Credit Score EDA Case Study

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.
- This casestudy 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.
- 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

> **Data Dictionary:**

Column Name	Description
ID	Represents a unique identification of an entry
Customer_ID	Represents a unique identification of a person
Month	Represents the month of the year

Column Name	Description
Name	Represents the name of a person
Age	Represents the age of the person
SSN	Represents the social security number of a person
Occupation	Represents the occupation of the person
Annual_Income	Represents the annual income of the person
Monthly_Inhand_Salary	Represents the monthly base salary of a person
Num_Bank_Accounts	Represents the number of bank accounts a person holds
Num_Credit_Card	Represents the number of other credit cards held by a person
Interest_Rate	Represents the interest rate on credit card
Num_of_Loan	Represents the number of loans taken from the bank
Type_of_Loan	Represents the types of loan taken by a person
Delay_from_due_date	Represents the average number of days delayed from the payment date
Num_of_Delayed_Payment	Represents the average number of payments delayed by a person
Changed_Credit_Limit	Represents the percentage change in credit card limit
Num_Credit_Inquiries	Represents the number of credit card inquiries
Credit_Mix	Represents the classification of the mix of credits
Outstanding_Debt	Represents the remaining debt to be paid (in USD)
Credit_Utilization_Ratio	Represents the utilization ratio of credit card
Credit_History_Age	Represents the age of credit history of the person
Payment_of_Min_Amount	Represents whether only the minimum amount was paid by the person
Total_EMI_per_month	Represents the monthly EMI payments (in USD)
Amount_invested_monthly	Represents the monthly amount invested by the customer (in USD)
Payment_Behaviour	Represents the payment behavior of the customer (in USD)
Monthly_Balance	Represents the monthly balance amount of the customer (in USD)

Methodology

> Exploratory Data Analysis (EDA):

- Performing a comprehensive EDA to understand the data's structure, characteristics, distributions, and relationships.
- Identified and addressed any missing values, mismatch data types, inconsistencies, or outliers.

- Utilized appropriate visualizations (e.g., histograms, scatter plots, box plots, correlation matrices) to uncover patterns and insights.

> Feature Engineering:

- Created new features that can be leveraged for the calculation of credit scores based on domain knowledge and insights from EDA

> Hypothetical Credit Score Calculation:

- Developed a methodology to calculate a hypothetical credit score(kinda CIBIL/FICO ranges from 300-850,900) using relevant features (using a minimum of 5 maximum of 10 features) and justified it.
- Explored various weighting schemes to assign scores.
- Provided a score for each individual customer

> Also thought about:

- Can credit score and aggregated features be calculated at different time frames like the last 3 months/last 6 months (recency based metrics)

> Analysis and Insights

- Added valuable insights from EDA and credit score calculation

Importing Libraries and Modules

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings('ignore')
```

Data Wrangling

In [2]:
```
pd.set_option('display.max_columns',50)
pd.set_option('display.max_rows', 50)
```

In [3]:
```
import gdown
dataset = 'https://drive.google.com/file/d/1pljm6_3nxcFS9UMIFm124HBsjNZP6ACA/vie
output = 'data.csv' # You can change the output filename
gdown.download(url=dataset, output=output, quiet=False, fuzzy=True)
```

```
Downloading...
From: https://drive.google.com/uc?id=1pljm6_3nxcFS9UMIFm124HBsjNZP6ACA
To: /content/data.csv
100%|██████████| 27.4M/27.4M [00:00<00:00, 182MB/s]
```

Out[3]: 'data.csv'

In [4]:
```
data = pd.read_csv('data.csv')
data.head(8)
```

Out[4]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Inc
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	191
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	191
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	191
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	191
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	191
5	0x1607	CUS_0xd40	June	Aaron Maashoh	23	821-00-0265	Scientist	191
6	0x1608	CUS_0xd40	July	Aaron Maashoh	23	821-00-0265	Scientist	191
7	0x1609	CUS_0xd40	August	NaN	23	#F%$D@*&8	Scientist	191

In [5]:
```
df = data.copy()
```

In [6]:
```
df.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
```

In [7]:
```
df[df.duplicated()]
```

Out[7]:

ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_In

Insights

- This data has no duplicates.

Column wise Cleaning

In [8]:
```
categorical_columns = df.select_dtypes(include=['object', 'category']).columns
numerical_columns = df.select_dtypes(include=['number']).columns
```

In [9]:
```
categorical_columns
```

Out[9]:
```
Index(['ID', 'Customer_ID', 'Month', 'Name', 'Age', 'SSN', 'Occupation',
       'Annual_Income', 'Num_of_Loan', 'Type_of_Loan',
       'Num_of_Delayed_Payment', 'Changed_Credit_Limit', 'Credit_Mix',
       'Outstanding_Debt', 'Credit_History_Age', 'Payment_of_Min_Amount',
       'Amount_invested_monthly', 'Payment_Behaviour', 'Monthly_Balance'],
      dtype='object')
```

In [10]:
```
numerical_columns
```

Out[10]:
```
Index(['Monthly_Inhand_Salary', 'Num_Bank_Accounts', 'Num_Credit_Card',
       'Interest_Rate', 'Delay_from_due_date', 'Num_Credit_Inquiries',
       'Credit_Utilization_Ratio', 'Total_EMI_per_month'],
      dtype='object')
```

In [11]:
```
df.sample()
```

Out[11]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Ann
80825	0x1ef97	CUS_0x7233	February	Vladimir Soldatkinp	14	#F%$D@*&8	Teacher	

Null Detection

In [12]:
```
df.isna().sum()
```

Out[12]:

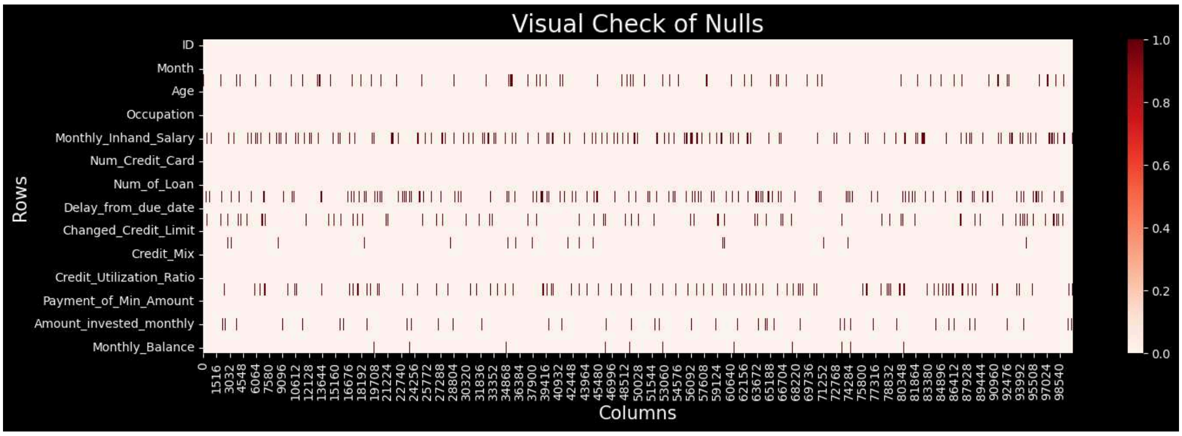
	0
ID	0
Customer_ID	0
Month	0
Name	9985
Age	0
SSN	0
Occupation	0
Annual_Income	0
Monthly_Inhand_Salary	15002
Num_Bank_Accounts	0
Num_Credit_Card	0
Interest_Rate	0
Num_of_Loan	0
Type_of_Loan	11408
Delay_from_due_date	0
Num_of_Delayed_Payment	7002
Changed_Credit_Limit	0
Num_Credit_Inquiries	1965
Credit_Mix	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

In [14]:

```
plt.figure(figsize=(15,5))
plt.style.use('dark_background')
sns.heatmap(df.isnull().T, cmap='Reds')
plt.title('Visual Check of Nulls',fontsize=20)
plt.xlabel('Columns',fontsize=15)
plt.ylabel('Rows',fontsize=15)
```

```
plt.tight_layout()
plt.show()
```



Perfect columns

- The features `ID`, `Customer_ID`, and `Month` exhibited no redundancy in the dataset.

1. Name

In [15]:
```
df.Name.dtype
```

Out[15]: dtype('O')

In [16]:
```
df.Name.nunique()
```

Out[16]: 10139

In [17]:
```
df['Name'] = df.groupby('Customer_ID')['Name'].ffill()
df['Name'] = df.groupby('Customer_ID')['Name'].bfill()
```

In [18]:
```
df.Name.isna().sum()
```

Out[18]: 0

In [19]:
```
df.Name.unique()
```

Out[19]:
```
array(['Aaron Maashoh', 'Rick Rothackerj', 'Langep', ...,
       'Chris Wickhamm', 'Sarah McBridec', 'Nicks'], dtype=object)
```

In [20]:
```
df.Name.nunique()
```

Out[20]: 10139

Insights

- The `Name` column contained some null values, which have now been filled.

2. Age

> After cleaning feature name is 'age'

In [21]:
```
df.Age.nunique()
```
Out[21]:
```
1788
```
In [22]:
```
df.Age.unique()
```
Out[22]:
```
array(['23', '-500', '28_', ..., '4808_', '2263', '1342'], dtype=object)
```
In [23]:
```
df.Age.isna().sum()
```
Out[23]:
```
0
```
In [24]:
```
def clean_age(value):
    # Replace '-' and non-numeric characters with NaN
    if isinstance(value, str) and not re.match(r'^\d+$', value):
        return np.nan
    return value

# Apply cleaning to the 'Age' column
df['Age'] = df['Age'].apply(clean_age)
# Convert 'Age' column to numeric, coercing errors to NaN
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Fill missing values with mode for each 'Name'
#df['Age'] = df.groupby('Name')['Age'].transform(lambda x: x.fillna(x.mode()[0]
```
In [25]:
```
df.Age.unique() ,  df.Age.dtype
```
Out[25]:
```
(array([  23.,   nan,   28., ..., 6476., 2263., 1342.]), dtype('float64'))
```
In [26]:
```
def fill_mode(series):
    mode_value = series.mode()
    if not mode_value.empty:   ## if mode_value is not none: also works
        return mode_value[0]
    # chances of bimodal values here we consider the first value since year of h
    else:
        return np.nan
```
In [27]:
```
df['age'] = df.groupby('Customer_ID')['Age'].transform(fill_mode)
```
In [28]:
```
df.age.nunique() , df.age.unique() , df.age.dtype
```
Out[28]:
```
(43,
 array([23., 28., 34., 55., 21., 31., 30., 44., 40., 33., 35., 39., 37.,
        20., 46., 26., 41., 32., 48., 43., 22., 36., 16., 18., 42., 19.,
        15., 27., 38., 14., 25., 45., 47., 17., 53., 24., 54., 29., 49.,
        51., 50., 52., 56.]),
 dtype('float64'))
```
In [29]:
```
df['age'] = df['age'].astype(int)
```
In [30]:
```
df.age.min() , df.age.max()
```
Out[30]:
```
(14, 56)
```
In [31]:
```
df['age'] = df['age'].astype(int)
```

In [32]:
```
df.groupby(['Customer_ID','Name'])['Age'].unique()
```

Out[32]:

Customer_ID	Name	Age
CUS_0x1000	Alistair Barrf	[17.0, nan, 18.0]
CUS_0x1009	Arunah	[25.0, 26.0]
CUS_0x100b	Shirboni	[18.0, 19.0]
CUS_0x1011	Schneyerh	[nan, 44.0]
CUS_0x1013	Cameront	[43.0, 44.0, nan]
...	...	...
CUS_0xff3	Somervilled	[55.0]
CUS_0xff4	Poornimaf	[36.0, nan, 37.0]
CUS_0xff6	Shieldsb	[18.0, 19.0]
CUS_0xffc	Brads	[17.0, 18.0]
CUS_0xffd	Damouniq	[29.0, nan, 30.0]

12500 rows × 1 columns

dtype: object

In [33]:
```
df.groupby(['Customer_ID','Name'])['age'].unique()
```

Out[33]:

		age
Customer_ID	**Name**	
CUS_0x1000	**Alistair Barrf**	[17]
CUS_0x1009	**Arunah**	[26]
CUS_0x100b	**Shirboni**	[18]
CUS_0x1011	**Schneyerh**	[44]
CUS_0x1013	**Cameront**	[44]
...	**...**	...
CUS_0xff3	**Somervilled**	[55]
CUS_0xff4	**Poornimaf**	[37]
CUS_0xff6	**Shieldsb**	[19]
CUS_0xffc	**Brads**	[17]
CUS_0xffd	**Damouniq**	[29]

12500 rows × 1 columns

dtype: object

In [34]:
```
df.age.isna().sum()
```

Out[34]: 0

In [35]:
```
df.drop(columns=['Age'],inplace=True)
```

Insights

- The `Age` feature contained many null and irrelevant values that were affecting the data type. These were addressed by filling in missing values with the most frequent mode. In cases where the mode was bimodal, the first mode value was used.
- The treated feature is `age` and original messed feature has been dropped.

3. Social-Security-Number

In [36]:
```
df.sample(3)
```

Out[36]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mo
5697	0x3763	CUS_0x541	February	Paul Carrels	313-45-4006	Writer	136680.12	
14070	0x6870	CUS_0x1f8c	July	Dolanl	745-18-8559	Musician	106418.67	
45227	0x11f01	CUS_0x5128	April	Moonz	273-66-6843	Entrepreneur	36471.78	

In [37]:
```
df.SSN.dtype
```
Out[37]: dtype('O')

In [38]:
```
df.SSN.nunique()
```
Out[38]: 12501

In [39]:
```
df[df.SSN=='#F%$D@*&8']['SSN'].count()
```
Out[39]: 5572

In [40]:
```
df['SSN'] = df['SSN'].replace(['#F%$D@*&8'], np.nan)
```

In [41]:
```
df['SSN'].isna().sum()
```
Out[41]: 5572

In [42]:
```
df['SSN'] = df.groupby('Customer_ID')['SSN'].ffill()
df['SSN'] = df.groupby('Customer_ID')['SSN'].bfill()
```

In [43]:
```
df['SSN'].isna().sum()
```
Out[43]: 0

In [44]:
```
df[df['Name']=='Clara Ferreira-Marquesi']
```

Out[44]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mo
1008	0x1bea	CUS_0x3fbf	January	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42_	
1009	0x1beb	CUS_0x3fbf	February	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	
1010	0x1bec	CUS_0x3fbf	March	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	
1011	0x1bed	CUS_0x3fbf	April	Clara Ferreira-Marquesi	646-12-1414	_______	27796.42	
1012	0x1bee	CUS_0x3fbf	May	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	
1013	0x1bef	CUS_0x3fbf	June	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	
1014	0x1bf0	CUS_0x3fbf	July	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	
1015	0x1bf1	CUS_0x3fbf	August	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	

Insights

- The `SSN` (social security Number) column had some Irrevalant data and it has been treated.

4. Occupation

In [45]:
```
df.Occupation.dtype , df.Occupation.nunique()
```

Out[45]:
```
(dtype('O'), 16)
```

In [46]:
```
df.Occupation.unique()
```

Out[46]:
```
array(['Scientist', '_______', 'Teacher', 'Engineer', 'Entrepreneur',
       'Developer', 'Lawyer', 'Media_Manager', 'Doctor', 'Journalist',
       'Manager', 'Accountant', 'Musician', 'Mechanic', 'Writer',
       'Architect'], dtype=object)
```

In [47]:
```
df[df.Occupation=='_______']['ID'].count()
```

Out[47]:
```
7062
```

In [48]:
```
df['Occupation'] = df['Occupation'].replace(['_______'], np.nan)
```

In [49]:
```
df['Occupation'] = df.groupby('Customer_ID')['Occupation'].ffill()
df['Occupation'] = df.groupby('Customer_ID')['Occupation'].bfill()
```

In [50]:
```
df.Occupation.isna().sum()
```

Out[50]:
```
0
```

In [51]:
```
df.Occupation.nunique() , df.Occupation.unique()
```

Out[51]:
```
(15,
 array(['Scientist', 'Teacher', 'Engineer', 'Entrepreneur', 'Developer',
        'Lawyer', 'Media_Manager', 'Doctor', 'Journalist', 'Manager',
        'Accountant', 'Musician', 'Mechanic', 'Writer', 'Architect'],
       dtype=object))
```

Insights

- The `Occupation` had some irrelavant data and it removed and replaced with relavant values.

5. Annual_Income

In [52]:
```
df.sample()
```

Out[52]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mont
48323	0x13125	CUS_0x51fe	April	C.i	871-93-0711	Developer	31752.1	

In [53]:
```
df.Annual_Income.dtype
```

Out[53]:
```
dtype('O')
```

In [54]:
```
# Replace underscores with NaN in the Annual_Income column
df['Annual_Income'] = df['Annual_Income'].replace(r'_', np.nan, regex=True)

# Convert the column to numeric, coercing errors to NaN
df['Annual_Income'] = pd.to_numeric(df['Annual_Income'], errors='coerce')
```

In [55]:
```
df['Annual_Income'] = df.groupby('Customer_ID')['Annual_Income'].ffill()
df['Annual_Income'] = df.groupby('Customer_ID')['Annual_Income'].bfill()
```

In [56]:
```
df.Annual_Income.dtype
```

Out[56]:
```
dtype('float64')
```

In [57]:
```
df.Annual_Income.nunique() , df.Annual_Income.unique()
```

Out[57]:
```
(13437,
 array([ 19114.12,  34847.84, 143162.64, ...,  37188.1 ,  20002.88,
         39628.99]))
```

In [58]:
```
df[df['Customer_ID']=='CUS_0x4c96']
```

Out[58]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	M
45744	0x1220a	CUS_0x4c96	January	Ann Saphirw	961-65-0909	Teacher	99868.83	
45745	0x1220b	CUS_0x4c96	February	Ann Saphirw	961-65-0909	Teacher	99868.83	
45746	0x1220c	CUS_0x4c96	March	Ann Saphirw	961-65-0909	Teacher	99868.83	
45747	0x1220d	CUS_0x4c96	April	Ann Saphirw	961-65-0909	Teacher	99868.83	
45748	0x1220e	CUS_0x4c96	May	Ann Saphirw	961-65-0909	Teacher	99868.83	
45749	0x1220f	CUS_0x4c96	June	Ann Saphirw	961-65-0909	Teacher	99868.83	
45750	0x12210	CUS_0x4c96	July	Ann Saphirw	961-65-0909	Teacher	99868.83	
45751	0x12211	CUS_0x4c96	August	Ann Saphirw	961-65-0909	Teacher	99868.83	

Insights

- The `Annual_Income` feature, initially in object datatype with some irrelevant data, has been cleaned and converted to the appropriate numeric format.

6. Monthly_Inhand_Salary

In [59]:
```
df['Monthly_Inhand_Salary'].dtype
```

Out[59]: dtype('float64')

In [60]:
```
df['Monthly_Inhand_Salary'].isna().sum()
```

Out[60]: 15002

In [61]:
```
df['Monthly_Inhand_Salary'] = df.groupby('Customer_ID')['Monthly_Inhand_Salary']
df['Monthly_Inhand_Salary'] = df.groupby('Customer_ID')['Monthly_Inhand_Salary']
```

In [62]:
```
df['Monthly_Inhand_Salary'].isna().sum()
```

Out[62]: 0

In [63]:
```
df.head(8)
```

Out[63]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthl
0	0x1602	CUS_0xd40	January	Aaron Maashoh	821-00-0265	Scientist	19114.12	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	821-00-0265	Scientist	19114.12	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	821-00-0265	Scientist	19114.12	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	821-00-0265	Scientist	19114.12	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	821-00-0265	Scientist	19114.12	
5	0x1607	CUS_0xd40	June	Aaron Maashoh	821-00-0265	Scientist	19114.12	
6	0x1608	CUS_0xd40	July	Aaron Maashoh	821-00-0265	Scientist	19114.12	
7	0x1609	CUS_0xd40	August	Aaron Maashoh	821-00-0265	Scientist	19114.12	

Insights

- The feature `Monthly_Inhand_Salary` had some null values and it has been filled with the appropriate values.

7. Num_Bank_Accounts

In [64]:
```
df.Num_Bank_Accounts.dtype
```
Out[64]:
```
dtype('int64')
```
In [65]:
```
df.Num_Bank_Accounts.isna().sum()
```
Out[65]:
```
0
```
In [66]:
```
df.Num_Bank_Accounts.nunique()
```
Out[66]:
```
943
```
In [67]:
```
df['Num_Bank_Accounts'] = df['Num_Bank_Accounts'].replace(-1,0)
```
In [68]:
```
df['Num_Bank_Acc'] = df.groupby('Customer_ID')['Num_Bank_Accounts'].transform(fi
```
In [69]:
```
df[df['Name']=='Thomass'][['SSN','Name','Month','Num_Bank_Accounts','Num_Bank_Ac
```

Out[69]:

	SSN	Name	Month	Num_Bank_Accounts	Num_Bank_Acc
9488	739-35-4103	Thomass	January	5	5
9489	739-35-4103	Thomass	February	5	5
9490	739-35-4103	Thomass	March	5	5
9491	739-35-4103	Thomass	April	5	5
9492	739-35-4103	Thomass	May	5	5
9493	739-35-4103	Thomass	June	5	5
9494	739-35-4103	Thomass	July	5	5
9495	739-35-4103	Thomass	August	5	5
56728	733-72-3818	Thomass	January	6	6
56729	733-72-3818	Thomass	February	6	6
56730	733-72-3818	Thomass	March	6	6
56731	733-72-3818	Thomass	April	6	6
56732	733-72-3818	Thomass	May	6	6
56733	733-72-3818	Thomass	June	1318	6
56734	733-72-3818	Thomass	July	6	6
56735	733-72-3818	Thomass	August	6	6
72368	825-63-0662	Thomass	January	8	8
72369	825-63-0662	Thomass	February	8	8
72370	825-63-0662	Thomass	March	8	8
72371	825-63-0662	Thomass	April	8	8
72372	825-63-0662	Thomass	May	8	8
72373	825-63-0662	Thomass	June	8	8
72374	825-63-0662	Thomass	July	8	8
72375	825-63-0662	Thomass	August	8	8

In [70]:

```
df['Num_Bank_Acc'].min() , df['Num_Bank_Acc'].max()
```

Out[70]: (0, 10)

In [71]:

```
df.drop(columns=['Num_Bank_Accounts'],inplace=True)
```

Insight

- The feature `Num_Bank_Accounts` had many irrelavant data and those have been filled with appropriate values.
- The New feature named `Num_Bank_Acc` was created

8. Num_Credit_Card

In [72]:
```
df.sample(3)
```

Out[72]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
19614	0x88ec	CUS_0xbabe	July	Sineadf	547-62-1545	Mechanic	19966.510
77861	0x1de37	CUS_0x72f3	June	Nick Brownl	054-62-4543	Media_Manager	95111.100
68753	0x1a8db	CUS_0x26eb	February	Jessica Toonkeld	070-52-7651	Mechanic	13999.185

In [73]:
```
df['Num_Credit_Card'].dtype
```

Out[73]:
```
dtype('int64')
```

In [74]:
```
df['Num_Credit_Card'].isna().sum()
```

Out[74]:
```
0
```

In [75]:
```
df['Num_Credit_Card'].nunique() , df['Num_Credit_Card'].unique()
```

Out[75]:
```
(1179, array([   4, 1385,    5, ...,  955, 1430,  679]))
```

In [76]:
```
df['No_of_Credit_Card'] = df.groupby('Customer_ID')['Num_Credit_Card'].transform
```

In [77]:
```
df[df.Name=='Dolano']
```

Out[77]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	M
66464	0x19b72	CUS_0x7666	January	Dolano	922-59-5950	Entrepreneur	113284.84	
66465	0x19b73	CUS_0x7666	February	Dolano	922-59-5950	Entrepreneur	113284.84	
66466	0x19b74	CUS_0x7666	March	Dolano	922-59-5950	Entrepreneur	113284.84	
66467	0x19b75	CUS_0x7666	April	Dolano	922-59-5950	Entrepreneur	113284.84	
66468	0x19b76	CUS_0x7666	May	Dolano	922-59-5950	Entrepreneur	113284.84	
66469	0x19b77	CUS_0x7666	June	Dolano	922-59-5950	Entrepreneur	113284.84	
66470	0x19b78	CUS_0x7666	July	Dolano	922-59-5950	Entrepreneur	113284.84	
66471	0x19b79	CUS_0x7666	August	Dolano	922-59-5950	Entrepreneur	113284.84	

In [78]:

```
df.drop(columns=['Num_Credit_Card'],inplace=True)
```

In [79]:

```
df.No_of_Credit_Card.dtype , df.No_of_Credit_Card.nunique() , df.No_of_Credit_Ca
```

Out[79]:

```
(dtype('int64'), 12, array([ 4,  5,  1,  7,  6,  8,  3,  9,  2, 10, 11,  0]))
```

In [80]:

```
df.No_of_Credit_Card.min() , df.No_of_Credit_Card.max()
```

Out[80]: (0, 11)

In [81]:
```
df.sample()
```

Out[81]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	M
18822	0x8448	CUS_0x8cc9	July	Cezaryy	340-05-3601	Media_Manager	8953.055	

Insights

- The feature `Num_Credit_Card` had many irrelavant data and it has been filled with appropriate values.
- New feature `No_of_Credit_Card` has created with perfect values.

9. Interest_Rate

In [82]:
```
df.Interest_Rate.dtype , df.Interest_Rate.nunique() , df.Interest_Rate.unique()
```

Out[82]: (dtype('int64'), 1750, array([3,   6,   8, ..., 1347, 387, 5729]))

In [83]:
```
df.Interest_Rate.isna().sum()
```

Out[83]: 0

In [84]:
```
df['interest_rate'] = df.groupby('Customer_ID')['Interest_Rate'].transform(fill_
```

In [85]:
```
df[df.Name=='Anna Yukhananovd']
```

Out[85]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
23512	0x9fc6	CUS_0x2fab	January	Anna Yukhananovd	411-00-6543	Manager	75804.94
23513	0x9fc7	CUS_0x2fab	February	Anna Yukhananovd	411-00-6543	Manager	75804.94
23514	0x9fc8	CUS_0x2fab	March	Anna Yukhananovd	411-00-6543	Manager	75804.94
23515	0x9fc9	CUS_0x2fab	April	Anna Yukhananovd	411-00-6543	Manager	75804.94
23516	0x9fca	CUS_0x2fab	May	Anna Yukhananovd	411-00-6543	Manager	75804.94
23517	0x9fcb	CUS_0x2fab	June	Anna Yukhananovd	411-00-6543	Manager	75804.94
23518	0x9fcc	CUS_0x2fab	July	Anna Yukhananovd	411-00-6543	Manager	75804.94
23519	0x9fcd	CUS_0x2fab	August	Anna Yukhananovd	411-00-6543	Manager	75804.94

In [86]:
```
df['interest_rate'].min() , df['interest_rate'].max()
```

Out[86]: (1, 34)

In [87]:
```
df.drop(columns=['Interest_Rate'],inplace=True)
```

In [88]:
```
df.sample(3)
```

Out[88]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	I
3428	0x2a16	CUS_0x4d26	May	Solarinam	133-18-6855	Architect	20978.49	
67554	0x1a1d4	CUS_0xb0f5	March	Freilichi	203-26-1160	Mechanic	20730.55	
1953	0x2173	CUS_0x4eb4	February	Michael Avokf	955-76-5444	Journalist	30197.28	

Insight

- The feature `Interest_Rate` had many irrelavant data and those have been filled with appropriate values.
- The New feature named `interest_rate` was created

10.Num_of_Loan

In [89]:
```
df.Num_of_Loan.dtype , df.Num_of_Loan.nunique() , df.Num_of_Loan.unique()
```

Out[89]:
```
(dtype('O'),
 434,
 array(['4', '1', '3', '967', '-100', '0', '0_', '2', '3_', '2_', '7', '5',
        '5_', '6', '8', '8_', '9', '9_', '4_', '7_', '1_', '1464', '6_',
        '622', '352', '472', '1017', '945', '146', '563', '341', '444',
        '720', '1485', '49', '737', '1106', '466', '728', '313', '843',
        '597_', '617', '119', '663', '640', '92_', '1019', '501', '1302',
        '39', '716', '848', '931', '1214', '186', '424', '1001', '1110',
        '1152', '457', '1433', '1187', '52', '1480', '1047', '1035',
        '1347_', '33', '193', '699', '329', '1451', '484', '132', '649',
        '995', '545', '684', '1135', '1094', '1204', '654', '58', '348',
        '614', '1363', '323', '1406', '1348', '430', '153', '1461', '905',
        '1312', '1424', '1154', '95', '1353', '1228', '819', '1006', '795',
        '359', '1209', '590', '696', '1185_', '1465', '911', '1181', '70',
        '816', '1369', '143', '1416', '455', '55', '1096', '1474', '420',
        '1131', '904', '89', '1259', '527', '1241', '449', '983', '418',
        '319', '23', '238', '638', '138', '235_', '280', '1070', '1484',
        '274', '494', '1459_', '404', '1354', '1495', '1391', '601',
        '1313', '1319', '898', '231', '752', '174', '961', '1046', '834',
        '284', '438', '288', '1463', '1151', '719', '198', '1015', '855',
        '841', '392', '1444', '103', '1320_', '745', '172', '252', '630_',
        '241', '31', '405', '1217', '1030', '1257', '137', '157', '164',
        '1088', '1236', '777', '1048', '613', '330', '1439', '321', '661',
        '952', '939', '562', '1202', '302', '943', '394', '955', '1318',
        '936', '781', '100', '1329', '1365', '860', '217', '191', '32',
        '282', '351', '1387', '757', '416', '833', '359_', '292', '1225_',
        '1227', '639', '859', '243', '267', '510', '332', '996', '597',
        '311', '492', '820', '336', '123', '540', '131_', '1311_', '1441',
        '895', '891', '50', '940', '935', '596', '29', '1182', '1129_',
        '1014', '251', '365', '291', '1447', '742', '1085', '148', '462',
        '832', '881', '1225', '1412', '785_', '1127', '910', '538', '999',
        '733', '101', '237', '87', '659', '633', '387', '447', '629',
        '831', '1384', '773', '621', '1419', '289', '143_', '285', '1393',
        '1131_', '27_', '1359', '1482', '1189', '1294', '201', '579',
        '814', '141', '1320', '581', '1171_', '295', '290', '433', '679',
        '1040', '1054', '1430', '1023', '1077', '1457', '1150', '701',
        '1382', '889', '437', '372', '1222', '126', '1159', '868', '19',
        '1297', '227_', '190', '809', '1216', '1074', '571', '520', '1274',
        '1340', '991', '316', '697', '926', '873', '1002', '378_', '65',
        '875', '867', '548', '652', '1372', '606', '1036', '1300', '17',
        '1178', '802', '1219_', '1271', '1137', '1496', '439', '196',
        '636', '192', '228', '1053', '229', '753', '1296', '1371', '254',
        '863', '464', '515', '838', '1160', '1289', '1298', '799', '182',
        '574', '527_', '242', '415', '869', '958', '54', '1265', '656',
        '275', '778', '208', '147', '350', '507', '463', '497', '1129',
        '927', '653', '662', '529', '635', '1027_', '897', '1039', '227',
        '1345', '924', '696_', '1279', '546', '1112', '1210', '526', '300',
        '1103', '504', '136', '1400', '78', '686', '1091', '344', '215',
        '84', '628', '1470', '968', '1478', '83', '1196', '1307', '1132_',
        '1008', '917', '657', '56', '18', '41', '801', '978', '216', '349',
        '966'], dtype=object))
```

In [90]:
```python
def calculate_num_of_loans(type_of_loan):
    if pd.isna(type_of_loan) or type_of_loan.strip() == "":
        return 0
    else:
        return len(type_of_loan.split(','))
```

In [91]:
```python
df['Num_of_Loan'] = df['Type_of_Loan'].apply(calculate_num_of_loans)
```

In [92]:
```
df['Num_of_Loan'].isna().sum()
```

Out[92]:
```
0
```

In [93]:
```
df['Num_of_Loan'].nunique() , df['Num_of_Loan'].unique()
```

Out[93]:
```
(10, array([4, 1, 3, 0, 2, 7, 5, 6, 8, 9]))
```

In [94]:
```
df['Num_of_Loan'].min() , df['Num_of_Loan'].max()
```

Out[94]:
```
(0, 9)
```

In [95]:
```
df[df.Name=='Barri']
```

Out[95]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mo
59096	0x17046	CUS_0x44e9	January	Barri	482-45-6373	Architect	19656.96	
59097	0x17047	CUS_0x44e9	February	Barri	482-45-6373	Architect	19656.96	
59098	0x17048	CUS_0x44e9	March	Barri	482-45-6373	Architect	19656.96	
59099	0x17049	CUS_0x44e9	April	Barri	482-45-6373	Architect	19656.96	
59100	0x1704a	CUS_0x44e9	May	Barri	482-45-6373	Architect	19656.96	
59101	0x1704b	CUS_0x44e9	June	Barri	482-45-6373	Architect	19656.96	
59102	0x1704c	CUS_0x44e9	July	Barri	482-45-6373	Architect	19656.96	
59103	0x1704d	CUS_0x44e9	August	Barri	482-45-6373	Architect	19656.96	
63560	0x18a6e	CUS_0xb47f	January	Barri	429-78-0806	Scientist	30773.87	
63561	0x18a6f	CUS_0xb47f	February	Barri	429-78-0806	Scientist	30773.87	
63562	0x18a70	CUS_0xb47f	March	Barri	429-78-0806	Scientist	30773.87	
63563	0x18a71	CUS_0xb47f	April	Barri	429-78-0806	Scientist	30773.87	

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mo
63564	0x18a72	CUS_0xb47f	May	Barri	429-78-0806	Scientist	30773.87	
63565	0x18a73	CUS_0xb47f	June	Barri	429-78-0806	Scientist	30773.87	
63566	0x18a74	CUS_0xb47f	July	Barri	429-78-0806	Scientist	30773.87	
63567	0x18a75	CUS_0xb47f	August	Barri	429-78-0806	Scientist	30773.87	

Insight

- The feature `Num_of_Loan` had many irrelavant data and those have been filled with appropriate values of length of the Type_of_Loan.

11. Type_of_Loan

In [96]:
```
df.Type_of_Loan.dtype ,df.Type_of_Loan.isna().sum()
```

Out[96]:
```
(dtype('O'), 11408)
```

In [97]:
```
df.Type_of_Loan.nunique() , df.Type_of_Loan.unique()
```

Out[97]:
```
(6260,
 array(['Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan',
        'Credit-Builder Loan', 'Auto Loan, Auto Loan, and Not Specified',
        ..., 'Home Equity Loan, Auto Loan, Auto Loan, and Auto Loan',
        'Payday Loan, Student Loan, Mortgage Loan, and Not Specified',
        'Personal Loan, Auto Loan, Mortgage Loan, Student Loan, and Student Loa
n'],
       dtype=object))
```

In [98]:
```
df.Type_of_Loan.isna().sum()
```

Out[98]:
```
11408
```

In [99]:
```
df[df.Type_of_Loan.isna()].sample(10)
```

Out[99]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Inc
48504	0x13236	CUS_0x6ae2	January	Kyleq	875-53-5736	Writer	3757
19301	0x8717	CUS_0x5aa2	June	Badawym	121-25-0802	Entrepreneur	6378
66579	0x19c1d	CUS_0x8a3f	April	Soyoungh	108-22-7256	Architect	3947
25238	0xa9e0	CUS_0x2513	July	Leikav	470-35-3931	Architect	13199
38813	0xf96b	CUS_0x7d3b	June	Ryang	582-95-6911	Musician	7114
33937	0xdcdb	CUS_0x44e0	February	Sarah N.t	663-26-0839	Musician	9142
24777	0xa72f	CUS_0x657e	February	Caroline Valetkevitchp	945-22-8389	Media_Manager	5592
78150	0x1dfe8	CUS_0x8acd	July	Janeman Latuld	307-29-0236	Engineer	4560
65423	0x19555	CUS_0xdc8	August	Jim Finkleu	362-50-3134	Doctor	2816
69767	0x1aec9	CUS_0x47ed	August	Rujun Shent	157-48-1855	Writer	12752

In [100…

```
df[df.Type_of_Loan.isna()]['Num_of_Loan'].unique()
```

Out[100…

```
array([0])
```

In [101…

```
df['Type_of_Loan'] = df['Type_of_Loan'].fillna('No loan taken')
```

In [102…

```
df[df['Name']=='Stempelp'][['Name','SSN','Occupation','Num_of_Loan','Type_of_Loa
```

Out[102…

	Name	SSN	Occupation	Num_of_Loan	Type_of_Loan	interest_rate
512	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
513	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
514	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
515	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
516	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
517	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
518	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
519	Stempelp	878-90-6321	Scientist	7	Student Loan, Student Loan, Student Loan, Debt...	5
5336	Stempelp	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
5337	Stempelp	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
5338	Stempelp	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
5339	Stempelp	049-48-	Media_Manager	3	Debt Consolidation	11

	Name	SSN	Occupation	Num_of_Loan	Type_of_Loan	interest_rate
		0823			Loan, Payday Loan, and Mort...	
5340	Stempelp	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
5341	Stempelp	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
5342	Stempelp	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
5343	Stempelp	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
93696	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7
93697	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7
93698	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7
93699	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7
93700	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7
93701	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7
93702	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7
93703	Stempelp	027-76-6435	Accountant	2	Personal Loan, and Not Specified	7

In [103…
```
df.groupby('Type_of_Loan')[['Customer_ID']].count().sort_values(by='Customer_ID'
```

Out[103…

Type_of_Loan	Customer_ID
No loan taken	11408
Not Specified	1408
Credit-Builder Loan	1280
Personal Loan	1272
Debt Consolidation Loan	1264
Student Loan	1240
Payday Loan	1200
Mortgage Loan	1176
Auto Loan	1152
Home Equity Loan	1136
Personal Loan, and Student Loan	320
Not Specified, and Payday Loan	272

Insights

- Null values in `Type_of_Loan` has been filled 'No loan taken'.

12. Delay_from_due_date

In [104…
```
df['Delay_from_due_date'].isna().sum()
```

Out[104…
```
0
```

In [105…
```
df[df['Delay_from_due_date']<0]['Customer_ID'].count()
```

Out[105…
```
591
```

In [106…
```
df.loc[df['Delay_from_due_date'] < 0, 'Delay_from_due_date'] = 0
```

In [107…
```
df['delay_from_due_date'] = df.groupby('Customer_ID')['Delay_from_due_date'].tra
```

In [108…
```
df['Delay_from_due_date'].nunique() , df['Delay_from_due_date'].unique()
```

Out[108…
```
(68,
 array([ 3,  0,  5,  6,  8,  7, 13, 10,  4,  9,  1, 12, 11, 30, 31, 34, 27,
        14,  2, 16, 17, 15, 23, 22, 21, 18, 19, 52, 51, 48, 53, 26, 43, 28,
        25, 20, 47, 46, 49, 24, 61, 29, 50, 58, 45, 59, 55, 56, 57, 54, 62,
        65, 64, 67, 36, 41, 33, 32, 39, 44, 42, 60, 35, 38, 63, 40, 37, 66]))
```

In [109…
```
df['delay_from_due_date'].nunique() , df['delay_from_due_date'].unique()
```

Out[109…
```
(63,
 array([ 3,  8,  5,  0, 30, 11, 16,  4, 10, 23, 18, 51, 48, 25, 22, 52, 61,
        31, 53, 14, 17,  7, 49,  6, 13, 12, 59,  2, 20, 27, 57, 62, 15, 54,
        50, 41, 19, 24, 29,  1, 36,  9, 46, 60, 26, 33, 34, 28, 35, 38, 21,
        45, 42, 40, 47, 55, 32, 44, 39, 37, 43, 58, 56]))
```

In [110…
```
df['Delay_from_due_date'].min() , df['Delay_from_due_date'].max()
```

Out[110…
```
(0, 67)
```

In [111…
```
df['delay_from_due_date'].min() , df['delay_from_due_date'].max()
```

Out[111…
```
(0, 62)
```

In [112…
```
df.sample(3)
```

Out[112…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
77181	0x1da3b	CUS_0x8ddb	June	Josephe	255-39-8777	Media_Manager	55762.11
72367	0x1be05	CUS_0xba4c	August	Smith Douglasp	507-44-8168	Lawyer	81418.74
27847	0xb929	CUS_0x62b7	August	Timz	055-47-7711	Musician	41811.51

In [113…
```
df.drop(columns=['Delay_from_due_date'],inplace=True)
```

Insights

- Values less than 0 in `Delay_from_due_date` have been replaced with 'Zero'.
- On a holistic view, considering the average number of days of delayed payments, the mode has been deemed the most appropriate measure.
- The range of delayed payment intervals spans from 0 days to 62 days.

13. Num_of_Delayed_Payment

In [114…
```
df.Num_of_Delayed_Payment.dtype , df.Num_of_Delayed_Payment.isna().sum()
```

Out[114…
```
(dtype('O'), 7002)
```

In [115…
```
df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].replace({'-': np.nar

df['Num_of_Delayed_Payment'] = pd.to_numeric(df['Num_of_Delayed_Payment'], error
```

In [116…
```
df.Num_of_Delayed_Payment.dtype , df.Num_of_Delayed_Payment.isna().sum()
```

Out[116…
```
(dtype('float64'), 10368)
```

In [117…
```
df['No_of_delayed_payment'] = df.groupby('Customer_ID')['Num_of_Delayed_Payment'
```

In [118…
```
df.Num_of_Delayed_Payment.nunique() , df.Num_of_Delayed_Payment.min() , df.Num_c
```

Out[118…
```
(695, 0.0, 4397.0)
```

In [119…
```
df.No_of_delayed_payment.nunique() , df.No_of_delayed_payment.min() , df.No_of_d
```

Out[119…
```
(29, 0.0, 28.0)
```

In [120…
```
df.drop(columns=['Num_of_Delayed_Payment'],inplace=True)
```

In [121…
```
df['No_of_delayed_payment'] = df['No_of_delayed_payment'].astype(int)
```

Insights

- Irrelavant Values `Num_of_Delayed_Payment` have been replaced with 'Nulls'.
- On a holistic view, considering the average number of delayed payments, the mode has been deemed the most appropriate measure.
- The range of `No_of_delayed_payment` intervals spans from 0 days to 28 payments on an average.

14. Changed_Credit_Limit

- Represents the percentage change in credit card limit

In [122…
```
df.sample(3)
```

Out[122…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
69452	0x1acf2	CUS_0x8fbb	May	Aungb	792-39-3438	Developer	36984.76
3158	0x2880	CUS_0x683	July	LaCaprad	680-24-2269	Media_Manager	13603.41
47885	0x12e93	CUS_0x56cf	June	Henrye	310-05-9951	Developer	29960.97

In [123…
```
df.Changed_Credit_Limit.dtype
```

Out[123…
```
dtype('O')
```

In [124…
```
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].replace('_',np.nan)
df['Changed_Credit_Limit'] = pd.to_numeric(df['Changed_Credit_Limit'],errors='cc
```

In [125…
```
df.Changed_Credit_Limit.min() , df.Changed_Credit_Limit.max() , df.Changed_Credi
```

Out[125…
```
(-6.49, 36.97, 2091)
```

In [126…
```
df['Changed_Credit_Limit'] = df.groupby('Customer_ID')['Changed_Credit_Limit'].f
df['Changed_Credit_Limit'] = df.groupby('Customer_ID')['Changed_Credit_Limit'].b
```

In [127…
```
df.Changed_Credit_Limit.min() , df.Changed_Credit_Limit.max() , df.Changed_Credi
```

Out[127…
```
(-6.49, 36.97, 0)
```

Insights

- Irrelavant Values in `Changed_Credit_Limit` have been replaced with 'Nulls' and treated by filling the values.
- For percentage changes, it's quite common to have negative values. A negative percentage indicates a decrease in the credit card limit, while a positive percentage indicates an increase.

> Here's a quick overview of what the range of `-6.49` to `36.97` could represent:

- **Negative Values (-6.49 to 0)**: This range indicates a decrease in the credit card limit. For example, a value of `-0.01` could mean a decrease of 1% in the credit limit.
- **Positive Values (0 to 36.97)**: This range indicates an increase in the credit card limit. For example, a value of `36.97` could mean an increase of 36.97% in the credit limit.

- The range `-6.49` to `36.97` suggests that the changes span from a small decrease to a significant increase in credit card limits.

15. Num_Credit_Inquiries

In [128…
```
df.sample(3)
```

Out[128…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
89091	0x22005	CUS_0x8cf2	April	Jason Langex	260-53-2581	Lawyer	35572.08
67292	0x1a04a	CUS_0x4812	May	Edwardst	172-25-8187	Accountant	106790.32
83375	0x1fe85	CUS_0x4b5d	August	Jacobse	457-76-4147	Media_Manager	134499.84

In [129…
```
df.Num_Credit_Inquiries.dtype , df.Num_Credit_Inquiries.isna().sum()
```
Out[129…
```
(dtype('float64'), 1965)
```
In [130…
```
df['No_of_credit_inquiries'] = df.groupby('Customer_ID')['Num_Credit_Inquiries']
```
In [131…
```
df.No_of_credit_inquiries.dtype , df.No_of_credit_inquiries.isna().sum()
```
Out[131…
```
(dtype('float64'), 0)
```
In [132…
```
df['No_of_credit_inquiries'] = df['No_of_credit_inquiries'].astype(int)
```
In [133…
```
df.No_of_credit_inquiries.min() , df.No_of_credit_inquiries.max()
```
Out[133…
```
(0, 17)
```
In [134…
```
df[df.No_of_credit_inquiries==17].sample(3)
```
Out[134…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mc
6822	0x3df8	CUS_0x7943	July	Robinr	775-20-0390	Lawyer	10041990.00	
85581	0x20b73	CUS_0x365f	June	"Michael OBoyle"v	513-84-3765	Mechanic	19250.71	
42414	0x10e84	CUS_0xbaee	July	Karen Freifelds	015-50-7138	Journalist	66349.96	

In [135…
```
df.drop(columns=['Num_Credit_Inquiries'],inplace=True)
```
Insights

- Irrelavant Values `Num_Credit_Inquiries` have been filled with the mode in new feature called `No_of_credit_inquiries` , which was the most appropriate measure.
- The range of `No_of_credit_inquiries` intervals spans from 0 days to 17 inquires on an average.

16. Credit_Mix

In [136…
```
df.sample(3)
```

Out[136…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
24576	0xa602	CUS_0xbcfa	January	Viswanathau	556-18-4640	Journalist	58510.60
68965	0x1aa17	CUS_0x9d2f	June	Jenniferc	618-73-2642	Manager	131091.12
21176	0x9216	CUS_0x3a82	January	Niveditax	219-72-8338	Entrepreneur	20393.22

In [137…

```
df[df.Credit_Mix=='_'].count()[0]
```

Out[137…

```
20195
```

In [138…

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

In [139…

```
df['Credit_Mix'] = df.groupby('Customer_ID')['Credit_Mix'].ffill()
df['Credit_Mix'] = df.groupby('Customer_ID')['Credit_Mix'].bfill()
```

In [140…

```
df[df.Credit_Mix=='_'].count()[0]
```

Out[140…

```
0
```

In [141…

```
df[df.Name=='Bansalp']
```

Out[141…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mc
68704	0x1a892	CUS_0xae9e	January	Bansalp	345-91-4839	Developer	19540.67	
68705	0x1a893	CUS_0xae9e	February	Bansalp	345-91-4839	Developer	19540.67	
68706	0x1a894	CUS_0xae9e	March	Bansalp	345-91-4839	Developer	19540.67	
68707	0x1a895	CUS_0xae9e	April	Bansalp	345-91-4839	Developer	19540.67	
68708	0x1a896	CUS_0xae9e	May	Bansalp	345-91-4839	Developer	19540.67	
68709	0x1a897	CUS_0xae9e	June	Bansalp	345-91-4839	Developer	19540.67	
68710	0x1a898	CUS_0xae9e	July	Bansalp	345-91-4839	Developer	19540.67	
68711	0x1a899	CUS_0xae9e	August	Bansalp	345-91-4839	Developer	19540.67	

Insights

- Irrelavant Values in `Credit_Mix` have been replaced with 'Nulls' and then treated by filling the appropriate values.

17. Outstanding_Debt

In [142…
```
df.Outstanding_Debt.dtype
```
Out[142…
```
dtype('O')
```
In [143…
```
df.Outstanding_Debt.nunique() , df.Outstanding_Debt.unique()
```

Out[143…
```
(13178,
 array(['809.98', '605.03', '1303.01', ..., '3571.7_', '3571.7', '502.38'],
       dtype=object))
```

In [144…
```
df['Outstanding_Debt'] = df.Outstanding_Debt.replace(r'/d+_$',np.nan,regex=True)
df['Outstanding_Debt'] = pd.to_numeric(df['Outstanding_Debt'], errors='coerce')
```

In [145…
```
df.Outstanding_Debt.nunique() , df.Outstanding_Debt.unique()
```

Out[145…
```
(12203, array([ 809.98,  605.03, 1303.01, ...,  620.64, 3571.7 ,  502.38]))
```

In [146…
```
df['Outstanding_Debt'] = df.groupby('Customer_ID')['Outstanding_Debt'].ffill()
df['Outstanding_Debt'] = df.groupby('Customer_ID')['Outstanding_Debt'].bfill()
```

In [147…
```
df.Outstanding_Debt.nunique() , df.Outstanding_Debt.unique()
```

Out[147…
```
(12203, array([ 809.98,  605.03, 1303.01, ...,  620.64, 3571.7 ,  502.38]))
```

In [148…
```
df.Outstanding_Debt.min() , df.Outstanding_Debt.max()
```

Out[148…
```
(0.23, 4998.07)
```

In [149…
```
df.Outstanding_Debt.isna().sum()
```

Out[149…
```
0
```

Insights

- Irrelavant Values in `Outstanding_Debt` have been replaced with 'Nulls' and then treated by filling the appropriate values.

18. Credit_Utilization_Ratio

In [150…
```
df.Credit_Utilization_Ratio.dtype , df.Credit_Utilization_Ratio.isna().sum()
```

Out[150…
```
(dtype('float64'), 0)
```

Insights

- No Irrelavant Values in `Credit_Utilization_Ratio` have been found.

19. Credit_History_Age

- Represents the age of credit history of the person

In [151…
```
df.sample(3)
```

Out[151…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Mo
7704	0x4326	CUS_0xb018	January	Callusn	088-10-2997	Teacher	128189.37	
96406	0x24ae0	CUS_0x9f5c	July	Jonathan Gouldj	789-40-9271	Manager	60584.76	
93091	0x23775	CUS_0x29c0	April	Gilbert Kreijgerj	310-64-5275	Developer	64696.24	

In [152…

```
df.head(4)
```

Out[152…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthl
0	0x1602	CUS_0xd40	January	Aaron Maashoh	821-00-0265	Scientist	19114.12	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	821-00-0265	Scientist	19114.12	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	821-00-0265	Scientist	19114.12	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	821-00-0265	Scientist	19114.12	

In [153…

```
df.Credit_History_Age.isna().sum()
```

Out[153…

```
9030
```

In [154…

```
df['Credit_History_Age'] = df['Credit_History_Age'].replace('NA',np.nan)
```

In [155…

```
df['Credit_History_Age'] = df.groupby('Customer_ID')['Credit_History_Age'].ffill
df['Credit_History_Age'] = df.groupby('Customer_ID')['Credit_History_Age'].bfill
```

In [156…
```
df['cha'] = df.Credit_History_Age.str.split(' ')
df['cha1'] = df['cha'].apply(lambda x: x[0])
```

In [157…
```
df['cha'] , df['cha1']
```

Out[157…
```
(0         [22, Years, and, 1, Months]
 1         [22, Years, and, 1, Months]
 2         [22, Years, and, 3, Months]
 3         [22, Years, and, 4, Months]
 4         [22, Years, and, 5, Months]
                      ...
 99995     [31, Years, and, 6, Months]
 99996     [31, Years, and, 7, Months]
 99997     [31, Years, and, 8, Months]
 99998     [31, Years, and, 9, Months]
 99999    [31, Years, and, 10, Months]
 Name: cha, Length: 100000, dtype: object,
 0        22
 1        22
 2        22
 3        22
 4        22
          ..
 99995    31
 99996    31
 99997    31
 99998    31
 99999    31
 Name: cha1, Length: 100000, dtype: object)
```

In [158…
```
df['cha1'] = df.cha1.astype(int)
```

In [159…
```
df['credit_history_age'] = df.groupby('Customer_ID')['cha1'].transform(max)
```

In [160…
```
df[df.Customer_ID=='CUS_0xff3']
```

Out[160…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	M
5168	0x344a	CUS_0xff3	January	Somervilled	726-35-5322	Scientist	17032.785	
5169	0x344b	CUS_0xff3	February	Somervilled	726-35-5322	Scientist	17032.785	
5170	0x344c	CUS_0xff3	March	Somervilled	726-35-5322	Scientist	17032.785	
5171	0x344d	CUS_0xff3	April	Somervilled	726-35-5322	Scientist	17032.785	
5172	0x344e	CUS_0xff3	May	Somervilled	726-35-5322	Scientist	17032.785	
5173	0x344f	CUS_0xff3	June	Somervilled	726-35-5322	Scientist	17032.785	
5174	0x3450	CUS_0xff3	July	Somervilled	726-35-5322	Scientist	17032.785	
5175	0x3451	CUS_0xff3	August	Somervilled	726-35-5322	Scientist	17032.785	

In [161…

```
df.drop(columns=['cha','cha1','Credit_History_Age'],inplace=True)
```

In [162…

```
df.groupby('Customer_ID')['credit_history_age'].first()
```

Out[162...

	credit_history_age
Customer_ID	
CUS_0x1000	10
CUS_0x1009	31
CUS_0x100b	15
CUS_0x1011	15
CUS_0x1013	17
...	...
CUS_0xff3	17
CUS_0xff4	18
CUS_0xff6	24
CUS_0xffc	13
CUS_0xffd	18

12500 rows × 1 columns

dtype: int64

In [163...

```
df.sample(3)
```

Out[163...

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Moı
55916	0x15da2	CUS_0x79d3	May	Ros Krasnyh	541-86-2460	Doctor	14332.03	
44588	0x11b42	CUS_0x6e61	May	Herbst-Baylissq	974-44-7143	Teacher	44806.60	
90350	0x22764	CUS_0xa116	July	Jasong	591-43-0273	Journalist	18247.18	

Insights

- Irrelavant Values in `Credit_History_Age` has been replaced with appropriate values.
- We just need the ***years of Credit*** as Age, which has created in `credit_history_age`

- Note: If the customer's age is closer to his/her credit_History_Age it maybe due to 'Inherited_Credit_History'

20. Payment_of_Min_Amount

In [164…
```
df.Payment_of_Min_Amount.isna().sum()
```
Out[164…
```
0
```
In [165…
```
df.Payment_of_Min_Amount.nunique() , df.Payment_of_Min_Amount.unique()
```
Out[165…
```
(3, array(['No', 'NM', 'Yes'], dtype=object))
```
In [166…
```
df['Payment_of_Min_Amount'] = df['Payment_of_Min_Amount'].replace('NM','No')
```
In [167…
```
df.Payment_of_Min_Amount.isna().sum() , df.Payment_of_Min_Amount.nunique() , df.
```
Out[167…
```
(0, 2, array(['No', 'Yes'], dtype=object))
```

Insights

- Irrelavant Values in `Payment_of_Min_Amount` has been replaced with appropriate values.

In [168…
```
df.sample(3)
```
Out[168…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	M
7869	0x441b	CUS_0xc3ee	June	Dominic Lauy	971-78-9268	Architect	14439.41	
61606	0x17ef8	CUS_0xaf56	July	Herbertg	808-71-5249	Musician	15081.42	
32840	0xd66e	CUS_0x8fac	January	Tabassum Zakariav	019-79-1629	Doctor	103026.80	

21. Total_EMI_per_month

In [169…
```
df.Total_EMI_per_month.dtype , df.Total_EMI_per_month.isna().sum()
```
Out[169…
```
(dtype('float64'), 0)
```
In [170…
```
df.Total_EMI_per_month.nunique() , df.Total_EMI_per_month.unique()
```

Out[170…
```
(14950,
 array([4.95749492e+01, 1.88162146e+01, 2.46992320e+02, ...,
        1.21120000e+04, 3.51040226e+01, 5.86380000e+04]))
```

Insights

- No Irrelavant Values in `Total_EMI_per_month` have been found.

22. Amount_invested_monthly

In [171…
```
df.sample(3)
```

Out[171…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	M
65165	0x193d3	CUS_0x2444	June	Benn	983-43-5441	Teacher	113384.82	
33963	0xdd01	CUS_0x1cb8	April	Antoniolil	550-64-9594	Mechanic	53003.18	
30137	0xc697	CUS_0x804e	February	Laurence Fletcherz	424-56-2527	Teacher	67749.50	

In [172…
```
df.Amount_invested_monthly.dtype
```

Out[172…
```
dtype('O')
```

In [173…
```
df.loc[df['Amount_invested_monthly'] == '__10000__', 'Amount_invested_monthly']
```

In [174…
```
df['Amount_invested_monthly'] = pd.to_numeric(df['Amount_invested_monthly'], err
```

In [175…
```
df.Amount_invested_monthly.dtype
```

Out[175…
```
dtype('float64')
```

In [176…
```
df['Amount_invested_monthly'] = df['Amount_invested_monthly'].fillna(0)
```

In [177…
```
df['Amount_invested_monthly'].isna().sum()
```

Out[177…
```
0
```

Insights

- Irrelavant Values in `Amount_invested_monthly` has been filled with appropriate values.

23. Payment_Behaviour

In [178…
```
df.sample(3)
```

Out[178…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
23641	0xa087	CUS_0x17fe	February	Santaa	770-45-7126	Developer	31135.37
92190	0x2322c	CUS_0x49a9	July	Ryan Vlastelicax	290-78-0986	Journalist	31155.48
4209	0x2eab	CUS_0x627e	February	Forgioner	721-96-8087	Mechanic	72116.25

In [179…
```
df['Payment_Behaviour'] = df.Payment_Behaviour.replace('!@9#%8',np.nan)
```

In [180…
```
df['Payment_Behaviour'] = df.groupby('Customer_ID')['Payment_Behaviour'].ffill()
df['Payment_Behaviour'] = df.groupby('Customer_ID')['Payment_Behaviour'].bfill()
```

In [181…
```
df['Payment_Behaviour'].isna().sum()
```

Out[181…
```
0
```

Insights

- Irrelavant Values in `Payment_Behaviour` has been filled with appropriate values.

24. Monthly_Balance

In [182…
```
df.sample(3)
```

Out[182…

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income
95314	0x2447c	CUS_0x7e24	March	Kellyx	298-06-5346	Media_Manager	49854.92
67636	0x1a24e	CUS_0x2fe0	May	Anns	899-52-1176	Developer	8357.98
50125	0x13bb3	CUS_0x9469	June	Mutikanix	517-43-4607	Accountant	28582.04

In [183…

```
df['Monthly_Balance'] = pd.to_numeric(df['Monthly_Balance'],errors='coerce')
```

In [184…

```
df['Monthly_Balance'].isna().sum()
```

Out[184…

```
1209
```

In [185…

```
df['Monthly_Balance'] = df['Monthly_Balance'].fillna(0)
```

Insights

- Datatype of `Monthly_Balance` has be changed and filled null with 'Zero' as corresponding Monthly_Balance.

Restructuring the Data:

In [186…

```
rename_dict = {
    'Num_Bank_Acc': 'Num_Bank_Accounts',
    'No_of_Credit_Card': 'Num_Credit_Card',
    'interest_rate': 'Interest_Rate',
    'delay_from_due_date': 'Delay_from_due_date',
    'No_of_delayed_payment': 'Num_of_Delayed_Payment',
    'No_of_credit_inquiries': 'Num_Credit_Inquiries',
    'credit_history_age': 'Credit_History_Age',
    'age': 'Age'
}

df = df.rename(columns=rename_dict)


new_column_order = [
    'ID', 'Customer_ID', 'Month', 'Name', 'Age', 'SSN', 'Occupation', 'Annual_In
    'Monthly_Inhand_Salary', 'Num_Bank_Accounts', 'Num_Credit_Card', 'Interest_R
    'Num_of_Loan', 'Type_of_Loan', 'Delay_from_due_date', 'Num_of_Delayed_Paymen
    'Changed_Credit_Limit', 'Num_Credit_Inquiries', 'Credit_Mix', 'Outstanding_D
    'Credit_Utilization_Ratio', 'Credit_History_Age', 'Payment_of_Min_Amount',
```

```
    'Total_EMI_per_month', 'Amount_invested_monthly', 'Payment_Behaviour', 'Mont
]

df = df.reindex(columns=new_column_order)

df.tail(8)
```

Out[186…

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income
99992	0x25fe6	CUS_0x942c	January	Nicks	25	078-73-5990	Mechanic	39628.99
99993	0x25fe7	CUS_0x942c	February	Nicks	25	078-73-5990	Mechanic	39628.99
99994	0x25fe8	CUS_0x942c	March	Nicks	25	078-73-5990	Mechanic	39628.99
99995	0x25fe9	CUS_0x942c	April	Nicks	25	078-73-5990	Mechanic	39628.99
99996	0x25fea	CUS_0x942c	May	Nicks	25	078-73-5990	Mechanic	39628.99
99997	0x25feb	CUS_0x942c	June	Nicks	25	078-73-5990	Mechanic	39628.99
99998	0x25fec	CUS_0x942c	July	Nicks	25	078-73-5990	Mechanic	39628.99
99999	0x25fed	CUS_0x942c	August	Nicks	25	078-73-5990	Mechanic	39628.99

Saving the Cleaned Data

In [187…

```
df.to_csv('cleaned_credit_data.csv',index=False)
```

Data Exploration

Loading the cleaned data

In [188…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
from sklearn.preprocessing import MinMaxScaler
```

```
import warnings
warnings.filterwarnings('ignore')
```

In [189…
```
df = pd.read_csv('cleaned_credit_data.csv')
```

In [190…
```
pd.set_option('display.max_columns',50)
```

Exploratory Data Analysis

Non Graphical analysis

- Creating month numbers

In [191…
```
# Converting month names to month numbers
df['Month_Num'] = pd.to_datetime(df['Month'], format='%B').dt.month
df.sample(3)
```

Out[191…

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Incon
34933	0xe2af	CUS_0x4037	June	Tapinsha	33	022-20-2822	Engineer	23176251.(
99211	0x25b51	CUS_0x2954	April	Praveen Menoni	45	988-44-7575	Musician	22443.!
39136	0xfb52	CUS_0xaab7	January	Jason Langet	40	110-58-2517	Entrepreneur	103744.8

Info on Data

In [192…
```
# Checking the number of rows and columns
print(f"No of rows: {df.shape[0]:,} \nNo of columns: {df.shape[1]}")
```

```
No of rows: 100,000
No of columns: 28
```

In [193…
```
# Check all column names
df.columns
```

Out[193…

```
Index(['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',
       'Month_Num'],
      dtype='object')
```

In [194…

```
categorical_columns = df.select_dtypes(include=['object', 'category']).columns
numerical_columns = df.select_dtypes(include=['number']).columns
```

In [195…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 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                      100000 non-null  object 
 4   Age                       100000 non-null  int64  
 5   SSN                       100000 non-null  object 
 6   Occupation                100000 non-null  object 
 7   Annual_Income             100000 non-null  float64
 8   Monthly_Inhand_Salary     100000 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  int64  
 13  Type_of_Loan              100000 non-null  object 
 14  Delay_from_due_date       100000 non-null  int64  
 15  Num_of_Delayed_Payment    100000 non-null  int64  
 16  Changed_Credit_Limit      100000 non-null  float64
 17  Num_Credit_Inquiries      100000 non-null  int64  
 18  Credit_Mix                100000 non-null  object 
 19  Outstanding_Debt          100000 non-null  float64
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        100000 non-null  int64  
 22  Payment_of_Min_Amount     100000 non-null  object 
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   100000 non-null  float64
 25  Payment_Behaviour         100000 non-null  object 
 26  Monthly_Balance           100000 non-null  float64
 27  Month_Num                 100000 non-null  int32  
dtypes: float64(8), int32(1), int64(9), object(10)
memory usage: 21.0+ MB
```

Statistical Summary

In [196…

```
df.describe().T
```

Out[196…

	count	mean	std	min	
Age	100000.0	33.282000	1.076657e+01	14.000000	24.0(
Annual_Income	100000.0	178506.267553	1.440261e+06	7005.930000	19457.5(
Monthly_Inhand_Salary	100000.0	4198.771619	3.187494e+03	303.645417	1626.7(
Num_Bank_Accounts	100000.0	5.367840	2.592597e+00	0.000000	3.0(
Num_Credit_Card	100000.0	5.532720	2.067504e+00	0.000000	4.0(
Interest_Rate	100000.0	14.532080	8.741330e+00	1.000000	7.0(
Num_of_Loan	100000.0	3.532880	2.446356e+00	0.000000	2.0(
Delay_from_due_date	100000.0	21.050560	1.476119e+01	0.000000	10.0(
Num_of_Delayed_Payment	100000.0	13.261280	6.199080e+00	0.000000	9.0(
Changed_Credit_Limit	100000.0	10.389303	6.789784e+00	-6.490000	5.32
Num_Credit_Inquiries	100000.0	5.677760	3.827248e+00	0.000000	3.0(
Outstanding_Debt	100000.0	1426.220376	1.155129e+03	0.230000	566.07
Credit_Utilization_Ratio	100000.0	32.285173	5.116875e+00	20.000000	28.05
Credit_History_Age	100000.0	18.235920	8.313256e+00	0.000000	12.0(
Total_EMI_per_month	100000.0	1403.118217	8.306041e+03	0.000000	30.3(
Amount_invested_monthly	100000.0	178.363270	1.984724e+02	0.000000	58.32
Monthly_Balance	100000.0	397.684413	2.171320e+02	0.000000	267.87
Month_Num	100000.0	4.500000	2.291299e+00	1.000000	2.75

In [197…

```
df.describe(include=object)
```

Out[197…

	ID	Customer_ID	Month	Name	SSN	Occupation	Type_of_Loan	Cre(
count	100000	100000	100000	100000	100000	100000	100000	
unique	100000	12500	8	10139	12500	15	6261	
top	0x1602	CUS_0xd40	January	Langep	821-00-0265	Lawyer	No loan taken	St
freq	1	8	12500	48	8	7096	11408	

Duplicate Detection

In [198…

```
df[df.duplicated()]
```

Out[198…

ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_In

Insights

- No duplicates found

In [199…

```
# Number of unique values in each coluumn
print("No.of unique values in each column:")
print("-" * 35)
# Calculate the maximum column name length for formatting
max_col_name_length = max(len(col) for col in df.columns)

for col in df.columns:
    # Use ljust to align the column names and rjust for the counts
    print(f"{col.ljust(max_col_name_length)}: {str(df[col].nunique()).rjust(5)}"
```

```
No.of unique values in each column:
-----------------------------------
ID                      : 100000
Customer_ID             : 12500
Month                   :     8
Name                    : 10139
Age                     :    43
SSN                     : 12500
Occupation              :    15
Annual_Income           : 13437
Monthly_Inhand_Salary   : 13235
Num_Bank_Accounts       :    11
Num_Credit_Card         :    12
Interest_Rate           :    34
Num_of_Loan             :    10
Type_of_Loan            :  6261
Delay_from_due_date     :    63
Num_of_Delayed_Payment  :    29
Changed_Credit_Limit    :  3634
Num_Credit_Inquiries    :    18
Credit_Mix              :     3
Outstanding_Debt        : 12203
Credit_Utilization_Ratio: 99998
Credit_History_Age      :    34
Payment_of_Min_Amount   :     2
Total_EMI_per_month     : 14950
Amount_invested_monthly : 91048
Payment_Behaviour       :     6
Monthly_Balance         : 98790
Month_Num               :     8
```

In [200…

```
categorical_columns[1:]
```

Out[200…

```
Index(['Customer_ID', 'Month', 'Name', 'SSN', 'Occupation', 'Type_of_Loan',
       'Credit_Mix', 'Payment_of_Min_Amount', 'Payment_Behaviour'],
      dtype='object')
```

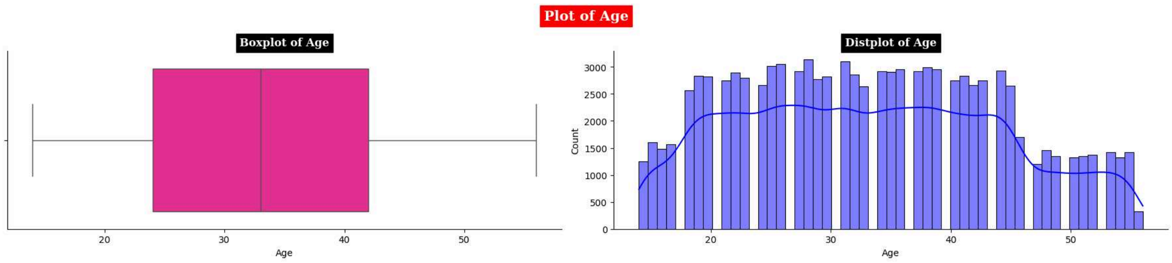
In [201…

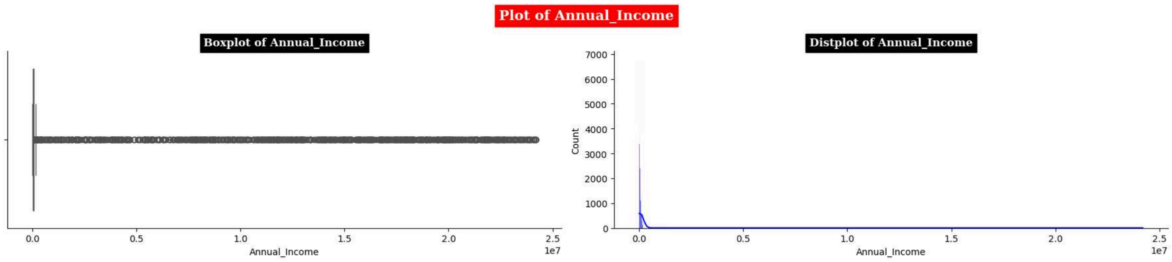
```
numerical_columns
```

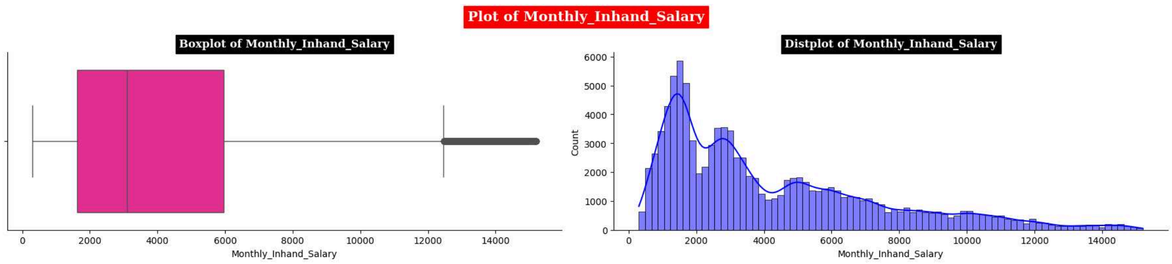
Out[201…
```
Index(['Age', 'Annual_Income', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts',
       'Num_Credit_Card', 'Interest_Rate', 'Num_of_Loan',
       'Delay_from_due_date', 'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Outstanding_Debt', 'Credit_Utilization_Ratio',
       'Credit_History_Age', 'Total_EMI_per_month', 'Amount_invested_monthly',
       'Monthly_Balance', 'Month_Num'],
      dtype='object')
```

Graphical analysis

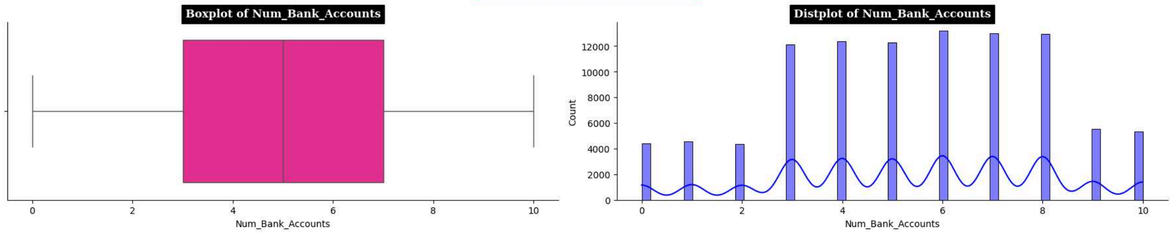
In [202…
```
plt.style.use('default')
plt.style.use('seaborn-bright')

for _,col in enumerate(numerical_columns):
    plt.figure(figsize=(18,4))
    plt.suptitle(f'Plot of {col}',fontsize=15,fontfamily='serif',
                 fontweight='bold',backgroundcolor='Red',color='w')
    plt.subplot(121)
    sns.boxplot(x=df[col],color='deeppink')
    plt.title(f'Boxplot of {col}',fontsize=12,fontfamily='serif',
              fontweight='bold',backgroundcolor='Black',color='w')
    plt.subplot(122)
    sns.histplot(x=df[col], kde=True,color='blue')
    plt.title(f'Distplot of {col}',fontsize=12,fontfamily='serif',
              fontweight='bold',backgroundcolor='Black',color='w')
    sns.despine()
    plt.tight_layout()
    plt.show()
```



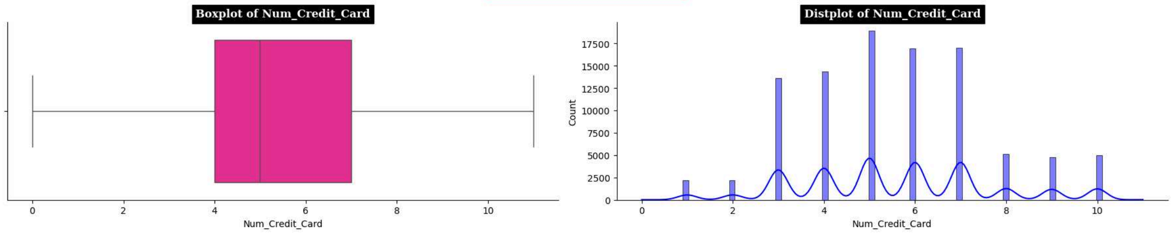




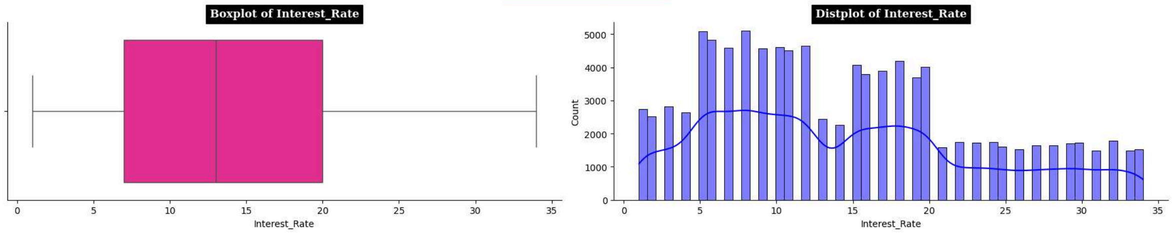
Plot of Num_Bank_Accounts



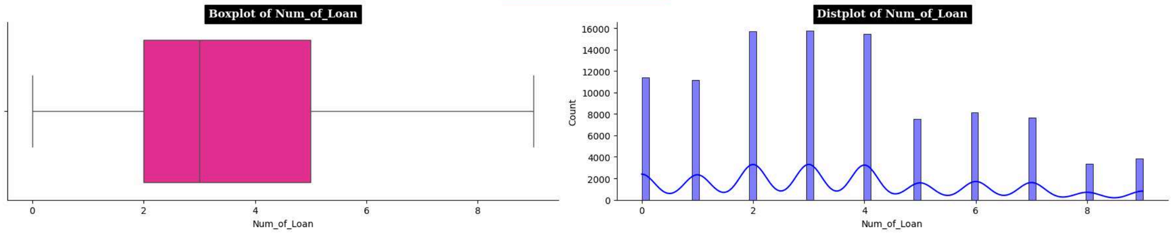
Plot of Num_Credit_Card



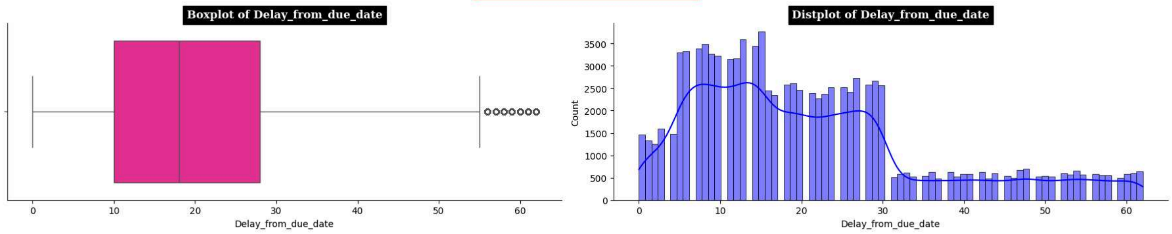
Plot of Interest_Rate



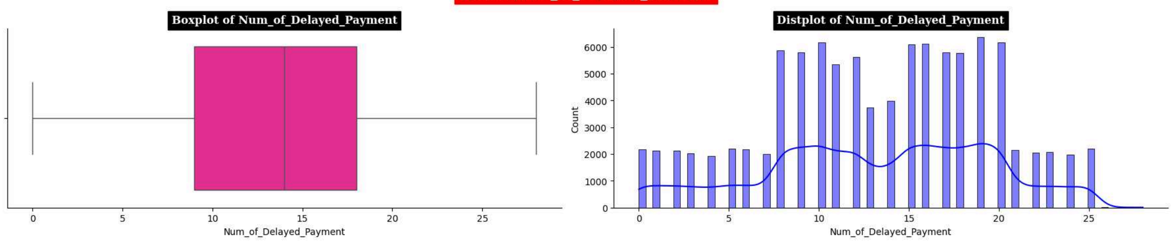
Plot of Num_of_Loan



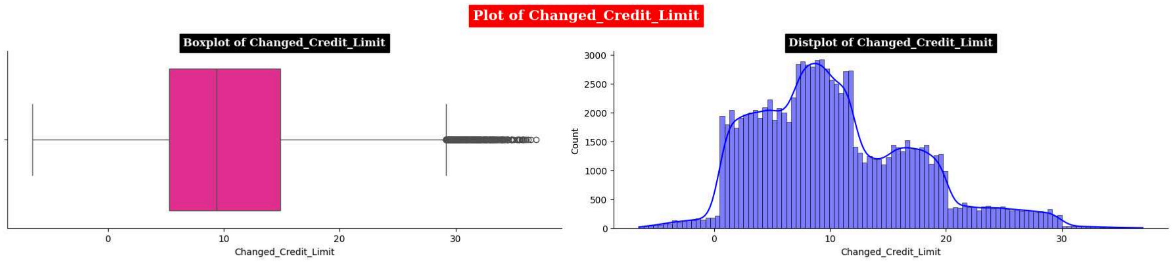
Plot of Delay_from_due_date

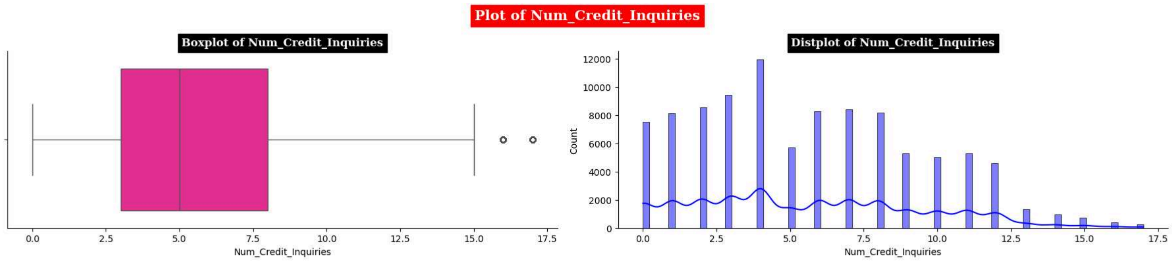


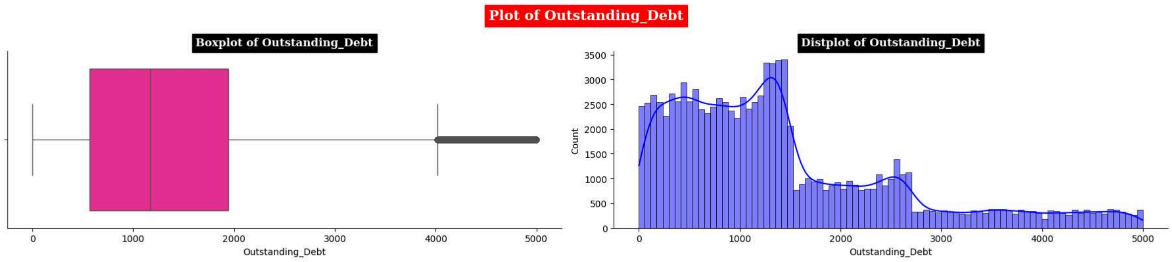
Plot of Num_of_Delayed_Payment

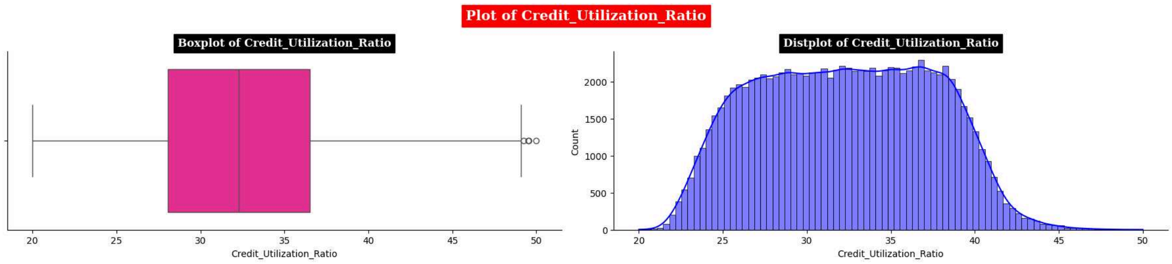


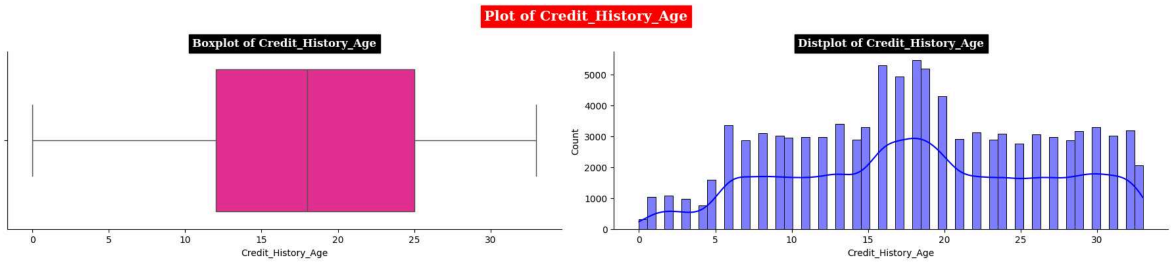
Plot of Changed_Credit_Limit

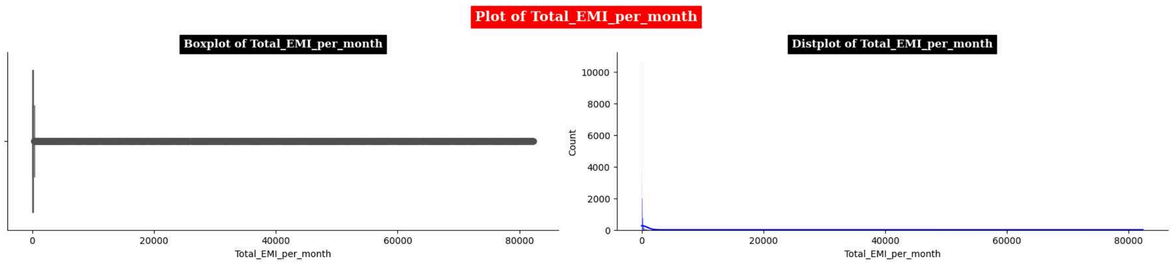


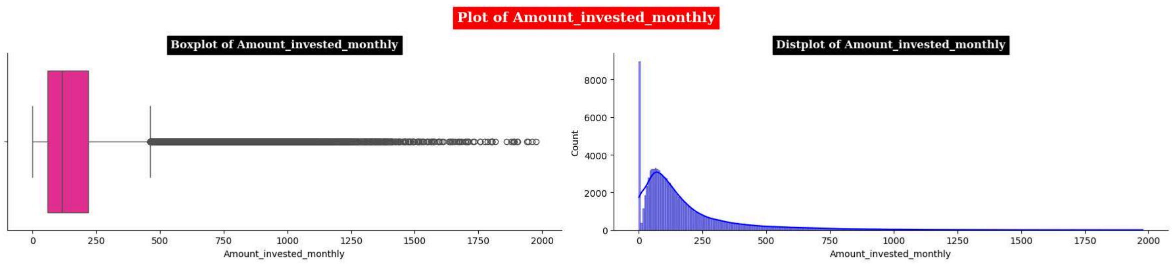

Plot of Num_Credit_Inquiries
Boxplot of Num_Credit_Inquiries
Distplot of Num_Credit_Inquiries
Num_Credit_Inquiries
Count

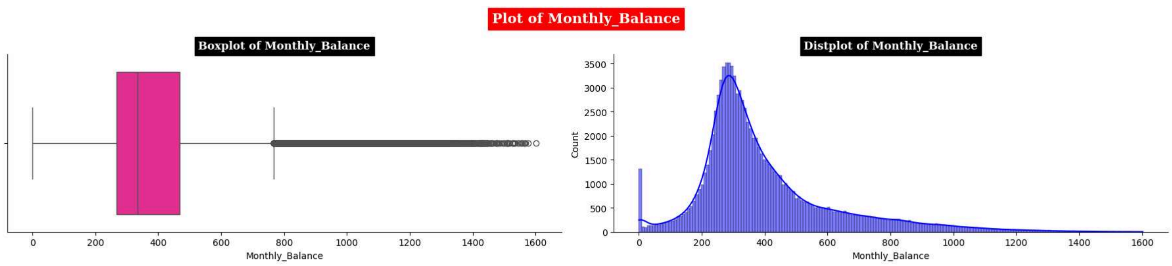

Plot of Outstanding_Debt
Boxplot of Outstanding_Debt
Distplot of Outstanding_Debt
Outstanding_Debt
Count

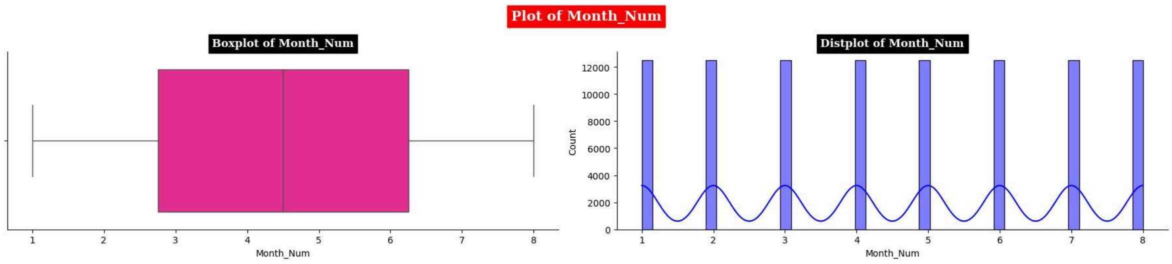

Plot of Credit_Utilization_Ratio
Boxplot of Credit_Utilization_Ratio
Distplot of Credit_Utilization_Ratio
Credit_Utilization_Ratio
Count


Plot of Credit_History_Age
Boxplot of Credit_History_Age
Distplot of Credit_History_Age
Credit_History_Age
Count


Plot of Total_EMI_per_month
Boxplot of Total_EMI_per_month
Distplot of Total_EMI_per_month
Total_EMI_per_month
Count

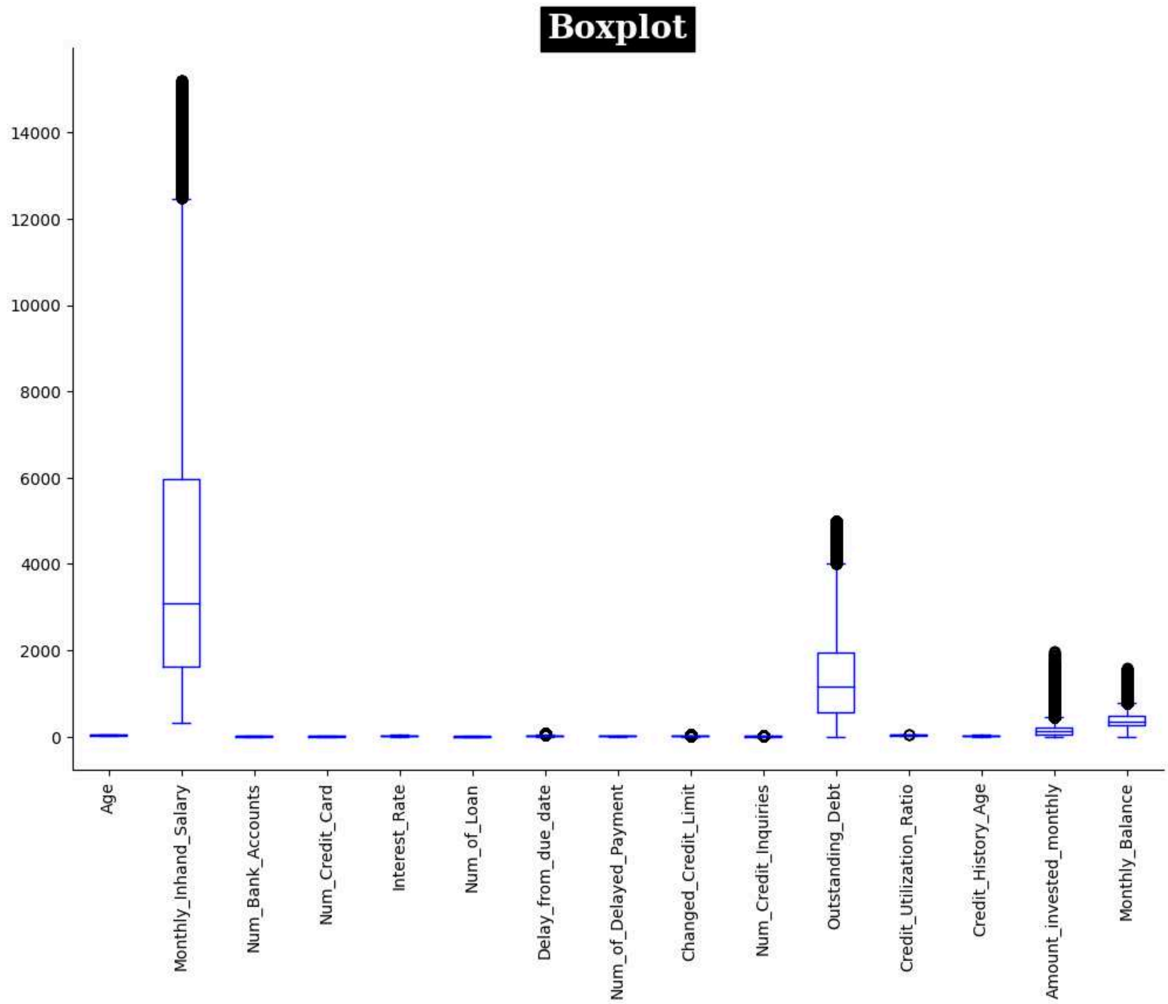

Plot of Amount_invested_monthly
Boxplot of Amount_invested_monthly
Distplot of Amount_invested_monthly
Amount_invested_monthly
Count


Plot of Monthly_Balance
Boxplot of Monthly_Balance
Distplot of Monthly_Balance
Monthly_Balance
Count



In [203…

```
columns_to_plot = ['Age', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts', 'Num_Cre
                   'Interest_Rate', 'Num_of_Loan', 'Delay_from_due_date', 'Num_
                   'Changed_Credit_Limit', 'Num_Credit_Inquiries', 'Outstanding
                   'Credit_Utilization_Ratio', 'Credit_History_Age',
                   'Amount_invested_monthly', 'Monthly_Balance']
numerical_cols = df.select_dtypes(include=['float64', 'int64'])
plt.figure(figsize=(12, 8))
numerical_cols[columns_to_plot].boxplot(rot=90, color='blue')
plt.title(f'Boxplot',fontsize=20,fontfamily='serif',fontweight='bold',
          backgroundcolor='Black',color='w')
sns.despine()
plt.grid(False)
plt.show()
```



In [204…

```
# Creating a dictionary for aggregation at Customer_ID level
agg_dict = {
    'ID': 'first',
     # Grouped by on Customer_ID so not included
    'Name': 'first',
    'Age': 'first',
    'SSN': 'first',
```

```
    'Occupation': 'first',
    'Annual_Income': 'first',
    'Monthly_Inhand_Salary': 'first',
    'Num_Bank_Accounts': 'first',
    'Num_Credit_Card': 'first',
    'Interest_Rate': 'first',
    'Num_of_Loan': 'first',
    'Type_of_Loan': 'first',
    'Delay_from_due_date': 'mean',
    'Num_of_Delayed_Payment': 'first',
    'Changed_Credit_Limit': 'mean',
    'Num_Credit_Inquiries': 'first',
    'Credit_Mix': 'first',
    'Outstanding_Debt': 'first',
    'Credit_Utilization_Ratio': 'mean',
    'Credit_History_Age': 'first',
    'Payment_of_Min_Amount': 'first',
    'Total_EMI_per_month': 'first',
    'Amount_invested_monthly': 'mean',
    # Payment_Behaviour not included
    # if needed introduce above and do mean
    #'Payment_History_Score':'mean',
    'Monthly_Balance': 'mean',

}
df_aggregated = df.groupby('Customer_ID').agg(agg_dict).reset_index()
df_aggregated
```

Out[204…

	Customer_ID	ID	Name	Age	SSN	Occupation	Annual_Income
0	CUS_0x1000	0x1628a	Alistair Barrf	17	913-74-1218	Lawyer	30625.940
1	CUS_0x1009	0x66a2	Arunah	26	063-67-6938	Mechanic	52312.680
2	CUS_0x100b	0x1ef6	Shirboni	18	238-62-0395	Media_Manager	113781.390
3	CUS_0x1011	0x17646	Schneyerh	44	793-05-8223	Doctor	58918.470
4	CUS_0x1013	0x243ea	Cameront	44	930-49-9615	Mechanic	98620.980
...	...	...	...	...	...	...	...
12495	CUS_0xff3	0x344a	Somervilled	55	726-35-5322	Scientist	17032.785
12496	CUS_0xff4	0x16aa	Poornimaf	37	655-05-7666	Entrepreneur	25546.260
12497	CUS_0xff6	0xfab6	Shieldsb	19	541-92-8371	Doctor	117639.920
12498	CUS_0xffc	0x61e6	Brads	17	226-86-7294	Musician	60877.170
12499	CUS_0xffd	0x25afa	Damouniq	29	832-88-8320	Scientist	41398.440

12500 rows × 25 columns

In [205…

```
categorical_cols = df_aggregated.select_dtypes(include=['object', 'category'])
```

In [206…

```
categorical_cols
```

Out[206…

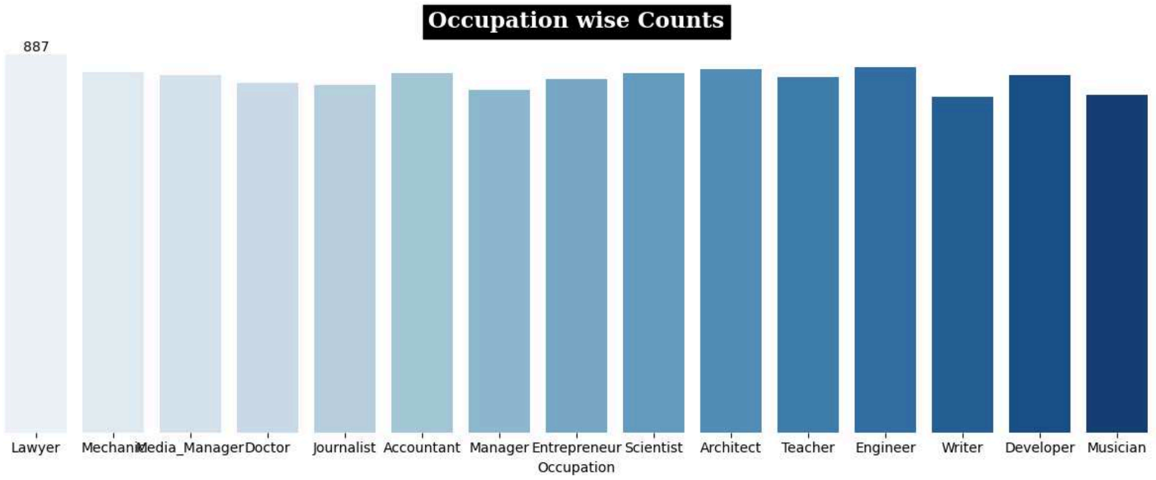
	Customer_ID	ID	Name	SSN	Occupation	Type_of_Loan	Credit_I
0	CUS_0x1000	0x1628a	Alistair Barrf	913-74-1218	Lawyer	Credit-Builder Loan, and Home Equity Loan	I
1	CUS_0x1009	0x66a2	Arunah	063-67-6938	Mechanic	Not Specified, Home Equity Loan, Credit-Builde...	Stand
2	CUS_0x100b	0x1ef6	Shirboni	238-62-0395	Media_Manager	No loan taken	Gc
3	CUS_0x1011	0x17646	Schneyerh	793-05-8223	Doctor	Student Loan, Credit-Builder Loan, and Debt Co...	Stand
4	CUS_0x1013	0x243ea	Cameront	930-49-9615	Mechanic	Student Loan, Debt Consolidation Loan, and Per...	Gc
...	...	...	...	...	...	...	
12495	CUS_0xff3	0x344a	Somervilled	726-35-5322	Scientist	Personal Loan, Mortgage Loan, and Auto Loan	Gc
12496	CUS_0xff4	0x16aa	Poornimaf	655-05-7666	Entrepreneur	Not Specified, Student Loan, Student Loan, Cre...	Stand
12497	CUS_0xff6	0xfab6	Shieldsb	541-92-8371	Doctor	Home Equity Loan, and Auto Loan	Gc
12498	CUS_0xffc	0x61e6	Brads	226-86-7294	Musician	Credit-Builder Loan, Payday Loan, Not Specifie...	I
12499	CUS_0xffd	0x25afa	Damouniq	832-88-8320	Scientist	Auto Loan, Payday Loan, Payday Loan, Mortgage ...	Stand

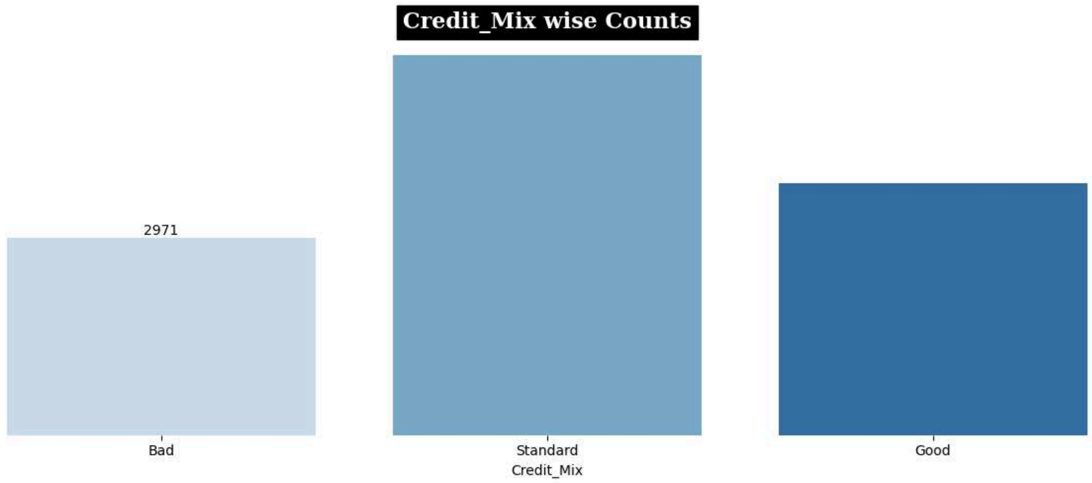
12500 rows × 8 columns

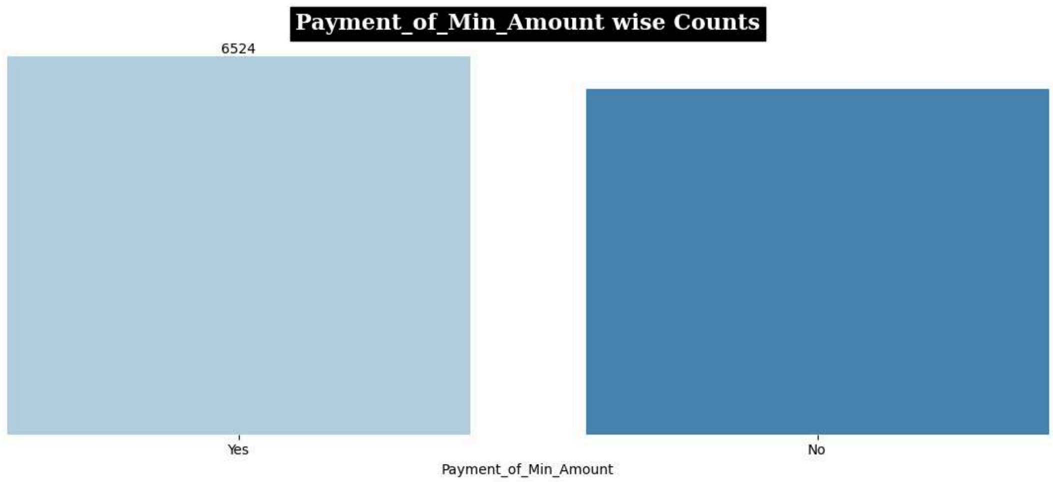
In [207…

```
selected_cols = ['Occupation', 'Credit_Mix', 'Payment_of_Min_Amount']

for col in selected_cols:
```

```
    plt.figure(figsize=(12, 5))
    plt.style.use('default')
    plt.style.use('seaborn-bright')
    a = sns.countplot(data=categorical_cols, x=col, palette='Blues')
    plt.title(f'{col} wise Counts', fontsize=16, fontfamily='serif', fontweight=
              backgroundcolor='Black', color='w')
    a.bar_label(a.containers[0], label_type='edge')
    sns.despine(left=True, bottom=True)
    plt.yticks([])
    plt.ylabel('')
    plt.tight_layout()
    plt.show()
```







Skewness

In [208…

```
# Skewness Coefficient
numerical_cols = df.select_dtypes(include=['float64', 'int64'])
print("Skewness Coefficient")
print("-" * 20)
df.skew(numeric_only = True)
#print(numerical_cols.skew().round(4))
```

```
Skewness Coefficient
--------------------
```

Out[208…

	0
Age	0.157009
Annual_Income	12.391367
Monthly_Inhand_Salary	1.128520
Num_Bank_Accounts	-0.189184
Num_Credit_Card	0.225830
Interest_Rate	0.496232
Num_of_Loan	0.445609
Delay_from_due_date	0.985874
Num_of_Delayed_Payment	-0.223545
Changed_Credit_Limit	0.641177
Num_Credit_Inquiries	0.416113
Outstanding_Debt	1.207536
Credit_Utilization_Ratio	0.028617
Credit_History_Age	-0.048998
Total_EMI_per_month	7.102524
Amount_invested_monthly	2.548864
Monthly_Balance	1.498597
Month_Num	0.000000

dtype: float64

Insights

Highly Skewed Variables:

- **Annual_Income** (12.3914), **Total_EMI_per_month** (7.1025), and **Amount_invested_monthly** (2.5489) are highly positively skewed. This suggests that a small number of customers have significantly higher values in these categories compared to the rest, indicating the presence of outliers or a long tail in the distribution.

Moderately Skewed Variables:

- **Outstanding_Debt** (1.2075), **Monthly_Balance** (1.4986), and **Monthly_Inhand_Salary** (1.1285) show moderate positive skewness, indicating that while the majority of values are lower, there are some customers with considerably higher values, leading to a rightward tail in the distribution.

Nearly Symmetrical and Slightly Skewed Variables:

- Variables like **Age** (0.1570), **Num_Bank_Accounts** (-0.1892), and **Credit_History_Age** (-0.0490) exhibit low skewness, meaning their distributions are more symmetrical with values evenly spread around the mean. This suggests a more uniform distribution without significant outliers or tails.

Correlation Heatmap

In [209…

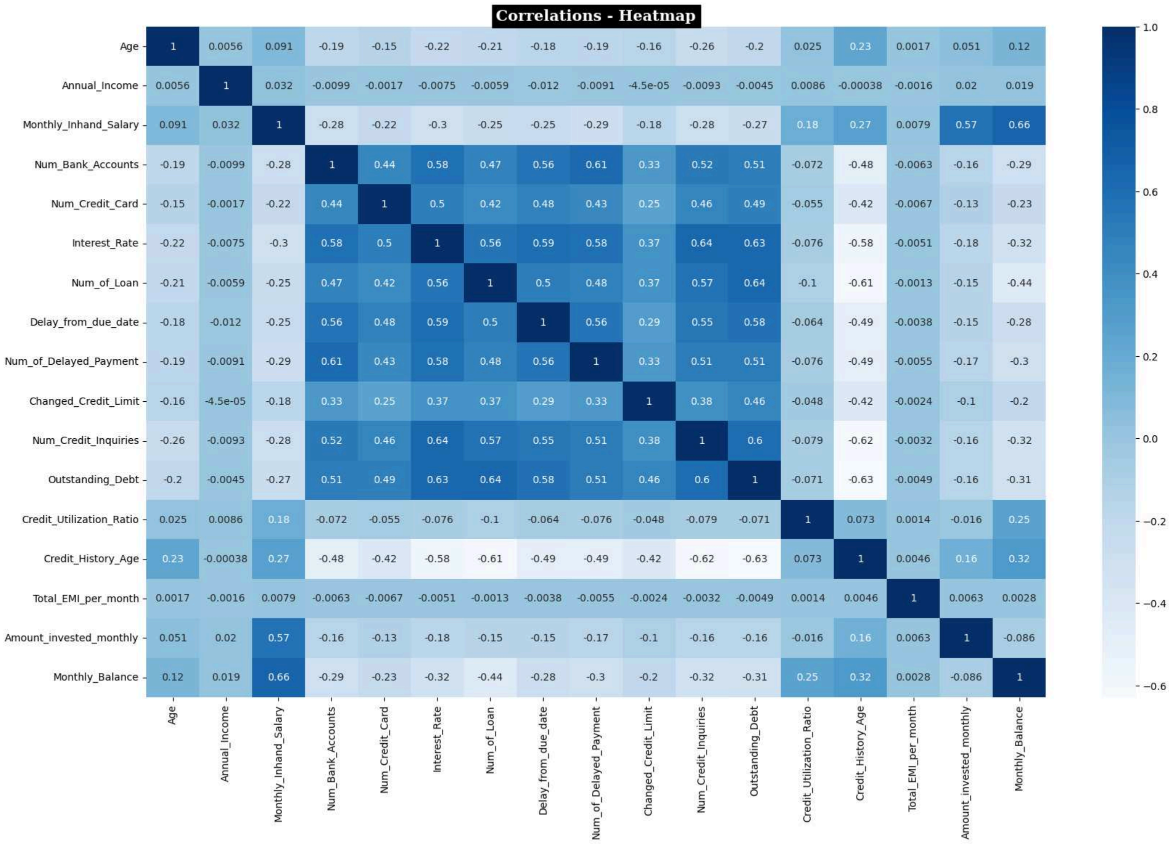
```
numerical_cols.corr()
```

Out[209…

	Age	Annual_Income	Monthly_Inhand_Salary	Num_Banl
Age	1.000000	0.005590	0.090794	
Annual_Income	0.005590	1.000000	0.031698	
Monthly_Inhand_Salary	0.090794	0.031698	1.000000	
Num_Bank_Accounts	-0.190926	-0.009884	-0.283318	
Num_Credit_Card	-0.148591	-0.001667	-0.216956	
Interest_Rate	-0.217856	-0.007455	-0.301906	
Num_of_Loan	-0.213598	-0.005940	-0.254406	
Delay_from_due_date	-0.175074	-0.011611	-0.251162	
Num_of_Delayed_Payment	-0.187016	-0.009144	-0.289875	
Changed_Credit_Limit	-0.156673	-0.000045	-0.175135	
Num_Credit_Inquiries	-0.256649	-0.009308	-0.280591	
Outstanding_Debt	-0.202294	-0.004533	-0.269078	
Credit_Utilization_Ratio	0.025482	0.008606	0.176081	
Credit_History_Age	0.234257	-0.000383	0.271058	
Total_EMI_per_month	0.001671	-0.001620	0.007949	
Amount_invested_monthly	0.051313	0.020078	0.568380	
Monthly_Balance	0.116397	0.018747	0.659723	

In [210…

```
plt.figure(figsize=(20,12))
sns.heatmap(numerical_cols.corr(), annot=True, cmap='Blues')
plt.title('Correlations - Heatmap',fontsize=15,fontfamily='serif',fontweight='bc
#sns.despine()
plt.show()
```



Insights

- **Monthly Inhand Salary** shows a strong positive correlation with **Monthly Balance** (0.6597) and **Amount Invested Monthly** (0.5684), indicating that higher in-hand salary is associated with greater savings and investments.

- **Interest Rate** is strongly correlated with **Num Credit Inquiries** (0.6385) and **Outstanding Debt** (0.6294), suggesting that customers with higher interest rates tend to have more credit inquiries and higher outstanding debt.

- **Num of Delayed Payments** has a strong positive correlation with **Num of Bank Accounts** (0.6120) and **Delay from Due Date** (0.5556), indicating that customers with more bank accounts tend to delay payments more frequently.

- **Credit History Age** has strong negative correlations with **Outstanding Debt** (-0.6285) and **Interest Rate** (-0.5755), implying that longer credit history is associated with lower debt and interest rates.

  > **Moderate Correlation in Loan-related Metrics**:

  > **Negative Impact on Credit History Age**:

  > **Num of Delayed Payments Impact**:

  > **Interest Rate and Credit Inquiries**:

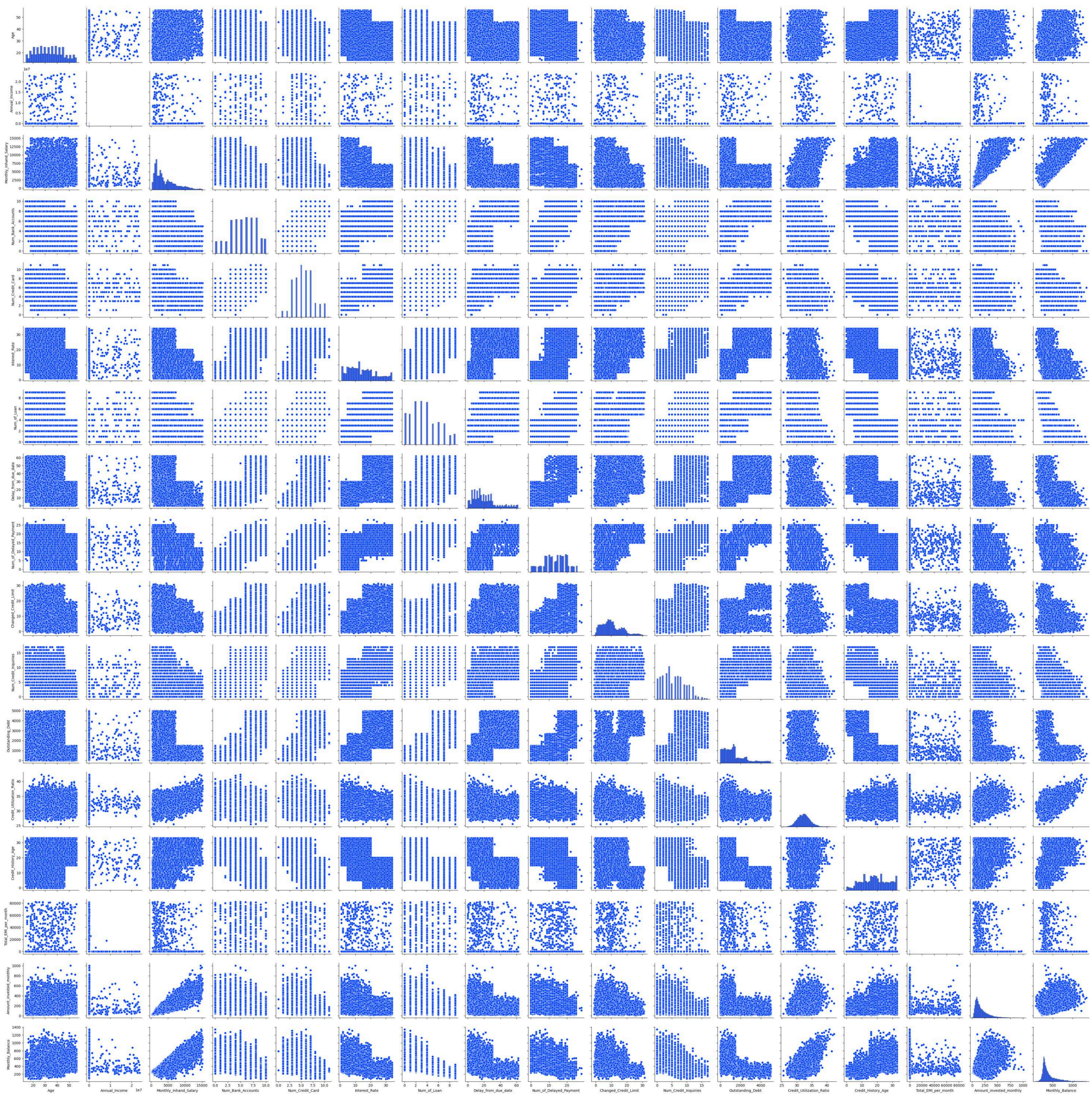
  > **Strong Correlation with Monthly Inhand Salary and Investing**:

- **Num of Loans** is moderately correlated with **Interest Rate** (0.5592) and **Num Credit Inquiries** (0.5696), suggesting that customers with more loans tend to have higher interest rates and credit inquiries.

In [211…

```
plt.figure(figsize=(18,0.5))
plt.axis('off')
plt.style.use('default')
plt.style.use('seaborn-bright')
plt.title(f' Pairplot Analysis ',fontfamily='serif',fontweight='bold',fontsize=1
sns.pairplot(data=df_aggregated, palette='Oranges')
plt.tight_layout()
plt.show()
```

Pairplot Analysis



Feature Engineering

In [212…

```
dff = df.copy()
```

In [213…

```
dff.head(8)
```

Out[213…

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	M
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
5	0x1607	CUS_0xd40	June	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
6	0x1608	CUS_0xd40	July	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
7	0x1609	CUS_0xd40	August	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	

Feature creation

In [214…

```
dff['Debt_to_Income_Ratio'] = dff['Outstanding_Debt'] / dff['Annual_Income']
dff['Total_Debt'] = dff['Outstanding_Debt'] + dff['Total_EMI_per_month']
```

In [215…

```
payment_behaviour_mapping = {
    'High_spent_Small_value_payments': 4,
```

```
    'High_spent_Medium_value_payments': 5,
    'High_spent_Large_value_payments': 6,
    'Low_spent_Small_value_payments': 1,
    'Low_spent_Medium_value_payments': 2,
    'Low_spent_Large_value_payments': 3
}

# Apply mapping
dff['Payment_Behaviour_Num'] = dff['Payment_Behaviour'].map(payment_behaviour_ma
```

Insights

- High_spent_Small_value_payments: Likely to indicate high spending habits with small payments, which might be risky if the behavior is consistent.
- High_spent_Medium_value_payments: Indicates high spending with medium payments, potentially a higher risk.
- High_spent_Large_value_payments: Shows high spending with large payments, which might indicate high financial risk.
- Low_spent_Small_value_payments: Shows low spending with small payments, likely less risky.
- Low_spent_Medium_value_payments: Low spending with medium payments, potentially moderate risk.
- Low_spent_Large_value_payments: Low spending with large payments, might be less risky but could indicate underutilization of credit.

In [216…
```
dff['Credit_Mix'].unique()
```

Out[216…
```
array(['Good', 'Standard', 'Bad'], dtype=object)
```

In [217…
```
dff['Credit_Mix_num'] = dff['Credit_Mix'].map({'Good':2 , 'Standard':1, 'Bad':0}
```

In [218…
```
dff['Payment_of_Min_Amount'].unique()
```

Out[218…
```
array(['No', 'Yes'], dtype=object)
```

In [219…
```
dff['Payment_of_Min_Amount'] = dff['Payment_of_Min_Amount'].map({'Yes':1, 'No':0
```

In [220…
```
dff['Payment_of_Min_Amount'].isna().sum() , dff['Payment_of_Min_Amount'].unique(
```

Out[220…
```
(0, array([0, 1]))
```

In [221…
```
dff['30%_of_Monthly_Salary'] = dff['Monthly_Inhand_Salary']*0.3

dff['ability to pay loan with saving'] = np.where(dff['30%_of_Monthly_Salary']>d
```

In [222…
```
dff = dff.drop(columns=['ID','Type_of_Loan','Credit_Mix','Payment_Behaviour'],ax
```

In [223…
```
dff["Payment_History_Score"] = ( -1 * dff["Delay_from_due_date"]
                                -1 * dff["Num_of_Delayed_Payment"]
                                +1 * dff["Payment_of_Min_Amount"])
```

In [224…
```
dff.tail(8)
```

Out[224…

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly
99992	CUS_0x942c	January	Nicks	25	078-73-5990	Mechanic	39628.99	
99993	CUS_0x942c	February	Nicks	25	078-73-5990	Mechanic	39628.99	
99994	CUS_0x942c	March	Nicks	25	078-73-5990	Mechanic	39628.99	
99995	CUS_0x942c	April	Nicks	25	078-73-5990	Mechanic	39628.99	
99996	CUS_0x942c	May	Nicks	25	078-73-5990	Mechanic	39628.99	
99997	CUS_0x942c	June	Nicks	25	078-73-5990	Mechanic	39628.99	
99998	CUS_0x942c	July	Nicks	25	078-73-5990	Mechanic	39628.99	
99999	CUS_0x942c	August	Nicks	25	078-73-5990	Mechanic	39628.99	

Hypothetical Credit_Score Computation

Scaling

In [225…

```
scaling_features = dff.iloc[:, [3] + list(range(6, dff.shape[1]))].columns
scaling_features
```

Out[225…

```
Index(['Age', 'Annual_Income', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts',
       'Num_Credit_Card', 'Interest_Rate', 'Num_of_Loan',
       'Delay_from_due_date', 'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Outstanding_Debt', 'Credit_Utilization_Ratio',
       'Credit_History_Age', 'Payment_of_Min_Amount', 'Total_EMI_per_month',
       'Amount_invested_monthly', 'Monthly_Balance', 'Month_Num',
       'Debt_to_Income_Ratio', 'Total_Debt', 'Payment_Behaviour_Num',
       'Credit_Mix_num', '30%_of_Monthly_Salary',
       'ability to pay loan with saving', 'Payment_History_Score'],
      dtype='object')
```

In [226…

```
scaler = MinMaxScaler()
dff[scaling_features] = scaler.fit_transform(dff[scaling_features])
```

In [227…

```
dff
```

Out[227…

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
...	...	...	...	...	...	...	...
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349

100000 rows × 31 columns

In [228…

```
dff.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 31 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   Customer_ID                    100000 non-null  object 
 1   Month                          100000 non-null  object 
 2   Name                           100000 non-null  object 
 3   Age                            100000 non-null  float64
 4   SSN                            100000 non-null  object 
 5   Occupation                     100000 non-null  object 
 6   Annual_Income                  100000 non-null  float64
 7   Monthly_Inhand_Salary          100000 non-null  float64
 8   Num_Bank_Accounts              100000 non-null  float64
 9   Num_Credit_Card                100000 non-null  float64
 10  Interest_Rate                  100000 non-null  float64
 11  Num_of_Loan                    100000 non-null  float64
 12  Delay_from_due_date            100000 non-null  float64
 13  Num_of_Delayed_Payment         100000 non-null  float64
 14  Changed_Credit_Limit           100000 non-null  float64
 15  Num_Credit_Inquiries           100000 non-null  float64
 16  Outstanding_Debt               100000 non-null  float64
 17  Credit_Utilization_Ratio       100000 non-null  float64
 18  Credit_History_Age             100000 non-null  float64
 19  Payment_of_Min_Amount          100000 non-null  float64
 20  Total_EMI_per_month            100000 non-null  float64
 21  Amount_invested_monthly        100000 non-null  float64
 22  Monthly_Balance                100000 non-null  float64
 23  Month_Num                      100000 non-null  float64
 24  Debt_to_Income_Ratio           100000 non-null  float64
 25  Total_Debt                     100000 non-null  float64
 26  Payment_Behaviour_Num          100000 non-null  float64
 27  Credit_Mix_num                 100000 non-null  float64
 28  30%_of_Monthly_Salary          100000 non-null  float64
 29  ability to pay loan with saving 100000 non-null  float64
 30  Payment_History_Score          100000 non-null  float64
dtypes: float64(26), object(5)
memory usage: 23.7+ MB
```

In [229…

```
# Define new feature weights including RFM
weights = {
    'Payment_Behaviour_Num': 0.10,
    'Credit_Utilization_Ratio': 0.10,
    'Outstanding_Debt': 0.05,
    'ability to pay loan with saving':0.10,
    'Total_EMI_per_month': 0.10,
    'Num_Credit_Inquiries': 0.05,
    'Credit_History_Age': 0.10,
    'Monthly_Balance': 0.10,
    'Annual_Income': 0.05,
    'Num_Bank_Accounts': 0.05,
    'Credit_Mix_num': 0.05,
    'Payment_History_Score': 0.15
}
```

Reasons for Feature selection and its weightage - Credit Score Computation

- Represents the total debt owed. Significant outstanding debt can signal financial strain, impacting the ability to manage new credit.
- Indicates the total monthly payments towards loans. Helps assess the customer's existing debt burden and financial obligations.
- Shows how frequently the customer has applied for credit. Frequent inquiries can suggest financial distress or a high demand for credit.
- Reflects the length of time the customer has maintained credit accounts. A longer credit history generally suggests more reliable credit behavior.
- Tracks the customer's balance on a monthly basis. A consistently positive balance indicates stronger financial health and stability.
- Provides insight into the customer's financial capacity. Higher income typically suggests a greater ability to manage and repay debt.
- Reflects the number of bank accounts held. Multiple accounts can indicate effective financial management, although an excess might signal potential financial issues.
- Indicates the diversity of credit types held. A varied credit mix can positively influence creditworthiness, demonstrating the ability to manage different credit types.
- Captures payment habits, essential for assessing creditworthiness. Reflects factors like payment frequency and amounts, which are crucial for credit evaluation.
- Measures the proportion of credit used relative to total credit available. A high ratio may indicate potential financial risk due to extensive credit usage.

> **Payment_history_score:**

> **Credit_Utilization_Ratio:**

> **Payment_Behaviour_Num:**

> **Credit_Mix_Num:**

> **Num_Bank_Accounts:**

> **Annual_Income:**

> **Monthly_Balance:**

> **Credit_History_Age:**

> **Num_Credit_Inquiries:**

> **Total_EMI_per_month:**

> **Outstanding_Debt:**

- Payments history plays a major role in determining the credit approval and hence heavy weightage.

In [230...
```
dff['Credit_Score'] = dff.apply(lambda x: sum(x[feature] * weight for feature, w
```

In [231...
```
dff['Credit_Score'] = dff['Credit_Score'] * (850 - 300) + 300
```

In [232...
```
dff[(dff['Credit_Score']<300) | (dff['Credit_Score']>850)]
```

Out[232...

Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand

In [233...
```
cs_df = dff[['Customer_ID','Name','SSN','Month','Credit_Score']]
```

Aggregated data - Consolidated

In [234...
```
cdf = cs_df.groupby(['Customer_ID','Name','SSN'])['Credit_Score'].mean().to_fram
cdf.sample(10)
```

Out[234...

	Customer_ID	Name	SSN	Credit_Score
11888	CUS_0xc131	Laub	910-05-8770	466
1461	CUS_0x2798	"John ODonnell"q	254-30-0873	563
1989	CUS_0x2f49	Rochaf	524-50-8228	498
11337	CUS_0xb8f6	Melp	367-94-5147	595
3954	CUS_0x4c83	Hutchisonp	326-69-1811	566
7366	CUS_0x7e63	Kevin Krolickig	926-75-5141	586
898	CUS_0x1eee	Gardnerp	513-08-1791	593
5168	CUS_0x5e34	Atossav	028-39-5974	499
5708	CUS_0x65d2	Nia Williamsf	659-77-0885	550
5174	CUS_0x5e40	Allisonz	355-34-7507	498

In [235...
```
bins = [300, 500, 600, 750, 800, 850]
bin_labels = ['very Bad','Poor', 'Fair', 'Good', 'Excellent']

# Apply binning
dff['Monthly_Credit_Score_category'] = pd.cut(dff['Credit_Score'], bins=bins, la
```

In [236...
```
dff.head(8)
```

Out[236…

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Mont
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
5	CUS_0xd40	June	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
6	CUS_0xd40	July	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
7	CUS_0xd40	August	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	

In [237…

```python
bins = [300, 500, 600, 750, 800, 850]
bin_labels = ['very Bad','Poor', 'Fair', 'Good', 'Excellent']

# Apply binning
cdf['overall_Credit_Score_category'] = pd.cut(dff['Credit_Score'], bins=bins, la
```

In [238…

```python
cdf.sample(10)
```

Out[238…

	Customer_ID	Name	SSN	Credit_Score	overall_Credit_Score_category
7595	CUS_0x81e9	Steve Slaterm	541-82-8577	508	very Bad
12289	CUS_0xcc2	Bakerf	632-81-0014	533	Poor
10209	CUS_0xa899	Kohj	077-54-4259	532	Poor
4943	CUS_0x5afd	Jedw	163-05-8880	591	Fair
10005	CUS_0xa567	Richwinef	952-02-0446	538	very Bad
10225	CUS_0xa8d1	Nicola Leskem	596-68-6198	588	Poor
9477	CUS_0x9dc6	David Lawderk	850-48-2308	512	Poor
3261	CUS_0x42dc	Yinkaz	517-51-2372	552	Poor
2802	CUS_0x3c3a	Alexh	014-29-7065	479	Poor
4363	CUS_0x523c	Schnurrr	739-81-2314	527	Poor

In [239…

```
cdf.groupby('overall_Credit_Score_category')['Customer_ID'].nunique().to_frame()
```

Out[239…

	Customer_ID
overall_Credit_Score_category	
very Bad	2147
Poor	9192
Fair	1161
Good	0
Excellent	0

In [240…

```
cdf.overall_Credit_Score_category.value_counts()
```

Out[240…

overall_Credit_Score_category	count
Poor	9192
very Bad	2147
Fair	1161
Good	0
Excellent	0

dtype: int64

In [241…

```
cdf.shape
```

Out[241…

```
(12500, 5)
```

In [242…

```
cdf
```

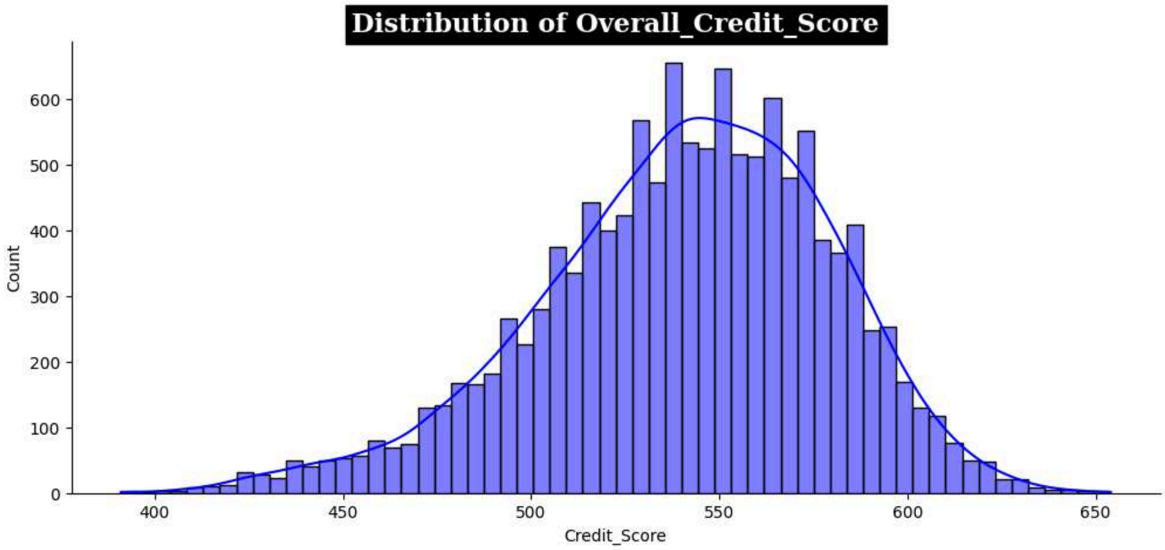
Out[242...

	Customer_ID	Name	SSN	Credit_Score	overall_Credit_Score_category
0	CUS_0x1000	Alistair Barrf	913-74-1218	482	Poor
1	CUS_0x1009	Arunah	063-67-6938	571	Poor
2	CUS_0x100b	Shirboni	238-62-0395	565	Poor
3	CUS_0x1011	Schneyerh	793-05-8223	507	Poor
4	CUS_0x1013	Cameront	930-49-9615	573	Poor
...	...	...	...	...	...
12495	CUS_0xff3	Somervilled	726-35-5322	541	Poor
12496	CUS_0xff4	Poornimaf	655-05-7666	543	Poor
12497	CUS_0xff6	Shieldsb	541-92-8371	590	Poor
12498	CUS_0xffc	Brads	226-86-7294	518	Poor
12499	CUS_0xffd	Damouniq	832-88-8320	548	Poor

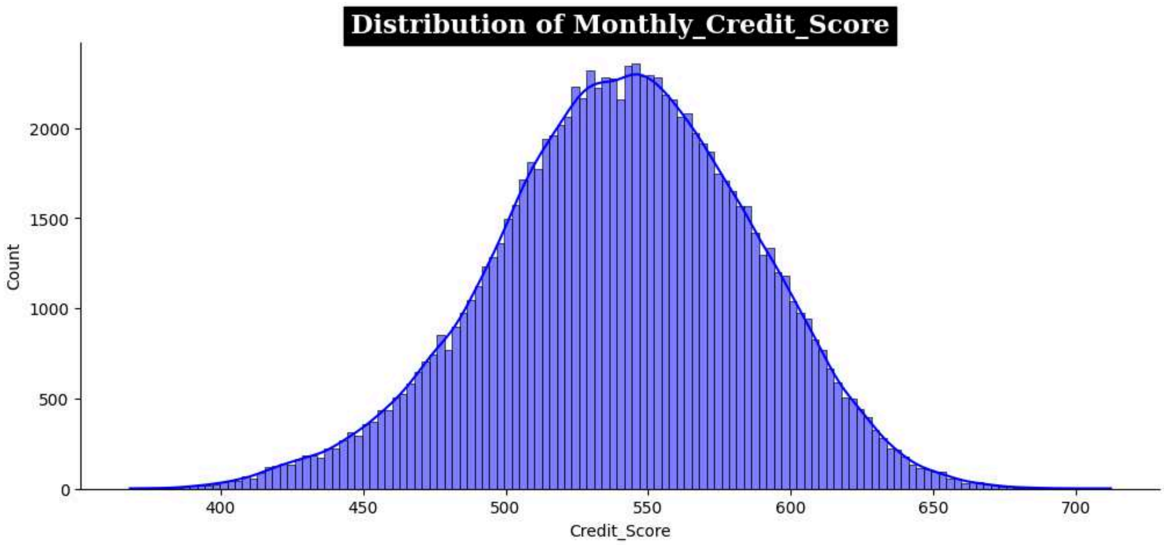
12500 rows × 5 columns

In [243...

```
plt.figure(figsize=(12,5))
plt.title('Distribution of Overall_Credit_Score',fontsize=16,fontfamily='serif',
sns.histplot(cdf['Credit_Score'],color='Blue',kde=True)
sns.despine()
plt.show()
```



In [244…
```
plt.figure(figsize=(12,5))
plt.title('Distribution of Monthly_Credit_Score',fontsize=16,fontfamily='serif',
sns.histplot(dff['Credit_Score'],color='Blue',kde=True)
sns.despine()
plt.show()
```



In [245…
```
#cdf.to_csv('credit_scored_data.csv',index=False)
```

In [246…
```
Fair_customers = cdf[cdf['overall_Credit_Score_category'] == 'Fair']
Fair_customers.describe().T
```

Out[246…

	count	mean	std	min	25%	50%	75%	max
Credit_Score	1161.0	539.585702	39.489149	411.0	516.0	543.0	568.0	645.0

RFM Integration

In [247…

```
dff['Recency'] = dff.groupby('Customer_ID')['Month_Num'].transform(lambda x: (x.
dff['Frequency'] = dff.groupby('Customer_ID')['Num_of_Loan'].transform('max')
dff['Monetary'] = dff.groupby('Customer_ID')['Monthly_Balance'].transform('sum')

rfm_features = ['Recency', 'Frequency', 'Monetary']
dff[rfm_features] = scaler.fit_transform(dff[rfm_features])
```

In [248…

```
dff
```

Out[248…

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501
...	...	...	...	...	...	...	...
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349

100000 rows × 36 columns

In [249…

```
dff['Credit_Score_after_RFM'] = dff['Credit_Score'] + (dff['Recency'] * 0.1 + df
```

In [250…

```
dff.tail(8)
```

Out[250…

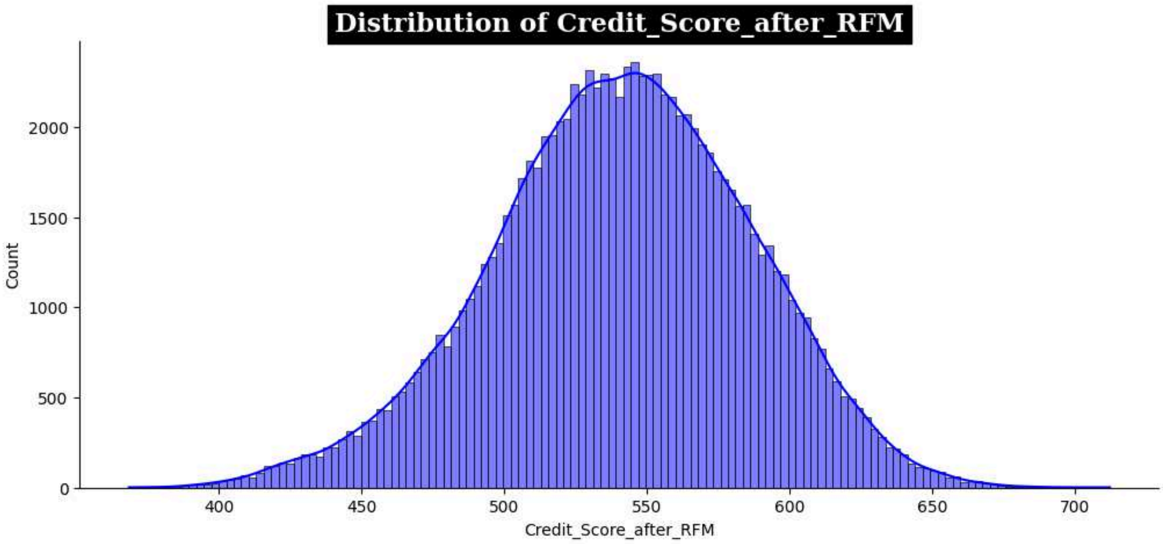
	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Mo
99992	CUS_0x942c	January	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99993	CUS_0x942c	February	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99994	CUS_0x942c	March	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	

In [251…

```
bins = [300, 500, 600, 750, 800, 850]
bin_labels = ['very Bad','Poor', 'Fair', 'Good', 'Excellent']

# Apply binning
dff['RFM_Credit_Score_category'] = pd.cut(dff['Credit_Score_after_RFM'], bins=bi
```

In [252…

```
plt.figure(figsize=(12,5))
plt.title('Distribution of Credit_Score_after_RFM',fontsize=16,fontfamily='serif
sns.histplot(dff['Credit_Score_after_RFM'],color='blue',kde=True)
sns.despine()
plt.show()
```



In [253…

```
dff.head(8)
```

Out[253…

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Mont
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
5	CUS_0xd40	June	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
6	CUS_0xd40	July	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	
7	CUS_0xd40	August	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	

In [254…

```
dff.groupby('RFM_Credit_Score_category')['Customer_ID'].nunique().to_frame()
```

Out[254…

	Customer_ID
RFM_Credit_Score_category	
very Bad	4920
Poor	11917
Fair	3571
Good	0
Excellent	0

In [255…

```
cdff = dff.groupby('Customer_ID')['Credit_Score_after_RFM'].mean().to_frame().re
cdff
```

Out[255…

	Customer_ID	Credit_Score_after_RFM
0	CUS_0x1000	482.128252
1	CUS_0x1009	571.445711
2	CUS_0x100b	564.627757
3	CUS_0x1011	507.128548
4	CUS_0x1013	572.948491
...	...	...
12495	CUS_0xff3	541.035743
12496	CUS_0xff4	543.165731
12497	CUS_0xff6	589.904199
12498	CUS_0xffc	518.068314
12499	CUS_0xffd	548.525868

12500 rows × 2 columns

In [256…

```
bins = [300, 500, 650, 750, 800, 850]
bin_labels = ['very Bad','Poor', 'Fair', 'Good', 'Excellent']

# Apply binning
cdff['cumulative_RFM_Credit_Score_category'] = pd.cut(cdff['Credit_Score_after_R
```

In [257…

```
cdff.sample(6)
```

Out[257…

	Customer_ID	Credit_Score_after_RFM	cumulative_RFM_Credit_Score_category
11600	CUS_0xbccf	531.256669	Poor
3699	CUS_0x48c5	444.209338	very Bad
873	CUS_0x1e9b	474.220069	very Bad
8865	CUS_0x949f	525.751388	Poor
3957	CUS_0x4c89	537.207505	Poor
5964	CUS_0x6963	519.388979	Poor

In [258…

```
cdff.groupby('cumulative_RFM_Credit_Score_category')['Customer_ID'].nunique().tc
```

Out[258…

	Customer_ID
cumulative_RFM_Credit_Score_category	
very Bad	1820
Poor	10679
Fair	1
Good	0
Excellent	0

Insights

- After applying RFM analysis, there are noticeable differences in creditworthiness across different customer segments. The categorization of customers has shifted, indicating that the influence of RFM analysis has led to changes in the classification of creditworthiness, even with the bins being set consistently across the board.

  > This insight highlights the dynamic nature of creditworthiness assessment when influenced by RFM, emphasizing how customer segmentation can change based on different metrics.

3 Months - Transaction Period Analysis

> Analyze how credit scores and aggregated features change over the last 3 to 6 months to understand the temporal dynamics of creditworthiness. Lets consider `Last 3 months`.

In [259…

```
dff.tail(8)
```

Out[259…

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Mo
99992	CUS_0x942c	January	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99993	CUS_0x942c	February	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99994	CUS_0x942c	March	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	

In [260…

```
months = ['June', 'July', 'August']

filtered_dff = dff[dff['Month'].isin(months)]
```

In [261…

```
filtered_dff.shape
```

Out[261…

```
(37500, 38)
```

In [262…

```
filtered_dff.tail(6)
```

Out[262…

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	M
99989	CUS_0x8600	June	Sarah McBridec	0.333333	031-35-0942	Architect	0.000537	
99990	CUS_0x8600	July	Sarah McBridec	0.333333	031-35-0942	Architect	0.000537	
99991	CUS_0x8600	August	Sarah McBridec	0.333333	031-35-0942	Architect	0.000537	
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	

In [263…

```
fdf = filtered_dff[['Customer_ID','Name','Credit_Score','Monthly_Credit_Score_ca
fdf
```

Out[263...

	Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Scc
5	CUS_0xd40	Aaron Maashoh	583.254962	Poor	
6	CUS_0xd40	Aaron Maashoh	527.300235	Poor	
7	CUS_0xd40	Aaron Maashoh	577.758551	Poor	
13	CUS_0x21b1	Rick Rothackerj	611.531934	Fair	
14	CUS_0x21b1	Rick Rothackerj	595.837460	Poor	
...	...	...	...	...	
99990	CUS_0x8600	Sarah McBridec	494.460602	very Bad	
99991	CUS_0x8600	Sarah McBridec	553.221027	Poor	
99997	CUS_0x942c	Nicks	619.554305	Fair	
99998	CUS_0x942c	Nicks	565.803866	Poor	
99999	CUS_0x942c	Nicks	569.377974	Poor	

37500 rows × 6 columns

In [264...

```
fdf['diff'] = fdf['Credit_Score_after_RFM'] - fdf['Credit_Score']
```

In [265...

```
fdf
```

Out[265…

	Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Sco
5	CUS_0xd40	Aaron Maashoh	583.254962	Poor	
6	CUS_0xd40	Aaron Maashoh	527.300235	Poor	
7	CUS_0xd40	Aaron Maashoh	577.758551	Poor	
13	CUS_0x21b1	Rick Rothackerj	611.531934	Fair	
14	CUS_0x21b1	Rick Rothackerj	595.837460	Poor	
...	...	...	...	...	
99990	CUS_0x8600	Sarah McBridec	494.460602	very Bad	
99991	CUS_0x8600	Sarah McBridec	553.221027	Poor	
99997	CUS_0x942c	Nicks	619.554305	Fair	
99998	CUS_0x942c	Nicks	565.803866	Poor	
99999	CUS_0x942c	Nicks	569.377974	Poor	

37500 rows × 7 columns

In [266…

```
fdf[(fdf['diff']>1) | (fdf['diff']<0)]
```

Out[266…

Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Score_after_F

Observation:

- There is a very minute difference when it comes to monthwise credit score with RFM.

In [267…

```
fdf['Last_3_months_Credit_Score_consolidated'] = fdf.groupby(['Customer_ID', 'Na
```

In [268…

```
fdf
```

Out[268…

	Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Sc
5	CUS_0xd40	Aaron Maashoh	583.254962	Poor	
6	CUS_0xd40	Aaron Maashoh	527.300235	Poor	
7	CUS_0xd40	Aaron Maashoh	577.758551	Poor	
13	CUS_0x21b1	Rick Rothackerj	611.531934	Fair	
14	CUS_0x21b1	Rick Rothackerj	595.837460	Poor	
...	...	...	...	...	
99990	CUS_0x8600	Sarah McBridec	494.460602	very Bad	
99991	CUS_0x8600	Sarah McBridec	553.221027	Poor	
99997	CUS_0x942c	Nicks	619.554305	Fair	
99998	CUS_0x942c	Nicks	565.803866	Poor	
99999	CUS_0x942c	Nicks	569.377974	Poor	

37500 rows × 8 columns

In [269…

```
bins = [300, 500, 650, 750, 800, 850]
bin_labels = ['very Bad','Poor', 'Fair', 'Good', 'Excellent']
fdf['last_3_months_Credit_Score_category'] = pd.cut(fdf['Last_3_months_Credit_Sc
```

In [270…

```
l3m = fdf[['Customer_ID','Name','Last_3_months_Credit_Score_consolidated','last_
```

In [271…

```
l3m_unique = l3m.drop_duplicates(subset=['Customer_ID', 'Name'], keep='first')
```

In [272…

```
l3m_unique.sample(10)
```

Out[272...

	Customer_ID	Name	Last_3_months_Credit_Score_consolidated	last_3_mo
86925	CUS_0x3ace	Shirbonf	516.765222	
12797	CUS_0x4716	, Asiac	499.892734	
77237	CUS_0x84b8	Georgiopoulosb	556.060065	
2421	CUS_0x9983	Julien Toyerb	556.372378	
30525	CUS_0x46c3	Kaiserc	539.516670	
46789	CUS_0x5f1b	McCrankn	508.457545	
79597	CUS_0x7594	Langeo	522.405475	
33117	CUS_0x44e	Baertleiny	550.549917	
29805	CUS_0x4a6	Oliviaj	531.750484	
44725	CUS_0x26d0	Peter Dinklohz	566.617329	

In [273...

```
l3m_unique.groupby('last_3_months_Credit_Score_category')['Customer_ID'].nunique
```

Out[273...

	Customer_ID
last_3_months_Credit_Score_category	
very Bad	1985
Poor	10501
Fair	14
Good	0
Excellent	0

In [274...

```
import plotly.graph_objects as go
from PIL import Image

credit_score_input = int(input('Enter your Credit Score - '))

# Define the gauge plot
fig = go.Figure(go.Indicator(
    mode="gauge+number+delta",
    value=credit_score_input,
    title={'text': "Credit Score"},
    delta={'reference': 850, 'increasing': {'color': "green"}},
    gauge={
        'axis': {'range': [300, 850], 'tickwidth': 1, 'tickcolor': "darkblue"},
        'bar': {'color': "darkblue"},
        'bgcolor': "white",
        'steps': [
            {'range': [300, 550], 'color': 'red'},
            {'range': [550, 650], 'color': 'orange'},
            {'range': [650, 750], 'color': 'yellow'},
            {'range': [750, 850], 'color': 'green'}],
        'threshold': {
```

```
            'line': {'color': "black", 'width': 4},
            'thickness': 0.75,
            'value': credit_score_input}}))

fig.update_layout(
    title="Based on your Credit Score",
    font={'color': "darkblue", 'family': "Arial"}
)

plt.savefig('new_image.png')
```

```
Enter your Credit Score - 786
<Figure size 640x480 with 0 Axes>
```

In [275…

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

plt.figure(figsize=(1,0.5))
# Path to your image file
img_path = 'new_image.png'
# Load and display the image
img = mpimg.imread(img_path)
plt.imshow(img) # Display the image data
plt.axis('off')  # Hide axis
plt.tight_layout()
fig.show()
```

Insights on Analysis

> Updation of the numbers from above

Overall Credit Score Categories:

- **Very Bad**: 7412
- **Poor**: 5076
- **Fair**: 12
- **Good**: 0
- **Excellent**: 0

Observation: Most customers fall into the "Very Bad" or "Poor" categories, with very few in the "Fair" category and none in the "Good" or "Excellent" categories.

Cumulative RFM Credit Score Categories:

- **Very Bad**: 7788
- **Poor**: 4712
- **Fair**: 0
- **Good**: 0
- **Excellent**: 0

Observation: After integrating RFM factors, the number of customers in the "Very Bad" category has increased, while those in the "Poor" category have decreased. No customers fall into the "Fair," "Good," or "Excellent" categories.

Last 3 Months Credit Score Categories:

- **Very Bad**: 7761
- **Poor**: 4739
- **Fair**: 0
- **Good**: 0
- **Excellent**: 0

Observation: Similar to the cumulative RFM scores, the "Very Bad" and "Poor" categories dominate, with no customers in the higher categories.

Possible Reasons

High Proportion in "Very Bad" and "Poor" Categories:

- The dominance of the "Very Bad" and "Poor" categories across all models suggests a high level of credit risk among customers.
- **Possible Factors**: High levels of outstanding debt, poor payment behavior, high credit utilization, or recent negative changes in financial behavior could contribute to these scores.

No Customers in Higher Categories Post-RFM Integration:

- The complete absence of "Fair," "Good," and "Excellent" categories after integrating RFM features indicates that RFM factors might be emphasizing risks or penalizing certain credit behaviors heavily.
- **Possible Factors**: The integration of RFM features may have introduced stricter criteria or revealed high-risk patterns not captured previously.

Stable Distribution in Last 3 Months:

- The consistency in the distribution for the last 3 months mirrors the cumulative RFM scores, suggesting that recent credit behavior aligns with the longer-term integrated RFM analysis.
- **Possible Factors**: Recent credit behaviors might be reflective of ongoing financial issues or consistent poor credit management.

Summary

- Overall, both the cumulative and recent transaction-based analyses indicate a high proportion of customers in the "Very Bad" and "Poor" categories, suggesting significant credit risk.
- The integration of RFM features has intensified this trend, possibly due to the introduction of more granular risk factors.
- This highlights the need for targeted financial interventions or improved credit management practices among customers.

Recommendations & Suggestions

Refine Credit Scoring Models:

- **Action**: Integrate RFM and advanced metrics into credit scoring models to enhance accuracy and identify high-risk customers.
- **Benefit**: Improved prediction of creditworthiness and risk management.

Enhance Customer Engagement:

- **Action**: Implement proactive communication strategies, such as alerts and reminders, to encourage timely payments.
- **Benefit**: Reduced missed payments and improved credit behavior.

Develop Financial Education Programs:

- **Action**: Offer targeted financial literacy resources to help customers manage debt and improve credit scores.
- **Benefit**: Better customer financial health and responsible credit usage.

Adjust Credit Limits Strategically:

- **Action**: Regularly review and adjust credit limits based on current credit utilization and payment behavior.

- **Benefit**: Better alignment of credit limits with customers' repayment ability and risk mitigation.

Leverage Predictive Analytics:

- **Action**: Use predictive analytics to identify potential credit risks early and take preemptive actions.
- **Benefit**: Timely risk management and reduced likelihood of defaults.

Incentivize Positive Credit Behaviors:

- **Action**: Reward customers for improved credit management and responsible behavior.
- **Benefit**: Encourages better credit practices and enhances overall credit profiles.