Mix                     0
Outstanding_Debt               0
Credit_Utilization_Ratio       0
Credit_History_Age          9030
Payment_of_Min_Amount          0
Total_EMI_per_month            0
Amount_invested_monthly     4479
Payment_Behaviour              0
Monthly_Balance             1200
dtype: int64
```

Column : Name

[ ]:
```
data.sort_values(by = ['Customer_ID','Month'], inplace = True)
```

[ ]:
```
data['Name'] = data.groupby('Customer_ID')['Name'].fillna(method = 'ffill').
↪fillna(method = 'bfill')
```

Column: Age

```
[ ]: data['Age'].value_counts()
```

```
[ ]: Age
     38        2833
     28        2829
     31        2806
     26        2792
     32        2749
               …
     2204         1
     2474_        1
     620          1
     6922         1
     6494         1
     Name: count, Length: 1788, dtype: int64
```

```
[ ]: data['Age'] = data['Age'].where((data['Age'] >= 0) & (data['Age'] <= 120), pd.
      ↪NA)
     data['Age'] = data.groupby('Customer_ID')['Age'].transform(lambda x: x.fillna(x.
      ↪mode().iloc[0]))
```

```
[ ]: data['Age'] = data.groupby('Customer_ID')['Age'].transform(lambda x: x.
      ↪replace(x.max(),x.mode().iloc[0]))
     data['Age'] = data.groupby('Customer_ID')['Age'].transform(lambda x: x.
      ↪replace(x.min(),x.mode().iloc[0]))
```

```
[ ]: data['Age']= data['Age'].astype(int)
```

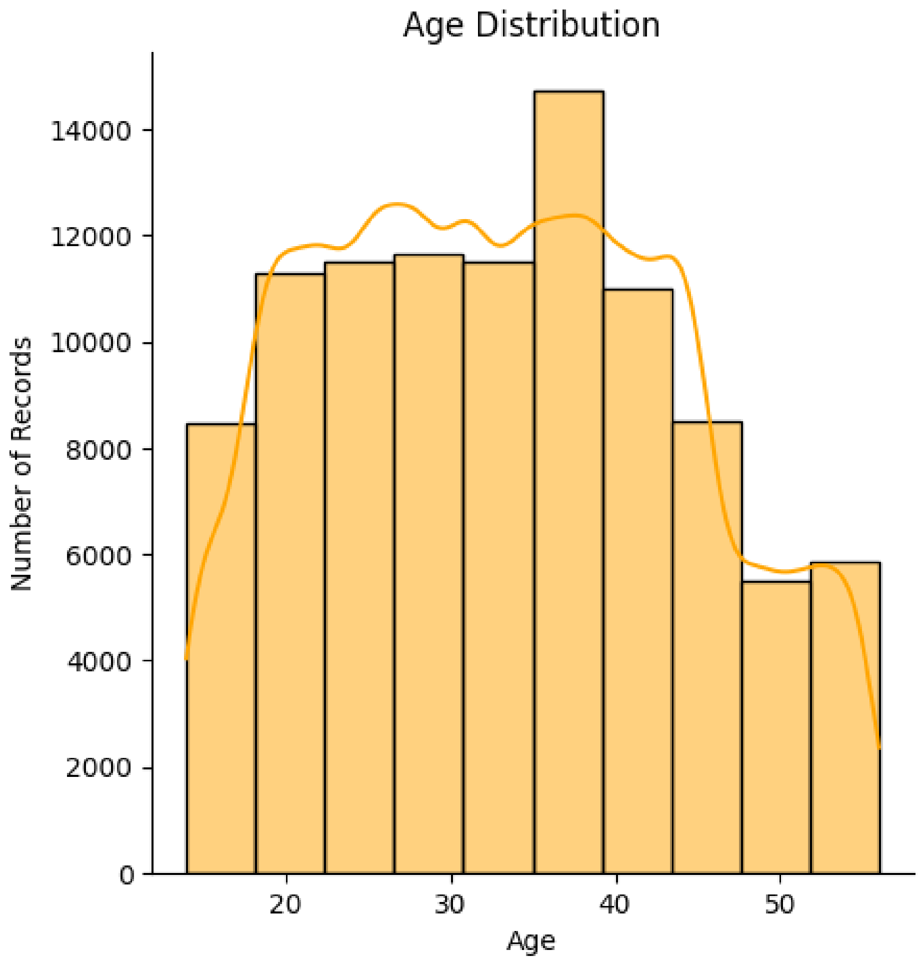
```
[ ]: data['Age'].unique()
```

```
[ ]: array([17, 26, 18, 44, 27, 15, 51, 30, 40, 37, 50, 20, 41, 46, 24, 54, 32,
            38, 14, 43, 22, 55, 45, 29, 48, 35, 39, 25, 19, 36, 21, 31, 42, 23,
            28, 33, 49, 34, 53, 52, 47, 16, 56])
```

```
[ ]: data['Age'].nunique()
```

```
[ ]: 43
```

```
[ ]: sns.displot(data=data, x=data['Age'], kde=True, bins = 10, color = 'orange')
     plt.xlabel('Age')
     plt.ylabel('Number of Records')
     plt.title('Age Distribution')
     plt.xticks(rotation=0)
     plt.show()
```



Summary:

- There were **1788** unique values in Age column which was cleaned up to arrive at **43** unique age values.

Column : SSN

[ ]:
```
data['SSN'].value_counts()
```

[ ]:
```
SSN
#F%$D@*&8      5572
913-74-1218       8
196-69-7786       8
971-11-8511       8
276-64-8276       8
                 …
838-33-4811       4
286-44-9634       4
```

```
753-72-2651       4
331-28-1921       4
604-62-6133       4
Name: count, Length: 12501, dtype: int64
```

[ ]:
```python
data['SSN'] = data['SSN'].str.replace('_', '')
```

[ ]:
```python
def replace_irregular_ssn(group):
  actual_ssn = group.loc[group['SSN'] != '#F%$D@*&8', 'SSN'].iloc[0]
  group_ssn = group.loc[group['SSN'] == '#F%$D@*&8', 'SSN'] = actual_ssn
  return group
data = data.groupby('Customer_ID').apply(replace_irregular_ssn).
 ↪reset_index(drop=True)
```

[ ]:
```python
data['SSN'].value_counts()
```

[ ]:
```
SSN
913-74-1218    8
523-90-6933    8
236-25-0124    8
331-24-3360    8
311-38-7874    8
              ..
360-58-3081    8
341-94-5301    8
702-76-0398    8
282-99-1365    8
832-88-8320    8
Name: count, Length: 12500, dtype: int64
```

[ ]:
```python
data['SSN'].nunique()
```

[ ]:
```
12500
```

[ ]:
```python
data[data['SSN']=='#F%$D@*&8']
```

[ ]:
```
Empty DataFrame
Columns: [ID, Customer_ID, Month, Name, Age, SSN, Occupation, Annual_Income,
Monthly_Inhand_Salary, Num_Bank_Accounts, Num_Credit_Card, Interest_Rate,
Num_of_Loan, Type_of_Loan, Delay_from_due_date, Num_of_Delayed_Payment,
Changed_Credit_Limit, Num_Credit_Inquiries, Credit_Mix, Outstanding_Debt,
Credit_Utilization_Ratio, Credit_History_Age, Payment_of_Min_Amount,
Total_EMI_per_month, Amount_invested_monthly, Payment_Behaviour,
Monthly_Balance]
Index: []

[0 rows x 27 columns]
```

Summary:

- **5572** entries were garbage data
- Now unique values are normalized to **12500** which is equal unique customers in the dataset.

Column: Occupation

[ ]:
```
data['Occupation'].value_counts()
```

[ ]:
```
Occupation
_______          7062
Lawyer           6575
Architect        6355
Engineer         6350
Scientist        6299
Mechanic         6291
Accountant       6271
Developer        6235
Media_Manager    6232
Teacher          6215
Entrepreneur     6174
Doctor           6087
Journalist       6085
Manager          5973
Musician         5911
Writer           5885
Name: count, dtype: int64
```

[ ]:
```
data['Occupation'].str.get_dummies().sum(axis = 1).value_counts()[2:]
```

[ ]:
```
Series([], Name: count, dtype: int64)
```

[ ]:
```
def replace_underscore_occupation(group):
    mode_occupation = group['Occupation'].mode().iloc[0]
    if mode_occupation != '_______':
        group['Occupation'] = group['Occupation'].replace('_______',␣
 ↪mode_occupation)
    else:
        non_underscore_modes = group['Occupation'][group['Occupation'] !=␣
 ↪'_______'].mode()
        if not non_underscore_modes.empty:
            non_underscore_mode = non_underscore_modes.iloc[0]
            group['Occupation'] = group['Occupation'].replace('_______',␣
 ↪non_underscore_mode)

    return group
data = data.groupby('Customer_ID').apply(replace_underscore_occupation).
 ↪reset_index(drop=True)
```

[ ]:
```
data['Occupation'].value_counts()
```

[ ]:
```
Occupation
Lawyer           7096
Engineer         6864
Architect        6824
Mechanic         6776
Accountant       6744
Scientist        6744
Media_Manager    6720
Developer        6720
Teacher          6672
Entrepreneur     6648
Doctor           6568
Journalist       6536
Manager          6432
Musician         6352
Writer           6304
Name: count, dtype: int64
```

[ ]:
```
data['Occupation'].nunique()
```

[ ]:
```
15
```

Summary:

- From the original data, we had **7062** junk values which were cleaned to now have **15** unique Occupation categories.

Column: Annual Income

[ ]:
```
data['Annual_Income'] = data['Annual_Income'].str.replace('_', '')
data['Annual_Income'] = data['Annual_Income'].astype(float)
```

[ ]:
```
data['Annual_Income'].isna().sum()
```

[ ]:
```
0
```

[ ]:
```
data['Annual_Income'] = data.groupby('Customer_ID')['Annual_Income'].
↪transform(lambda x: x.mode().iloc[0])
```

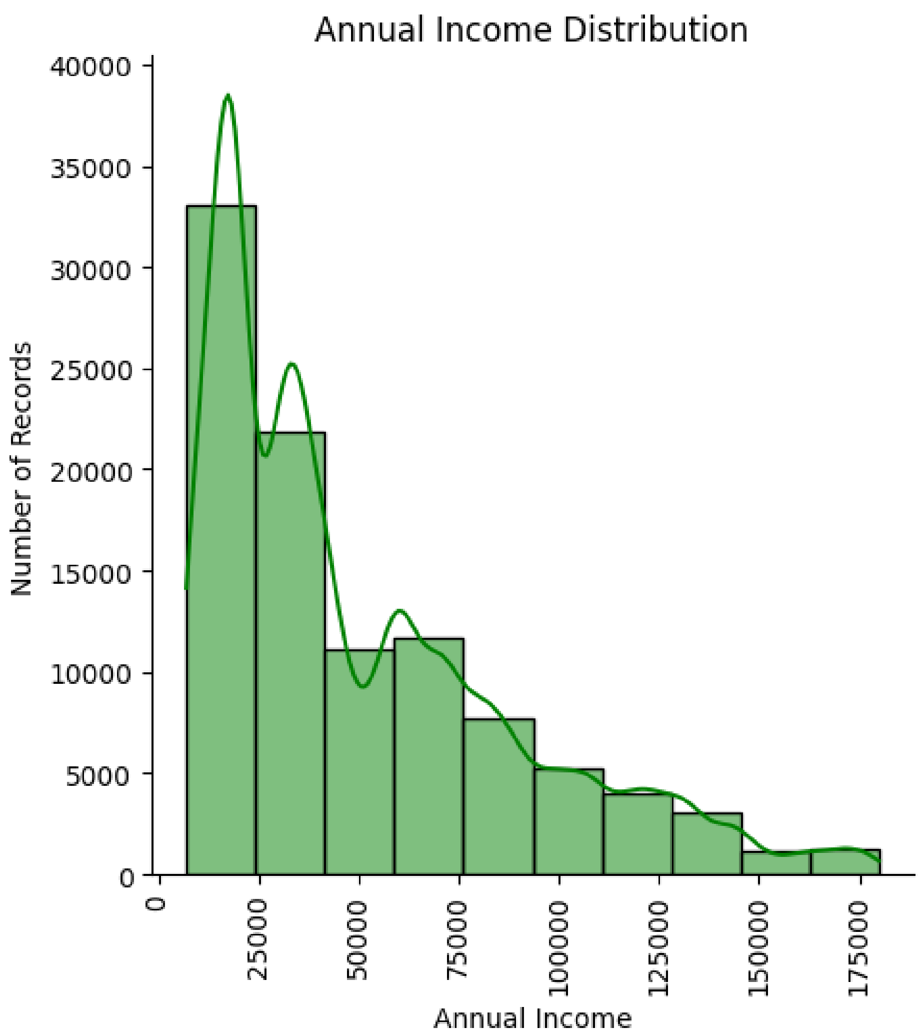
[ ]:
```
data['Annual_Income'].max()
```

[ ]:
```
179987.28
```

[ ]:
```
data['Annual_Income'].min()
```

[ ]:
```
7005.93
```

[ ]:
```
sns.displot(data=data, x=data['Annual_Income'], kde=True, bins=10, color =␣
  ↪'green')
plt.xlabel('Annual Income')
plt.ylabel('Number of Records')
plt.title('Annual Income Distribution')
plt.xticks(rotation= 90)
plt.show()
```



Summary:

- Annual income has been cleaned up. Large number of customers have Annual income between **25 to 50 K**. Distribution of Annual Income is **right skewed**.

Column: Monthly_Inhand_Salary

[ ]:
```
data['Monthly_Inhand_Salary'].value_counts()
```

[ ]:

```
Monthly_Inhand_Salary
6358.956667     15
6082.187500     15
6769.130000     15
2295.058333     15
3080.555000     14
                ..
440.040880       1
9322.687972      1
13102.045570     1
10015.673330     1
8836.177500      1
Name: count, Length: 13235, dtype: int64
```

[ ]:

```
nan_count_by_customer = data.groupby('Customer_ID')['Monthly_Inhand_Salary'].
 ↪apply(lambda x: x.isna().sum())
nan_count_by_customer.value_counts()
```

[ ]:

```
Monthly_Inhand_Salary
1    4862
0    3401
2    2904
3    1048
4     240
5      42
6       3
Name: count, dtype: int64
```

[ ]:

```
data.sort_values(by=['Customer_ID', 'Month'], inplace=True)
data['Monthly_Inhand_Salary'] = data.
 ↪groupby('Customer_ID')['Monthly_Inhand_Salary'].fillna(method='ffill').
 ↪fillna(method='bfill')
```

[ ]:

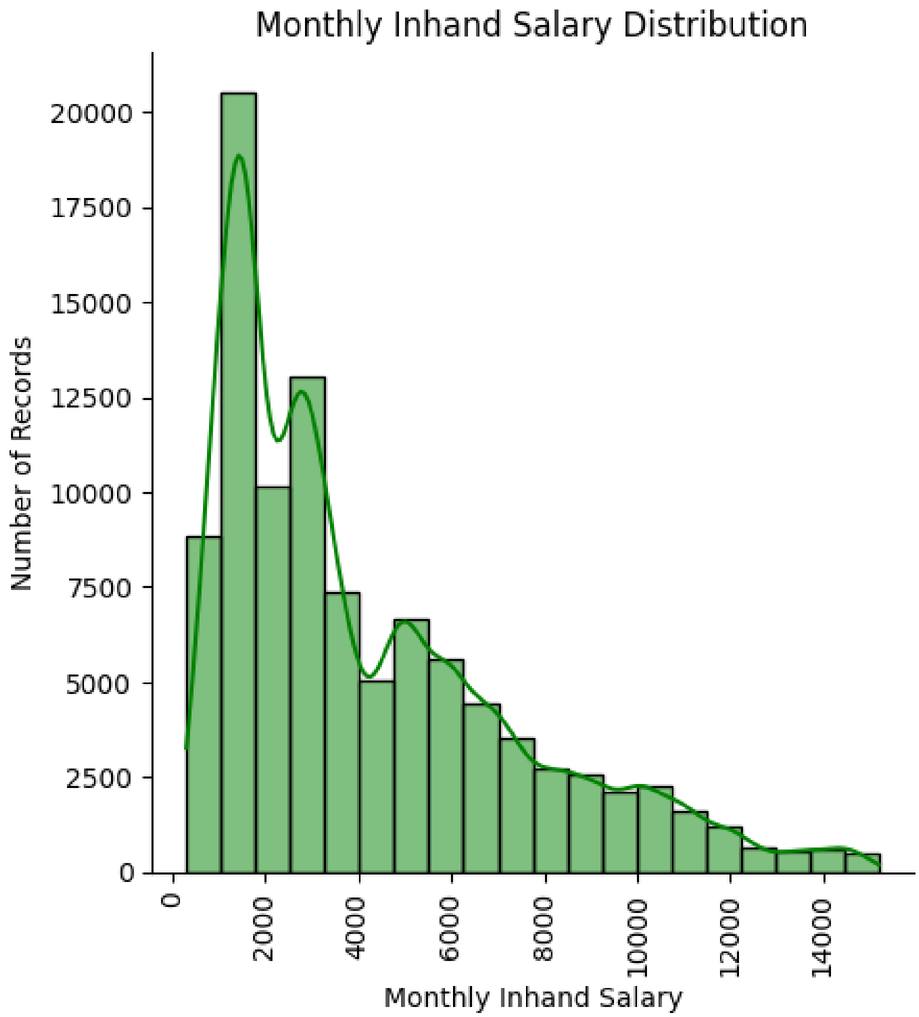
```
data['Monthly_Inhand_Salary'].isna().sum()
```

[ ]:

```
0
```

[ ]:

```
sns.displot(data= data, x= data['Monthly_Inhand_Salary'], kde=True, bins = 20,␣
 ↪color = 'green')
plt.xlabel('Monthly Inhand Salary')
plt.ylabel('Number of Records')
plt.title('Monthly Inhand Salary Distribution')
plt.xticks(rotation=90)
plt.show()
```



Summary:

- **Null values** are filled using **forward fill** and **backfill** function from python. **No outliers** visible in this column and also this data is **right skewed** again and aligned with Annual Income of customer with most of the monthly income is on **lower side** for most of them i.e. between **1.5K to 3.5K**

Column: Num_Bank_Accounts

[ ]:
```
data['Num_Bank_Accounts'].value_counts()
```

[ ]:
```
Num_Bank_Accounts
6        13001
7        12823
8        12765
4        12186
```

```
5         12118
          …
795           1
1252          1
935           1
1350          1
796           1
Name: count, Length: 943, dtype: int64
```

[ ]:
```
grouped_modes = data.groupby('Customer_ID')['Num_Bank_Accounts'].apply(lambda x:
  ↪ x.mode().iloc[0])
data['Num_Bank_Accounts'] = data['Num_Bank_Accounts'].
  ↪mask(data['Num_Bank_Accounts'] != data['Customer_ID'].map(grouped_modes),␣
  ↪data['Customer_ID'].map(grouped_modes))
data['Num_Bank_Accounts'] = data['Num_Bank_Accounts'].apply(lambda x: 1 if x <=␣
  ↪0 else x)
```

[ ]:
```
data['Num_Bank_Accounts'].value_counts().sort_values()
```

[ ]:
```
Num_Bank_Accounts
2      4352
10     5328
9      5512
1      8952
3     12096
5     12272
4     12392
8     12936
7     12976
6     13184
Name: count, dtype: int64
```

[ ]:
```
data['Num_Bank_Accounts'].nunique()
```

[ ]:
```
10
```

[ ]:
```
sns.countplot(data= data, x= data['Num_Bank_Accounts'], color = 'purple')
plt.xlabel('Number of Bank Accounts')
plt.ylabel('Number of Records')
plt.title('Bank Accounts Distribution')
plt.xticks(rotation=0)
plt.show()
```

Summary:

- Original data had **some outliers**, **negative values** which were cleaned up. Total no. of unique values for Number of Bank accounts are **10**. Majority of customers had Number of accounts between **3-8**.

Column: Num_Credit_Card

[ ]:
```
data['Num_Credit_Card'].value_counts()
```

[ ]:
```
Num_Credit_Card
5          18459
7          16615
6          16559
4          14030
3          13277
           …
422            1
62             1
1348           1
819            1
```

```
1108        1
Name: count, Length: 1179, dtype: int64
```

[ ]:
```
grouped_modes = data.groupby('Customer_ID')['Num_Credit_Card'].apply(lambda x:␣
↪x.mode().iloc[0])
data['Num_Credit_Card'] = data.apply(lambda row:␣
↪grouped_modes[row['Customer_ID']] if row['Num_Credit_Card'] !=␣
↪grouped_modes[row['Customer_ID']] else row['Num_Credit_Card'], axis=1)
```

[ ]:
```
data['Num_Credit_Card'].value_counts().sort_values(ascending=True)
```

[ ]:
```
Num_Credit_Card
0        16
11       40
1      2184
2      2208
9      4736
10     4960
8      5096
3     13576
4     14336
6     16960
7     16984
5     18904
Name: count, dtype: int64
```

[ ]:
```
sns.countplot(data=data, x=data['Num_Credit_Card'], color = "pink")
plt.xlabel('Number of Credit_Cards')
plt.ylabel('Number of Records')
plt.title('Credit Card Distribution')
plt.xticks(rotation=90)
plt.show()
```

Summary:

- There were outliers in the data with **1179** unique values which was cleaned up to arrive at **12 unique categories**. Maximum number of customers fall into holding **3-7** credit cards.

Column: Interest_Rate

[ ]:
```
data['Interest_Rate'].value_counts()
```

[ ]:
```
Interest_Rate
8         5012
5         4979
6         4721
12        4540
10        4540
          …
295          1
3395         1
4323         1
3225         1
3808         1
```

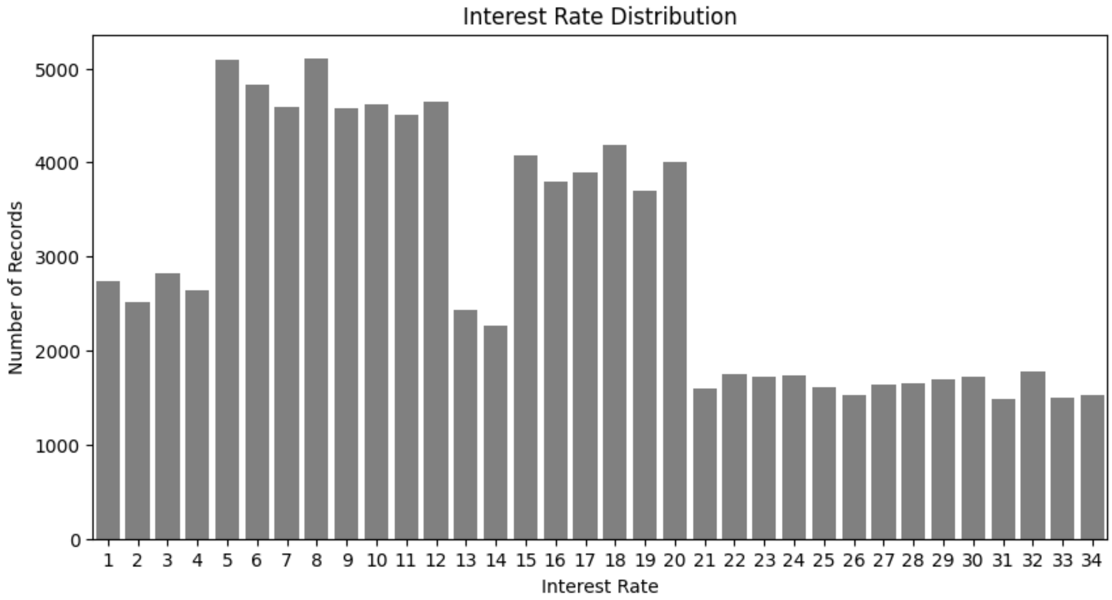
```
Name: count, Length: 1750, dtype: int64
```

[ ]:
```python
grouped_modes = data.groupby('Customer_ID')['Interest_Rate'].apply(lambda x: x.
 ↪mode().iloc[0])
data['Interest_Rate'] = data.apply(lambda row:␣
 ↪grouped_modes[row['Customer_ID']] if row['Interest_Rate'] !=␣
 ↪grouped_modes[row['Customer_ID']] else row['Interest_Rate'], axis=1)
```

[ ]:
```python
data['Interest_Rate'].value_counts().sort_values(ascending=True)
```

[ ]:
```
Interest_Rate
31    1488
33    1496
34    1528
26    1528
21    1592
25    1608
27    1640
28    1648
29    1696
23    1720
30    1728
24    1736
22    1752
32    1776
14    2272
13    2432
2     2520
4     2640
1     2744
3     2824
19    3704
16    3800
17    3888
20    4008
15    4072
18    4192
11    4512
9     4576
7     4584
10    4616
12    4648
6     4832
5     5096
8     5104
Name: count, dtype: int64
```

[ ]:
```
plt.figure(figsize=(10,5))
sns.countplot(data= data, x= data['Interest_Rate'], color = "grey")
plt.xlabel('Interest Rate')
plt.ylabel('Number of Records')
plt.title('Interest Rate Distribution')
plt.xticks(rotation=0)
plt.show()
```



Summary:

- There were **outliers** in the data which upon clean-up gave **Interest Rates in the range of 1% - 34%**.

Column: Num_of_Loan

[ ]:
```
data['Num_of_Loan'].unique()
```

[ ]:
```
array(['2', '1094', '4', '4_', '0', '0_', '3', '8', '-100', '8_', '1',
       '1_', '9', '7', '1222', '6', '5', '119', '3_', '6_', '2_', '9_',
       '143_', '7_', '5_', '1150', '351', '52', '95', '614', '504',
       '1241', '1496', '17', '966', '330', '290', '193', '520', '50',
       '1265', '352', '571', '190', '995', '55', '433', '590', '661',
       '313', '1027_', '92_', '1017', '904', '1132_', '1008', '49', '737',
       '546', '1096', '1461', '548', '939', '243', '1014', '924', '526',
       '1447', '1228', '1129', '968', '285', '1484', '716', '1236', '801',
       '809', '137', '208', '875', '1187', '621', '350', '911', '1023',
       '855', '802', '967', '1296', '640', '1131_', '639', '1365', '254',
       '1040', '141', '349', '659', '1480', '1259', '889', '70', '344',
```

```
       '898', '41', '1412', '1353', '720', '1154', '295', '238', '100',
       '54', '237', '868', '1214', '873', '33', '895', '1482', '1384',
       '182', '1289', '439', '563', '31', '597', '649', '1053', '1036',
       '1457', '814', '484', '1359', '252', '282', '945', '65', '781',
       '905', '545', '684', '1400', '1035', '84', '372', '143', '733',
       '103', '58', '251', '27_', '848', '652', '1416', '999', '1451',
       '996', '527', '773', '302', '18', '392', '1294', '910', '628',
       '430', '404', '728', '799', '745', '1217', '515', '147', '1135',
       '449', '1474', '697', '1297', '1307', '123', '1106', '1463',
       '1219_', '1433', '191', '501', '464', '654', '1320_', '438', '510',
       '860', '891', '132', '638', '138', '926', '753', '267', '606',
       '983', '1406', '1345', '841', '816', '663', '1439', '323', '1137',
       '1103', '56', '164', '437', '89', '201', '23', '1391', '1181',
       '348', '686', '1015', '341', '1348', '1329', '1182', '148', '529',
       '527_', '231', '1196', '1464', '562', '1152', '622', '955', '1470',
       '336', '447', '897', '1257', '752', '1225', '679', '288', '943',
       '1459_', '1210', '29', '1227', '1372', '1085', '235_', '1048',
       '291', '1319', '1039', '227_', '834', '1001', '153', '629', '1019',
       '1369', '1393', '778', '742', '613', '1318', '936', '316', '1444',
       '1151', '931', '1204', '172', '635', '311', '1209', '831', '1030',
       '229', '1054', '444', '832', '394', '1127', '1091', '1002', '462',
       '1387', '1363', '1088', '1279', '1419', '843', '1112', '87', '917',
       '833', '280', '581', '859', '952', '596', '1216', '378_', '1313',
       '1430', '1185_', '174', '275', '497', '284', '630_', '198', '1495',
       '1311_', '1441', '274', '540', '601', '935', '216', '719', '332',
       '1160', '32', '192', '1354', '1312', '1225_', '838', '242', '329',
       '1110', '1340', '958', '701', '1047', '387', '820', '579', '1202',
       '186', '636', '1371', '961', '126', '940', '157', '1382', '101',
       '1320', '241', '1424', '863', '1300', '1302', '1159', '819', '507',
       '696', '217', '538', '463', '1478', '321', '196', '466', '633',
       '289', '146', '785_', '359', '1465', '867', '662', '574', '1298',
       '1077', '494', '1171_', '1485', '455', '136', '39', '300', '1271',
       '1347_', '424', '1131', '131_', '699', '365', '19', '415', '869',
       '227', '657', '1046', '1178', '777', '359_', '292', '228', '492',
       '420', '1274', '416', '927', '78', '215', '457', '1006', '1189',
       '83', '795', '881', '405', '757', '978', '319', '597_', '1129_',
       '1074', '1070', '696_', '991', '653', '617', '656', '418', '472'],
      dtype=object)
```

[ ]:
```python
data['Num_of_Loan'] = data['Num_of_Loan'].str.replace('_', '')
data['Num_of_Loan'] = data['Num_of_Loan'].str.replace('-', '')
data['Num_of_Loan'] = data['Num_of_Loan'].astype(int)
```

[ ]:
```python
data['Num_of_Loan'].value_counts()
```

[ ]:
```
Num_of_Loan
3       15104
```

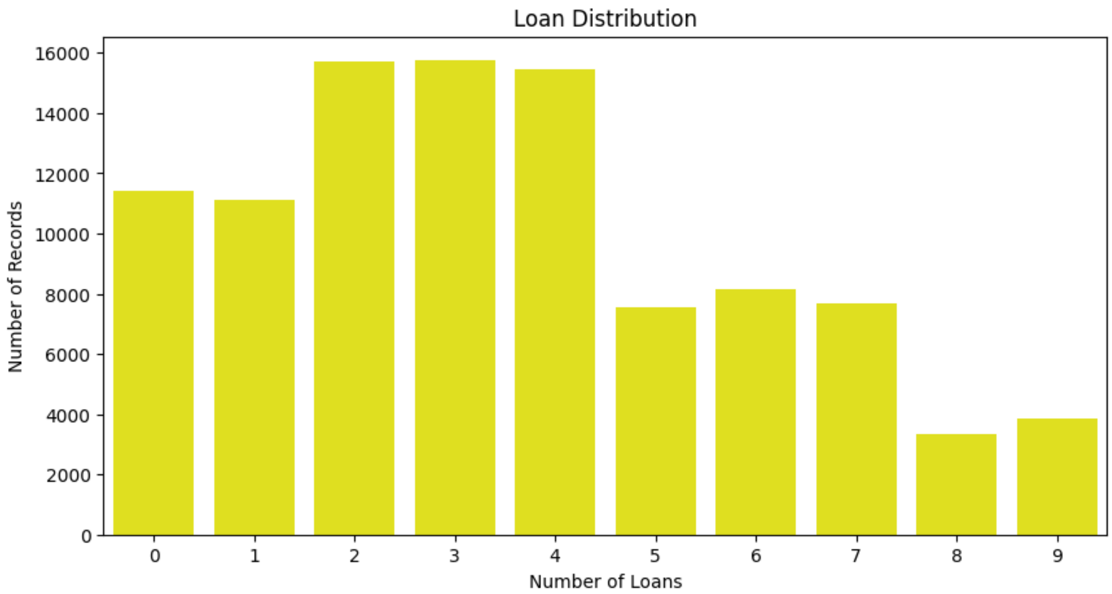
```
2       15032
4       14743
0       10930
1       10606
        …
860         1
510         1
438         1
571         1
472         1
Name: count, Length: 413, dtype: int64
```

[ ]:
```
grouped_modes = data.groupby('Customer_ID')['Num_of_Loan'].apply(lambda x: x.
 ↪mode().iloc[0])
data['Num_of_Loan'] = data.apply(lambda row: grouped_modes[row['Customer_ID']]␣
 ↪if row['Num_of_Loan'] != grouped_modes[row['Customer_ID']] else␣
 ↪row['Num_of_Loan'], axis=1)
```

[ ]:
```
data['Num_of_Loan'].value_counts()
```

[ ]:
```
Num_of_Loan
3    15752
2    15712
4    15456
0    11408
1    11128
6     8144
7     7680
5     7528
9     3856
8     3336
Name: count, dtype: int64
```

[ ]:
```
plt.figure(figsize=(10,5))
sns.countplot(data=data, x=data['Num_of_Loan'], color = 'yellow')
plt.xlabel('Number of Loans')
plt.ylabel('Number of Records')
plt.title('Loan Distribution')
plt.xticks(rotation=0)
plt.show()
```



Column: Type_of_Loan

[ ]:
```
data['Type_of_Loan'].value_counts()
```

[ ]:
```
Type_of_Loan
Not Specified
12816
Credit-Builder Loan
1280
Personal Loan
1272
Debt Consolidation Loan
1264
Student Loan
1240
           …
Home Equity Loan, Payday Loan, Credit-Builder Loan, Not Specified, and Home
Equity Loan        8
Personal Loan, Student Loan, Personal Loan, and Home Equity Loan
8
Home Equity Loan, Payday Loan, Not Specified, and Home Equity Loan
8
Home Equity Loan, Mortgage Loan, and Payday Loan
8
Auto Loan, Payday Loan, Payday Loan, Mortgage Loan, Payday Loan, and Home Equity
Loan          8
Name: count, Length: 6260, dtype: int64
```

[ ]:
```python
data['Type_of_Loan'].isna().sum()
```

[ ]:
```
11408
```

[ ]:
```python
filter_data = data[pd.isna(data['Type_of_Loan'])]
```

[ ]:
```python
filter_data[['Customer_ID','Num_of_Loan','Num_Credit_Card','Type_of_Loan']]
```

[ ]:
```
       Customer_ID  Num_of_Loan  Num_Credit_Card Type_of_Loan
16      CUS_0x100b            0                4          NaN
17      CUS_0x100b            0                4          NaN
18      CUS_0x100b            0                4          NaN
19      CUS_0x100b            0                4          NaN
20      CUS_0x100b            0                4          NaN
…             …           …                …          …
99947   CUS_0xfe5            0                4          NaN
99948   CUS_0xfe5            0                4          NaN
99949   CUS_0xfe5            0                4          NaN
99950   CUS_0xfe5            0                4          NaN
99951   CUS_0xfe5            0                4          NaN

[11408 rows x 4 columns]
```

[ ]:
```python
data.loc[(data['Num_of_Loan'] == 0) & (data['Num_Credit_Card'] > 0),␣
↪'Type_of_Loan'] = data['Type_of_Loan'].fillna('Not Specified')
```

[ ]:
```python
data.loc[(data['Num_of_Loan'] == 0) & (data['Num_Credit_Card'] == 0) &␣
↪(data['Total_EMI_per_month'] == 0), 'Type_of_Loan'] = 'Not Specified'
```

[ ]:
```python
loan_types = data['Type_of_Loan'].str.replace('and', ',').str.get_dummies(', ')

# Concatenate the new columns with the original DataFrame
data  = pd.concat([data, loan_types], axis=1)
```

Used 1-hot coding to convert these columns

9 types of Loans:

- Auto Loan
- Credit-Builder Loan
- Debt consolidation Loan
- Home equity Loan
- Mortgage Loan
- Payday Loan
- Student Loan
- Personal Loan
- Not Specified

```
[ ]: col_order = ['ID', 'Customer_ID', 'Month', 'Name', 'Age', 'SSN', 'Occupation',
            'Annual_Income', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts',
            'Num_Credit_Card', 'Interest_Rate', 'Num_of_Loan', 'Type_of_Loan','Auto␣
      ↪Loan',
            'Credit-Builder Loan', 'Debt Consolidation Loan', 'Home Equity Loan',
            'Mortgage Loan', 'Not Specified', 'Payday Loan', 'Personal Loan',
            'Student Loan','Delay_from_due_date', 'Num_of_Delayed_Payment',␣
      ↪'Changed_Credit_Limit',
            'Num_Credit_Inquiries', 'Credit_Mix', 'Outstanding_Debt',
            'Credit_Utilization_Ratio', 'Credit_History_Age',
            'Payment_of_Min_Amount', 'Total_EMI_per_month',
            'Amount_invested_monthly', 'Payment_Behaviour', 'Monthly_Balance']
```

```
[ ]: data = data[col_order]
```

```
[ ]: data['Type_of_Loan'].isna().sum()
```

```
[ ]: 0
```

Column : Delay_from_due_date

```
[ ]: data['Delay_from_due_date'].unique()
```

```
[ ]: array([64, 57, 62, 67, 10,  5,  8,  3, 14, 19,  9, 27, 29, 12, 16,  6, 24,
             0, -4, -5,  1, 15, 23, 28, 18, 13, 11, 25, 50, 47, 48, 46,  7,  2,
            -3,  4, 30, 21, 17, 20, 22, 35, 40, 26, 31, 58, 59, 63, 37, 42, 43,
            38, 55, 41, 36, 52, 54, 53, 49, -2, 44, 39, 61, 34, 33, -1, 45, 51,
            60, 66, 56, 32, 65])
```

```
[ ]: data['Delay_from_due_date'].max()
```

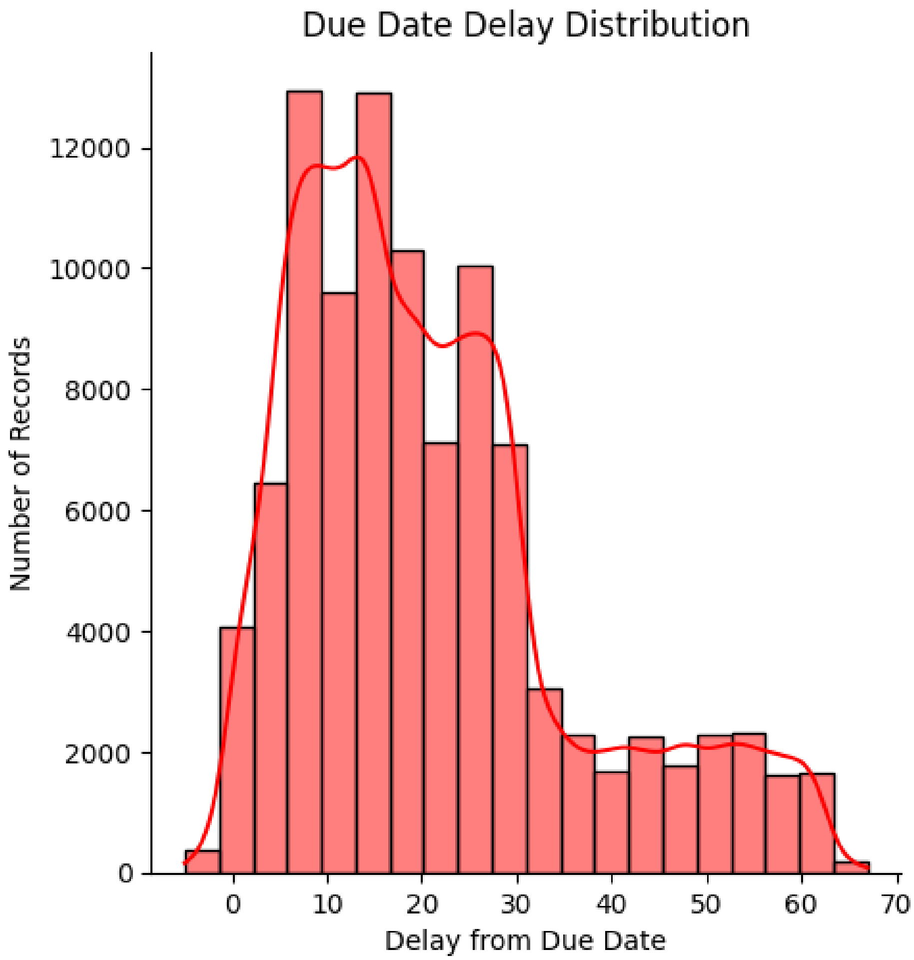
```
[ ]: 67
```

```
[ ]: data['Delay_from_due_date'].min()
```

```
[ ]: -5
```

```
[ ]: plt.figure(figsize=(10,5))
     sns.displot(data = data, x= data['Delay_from_due_date'], kde= True, bins = 20,␣
      ↪color = "red")
     plt.xlabel('Delay from Due Date')
     plt.ylabel('Number of Records')
     plt.title('Due Date Delay Distribution')
     plt.xticks(rotation=0)
     plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```



Summary:

- Delay from due date ranges from **5 day prior** to **67 days delayed**

Column : Num_of_Delayed_Payment

[ ]:
```
data['Num_of_Delayed_Payment'].isna().sum()
```

[ ]: 7002

[ ]:
```
data['Num_of_Delayed_Payment'].unique()
```

[ ]:
```
array(['25', '26', '23', '28', '18', '16', '1749', '19', '7', '8', '9',
       '15', '13', nan, '12', '17_', '10', '20', '22', '1', '5', '2',
       '11', '17', '15_', '14', '3', '4', '6', '21', '8_', '11_', '0',
       '2230', '24', '18_', '-2', '19_', '1636', '20_', '-1', '16_',
       '921', '9_', '1766', '21_', '12_', '6_', '1_', '25_', '0_', '-3',
       '1572', '5_', '14_', '3_', '3162', '27', '1034', '4211', '4_',
```

'2712', '1832', '22_', '3251', '7_', '867', '13_', '4106', '3951',
'2216', '24_', '10_', '2_', '1640', '2142_', '754', '974', '1180',
'1359', '320', '2250', '3621', '2438', '531', '3738', '2566',
'719', '4326', '223', '1833', '3881', '23_', '439', '1614', '3495',
'960', '4075', '3119', '4302', '121', '2081', '3894', '3484',
'2594', '4126', '3944', '2553', '1820', '819', '27_', '3629',
'2080', '1480', '2801', '359', '94', '473', '2072', '2604', '306',
'1633', '4262', '2488', '2008', '2955', '1647', '1691', '468',
'1150', '3491', '4178', '1215', '3793', '3623', '2672', '2508',
'1867', '4340', '1862', '1282', '1422', '441', '1204', '519',
'2938', '371', '594', '663_', '46', '3458', '2658', '4134', '2907',
'2047', '4011', '2991', '4319', '674', '4216', '2671', '-2_',
'2323', '271', '2184', '2628', '2381', '3191', '2376', '2260',
'4005', '426', '399', '337', '3069', '3156', '4231', '1750', '372',
'2378', '876', '2279', '3545', '1222', '3764', '1663', '3200',
'1890', '2728', '4069', '559', '1598', '3316', '2753', '1687',
'281', '84', '4047', '1354', '4135', '2533', '2018', '708', '1509',
'4360', '3726', '1825', '1864', '3112', '1329', '-3_', '733',
'1765', '775', '3684', '3212', '3478', '2400', '4278', '3636',
'871', '3946', '3900', '2534', '49', '26_', '197', '1295_', '1841',
'1478', '4172', '2638', '3972', '1211', '905', '1699', '2324',
'1325', '1706', '2056', '2903', '2569', '4293', '2621', '2924',
'1792', '1338', '3107', '430', '714', '2015', '2879', '1673',
'4024', '415', '2569_', '-1_', '1900', '1852', '2945', '4249',
'195', '2280', '132', '384', '3148', '642', '3539', '3905', '3171',
'3050', '1911', '804', '2493', '85', '1463', '3208', '3031',
'2560', '1795', '1664', '3739', '1481', '3861_', '1172', '1014',
'1106', '4219', '3751', '3051', '1989', '2149', '1323_', '739',
'47', '1735', '2255', '1263', '1718', '2566_', '4002', '4295',
'1402', '1086', '3329', '2873', '4113', '3037', '848_', '813',
'2413', '2521', '2142', '926', '3707', '210', '2348', '3216',
'1450', '2021', '2766', '3340', '3447', '1328', '2913', '615',
'4241', '3313', '1994', '2420', '532', '538', '1411', '2511',
'3529', '4169', '107', '1191', '2823', '283', '3580', '2354',
'3765', '1332', '1530', '3926', '3706', '3099', '3790', '1850',
'2131', '2697', '2239', '162', '2590', '904', '1370', '847',
'3103', '3661', '1216', '544', '1985', '4185', '3502', '3533',
'106', '3368', '1301', '853', '3840_', '4191', '523', '3318',
'2128', '1015', '4022', '4280', '585', '2578', '3819', '972',
'602', '2060', '2278', '264', '3845', '1502', '221', '3688',
'1154', '1473', '666', '3920_', '2237_', '1243', '1976', '1192',
'450', '1552', '1278', '3097_', '851', '3040', '2127', '1685',
'4096', '4042', '1511', '1523', '3815', '3855', '4161', '133',
'3750', '252', '2397', '217', '88', '2529', '309', '273', '2286',
'1079', '2694', '166', '3632', '1443', '1199', '4107', '2875',
'834', '808', '2429', '3457', '2219_', '577', '3721', '3011',
'2729', '2492', '4282', '182', '3858', '1743', '2615', '3092',

```
       '2950', '3536', '3355', '1823', '238', '2943', '4077', '4095',
       '3865', '1861', '3708', '183_', '1184', '846', '709', '4239',
       '2926', '1087_', '2707', '4159', '1371', '3142', '2882', '787',
       '3392', '2793', '3568', '845', '1975', '1073', '3919', '3909',
       '2334', '640', '1541', '2759', '4023', '2751', '1471', '1256',
       '2657', '2274', '1096', '3009', '1164', '3155', '2148', '2737',
       '86', '3522', '4281', '2523', '3489', '3177', '3154', '3415',
       '1606', '1967', '3864', '3300', '1392', '1869', '1177', '3407',
       '887', '145', '4144', '4384', '969', '3499', '2854', '1538',
       '3559', '3402', '2666', '1004', '2705', '2314', '2138', '3754',
       '583', '98', '2044', '1697', '2959', '3722', '933', '4051', '2655',
       '1849', '2689', '3222', '2552', '2794_', '2006', '829', '1063',
       '28_', '2162', '3105', '1045', '1859', '4397', '1337', '3060',
       '3467', '683', '2677', '938', '2956', '1389', '1653', '351', '693',
       '3243', '1941', '2165', '2070', '4270', '2141', '4019', '3260',
       '2461', '3404', '2007', '2616', '482', '3268', '398', '1571',
       '3488', '2617', '2810', '2311', '700', '2756', '1181', '2896',
       '4128', '3083', '3078', '416', '2503', '1473_', '2506', '742',
       '3229', '3253', '4053', '1553', '1236', '2591', '1732', '707',
       '4164', '411', '4292', '3115', '749', '2185', '1946', '3584',
       '1953', '3978', '541', '3827', '809', '142', '2276', '2317',
       '3749', '2587', '2636', '3416', '3370', '3766', '2278_', '4311',
       '1489', '130', '294', '827', '3796', '1801', '1218', '4059',
       '2768', '4266', '1579', '1952', '2457', '3179', '290', '2589',
       '1608', '2196', '2820', '2418', '3245', '2076', '2573', '1133',
       '2812', '2498', '1668', '2777', '3870', '186', '2860', '2609',
       '3955', '2300', '2570', '508', '793', '1954', '211', '80', '1775',
       '676', '1049', '2384', '1891', '102', '4344', '1061', '1879',
       '3574', '662', '529', '3043', '2834', '3104', '1060', '929',
       '2297', '659', '2262', '3878', '4324', '3336', '4388', '2450',
       '3511', '3763', '4251', '192', '3960', '4043', '1996', '1178',
       '2660', '3776', '3660', '1874', '1534', '3175', '2645', '4139',
       '996', '2351', '2352', '2001', '3880', '1018', '758_', '4337',
       '3869', '823', '2544', '2585', '497', '3274', '3456', '2385',
       '196', '923', '2431', '3010', '2243', '1884', '778', '175', '2167',
       '2222', '1531', '72', '265', '2954', '800', '3847', '779', '4037',
       '3391', '4298', '2919', '3492', '52', '1498', '328', '1536',
       '2204', '1087'], dtype=object)
```

[ ]:
```python
data['Num_of_Delayed_Payment'] = data['Num_of_Delayed_Payment'].str.
 ↪replace('_', '')
data['Num_of_Delayed_Payment'] = data['Num_of_Delayed_Payment'].str.
 ↪replace('-', '')
data['Num_of_Delayed_Payment'] = data['Num_of_Delayed_Payment'].astype(float)
```

[ ]:
```python
data['Num_of_Delayed_Payment'].value_counts().sort_values(ascending=True)
```

[ ]:
```
Num_of_Delayed_Payment
1668.0       1
2658.0       1
3458.0       1
439.0        1
531.0        1
          …
15.0      5237
10.0      5309
16.0      5312
17.0      5412
19.0      5481
Name: count, Length: 708, dtype: int64
```

[ ]:
```python
data1 = data[pd.isna(data['Num_of_Delayed_Payment'])]
data1[['Customer_ID','Num_of_Loan','Num_Credit_Card','Num_of_Delayed_Payment']]
```

[ ]:
```
      Customer_ID  Num_of_Loan  Num_Credit_Card  Num_of_Delayed_Payment
26     CUS_0x1011            3                3                     NaN
31     CUS_0x1011            3                3                     NaN
33     CUS_0x1013            3                3                     NaN
55     CUS_0x1018            8                7                     NaN
66     CUS_0x102d            1                3                     NaN
…             …          …              …                     …
99935   CUS_0xfe3            4                5                     NaN
99937   CUS_0xfe4            7                3                     NaN
99942   CUS_0xfe4            7                3                     NaN
99980   CUS_0xff6            2                6                     NaN
99999   CUS_0xffd            6                7                     NaN

[7002 rows x 4 columns]
```

[ ]:
```python
grouped_modes = data.groupby('Customer_ID')['Num_of_Delayed_Payment'].
 ↪apply(lambda x: x.mode().iloc[0])
data['Num_of_Delayed_Payment'] = data.apply(lambda row:␣
 ↪grouped_modes[row['Customer_ID']] if row['Num_of_Delayed_Payment'] !=␣
 ↪grouped_modes[row['Customer_ID']] else row['Num_of_Delayed_Payment'], axis=1)
```

[ ]:
```python
grouped_modes
```

[ ]:
```
Customer_ID
CUS_0x1000    25.0
CUS_0x1009    18.0
CUS_0x100b     7.0
CUS_0x1011    15.0
CUS_0x1013     9.0
              …
```

```
CUS_0xff3      9.0
CUS_0xff4     10.0
CUS_0xff6      4.0
CUS_0xffc     16.0
CUS_0xffd     12.0
Name: Num_of_Delayed_Payment, Length: 12500, dtype: float64
```

[ ]:
```
data['Num_of_Delayed_Payment'].value_counts()
```

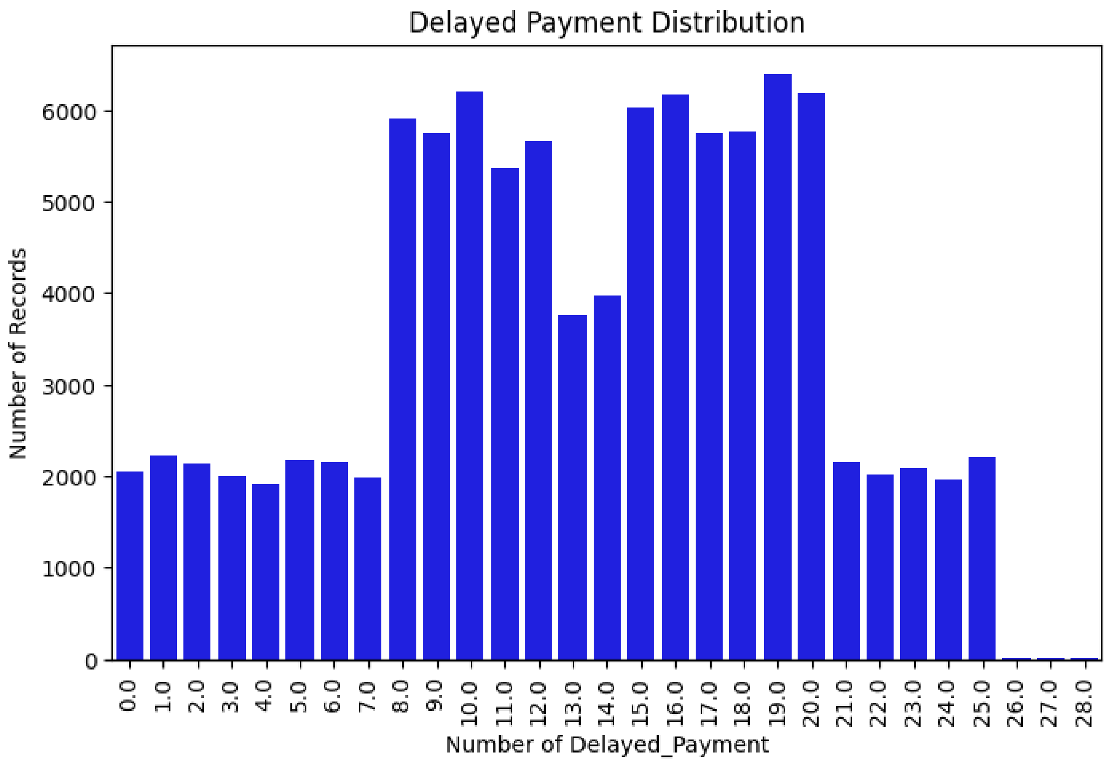
[ ]:
```
Num_of_Delayed_Payment
19.0    6392
10.0    6200
20.0    6184
16.0    6160
15.0    6032
8.0     5904
18.0    5760
17.0    5752
9.0     5744
12.0    5664
11.0    5368
14.0    3976
13.0    3752
1.0     2232
25.0    2208
5.0     2176
6.0     2160
21.0    2152
2.0     2136
23.0    2088
0.0     2056
22.0    2024
3.0     2000
7.0     1976
24.0    1968
4.0     1912
27.0       8
28.0       8
26.0       8
Name: count, dtype: int64
```

[ ]:
```
data['Num_of_Delayed_Payment'].isna().sum()
```

[ ]:
```
0
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.countplot(data = data, x= data['Num_of_Delayed_Payment'], color = "blue")
```

```
plt.xlabel('Number of Delayed_Payment')
plt.ylabel('Number of Records')
plt.title('Delayed Payment Distribution')
plt.xticks(rotation=90)
plt.show()
```



Summary:

Delayed Payments frequency is high between **9 to 20 intervals**

Column : Changed_Credit_Limit

[ ]:
```
data['Changed_Credit_Limit'].dtypes
```

[ ]: dtype('O')

[ ]:
```
data['Changed_Credit_Limit'].value_counts()
```

[ ]:
```
Changed_Credit_Limit
_           2091
8.22         135
11.5         127
11.32        126
7.35         121
```

```
         …
-2.02        1
35.84        1
-4.88        1
-3.49        1
33.61        1
Name: count, Length: 3635, dtype: int64
```

[ ]:
```
data['Changed_Credit_Limit'] = data['Changed_Credit_Limit'].str.replace('_', '')
data['Changed_Credit_Limit'] = data['Changed_Credit_Limit'].str.replace('-', '')
data['Changed_Credit_Limit'] = data['Changed_Credit_Limit'].replace('', '0')
data['Changed_Credit_Limit'] = data['Changed_Credit_Limit'].astype(float)
```

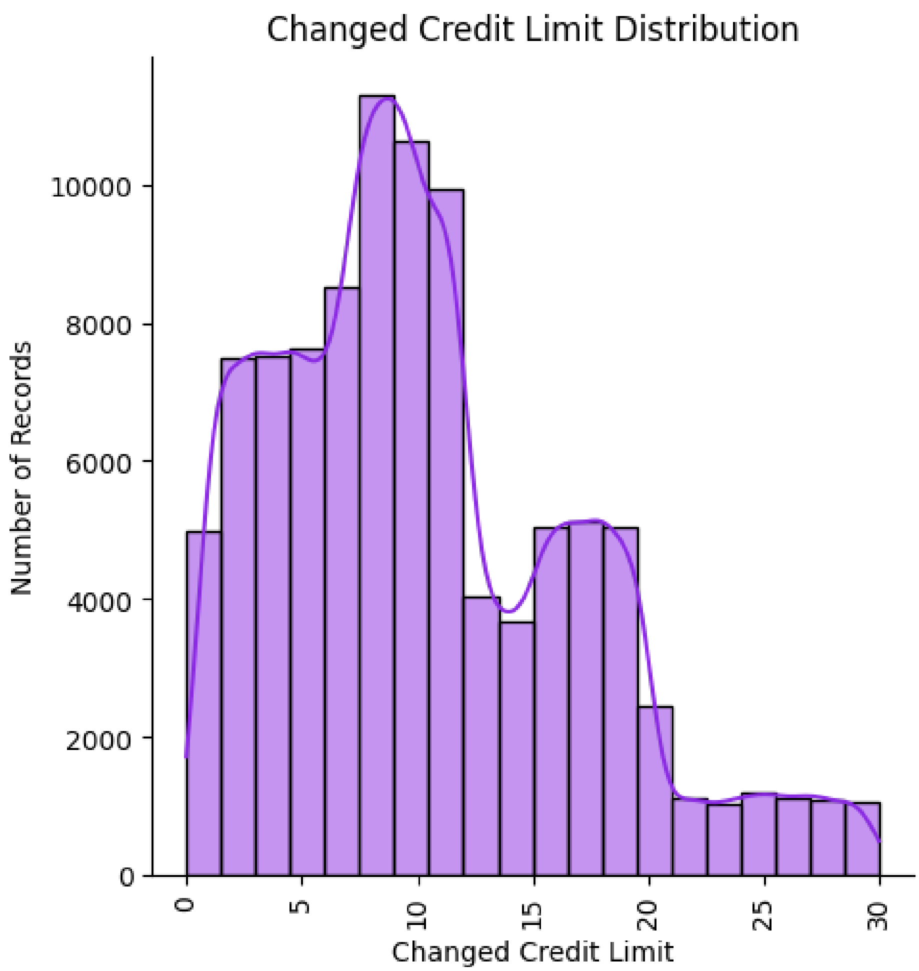
[ ]:
```
grouped_modes = data.groupby('Customer_ID')['Changed_Credit_Limit'].
  ↪apply(lambda x: x.mode().iloc[0])
data['Changed_Credit_Limit'] = data.apply(lambda row:␣
  ↪grouped_modes[row['Customer_ID']] if row['Changed_Credit_Limit'] !=␣
  ↪grouped_modes[row['Customer_ID']] else row['Changed_Credit_Limit'], axis=1)
```

[ ]:
```
data['Changed_Credit_Limit'].value_counts()
```

[ ]:
```
Changed_Credit_Limit
8.22     152
11.50    152
11.32    144
7.69     136
7.35     136
        …
21.87      8
29.90      8
26.06      8
29.14      8
23.16      8
Name: count, Length: 2521, dtype: int64
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.displot(data= data, x=data['Changed_Credit_Limit'], kde=True, bins=20,␣
  ↪color = "blueviolet")
plt.xlabel('Changed Credit Limit')
plt.ylabel('Number of Records')
plt.title('Changed Credit Limit Distribution')
plt.xticks(rotation=90)
plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```



Column : Num_Credit_inquiries

```
[ ]: data['Num_Credit_Inquiries'].isna().sum()
```

```
[ ]: 1965
```

```
[ ]: data['Num_Credit_Inquiries'].value_counts()
```

```
[ ]: Num_Credit_Inquiries
     4.0       11271
     3.0        8890
     6.0        8111
     7.0        8058
     2.0        8028
               …
     253.0         1
     2352.0        1
```

```
2261.0        1
519.0         1
1801.0        1
Name: count, Length: 1223, dtype: int64
```

[ ]:
```python
data2 = data[pd.isna(data['Num_Credit_Inquiries'])]
data2[['Customer_ID','Num_of_Loan','Num_Credit_Card','Num_Credit_Inquiries']]
```

[ ]:

	Customer_ID	Num_of_Loan	Num_Credit_Card	Num_Credit_Inquiries
55	CUS_0x1018	8	7	NaN
118	CUS_0x1041	9	8	NaN
161	CUS_0x1051	1	5	NaN
190	CUS_0x105b	0	4	NaN
235	CUS_0x107c	6	10	NaN
…	…	…	…	…
99847	CUS_0xfb4	4	6	NaN
99968	CUS_0xff4	5	7	NaN
99970	CUS_0xff4	5	7	NaN
99979	CUS_0xff6	2	6	NaN
99994	CUS_0xffd	6	7	NaN

[1965 rows x 4 columns]

[ ]:
```python
grouped_modes = data.groupby('Customer_ID')['Num_Credit_Inquiries'].
 ↪apply(lambda x: x.mode().iloc[0])
data['Num_Credit_Inquiries'] = data.apply(lambda row:␣
 ↪grouped_modes[row['Customer_ID']] if row['Num_Credit_Inquiries'] !=␣
 ↪grouped_modes[row['Customer_ID']] else row['Num_Credit_Inquiries'], axis=1)
```

[ ]:
```python
data['Num_Credit_Inquiries'].value_counts()
```

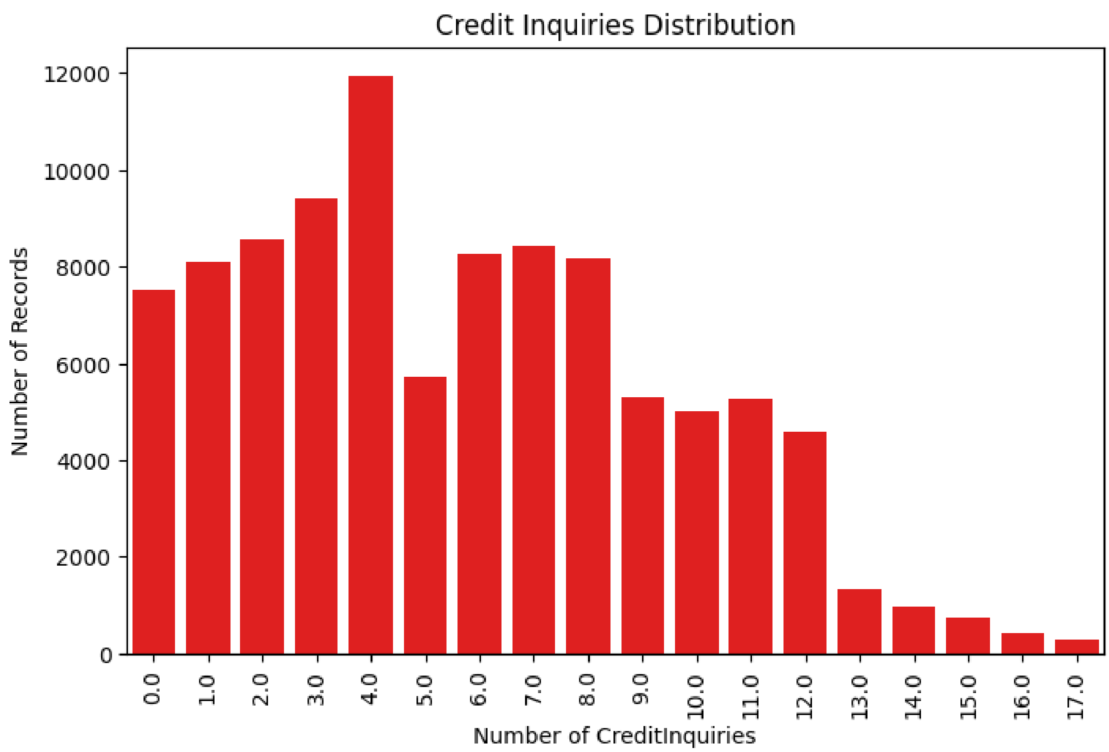
[ ]:
```
Num_Credit_Inquiries
4.0     11936
3.0      9416
2.0      8568
7.0      8416
6.0      8264
8.0      8152
1.0      8104
0.0      7504
5.0      5728
9.0      5304
11.0     5280
10.0     5016
12.0     4592
13.0     1344
14.0      960
```

```
15.0      728
16.0      416
17.0      272
Name: count, dtype: int64
```

[ ]:
```
data['Num_Credit_Inquiries'].isna().sum()
```

[ ]: 0

[ ]:
```
plt.figure(figsize=(8,5))
sns.countplot(data=data, x=data['Num_Credit_Inquiries'], color = "red")
plt.xlabel('Number of CreditInquiries')
plt.ylabel('Number of Records')
plt.title('Credit Inquiries Distribution')
plt.xticks(rotation=90)
plt.show()
```



Column : Credit_mix

[ ]:
```
data['Credit_Mix'].value_counts()
```

[ ]:
```
Credit_Mix
Standard    36479
```

```
Good        24337
_           20195
Bad         18989
Name: count, dtype: int64
```

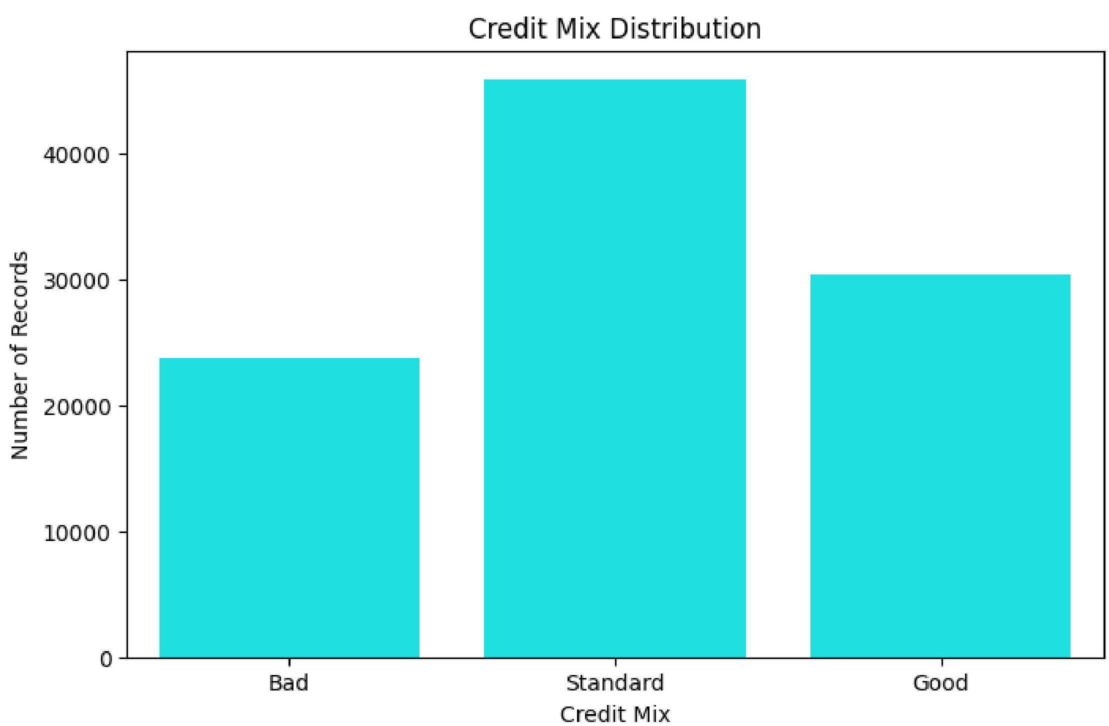
```
[ ]: data['Credit_Mix'] = data['Credit_Mix'].replace('_', np.nan)
```

```
[ ]: data.sort_values(by=['Customer_ID', 'Month'], inplace=True)
     data['Credit_Mix'] = data.groupby('Customer_ID')['Credit_Mix'].
      ↪fillna(method='ffill').fillna(method='bfill')
```

```
[ ]: data['Credit_Mix'].value_counts()
```

```
[ ]: Credit_Mix
     Standard    45848
     Good        30384
     Bad         23768
     Name: count, dtype: int64
```

```
[ ]: plt.figure(figsize=(8,5))
     sns.countplot(data= data, x= data['Credit_Mix'], color = "aqua")
     plt.xlabel('Credit Mix')
     plt.ylabel('Number of Records')
     plt.title('Credit Mix Distribution')
     plt.xticks(rotation=0)
     plt.show()
```



Summary:

- There are 3 types of Credit Mix i.e **Bad, Good & Standard**.

Column : Outstanding Debt

[ ]:
```
data['Outstanding_Debt'].value_counts()
```

[ ]:
```
Outstanding_Debt
1360.45     24
460.46      23
1151.7      23
1109.03     23
100.3       16
            ..
3530.13_     1
1181.44_     1
4078.71_     1
2362.56_     1
1799.87_     1
Name: count, Length: 13178, dtype: int64
```

[ ]:
```
data[['Customer_ID', 'Outstanding_Debt']]
```

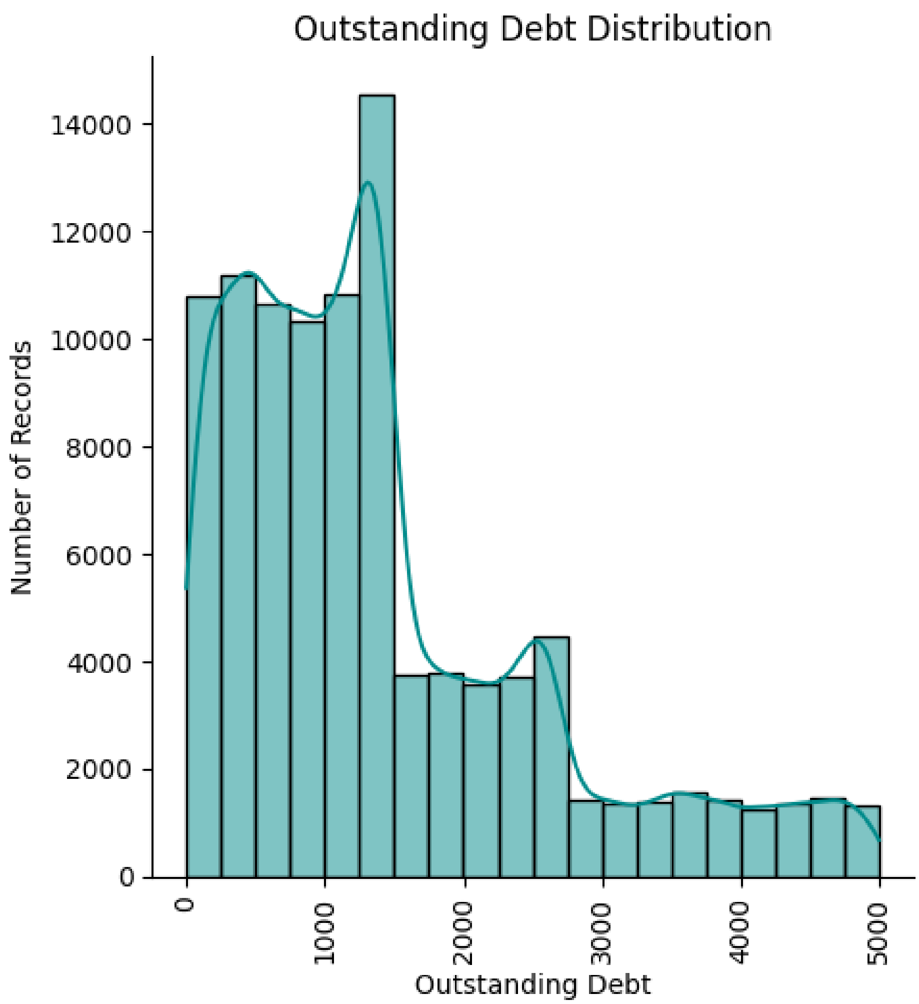
[ ]:
```
      Customer_ID Outstanding_Debt
0       CUS_0x1000          1562.91
1       CUS_0x1000          1562.91
2       CUS_0x1000          1562.91
3       CUS_0x1000          1562.91
4       CUS_0x1000          1562.91
…            …                …
99995    CUS_0xffd          1701.88
99996    CUS_0xffd          1701.88
99997    CUS_0xffd          1701.88
99998    CUS_0xffd          1701.88
99999    CUS_0xffd          1701.88

[100000 rows x 2 columns]
```

[ ]:
```
data['Outstanding_Debt'] = data['Outstanding_Debt'].str.replace('_', '')
data['Outstanding_Debt'] = data['Outstanding_Debt'].astype(float)
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.displot(data= data, x=data['Outstanding_Debt'], kde=True, bins=20, color =␣
 ↪"darkcyan")
plt.xlabel('Outstanding Debt')
```

```
plt.ylabel('Number of Records')
plt.title('Outstanding Debt Distribution')
plt.xticks(rotation=90)
plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```



Column : Credit Utilization Ratio

[ ]:
```
data['Credit_Utilization_Ratio'] =data['Credit_Utilization_Ratio'].round(2)
```

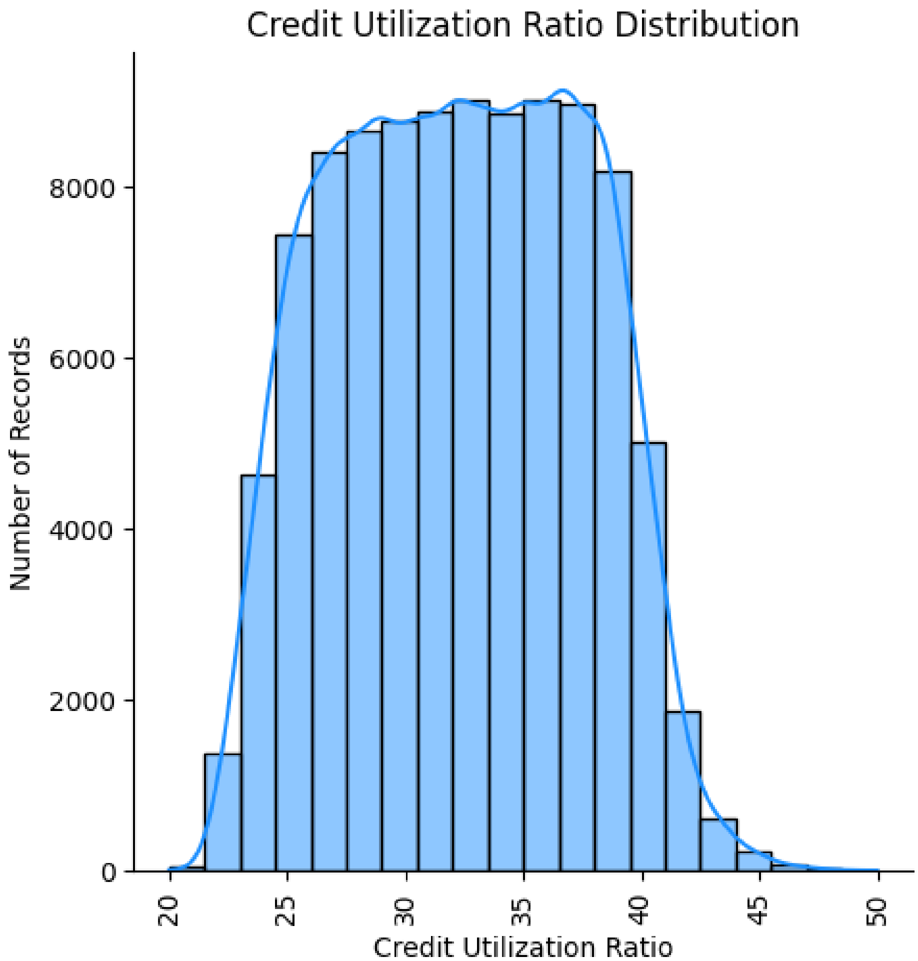
[ ]:
```
data['Credit_Utilization_Ratio'].value_counts()
```

[ ]:
```
Credit_Utilization_Ratio
32.52    86
36.70    84
```

```
29.13    83
31.05    83
29.01    82
         ..
47.29     1
45.27     1
45.70     1
43.80     1
46.28     1
Name: count, Length: 2478, dtype: int64
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.displot(data=df, x=df['Credit_Utilization_Ratio'], kde=True, bins=20, color␣
 ↪= "dodgerblue")
plt.xlabel('Credit Utilization Ratio')
plt.ylabel('Number of Records')
plt.title('Credit Utilization Ratio Distribution')
plt.xticks(rotation=90)
plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```



Column : Credit__history__Age

```
[ ]: data['Credit_History_Age'].isna().sum()
```

```
[ ]: 9030
```

```
[ ]: data['Credit_History_Age'].value_counts()
```

```
[ ]: Credit_History_Age
15 Years and 11 Months    446
19 Years and 4 Months     445
19 Years and 5 Months     444
17 Years and 11 Months    443
19 Years and 3 Months     441
                         …
0 Years and 3 Months       20
0 Years and 2 Months       15
```

```
33 Years and 7 Months      14
33 Years and 8 Months      12
0 Years and 1 Months        2
Name: count, Length: 404, dtype: int64
```

[ ]:
```
grouped_modes = data.groupby('Customer_ID')['Credit_History_Age'].apply(lambda␣
↪x: x.mode().iloc[0])
data['Credit_History_Age'] = data.apply(lambda row:␣
↪grouped_modes[row['Customer_ID']] if row['Credit_History_Age'] !=␣
↪grouped_modes[row['Customer_ID']] else row['Credit_History_Age'], axis=1)
```

[ ]:
```
data['Credit_History_Age'].isna().sum()
```

[ ]:
```
0
```

[ ]:
```
data3 = pd.DataFrame(data['Credit_History_Age'])

def convert_to_months(age_str):
    parts = age_str.split()
    years = int(parts[0])
    months = int(parts[3])
    total_months = years * 12 + months
    return total_months

data['Credit_History_Age_Num'] = data['Credit_History_Age'].apply(lambda x:␣
↪convert_to_months(x))
```

[ ]:
```
data[['Credit_History_Age','Credit_History_Age_Num']]
```

[ ]:
```
          Credit_History_Age  Credit_History_Age_Num
0      10 Years and 2 Months                     122
1      10 Years and 2 Months                     122
2      10 Years and 2 Months                     122
3      10 Years and 2 Months                     122
4      10 Years and 2 Months                     122
…                        …                       …
99995  18 Years and 2 Months                     218
99996  18 Years and 2 Months                     218
99997  18 Years and 2 Months                     218
99998  18 Years and 2 Months                     218
99999  18 Years and 2 Months                     218

[100000 rows x 2 columns]
```

[ ]:
```
data['Credit_History_Age_Num'].max()
```

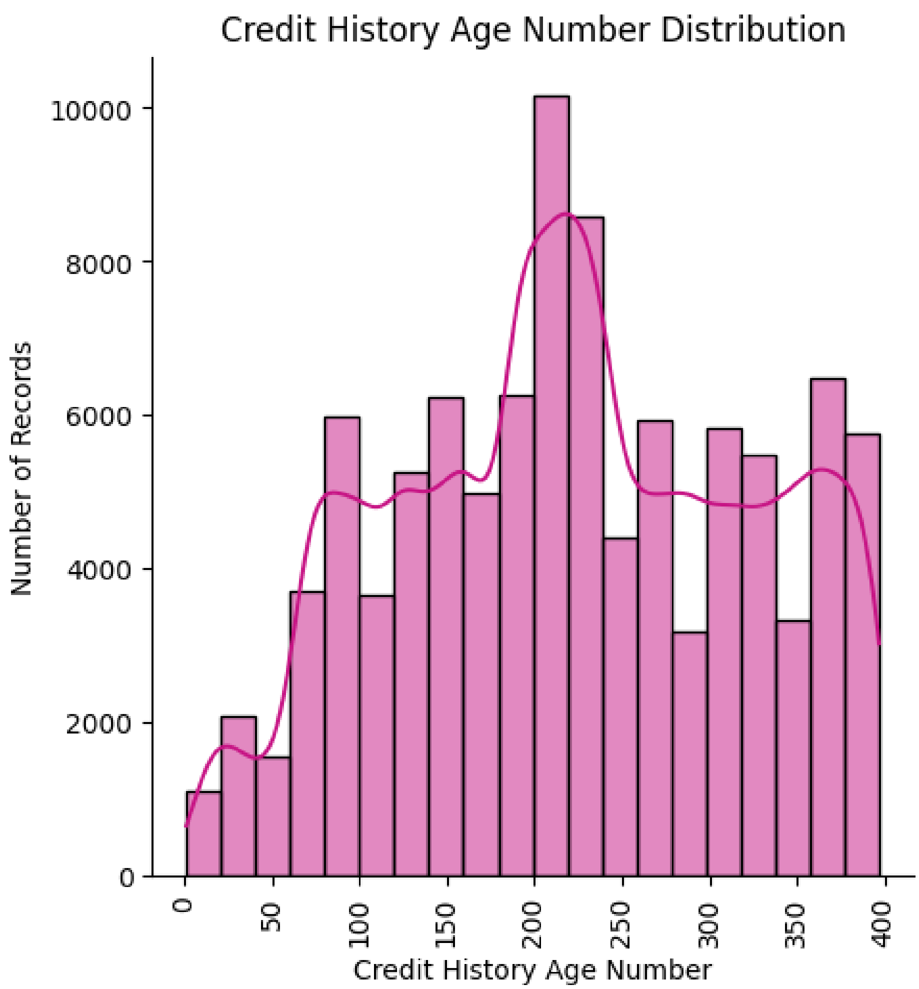
[ ]:
```
397
```

[ ]:
```python
data['Credit_History_Age_Num'].min()
```

[ ]: 1

[ ]:
```python
plt.figure(figsize=(8,5))
sns.displot(data=data, x=data['Credit_History_Age_Num'], kde=True, bins=20,␣
 ↪color = "mediumvioletred")
plt.xlabel('Credit History Age Number')
plt.ylabel('Number of Records')
plt.title('Credit History Age Number Distribution')
plt.xticks(rotation=90)
plt.show()
```

<Figure size 800x500 with 0 Axes>



Summary:

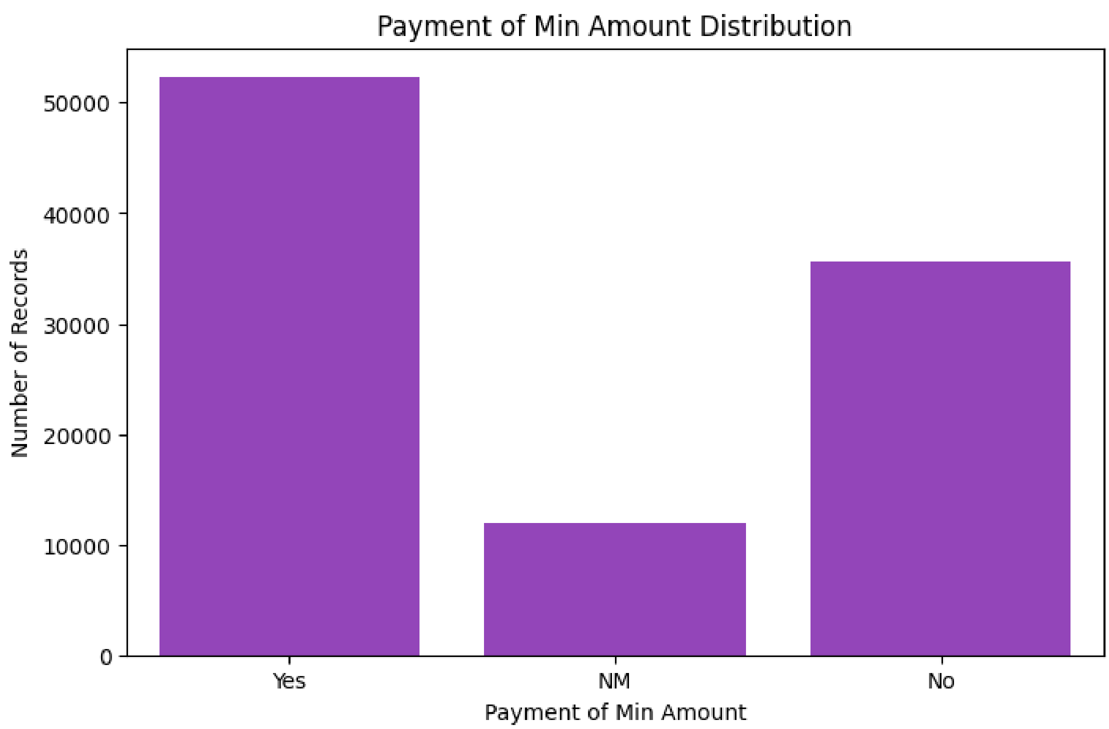
Converted the data from illogical way to analytical format for computation.

Column : Payment_of_min_Amount

[ ]:
```
data['Payment_of_Min_Amount'].value_counts()
```

[ ]:
```
Payment_of_Min_Amount
Yes    52326
No     35667
NM     12007
Name: count, dtype: int64
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.countplot(data=data, x=data['Payment_of_Min_Amount'], color = "darkorchid")
plt.xlabel('Payment of Min Amount')
plt.ylabel('Number of Records')
plt.title('Payment of Min Amount Distribution')
plt.xticks(rotation=0)
plt.show()
```



Column : Total_EMI_per_month

[ ]:
```
data['Total_EMI_per_month'].value_counts()
```

[ ]:
```
Total_EMI_per_month
0.000000        10613
42.941090           8
72.798279           8
119.461755          8
263.655491          8
                    …
39156.000000        1
26128.000000        1
75532.000000        1
78386.000000        1
22380.000000        1
Name: count, Length: 14950, dtype: int64
```

[ ]:
```
grouped_modes = data.groupby('Customer_ID')['Total_EMI_per_month'].apply(lambda␣
↪x: x.mode().iloc[0])
data['Total_EMI_per_month'] = data.apply(lambda row:␣
↪grouped_modes[row['Customer_ID']] if row['Total_EMI_per_month'] !=␣
↪grouped_modes[row['Customer_ID']] else row['Total_EMI_per_month'], axis=1)
```

[ ]:
```
data['Total_EMI_per_month'].max()
```

[ ]:
```
1779.103254
```

[ ]:
```
data['Total_EMI_per_month'].min()
```

[ ]:
```
0.0
```

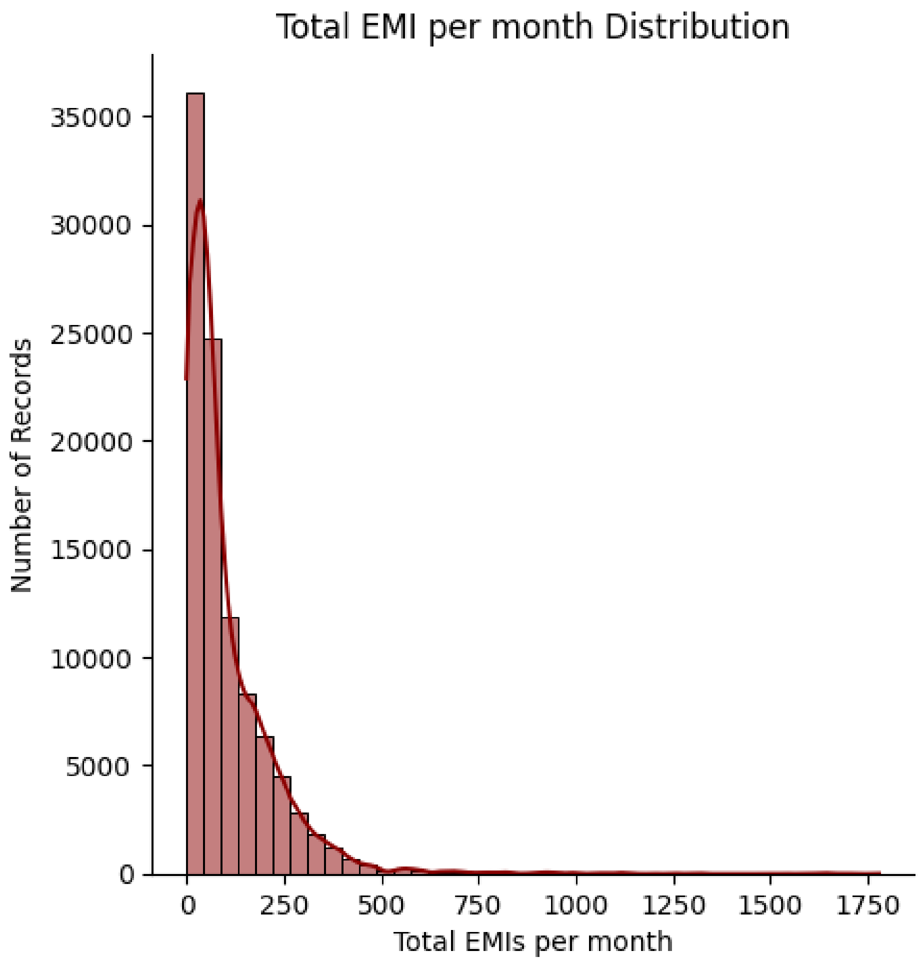
[ ]:
```
data['Total_EMI_per_month'].value_counts()
```

[ ]:
```
Total_EMI_per_month
0.000000      11072
42.941090         8
107.489365        8
78.047064         8
230.815449        8
                  …
341.841495        8
400.386423        8
85.356930         8
61.845295         8
182.976650        8
Name: count, Length: 11117, dtype: int64
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.displot(data=data, x=data['Total_EMI_per_month'], kde=True, bins=40, color␣
↪= "darkred")
```

```
plt.xlabel('Total EMIs per month')
plt.ylabel('Number of Records')
plt.title('Total EMI per month Distribution')
plt.xticks(rotation=0)
plt.show()
```

<Figure size 800x500 with 0 Axes>



Column : Amount_invested_monthly

[ ]:
```
data['Amount_invested_monthly'].isna().sum()
```

[ ]: 4479

[ ]:
```
data['Amount_invested_monthly'].value_counts()
```

[ ]:
```
Amount_invested_monthly
__10000__      4305
0               169
87.90990881       1
459.5317247       1
752.475627        1
               …
105.7266479       1
138.9942681       1
289.9612607       1
76.53803865       1
104.6294735       1
Name: count, Length: 91049, dtype: int64
```

[ ]:
```
data['Amount_invested_monthly'] = pd.
 ↪to_numeric(data['Amount_invested_monthly'], errors='coerce')
data['Amount_invested_monthly'] = data['Amount_invested_monthly'].replace(0, np.
 ↪nan)
data['Amount_invested_monthly']
```

[ ]:
```
0         87.909909
1         77.314276
2        176.132567
3        244.750283
4        266.597160
            …
99995    195.529273
99996    257.989693
99997     47.007379
99998    336.130231
99999    104.629474
Name: Amount_invested_monthly, Length: 100000, dtype: float64
```

[ ]:
```
data[data['Amount_invested_monthly']==0]
```

[ ]:
```
Empty DataFrame
Columns: [ID, Customer_ID, Month, Name, Age, SSN, Occupation, Annual_Income,
Monthly_Inhand_Salary, Num_Bank_Accounts, Num_Credit_Card, Interest_Rate,
Num_of_Loan, Type_of_Loan, Auto Loan, Credit-Builder Loan, Debt Consolidation
Loan, Home Equity Loan, Mortgage Loan, Not Specified, Payday Loan, Personal
Loan, Student Loan, Delay_from_due_date, Num_of_Delayed_Payment,
Changed_Credit_Limit, Num_Credit_Inquiries, Credit_Mix, Outstanding_Debt,
Credit_Utilization_Ratio, Credit_History_Age, Payment_of_Min_Amount,
Total_EMI_per_month, Amount_invested_monthly, Payment_Behaviour,
Monthly_Balance, Credit_History_Age_Num]
Index: []
```

```
[0 rows x 37 columns]
```

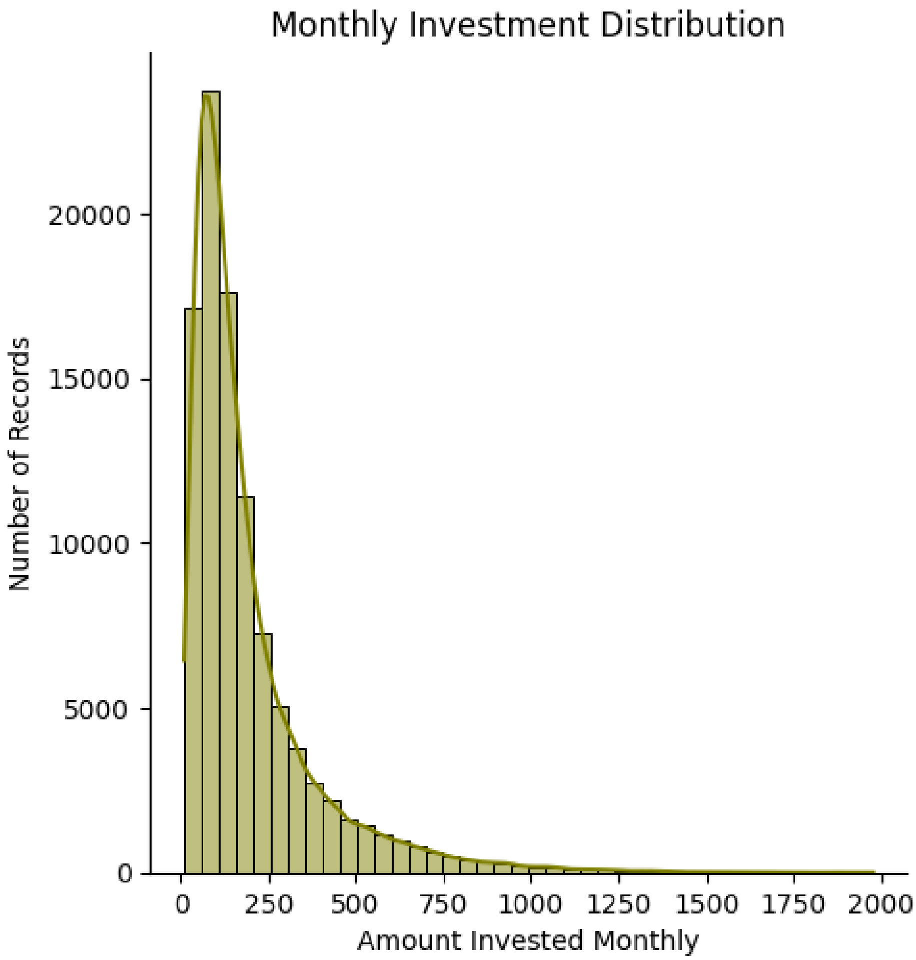
```
[ ]: data['Amount_invested_monthly'].isna().sum()
```

```
[ ]: 8953
```

```
[ ]: mean_per_customer = data.groupby('Customer_ID')['Amount_invested_monthly'].
      ↪mean()
     mask = data['Amount_invested_monthly'].isna()
     data.loc[mask, 'Amount_invested_monthly'] = data.loc[mask, 'Customer_ID'].
      ↪map(mean_per_customer)
```

```
[ ]: plt.figure(figsize=(8,5))
     sns.displot(data=data, x=data['Amount_invested_monthly'], kde=True, bins=40,␣
      ↪color = "olive")
     plt.xlabel('Amount Invested Monthly')
     plt.ylabel('Number of Records')
     plt.title('Monthly Investment Distribution')
     plt.xticks(rotation=0)
     plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```



[ ]:
```
data['Amount_invested_monthly'].isna().sum()
```

[ ]:
```
0
```

[ ]:
```
data['Amount_invested_monthly'] = data['Amount_invested_monthly'].round(2)
```

Column : Payment_Behaviour

[ ]:
```
data['Payment_Behaviour'].value_counts()
```

[ ]:
```
Payment_Behaviour
Low_spent_Small_value_payments      25513
High_spent_Medium_value_payments    17540
Low_spent_Medium_value_payments     13861
High_spent_Large_value_payments     13721
High_spent_Small_value_payments     11340
Low_spent_Large_value_payments      10425
```

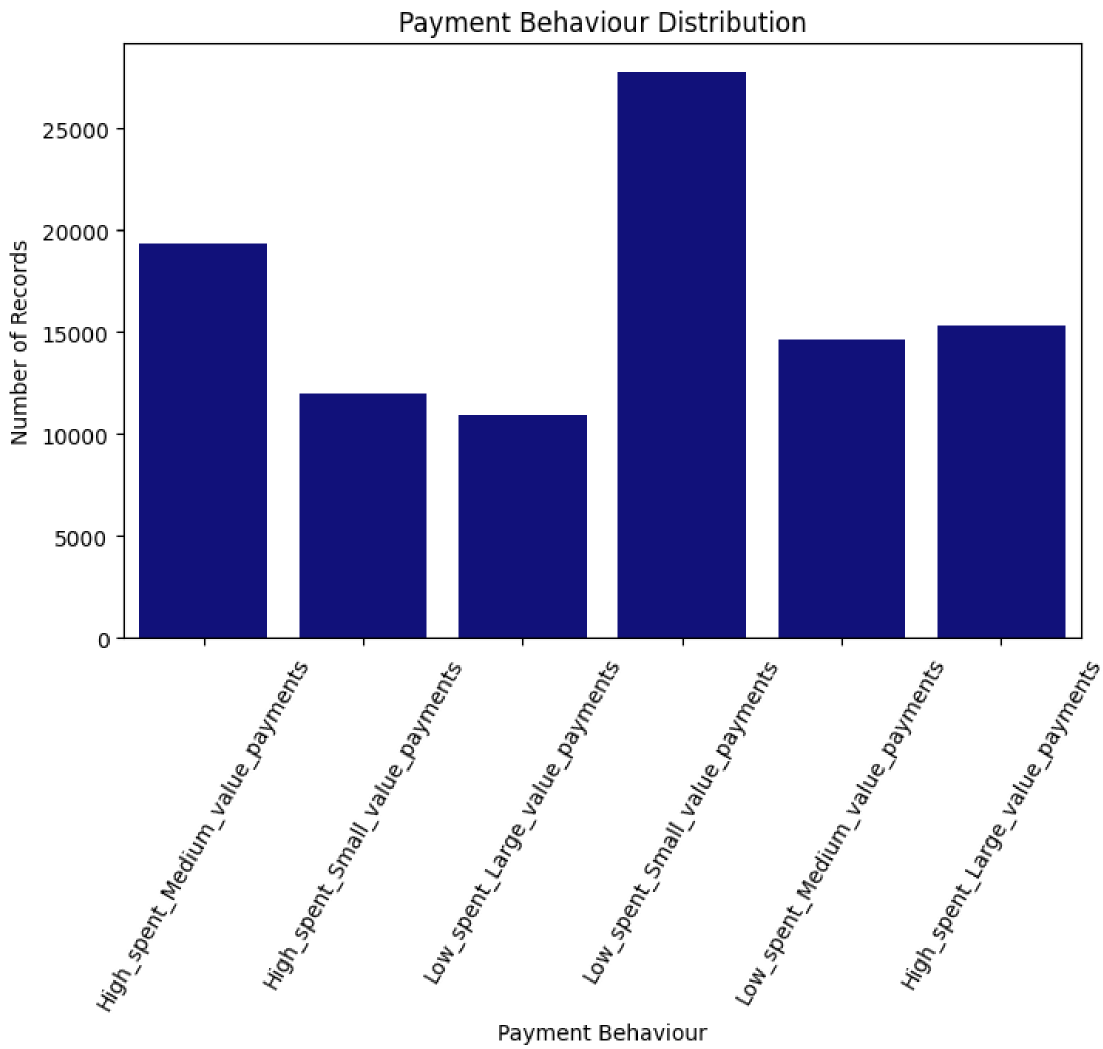
```
!@9#%8                               7600
Name: count, dtype: int64
```

[ ]:
```
data['Payment_Behaviour'] = data['Payment_Behaviour'].replace('!@9#%8', np.nan)
data['Payment_Behaviour'] = data.groupby('Customer_ID')['Payment_Behaviour'].
 ↪transform(lambda x: x.fillna(x.mode().iloc[0]))
```

[ ]:
```
data['Payment_Behaviour'].value_counts()
```

[ ]:
```
Payment_Behaviour
Low_spent_Small_value_payments      27767
High_spent_Medium_value_payments    19366
High_spent_Large_value_payments     15348
Low_spent_Medium_value_payments     14621
High_spent_Small_value_payments     11980
Low_spent_Large_value_payments      10918
Name: count, dtype: int64
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.countplot(data= data, x=data['Payment_Behaviour'], color = "darkblue")
plt.xlabel('Payment Behaviour')
plt.ylabel('Number of Records')
plt.title('Payment Behaviour Distribution')
plt.xticks(rotation=60)
plt.show()
```



Column : Monthly_Balance

[ ]:
```
data['Monthly_Balance'].isna().sum()
```

[ ]:
```
1200
```

[ ]:
```
data['Monthly_Balance'].value_counts()
```

[ ]:
```
Monthly_Balance
__-333333333333333333333333333__    9
350.0148691                         2
695.0571561                         2
419.7651674                         1
615.6677195                         1
                                   ..
```

```
259.3760946                    1
343.7619864                    1
288.6680278                    1
468.4784226                    1
337.380877                     1
Name: count, Length: 98790, dtype: int64
```

[ ]:
```
data['Monthly_Balance'].nunique()
```

[ ]:
```
98790
```

[ ]:
```
data['Monthly_Balance'] = data['Monthly_Balance'].
 ↪replace('__-333333333333333333333333333__', np.nan)
data['Monthly_Balance'] = pd.to_numeric(data['Monthly_Balance'],␣
 ↪errors='coerce')
data['Monthly_Balance'] = data.groupby('Customer_ID')['Monthly_Balance'].
 ↪transform(lambda x: x.fillna(x.mean()))
```

[ ]:
```
data['Monthly_Balance'].value_counts()
```

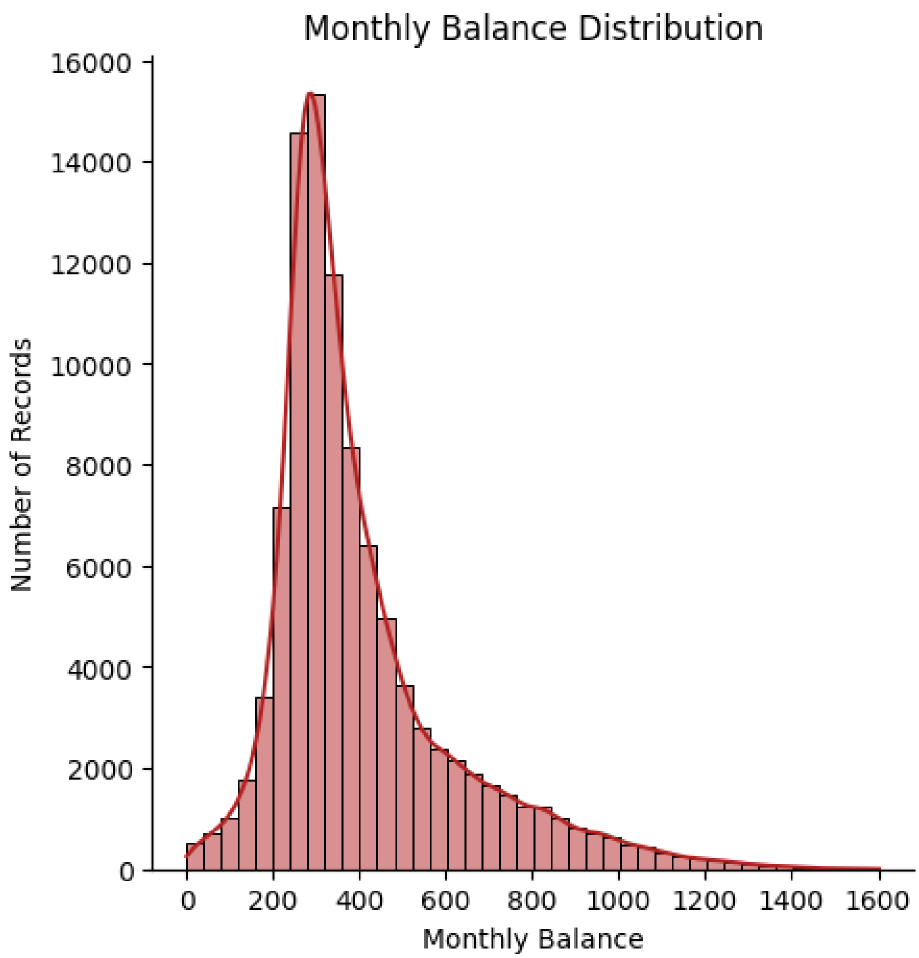
[ ]:
```
Monthly_Balance
261.565962    5
464.392372    4
238.332338    4
215.181452    4
164.119697    4
             ..
319.503931    1
345.075800    1
338.115057    1
344.112554    1
337.380877    1
Name: count, Length: 99757, dtype: int64
```

[ ]:
```
data['Monthly_Balance'].isna().sum()
```

[ ]:
```
0
```

[ ]:
```
plt.figure(figsize=(8,5))
sns.displot(data= data, x= data['Monthly_Balance'], kde=True, bins=40, color =␣
 ↪"firebrick")
plt.xlabel('Monthly Balance')
plt.ylabel('Number of Records')
plt.title('Monthly Balance Distribution')
plt.xticks(rotation=0)
plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```



Heatmap: Correlation Check

[ ]:
```python
data_heatmap = data.select_dtypes(include = ['number'])
```

[ ]:
```python
data_heatmap
```

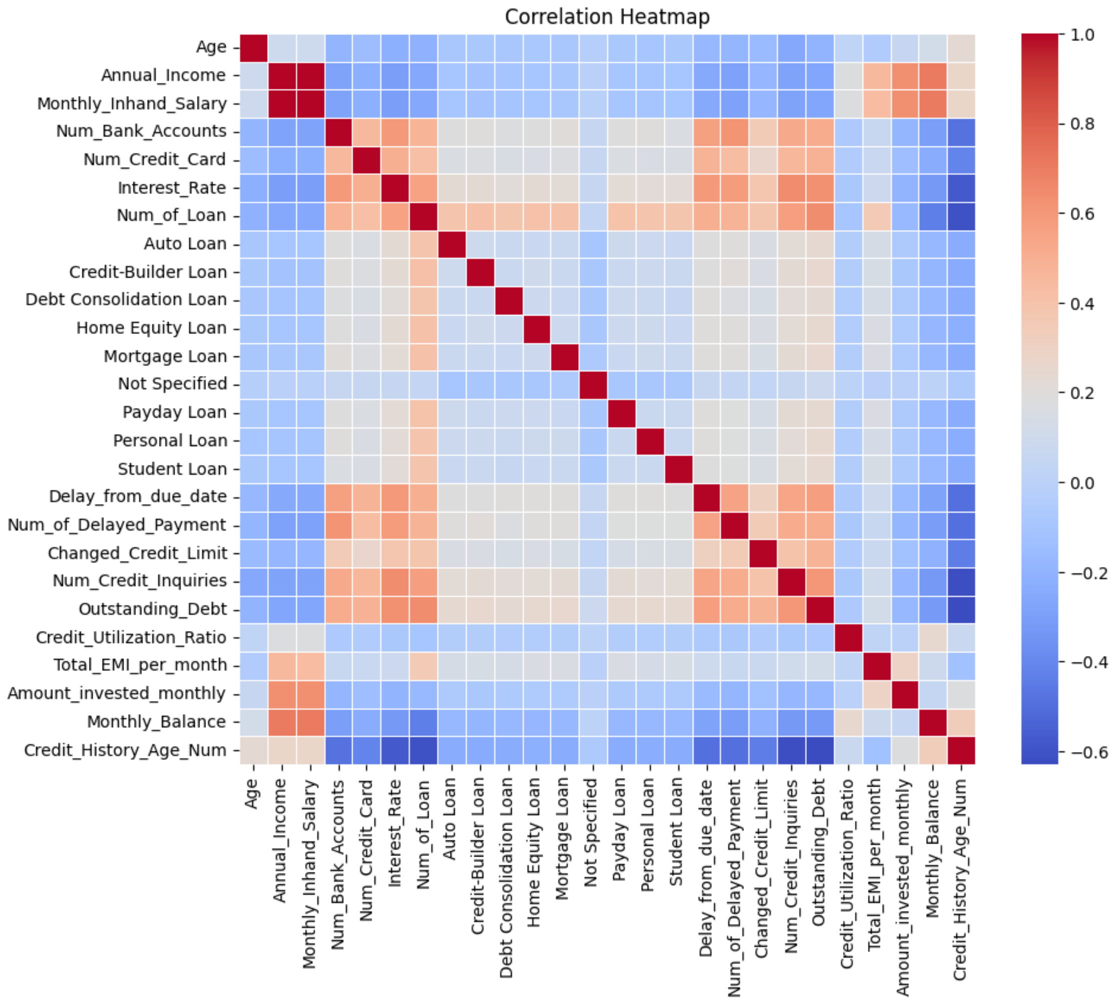
[ ]:
```
       Age  Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  \
0       17       30625.94            2706.161667                  6
1       17       30625.94            2706.161667                  6
2       17       30625.94            2706.161667                  6
3       17       30625.94            2706.161667                  6
4       17       30625.94            2706.161667                  6
…      …           …                     …                   …
99995   29       41398.44            3749.870000                  8
99996   29       41398.44            3749.870000                  8
```

```
99997   29       41398.44           3749.870000                 8
99998   29       41398.44           3749.870000                 8
99999   29       41398.44           3749.870000                 8

       Num_Credit_Card  Interest_Rate  Num_of_Loan  Auto Loan  \
0                    5             27            2          0
1                    5             27            2          0
2                    5             27            2          0
3                    5             27            2          0
4                    5             27            2          0
…                  …             …           …         …
99995                7             13            6          1
99996                7             13            6          1
99997                7             13            6          1
99998                7             13            6          1
99999                7             13            6          1

       Credit-Builder Loan  Debt Consolidation Loan  …  Delay_from_due_date  \
0                        1                        0  …                   64
1                        1                        0  …                   57
2                        1                        0  …                   62
3                        1                        0  …                   62
4                        1                        0  …                   62
…                      …                      …  …                 …
99995                    0                        0  …                   25
99996                    0                        0  …                   23
99997                    0                        0  …                   23
99998                    0                        0  …                   25
99999                    0                        0  …                   23

       Num_of_Delayed_Payment  Changed_Credit_Limit  Num_Credit_Inquiries  \
0                        25.0                  1.63                  11.0
1                        25.0                  1.63                  11.0
2                        25.0                  1.63                  11.0
3                        25.0                  1.63                  11.0
4                        25.0                  1.63                  11.0
…                         …                   …                   …
99995                    12.0                 10.07                   7.0
99996                    12.0                 10.07                   7.0
99997                    12.0                 10.07                   7.0
99998                    12.0                 10.07                   7.0
99999                    12.0                 10.07                   7.0

       Outstanding_Debt  Credit_Utilization_Ratio  Total_EMI_per_month  \
0               1562.91                     32.84             42.94109
1               1562.91                     30.08             42.94109
2               1562.91                     29.44             42.94109
```

```
3               1562.91                      26.61             42.94109
4               1562.91                      38.15             42.94109
…                 …                        …                 …
99995           1701.88                      29.51            182.97665
99996           1701.88                      33.92            182.97665
99997           1701.88                      36.97            182.97665
99998           1701.88                      25.18            182.97665
99999           1701.88                      26.17            182.97665

       Amount_invested_monthly  Monthly_Balance  Credit_History_Age_Num
0                        87.91       419.765167                     122
1                        77.31       400.360800                     122
2                       176.13       311.542510                     122
3                       244.75       252.924793                     122
4                       266.60       251.077916                     122
…                          …                …                    …
99995                   195.53       266.481077                     218
99996                   257.99       194.020657                     218
99997                    47.01       395.002972                     218
99998                   336.13       145.880120                     218
99999                   104.63       337.380877                     218

[100000 rows x 26 columns]
```

[ ]:

```
plt.figure(figsize=(10, 8))
sns.heatmap(data_heatmap.corr(), cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



Summary:

- Strong positive correlation for features like Annual_Income, Monthly_Inhand_Salary, Monthly_Balance & Amount_Invested_Monthly.
- Positive Correlation can also be found among features like Num_Credit_inquiries, Outstanding_debt, Num_of_delayed_payment, Num_Bank_Account.
- Strong negative correlation can be found among Credit_history_age, Outstanding_debt, Num_of_loan, Interest_rate.

[ ]:
```
data.describe().T
```

[ ]:
```
                          count          mean           std          min  \
Age                    100000.0     33.282000     10.766568    14.000000
Annual_Income          100000.0  50505.123449  38299.422093  7005.930000
Monthly_Inhand_Salary  100000.0   4198.262107   3187.363227   303.645417
Num_Bank_Accounts      100000.0      5.411840      2.508237     1.000000
Num_Credit_Card        100000.0      5.532720      2.067504     0.000000
```

```
Interest_Rate             100000.0     14.532080     8.741330     1.000000
Num_of_Loan               100000.0      3.532880     2.446356     0.000000
Auto Loan                 100000.0      0.305600     0.460663     0.000000
Credit-Builder Loan       100000.0      0.317280     0.465420     0.000000
Debt Consolidation Loan   100000.0      0.310400     0.462660     0.000000
Home Equity Loan          100000.0      0.314000     0.464119     0.000000
Mortgage Loan             100000.0      0.313600     0.463958     0.000000
Not Specified             100000.0      0.430880     0.495202     0.000000
Payday Loan               100000.0      0.319440     0.466262     0.000000
Personal Loan             100000.0      0.311040     0.462921     0.000000
Student Loan              100000.0      0.310400     0.462660     0.000000
Delay_from_due_date       100000.0     21.068780    14.860104    -5.000000
Num_of_Delayed_Payment    100000.0     13.266640     6.194986     0.000000
Changed_Credit_Limit      100000.0     10.392847     6.511672     0.000000
Num_Credit_Inquiries      100000.0      5.677760     3.827248     0.000000
Outstanding_Debt          100000.0   1426.220376  1155.129026     0.230000
Credit_Utilization_Ratio  100000.0     32.285183     5.116880    20.000000
Total_EMI_per_month       100000.0    105.543371   125.810030     0.000000
Amount_invested_monthly   100000.0    195.838897   195.041856    10.010000
Monthly_Balance           100000.0    403.120320   214.014558     0.007760
Credit_History_Age_Num    100000.0    220.156240    99.580975     1.000000

                                  25%           50%           75%  \
Age                         24.000000     33.000000     42.000000
Annual_Income            19342.972500  36999.705000  71683.470000
Monthly_Inhand_Salary     1626.594167   3096.066250   5957.715000
Num_Bank_Accounts            3.000000      5.000000      7.000000
Num_Credit_Card              4.000000      5.000000      7.000000
Interest_Rate                7.000000     13.000000     20.000000
Num_of_Loan                  2.000000      3.000000      5.000000
Auto Loan                    0.000000      0.000000      1.000000
Credit-Builder Loan          0.000000      0.000000      1.000000
Debt Consolidation Loan      0.000000      0.000000      1.000000
Home Equity Loan             0.000000      0.000000      1.000000
Mortgage Loan                0.000000      0.000000      1.000000
Not Specified                0.000000      0.000000      1.000000
Payday Loan                  0.000000      0.000000      1.000000
Personal Loan                0.000000      0.000000      1.000000
Student Loan                 0.000000      0.000000      1.000000
Delay_from_due_date         10.000000     18.000000     28.000000
Num_of_Delayed_Payment       9.000000     14.000000     18.000000
Changed_Credit_Limit         5.500000      9.340000     14.670000
Num_Credit_Inquiries         3.000000      5.000000      8.000000
Outstanding_Debt           566.072500   1166.155000   1945.962500
Credit_Utilization_Ratio    28.050000     32.310000     36.500000
Total_EMI_per_month         29.049047     66.033915    145.582332
Amount_invested_monthly     74.640000    131.210000    239.480000
```

```
Monthly_Balance            270.189030    337.114461    471.570652
Credit_History_Age_Num     142.000000    216.000000    299.000000

                                   max
Age                          56.000000
Annual_Income            179987.280000
Monthly_Inhand_Salary     15204.633330
Num_Bank_Accounts            10.000000
Num_Credit_Card              11.000000
Interest_Rate                34.000000
Num_of_Loan                   9.000000
Auto Loan                     1.000000
Credit-Builder Loan           1.000000
Debt Consolidation Loan       1.000000
Home Equity Loan              1.000000
Mortgage Loan                 1.000000
Not Specified                 1.000000
Payday Loan                   1.000000
Personal Loan                 1.000000
Student Loan                  1.000000
Delay_from_due_date          67.000000
Num_of_Delayed_Payment       28.000000
Changed_Credit_Limit         29.980000
Num_Credit_Inquiries         17.000000
Outstanding_Debt           4998.070000
Credit_Utilization_Ratio     50.000000
Total_EMI_per_month        1779.103254
Amount_invested_monthly    1977.330000
Monthly_Balance            1602.040519
Credit_History_Age_Num      397.000000
```

[ ]:
```
data.describe(include='object').T
```

[ ]:

	count	unique	top	freq
ID	100000	100000	0x1628d	1
Customer_ID	100000	12500	CUS_0x1000	8
Month	100000	8	April	12500
Name	100000	10139	Jessicad	48
SSN	100000	12500	913-74-1218	8
Occupation	100000	15	Lawyer	7096
Type_of_Loan	100000	6260	Not Specified	12816
Credit_Mix	100000	3	Standard	45848
Credit_History_Age	100000	249	15 Years and 10 Months	3488
Payment_of_Min_Amount	100000	3	Yes	52326
Payment_Behaviour	100000	6	Low_spent_Small_value_payments	27767

Label Encoding Features

[ ]:
```
data["Payment_of_Min_Amount"] = data["Payment_of_Min_Amount"].replace({"Yes":␣
↪1, "No": 0, "NM": 0})
```

[ ]:
```
data["Credit_Mix"] = data["Credit_Mix"].replace({"Standard": 1, "Good": 2,␣
↪"Bad": 0})
```

[ ]:
```
data["Payment_Behaviour"] = data["Payment_Behaviour"].replace({
      "Low_spent_Small_value_payments": 1,
      "High_spent_Medium_value_payments": 2,
      "Low_spent_Medium_value_payments": 3,
      "High_spent_Large_value_payments": 4,
      "High_spent_Small_value_payments": 5,
      "Low_spent_Large_value_payments": 6
  })
```

Summary:

- Low_spent_Small_value_payments: 1
- High_spent_Small_value_payments: 2
- Low_spent_Medium_value_payments: 3
- High_spent_Medium_value_payments: 4
- Low_spent_Large_value_payments: 5
- High_spent_Large_value_payments: 6

This numeric representation captures the hierarchy where higher numbers represent higher spent value or larger payments.

Feature Engineering

1. Debt to Income Ratio:

[ ]:
```
data['Monthly_Debt_to_Income_Ratio'] = data['Outstanding_Debt'] /␣
↪data['Monthly_Inhand_Salary']
```

2. Debt Repayment Capacity

[ ]:
```
data['Monthly_Debt_Repayment_Capacity'] = data['Monthly_Inhand_Salary'] -␣
↪data['Total_EMI_per_month']
```

3. Payment History Score

[ ]:
```
data["Payment_History_Score"] = (
      - 1 * data["Delay_from_due_date"]
      - 1 * data["Num_of_Delayed_Payment"]
      + 1 * data["Payment_of_Min_Amount"]
  )
```

2 Credit Score Calculation:

Selected features for credit score calulcation with their weights:

1. Payment history score

- Weight: 0.30
- Strongest predictor of future credit behavior.

2. Credit History Age in Months

- Weight: 0.20
- Longer credit history indicates responsible credit usage.Weighted moderately to reflect its significance.

3. Monthly Debt-to-Income Ratio (MDTIR)

- Weight: 0.15
- Lower ratio indicates better ability to manage debt.Weighted lower due to potential fluctuations in income.

4. Credit Utilization Ratio

- Weight: 0.10
- Lower ratio suggests responsible credit card usage.Weighted lower as it's a snapshot of current utilization.

5. Monthly Debt Repayment Capacity

- Weight: 0.05
- Reflects ability to manage existing debt.

6. Outstanding Debt

- Weight: 0.05
- Higher debt increases risk of default.

7. Num_Credit_Inquiries

- Weight: 0.05
- Fewer inquiries suggest lower credit-seeking behavior.

8. Payment Behaviour

- Weight: 0.05
- Insights into spending patterns and payment tendencies.

9. Credit Mix

- Weight: 0.05
- Taking different types of credit

3 Calculate & Calibrate Credit Score:

[ ]:
```python
def calculate_credit_score(data):

  #Group by Customer ID, handling month-level data and calculating scores
  grouped_data = data.groupby("Customer_ID").agg(
      Payment_History_Score           = ("Payment_History_Score", "mean"),
```

```python
        Credit_History_Age_Num          = ("Credit_History_Age_Num", "max"),      #Use maximum history age
        Monthly_Debt_to_Income_Ratio    = ("Monthly_Debt_to_Income_Ratio", "mean"),
        Credit_Utilization_Ratio        = ("Credit_Utilization_Ratio", "mean"),
        Monthly_Debt_Repayment_Capacity = ("Monthly_Debt_Repayment_Capacity",'mean'),
        Outstanding_Debt                = ("Outstanding_Debt", "mean"),
        Num_Credit_Inquiries            = ("Num_Credit_Inquiries", "sum"),
        Payment_Behaviour               = ("Payment_Behaviour","mean"),      #Use average payment behaviour encoding
        Credit_Mix                      = ("Credit_Mix","mean")
    )

    #Standardize values for numerical features
    grouped_data = (grouped_data - grouped_data.mean()) / grouped_data.std()

    # Calculate weighted scores
    grouped_data["credit_score"] = (
        0.30 * grouped_data["Payment_History_Score"]
        + 0.20 * grouped_data["Credit_History_Age_Num"]
        + 0.15 * (1-grouped_data["Monthly_Debt_to_Income_Ratio"]) #Inverse relation as lower the value better the financials
        + 0.10 * (1-grouped_data["Credit_Utilization_Ratio"]) #inverse relation
        + 0.05 * grouped_data["Monthly_Debt_Repayment_Capacity"]
        + 0.05 * grouped_data["Outstanding_Debt"]
        + 0.05 * (1-grouped_data["Num_Credit_Inquiries"]) #Inverse relation
        + 0.05 * grouped_data["Payment_Behaviour"]
        + 0.05 * grouped_data["Credit_Mix"]
    )

    #Normalize scores to a range of 0 to 100
    grouped_data["credit_score"] = (grouped_data["credit_score"] - grouped_data["credit_score"].min()) / (grouped_data["credit_score"].max() - grouped_data["credit_score"].min()) * 100

    #Map scores to the original FICO scale (300 to 850)
    min_range, max_range = 300, 850
    grouped_data["credit_score"] = (grouped_data["credit_score"] * (max_range - min_range) / 100) + min_range

    return grouped_data.reset_index()


    # Calculate scores for all customers
credit_scores_data = calculate_credit_score(data)
```

```
credit_scores_data[["Customer_ID","credit_score"]]
```

[ ]:
```
      Customer_ID  credit_score
0      CUS_0x1000    495.215414
1      CUS_0x1009    765.188490
2      CUS_0x100b    725.966507
3      CUS_0x1011    678.701861
4      CUS_0x1013    731.596559
…             …           …
12495   CUS_0xff3    682.252337
12496   CUS_0xff4    687.775950
12497   CUS_0xff6    799.658279
12498   CUS_0xffc    561.434848
12499   CUS_0xffd    676.226487

[12500 rows x 2 columns]
```

4 Customer with Highest Credit Score:

[ ]:
```
max_value = credit_scores_data['credit_score'].max()
```

[ ]:
```
credit_scores_data[credit_scores_data['credit_score'] == max_value].T
```

[ ]:
```
                                    5701
Customer_ID                   CUS_0x65bf
Payment_History_Score            1.45596
Credit_History_Age_Num          1.745692
Monthly_Debt_to_Income_Ratio   -0.617335
Credit_Utilization_Ratio        -0.53696
Monthly_Debt_Repayment_Capacity 2.484007
Outstanding_Debt               -0.796567
Num_Credit_Inquiries           -1.222183
Payment_Behaviour               1.033527
Credit_Mix                       1.27412
credit_score                       850.0
```

Customer with Lowest Credit Score:

[ ]:
```
min_value = credit_scores_data['credit_score'].min()
```

[ ]:
```
credit_scores_data[credit_scores_data['credit_score'] == min_value].T
```

[ ]:
```
                                    8310
Customer_ID                   CUS_0x8c6f
Payment_History_Score          -1.830869
Credit_History_Age_Num         -1.507828
```

```
Monthly_Debt_to_Income_Ratio        10.03639
Credit_Utilization_Ratio           -0.798419
Monthly_Debt_Repayment_Capacity     -1.22304
Outstanding_Debt                    1.971018
Num_Credit_Inquiries                2.696945
Payment_Behaviour                  -1.124135
Credit_Mix                         -1.454655
credit_score                           300.0
```

5 Insights & Takeaways:

1. Unique Customers: 12500
2. Data information of Customer profile is available between Jan to Aug i.e. only for 8 months
3. Various type of Loans availed by customers were -

- Auto Loan
- Credit-Builder Loan
- Debt Consolidation Loan
- Home Equity Loan
- Mortgage Loan
- Payday Loan
- Personal Loan
- Student Loan
- Unspecified Loan

4. Most customers have a low annual income and distribution is right skewed.
5. Most customers have a low monthly income and distribution is right skewed.
6. Majority of customers has no. of bank accounts between 3 to 8.
7. Number of credit cards range from 0 to 11 with most of the customers having credit cards in the range of 3 to 7 with peak at 5.
8. Interest rate is spread across 1% to 34%.
9. Very few customers invest greater than 2k amount per month.
10. Customers typically take anywhere from 2 to 4 loans, with the maximum number being 9.
11. Typically, most customers belong to the Low_spent_small_value_payments and High_spent_medium-value_payments.
12. Minimum Credit history is 1 month with highest as 397.

For credit score calculation we have used following features with their respective weights

1. Payment histroy score: (Weight: 0.30)
2. Credit History Age in Months (Weight: 0.20)
3. Monthly Debt-to-Income Ratio (MDTIR) (Weight: 0.15)
4. 4redit Utilization Ratio (Weight: 0.10)
5. Monthly Debt Repayment Capacity (Weight: 0.05)
6. Outstanding Debt (Weight: 0.05)
7. Num_Credit_Inquiries (Weight: 0.05)
8. Payment Behaviour (Weight: 0.05)
9. Credit_mix (Weight: 0.05)

6 Recommendations:

- The current credit score model uses a basic set of factors to calculate scores. To enhance reliability, we can delve into adjusting the importance of each factor through various weighting schemes. For example, we might assign more weight to factors that have a stronger impact on creditworthiness, such as payment history and credit utilization. This way, the model can better reflect the nuances of individual financial behavior.
- Consider expanding the set of features used for credit score calculation. This could involve incorporating alternative data sources such as social media behavior, rental payment history, or utility bill payments. Experimenting with new features can provide a more comprehensive and accurate representation of an individual's financial responsibility and creditworthiness.
- Engage with domain experts, such as credit analysts and financial professionals, to gain insights into the nuances of creditworthiness. Their expertise can guide the selection of features, model design, and interpretation of results, ultimately improving the reliability of the credit score.