💳 Case Study 💳
💴 Credit score Computation 💴
🔎 EDA 🔍

Analysed by : KASI

In [264…

🧐 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
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

Column Name	Description
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

> References

- https://thedocs.worldbank.org/en/doc/935891585869698451-0130022020/original/CREDITSCORINGAPPROACHESGUIDELINESFINALWEB.pdf
- https://www.iifl.com/blogs/cibil-score/different-types-of-credit-score
- https://www.etmoney.com/learn/loans/difference-between-cibil-score-and-credit-score/

> 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]:
```
data = pd.read_csv('Credit_score.csv')
data.head(8)
```

Out[3]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Ca
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	3	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.12	NaN	3	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	3	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	
5	0x1607	CUS_0xd40	June	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	NaN	3	
6	0x1608	CUS_0xd40	July	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	
7	0x1609	CUS_0xd40	August	NaN	23	#F%$D@*&8	Scientist	19114.12	1824.843333	3	

In [4]:
```
df = data.copy()
```

In [5]:
```
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 [6]:
```
df[df.duplicated()]
```

Out[6]:

ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate

🦹 Insights

- This data has no duplicates.

🧹 Column wise Cleaning

In [7]:
```
categorical_columns = df.select_dtypes(include=['object', 'category']).columns
numerical_columns = df.select_dtypes(include=['number']).columns
```

In [8]:
```
categorical_columns
```

Out[8]:
```
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 [9]:
```
numerical_columns
```

Out[9]:
```
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 [10]:
```
df.sample()
```

Out[10]:

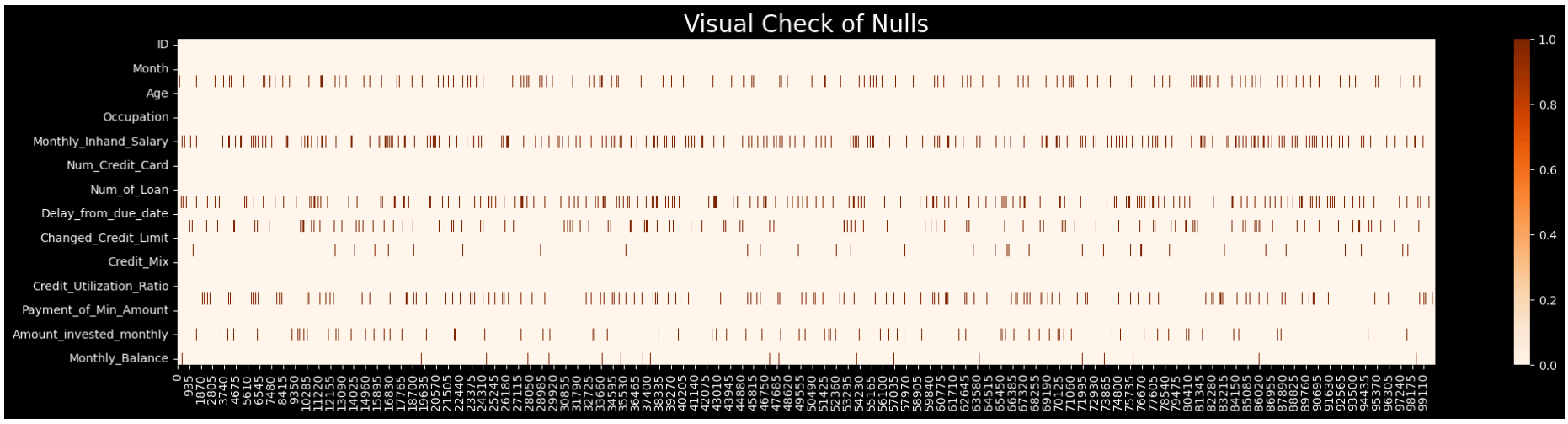
	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_C
73649	0x1c58b	CUS_0xb742	February	Martellc	18	630-69-4858	Media_Manager	40846.36	3614.863333	3	

🔹 ❓ Null Detection

In [11]:
```
df.isna().sum()
```

Out[11]:
```
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=(24,5))
plt.style.use('dark_background')
sns.heatmap(df.isnull().T,cmap='Oranges')
plt.title('Visual Check of Nulls',fontsize=20)
plt.show()
```



In [ ]:
```
"""# wiping out irrevelant data

#tried not working... look into it
'''def replace_with_null(df):
    # regex pattern to match underscores or hyphens
    pattern = r'[_-]'
    df.replace(to_replace=pattern, value=np.nan, regex=True, inplace=True)
    return df'''

def replace_with_null(value):
    pattern = r'[!@9#%8|#F%$D@*&8|NA]'

    if isinstance(value, str) and re.search(pattern, value):
        return None
    return value

df = df.applymap(replace_with_null)
df.head(8)"""

# Creating a lot of ruckus.. lets try column wise
```

Out[ ]:
```
"# wiping out irrevelant data\n\n#tried not working... look into it\n'''def replace_with_null(df):\n    # regex pattern to match underscores or hyphens\n    pattern = r'[_-]'\n    df.replace(to_replace=pattern, value=np.nan, regex=True, inplace=True)\n    return df'''\n\ndef replace_with_null(value):\n    pattern = r'[!@9#%8|#F%$D@*&8|NA]'\n\n    if isinstance(value, str) and re.search(pattern, value):\n        return None\n    return value\n\ndf = df.applymap(replace_with_null)\ndf.head(8)"
```

🦹 Perfect columns

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

1️⃣ Name

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

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

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

Out[ ]:
```
10139
```

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

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

Out[ ]:
```
np.int64(0)
```

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

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

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

Out[ ]:
```
10139
```

🦹 Insights

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

2️⃣ Age

> After cleaning feature name is 'age'

In [ ]:
```
df.Age.nunique()
```

Out[ ]:
```
1788
```

In [ ]:
```
df.Age.unique()
```

Out[ ]:
```
array(['23', '-500', '28_', ..., '4808_', '2263', '1342'], dtype=object)
```

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

Out[ ]:
```
np.int64(0)
```

In [ ]:
```
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] if not x.mode().empty else np.nan))
```

In [ ]:
```
df.Age.unique() ,  df.Age.dtype
```

Out[ ]:
```
(array([  23.,   nan,   28., ..., 6476., 2263., 1342.]), dtype('float64'))
```

In [ ]:
```
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 happening is not mentioned.
    else:
        return np.nan
```

In [ ]:
```
df['age'] = df.groupby('Customer_ID')['Age'].transform(fill_mode)
```

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

Out[ ]:
```
(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 [ ]:
```
df['age'] = df['age'].astype(int)
```

In [ ]:
```
df.age.min() , df.age.max()
```

Out[ ]:
```
(np.int64(14), np.int64(56))
```

In [ ]:
```
df['age'] = df['age'].astype(int)
```

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

Out[ ]:
```
Customer_ID  Name
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]
Name: Age, Length: 12500, dtype: object
```

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

Out[ ]:
```
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]
Name: age, Length: 12500, dtype: object
```

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

Out[ ]:
```
np.int64(0)
```

In [ ]:
```
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.

3️⃣ Social-Security-Number

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Ir
87843	0x218b5	CUS_0x14f1	April	Viswanathag	058-16-7275	Engineer	40701.88	NaN	8	9	

In [ ]:
```
df.SSN.dtype
```

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

In [ ]:
```
df.SSN.nunique()
```

Out[ ]:
```
12501
```

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

Out[ ]:
```
np.int64(5572)
```

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

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

Out[ ]:
```
np.int64(5572)
```

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

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

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Intere
1008	0x1bea	CUS_0x3fbf	January	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42_	2300.368333	6	5	
1009	0x1beb	CUS_0x3fbf	February	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	2300.368333	6	5	
1010	0x1bec	CUS_0x3fbf	March	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	2300.368333	6	5	
1011	0x1bed	CUS_0x3fbf	April	Clara Ferreira-Marquesi	646-12-1414	_____	27796.42	2300.368333	6	5	
1012	0x1bee	CUS_0x3fbf	May	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	2300.368333	6	5	
1013	0x1bef	CUS_0x3fbf	June	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	2300.368333	6	5	
1014	0x1bf0	CUS_0x3fbf	July	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	2300.368333	6	5	
1015	0x1bf1	CUS_0x3fbf	August	Clara Ferreira-Marquesi	646-12-1414	Accountant	27796.42	2300.368333	6	5	

In [ ]:
```
# could have also used this method.
"""# Function to compute mode and fill NaN
def fill_mode(series):
    mode_value = series.mode()
    if not mode_value.empty:
        return series.fillna(mode_value[0])
```

```
    else:
        return series

# Apply the function to each group
df['SSN'] = df.groupby('Customer_ID')['SSN'].transform(fill_mode)"""
```

Out[ ]:
```
"# Function to compute mode and fill NaN\ndef fill_mode(series):\n    mode_value = series.mode()\n    if not mode_value.empty:\n return series.fillna(mode_value[0])\n    else:\n        return series\n\n# Apply the function to each group\ndf['SSN'] = df.grou pby('Customer_ID')['SSN'].transform(fill_mode)"
```

🕵️ Insights

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

4️⃣ Occupation

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

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

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

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

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

Out[ ]:
```
np.int64(7062)
```

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

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

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

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:
```
(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.

5️⃣ Annual_Income

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest
98406	0x25698	CUS_0xac4a	July	Alexw	514-32-8216	Mechanic	25883.65	2324.970833	4	6	

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

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

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

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

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

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

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

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Inte
45744	0x1220a	CUS_0x4c96	January	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	
45745	0x1220b	CUS_0x4c96	February	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	
45746	0x1220c	CUS_0x4c96	March	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	
45747	0x1220d	CUS_0x4c96	April	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	
45748	0x1220e	CUS_0x4c96	May	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	
45749	0x1220f	CUS_0x4c96	June	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	
45750	0x12210	CUS_0x4c96	July	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	
45751	0x12211	CUS_0x4c96	August	Ann Saphirw	961-65-0909	Teacher	99868.83	8332.4025	2	3	

🕵️ 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 [ ]:
```
df['Monthly_Inhand_Salary'].dtype
```

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

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

Out[ ]:
```
np.int64(15002)
```

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

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

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_R
0	0x1602	CUS_0xd40	January	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	
5	0x1607	CUS_0xd40	June	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	
6	0x1608	CUS_0xd40	July	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	
7	0x1609	CUS_0xd40	August	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	3	4	

🦹‍♂️Insights

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

7️⃣ Num_Bank_Accounts

In [ ]:
```
df.Num_Bank_Accounts.dtype
```

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

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

Out[ ]:
```
np.int64(0)
```

In [ ]:
```
df.Num_Bank_Accounts.nunique()
```

Out[ ]:
```
943
```

In [ ]:
```
df['Num_Bank_Accounts'] = df['Num_Bank_Accounts'].replace(-1,0)
```

In [ ]:
```
df['Num_Bank_Acc'] = df.groupby('Customer_ID')['Num_Bank_Accounts'].transform(fill_mode)
```

In [ ]:
```
df[df['Name']=='Thomass'][['SSN','Name','Month','Num_Bank_Accounts','Num_Bank_Acc']]
```

Out[ ]:

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

Out[ ]:
```
(np.int64(0), np.int64(10))
```

In [ ]:
```
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

8️ Num_Credit_Card

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Credit_Card	Interest_Rate	Num_of_Loan
53332	0x14e7e	CUS_0x12cb	May	Sakarif	075-60-6485	Writer	45188.52	3868.71	10	22	7

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

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

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

Out[ ]:
```
np.int64(0)
```

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

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

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

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Credit_Card	Interest_Rate	Num_of_Loa
66464	0x19b72	CUS_0x7666	January	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	4	7	
66465	0x19b73	CUS_0x7666	February	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	4	7	
66466	0x19b74	CUS_0x7666	March	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	18	7	
66467	0x19b75	CUS_0x7666	April	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	4	7	
66468	0x19b76	CUS_0x7666	May	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	1089	7	
66469	0x19b77	CUS_0x7666	June	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	141	7	
66470	0x19b78	CUS_0x7666	July	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	4	7	
66471	0x19b79	CUS_0x7666	August	Dolano	922-59-5950	Entrepreneur	113284.84	9516.403333	4	7	

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

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

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

In [ ]:
```
df.No_of_Credit_Card.min() , df.No_of_Credit_Card.max()
```

Out[ ]:
```
(np.int64(0), np.int64(11))
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Interest_Rate	Num_of_Loan	Type_of_Loan	D
20882	0x905c	CUS_0xa73b	March	Niveditak	341-87-3689	Scientist	26526.93	2199.5775	13	4	Student Loan, Not Specified, Home Equity Loan,...	

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

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

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

Out[ ]:
```
np.int64(0)
```

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

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Interest_Rate	Num_of_Loan	Type_of_Loa
23512	0x9fc6	CUS_0x2fab	January	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	5788	3	Credit-Builde Loan, Hom Equity Loar and Mor.
23513	0x9fc7	CUS_0x2fab	February	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	10	3	Credit-Builde Loan, Hom Equity Loar and Mor.
23514	0x9fc8	CUS_0x2fab	March	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	10	-100	Credit-Builde Loan, Hom Equity Loar and Mor.
23515	0x9fc9	CUS_0x2fab	April	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	10	3	Credit-Builde Loan, Hom Equity Loar and Mor.
23516	0x9fca	CUS_0x2fab	May	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	10	3_	Credit-Builde Loan, Hom Equity Loar and Mor.
23517	0x9fcb	CUS_0x2fab	June	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	10	3	Credit-Builde Loan, Hom Equity Loar and Mor.
23518	0x9fcc	CUS_0x2fab	July	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	10	3	Credit-Builde Loan, Hom Equity Loar and Mor.
23519	0x9fcd	CUS_0x2fab	August	Anna Yukhananovd	411-00-6543	Manager	75804.94	6120.078333	10	3_	Credit-Builde Loan, Hom Equity Loar and Mor.

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

Out[ ]:
```
(np.int64(1), np.int64(34))
```

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

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Delay_from_due_d
20978	0x90ec	CUS_0x7e7f	March	Emotou	171-52-8384	Doctor	73439.26	6144.938333	4	Debt Consolidation Loan, Auto Loan, Auto Loan,...	

🕵️ 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

1️⃣0️⃣ Num_of_Loan

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

Out[ ]:
```
(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 [ ]:
```
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 [ ]:
```
df['Num_of_Loan'] = df['Type_of_Loan'].apply(calculate_num_of_loans)
```

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

Out[ ]:
```
np.int64(0)
```

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

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

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

Out[ ]:
```
(np.int64(0), np.int64(9))
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Delay_from_due_
59096	0x17046	CUS_0x44e9	January	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
59097	0x17047	CUS_0x44e9	February	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
59098	0x17048	CUS_0x44e9	March	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
59099	0x17049	CUS_0x44e9	April	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
59100	0x1704a	CUS_0x44e9	May	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
59101	0x1704b	CUS_0x44e9	June	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
59102	0x1704c	CUS_0x44e9	July	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
59103	0x1704d	CUS_0x44e9	August	Barri	482-45-6373	Architect	19656.96	1339.080000	9	Payday Loan, Credit-Builder Loan, Payday Loan,...	
63560	0x18a6e	CUS_0xb47f	January	Barri	429-78-0806	Scientist	30773.87	2593.489167	2	Auto Loan, and Debt Consolidation Loan	
63561	0x18a6f	CUS_0xb47f	February	Barri	429-78-0806	Scientist	30773.87	2593.489167	2	Auto Loan, and Debt Consolidation Loan	
63562	0x18a70	CUS_0xb47f	March	Barri	429-78-0806	Scientist	30773.87	2593.489167	2	Auto Loan, and Debt Consolidation Loan	
63563	0x18a71	CUS_0xb47f	April	Barri	429-78-0806	Scientist	30773.87	2593.489167	2	Auto Loan, and Debt Consolidation Loan	
63564	0x18a72	CUS_0xb47f	May	Barri	429-78-0806	Scientist	30773.87	2593.489167	2	Auto Loan, and Debt Consolidation Loan	
63565	0x18a73	CUS_0xb47f	June	Barri	429-78-0806	Scientist	30773.87	2593.489167	2	Auto Loan, and Debt Consolidation Loan	
63566	0x18a74	CUS_0xb47f	July	Barri	429-78-0806	Scientist	30773.87	2297.979398	2	Auto Loan, and Debt Consolidation Loan	
63567	0x18a75	CUS_0xb47f	August	Barri	429-78-0806	Scientist	30773.87	2297.979398	2	Auto Loan, and Debt Consolidation Loan	

🕵️ 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.

1️⃣ 1️⃣ Type_of_Loan

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

Out[ ]:
```
(dtype('O'), np.int64(11408))
```

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

Out[ ]:
```
(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 Loan'],
       dtype=object))
```

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

Out[ ]:
```
np.int64(11408)
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Delay_fr
63476	0x189ee	CUS_0xb967	May	Jonathanm	914-80-2762	Doctor	33131.600	2770.966667	0	NaN	
85611	0x20ba1	CUS_0x1848	April	Glennl	766-42-5182	Musician	33834.470	2630.539167	0	NaN	
1169	0x1cdb	CUS_0x32a5	February	"John ODonnell"f	900-72-1051	Engineer	16852.635	1691.386250	0	NaN	
98290	0x255ec	CUS_0xa407	March	McBrideo	354-91-6159	Entrepreneur	34599.940	2942.328333	0	NaN	
78065	0x1df6b	CUS_0x7968	February	Suzanne Barlyne	293-25-3124	Media_Manager	142725.800	11867.816670	0	NaN	
56929	0x16393	CUS_0x8aa2	February	Leila Abboudg	231-43-8484	Writer	68103.810	5940.317500	0	NaN	
86626	0x21194	CUS_0x988c	March	Kevinz	083-64-0011	Writer	29101.850	2485.154167	0	NaN	
53566	0x14fdc	CUS_0x383c	July	Jamesk	329-16-0543	Teacher	89342.580	7644.215000	0	NaN	
53531	0x14fa9	CUS_0x4feb	April	Skariachanc	921-30-7278	Journalist	33726.600	2477.672444	0	NaN	
74528	0x1cab2	CUS_0x614f	January	Rhysj	622-65-0067	Manager	44624.660	3696.721667	0	NaN	

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

Out[ ]:
```
array([0])
```

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

In [ ]:
```
df[df['Name']=='Stempelp'][['Name','SSN','Occupation','Num_of_Loan','Type_of_Loan','interest_rate']]
```

Out[ ]:

	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	049-48-0823	Media_Manager	3	Debt Consolidation Loan, Payday Loan, and Mort...	11
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 [ ]:
```
df.groupby('Type_of_Loan')[['Customer_ID']].count().sort_values(by='Customer_ID',ascending=False)[:12]
```

Out[ ]:

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'.

1️⃣ 2️⃣ Delay_from_due_date

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

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:
```
np.int64(591)
```

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

In [ ]:
```
df['delay_from_due_date'] = df.groupby('Customer_ID')['Delay_from_due_date'].transform(fill_mode)
```

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

Out[ ]:
```
(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 [ ]:
```
df['delay_from_due_date'].nunique() , df['delay_from_due_date'].unique()
```

Out[ ]:
```
(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 [ ]:
```
df['Delay_from_due_date'].min() , df['Delay_from_due_date'].max()
```

Out[ ]:
```
(np.int64(0), np.int64(67))
```

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

Out[ ]:
```
(np.int64(0), np.int64(62))
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Delay_from_due_da
84003	0x20235	CUS_0xb38f	April	Koh Guib	076-83-7903	Musician	56500.7	4765.391667	2	Student Loan, and Not Specified	

In [ ]:
```
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.

1 3 Num_of_Delayed_Payment

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

Out[ ]:
```
(dtype('O'), np.int64(7002))
```

In [ ]:
```
df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].replace({'-': np.nan, '_': np.nan}, regex=True)

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

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

Out[ ]:
```
(dtype('float64'), np.int64(10368))
```

In [ ]:
```
df['No_of_delayed_payment'] = df.groupby('Customer_ID')['Num_of_Delayed_Payment'].transform(fill_mode)
```

In [ ]:
```
df.Num_of_Delayed_Payment.nunique() , df.Num_of_Delayed_Payment.min() , df.Num_of_Delayed_Payment.max()
```

Out[ ]:
```
(695, np.float64(0.0), np.float64(4397.0))
```

In [ ]:
```
df.No_of_delayed_payment.nunique() , df.No_of_delayed_payment.min() , df.No_of_delayed_payment.max()
```

Out[ ]:
```
(29, np.float64(0.0), np.float64(28.0))
```

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

In [ ]:
```
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.

1 3 Changed_Credit_Limit

- Represents the percentage change in credit card limit

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit
40190	0x1017c	CUS_0xabf6	July	Klaymanv	391-94-1112	Mechanic	37421.88	3060.49	3	Credit-Builder Loan, Debt Consolidation Loan, ...	

In [ ]:
```
df.Changed_Credit_Limit.dtype
```

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

In [ ]:
```
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].replace('_',np.nan)
df['Changed_Credit_Limit'] = pd.to_numeric(df['Changed_Credit_Limit'],errors='coerce')
```

In [ ]:
```
df.Changed_Credit_Limit.min() , df.Changed_Credit_Limit.max() , df.Changed_Credit_Limit.isna().sum()
```

Out[ ]:
```
(np.float64(-6.49), np.float64(36.97), np.int64(2091))
```

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

In [ ]:
```
df.Changed_Credit_Limit.min() , df.Changed_Credit_Limit.max() , df.Changed_Credit_Limit.isna().sum()
```

Out[ ]:
```
(np.float64(-6.49), np.float64(36.97), np.int64(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.

1 4 Num_Credit_Inquiries

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit_Lim
4706	0x3194	CUS_0x472b	March	Carrickt	390-24-8662	Writer	29661.64	2377.803333	7	Auto Loan, Personal Loan, Personal Loan, Credi...	12

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

Out[ ]:
```
(dtype('float64'), np.int64(1965))
```

In [ ]:
```
df['No_of_credit_inquiries'] = df.groupby('Customer_ID')['Num_Credit_Inquiries'].transform(fill_mode)
```

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

Out[ ]:
```
(dtype('float64'), np.int64(0))
```

In [ ]:
```
df['No_of_credit_inquiries'] = df['No_of_credit_inquiries'].astype(int)
```

In [ ]:
```
df.No_of_credit_inquiries.min() , df.No_of_credit_inquiries.max()
```

Out[ ]:
```
(np.int64(0), np.int64(17))
```

In [ ]:
```
df[df.No_of_credit_inquiries==17].sample()
```

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Cre
14180	0x6916	CUS_0x392a	May	Jan Strupczewskiq	001-51-4145	Lawyer	14171.23	1066.935833	2	Student Loan, and Auto Loan	

In [ ]:
```
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.

15 Credit_Mix

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Cr
8579	0x4845	CUS_0x81d	April	Georgiopoulosx	806-28-3875	Scientist	43452.15	3864.0125	5	Student Loan, Payday Loan, Credit-Builder Loan...	

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

Out[ ]:
```
np.int64(20195)
```

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

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

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

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit
68704	0x1a892	CUS_0xae9e	January	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	
68705	0x1a893	CUS_0xae9e	February	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	
68706	0x1a894	CUS_0xae9e	March	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	
68707	0x1a895	CUS_0xae9e	April	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	
68708	0x1a896	CUS_0xae9e	May	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	
68709	0x1a897	CUS_0xae9e	June	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	
68710	0x1a898	CUS_0xae9e	July	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	
68711	0x1a899	CUS_0xae9e	August	Bansalp	345-91-4839	Developer	19540.67	1891.389167	6	Not Specified, Mortgage Loan, Debt Consolidati...	

🕵️ Insights

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

16 Outstanding_Debt

In [ ]:
```
df.Outstanding_Debt.dtype
```

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

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

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

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

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

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

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

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

In [ ]:
```
df.Outstanding_Debt.min() , df.Outstanding_Debt.max()
```

Out[ ]:
```
(np.float64(0.23), np.float64(4998.07))
```

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

Out[ ]:
```
np.int64(0)
```

Insights

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

17 Credit_Utilization_Ratio

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

Out[ ]:
```
(dtype('float64'), np.int64(0))
```

Insights

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

18 Credit_History_Age

- Represents the age of credit history of the person

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Cre
64393	0x18f4f	CUS_0x7270	February	ODonnell"m	834-23-2429	Journalist	24859.96	2045.663333	3	Not Specified, Debt Consolidation Loan, and Au...	

In [ ]:
```
df.head(1)
```

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit_Limit
0	0x1602	CUS_0xd40	January	Aaron Maashoh	821-00-0265	Scientist	19114.12	1824.843333	4	Auto Loan, Credit-Builder Loan, Personal Loan,...	11.27

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

Out[ ]:
```
np.int64(9030)
```

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

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

In [ ]:
```
df['cha'] , df['cha1']
```

Out[ ]:
```
(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 [ ]:
```
df['cha1'] = df.cha1.astype(int)
```

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

In [ ]:
```
df[df.Customer_ID=='CUS_0xff3']
```

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Cred
5168	0x344a	CUS_0xff3	January	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	
5169	0x344b	CUS_0xff3	February	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	
5170	0x344c	CUS_0xff3	March	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	
5171	0x344d	CUS_0xff3	April	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	
5172	0x344e	CUS_0xff3	May	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	
5173	0x344f	CUS_0xff3	June	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	
5174	0x3450	CUS_0xff3	July	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	
5175	0x3451	CUS_0xff3	August	Somervilled	726-35-5322	Scientist	17032.785	1176.39875	3	Personal Loan, Mortgage Loan, and Auto Loan	

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

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

Out[ ]:
```
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
Name: credit_history_age, Length: 12500, dtype: int64
```

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit_Li
48906	0x13490	CUS_0x737a	March	Toml	643-54-4901	Engineer	142277.72	12004.47667	0	No loan taken	2

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'

19 Payment_of_Min_Amount

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

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:
```
(3, array(['No', 'NM', 'Yes'], dtype=object))
```

In [ ]:
```
df['Payment_of_Min_Amount'] = df['Payment_of_Min_Amount'].replace('NM','No')
```

In [ ]:
```
df.Payment_of_Min_Amount.isna().sum() , df.Payment_of_Min_Amount.nunique() , df.Payment_of_Min_Amount.unique()
```

Out[ ]:
```
(np.int64(0), 2, array(['No', 'Yes'], dtype=object))
```

Insights

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

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credi
42440	0x10eae	CUS_0x86da	January	Jed Horowitzc	374-78-2060	Manager	19294.46	1392.871667	8	Personal Loan, Not Specified, Home Equity Loan...	

20 Total_EMI_per_month

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

Out[ ]:
```
(dtype('float64'), np.int64(0))
```

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

Out[ ]:
```
(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.

21 Amount_invested_monthly

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit_L
72814	0x1c0a4	CUS_0x86c7	July	Steven C.a	232-33-5704	Entrepreneur	55563.4	4550.283333	0	No loan taken	

In [ ]:
```
df.Amount_invested_monthly.dtype
```

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

In [ ]:
```
df.loc[df['Amount_invested_monthly'] == '__10000__', 'Amount_invested_monthly'] = 0
```

In [ ]:
```
df['Amount_invested_monthly'] = pd.to_numeric(df['Amount_invested_monthly'], errors='coerce')
```

In [ ]:
```
df.Amount_invested_monthly.dtype
```

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

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

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

Out[ ]:
```
np.int64(0)
```

Insights

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

2 2 Payment_Behaviour

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit
81549	0x1f3d3	CUS_0x6a11	June	dee	064-35-8820	Media_Manager	33352.41	2812.3675	2	Auto Loan, and Auto Loan	

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

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

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

Out[ ]:
```
np.int64(0)
```

Insights

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

2 3 Monthly_Balance

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

Out[ ]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Type_of_Loan	Changed_Credit
72002	0x1bbe4	CUS_0x2863	March	Serapiow	150-77-6507	Architect	80011.64	6824.636667	4	Debt Consolidation Loan, Debt Consolidation Lo...	

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

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

Out[ ]:
```
np.int64(1209)
```

In [ ]:
```
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 [ ]:
```
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_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'
]

df = df.reindex(columns=new_column_order)

df.tail(8)
```

Out[ ]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card
99992	0x25fe6	CUS_0x942c	January	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6
99993	0x25fe7	CUS_0x942c	February	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6
99994	0x25fe8	CUS_0x942c	March	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6
99995	0x25fe9	CUS_0x942c	April	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6
99996	0x25fea	CUS_0x942c	May	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6
99997	0x25feb	CUS_0x942c	June	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6
99998	0x25fec	CUS_0x942c	July	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6
99999	0x25fed	CUS_0x942c	August	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6

📂Saving the Cleaned Data

In [ ]:
```
df.to_csv('cleaned_credit_data.csv',index=False)
```

📒 Data Exploration

🗃️Loading the cleaned data

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')
```

In [15]:
```
df = pd.read_csv('cleaned_credit_data.csv')
```

In [16]:
```
pd.set_option('display.max_columns',50)
```

🕵️**Exploratory Data Analysis**

Non Graphical analysis

- Creating month numbers

In [17]:
```
# Converting month names to month numbers
df['Month_Num'] = pd.to_datetime(df['Month'], format='%B').dt.month
df.sample()
```

Out[17]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	I
34930	0xe2ac	CUS_0x4037	March	Tapinsha	33	022-20-2822	Engineer	92509.38	7876.115	7	4	

Info on Data

In [7]:
```
# 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 [8]:
```
# Check all column names
df.columns
```

Out[8]:
```
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 [18]:
```
categorical_columns = df.select_dtypes(include=['object', 'category']).columns
numerical_columns = df.select_dtypes(include=['number']).columns
```

In [10]:
```
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 [11]:
```
df.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
Age	100000.0	33.282000	1.076657e+01	14.000000	24.000000	33.000000	42.000000	5.600000e+01
Annual_Income	100000.0	178506.267553	1.440261e+06	7005.930000	19457.500000	37578.610000	72814.860000	2.419806e+07
Monthly_Inhand_Salary	100000.0	4198.771619	3.187494e+03	303.645417	1626.761667	3096.378333	5961.745000	1.520463e+04
Num_Bank_Accounts	100000.0	5.367840	2.592597e+00	0.000000	3.000000	5.000000	7.000000	1.000000e+01
Num_Credit_Card	100000.0	5.532720	2.067504e+00	0.000000	4.000000	5.000000	7.000000	1.100000e+01
Interest_Rate	100000.0	14.532080	8.741330e+00	1.000000	7.000000	13.000000	20.000000	3.400000e+01
Num_of_Loan	100000.0	3.532880	2.446356e+00	0.000000	2.000000	3.000000	5.000000	9.000000e+00
Delay_from_due_date	100000.0	21.050560	1.476119e+01	0.000000	10.000000	18.000000	28.000000	6.200000e+01
Num_of_Delayed_Payment	100000.0	13.261280	6.199080e+00	0.000000	9.000000	14.000000	18.000000	2.800000e+01
Changed_Credit_Limit	100000.0	10.389303	6.789784e+00	-6.490000	5.320000	9.400000	14.860000	3.697000e+01
Num_Credit_Inquiries	100000.0	5.677760	3.827248e+00	0.000000	3.000000	5.000000	8.000000	1.700000e+01
Outstanding_Debt	100000.0	1426.220376	1.155129e+03	0.230000	566.072500	1166.155000	1945.962500	4.998070e+03
Credit_Utilization_Ratio	100000.0	32.285173	5.116875e+00	20.000000	28.052567	32.305784	36.496663	5.000000e+01
Credit_History_Age	100000.0	18.235920	8.313256e+00	0.000000	12.000000	18.000000	25.000000	3.300000e+01
Total_EMI_per_month	100000.0	1403.118217	8.306041e+03	0.000000	30.306660	69.249473	161.224249	8.233100e+04
Amount_invested_monthly	100000.0	178.363270	1.984724e+02	0.000000	58.325837	116.545252	220.039055	1.977326e+03
Monthly_Balance	100000.0	397.684413	2.171320e+02	0.000000	267.871374	334.806633	467.670597	1.602041e+03
Month_Num	100000.0	4.500000	2.291299e+00	1.000000	2.750000	4.500000	6.250000	8.000000e+00

In [12]:
```
df.describe(include=object)
```

Out[12]:

	ID	Customer_ID	Month	Name	SSN	Occupation	Type_of_Loan	Credit_Mix	Payment_of_Min_Amount	Payment_Behaviour
count	100000	100000	100000	100000	100000	100000	100000	100000	100000	100000
unique	100000	12500	8	10139	12500	15	6261	3	2	6
top	0x25fb6	CUS_0x942c	January	Langep	078-73-5990	Lawyer	No loan taken	Standard	Yes	Low_spent_Small_value_payments
freq	1	8	12500	48	8	7096	11408	45848	52326	27588

🥨 Duplicate Detection

In [13]:
```
df[df.duplicated()]
```

Out[13]:

ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate

🦹 Insights

- No duplicates found

In [18]:
```
# 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 [20]:
```
categorical_columns[1:]
```

Out[20]:
```
Index(['Customer_ID', 'Month', 'Name', 'SSN', 'Occupation', 'Type_of_Loan',
       'Credit_Mix', 'Payment_of_Min_Amount', 'Payment_Behaviour'],
      dtype='object')
```

In [22]:
```
numerical_columns
```

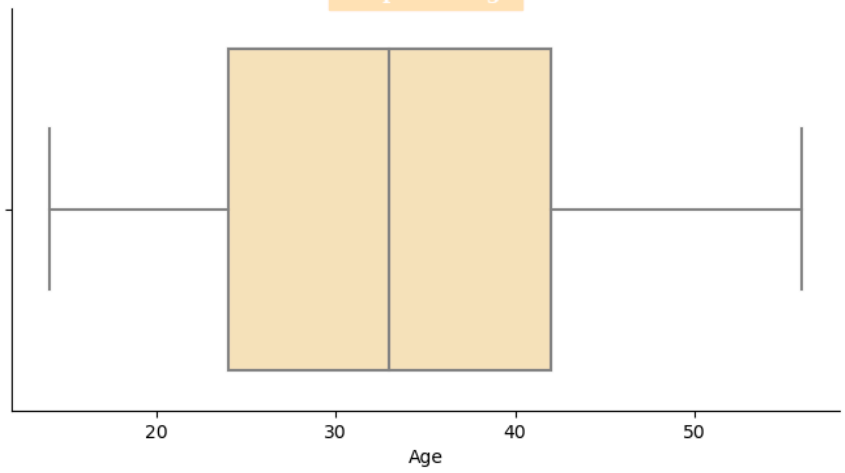
Out[22]:
```
Index(['Age', 'Annual_Income', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts',
       'Num_Credit_Card', 'Interest_Rate', 'Num_of_Loan',
       'Delay_from_due_date', 'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Outstanding_Debt', 'Credit_Utilization_Ratio',
       'Credit_History_Age', 'Total_EMI_per_month', 'Amount_invested_monthly',
       'Monthly_Balance', 'Month_Num'],
      dtype='object')
```

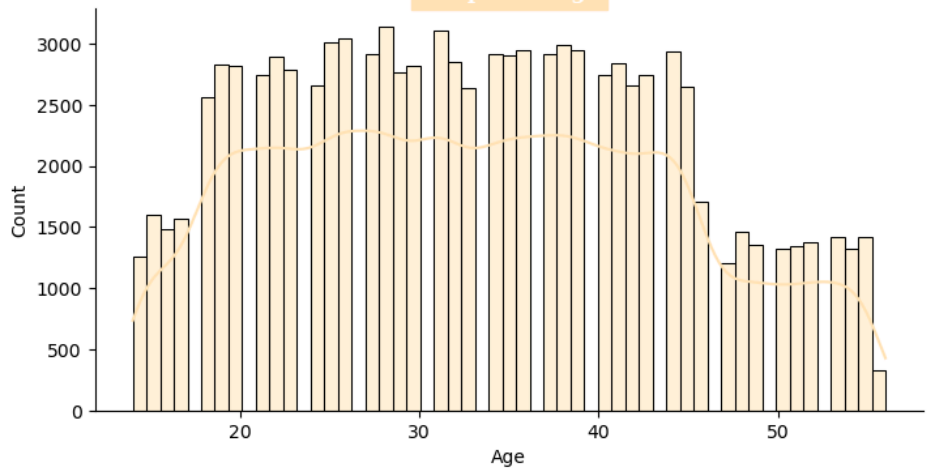
📊 **Graphical analysis**

In [19]:
```
cp = [
    "#FFE5B4",  # Very Light Orange
    "#FFD89B",  # Light Orange
    "#FFC074",  # Peach Orange
    "#FFA94D",  # Sandy Orange
    "#FF9229",  # Orange (Saffron)
    "#FF7518",  # Deep Saffron
    "#FF6700",  # Bright Orange
    "#FF4500",  # Orange-Red
    "#FF8C00",  # Dark Orange
    "#FF7F24",# Saffron Orange
    "#FFE5B4",  # Very Light Orange
    "#FFD89B",  # Light Orange
    "#FFC074",  # Peach Orange
    "#FFA94D",  # Sandy Orange
    "#FF9229",  # Orange (Saffron)
    "#FF7518",  # Deep Saffron
    "#FF6700",  # Bright Orange
    "#FF4500",  # Orange-Red
    "#FF8C00",  # Dark Orange
    "#FF7F24"
]
```

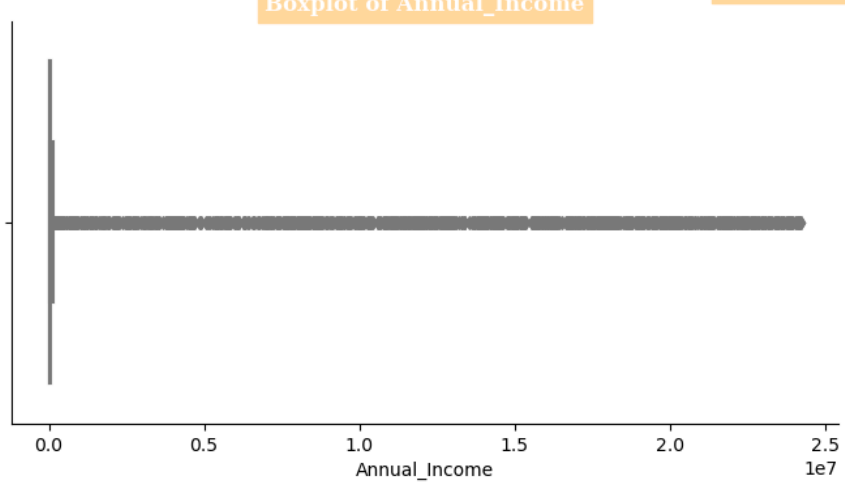
In [7]:
```
plt.style.use('default')
plt.style.use('seaborn-bright')

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

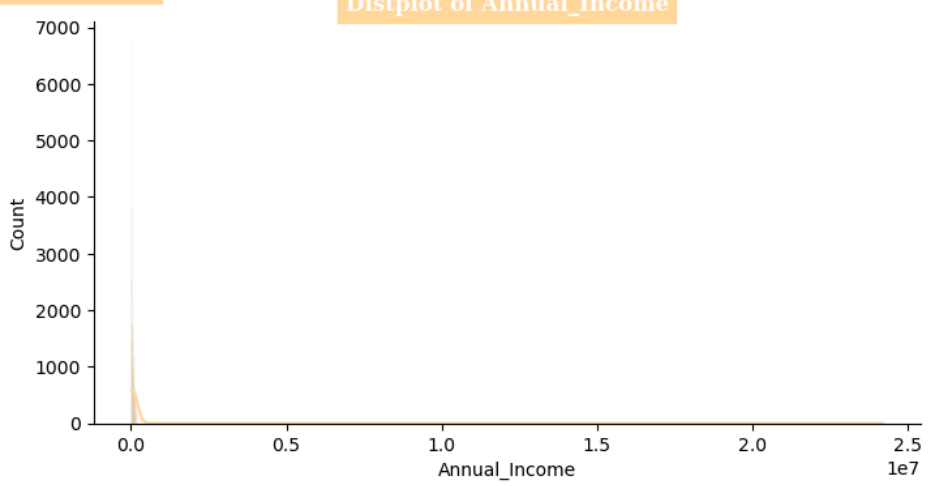
plot of Age



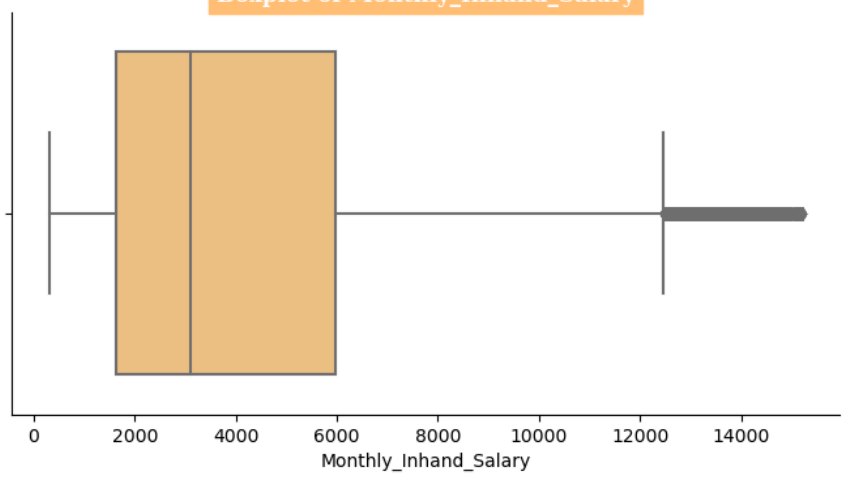


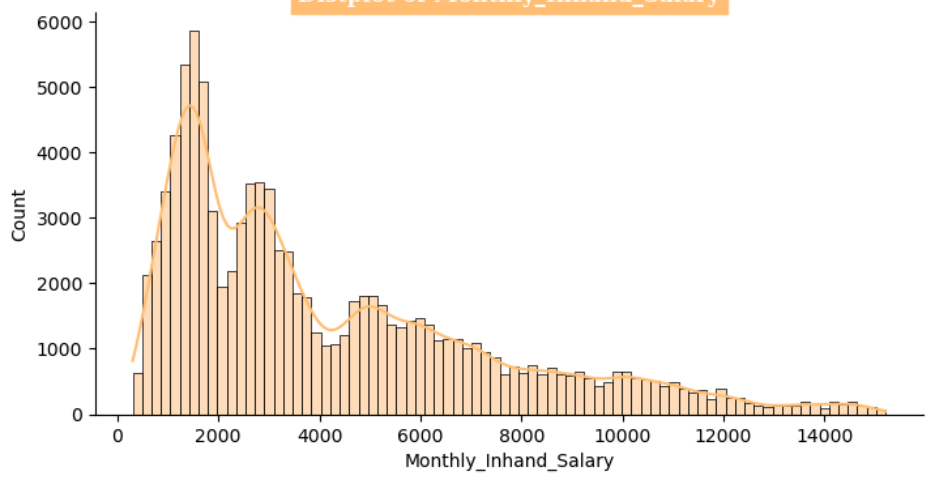
plot of Annual_Income



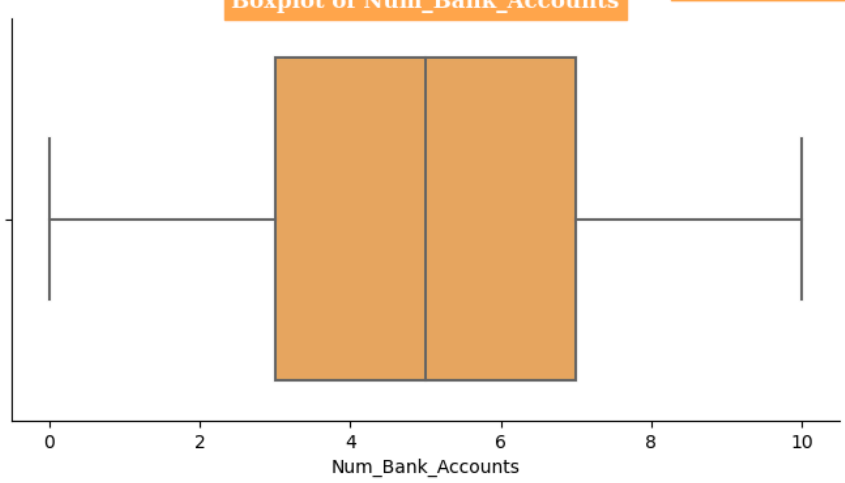


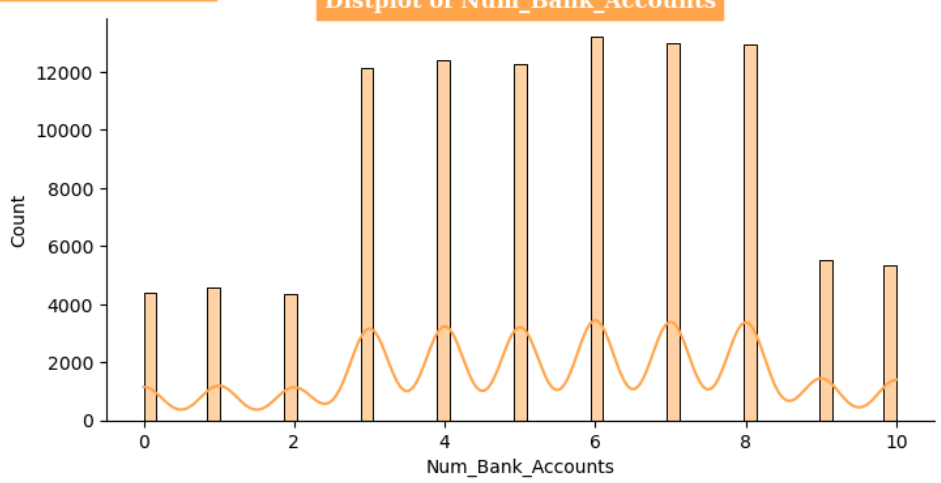
plot of Monthly_Inhand_Salary



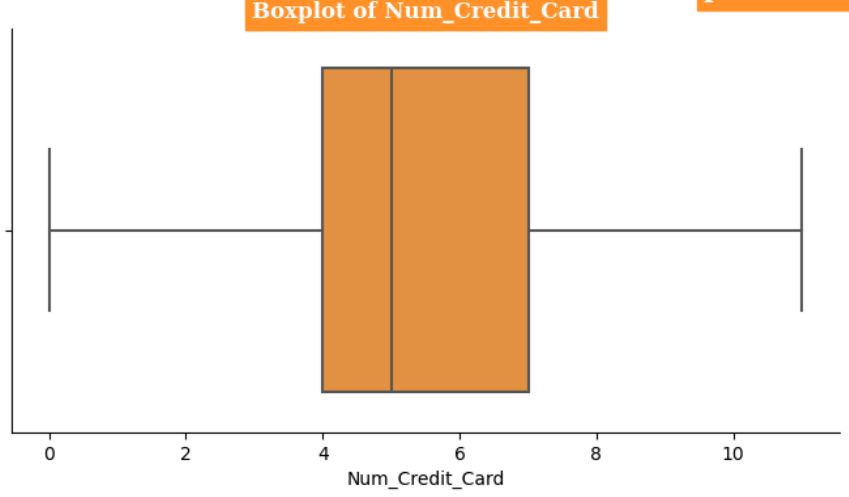


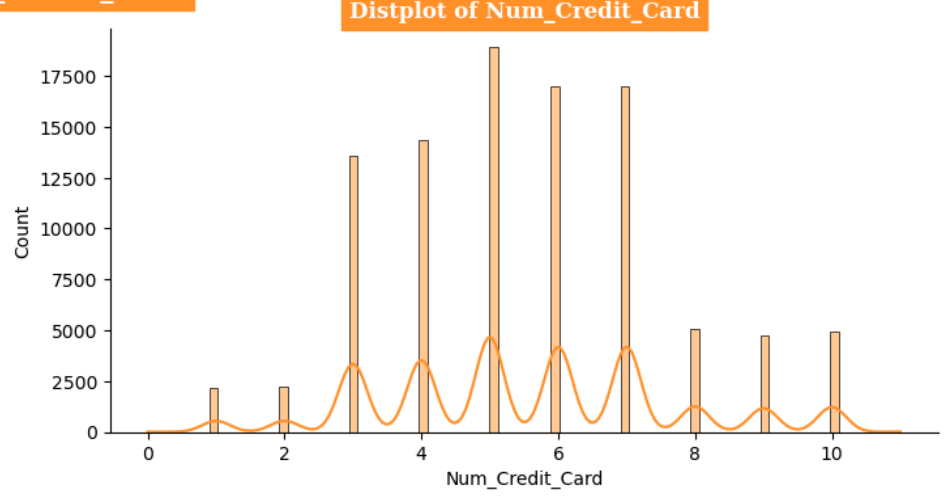
plot of Num_Bank_Accounts



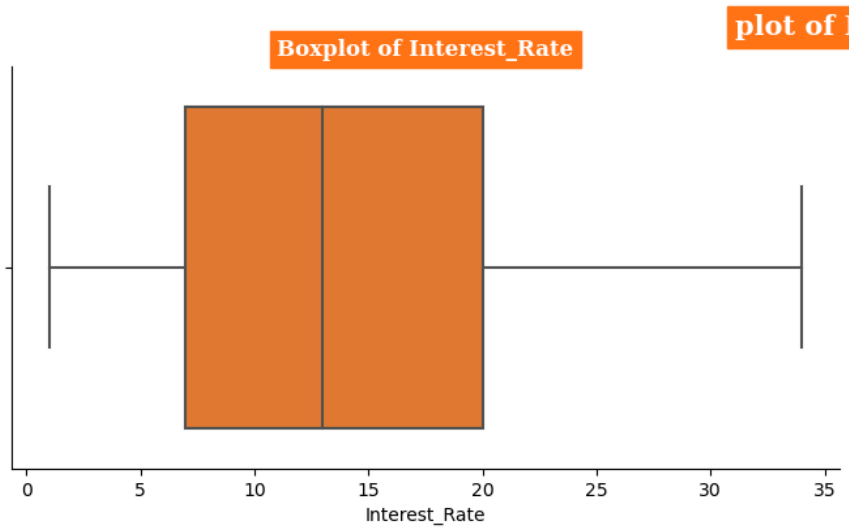


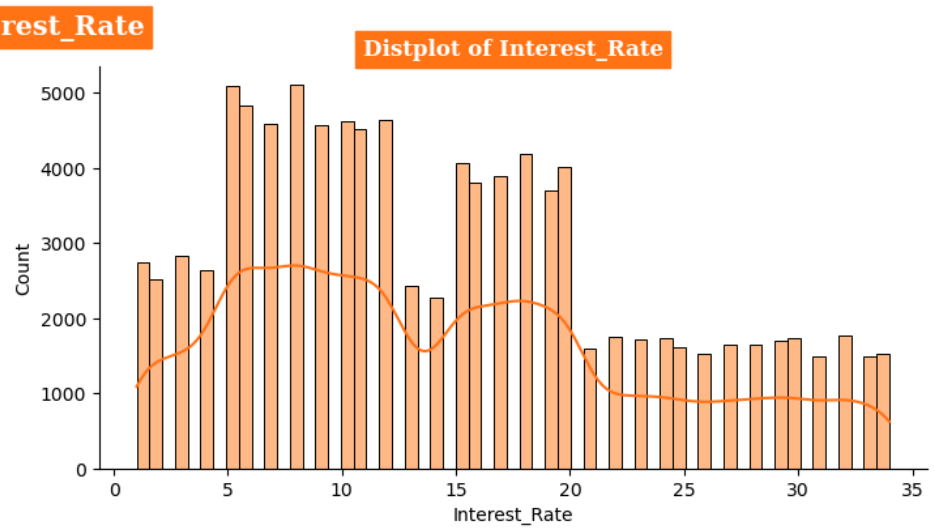
plot of Num_Credit_Card



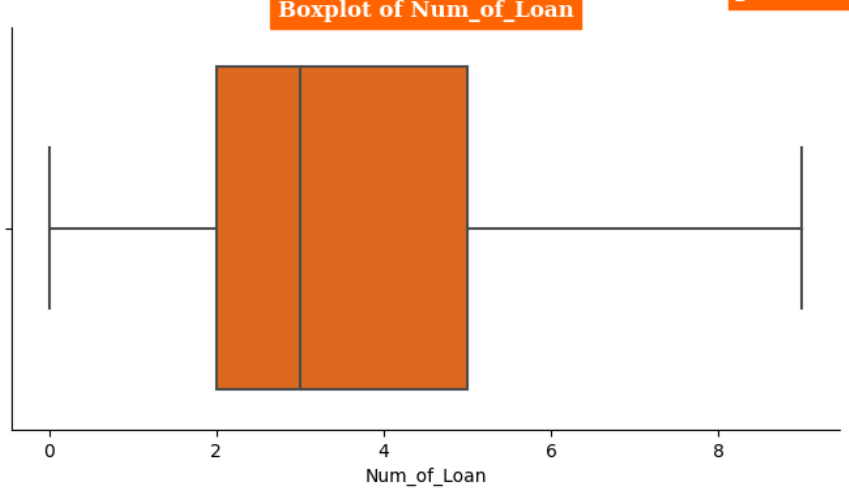


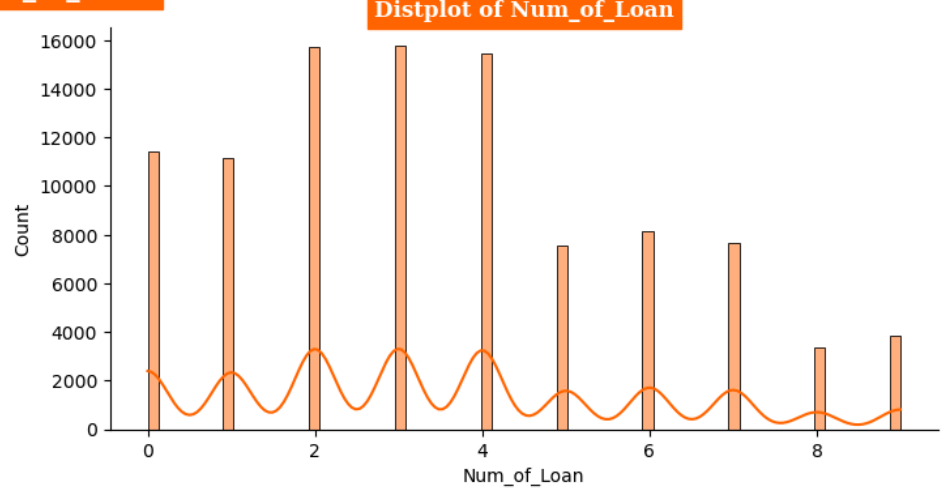
plot of Interest_Rate



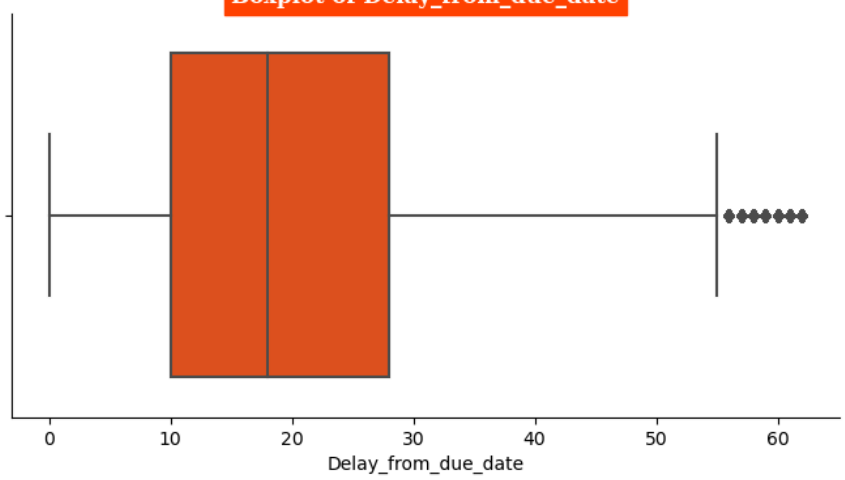


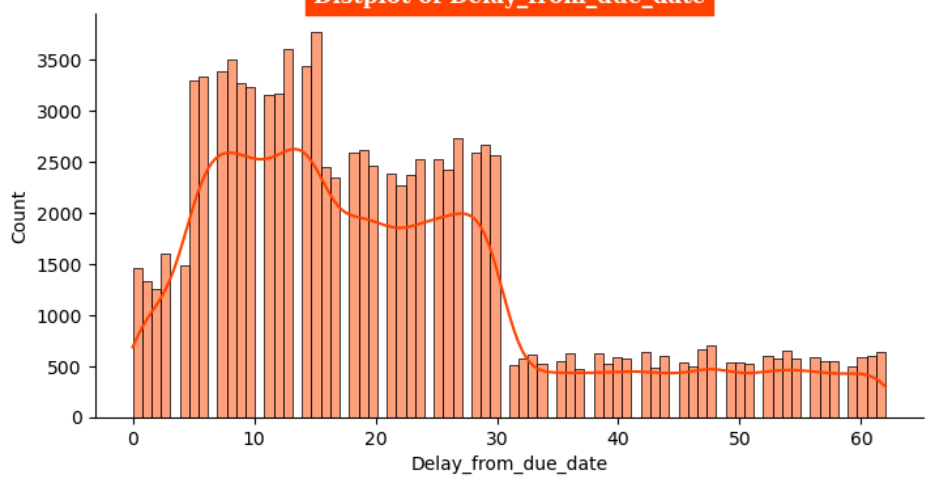
plot of Num_of_Loan



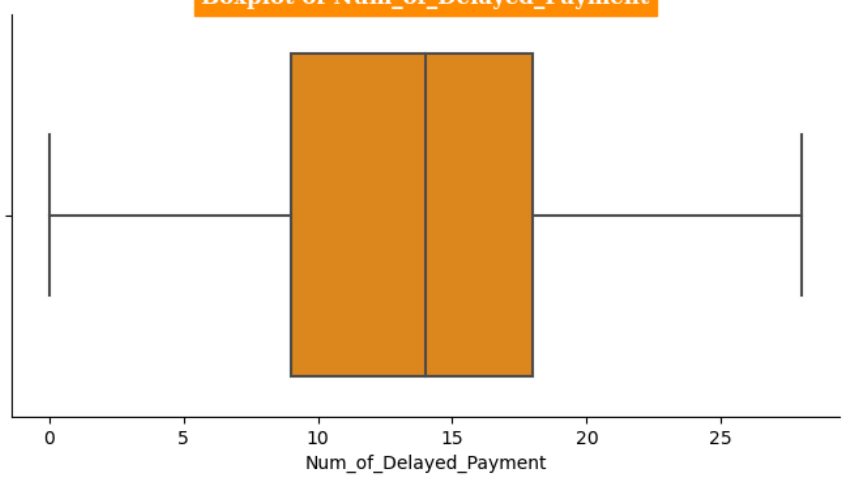


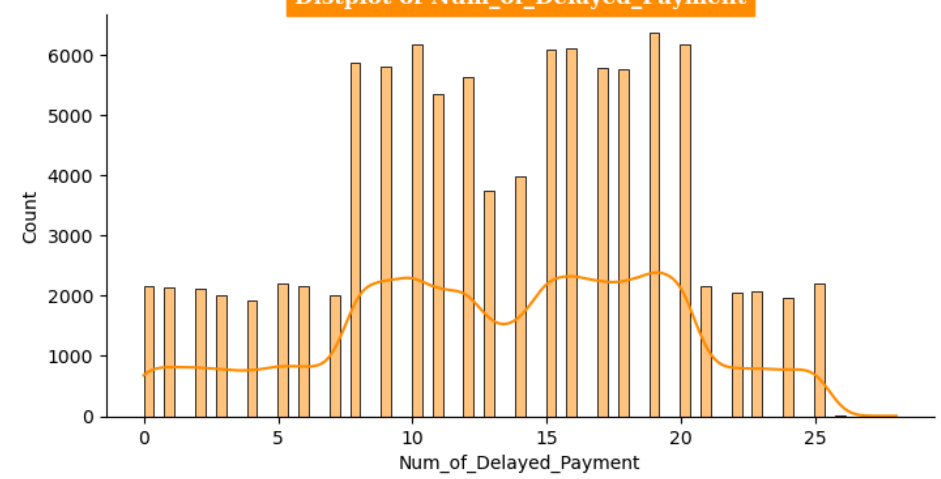
plot of Delay_from_due_date



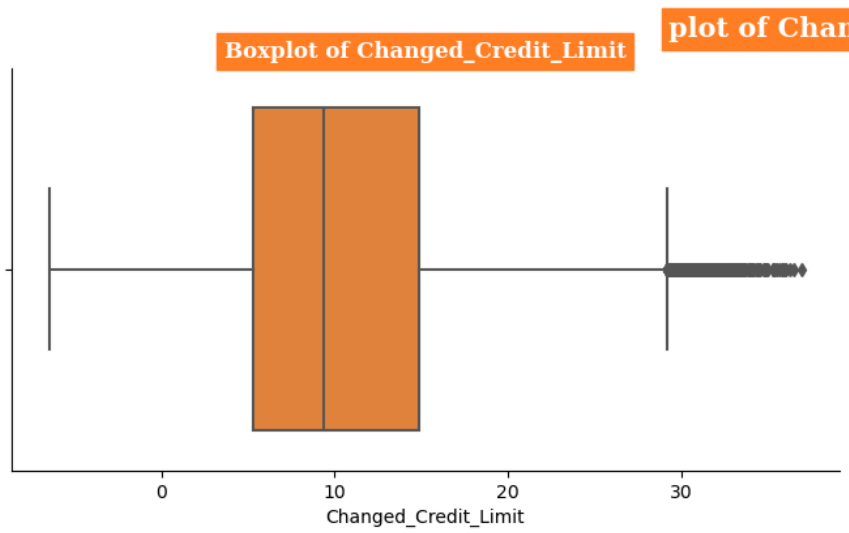


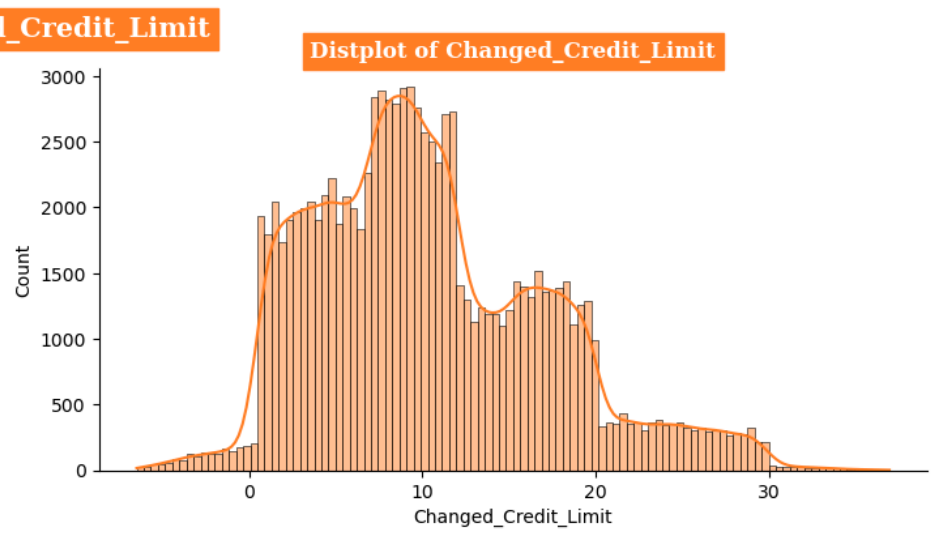
plot of Num_of_Delayed_Payment





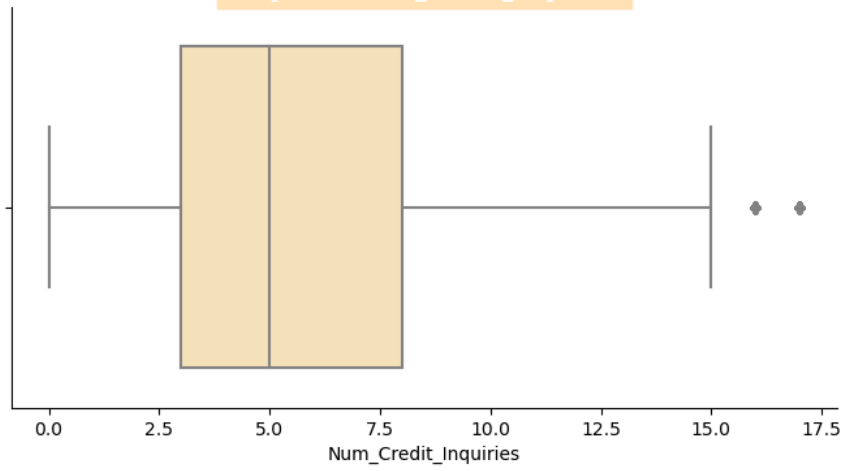
plot of Changed_Credit_Limit



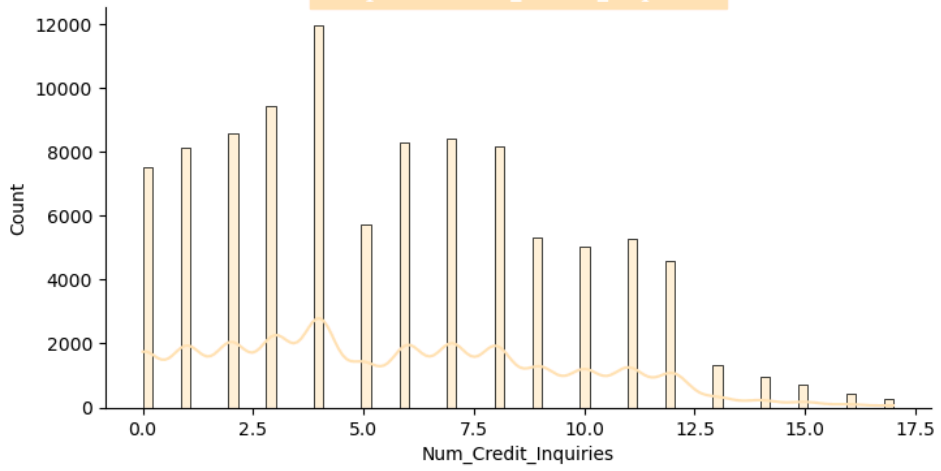


plot of Num_Credit_Inquiries

Boxplot of Num_Credit_Inquiries

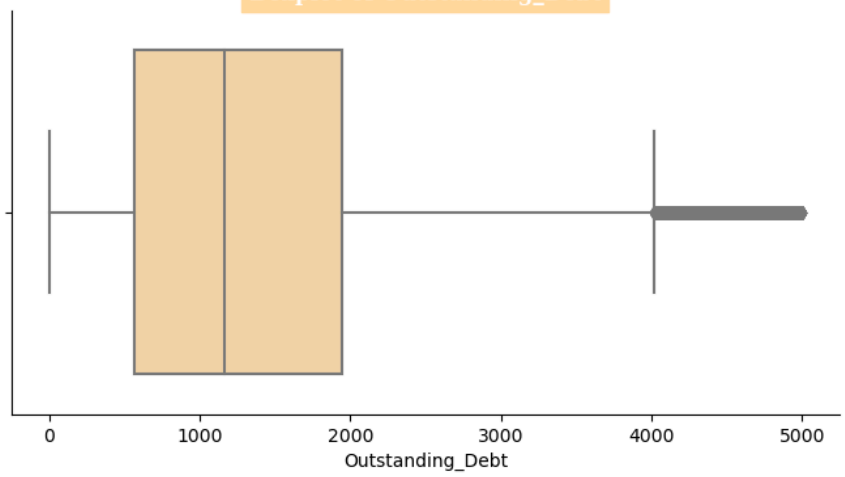


Distplot of Num_Credit_Inquiries

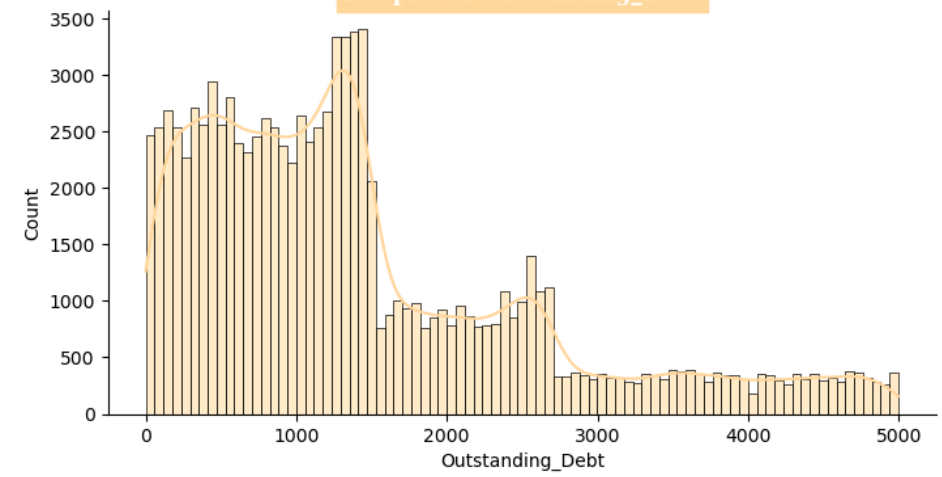


plot of Outstanding_Debt

Boxplot of Outstanding_Debt

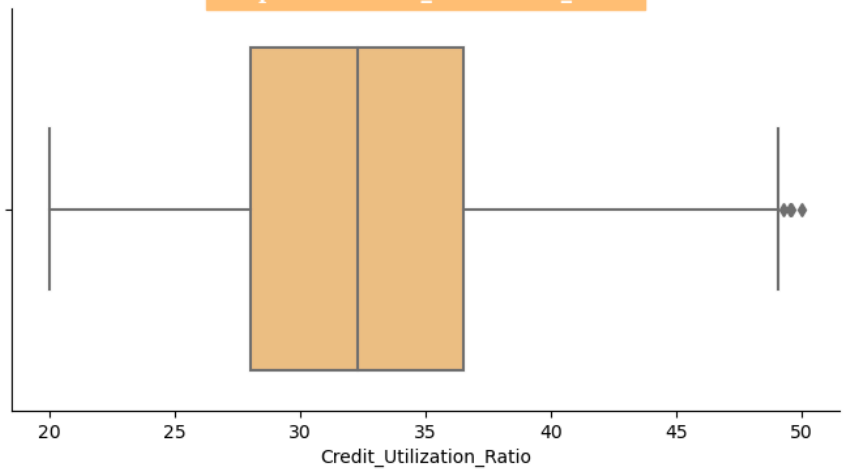


Distplot of Outstanding_Debt

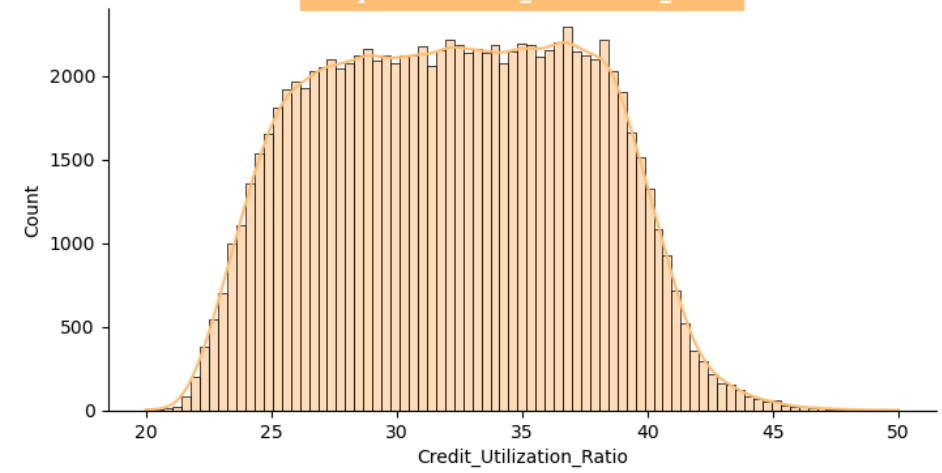


plot of Credit_Utilization_Ratio

Boxplot of Credit_Utilization_Ratio

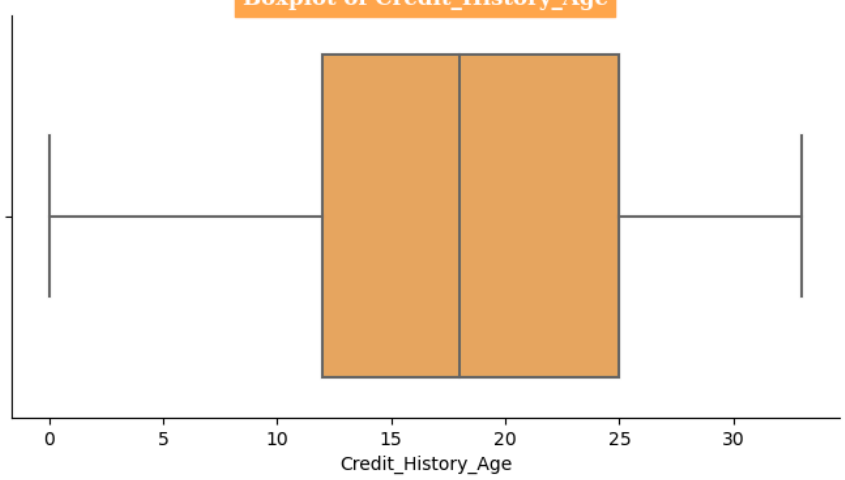


Distplot of Credit_Utilization_Ratio

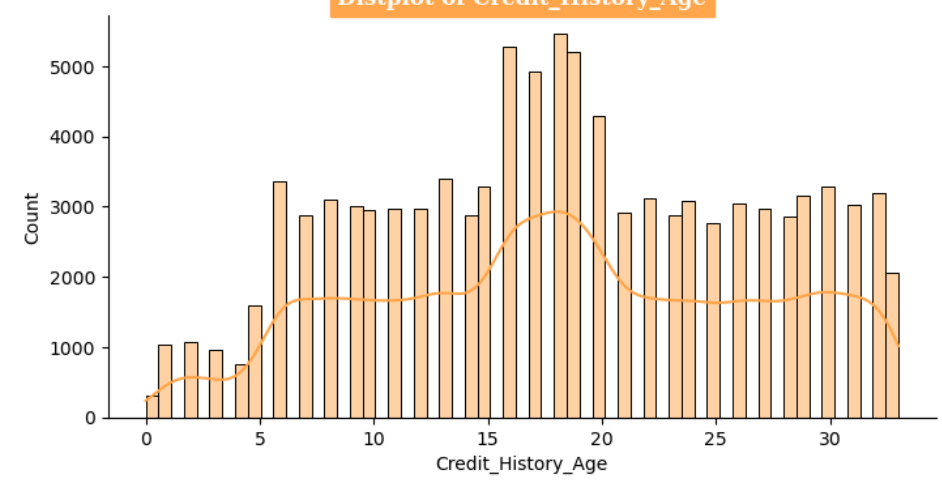


plot of Credit_History_Age

Boxplot of Credit_History_Age

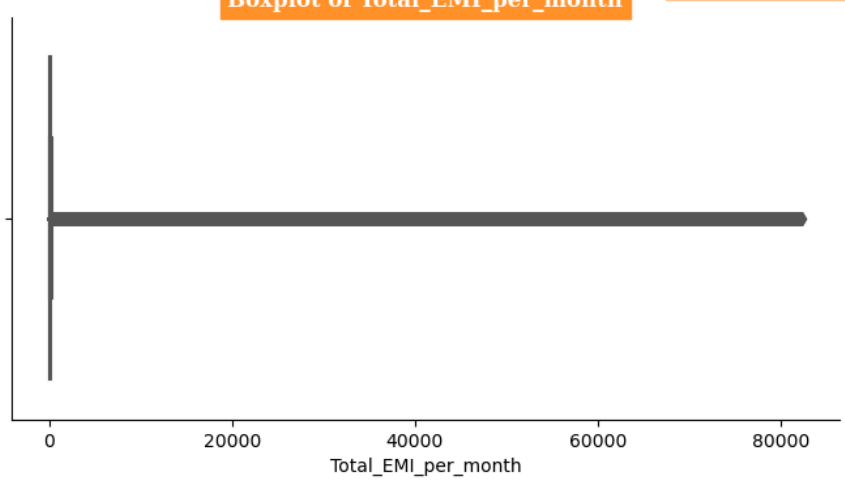


Distplot of Credit_History_Age

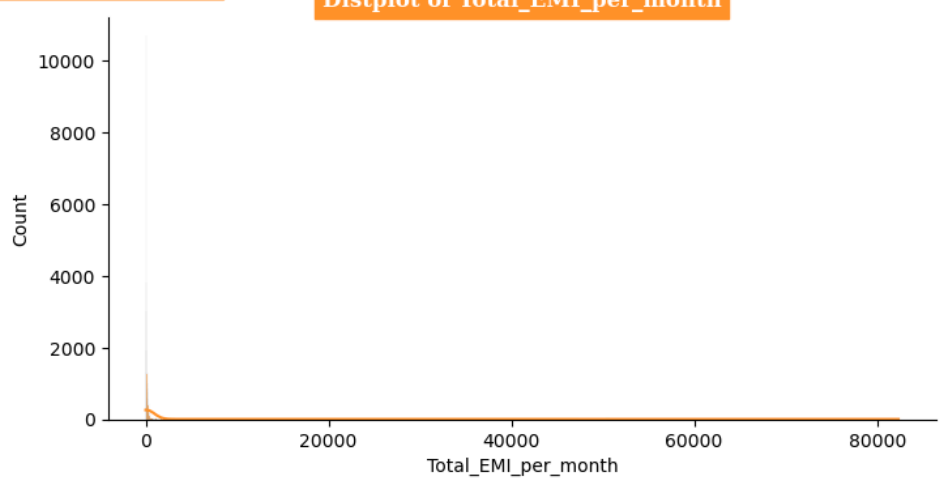


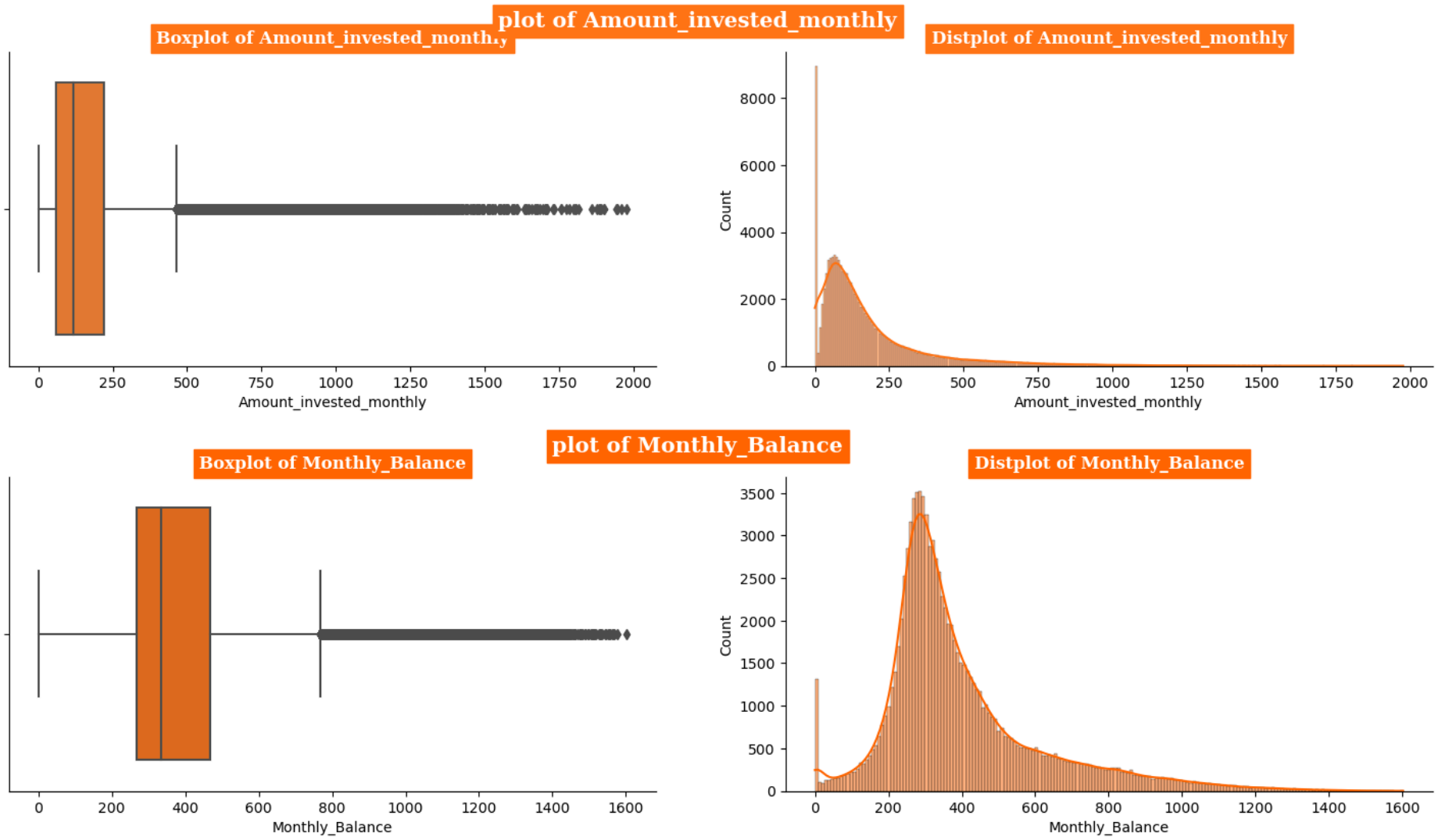
plot of Total_EMI_per_month

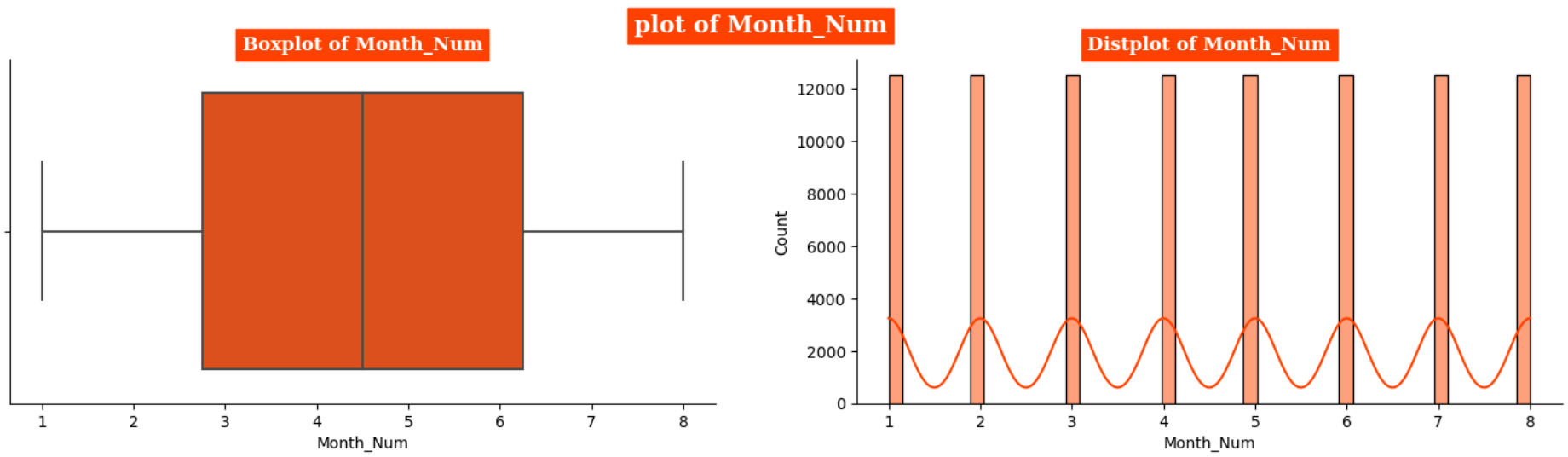
Boxplot of Total_EMI_per_month



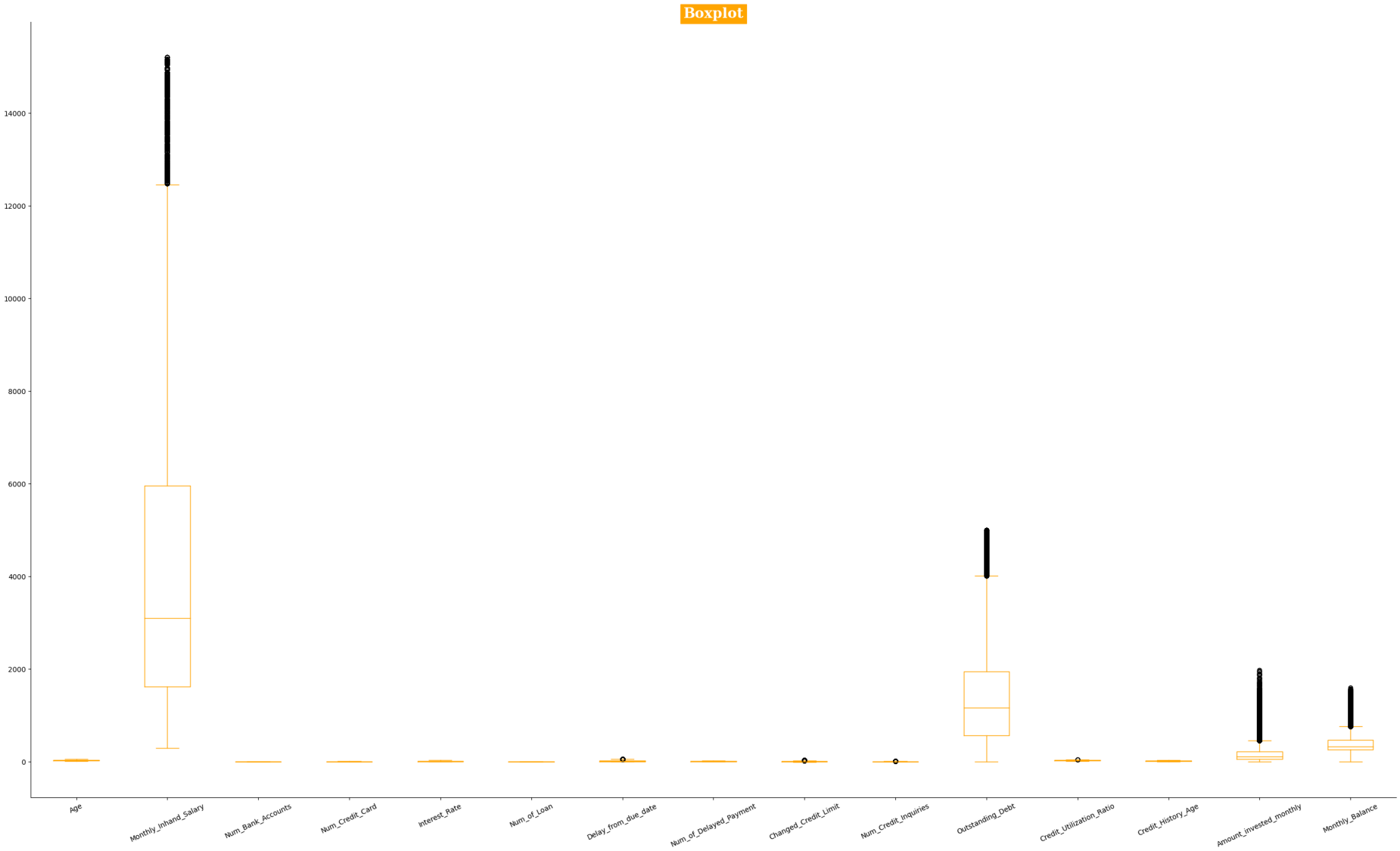
Distplot of Total_EMI_per_month







In [34]:
```
columns_to_plot = ['Age', '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',
                   'Amount_invested_monthly', 'Monthly_Balance']

plt.figure(figsize=(35, 20))
numerical_cols[columns_to_plot].boxplot(rot=25, color='orange')
plt.title(f'Boxplot',fontsize=20,fontfamily='serif',fontweight='bold',backgroundcolor='orange',color='w')
sns.despine()
plt.grid(False)
plt.show()
```



In [20]:

```python
# 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
    'Monthly_Balance': 'mean',

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

Out[20]:

	Customer_ID	ID	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Ir
0	CUS_0x1000	0x1628a	Alistair Barrf	17	913-74-1218	Lawyer	30625.940	2706.161667	6	5	
1	CUS_0x1009	0x66a2	Arunah	26	063-67-6938	Mechanic	52312.680	4250.390000	6	5	
2	CUS_0x100b	0x1ef6	Shirboni	18	238-62-0395	Media_Manager	113781.390	9549.782500	1	4	
3	CUS_0x1011	0x17646	Schneyerh	44	793-05-8223	Doctor	58918.470	5208.872500	3	3	
4	CUS_0x1013	0x243ea	Cameront	44	930-49-9615	Mechanic	98620.980	7962.415000	3	3	
...	...	...	...	...	...	...	...	...	...	...	...
12495	CUS_0xff3	0x344a	Somervilled	55	726-35-5322	Scientist	17032.785	1176.398750	0	6	
12496	CUS_0xff4	0x16aa	Poornimaf	37	655-05-7666	Entrepreneur	25546.260	2415.855000	8	7	
12497	CUS_0xff6	0xfab6	Shieldsb	19	541-92-8371	Doctor	117639.920	9727.326667	5	6	
12498	CUS_0xffc	0x61e6	Brads	17	226-86-7294	Musician	60877.170	5218.097500	6	8	
12499	CUS_0xffd	0x25afa	Damouniq	29	832-88-8320	Scientist	41398.440	3749.870000	8	7	

12500 rows × 25 columns

In [21]:
```
categorical_cols = df_aggregated.select_dtypes(include=['object', 'category'])
```

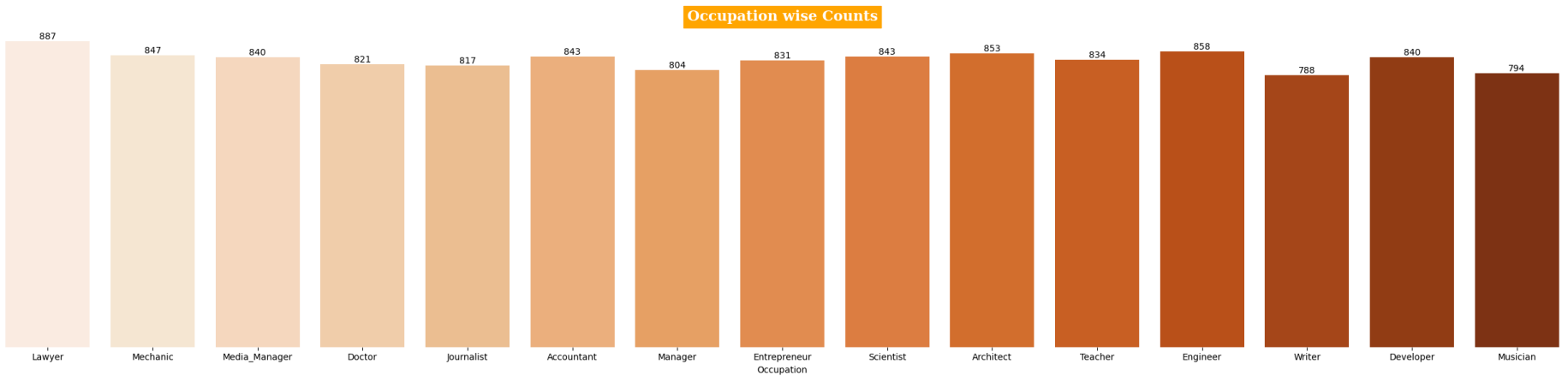
In [12]:
```
categorical_cols
```

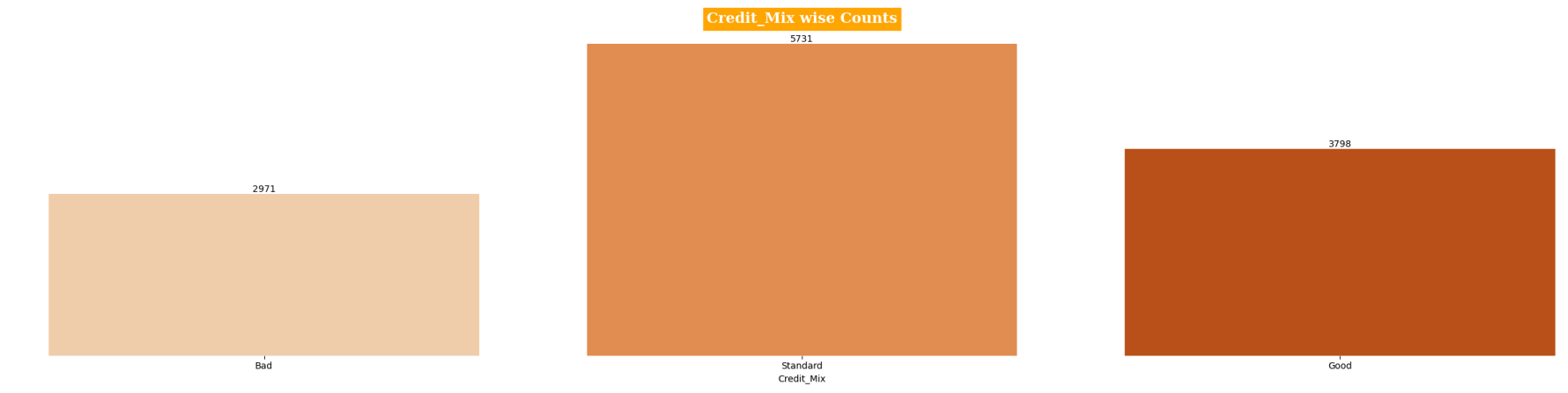
Out[12]:

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

12500 rows × 8 columns

In [15]:
```
selected_cols = ['Occupation', 'Credit_Mix', 'Payment_of_Min_Amount']

for col in selected_cols:
    plt.figure(figsize=(25, 6))
    plt.style.use('default')
    plt.style.use('seaborn-bright')
    a = sns.countplot(data=categorical_cols, x=col, palette='Oranges')
    plt.title(f'{col} wise Counts', fontsize=16, fontfamily='serif', fontweight='bold', backgroundcolor='orange', 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 [22]:
```
# 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))
```

Out[22]:
```
Skewness Coefficient
--------------------
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:

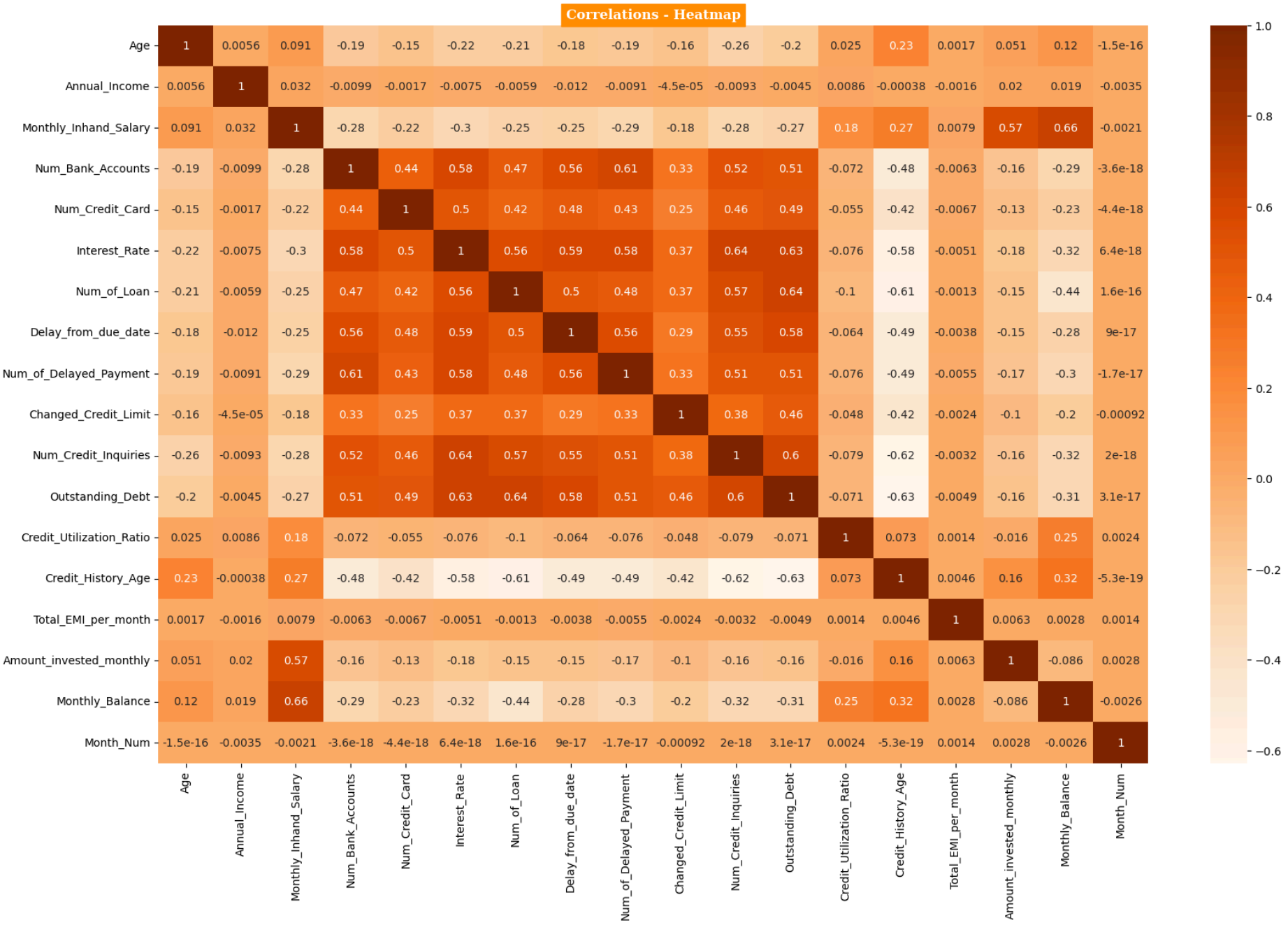
- Variables like **Age** (0.1570), **Num_Bank_Accounts** (-0.1892), and **Credit_History_Age** (-0.0490) exhibit low skewness, meaning their distributions are more symmetrical with values evenly spread around the mean. This suggests a more uniform distribution without significant outliers or tails.

In [21]:
```
numerical_cols.corr()
```

Out[21]:

	Age	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_Loan	De
Age	1.000000e+00	0.005590	0.090794	-1.909258e-01	-1.485908e-01	-2.178565e-01	-2.135980e-01	
Annual_Income	5.589605e-03	1.000000	0.031698	-9.883550e-03	-1.667479e-03	-7.455159e-03	-5.939649e-03	
Monthly_Inhand_Salary	9.079397e-02	0.031698	1.000000	-2.833183e-01	-2.169563e-01	-3.019058e-01	-2.544056e-01	
Num_Bank_Accounts	-1.909258e-01	-0.009884	-0.283318	1.000000e+00	4.423838e-01	5.844557e-01	4.725037e-01	
Num_Credit_Card	-1.485908e-01	-0.001667	-0.216956	4.423838e-01	1.000000e+00	4.979214e-01	4.178346e-01	
Interest_Rate	-2.178565e-01	-0.007455	-0.301906	5.844557e-01	4.979214e-01	1.000000e+00	5.591562e-01	
Num_of_Loan	-2.135980e-01	-0.005940	-0.254406	4.725037e-01	4.178346e-01	5.591562e-01	1.000000e+00	
Delay_from_due_date	-1.750736e-01	-0.011611	-0.251162	5.640608e-01	4.830521e-01	5.929428e-01	5.045482e-01	
Num_of_Delayed_Payment	-1.870159e-01	-0.009144	-0.289875	6.120210e-01	4.300217e-01	5.809739e-01	4.818863e-01	
Changed_Credit_Limit	-1.566733e-01	-0.000045	-0.175135	3.307754e-01	2.528722e-01	3.666907e-01	3.700487e-01	
Num_Credit_Inquiries	-2.566488e-01	-0.009308	-0.280591	5.229506e-01	4.632187e-01	6.385437e-01	5.696246e-01	
Outstanding_Debt	-2.022942e-01	-0.004533	-0.269078	5.070425e-01	4.902662e-01	6.294144e-01	6.387126e-01	
Credit_Utilization_Ratio	2.548171e-02	0.008606	0.176081	-7.186379e-02	-5.535274e-02	-7.569978e-02	-1.004692e-01	
Credit_History_Age	2.342567e-01	-0.000383	0.271058	-4.847938e-01	-4.165627e-01	-5.754552e-01	-6.051535e-01	
Total_EMI_per_month	1.671218e-03	-0.001620	0.007949	-6.336971e-03	-6.683198e-03	-5.089679e-03	-1.266348e-03	
Amount_invested_monthly	5.131321e-02	0.020078	0.568380	-1.640382e-01	-1.280136e-01	-1.767488e-01	-1.490007e-01	
Monthly_Balance	1.163969e-01	0.018747	0.659723	-2.926722e-01	-2.325054e-01	-3.210812e-01	-4.365002e-01	
Month_Num	-1.490159e-16	-0.003516	-0.002109	-3.606326e-18	-4.444433e-18	6.403426e-18	1.550535e-16	

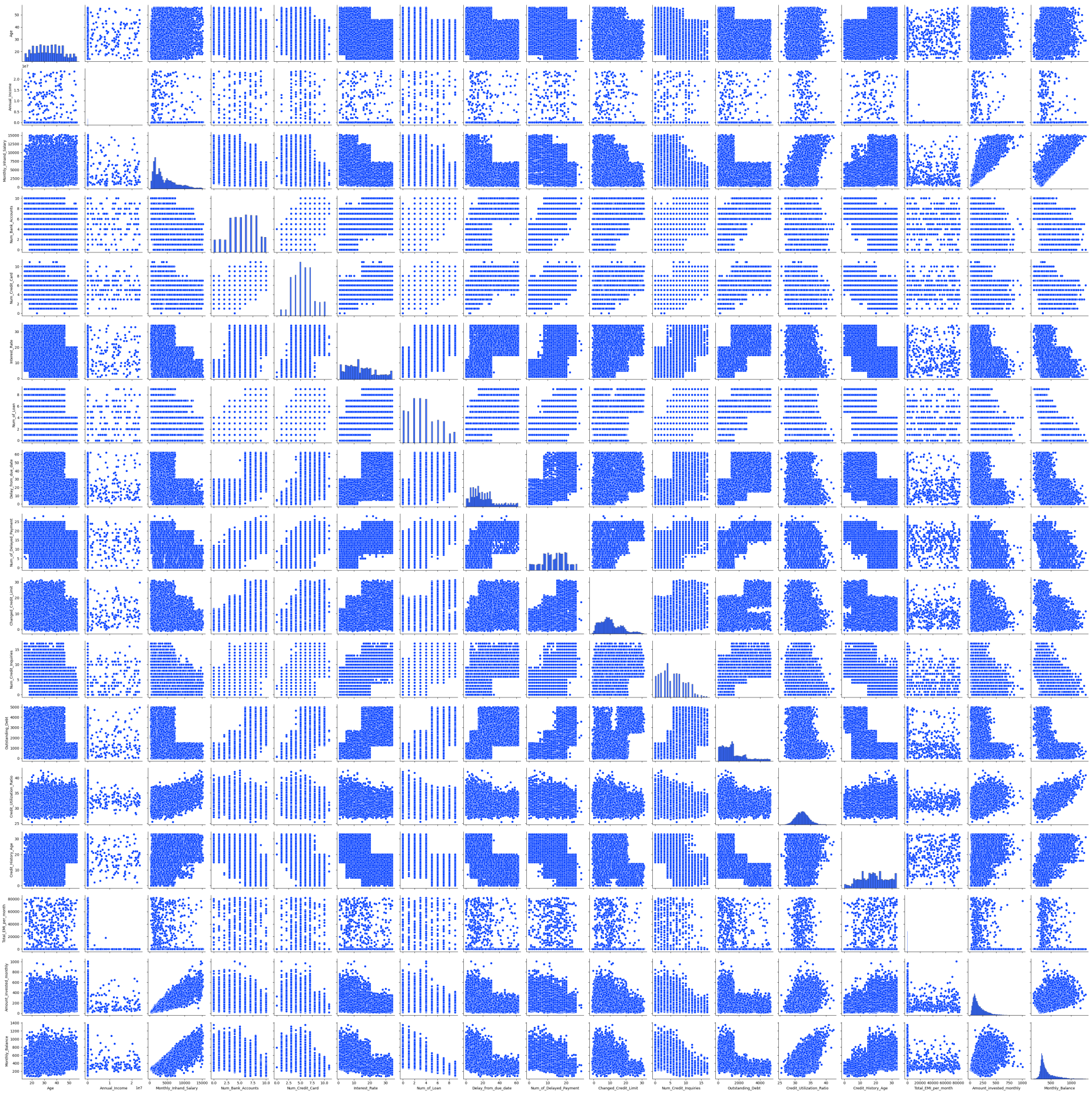
In [18]:
```
plt.figure(figsize=(20,12))
sns.heatmap(numerical_cols.corr(), annot=True, cmap='Oranges')
plt.title('Correlations - Heatmap',fontsize=12,fontfamily='serif',fontweight='bold',backgroundcolor='darkorange',color='w')
plt.show()
```



🕵️‍♂️ Insights

> **Strong Correlation with Monthly Inhand Salary and Investing**:
>
> - **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 and Credit Inquiries**:
>
> - **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 Impact**:
>
> - **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.
>
> **Negative Impact on Credit History Age**:
>
> - **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**:
>
> - **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 [25]:
```
plt.figure(figsize=(18,0.05))
plt.axis('off')
plt.style.use('default')
plt.style.use('seaborn-bright')
plt.title(f' Pairplot Analysis ',fontfamily='serif',fontweight='bold',fontsize=15,backgroundcolor='orange',color='w')
sns.pairplot(data=df_aggregated,palette='Oranges')
plt.show()
```



Feature Engineering

In [166…

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

In [167…

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

Out[167]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Int
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	
5	0x1607	CUS_0xd40	June	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	
6	0x1608	CUS_0xd40	July	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	
7	0x1609	CUS_0xd40	August	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	4	

Feature creation

In [168…
```
dff['Debt_to_Income_Ratio'] = dff['Outstanding_Debt'] / dff['Annual_Income']
dff['Total_Debt'] = dff['Outstanding_Debt'] + dff['Total_EMI_per_month']
```

In [169…
```
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_mapping)
```

🕵️ 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 [170…
```
dff['Credit_Mix'].unique()
```

Out[170]:
```
array(['Good', 'Standard', 'Bad'], dtype=object)
```

In [171…
```
dff['Credit_Mix_num'] = dff['Credit_Mix'].map({'Good':2 , 'Standard':1, 'Bad':0})
```

In [172…
```
dff['Payment_of_Min_Amount'].unique()
```

Out[172]:
```
array(['No', 'Yes'], dtype=object)
```

In [173…
```
dff['Payment_of_Min_Amount'] = dff['Payment_of_Min_Amount'].map({'Yes':1, 'No':0})
```

In [174…
```
dff['Payment_of_Min_Amount'].isna().sum() , dff['Payment_of_Min_Amount'].unique()
```

Out[174]:
```
(0, array([0, 1], dtype=int64))
```

In [175…
```
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']>dff['Monthly_Balance'],1,0)
```

In [176…
```
dff = dff.drop(columns=['ID','Type_of_Loan','Credit_Mix','Payment_Behaviour'],axis=1)
```

In [177…
```
dff.tail(8)
```

Out[177]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_R
99992	CUS_0x942c	January	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	
99993	CUS_0x942c	February	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	
99994	CUS_0x942c	March	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	
99995	CUS_0x942c	April	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	
99996	CUS_0x942c	May	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	
99997	CUS_0x942c	June	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	
99998	CUS_0x942c	July	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	
99999	CUS_0x942c	August	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	6	

Hypothetical Credit_Score Computation

⚖️ Scaling

In [178…
```
scaling_features = dff.iloc[:, [3] + list(range(6, dff.shape[1]))].columns
scaling_features
```

Out[178]:
```
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'],
      dtype='object')
```

In [179…
```
scaler = MinMaxScaler()
dff[scaling_features] = scaler.fit_transform(dff[scaling_features])
```

In [180…
```
dff
```

Out[180]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Int
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
...	...	...	...	...	...	...	...	...	...	...	...
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	

100000 rows × 30 columns

In [181…
```
dff.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 30 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
dtypes: float64(25), object(5)
memory usage: 22.9+ MB
```

In [184…
```
# 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,
}
```

Reasons for feature selection and its weightage -- Credit Score Computation

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

In [185…
```
dff['Credit_Score'] = dff.apply(lambda x: sum(x[feature] * weight for feature, weight in weights.items()), axis=1)
```

In [186…
```
dff['Credit_Score'] = dff['Credit_Score'] * (850 - 300) + 300
```

In [187…
```
dff[(dff['Credit_Score']<300) | (dff['Credit_Score']>850)]
```

Out[187]:

Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Nu

In [188…
```
cs_df = dff[['Customer_ID','Name','SSN','Month','Credit_Score']]
```

🖇 Aggregated data - Consolidated

In [189…
```
cdf = cs_df.groupby(['Customer_ID','Name','SSN'])['Credit_Score'].mean().to_frame().round().astype(int).reset_index()
cdf.sample(10)
```

Out[189]:

	Customer_ID	Name	SSN	Credit_Score
2235	CUS_0x333f	rac	777-65-0580	501
7321	CUS_0x7dcd	Peter Hendersonr	812-70-4107	500
5615	CUS_0x6473	Lisa Juccaw	116-72-1150	484
5504	CUS_0x62e4	Nigel Daviesy	336-90-0523	497
11676	CUS_0xbdf7	Nhlapok	698-60-8051	477
6128	CUS_0x6bad	Leika Kiharac	415-46-9238	490
7485	CUS_0x804	Dhanya Skariachanx	857-32-0525	479
373	CUS_0x1612	en Klaymanf	147-97-2500	522
1098	CUS_0x2224	Andreasi	855-02-9559	496
6209	CUS_0x6ced	Bernie Woodalln	563-01-6338	477

In [190…
```
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, labels=bin_labels, right=True)
```

In [191…
```
dff.head(8)
```

Out[191]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
5	CUS_0xd40	June	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
6	CUS_0xd40	July	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
7	CUS_0xd40	August	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0

In [192…
```
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, labels=bin_labels, right=True)
```

In [193…
```
cdf.sample(10)
```

Out[193]:

	Customer_ID	Name	SSN	Credit_Score	overall_Credit_Score_category
11851	CUS_0xc08b	Poornimax	464-69-1352	496	very Bad
4677	CUS_0x56d1	Bena	203-76-0108	524	very Bad
11156	CUS_0xb66e	Prodhand	450-16-3479	495	very Bad
5459	CUS_0x624b	Bernieo	847-17-3721	496	very Bad
133	CUS_0x1249	Soyoung Kimm	851-75-4209	401	very Bad
8668	CUS_0x919a	Kiharar	955-69-8416	512	very Bad
2595	CUS_0x38cf	Mattx	151-48-0792	520	Poor
3483	CUS_0x45d2	Charlesh	548-90-9826	509	Poor
10546	CUS_0xad9d	Valentina Zaf	771-56-8439	522	Poor
10502	CUS_0xacec	Laurentn	661-67-1317	512	very Bad

In [194…
```
cdf.groupby('overall_Credit_Score_category')['Customer_ID'].nunique().to_frame()
```

Out[194]:

overall_Credit_Score_category	Customer_ID
very Bad	7412
Poor	5076
Fair	12
Good	0
Excellent	0

In [195…
```
cdf.overall_Credit_Score_category.value_counts()
```

Out[195]:
```
very Bad     7412
Poor         5076
Fair           12
Good            0
Excellent       0
Name: overall_Credit_Score_category, dtype: int64
```

In [196…
```
cdf.shape
```

Out[196]:
```
(12500, 5)
```

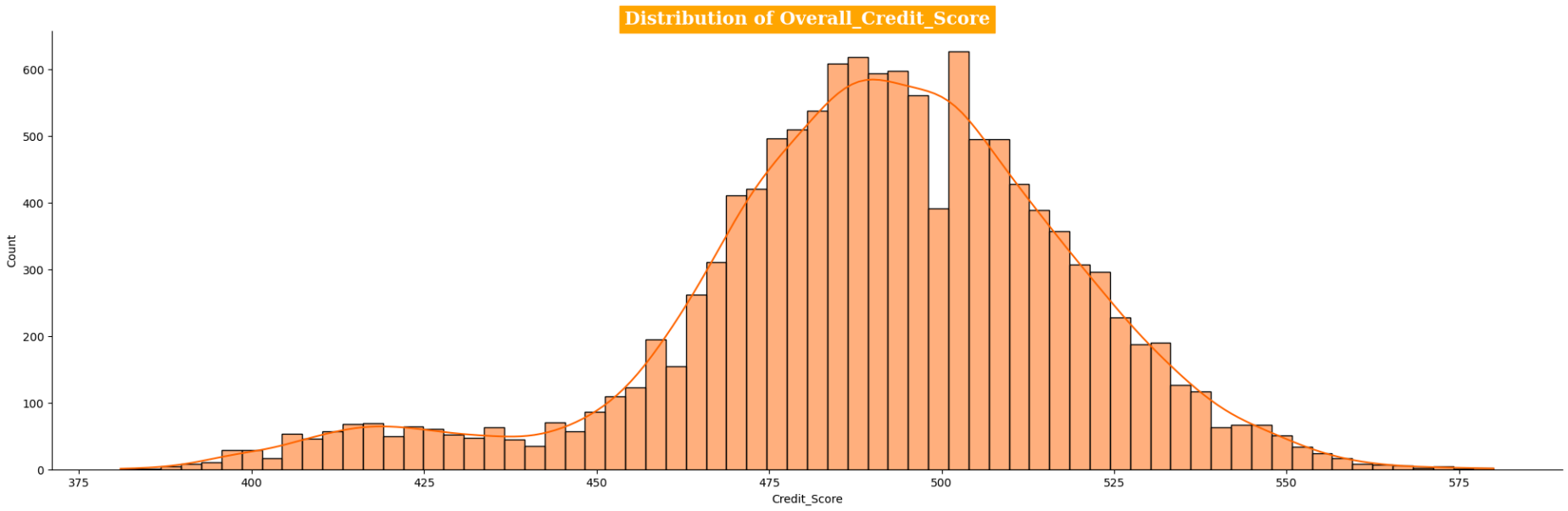
In [197…
```
cdf
```

Out[197]:

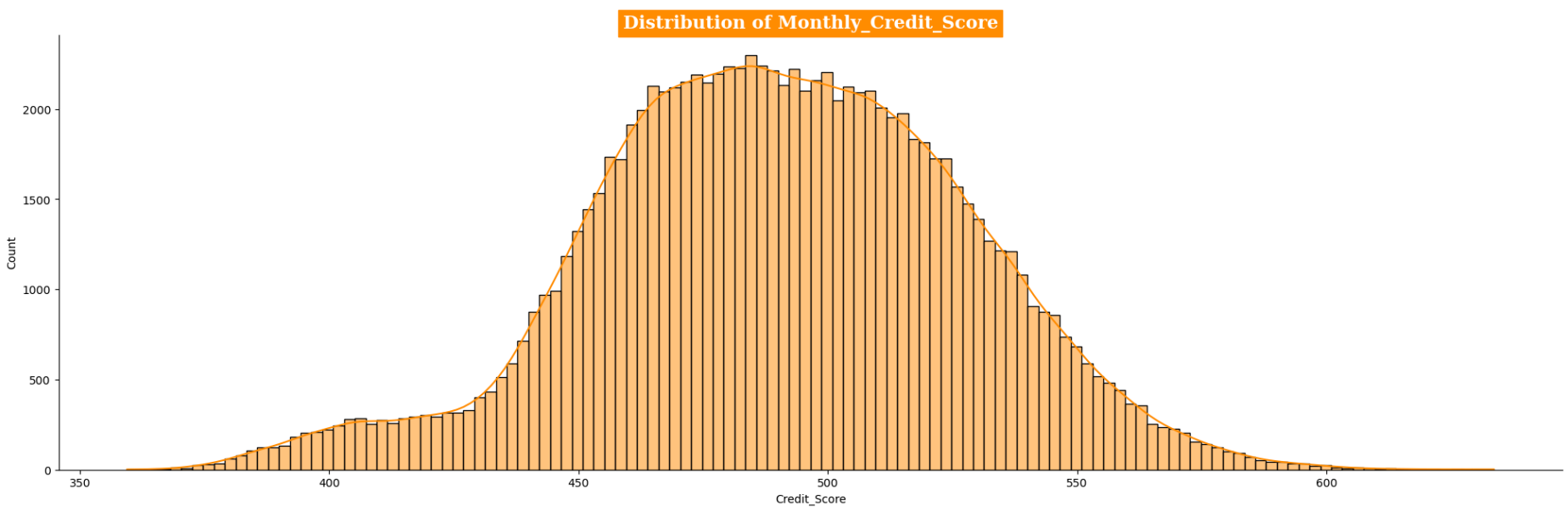
	Customer_ID	Name	SSN	Credit_Score	overall_Credit_Score_category
0	CUS_0x1000	Alistair Barrf	913-74-1218	481	very Bad
1	CUS_0x1009	Arunah	063-67-6938	513	very Bad
2	CUS_0x100b	Shirboni	238-62-0395	502	very Bad
3	CUS_0x1011	Schneyerh	793-05-8223	464	very Bad
4	CUS_0x1013	Cameront	930-49-9615	510	Poor
...	...	...	...	...	...
12495	CUS_0xff3	Somervilled	726-35-5322	479	very Bad
12496	CUS_0xff4	Poornimaf	655-05-7666	484	very Bad
12497	CUS_0xff6	Shieldsb	541-92-8371	512	very Bad
12498	CUS_0xffc	Brads	226-86-7294	493	Poor
12499	CUS_0xffd	Damouniq	832-88-8320	498	very Bad

12500 rows × 5 columns

In [198…
```
plt.figure(figsize=(24,7))
plt.title('Distribution of Overall_Credit_Score',fontsize=16,fontfamily='serif',fontweight='bold',backgroundcolor='orange',color=
sns.histplot(cdf['Credit_Score'],color=cp[6],kde=True)
sns.despine()
plt.show()
```



In [199…
```
plt.figure(figsize=(24,7))
plt.title('Distribution of Monthly_Credit_Score',fontsize=16,fontfamily='serif',fontweight='bold',backgroundcolor='darkorange',co
sns.histplot(dff['Credit_Score'],color=cp[8],kde=True)
sns.despine()
plt.show()
```



In [200…
```
#cdf.to_csv('credit_scored_data.csv',index=False)
```

In [201…
```
Fair_customers = cdf[cdf['overall_Credit_Score_category'] == 'Fair']
Fair_customers.describe().T
```

Out[201]:

	count	mean	std	min	25%	50%	75%	max
Credit_Score	12.0	492.25	21.166547	462.0	473.0	495.0	510.25	520.0

RFM Integration

In [202…
```
dff['Recency'] = dff.groupby('Customer_ID')['Month_Num'].transform(lambda x: (x.max() - 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 [203…
```
dff
```

Out[203]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Int
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	
...	...	...	...	...	...	...	...	...	...	...	...
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	

100000 rows × 35 columns

In [204…
```
dff['Credit_Score_after_RFM'] = dff['Credit_Score'] + (dff['Recency'] * 0.1 + dff['Frequency'] * 0.1 + dff['Monetary'] * 0.1)
```

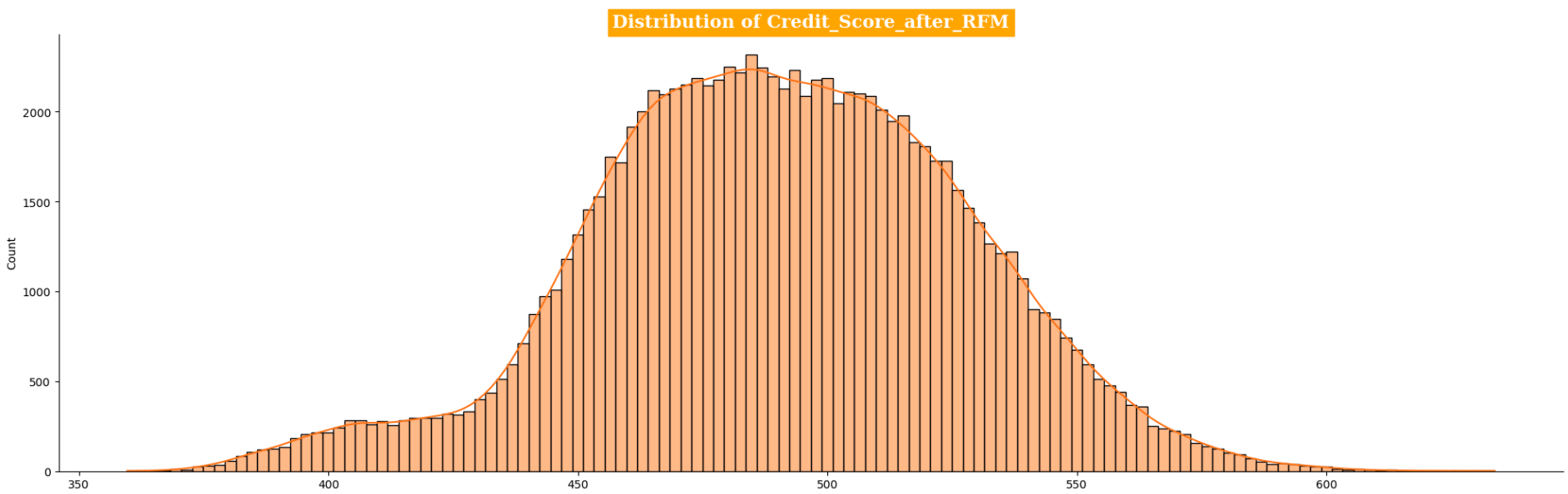
In [205…
```
dff.tail(8)
```

Out[205]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Inter
99992	CUS_0x942c	January	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99993	CUS_0x942c	February	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99994	CUS_0x942c	March	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	

In [206…
```
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=bins, labels=bin_labels, right=True)
```

In [207…
```
plt.figure(figsize=(24,7))
plt.title('Distribution of Credit_Score_after_RFM',fontsize=16,fontfamily='serif',fontweight='bold',backgroundcolor='orange',colo
sns.histplot(dff['Credit_Score_after_RFM'],color=cp[5],kde=True)
sns.despine()
plt.show()
```



In [208…
```
dff.head(8)
```

Out[208]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest
0	CUS_0xd40	January	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
1	CUS_0xd40	February	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
2	CUS_0xd40	March	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
3	CUS_0xd40	April	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
4	CUS_0xd40	May	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
5	CUS_0xd40	June	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
6	CUS_0xd40	July	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0
7	CUS_0xd40	August	Aaron Maashoh	0.214286	821-00-0265	Scientist	0.000501	0.102087	0.3	0.363636	0.0

In [219…
```
dff.groupby('RFM_Credit_Score_category')['Customer_ID'].nunique().to_frame()
```

Out[219]:

RFM_Credit_Score_category	Customer_ID
very Bad	12128
Poor	10304
Fair	67
Good	0
Excellent	0

In [221…
```
cdff = dff.groupby('Customer_ID')['Credit_Score_after_RFM'].mean().to_frame().reset_index()
cdff
```

Out[221]:

	Customer_ID	Credit_Score_after_RFM
0	CUS_0x1000	481.179976
1	CUS_0x1009	512.771142
2	CUS_0x100b	502.041550
3	CUS_0x1011	463.626393
4	CUS_0x1013	510.362284
...	...	...
12495	CUS_0xff3	479.397812
12496	CUS_0xff4	484.491162
12497	CUS_0xff6	512.145579
12498	CUS_0xffc	493.413142
12499	CUS_0xffd	498.267247

12500 rows × 2 columns

In [227…
```
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_RFM'], bins=bins, labels=bin_labels, right=True)
```

In [228…
```
cdff.sample(6)
```

Out[228]:

	Customer_ID	Credit_Score_after_RFM	cumulative_RFM_Credit_Score_category
7319	CUS_0x7dc5	559.402651	Poor
3794	CUS_0x4a3d	485.454920	very Bad
9484	CUS_0x9dec	485.602284	very Bad
7091	CUS_0x7a4d	508.281972	Poor
11700	CUS_0xbe4d	474.507499	very Bad
6391	CUS_0x6fa1	458.118087	very Bad

In [229…
```
cdff.groupby('cumulative_RFM_Credit_Score_category')['Customer_ID'].nunique().to_frame()
```

Out[229]:

cumulative_RFM_Credit_Score_category	Customer_ID
very Bad	7788
Poor	4712
Fair	0
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 [230…
```
dff.tail(8)
```

Out[230]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Inter
99992	CUS_0x942c	January	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(
99993	CUS_0x942c	February	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(
99994	CUS_0x942c	March	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(
99995	CUS_0x942c	April	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(
99996	CUS_0x942c	May	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	(

In [235…
```
months = ['June', 'July', 'August']

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

In [236…
```
filtered_dff.shape
```

Out[236]:
```
(37500, 37)
```

In [237…
```
filtered_dff.tail(6)
```

Out[237]:

	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Inte
99989	CUS_0x8600	June	Sarah McBridec	0.333333	031-35-0942	Architect	0.000537	0.109138	1.0	0.727273	
99990	CUS_0x8600	July	Sarah McBridec	0.333333	031-35-0942	Architect	0.000537	0.109138	1.0	0.727273	
99991	CUS_0x8600	August	Sarah McBridec	0.333333	031-35-0942	Architect	0.000537	0.109138	1.0	0.727273	
99997	CUS_0x942c	June	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99998	CUS_0x942c	July	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	
99999	CUS_0x942c	August	Nicks	0.261905	078-73-5990	Mechanic	0.001349	0.205072	0.4	0.545455	

In [239…
```
fdf = filtered_dff[['Customer_ID','Name','Credit_Score','Monthly_Credit_Score_category','Credit_Score_after_RFM','RFM_Credit_Scor
fdf
```

Out[239]:

	Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Score_after_RFM	RFM_Credit_Score_category
5	CUS_0xd40	Aaron Maashoh	507.392893	Poor	507.483786	Poor
6	CUS_0xd40	Aaron Maashoh	451.438166	very Bad	451.514773	very Bad
7	CUS_0xd40	Aaron Maashoh	501.896482	Poor	501.958804	Poor
13	CUS_0x21b1	Rick Rothackerj	535.669865	Poor	535.739199	Poor
14	CUS_0x21b1	Rick Rothackerj	519.975391	Poor	520.030439	Poor
...	...	...	...	...	...	...
99990	CUS_0x8600	Sarah McBridec	466.012326	very Bad	466.099276	very Bad
99991	CUS_0x8600	Sarah McBridec	524.772751	Poor	524.845415	Poor
99997	CUS_0x942c	Nicks	564.554305	Poor	564.630538	Poor
99998	CUS_0x942c	Nicks	510.803866	Poor	510.865814	Poor
99999	CUS_0x942c	Nicks	514.377974	Poor	514.425637	Poor

37500 rows × 6 columns

In [240…
```
fdf['diff'] = fdf['Credit_Score_after_RFM'] - fdf['Credit_Score']
```

In [242…
```
fdf
```

Out[242]:

	Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Score_after_RFM	RFM_Credit_Score_category	diff
5	CUS_0xd40	Aaron Maashoh	507.392893	Poor	507.483786	Poor	0.090893
6	CUS_0xd40	Aaron Maashoh	451.438166	very Bad	451.514773	very Bad	0.076607
7	CUS_0xd40	Aaron Maashoh	501.896482	Poor	501.958804	Poor	0.062321
13	CUS_0x21b1	Rick Rothackerj	535.669865	Poor	535.739199	Poor	0.069334
14	CUS_0x21b1	Rick Rothackerj	519.975391	Poor	520.030439	Poor	0.055048
...	...	...	...	...	...	...	...
99990	CUS_0x8600	Sarah McBridec	466.012326	very Bad	466.099276	very Bad	0.086950
99991	CUS_0x8600	Sarah McBridec	524.772751	Poor	524.845415	Poor	0.072664
99997	CUS_0x942c	Nicks	564.554305	Poor	564.630538	Poor	0.076234
99998	CUS_0x942c	Nicks	510.803866	Poor	510.865814	Poor	0.061948
99999	CUS_0x942c	Nicks	514.377974	Poor	514.425637	Poor	0.047662

37500 rows × 7 columns

In [246…
```
fdf[(fdf['diff']>1) | (fdf['diff']<0)]
```

Out[246]:

Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Score_after_RFM	RFM_Credit_Score_category	diff

✍️ ***Observation:***

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

In [250…
```
fdf['Last_3_months_Credit_Score_consolidated'] = fdf.groupby(['Customer_ID', 'Name'])['Credit_Score'].transform('mean')
```

In [251…
```
fdf
```

Out[251]:

	Customer_ID	Name	Credit_Score	Monthly_Credit_Score_category	Credit_Score_after_RFM	RFM_Credit_Score_category	diff	Last_3_month
5	CUS_0xd40	Aaron Maashoh	507.392893	Poor	507.483786	Poor	0.090893	
6	CUS_0xd40	Aaron Maashoh	451.438166	very Bad	451.514773	very Bad	0.076607	
7	CUS_0xd40	Aaron Maashoh	501.896482	Poor	501.958804	Poor	0.062321	
13	CUS_0x21b1	Rick Rothackerj	535.669865	Poor	535.739199	Poor	0.069334	
14	CUS_0x21b1	Rick Rothackerj	519.975391	Poor	520.030439	Poor	0.055048	
...	...	...	...	...	...	...	...	
99990	CUS_0x8600	Sarah McBridec	466.012326	very Bad	466.099276	very Bad	0.086950	
99991	CUS_0x8600	Sarah McBridec	524.772751	Poor	524.845415	Poor	0.072664	
99997	CUS_0x942c	Nicks	564.554305	Poor	564.630538	Poor	0.076234	
99998	CUS_0x942c	Nicks	510.803866	Poor	510.865814	Poor	0.061948	
99999	CUS_0x942c	Nicks	514.377974	Poor	514.425637	Poor	0.047662	

37500 rows × 8 columns

In [252…
```
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_Score_consolidated'], bins=bins, labels=bin_labels,
```

In [253…
```
l3m = fdf[['Customer_ID','Name','Last_3_months_Credit_Score_consolidated','last_3_months_Credit_Score_category']]
```

In [257…
```
l3m_unique = l3m.drop_duplicates(subset=['Customer_ID', 'Name'], keep='first')
```

In [258…
```
l3m_unique.sample(10)
```

Out[258]:

	Customer_ID	Name	Last_3_months_Credit_Score_consolidated	last_3_months_Credit_Score_category
39581	CUS_0x63d	Olivia Orani	473.768465	very Bad
32461	CUS_0x6ae7	Valetkevitchp	490.129574	very Bad
44901	CUS_0x77ca	Ricko	510.134388	Poor
89781	CUS_0x3229	Poornima Guptaz	459.817772	very Bad
38757	CUS_0x2283	Bartr	449.149124	very Bad
65933	CUS_0x585b	Bakerm	460.042341	very Bad
70549	CUS_0x8440	Reidu	509.073447	Poor
31789	CUS_0x1fe3	Baker Alinau	470.327517	very Bad
90269	CUS_0xd71	Andrea Shalal-Esaw	535.896700	Poor
9013	CUS_0x931c	Hubbardy	481.468630	very Bad

In [260…
```
l3m_unique.groupby('last_3_months_Credit_Score_category')['Customer_ID'].nunique().to_frame()
```

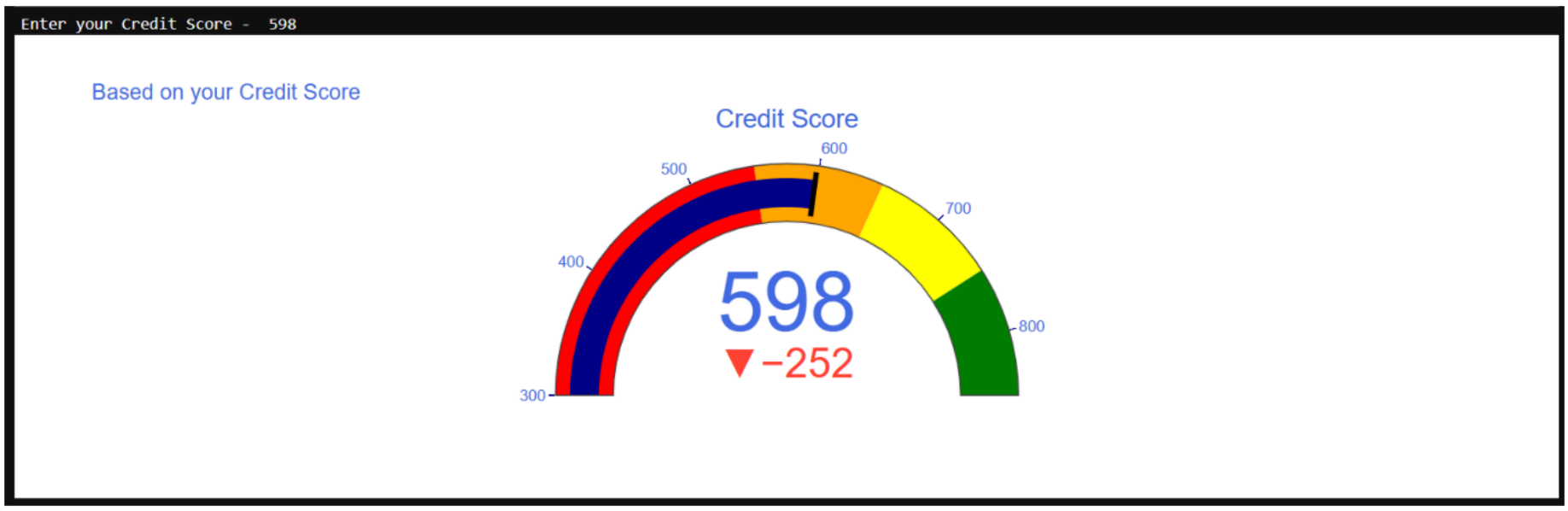
Out[260]:

last_3_months_Credit_Score_category	Customer_ID
very Bad	7761
Poor	4739
Fair	0
Good	0
Excellent	0

In [269…
```
import plotly.graph_objects as go

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': "royalblue", 'family': "Arial"},
)

fig.show()
```

In [2]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

plt.figure(figsize=(18,6))
# Path to your image file
img_path = 'above screenshot.png'
# Load and display the image
img = mpimg.imread(img_path)
plt.imshow(img)
plt.axis('off')  # Hide axis
plt.show()
```



🧐 ✍🏻 **Insights on Analysis**

> **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.

Analysed by : KASI
