In [63]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
```

In [64]:
```
import warnings
warnings.simplefilter('ignore')
```

In [65]:
```
df=pd.read_csv('Credit_score.csv')
```

PROBLEM STATEMENT:We are conducting this case study to formulate a credit score based on the parametres given in the data set

In [66]:
```
df.head()
```

Out[66]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthl
0	0x1602	CUS_0xd40	January	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
1	0x1603	CUS_0xd40	February	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	-500	821-00-0265	Scientist	19114.12	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	
4	0x1606	CUS_0xd40	May	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	

5 rows × 27 columns

In [67]:
```
df.shape
```

Out[67]:
```
(100000, 27)
```

There are 1 lakhs rows and 27 coloumns in this table

In [68]:
```
df.describe()
```

Out[68]:

	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Delay_
count	84998.000000	100000.000000	100000.00000	100000.000000	
mean	4194.170850	17.091280	22.47443	72.466040	
std	3183.686167	117.404834	129.05741	466.422621	
min	303.645417	-1.000000	0.00000	1.000000	
25%	1625.568229	3.000000	4.00000	8.000000	
50%	3093.745000	6.000000	5.00000	13.000000	
75%	5957.448333	7.000000	7.00000	20.000000	
max	15204.633330	1798.000000	1499.00000	5797.000000	

In [69]:
```
df.columns
```

Out[69]:
```
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_Li
mit',
       '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'],
      dtype='object')
```

Column 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_Loans--Represents the types of loan taken by a person Delay_from_due_date--Represents the average number of days delayed from the payment date Num_of_Delayed_Payment--Represents the average number of payments delayed by a person Changed_Credit_Limit--Represents the percentage change in credit card limit Num_Credit_Inquiries--Represents the number of credit card inquiries Credit_Mix--Represents the classification of the mix of credits Outstanding_Debt--Represents the remaining debt to be paid (in USD) Credit_Utilization_Ratio--Represents the utilization ratio of credit card Credit_History_Age--Represents the age of credit history of the person Payment_of_Min_Amount--Represents whether only the minimum amount was paid by the person Total_EMI_per_month--Represents the monthly EMI payments (in USD) Amount_invested_monthly--Represents the monthly amount invested by the customer (in USD) Payment_Behaviour--Represents the payment behavior of the customer (in USD) Monthly_Balance--Represents the monthly balance amount of the customer (in USD)

In [70]:
```
df.dtypes
```

Out[70]:
```
ID                          object
Customer_ID                 object
Month                       object
Name                        object
Age                         object
SSN                         object
Occupation                  object
Annual_Income               object
Monthly_Inhand_Salary      float64
Num_Bank_Accounts            int64
Num_Credit_Card              int64
Interest_Rate                int64
Num_of_Loan                 object
Type_of_Loan                object
Delay_from_due_date          int64
Num_of_Delayed_Payment      object
Changed_Credit_Limit        object
Num_Credit_Inquiries       float64
Credit_Mix                  object
```

Since many of the columns which are supposed to be integer or float type we are converting them using forced conversions as mentioned below.

In [71]:
```
col_list=[ 'Age', 'Annual_Income', 'Num_of_Loan', 'Delay_from_due_date',
       'Num_of_Delayed_Payment', 'Changed_Credit_Limit','Outstanding_Debt',
        'Amount_invested_monthly', 'Monthly_Balance']

for col in col_list:
    df[col] = pd.to_numeric(df[col], errors='coerce')


    df[col] = df[col].astype('float64')
```

In [72]:
```
df.dtypes
```

Out[72]:
```
ID                           object
Customer_ID                  object
Month                        object
Name                         object
Age                         float64
SSN                          object
Occupation                   object
Annual_Income               float64
Monthly_Inhand_Salary       float64
Num_Bank_Accounts             int64
Num_Credit_Card               int64
Interest_Rate                 int64
Num_of_Loan                 float64
Type_of_Loan                 object
Delay_from_due_date         float64
Num_of_Delayed_Payment      float64
Changed_Credit_Limit        float64
Num_Credit_Inquiries        float64
Credit_Mix                   object
Outstanding_Debt            float64
Credit_Utilization_Ratio    float64
Credit_History_Age           object
Payment_of_Min_Amount        object
Total_EMI_per_month         float64
Amount_invested_monthly     float64
Payment_Behaviour            object
Monthly_Balance             float64
dtype: object
```

In [73]:
```
df.drop(columns=['SSN','ID'],axis=1,inplace=True)
```

In [74]:
```
df.head()
```

Out[74]:

	Customer_ID	Month	Name	Age	Occupation	Annual_Income	Monthly_Inhand_Sal
0	CUS_0xd40	January	Aaron Maashoh	23.0	Scientist	19114.12	1824.843
1	CUS_0xd40	February	Aaron Maashoh	23.0	Scientist	19114.12	N
2	CUS_0xd40	March	Aaron Maashoh	-500.0	Scientist	19114.12	N
3	CUS_0xd40	April	Aaron Maashoh	23.0	Scientist	19114.12	N
4	CUS_0xd40	May	Aaron Maashoh	23.0	Scientist	19114.12	1824.843

5 rows × 25 columns

DATA CLEANING:Cleaning the coloumns 'Type_of_Loan' and 'Payment_Behaviour'

In [75]:
```
df['Type_of_Loan'].head(20)
```

Out[75]:
```
0     Auto Loan, Credit-Builder Loan, Personal Loan,...
1     Auto Loan, Credit-Builder Loan, Personal Loan,...
2     Auto Loan, Credit-Builder Loan, Personal Loan,...
3     Auto Loan, Credit-Builder Loan, Personal Loan,...
4     Auto Loan, Credit-Builder Loan, Personal Loan,...
5     Auto Loan, Credit-Builder Loan, Personal Loan,...
6     Auto Loan, Credit-Builder Loan, Personal Loan,...
7     Auto Loan, Credit-Builder Loan, Personal Loan,...
8                                   Credit-Builder Loan
9                                   Credit-Builder Loan
10                                  Credit-Builder Loan
11                                  Credit-Builder Loan
12                                  Credit-Builder Loan
13                                  Credit-Builder Loan
14                                  Credit-Builder Loan
15                                  Credit-Builder Loan
16              Auto Loan, Auto Loan, and Not Specified
17              Auto Loan, Auto Loan, and Not Specified
18              Auto Loan, Auto Loan, and Not Specified
19              Auto Loan, Auto Loan, and Not Specified
Name: Type_of_Loan, dtype: object
```

In [76]:
```
df['Type_of_Loan'] = df['Type_of_Loan'].astype(str)
def clean_loans(loans):
    loan_list=loans.replace('and',',').split(',')
    cleaned_loans = [loan.strip() for loan in loan_list if loan.strip()]
    unique_loans=set(cleaned_loans)
    unique_loans.discard('Not Specified')
    return ','.join(unique_loans)
df['Type_of_Loan']=df['Type_of_Loan'].apply(clean_loans)
```

In [77]:
```
df['Type_of_Loan']
```

Out[77]:
```
0        Credit-Builder Loan,Auto Loan,Home Equity Loan...
1        Credit-Builder Loan,Auto Loan,Home Equity Loan...
2        Credit-Builder Loan,Auto Loan,Home Equity Loan...
3        Credit-Builder Loan,Auto Loan,Home Equity Loan...
4        Credit-Builder Loan,Auto Loan,Home Equity Loan...
                               ...
99995                             Auto Loan,Student Loan
99996                             Auto Loan,Student Loan
99997                             Auto Loan,Student Loan
99998                             Auto Loan,Student Loan
99999                             Auto Loan,Student Loan
Name: Type_of_Loan, Length: 100000, dtype: object
```

In [78]:
```
df['Payment_Behaviour']
```

Out[78]:
```
0         High_spent_Small_value_payments
1          Low_spent_Large_value_payments
2         Low_spent_Medium_value_payments
3          Low_spent_Small_value_payments
4        High_spent_Medium_value_payments
                       ...
99995     High_spent_Large_value_payments
99996    High_spent_Medium_value_payments
99997     High_spent_Large_value_payments
99998      Low_spent_Large_value_payments
99999                              !@9#%8
Name: Payment_Behaviour, Length: 100000, dtype: object
```

In [79]:
```
import re  ##use regex to filter the words for spent and value
def clean_payment_type(payment):
    if isinstance(payment, str):

        cleaned_payment = re.sub(r'[^a-zA-Z0-9_ ]', '', payment)

        cleaned_payment = cleaned_payment.strip()

        if not cleaned_payment:
            return 'Invalid'
        return cleaned_payment
    else:

        return 'Invalid'
df['Payment_Behaviour']=df['Payment_Behaviour'].apply(clean_payment_type)
```

In [80]:
```
df['Payment_Behaviour']
```

Out[80]:
```
0         High_spent_Small_value_payments
1          Low_spent_Large_value_payments
2         Low_spent_Medium_value_payments
3          Low_spent_Small_value_payments
4        High_spent_Medium_value_payments
                       ...
99995     High_spent_Large_value_payments
99996    High_spent_Medium_value_payments
99997     High_spent_Large_value_payments
99998      Low_spent_Large_value_payments
99999                                  98
Name: Payment_Behaviour, Length: 100000, dtype: object
```

In [81]:
```
def payment_split(payment):
    if isinstance(payment,str):
        words=payment.replace('_',' ').split()
        if len(words)>2:
            return f'{words[0]}{words[1]} {words[2]}{words[3]}'#combining s
        else:
            return payment
    else:
        return payment
```

In [82]:
```
df['Payment_Behaviour']=df['Payment_Behaviour'].apply(payment_split)
```

In [83]:
```
df['Payment_Behaviour'].value_counts()
```

Out[83]:
```
Payment_Behaviour
Lowspent Smallvalue      25513
Highspent Mediumvalue    17540
Lowspent Mediumvalue     13861
Highspent Largevalue     13721
Highspent Smallvalue     11340
Lowspent Largevalue      10425
98                        7600
Name: count, dtype: int64
```

In [84]:
```
df['Payment_Behaviour'].unique()
```

Out[84]:
```
array(['Highspent Smallvalue', 'Lowspent Largevalue',
       'Lowspent Mediumvalue', 'Lowspent Smallvalue',
       'Highspent Mediumvalue', '98', 'Highspent Largevalue'],
      dtype=object)
```

In [85]:
```
mode_val=df['Payment_Behaviour'].mode()[0]
df['Payment_Behaviour'].replace('98',mode_val,inplace=True)
```

Encoding--Using a Label encoder we are mapping a numeric value to the payment behaviour class

In [86]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
df['Payment_Behaviour_encoded']=label_encoder.fit_transform(df['Payment_Beh
```

In [ ]:
```

```

In [152]:
```
df['Payment_Behaviour_encoded'].head(20),df['Payment_Behaviour'].head(20)
```

Out[152]:
```
(0     2
 1     3
 3     5
 4     1
 5     5
 6     5
 7     1
 8     5
 9     0
 11    4
 12    5
 13    0
 14    1
 15    5
 18    2
 24    3
 25    5
 26    1
 27    5
 28    3
 Name: Payment_Behaviour_encoded, dtype: int32,
 0      Highspent Smallvalue
 1       Lowspent Largevalue
 3       Lowspent Smallvalue
 4     Highspent Mediumvalue
 5       Lowspent Smallvalue
 6       Lowspent Smallvalue
 7     Highspent Mediumvalue
 8       Lowspent Smallvalue
 9      Highspent Largevalue
 11     Lowspent Mediumvalue
 12      Lowspent Smallvalue
 13     Highspent Largevalue
 14    Highspent Mediumvalue
 15      Lowspent Smallvalue
 18     Highspent Smallvalue
 24      Lowspent Largevalue
 25      Lowspent Smallvalue
 26    Highspent Mediumvalue
 27      Lowspent Smallvalue
 28      Lowspent Largevalue
 Name: Payment_Behaviour, dtype: object)
```

ENCODING MAPPING 0-HIGH SPENT LARGE VALUE 1-HIGH SPENT MEDIUM VALUE 2-HIGH SPENT SMALL VALUE 3-LOW SPENT LARGE VALUE 4-LOW SPENT MEDIUM VALUE 5-LOW SPENT SMALL VALUE

custom_mapping = { 'Lowspent Smallvalue': 0, 'Highspent Mediumvalue':4 , 'Lowspent Mediumvalue': 1, 'Highspent Largevalue':5 , 'Highspent Smallvalue': 3, 'Lowspent Largevalue': 2 }
df['Payment_mapped']=df['Payment_Behaviour_encoded'].map(custom_mapping)

Lowspent Smallvalue --2 Highspent Mediumvalue--0 Lowspent Mediumvalue--3 Highspent Largevalue--1 Highspent Smallvalue--2 Lowspent Largevalue--5

In [88]:
```
df['Payment_Behaviour_encoded']
```

Out[88]:
```
0        2
1        3
2        4
3        5
4        1
        ..
99995    0
99996    1
99997    0
99998    3
99999    5
Name: Payment_Behaviour_encoded, Length: 100000, dtype: int32
```

Convert 'Credit_History_Age' column to string.Fill the null values using mode values.Then convert the column to numerical column

In [89]:
```
df['Credit_History_Age'].astype(str)
```

Out[89]:
```
0         22 Years and 1 Months
1                           nan
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: Credit_History_Age, Length: 100000, dtype: object
```

In [90]:
```
mode_hist=df['Credit_History_Age'].mode()[0]
df['Credit_History_Age'].fillna(mode_hist,inplace=True)
```

In [91]:
```
df['Credit_History_Age'].isna().sum()
```

Out[91]:
```
0
```

Converting Credit_History column to a numerical column

In [92]:
```
def convert_history(credit_hist):
    if credit_hist=='NA':
        return np.nan
    part=credit_hist.split('and')
    year=float(part[0].split()[0])
    month=float(part[1].split()[0])
    return year+(month/12)
```

In [93]:
```
df['credit_history']=df['Credit_History_Age'].apply(convert_history)
```

In [94]:
```
df['credit_history']
df.drop(columns='Credit_History_Age',axis=1,inplace=True)
```

One Hot Encoding on Type of Loan

In [95]:
```
df['Type_of_Loan'].value_counts()
```

Out[95]:
```
Type_of_Loan
nan                                                                         11
408
Credit-Builder Loan                                                          2
160
Payday Loan                                                                  2
072
Debt Consolidation Loan                                                      2
056
Personal Loan                                                                1
992

...
Student Loan,Home Equity Loan,Payday Loan,Auto Loan
8
Personal Loan,Student Loan,Home Equity Loan,Payday Loan
8
Student Loan,Home Equity Loan,Personal Loan,Payday Loan
8
Personal Loan,Auto Loan,Student Loan,Home Equity Loan
8
Credit-Builder Loan,Payday Loan,Student Loan,Debt Consolidation Loan
8
Name: count, Length: 403, dtype: int64
```

In [96]:
```
df.dtypes
```

Out[96]:
```
Customer_ID                  object
Month                        object
Name                         object
Age                         float64
Occupation                   object
Annual_Income               float64
Monthly_Inhand_Salary       float64
Num_Bank_Accounts             int64
Num_Credit_Card               int64
Interest_Rate                 int64
Num_of_Loan                 float64
Type_of_Loan                 object
Delay_from_due_date         float64
Num_of_Delayed_Payment      float64
Changed_Credit_Limit        float64
Num_Credit_Inquiries        float64
Credit_Mix                   object
Outstanding_Debt            float64
Credit_Utilization_Ratio    float64
```

FILLING NULL VALUES

In [97]:
```
df['Monthly_Salary'] = df.groupby('Customer_ID')['Monthly_Inhand_Salary'].t
df.drop(columns='Monthly_Inhand_Salary',axis=1,inplace=True)
```

In [98]:
```
df['Age'] = df.groupby('Customer_ID')['Age'].transform(lambda x: x.fillna(m
df['Occupation'] = df.groupby('Customer_ID')['Occupation'].transform(lambda
df['Name'] = df.groupby('Customer_ID')['Name'].transform(lambda x: x.fillna
df['Annual_Income']=df.groupby('Customer_ID')['Annual_Income'].transform(la
```

In [99]:
```
df['Num_of_Loan'].fillna(0,inplace=True)
```

In [101]:
```
df.dtypes
```

Out[101]:
```
Customer_ID                  object
Month                        object
Name                         object
Age                         float64
Occupation                   object
Annual_Income               float64
Num_Bank_Accounts             int64
Num_Credit_Card               int64
Interest_Rate                 int64
Num_of_Loan                 float64
Type_of_Loan                 object
Delay_from_due_date         float64
Num_of_Delayed_Payment      float64
Changed_Credit_Limit        float64
Num_Credit_Inquiries        float64
Credit_Mix                   object
Outstanding_Debt            float64
Credit_Utilization_Ratio    float64
Payment_of_Min_Amount        object
Total_EMI_per_month         float64
Amount_invested_monthly     float64
Payment_Behaviour            object
Monthly_Balance             float64
Payment_Behaviour_encoded     int32
credit_history              float64
Monthly_Salary              float64
dtype: object
```

In [102]:
```
mean_val_amnt=df['Amount_invested_monthly'].mean()
print(mean_val_amnt)
df['Amount_invested_monthly'].fillna(mean_val_amnt,inplace=True)
```

```
195.53945602670254
```

In [103]:
```
monthly_bal_median=df['Monthly_Balance'].median()
df['Monthly_Balance'].fillna(monthly_bal_median,inplace=True)
```

In [104]:
```
df['Type_of_Loan'].fillna('Not Specified',inplace=True)
```

In [105]:
```
mode_delayed=df['Num_of_Delayed_Payment'].median()
df['Num_of_Delayed_Payment'].fillna(mode_delayed,inplace=True)
```

In [106]:
```
median_changedlimit=df['Changed_Credit_Limit'].median()
df['Changed_Credit_Limit'].fillna(median_changedlimit,inplace=True)
```

In [107]:
```
median_credit_inq=df['Num_Credit_Inquiries'].median()
df['Num_Credit_Inquiries'].fillna(median_credit_inq,inplace=True)
```

In [108]:
```
med_outstand=df['Outstanding_Debt'].median()
df['Outstanding_Debt'].fillna(med_outstand,inplace=True)
```

In [109]:
```
df.isnull().sum()
```

Out[109]:
```
Customer_ID                 0
Month                       0
Name                        0
Age                         0
Occupation                  0
Annual_Income               0
Num_Bank_Accounts           0
Num_Credit_Card             0
Interest_Rate               0
Num_of_Loan                 0
Type_of_Loan                0
Delay_from_due_date         0
Num_of_Delayed_Payment      0
Changed_Credit_Limit        0
Num_Credit_Inquiries        0
Credit_Mix                  0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Payment_of_Min_Amount       0
Total_EMI_per_month         0
Amount_invested_monthly     0
Payment_Behaviour           0
Monthly_Balance             0
Payment_Behaviour_encoded   0
credit_history              0
Monthly_Salary              0
dtype: int64
```

OUTLIERS

In [110]:
```
num_col=df.select_dtypes(include=['number'])
```

The presence of outlier is evident and hence treatment of the same is necessary using IQR method

In [111]:
```
num_col.columns
```

Out[111]:
```
Index(['Age', 'Annual_Income', '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_Rat
io',
       'Total_EMI_per_month', 'Amount_invested_monthly', 'Monthly_Balanc
e',
       'Payment_Behaviour_encoded', 'credit_history', 'Monthly_Salary'],
      dtype='object')
```

num_col.head() num_col.drop(['Age', 'Annual_Income', 'Num_Bank_Accounts', 'Num_Credit_Card', 'Interest_Rate','Num_of_Loan','Num_of_Delayed_Payment','Num_Credit_Inquiries','Total_EM

In [112]:
```
for col in enumerate(num_col):
    sns.boxplot(x=col[1],data=num_col)
    plt.show()
```

Since there are outliers present in the data we need to remove them using a suitable method.Here we will use IQR method to clear the outliers

In [113]:
```
Q1=num_col.quantile(0.25)
Q3=num_col.quantile(0.75)
IQR=Q3-Q1

mask=~((num_col< (Q1-1.5*IQR))|(num_col > (Q3 + 1.5*IQR))).any(axis=1)
df_new=df[mask]
df_new
```

Out[113]:

	Customer_ID	Month	Name	Age	Occupation	Annual_Income	Num_Bank_Accou
0	CUS_0xd40	January	Aaron Maashoh	23.0	Scientist	19114.12	
1	CUS_0xd40	February	Aaron Maashoh	23.0	Scientist	19114.12	
3	CUS_0xd40	April	Aaron Maashoh	23.0	Scientist	19114.12	
4	CUS_0xd40	May	Aaron Maashoh	23.0	Scientist	19114.12	
5	CUS_0xd40	June	Aaron Maashoh	23.0	Scientist	19114.12	
...	...	...	...	...	...	...	
99994	CUS_0x942c	March	Nicks	25.0	Mechanic	39628.99	
99995	CUS_0x942c	April	Nicks	25.0	Mechanic	39628.99	
99996	CUS_0x942c	May	Nicks	25.0	Mechanic	39628.99	
99998	CUS_0x942c	July	Nicks	25.0	Mechanic	39628.99	
99999	CUS_0x942c	August	Nicks	25.0	Mechanic	39628.99	

59563 rows × 26 columns

In [114]:
```
for col in enumerate(num_col):
    sns.boxplot(x=col[1],data=num_col)
    plt.show()
```

In [115]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

UNIVARAIATE ANALYSIS

In [116]:
```
df_sample=df_new.sample(n=1000,random_state=42)
```

In [117]:
```
df_sample
```

Out[117]:

	Customer_ID	Month	Name	Age	Occupation	Annual_Income	Num_Bank_A
37669	CUS_0xa4ba	June	Foon	39.0	Writer	34126.190	
50058	CUS_0xa57b	March	Temple-Westi	28.0	Teacher	20616.630	
333	CUS_0x6a1b	June	Toonkeln	33.0	Accountant	30788.440	
96761	CUS_0x206b	February	Claras	19.0	Entrepreneur	47641.530	
88804	CUS_0x26a9	May	Vlastelicac	27.0	Media_Manager	15155.010	
...	...	...	...	...	...	...	
94658	CUS_0x87a8	March	Bradm	27.0	Mechanic	41364.360	
62751	CUS_0xbd2c	August	Paul Carrelf	44.0	Scientist	29647.920	
77856	CUS_0x72f3	January	Nick Brownl	35.0	Media_Manager	95111.100	
29394	CUS_0x1fb5	March	Cruiseu	38.0	Scientist	51031.600	
15515	CUS_0x2123	April	Rauchz	18.0	Mechanic	15094.925	

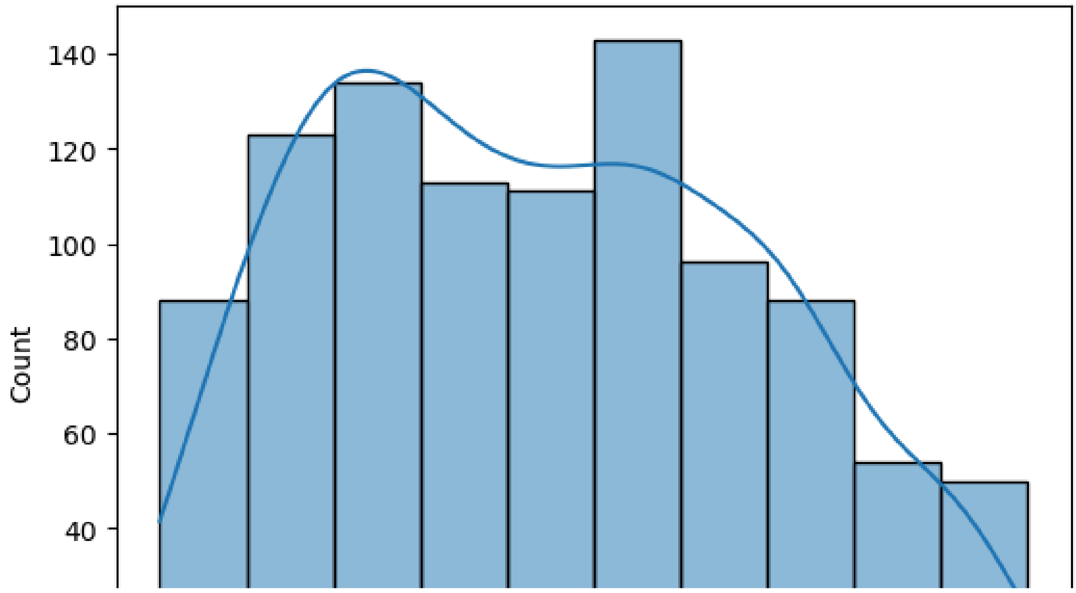
1000 rows × 26 columns

In [118]:
```
df['Age']
```

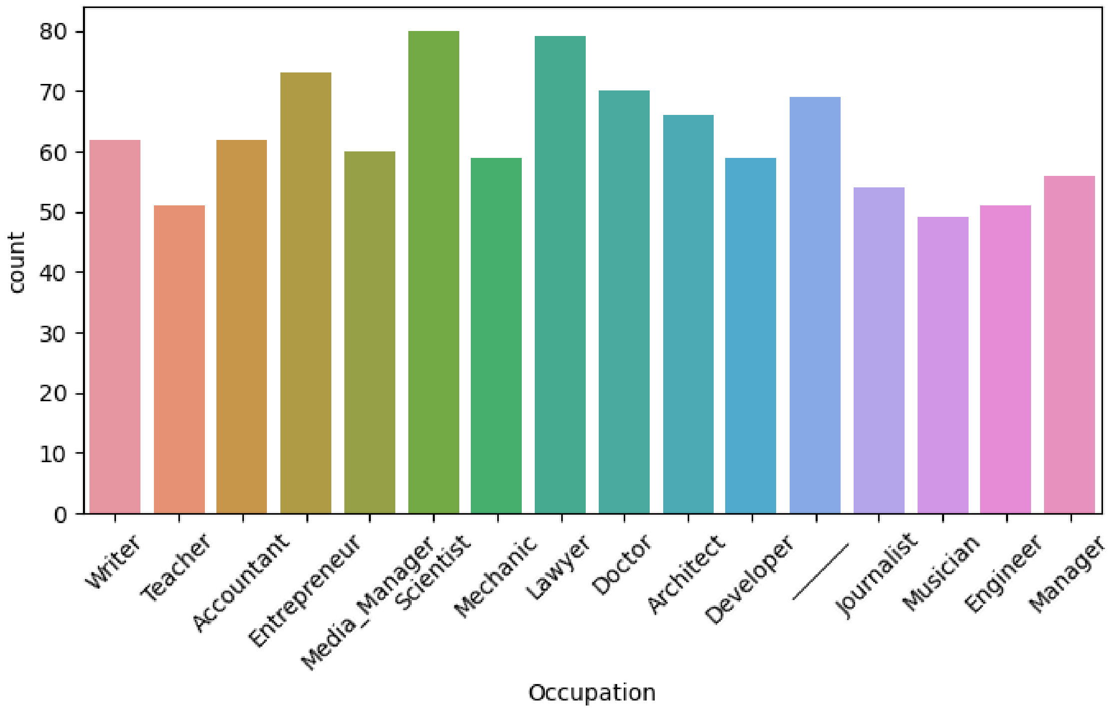
Out[118]:
```
0          23.0
1          23.0
2        -500.0
3          23.0
4          23.0
          ...
99995      25.0
99996      25.0
99997      25.0
99998      25.0
99999      25.0
Name: Age, Length: 100000, dtype: float64
```

In [119]:
```
sns.histplot(df_sample['Age'],bins=10,kde=True)
```

Out[119]: <Axes: xlabel='Age', ylabel='Count'>

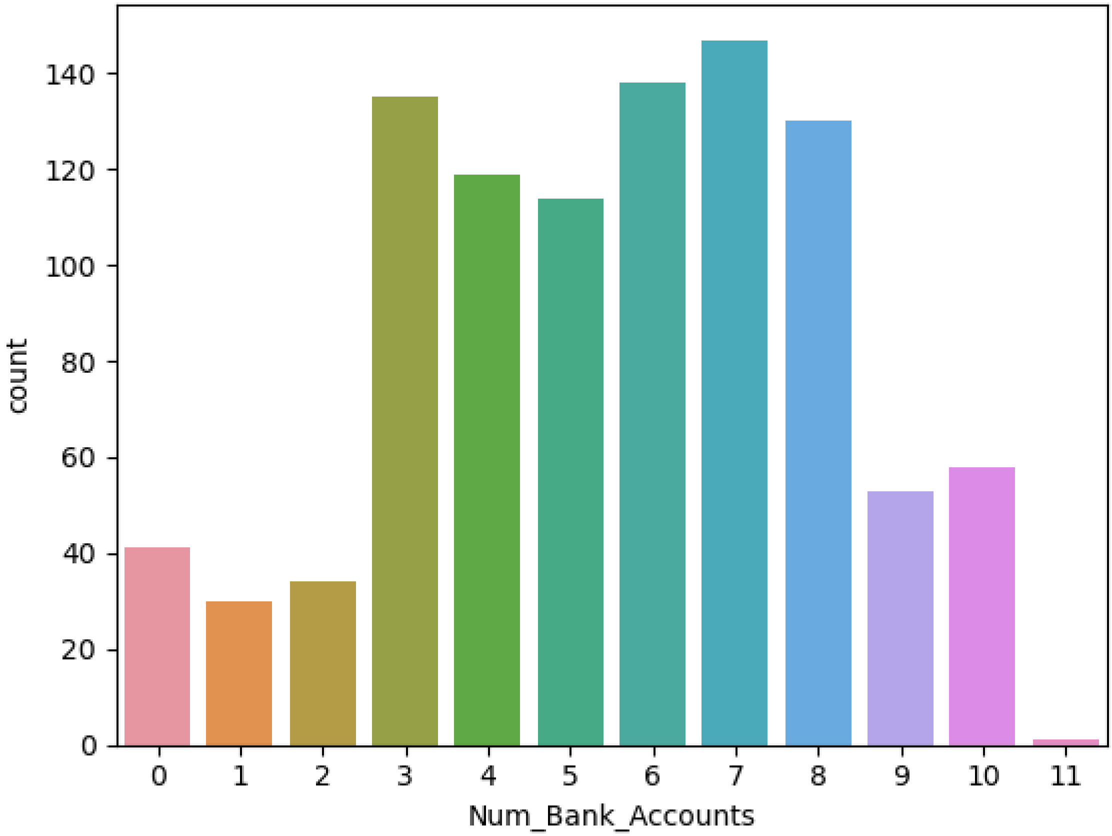


In [177]:
```
plt.figure(figsize=(8,4))
sns.countplot(x='Occupation',data=df_sample)
plt.xticks(rotation=45)
plt.show()
```



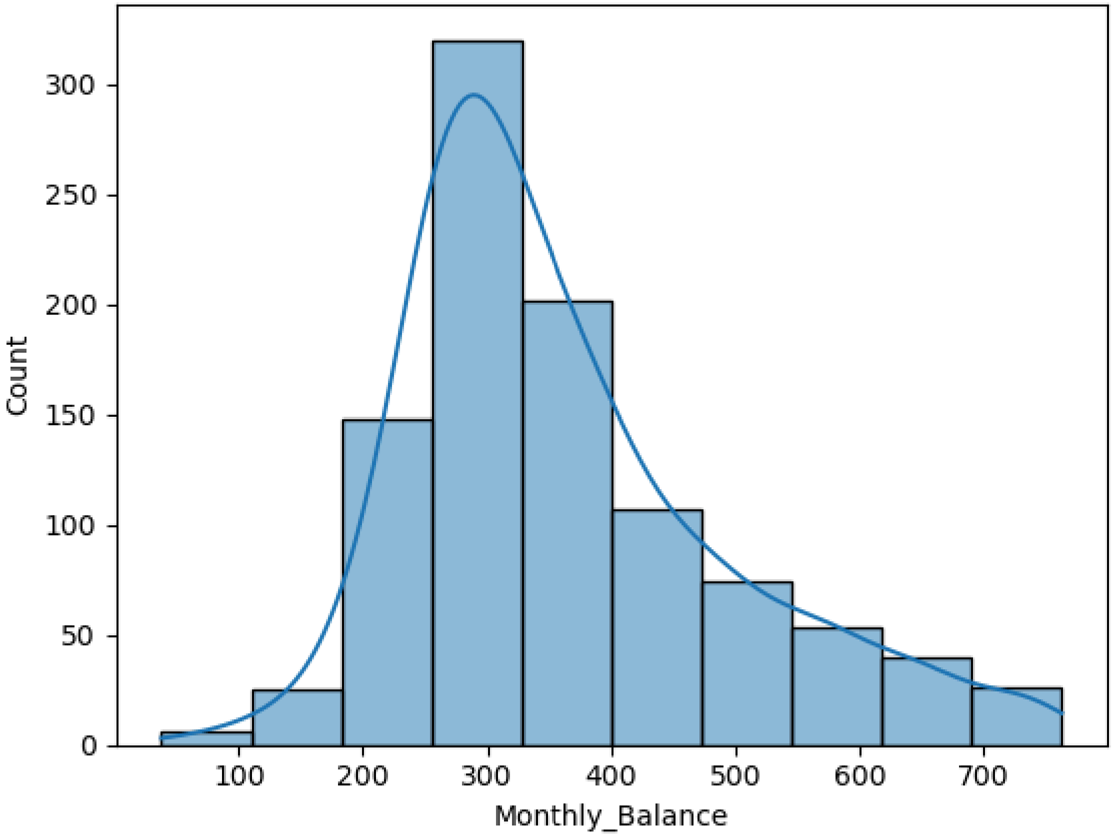
In [310]:
```
sns.countplot(x='Num_Bank_Accounts',data=df_sample)
```

Out[310]: <Axes: xlabel='Num_Bank_Accounts', ylabel='count'>



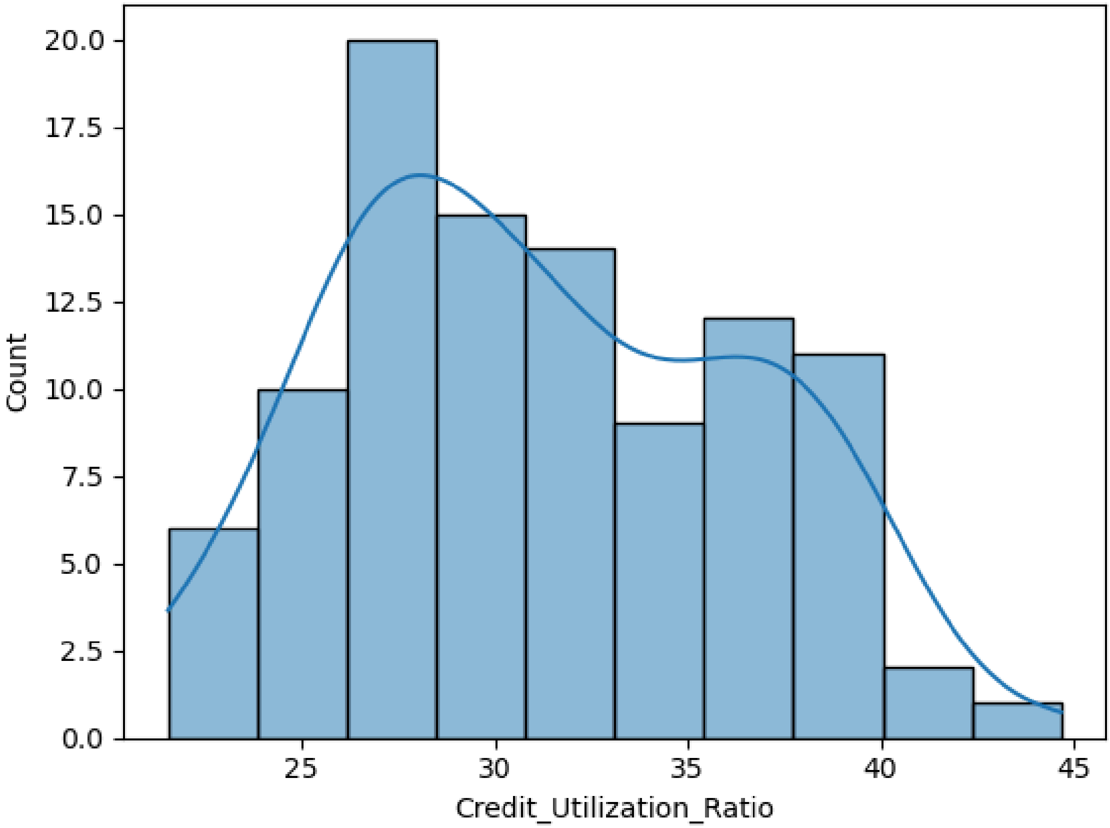
In [311]:
```
sns.histplot(df_sample['Monthly_Balance'],bins=10,kde=True)
```

Out[311]: <Axes: xlabel='Monthly_Balance', ylabel='Count'>



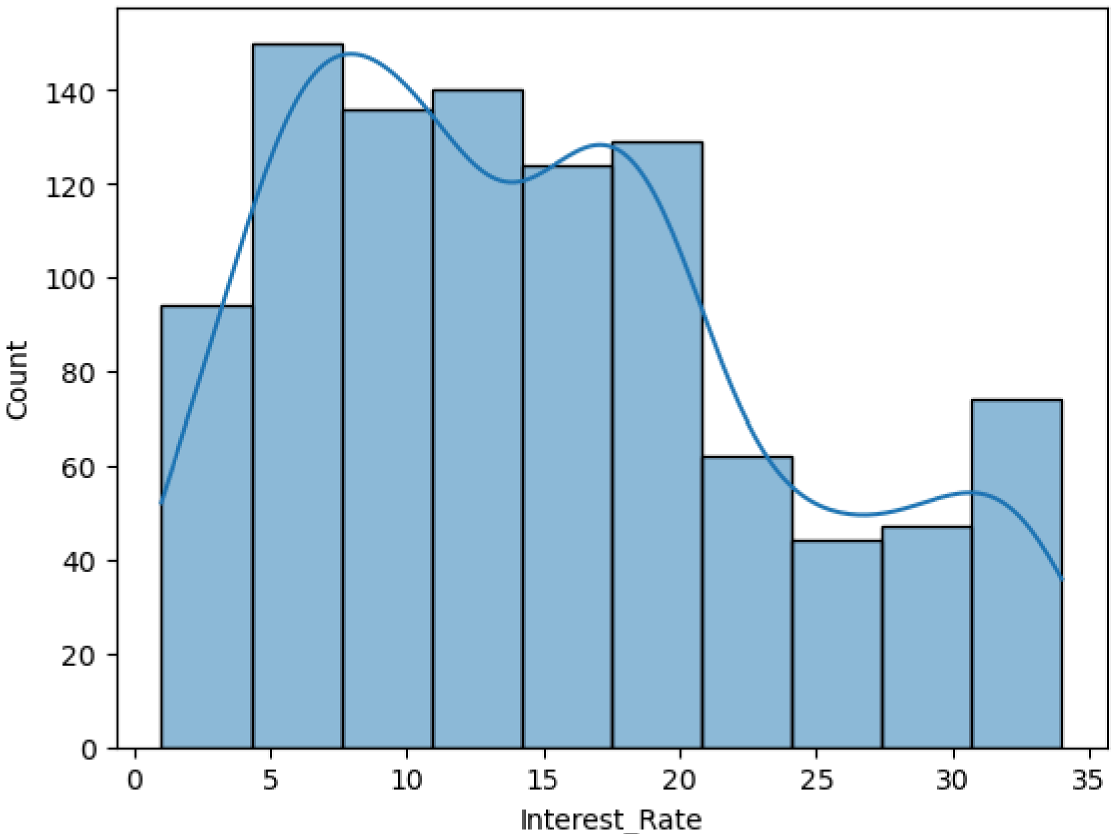
In [233]:
```
sns.histplot(df_sample['Credit_Utilization_Ratio'],bins=10,kde=True)
```

Out[233]: <Axes: xlabel='Credit_Utilization_Ratio', ylabel='Count'>



In [312]:
```
sns.histplot(df_sample['Interest_Rate'],bins=10,kde=True)
```

Out[312]: <Axes: xlabel='Interest_Rate', ylabel='Count'>

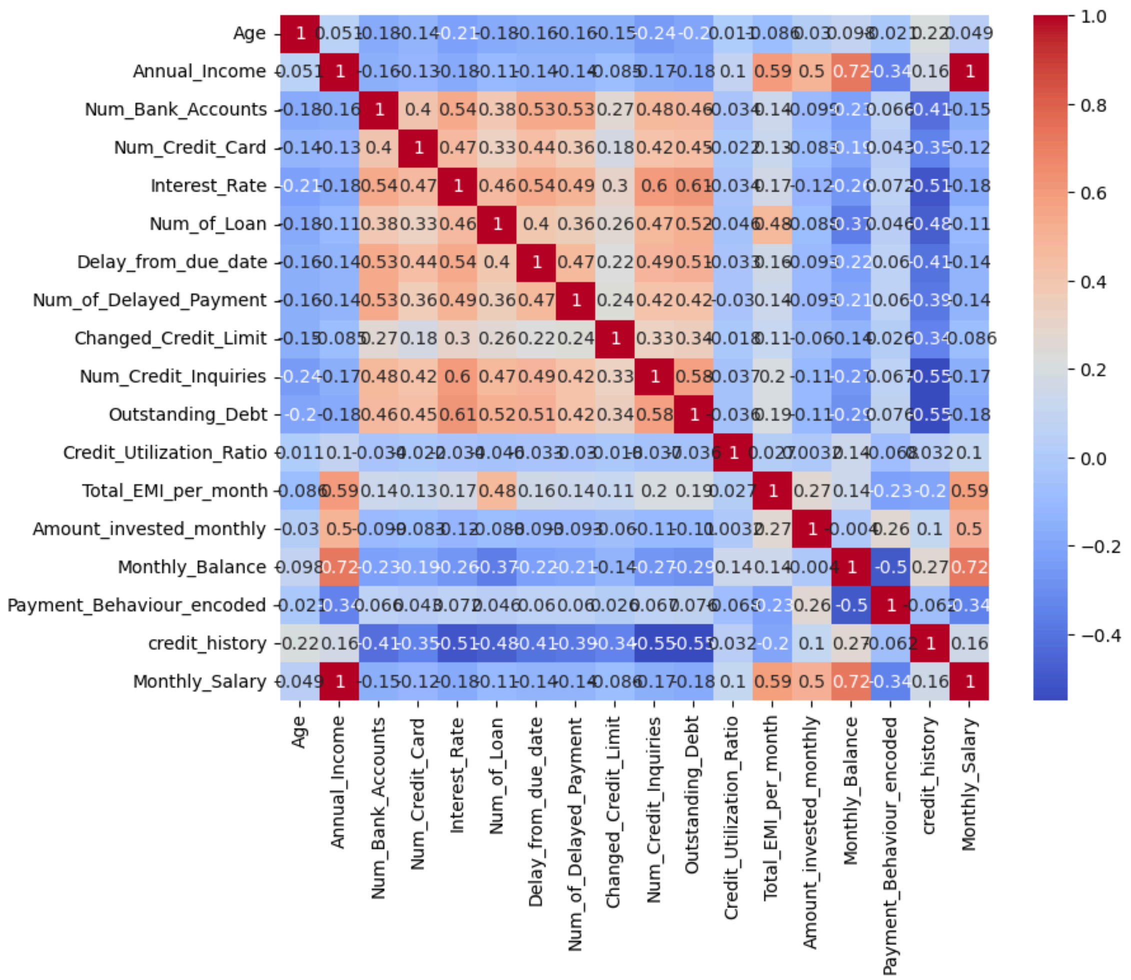


BI VARIATE ANALYSIS

In [120]:
```
num2_col=df_new.select_dtypes(include=['number'])
```

In [121]:
```
correlation_mat=num2_col.corr()
plt.figure(figsize=(9,7))
sns.heatmap(correlation_mat,cmap='coolwarm',annot=True)
```

Out[121]:
```
<Axes: >
```

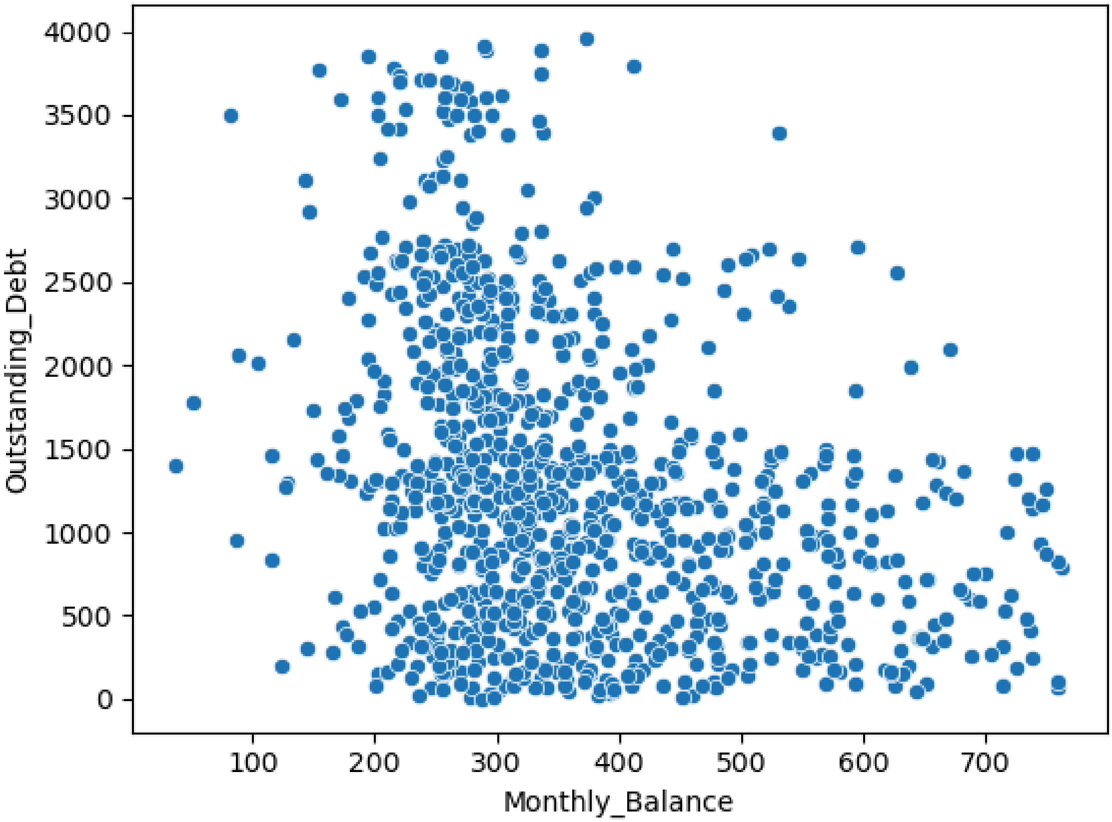


In [122]:
```
df_new.columns
```

Out[122]:
```
Index(['Customer_ID', 'Month', 'Name', 'Age', 'Occupation', 'Annual_Incom
e',
       'Num_Bank_Accounts', 'Num_Credit_Card', 'Interest_Rate', 'Num_of_Lo
an',
       'Type_of_Loan', 'Delay_from_due_date', 'Num_of_Delayed_Payment',
       'Changed_Credit_Limit', 'Num_Credit_Inquiries', 'Credit_Mix',
       'Outstanding_Debt', 'Credit_Utilization_Ratio', 'Payment_of_Min_Amo
unt',
       'Total_EMI_per_month', 'Amount_invested_monthly', 'Payment_Behaviou
r',
       'Monthly_Balance', 'Payment_Behaviour_encoded', 'credit_history',
       'Monthly_Salary'],
      dtype='object')
```

In [172]:
```
sns.scatterplot(data=df_sample,x='Monthly_Balance',y='Outstanding_Debt')
```

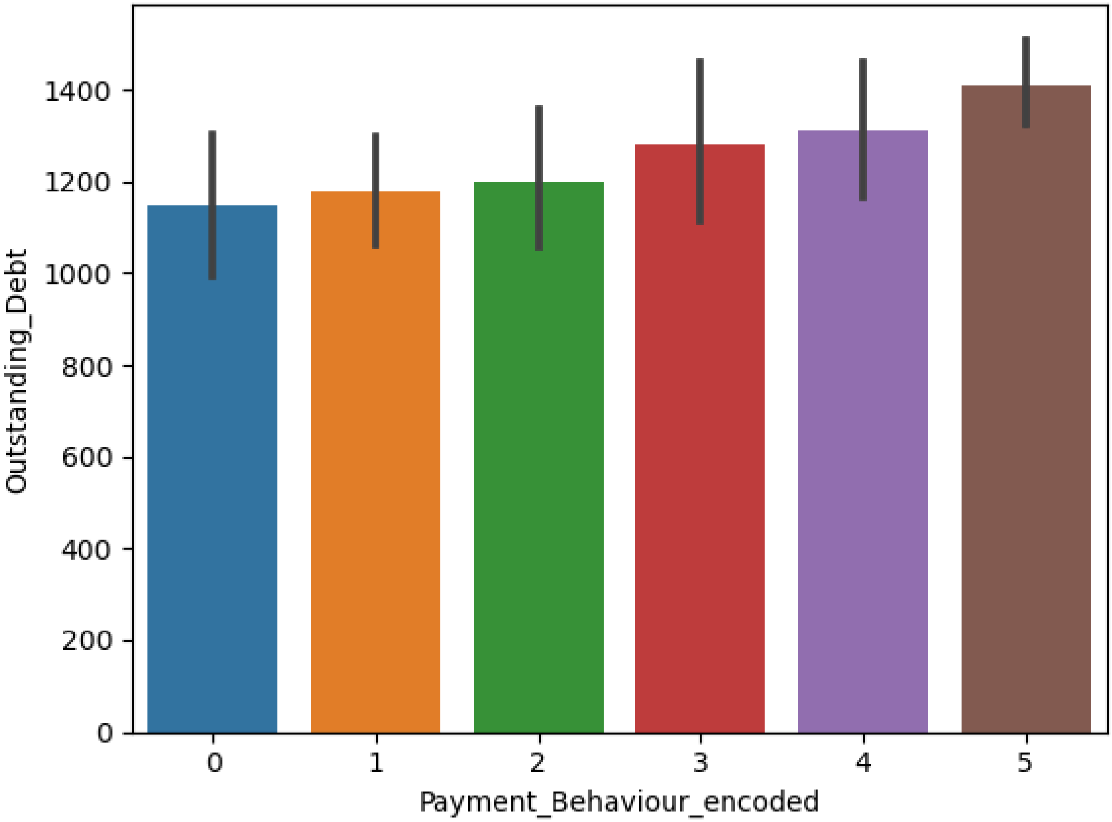
Out[172]:
```
<Axes: xlabel='Monthly_Balance', ylabel='Outstanding_Debt'>
```



Users with higher monthly balance has lower outstanding debt

In [148]:
```
sns.barplot(data=df_sample,x='Payment_Behaviour_encoded',y='Outstanding_Deb
```

Out[148]:
```
<Axes: xlabel='Payment_Behaviour_encoded', ylabel='Outstanding_Debt'>
```



0-HIGH SPENT LARGE VALUE 1-HIGH SPENT MEDIUM VALUE 2-HIGH SPENT SMALL VALUE 3-LOW SPENT LARGE VALUE 4-LOW SPENT MEDIUM VALUE 5-LOW SPENT SMALL VALUE

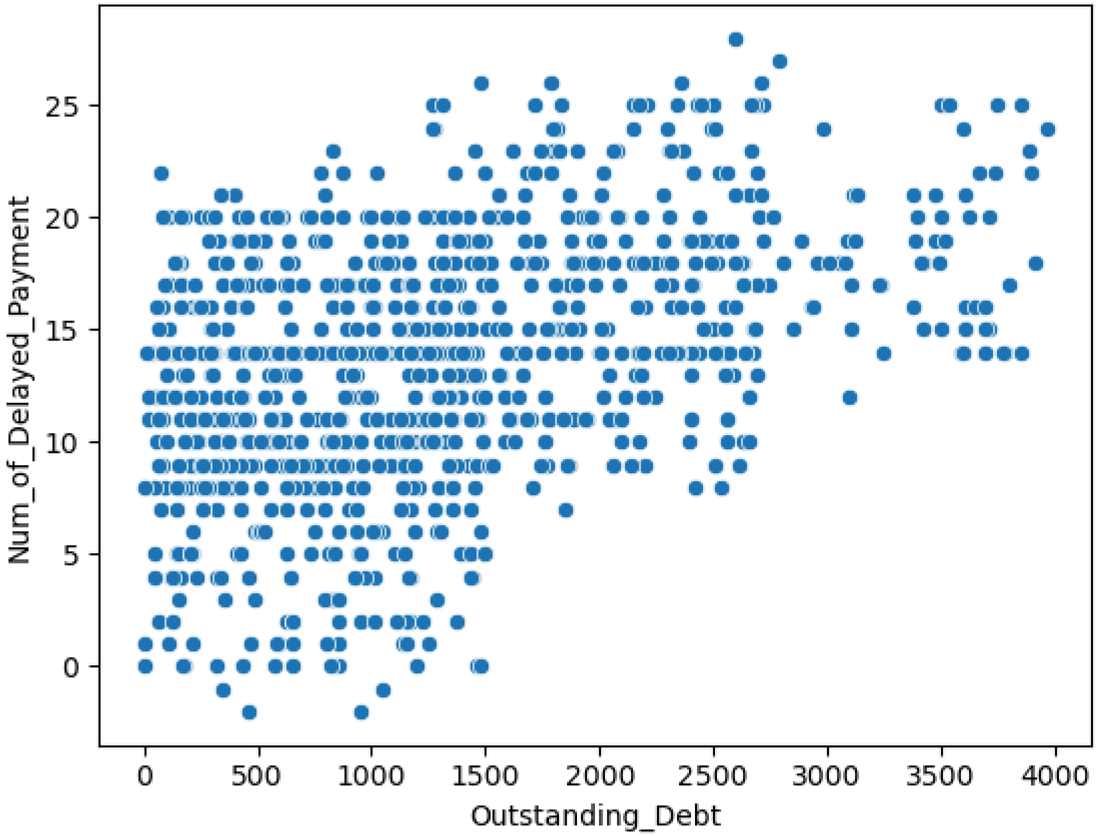
It is clear that High spent high value customers have lower outstanding debt may be due to their higher incomes

In [149]:
```
df['Payment_Behaviour_encoded']
```

Out[149]:
```
0        2
1        3
3        5
4        1
5        5
        ..
99994    1
99995    0
99996    1
99998    3
99999    5
Name: Payment_Behaviour_encoded, Length: 59563, dtype: int32
```

In [147]:
```
sns.scatterplot(data=df_sample,y='Num_of_Delayed_Payment',x='Outstanding_De
```

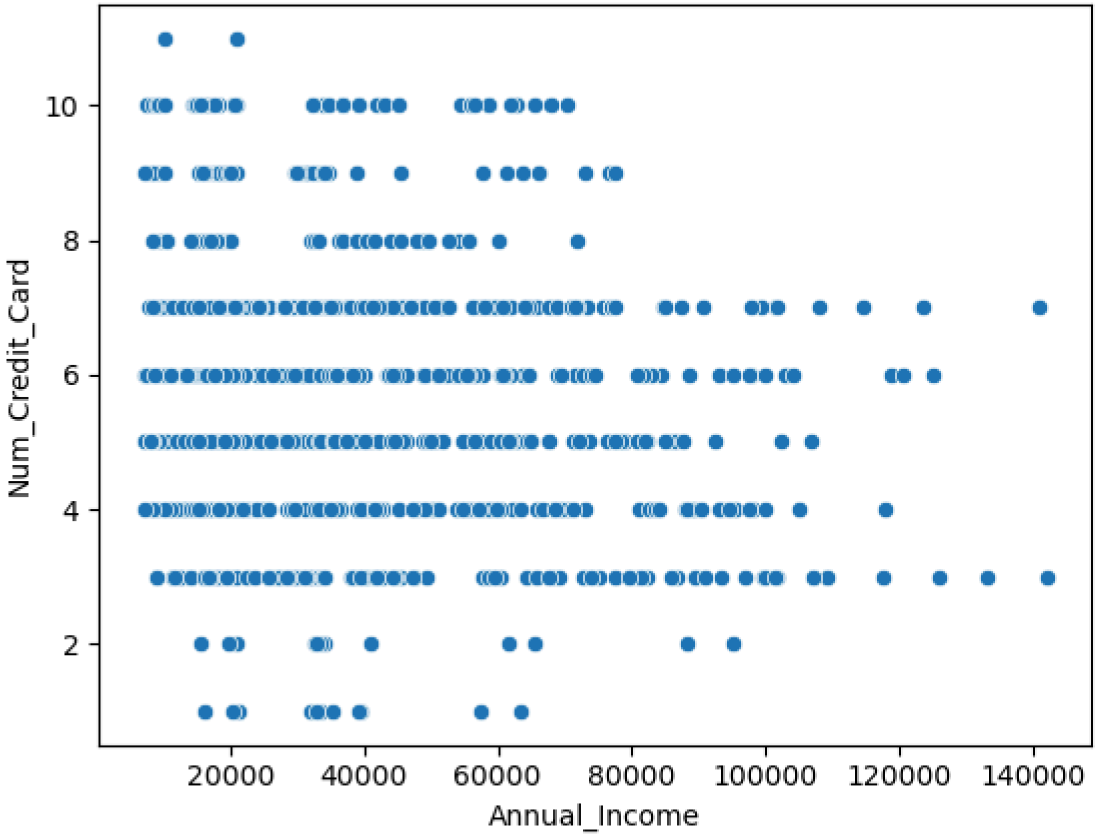
Out[147]: <Axes: xlabel='Outstanding_Debt', ylabel='Num_of_Delayed_Payment'>



It is clear indicato that as outstanding debt increases delayed payments also increases.Also smaller outsanding debts has higher cluster of delayed payments between 10 and 15

In [146]:
```
sns.scatterplot(data=df_sample,x='Annual_Income',y='Num_Credit_Card')
```

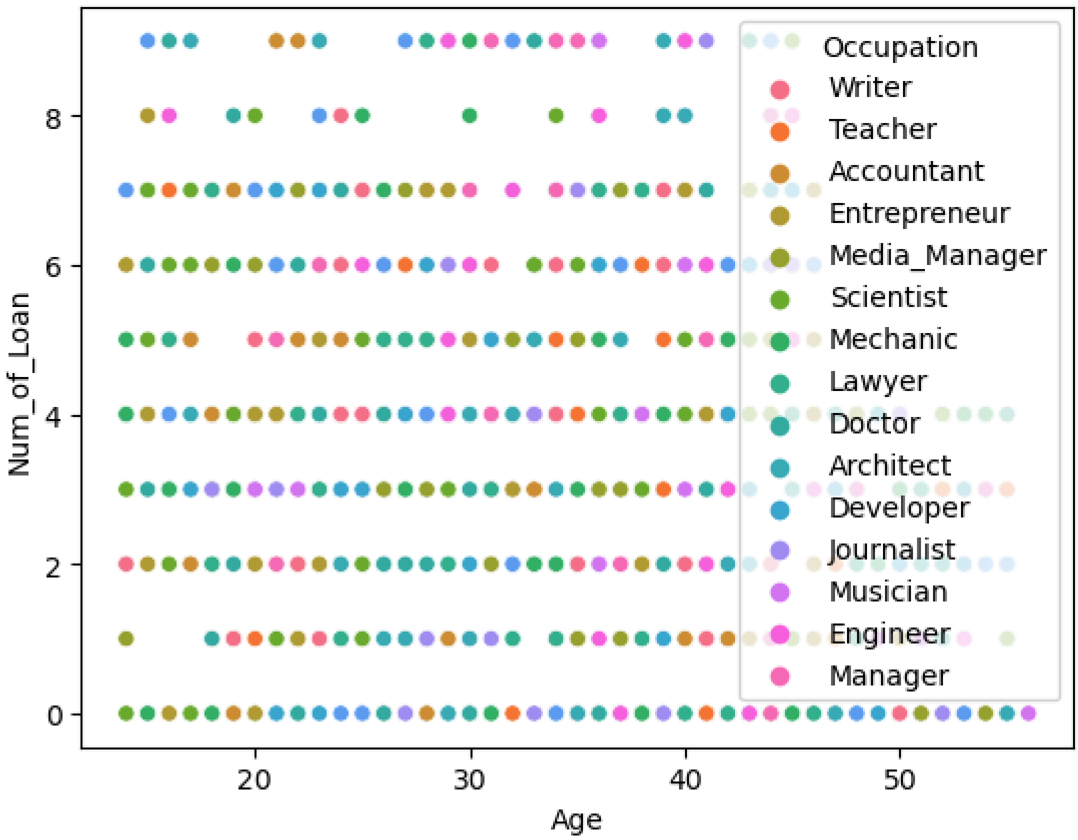
Out[146]:
```
<Axes: xlabel='Annual_Income', ylabel='Num_Credit_Card'>
```



Higher the annual income fewer the subscription for credit cards

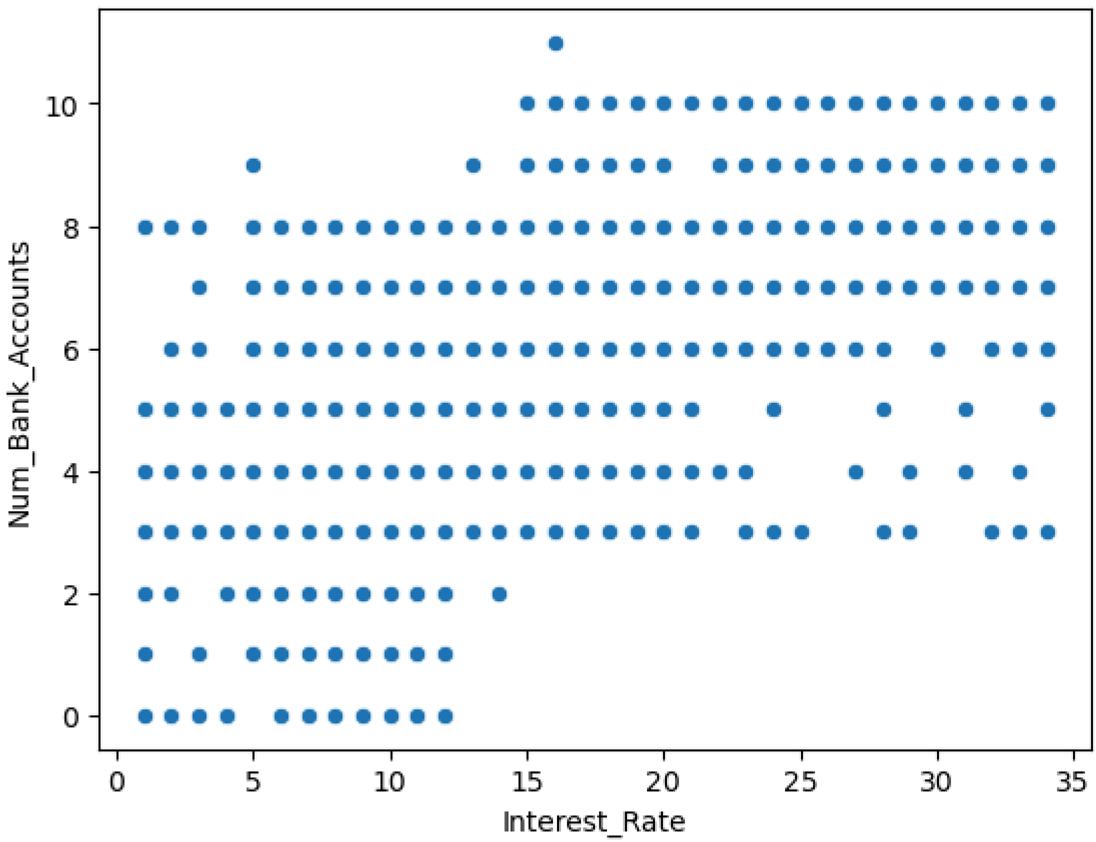
In [145]:
```
sns.scatterplot(data=df_sample,x='Age',y='Num_of_Loan',hue='Occupation')
```

Out[145]: <Axes: xlabel='Age', ylabel='Num_of_Loan'>



In [123]:
```
sns.scatterplot(data=df_sample,x='Interest_Rate',y='Num_Bank_Accounts')
```

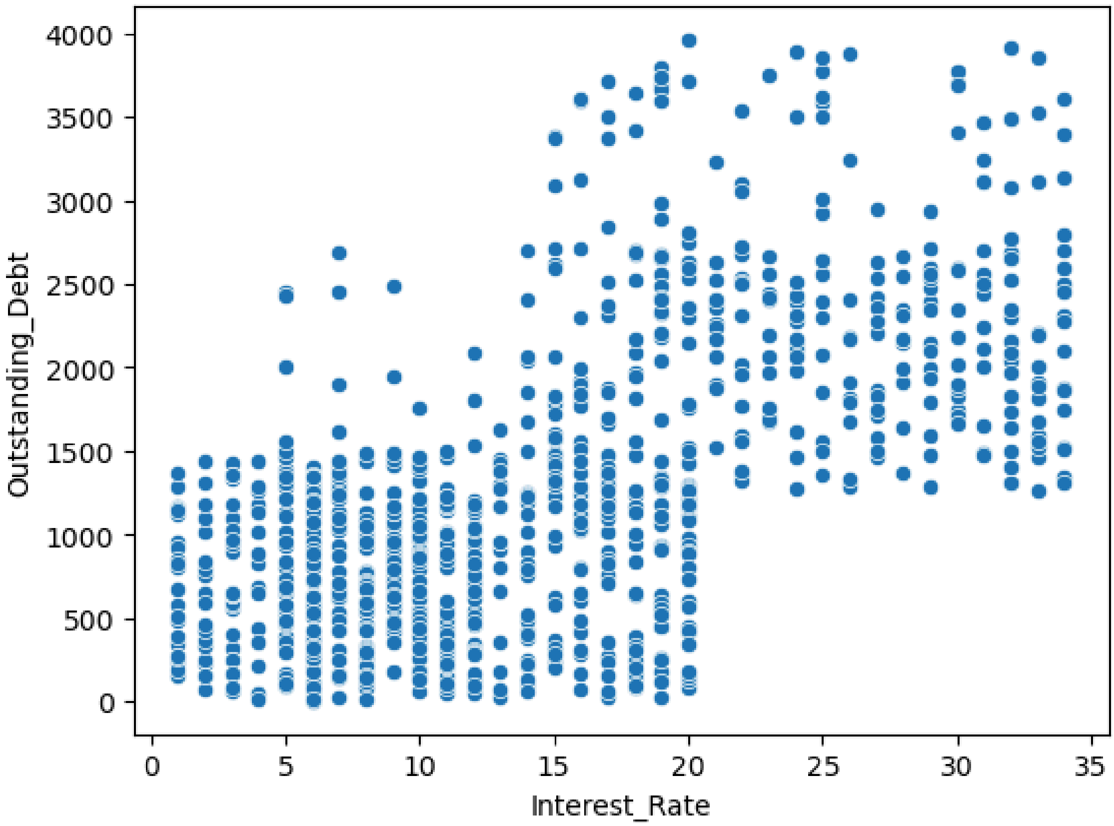
Out[123]: <Axes: xlabel='Interest_Rate', ylabel='Num_Bank_Accounts'>



Clearly as number of bank accounts increases intrest rate charged also increases

In [124]:
```
sns.scatterplot(data=df_sample,x='Interest_Rate',y='Outstanding_Debt')
```

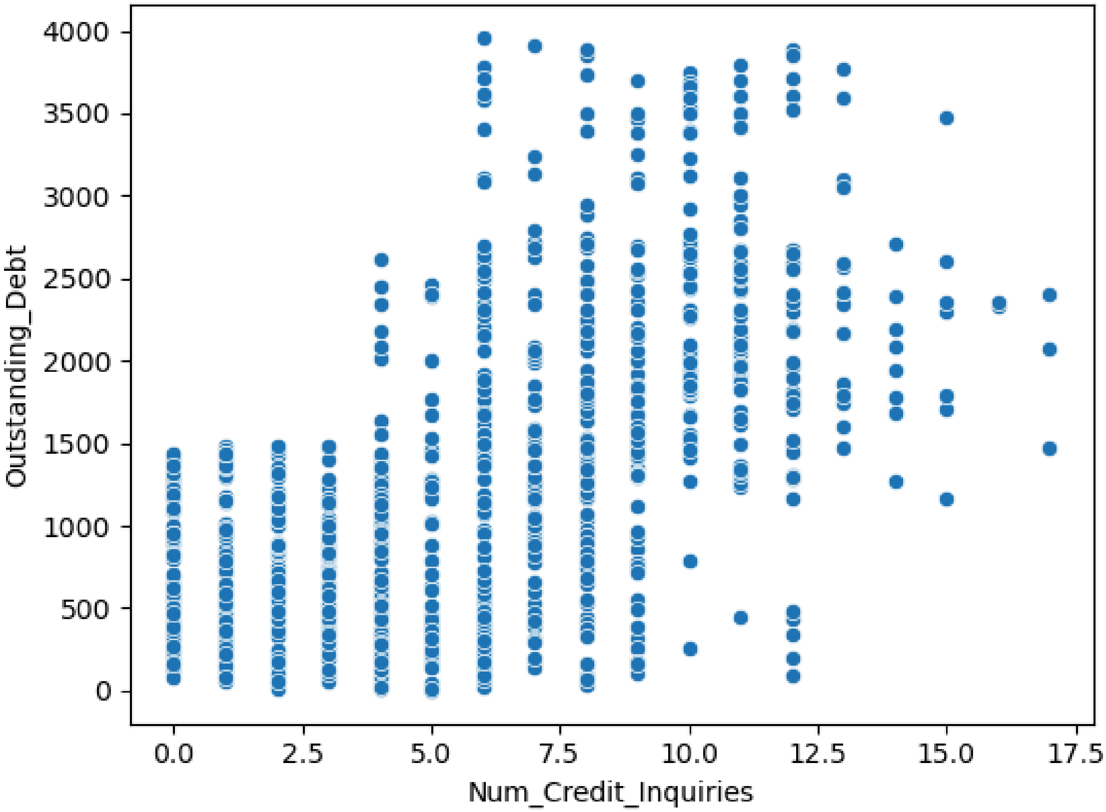
Out[124]: <Axes: xlabel='Interest_Rate', ylabel='Outstanding_Debt'>



As outstanding debt increases intrest rate also increases

In [125]:
```
sns.scatterplot(data=df_sample,x='Num_Credit_Inquiries',y='Outstanding_Debt
```

Out[125]: <Axes: xlabel='Num_Credit_Inquiries', ylabel='Outstanding_Debt'>



In [ ]:
```
sns.
```

In [ ]:

It is clear as outstanding debt increases credit enquiry increases

Feature Engineering:--Let's select some of the columns presented in our data set to arrive at the credit score calculation

FICO SCORE--A FICO credit score is a numerical representation of a person's creditworthiness, typically ranging from 300 to 850. The calculation of this score is based on five key factors, each weighted differently.

Payment History (30%) This is the most significant factor, reflecting whether a person has paid their credit accounts on time. Late payments, bankruptcies, and collections negatively impact this component.

Amounts Owed (30%) This factor considers the total amount of debt relative to available credit, known as the credit utilization ratio. A lower utilization ratio is preferable, ideally below 30%, as high amounts owed can indicate risk.

Length of Credit History (15%) A longer credit history generally contributes positively to the score. This includes the age of the oldest account, the newest account, and the average age of all accounts.

Outstanding Debt:It represents the remaning amount to be paid to clear the loan

Credit Type Score (10%) This includes recent credit inquiries and newly opened loan accounts. Frequent applications for new credit and along with the exisisting loan accounts can be seen as risky behavior and may lower the score.

Credit Mix (10%) A diverse range of credit types (e.g., credit cards, mortgages, installment loans) can positively influence the score. However, it’s not necessary to have one of each type

df['Payment_History_score']=df['Num_of_Delayed_Payment'] + (df['Payment_of_Min_Amount'] == 0 For arriving at payment hitory we use the coloumns Num_of_Delayed_Payment' and ['Payment_of_Min_Amount'] == 0 These 2 gives a fair idea of track record of the loan repayments

df['Credit_utilisation_ratio']##It is directly provided in the data set

df['credit_history']##The column was pre processed at the beginning of the document and onverted to a numerical column

df['Credit_Type_Score']=df['Num_of_loan']+df['Num_Credit_Card'] It gives an idea about the amount of liability a person owes currently

Column--- 'Credit_Mix' is provided in our data set as categorical.We are encoding using the label encoder to process the credit score which is provided below

In [153]:
```
df=df_new
```

In [154]:
```
mode_mix=df['Credit_Mix'].mode()[0]
mode_mix
df['Credit_Mix'].replace('_',mode_mix,inplace=True)
```

In [155]:
```
df['Credit_Mix'].unique()
```

Out[155]:
```
array([2, 1, 0])
```

In [156]:
```
df['Credit_Mix']=label_encoder.fit_transform(df['Credit_Mix'])
```

In [157]:
```
df['Credit_Mix']
```

Out[157]:
```
0        2
1        1
3        1
4        1
5        1
        ..
99994    2
99995    2
99996    2
99998    1
99999    1
Name: Credit_Mix, Length: 59563, dtype: int64
```

Monthly Investment Ratio

In [158]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

In [159]:
```
df[['Payment_History_Score', 'Credit_Utilization_Score', 'Credit_History_Le
    'Outstanding_Debt_Score', 'Credit_Type_Score', 'Credit_Mix_Score']] = s
    pd.DataFrame([df['Num_of_Delayed_Payment'] + (df['Payment_of_Min_Amount
                  df['Credit_Utilization_Ratio'],  # Credit utilization sco
                  df['credit_history'],  # Credit history length score
                  df['Outstanding_Debt'],  # Outstanding debt score
                  df['Num_of_Loan'] + df['Num_Credit_Card'],  # Types of cr
                  df['Credit_Mix']  # Credit inquiries score
                 ]).T)
```

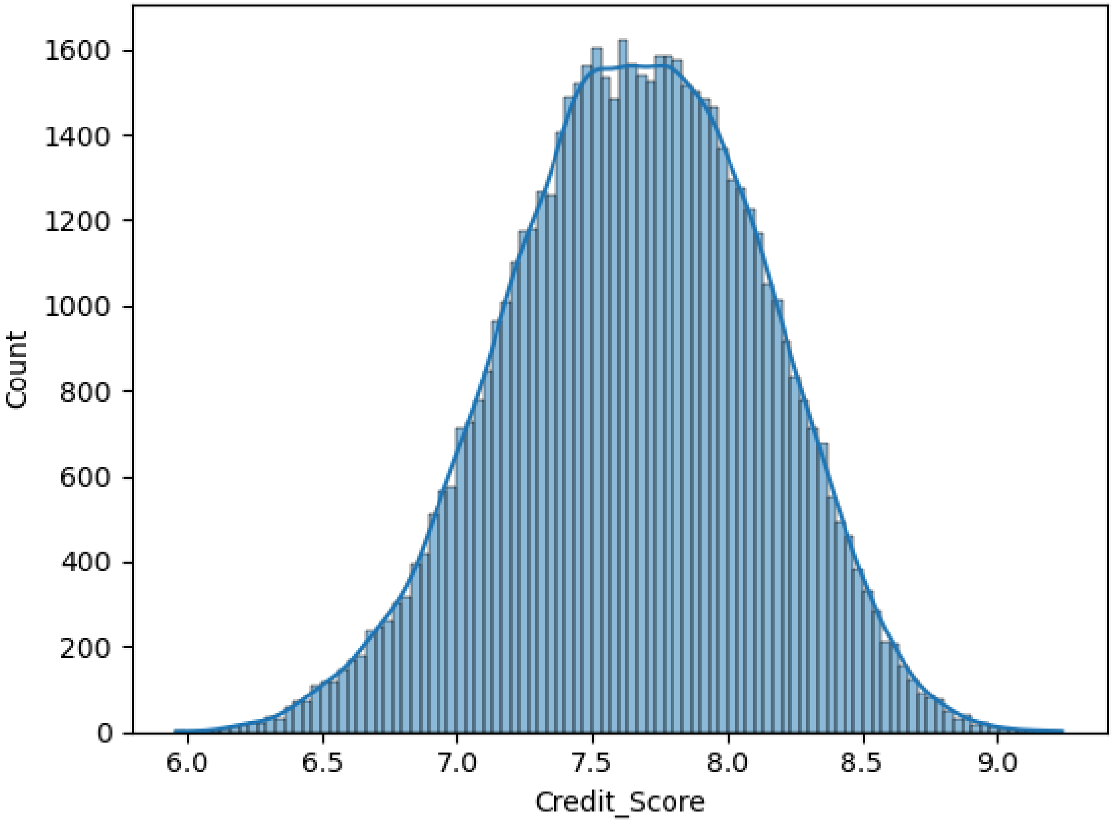
In [160]:
```
df['Credit_Score'] = (0.30 * df['Payment_History_Score'] +
                      0.30 * df['Credit_Utilization_Score'] +
                      0.15 * df['Credit_History_Length_Score'] +
                      0.10 * df['Outstanding_Debt_Score'] +
                      0.10 * df['Credit_Type_Score'] +
                      0.10 * df['Credit_Mix_Score'])

# Scale the credit score to 5-10 range (adjusted scaling)
df['Credit_Score'] = 5 + 5 * df['Credit_Score']
```

In [161]:
```
df['Credit_Score']
```

Out[161]:
```
0        7.144690
1        7.434413
3        7.056342
4        7.106916
5        6.787906
           ...   
99994    8.092681
99995    7.834772
99996    8.226926
99998    7.861259
99999    7.512675
Name: Credit_Score, Length: 59563, dtype: float64
```

In [162]:
```
sns.histplot(df['Credit_Score'],kde=True)
```

Out[162]: <Axes: xlabel='Credit_Score', ylabel='Count'>



In [140]:
```
df_sample2=df.sample(n=5000,random_state=50)
```

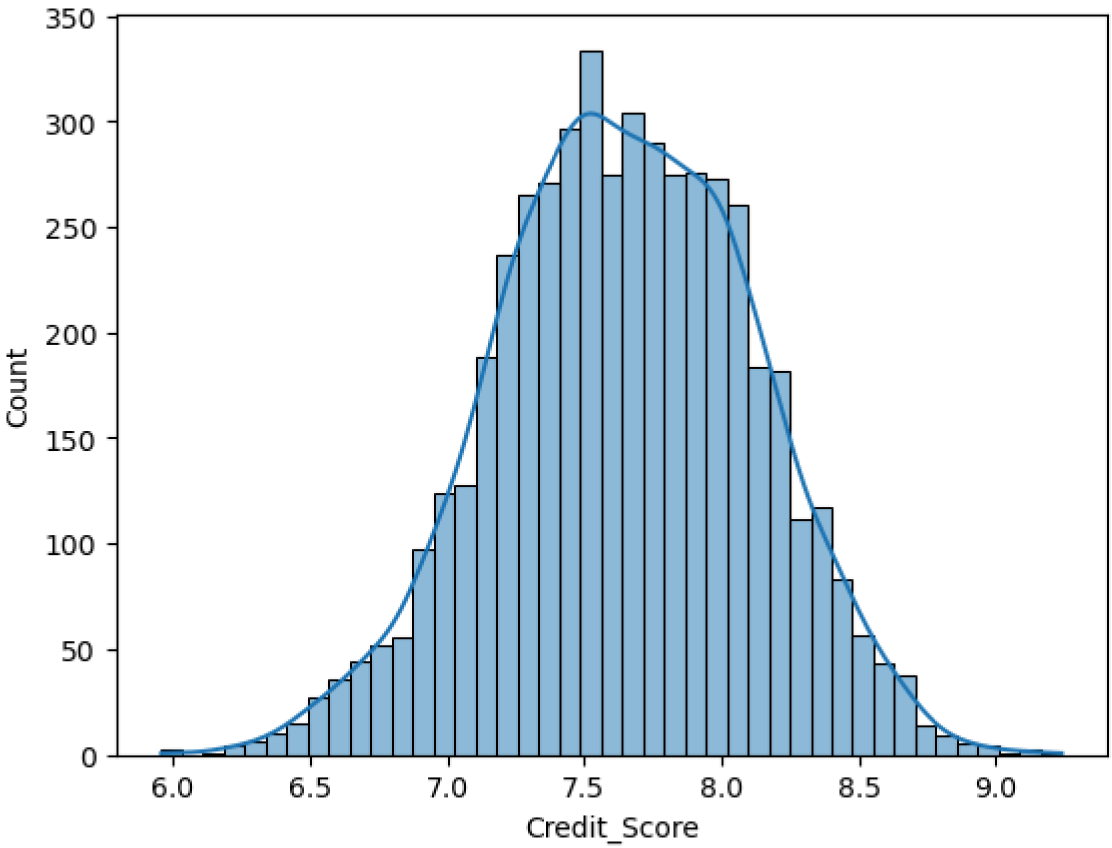
In [141]:
```
df_sample2
```

Out[141]:

	Customer_ID	Month	Name	Age	Occupation	Annual_Income	Num_Bank_A
51191	CUS_0xbe6f	August	Emotoy	41.0	Journalist	34945.160	
39273	CUS_0x6073	February	Rick Rothackern	37.0	Teacher	107595.680	
69126	CUS_0xa024	July	Alex rax	47.0	Lawyer	84358.500	
4001	CUS_0x8484	February	Tom Halsy	32.0	Lawyer	44390.760	
93741	CUS_0x1900	June	Strupczewskix	26.0	Accountant	8137.625	
...	...	...	...	...	...	...	
62544	CUS_0x727f	January	Lawlerj	32.0	Teacher	82183.340	
63735	CUS_0x45cb	August	Ethan Bilbyi	43.0	Journalist	58170.760	
76875	CUS_0x7a96	April	Langep	29.0	Manager	24424.700	
20812	CUS_0x76c1	May	Jessicaf	29.0	Writer	66422.980	
7144	CUS_0xb956	January	K.T. Arasuv	28.0	Mechanic	139538.320	

5000 rows × 33 columns

In [142]:
```
sns.histplot(df_sample2['Credit_Score'],kde=True)
plt.show()
```



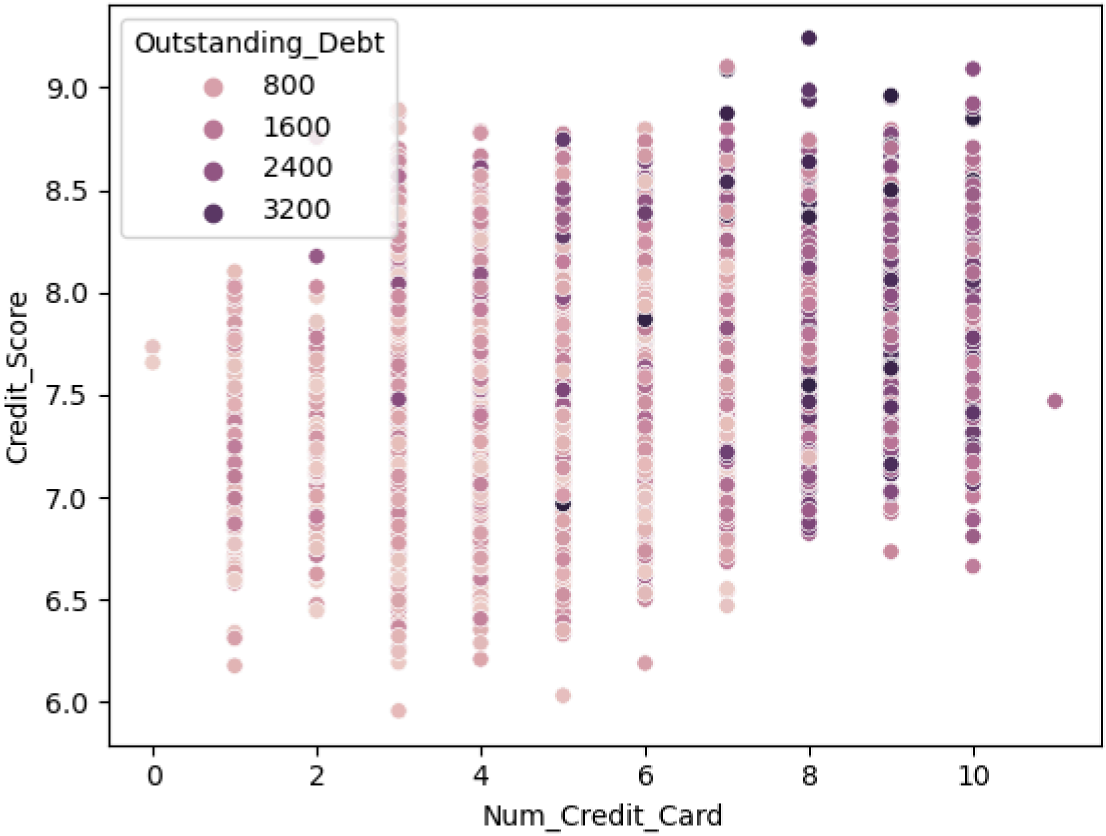
In [ ]:

In [163]:
```
df.columns
```

Out[163]:
```
Index(['Customer_ID', 'Month', 'Name', 'Age', 'Occupation', 'Annual_Incom
e',
       'Num_Bank_Accounts', 'Num_Credit_Card', 'Interest_Rate', 'Num_of_Lo
an',
       'Type_of_Loan', 'Delay_from_due_date', 'Num_of_Delayed_Payment',
       'Changed_Credit_Limit', 'Num_Credit_Inquiries', 'Credit_Mix',
       'Outstanding_Debt', 'Credit_Utilization_Ratio', 'Payment_of_Min_Amo
unt',
       'Total_EMI_per_month', 'Amount_invested_monthly', 'Payment_Behaviou
r',
       'Monthly_Balance', 'Payment_Behaviour_encoded', 'credit_history',
       'Monthly_Salary', 'Payment_History_Score', 'Credit_Utilization_Scor
e',
       'Credit_History_Length_Score', 'Outstanding_Debt_Score',
       'Credit_Type_Score', 'Credit_Mix_Score', 'Credit_Score'],
      dtype='object')
```

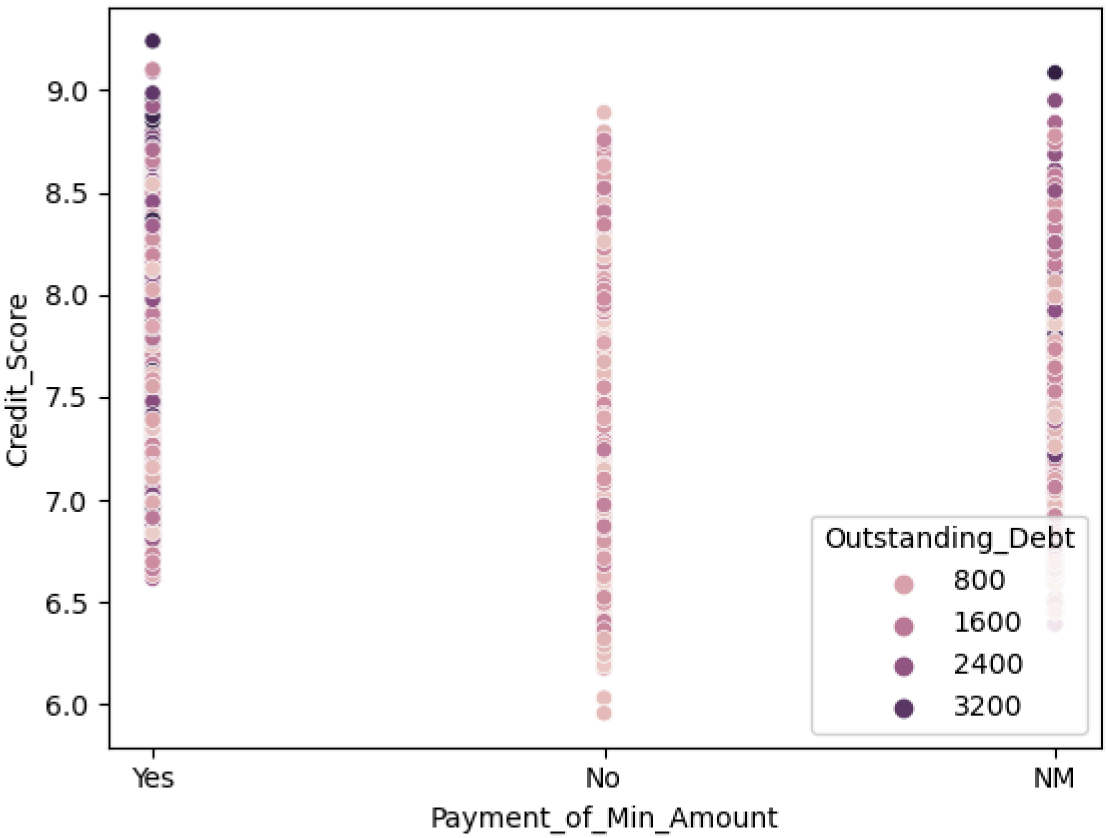
In [166]:
```
sns.scatterplot(data=df_sample2,x='Num_Credit_Card',y='Credit_Score',hue='O
```

Out[166]: <Axes: xlabel='Num_Credit_Card', ylabel='Credit_Score'>

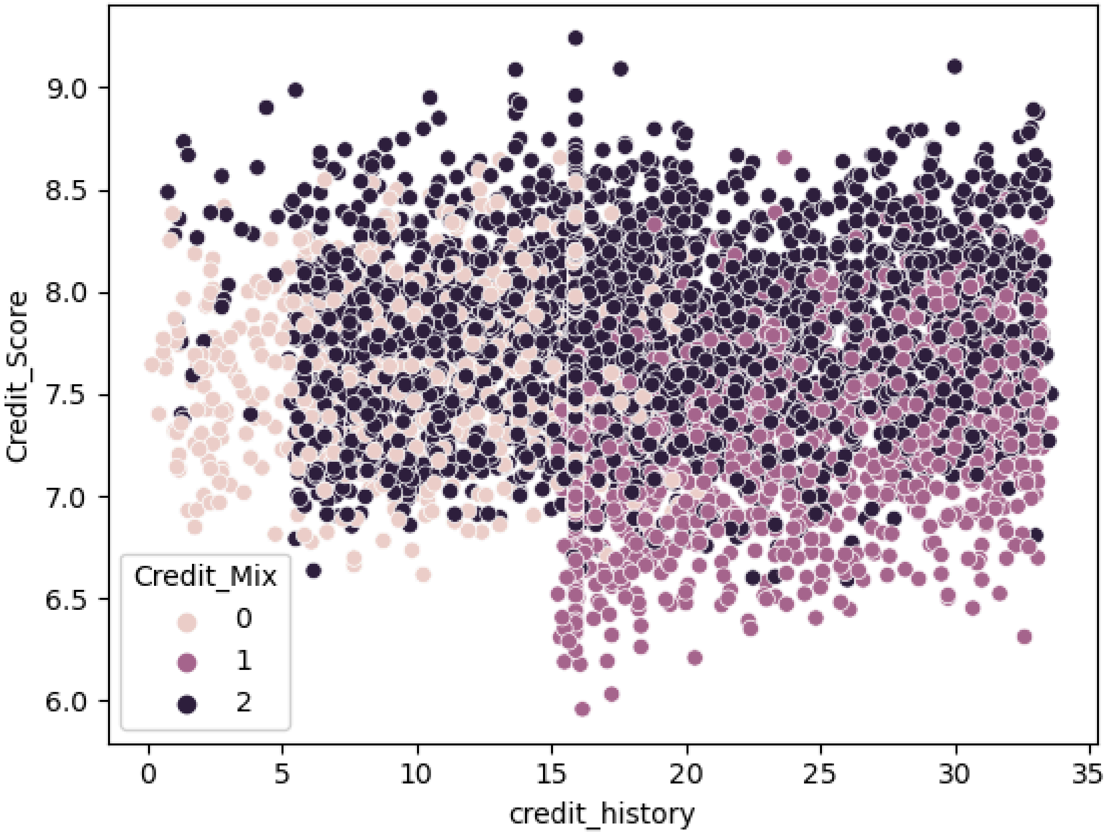


In [178]:
```
sns.scatterplot(data=df_sample2,x='Payment_of_Min_Amount',y='Credit_Score',
```

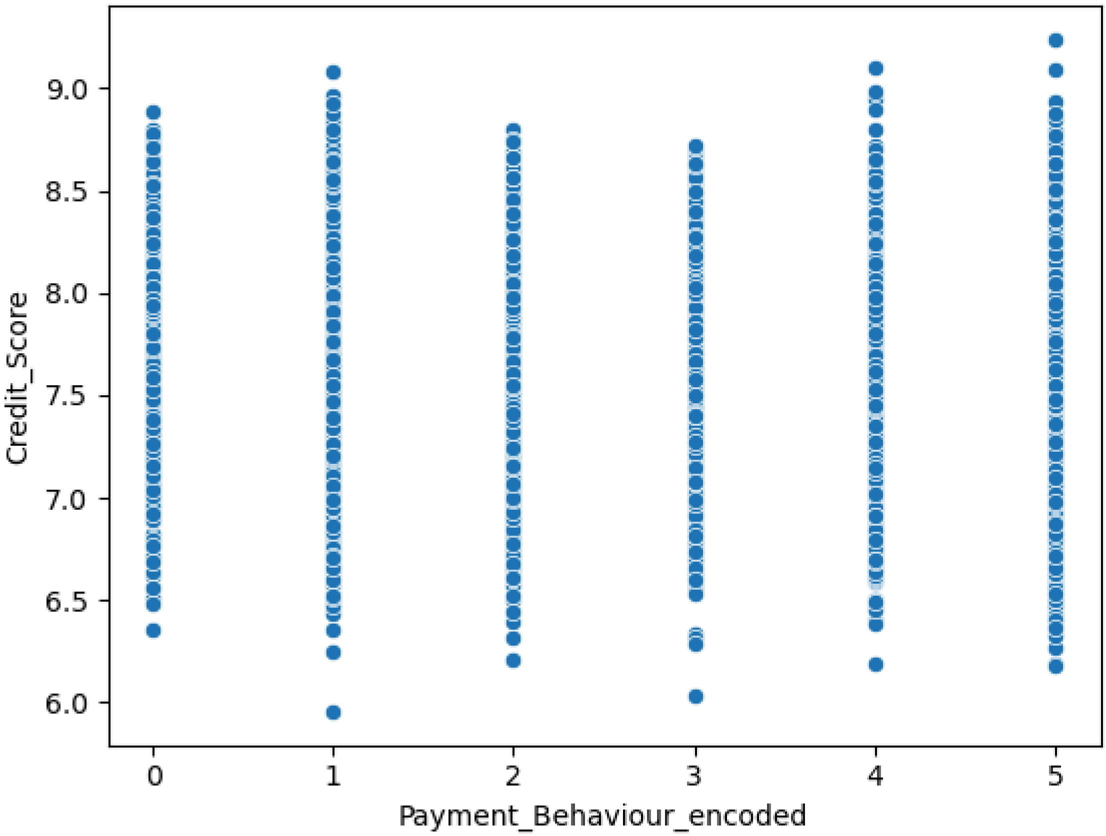
Out[178]: <Axes: xlabel='Payment_of_Min_Amount', ylabel='Credit_Score'>



In [170]:
```
sns.scatterplot(data=df_sample2,x='credit_history',y='Credit_Score',hue='Cr
```
Out[170]: <Axes: xlabel='credit_history', ylabel='Credit_Score'>

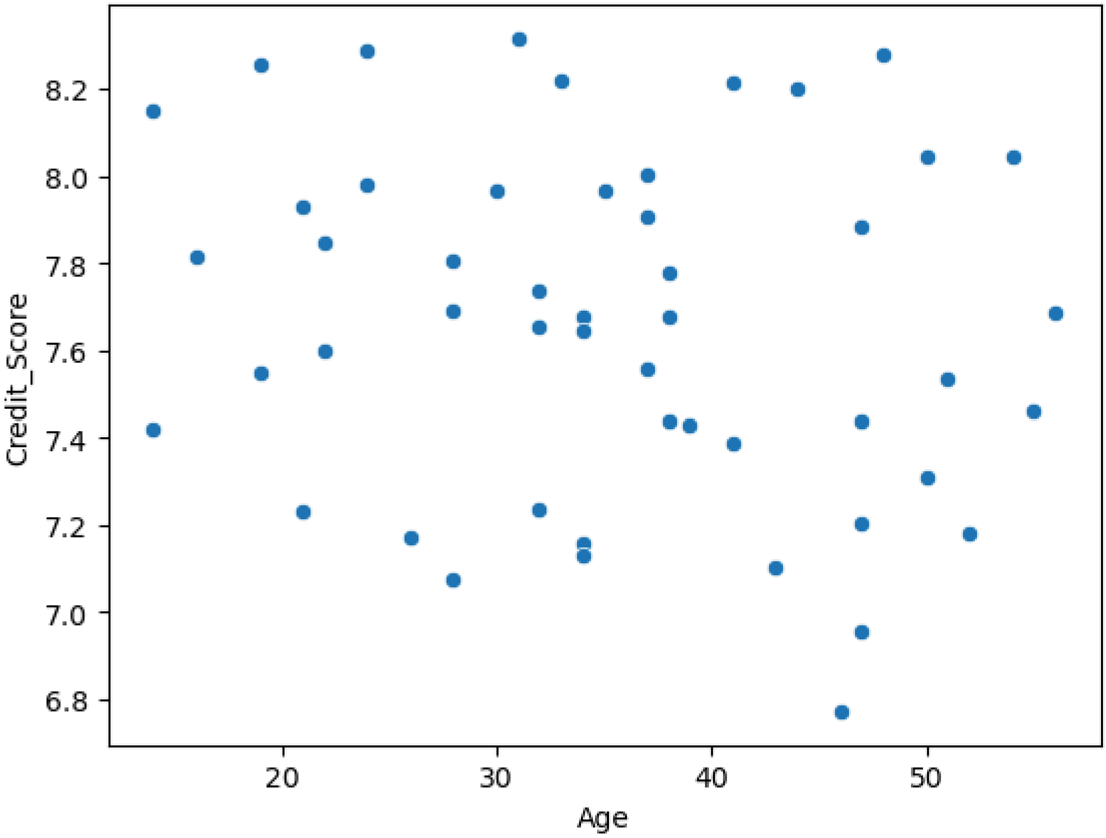


In [168]:
```
sns.scatterplot(data=df_sample2,x='Payment_Behaviour_encoded',y='Credit_Sco
```
Out[168]: <Axes: xlabel='Payment_Behaviour_encoded', ylabel='Credit_Score'>



Most of the applicants have a credit score in the range 7-8 which reflects a healthy credit culture

In [143]:
```
sns.scatterplot(x='Age',y='Credit_Score',data=df_sample2.head(50))
plt.show()
```



Age and credit score does not seems to bear a relation

INFERENCE: 1)The credit score data reveals that most of the scores lies between 7 and 8. 2)The age group of 40+ takes fewer credit.The age group 30-40 has highest takers of loans 3)Age group 30-40 displays the highest credit scores 4)Users with HIGH SPENT HIGH VALUE and LOW SPENT LOW VALUE have higher credit scores 5)Users with longer credit history and better credit mix has higher credit score 6)Users wo paid minimum amount has better credit scores compared to users who dont pay anything