Credit EDA & Credit Score Calculation with Python

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.

```
import pandas as pd
import numpy as np
```

```
!gdown 1MqxHAUN4J0emIpEDyZlMjIjkM6rLDNMf
```

```
Downloading...
From: https://drive.google.com/uc?id=1MqxHAUN4J0emIpEDyZlMjIjkM6rLDNMf
To: /content/Credit_score.csv
100% 27.4M/27.4M [00:00<00:00, 128MB/s]
```

```
data = pd.read_csv('Credit_score.csv')
```

```
<ipython-input-4-42a8ab7aaae7>:1: DtypeWarning: Columns (26) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv('Credit_score.csv')
```

```
data
```

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	...	Num_Credit_Inqu
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
99995	0x25fe9	CUS_0x942c	April	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	...	
99996	0x25fea	CUS_0x942c	May	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	...	
99997	0x25feb	CUS_0x942c	June	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	...	
99998	0x25fec	CUS_0x942c	July	Nicks	25	078-73-5990	Mechanic	39628.99	3359.415833	4	...	
99999	0x25fed	CUS_0x942c	August	Nicks	25	078-73-5990	Mechanic	39628.99_	3359.415833	4	...	

100000 rows × 27 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 27 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   ID                        100000 non-null  object
 1   Customer_ID               100000 non-null  object
 2   Month                     100000 non-null  object
 3   Name                      90015 non-null   object
 4   Age                       100000 non-null  object
 5   SSN                       100000 non-null  object
 6   Occupation                100000 non-null  object
 7   Annual_Income             100000 non-null  object
 8   Monthly_Inhand_Salary     84998 non-null   float64
 9   Num_Bank_Accounts         100000 non-null  int64
 10  Num_Credit_Card           100000 non-null  int64
 11  Interest_Rate             100000 non-null  int64
```

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

```
data.isna().sum()
```

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

```
data[['Customer_ID','Age']]
```

	Customer_ID	Age
0	CUS_0xd40	23
1	CUS_0xd40	23
2	CUS_0xd40	-500
3	CUS_0xd40	23
4	CUS_0xd40	23
...	...	...
99995	CUS_0x942c	25
99996	CUS_0x942c	25
99997	CUS_0x942c	25
99998	CUS_0x942c	25
99999	CUS_0x942c	25

100000 rows × 2 columns

```
data[data['Age'].str[-1]== "_"]['Age']
```

```
8          28_
54         34_
58         30_
71         24_
89         33_
          ...
99908    4808_
99922      38_
99933      38_
99942      48_
99987      28_
Name: Age, Length: 4939, dtype: object
```

```
data.loc[data['Age'].str[-1] == '_','Age'] = data['Age'].str[:-1]
```

```
data.loc[data['Age'].str[0] == '-','Age'] = data['Age'].str[1:]
```

```
data['Age']
```

```
0         23
1         23
2        500
3         23
4         23
        ...
99995     25
99996     25
99997     25
99998     25
99999     25
Name: Age, Length: 100000, dtype: object
```

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

```
0         23
1         23
2        500
3         23
4         23
        ...
99995     25
99996     25
99997     25
99998     25
99999     25
Name: Age, Length: 100000, dtype: int64
```

Referencing the maximum age of a person in th world as 122 then using this filtering out the outliers in it.

```
data.loc[data['Age']> 122,'Age'].count()
```

```
2770
```

there are 2770 entries where age are more than 122 years so to deal with these wrong entries we will use mode of age for every customer_ID and replace these wrong entries with mode of it

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

```
0        23
1        23
2        23
3        23
4        23
         ..
99995    25
99996    25
99997    25
99998    25
99999    25
Name: Age, Length: 100000, dtype: int64
```

```
data.loc[data['Age']> 122,'Age'].count()
```

```
0
```

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

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

```
data[['Customer_ID','Age']]
```

	Customer_ID	Age
0	CUS_0xd40	23
1	CUS_0xd40	23
2	CUS_0xd40	23

Dealing with missing value of Names and Monthly_Inhand_Salary in given data we can use mode of name for every customer_id which will replace the null value in it.

[ ] ↳ 2 cells hidden

For Name

[ ] ↳ 1 cell hidden

For Monthly_Inhand_Salary

```
data['Monthly_Inhand_Salary']= data.groupby('Customer_ID')['Monthly_Inhand_Salary'].transform(lambda x: x.mode()[0])
data.head(10)
```

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	...	Num_Credit_I
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	1824.843333	3	...	
2	0x1604	CUS_0xd40	March	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	...	
3	0x1605	CUS_0xd40	April	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	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	1824.843333	3	...	
6	0x1608	CUS_0xd40	July	Aaron Maashoh	23	821-00-0265	Scientist	19114.12	1824.843333	3	...	
7	0x1609	CUS_0xd40	August	Aaron Maashoh	23	#F%$D@*&8	Scientist	19114.12	1824.843333	3	...	
8	0x160e	CUS_0x21b1	January	Rick Rothackerj	28	004-07-5839	_______	34847.84	3037.986667	2	...	
9	0x160f	CUS_0x21b1	February	Rick Rothackerj	28	004-07-5839	Teacher	34847.84	3037.986667	2	...	

10 rows × 27 columns

```
data.isna().sum()
```

```
ID                              0
Customer_ID                     0
Month                           0
Name                            0
Age                             0
SSN                             0
Occupation                      0
Annual_Income                   0
Monthly_Inhand_Salary           0
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
```

SSN also have some discripancy in its value so we can treat it using mode replacement

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

Now Treating Type of Loan which have 11408 null values

```
data[['Customer_ID','Name','Month','Type_of_Loan']].head(45)
```

	Customer_ID	Name	Month	Type_of_Loan
0	CUS_0xd40	Aaron Maashoh	January	Auto Loan, Credit-Builder Loan, Personal Loan,...
1	CUS_0xd40	Aaron Maashoh	February	Auto Loan, Credit-Builder Loan, Personal Loan,...
2	CUS_0xd40	Aaron Maashoh	March	Auto Loan, Credit-Builder Loan, Personal Loan,...
3	CUS_0xd40	Aaron Maashoh	April	Auto Loan, Credit-Builder Loan, Personal Loan,...
4	CUS_0xd40	Aaron Maashoh	May	Auto Loan, Credit-Builder Loan, Personal Loan,...
5	CUS_0xd40	Aaron Maashoh	June	Auto Loan, Credit-Builder Loan, Personal Loan,...
6	CUS_0xd40	Aaron Maashoh	July	Auto Loan, Credit-Builder Loan, Personal Loan,...
7	CUS_0xd40	Aaron Maashoh	August	Auto Loan, Credit-Builder Loan, Personal Loan,...
8	CUS_0x21b1	Rick Rothackerj	January	Credit-Builder Loan
9	CUS_0x21b1	Rick Rothackerj	February	Credit-Builder Loan
10	CUS_0x21b1	Rick Rothackerj	March	Credit-Builder Loan
11	CUS_0x21b1	Rick Rothackerj	April	Credit-Builder Loan
12	CUS_0x21b1	Rick Rothackerj	May	Credit-Builder Loan
13	CUS_0x21b1	Rick Rothackerj	June	Credit-Builder Loan
14	CUS_0x21b1	Rick Rothackerj	July	Credit-Builder Loan
15	CUS_0x21b1	Rick Rothackerj	August	Credit-Builder Loan
16	CUS_0x2dbc	Langep	January	Auto Loan, Auto Loan, and Not Specified
17	CUS_0x2dbc	Langep	February	Auto Loan, Auto Loan, and Not Specified
18	CUS_0x2dbc	Langep	March	Auto Loan, Auto Loan, and Not Specified
19	CUS_0x2dbc	Langep	April	Auto Loan, Auto Loan, and Not Specified
20	CUS_0x2dbc	Langep	May	Auto Loan, Auto Loan, and Not Specified
21	CUS_0x2dbc	Langep	June	Auto Loan, Auto Loan, and Not Specified
22	CUS_0x2dbc	Langep	July	Auto Loan, Auto Loan, and Not Specified
23	CUS_0x2dbc	Langep	August	Auto Loan, Auto Loan, and Not Specified
24	CUS_0xb891	Jasond	January	Not Specified
25	CUS_0xb891	Jasond	February	Not Specified
26	CUS_0xb891	Jasond	March	Not Specified
27	CUS_0xb891	Jasond	April	Not Specified
28	CUS_0xb891	Jasond	May	Not Specified
29	CUS_0xb891	Jasond	June	Not Specified
30	CUS_0xb891	Jasond	July	Not Specified
31	CUS_0xb891	Jasond	August	Not Specified
32	CUS_0x1cdb	Deepaa	January	NaN
33	CUS_0x1cdb	Deepaa	February	NaN
34	CUS_0x1cdb	Deepaa	March	NaN
35	CUS_0x1cdb	Deepaa	April	NaN
36	CUS_0x1cdb	Deepaa	May	NaN
37	CUS_0x1cdb	Deepaa	June	NaN
38	CUS_0x1cdb	Deepaa	July	NaN
39	CUS_0x1cdb	Deepaa	August	NaN
40	CUS_0x95ee	Np	January	NaN
41	CUS_0x95ee	Np	February	NaN
42	CUS_0x95ee	Np	March	NaN
43	CUS_0x95ee	Np	April	NaN
44	CUS_0x95ee	Np	May	NaN

```
data.groupby(['Customer_ID','Name'])['Type_of_Loan'].apply(lambda x: x.isnull().all()).sum()
```

```
1426
```

from above we can see there are 1426 Customers who have no entry for their loan type so we can't do anything for these entry and leave it as it is but we can replace it with entries which are not null

```
data['Type_of_Loan']= data.groupby('Customer_ID')['Type_of_Loan'].transform(lambda x: x.fillna(x.mode().str[0]))
```

```
data.groupby(['Customer_ID','Name'])['Type_of_Loan']
```

```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x7e1607014cd0>
```

```
data.Customer_ID.nunique()
```

```
12500
```

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

```
       '2609', '4326', '4211', '823', '3011', '1608', '2860', '4219',
       '4047', '1531', '742', '52', '4024', '1673', '49', '2243', '1685',
       '1869', '2587', '3489', '749', '1164', '2616', '848_', '4134',
       '1530', '1502', '4075', '3845', '1060', '2573', '2128', '328',
       '640', '2585', '2230', '1795', '1180', '1534', '3739', '3313',
       '4191', '996', '372', '3340', '3177', '602', '787', '4135', '3878',
       '4059', '1218', '4051', '1766', '1359', '3107', '585', '1263',
       '2511', '709', '3632', '4077', '2943', '2793', '3245', '2317',
       '1640', '2237_', '3819', '252', '3978', '1498', '1833', '2737',
       '1192', '1481', '700', '271', '2286', '273', '1215', '3944',
       '2070', '1478', '3749', '871', '2508', '2959', '130', '294',
       '3097_', '3511', '415', '2196', '2138', '2149', '1874', '1553',
       '3847', '3222', '1222', '2907', '3051', '98', '1598', '416',
       '2314', '2955', '1691', '1450', '2021', '1636', '80', '3708',
       '195', '320', '2945', '1911', '3416', '3796', '4159', '2255',
       '938', '4397', '3776', '2148', '1994', '853', '1178', '1633',
       '196', '3864', '714', '1687', '1034', '468', '1337', '2044',
       '1541', '3661', '1211', '2645', '2007', '102', '1891', '3162',
       '3142', '2566_', '2766', '3881', '2728', '2671', '1952', '3580',
       '2705', '4251', '3840_', '972', '3119', '3502', '4185', '2954',
       '683', '1614', '1572', '4302', '3447', '1852', '2131', '1900',
       '1699', '133', '2018', '2127', '508', '210', '577', '1664', '2604',
       '1411', '2351', '867', '1371', '2352', '1191', '905', '4053',
       '3869', '933', '3660', '3300', '3629', '3208', '2142', '2521',
       '450', '583', '876', '121', '3919', '2560', '2578', '2060', '813',
       '1236', '1489', '4360', '1154', '2544', '4172', '2924', '426',
       '4270', '2768', '3909', '3951', '2712', '2498', '3171', '1750',
       '197', '2569', '265', '4293', '887', '2707', '2397', '4337',
       '4249', '2751', '2950', '1859', '107', '2348', '2506', '2810',
       '2873', '1301', '2262', '1890', '3078', '3865', '3268', '2777',
       '3105', '1278', '3793', '2276', '2879', '4298', '2141', '223',
       '2239', '846', '1862', '2756', '1181', '1184', '2617', '3972',
       '2334', '3900', '2759', '4169', '2280', '2492', '2729', '3750',
       '1825', '309', '2431', '3099', '2080', '2279', '2666', '3722',
       '1976', '529', '1985', '3060', '4278', '3212', '46', '3148',
       '3467', '4231', '3790', '473', '1536', '3955', '2324', '2381',
       '1177', '371', '2896', '3880', '2991', '4319', '1061', '662',
       '4144', '693', '2006', '3115', '2278_', '3751', '1861', '4262',
       '2913', '2615', '3492', '800', '3766', '384', '3407', '1087',
       '3329', '1086', '2216', '1087_', '2457', '3522', '3274', '3488',
       '2854', '238', '351', '3706', '4280', '4095', '2926', '1329',
       '3370', '283', '1392', '1743', '2429', '974', '3156', '1133',
       '4388', '3243', '4282', '2523', '4281', '3415', '2001', '441',
       '94', '3499', '969', '3368', '106', '1004', '2638', '3946', '2956',
       '4324', '85', '4113', '819', '615', '1172', '2553', '1765', '3495',
       '2820', '4239', '4340', '1295_', '2636', '4295', '1653', '1325',
       '1879', '1096', '1735', '3584', '1073', '1975', '3827', '2552',
       '3754', '2378', '532', '926', '2376', '3636', '3763', '778',
       '2621', '804', '754', '2418', '4019', '3926', '3861_', '3574',
       '175', '162', '2834', '3765', '2354', '523', '2274', '1606',
       '1443', '1354', '2142_', '1422', '2278', '1045', '4106', '3155',
       '666', '659', '3229', '1216', '2076', '1473_', '2384', '1954',
       '719', '2534', '4002', '541', '2875', '4344', '2081', '3894',
       '1256', '676', '4178', '399', '86', '1571', '4037', '1967', '4005',
       '3216', '1150', '2591', '1801', '3721', '1775', '2260', '3707',
       '4292', '1820', '145', '1480', '1850', '430', '217', '3920_',
       '1389', '1579', '3391', '2385', '3336', '3392', '3688', '221',
       '2047'], dtype=object)
```

for dealing with "_" we should do below steps

```
data.loc[data['Num_of_Delayed_Payment'].str[-1] == '_','Num_of_Delayed_Payment'] = data['Num_of_Delayed_Payment'].str[:-1]
```

for dealing with "-" or negative numbers as average number of payments delayed by a person so it can't be a negative number we will do below steps

```
data.loc[data['Num_of_Delayed_Payment'].str[0] == '-','Num_of_Delayed_Payment'] = data['Num_of_Delayed_Payment'].str[1:]
```

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

```
       '3533', '519', '2677', '2413', '4139', '2609', '4326', '4211',
       '823', '3011', '1608', '2860', '4219', '4047', '1531', '742', '52',
       '4024', '1673', '49', '2243', '1685', '1869', '2587', '3489',
       '749', '1164', '2616', '848', '4134', '1530', '1502', '4075',
       '3845', '1060', '2573', '2128', '328', '640', '2585', '2230',
       '1795', '1180', '1534', '3739', '3313', '4191', '996', '372',
       '3340', '3177', '602', '787', '4135', '3878', '4059', '1218',
       '4051', '1766', '1359', '3107', '585', '1263', '2511', '709',
       '3632', '4077', '2943', '2793', '3245', '2317', '1640', '2237',
       '3819', '252', '3978', '1498', '1833', '2737', '1192', '1481',
       '700', '271', '2286', '273', '1215', '3944', '2070', '1478',
       '3749', '871', '2508', '2959', '130', '294', '3097', '3511', '415',
       '2196', '2138', '2149', '1874', '1553', '3847', '3222', '1222',
       '2907', '3051', '98', '1598', '416', '2314', '2955', '1691',
       '1450', '2021', '1636', '80', '3708', '195', '320', '2945', '1911',
       '3416', '3796', '4159', '2255', '938', '4397', '3776', '2148',
       '1994', '853', '1178', '1633', '196', '3864', '714', '1687',
       '1034', '468', '1337', '2044', '1541', '3661', '1211', '2645',
       '2007', '102', '1891', '3162', '3142', '2766', '3881', '2728',
       '2671', '1952', '3580', '2705', '4251', '3840', '972', '3119',
       '3502', '4185', '2954', '683', '1614', '1572', '4302', '3447',
       '1852', '2131', '1900', '1699', '133', '2018', '2127', '508',
       '210', '577', '1664', '2604', '1411', '2351', '867', '1371',
       '2352', '1191', '905', '4053', '3869', '933', '3660', '3300',
       '3629', '3208', '2142', '2521', '450', '583', '876', '121', '3919',
       '2560', '2578', '2060', '813', '1236', '1489', '4360', '1154',
       '2544', '4172', '2924', '426', '4270', '2768', '3909', '3951',
       '2712', '2498', '3171', '1750', '197', '265', '4293', '887',
       '2707', '2397', '4337', '4249', '2751', '2950', '1859', '107',
       '2348', '2506', '2810', '2873', '1301', '2262', '1890', '3078',
       '3865', '3268', '2777', '3105', '1278', '3793', '2276', '2879',
       '4298', '2141', '223', '2239', '846', '1862', '2756', '1181',
       '1184', '2617', '3972', '2334', '3900', '2759', '4169', '2280',
       '2492', '2729', '3750', '1825', '309', '2431', '3099', '2080',
       '2279', '2666', '3722', '1976', '529', '1985', '3060', '4278',
       '3212', '46', '3148', '3467', '4231', '3790', '473', '1536',
       '3955', '2324', '2381', '1177', '371', '2896', '3880', '2991',
       '4319', '1061', '662', '4144', '693', '2006', '3115', '2278',
       '3751', '1861', '4262', '2913', '2615', '3492', '800', '3766',
       '384', '3407', '1087', '3329', '1086', '2216', '2457', '3522',
       '3274', '3488', '2854', '238', '351', '3706', '4280', '4095',
       '2926', '1329', '3370', '283', '1392', '1743', '2429', '974',
       '3156', '1133', '4388', '3243', '4282', '2523', '4281', '3415',
       '2001', '441', '94', '3499', '969', '3368', '106', '1004', '2638',
       '3946', '2956', '4324', '85', '4113', '819', '615', '1172', '2553',
       '1765', '3495', '2820', '4239', '4340', '1295', '2636', '4295',
       '1653', '1325', '1879', '1096', '1735', '3584', '1073', '1975',
       '3827', '2552', '3754', '2378', '532', '926', '2376', '3636',
       '3763', '778', '2621', '804', '754', '2418', '4019', '3926',
       '3861', '3574', '175', '162', '2834', '3765', '2354', '523',
       '2274', '1606', '1443', '1354', '1422', '1045', '4106', '3155',
       '666', '659', '3229', '1216', '2076', '2384', '1954', '719',
       '2534', '4002', '541', '2875', '4344', '2081', '3894', '1256',
       '676', '4178', '399', '86', '1571', '4037', '1967', '4005', '3216',
       '1150', '2591', '1801', '3721', '1775', '2260', '3707', '4292',
       '1820', '145', '1480', '1850', '430', '217', '3920', '1389',
       '1579', '3391', '2385', '3336', '3392', '3688', '221', '2047'],
      dtype=object)
```

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

```
7002
```

```
data[['Customer_ID','Num_of_Delayed_Payment']].head(30)
```

	Customer_ID	Num_of_Delayed_Payment
0	CUS_0xd40	7
1	CUS_0xd40	NaN
2	CUS_0xd40	7
3	CUS_0xd40	4
4	CUS_0xd40	NaN
5	CUS_0xd40	4
6	CUS_0xd40	8
7	CUS_0xd40	6
8	CUS_0x21b1	4
9	CUS_0x21b1	1
10	CUS_0x21b1	1
11	CUS_0x21b1	3
12	CUS_0x21b1	1

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

```
       '3533', '519', '2677', '2413', '4139', '2609', '4326', '4211',
       '823', '3011', '1608', '2860', '4219', '4047', '1531', '742', '52',
       '4024', '1673', '49', '2243', '1685', '1869', '2587', '3489',
       '749', '1164', '2616', '848', '4134', '1530', '1502', '4075',
       '3845', '1060', '2573', '2128', '328', '640', '2585', '2230',
       '1795', '1180', '1534', '3739', '3313', '4191', '996', '372',
       '3340', '3177', '602', '787', '4135', '3878', '4059', '1218',
       '4051', '1766', '1359', '3107', '585', '1263', '2511', '709',
       '3632', '4077', '2943', '2793', '3245', '2317', '1640', '2237',
       '3819', '252', '3978', '1498', '1833', '2737', '1192', '1481',
       '700', '271', '2286', '273', '1215', '3944', '2070', '1478',
       '3749', '871', '2508', '2959', '130', '294', '3097', '3511', '415',
       '2196', '2138', '2149', '1874', '1553', '3847', '3222', '1222',
       '2907', '3051', '98', '1598', '416', '2314', '2955', '1691',
       '1450', '2021', '1636', '80', '3708', '195', '320', '2945', '1911',
       '3416', '3796', '4159', '2255', '938', '4397', '3776', '2148',
       '1994', '853', '1178', '1633', '196', '3864', '714', '1687',
       '1034', '468', '1337', '2044', '1541', '3661', '1211', '2645',
       '2007', '102', '1891', '3162', '3142', '2766', '3881', '2728',
       '2671', '1952', '3580', '2705', '4251', '3840', '972', '3119',
       '3502', '4185', '2954', '683', '1614', '1572', '4302', '3447',
       '1852', '2131', '1900', '1699', '133', '2018', '2127', '508',
       '210', '577', '1664', '2604', '1411', '2351', '867', '1371',
       '2352', '1191', '905', '4053', '3869', '933', '3660', '3300',
       '3629', '3208', '2142', '2521', '450', '583', '876', '121', '3919',
       '2560', '2578', '2060', '813', '1236', '1489', '4360', '1154',
       '2544', '4172', '2924', '426', '4270', '2768', '3909', '3951',
       '2712', '2498', '3171', '1750', '197', '265', '4293', '887',
       '2707', '2397', '4337', '4249', '2751', '2950', '1859', '107',
       '2348', '2506', '2810', '2873', '1301', '2262', '1890', '3078',
       '3865', '3268', '2777', '3105', '1278', '3793', '2276', '2879',
       '4298', '2141', '223', '2239', '846', '1862', '2756', '1181',
       '1184', '2617', '3972', '2334', '3900', '2759', '4169', '2280',
       '2492', '2729', '3750', '1825', '309', '2431', '3099', '2080',
       '2279', '2666', '3722', '1976', '529', '1985', '3060', '4278',
       '3212', '46', '3148', '3467', '4231', '3790', '473', '1536',
       '3955', '2324', '2381', '1177', '371', '2896', '3880', '2991',
       '4319', '1061', '662', '4144', '693', '2006', '3115', '2278',
       '3751', '1861', '4262', '2913', '2615', '3492', '800', '3766',
       '384', '3407', '1087', '3329', '1086', '2216', '2457', '3522',
       '3274', '3488', '2854', '238', '351', '3706', '4280', '4095',
       '2926', '1329', '3370', '283', '1392', '1743', '2429', '974',
       '3156', '1133', '4388', '3243', '4282', '2523', '4281', '3415',
       '2001', '441', '94', '3499', '969', '3368', '106', '1004', '2638',
       '3946', '2956', '4324', '85', '4113', '819', '615', '1172', '2553',
       '1765', '3495', '2820', '4239', '4340', '1295', '2636', '4295',
       '1653', '1325', '1879', '1096', '1735', '3584', '1073', '1975',
       '3827', '2552', '3754', '2378', '532', '926', '2376', '3636',
       '3763', '778', '2621', '804', '754', '2418', '4019', '3926',
       '3861', '3574', '175', '162', '2834', '3765', '2354', '523',
       '2274', '1606', '1443', '1354', '1422', '1045', '4106', '3155',
       '666', '659', '3229', '1216', '2076', '2384', '1954', '719',
       '2534', '4002', '541', '2875', '4344', '2081', '3894', '1256',
       '676', '4178', '399', '86', '1571', '4037', '1967', '4005', '3216',
       '1150', '2591', '1801', '3721', '1775', '2260', '3707', '4292',
       '1820', '145', '1480', '1850', '430', '217', '3920', '1389',
       '1579', '3391', '2385', '3336', '3392', '3688', '221', '2047'],
      dtype=object)
```

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

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

```
0
```

```
data.isna().sum()
```

```
ID                              0
Customer_ID                     0
Month                           0
Name                            0
Age                             0
SSN                             0
Occupation                      0
Annual_Income                   0
Monthly_Inhand_Salary           0
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          0
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
```

Num_Credit_Inquiries

now dealing with

Num_Credit_Inquiries which Represents the number of credit card inquiries

Num_Credit_Inquiries 1965

[ ] ↳ *6 cells hidden*

Credit History Age

```
data.isna().sum()
```

```
ID                              0
Customer_ID                     0
Month                           0
Name                            0
Age                             0
SSN                             0
Occupation                      0
Annual_Income                   0
Monthly_Inhand_Salary           0
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          0
Changed_Credit_Limit            0
Num_Credit_Inquiries            0
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
```

```
data[['Customer_ID','Month','Name','Occupation','Annual_Income','Num_Credit_Card','Credit_History_Age']].head(25)
```

	Customer_ID	Month	Name	Occupation	Annual_Income	Num_Credit_Card	Credit_History_Age
0	CUS_0xd40	January	Aaron Maashoh	Scientist	19114.12	4	22 Years and 1 Months
1	CUS_0xd40	February	Aaron Maashoh	Scientist	19114.12	4	NaN
2	CUS_0xd40	March	Aaron Maashoh	Scientist	19114.12	4	22 Years and 3 Months
3	CUS_0xd40	April	Aaron Maashoh	Scientist	19114.12	4	22 Years and 4 Months
4	CUS_0xd40	May	Aaron Maashoh	Scientist	19114.12	4	22 Years and 5 Months
5	CUS_0xd40	June	Aaron Maashoh	Scientist	19114.12	4	22 Years and 6 Months
6	CUS_0xd40	July	Aaron Maashoh	Scientist	19114.12	4	22 Years and 7 Months
7	CUS_0xd40	August	Aaron Maashoh	Scientist	19114.12	4	NaN
8	CUS_0x21b1	January	Rick Rothackerj	_______	34847.84	4	26 Years and 7 Months
9	CUS_0x21b1	February	Rick Rothackerj	Teacher	34847.84	4	26 Years and 8 Months
10	CUS_0x21b1	March	Rick Rothackerj	Teacher	34847.84_	1385	26 Years and 9 Months
11	CUS_0x21b1	April	Rick Rothackerj	Teacher	34847.84	4	26 Years and 10 Months
12	CUS_0x21b1	May	Rick Rothackerj	Teacher	34847.84	4	26 Years and 11 Months
13	CUS_0x21b1	June	Rick Rothackerj	Teacher	34847.84	4	27 Years and 0 Months
14	CUS_0x21b1	July	Rick Rothackerj	Teacher	34847.84	4	27 Years and 1 Months

```
data['Credit_History_Age'].str.split(" ").str[-2]
```

```
0         1
1       NaN
2         3
3         4
4         5
       ...
99995     6
99996     7
99997     8
99998     9
99999    10
Name: Credit_History_Age, Length: 100000, dtype: object
```

Amount Invested Monthly

```
data[['Customer_ID','Month','Name','Occupation','Annual_Income','Num_Credit_Card','Amount_invested_monthly']].head(25)
```

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

```
4479
```

```
data[['Customer_ID','Month','Amount_invested_monthly']].head(30)
```

	Customer_ID	Month	Amount_invested_monthly
0	CUS_0xd40	January	80.41529544
1	CUS_0xd40	February	118.2802216
2	CUS_0xd40	March	81.69952126
3	CUS_0xd40	April	199.4580744
4	CUS_0xd40	May	41.42015309
5	CUS_0xd40	June	62.43017233
6	CUS_0xd40	July	178.3440674
7	CUS_0xd40	August	24.78521651
8	CUS_0x21b1	January	104.2918252
9	CUS_0x21b1	February	40.39123783
10	CUS_0x21b1	March	58.5159757
11	CUS_0x21b1	April	99.30622796
12	CUS_0x21b1	May	130.1154202
13	CUS_0x21b1	June	43.47719014
14	CUS_0x21b1	July	70.10177421
15	CUS_0x21b1	August	218.9043435
16	CUS_0x2dbc	January	168.4137027
17	CUS_0x2dbc	February	232.8603838
18	CUS_0x2dbc	March	__10000__
19	CUS_0x2dbc	April	825.2162699
20	CUS_0x2dbc	May	430.9475279
21	CUS_0x2dbc	June	257.8080994
22	CUS_0x2dbc	July	263.1741632
23	CUS_0x2dbc	August	__10000__
24	CUS_0xb891	January	81.22885871
25	CUS_0xb891	February	124.8818199
26	CUS_0xb891	March	83.4065088
27	CUS_0xb891	April	272.3340374
28	CUS_0xb891	May	__10000__
29	CUS_0xb891	June	84.95284817

```
data[data['Amount_invested_monthly'] == '__10000__']['Amount_invested_monthly'].shape
```

```
(4305,)
```

```
data['Amount_invested_monthly'].describe()
```

```
count         95521
unique        91049
top       __10000__
freq           4305
Name: Amount_invested_monthly, dtype: object
```

there are null 4479 null values and 4305 wrong entries in the 'Amount_invested_monthly' column so we have to deal with it seperately using .fillna() and .replace() and both of these would be replaced 1st with -1 then its type should get converted to float or int then -1 is replaced with median as this would be better choice keeping in mind with the outliers and width of entries numeric values for the given data

```
data['Amount_invested_monthly'] = data.groupby('Customer_ID')['Amount_invested_monthly'].transform(lambda x: x.fillna("-1"))
data['Amount_invested_monthly'] = data.groupby('Customer_ID')['Amount_invested_monthly'].transform(lambda x: x.replace("__10000__","-1"))
```

```
data[['Customer_ID','Month','Amount_invested_monthly']].head(30)
```

	Customer_ID	Month	Amount_invested_monthly
0	CUS_0xd40	January	80.41529544
1	CUS_0xd40	February	118.2802216
2	CUS_0xd40	March	81.69952126
3	CUS_0xd40	April	199.4580744
4	CUS_0xd40	May	41.42015309
5	CUS_0xd40	June	62.43017233
6	CUS_0xd40	July	178.3440674
7	CUS_0xd40	August	24.78521651
8	CUS_0x21b1	January	104.2918252
9	CUS_0x21b1	February	40.39123783
10	CUS_0x21b1	March	58.5159757
11	CUS_0x21b1	April	99.30622796
12	CUS_0x21b1	May	130.1154202
13	CUS_0x21b1	June	43.47719014
14	CUS_0x21b1	July	70.10177421
15	CUS_0x21b1	August	218.9043435
16	CUS_0x2dbc	January	168.4137027
17	CUS_0x2dbc	February	232.8603838
18	CUS_0x2dbc	March	-1
19	CUS_0x2dbc	April	825.2162699
20	CUS_0x2dbc	May	430.9475279
21	CUS_0x2dbc	June	257.8080994
22	CUS_0x2dbc	July	263.1741632
23	CUS_0x2dbc	August	-1
24	CUS_0xb891	January	81.22885871
25	CUS_0xb891	February	124.8818199

```
data['Amount_invested_monthly'] = data['Amount_invested_monthly'].astype(float)
```

27 CUS_0xb891 April 272.3340374

Now replacing -1 with the median value of 'Amount_invested_monthly'for same Customer_ID

29 CUS_0xb891 June 84.95284817

```
data['Amount_invested_monthly'] = data.groupby('Customer_ID')['Amount_invested_monthly'].transform(lambda x: x.replace(-1,x.median()))
```

```
data[['Customer_ID','Month','Amount_invested_monthly']].head(30)
```

	Customer_ID	Month	Amount_invested_monthly
0	CUS_0xd40	January	80.415295
1	CUS_0xd40	February	118.280222
2	CUS_0xd40	March	81.699521
3	CUS_0xd40	April	199.458074
4	CUS_0xd40	May	41.420153
5	CUS_0xd40	June	62.430172
6	CUS_0xd40	July	178.344067
7	CUS_0xd40	August	24.785217
8	CUS_0x21b1	January	104.291825

Now all null values and wrong entries has been replaced with median value of Amount_invested_monthly for same Customer_ID as can be check by below process

```
print("number of null values : ",data['Amount_invested_monthly'].isna().sum())
print("number of wrong entries __10000__  : ",data[data['Amount_invested_monthly']=='__10000__']['Amount_invested_monthly'].count())
```

```
number of null values :  0
number of wrong entries __10000__  :  0
```

Monthly Balance

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 27 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   ID                        100000 non-null  object 
 1   Customer_ID               100000 non-null  object 
 2   Month                     100000 non-null  object 
 3   Name                      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  object 
 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  object 
 13  Type_of_Loan              88592 non-null   object 
 14  Delay_from_due_date       100000 non-null  int64  
 15  Num_of_Delayed_Payment    100000 non-null  object 
 16  Changed_Credit_Limit      100000 non-null  object 
 17  Num_Credit_Inquiries      100000 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   100000 non-null  float64
 25  Payment_Behaviour         100000 non-null  object 
 26  Monthly_Balance           98800 non-null   object 
dtypes: float64(5), int64(5), object(17)
memory usage: 20.6+ MB
```

```
data[data.Monthly_Balance.isna()][['Customer_ID','Month','Monthly_Balance']]
```

	Customer_ID	Month	Monthly_Balance
197	CUS_0xa5f9	June	NaN
314	CUS_0x571f	March	NaN
388	CUS_0x9b3c	May	NaN
456	CUS_0x9d78	January	NaN
457	CUS_0x9d78	February	NaN
...	...	...	...
99820	CUS_0x40ad	May	NaN
99839	CUS_0x8788	August	NaN
99852	CUS_0x3048	May	NaN
99854	CUS_0x3048	July	NaN
99927	CUS_0x2654	August	NaN

1200 rows × 3 columns

```
data[['Customer_ID','Monthly_Balance']].iloc[450:470]
```

	Customer_ID	Monthly_Balance
450	CUS_0x3f5b	407.5699589
451	CUS_0x3f5b	803.4807269
452	CUS_0x3f5b	625.2923728
453	CUS_0x3f5b	708.5727909
454	CUS_0x3f5b	799.4042336
455	CUS_0x3f5b	808.782961
456	CUS_0x9d78	NaN
457	CUS_0x9d78	NaN
458	CUS_0x9d78	214.1906424
459	CUS_0x9d78	141.3300654
460	CUS_0x9d78	229.3651659
461	CUS_0x9d78	59.88619327
462	CUS_0x9d78	NaN
463	CUS_0x9d78	NaN
464	CUS_0x47db	515.0833063
465	CUS_0x47db	193.578861
466	CUS_0x47db	327.5495239
467	CUS_0x47db	455.0014209
468	CUS_0x47db	75.19634822
469	CUS_0x47db	233.9059894

inorder to deal with missing value we will use mean value of Monthly_Balance for every Customer_ID

since monthly_balance is string data type so first we will replace null value with 0 for identification then will change the datatype from str to float if some miscellaneous no numeric entity comes up while performing operation e.g. **-333333333333333333333333333** then will replace same with 0 then again will replace 0 with the mean of monthly_balance for every customer_ID

```
data['Monthly_Balance'] = data.groupby('Customer_ID')['Monthly_Balance'].transform(lambda x: x.fillna("0"))
data['Monthly_Balance'] = data.groupby('Customer_ID')['Monthly_Balance'].transform(lambda x: x.replace("__-333333333333333333333333333__","0"))
data['Monthly_Balance'] = data['Monthly_Balance'].astype(float)
data['Monthly_Balance'] = data.groupby('Customer_ID')['Monthly_Balance'].transform(lambda x: x.replace(0,x.mean()))
```

cross checking the missing value imputation

```
data[['Customer_ID','Monthly_Balance']].iloc[450:470]
```

Customer ID Monthly Balance

```
data[['Customer_ID','Monthly_Balance']].iloc[99845:99860]
```

	Customer_ID	Monthly_Balance
99845	CUS_0x944e	302.276968
99846	CUS_0x944e	284.018644
99847	CUS_0x944e	299.233745
99848	CUS_0x3048	381.241606
99849	CUS_0x3048	41.541911
99850	CUS_0x3048	243.341772
99851	CUS_0x3048	380.229816
99852	CUS_0x3048	187.515825
99853	CUS_0x3048	258.156482
99854	CUS_0x3048	187.515825
99855	CUS_0x3048	195.615013
99856	CUS_0x1285	328.089856
99857	CUS_0x1285	322.293263
99858	CUS_0x1285	308.748472
99859	CUS_0x1285	302.667526

```
data.isna().sum()
```

```
ID                              0
Customer_ID                     0
Month                           0
Name                            0
Age                             0
SSN                             0
Occupation                      0
Annual_Income                   0
Monthly_Inhand_Salary           0
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          0
Changed_Credit_Limit            0
Num_Credit_Inquiries            0
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         0
Payment_Behaviour               0
Monthly_Balance                 0
dtype: int64
```

Now its time to check every features in data for some wrong entries or changing its data type if required

```
data.Month.unique()
```

```
array(['January', 'February', 'March', 'April', 'May', 'June', 'July',
       'August'], dtype=object)
```

```
data.Name.unique()
```

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

```
data.Name.nunique()
```

```
10139
```

```
np.dtype(data.Age)
```

```
dtype('int64')
```

```
data.Age.unique()
```

```
array([23, 28, 34, 55, 21, 31, 30, 44, 40, 33, 35, 39, 37, 20, 46, 26, 41,
       32, 48, 43, 36, 16, 18, 42, 22, 19, 15, 27, 38, 14, 25, 45, 47, 17,
```

```
       53, 24, 54, 29, 49, 51, 50, 52, 56])
```

```
data.SSN.unique()
```

```
array(['821-00-0265', '004-07-5839', '486-85-3974', ..., '133-16-7738',
       '031-35-0942', '078-73-5990'], dtype=object)
```

```
data.Occupation.unique()
```

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

for Occupation column there is wrong etry we can see from above unique element so in order to deal with this we will use mode of occupation column for every Customer_ID

```
data[['Customer_ID','Name','Occupation']].head(50)
```

	Customer_ID	Name	Occupation
0	CUS_0xd40	Aaron Maashoh	Scientist
1	CUS_0xd40	Aaron Maashoh	Scientist
2	CUS_0xd40	Aaron Maashoh	Scientist
3	CUS_0xd40	Aaron Maashoh	Scientist
4	CUS_0xd40	Aaron Maashoh	Scientist
5	CUS_0xd40	Aaron Maashoh	Scientist
6	CUS_0xd40	Aaron Maashoh	Scientist
7	CUS_0xd40	Aaron Maashoh	Scientist

```
data['Occupation'] = data.groupby('Customer_ID')['Occupation'].transform(lambda x: x.replace("_______",x.mode().iloc[0]))
```

9	CUS_0x21b1	Rick Rothackeri	Teacher

```
data[['Customer_ID','Name','Occupation']].head(50)
```

	Customer_ID	Name	Occupation
0	CUS_0xd40	Aaron Maashoh	Scientist
1	CUS_0xd40	Aaron Maashoh	Scientist
2	CUS_0xd40	Aaron Maashoh	Scientist
3	CUS_0xd40	Aaron Maashoh	Scientist
4	CUS_0xd40	Aaron Maashoh	Scientist
5	CUS_0xd40	Aaron Maashoh	Scientist
6	CUS_0xd40	Aaron Maashoh	Scientist

from above process "___" is replaced with the mode of Occupation column for every Customer_ID

8 CUS_0x21b1 Rick Rothackerj Teacher

Annual Income

```
data.Annual_Income.isna().sum()
```

```
0
```

```
data.Annual_Income.loc[25:50]
```

```
25     30689.89
26     30689.89
27    30689.89_
28     30689.89
29     30689.89
30     30689.89
31     30689.89
32    35547.71_
33     35547.71
34     35547.71
35     35547.71
36     35547.71
37     35547.71
38     35547.71
39     35547.71
40     73928.46
41     73928.46
42     73928.46
43     73928.46
44     73928.46
45     73928.46
46     73928.46
47     73928.46
48     131313.4
49     131313.4
50     131313.4
Name: Annual_Income, dtype: object
```

```
data.Annual_Income.unique()
```

```
array(['19114.12', '34847.84', '34847.84_', ..., '20002.88', '39628.99',
       '39628.99_'], dtype=object)
```

34 CUS_0x1cdb Deepaa Developer

since there are annual incomes where the we can see froma bove result that the they are in string format as because they are in this format 34847.84_ or 39628.99_ so inorder to deal with this we can do the process as shown below

```
data.loc[data['Annual_Income'].str[-1] == '_','Annual_Income'] = data['Annual_Income'].str[:-1]
data.Annual_Income = data.Annual_Income.astype(float)
data.Annual_Income.unique()
```

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

41 CUS_0x95ee Np Lawyer

Monthly_Inhand_Salary

```
data.Monthly_Inhand_Salary.unique()
```

```
array([ 1824.843333,  3037.986667, 12187.22    , ...,  3097.008333,
        1929.906667,  3359.415833])
```

47 CUS_0x95ee Np Lawyer

Num_Bank_Accounts :

Represents the number of bank accounts a person holds

```
data.Num_Bank_Accounts.unique()
```

```
       1368, 1353,  508, 1725, 1413, 1239,  847, 1317,  570,   -1,  302,
        456,  198,  122,  832,  167, 1547, 1666, 1241,  275,  566, 1779,
        201,  334, 1169,  834, 1174, 1040,  530, 1676, 1468, 1093,  706,
        489, 1592,  688,  830, 1784, 1543, 1600, 1178,  228,  483, 1501,
        950,  548,  870, 1211,  604,  804,  129,  540, 1702, 1636, 1680,
       1443, 1252,  499,  180,  702,  885, 1652,  795,  938,  833, 1654,
       1793,  303, 1621, 1516, 1138,   32,  160, 1491,   83,  423,  928,
        339,  931,  243, 1756,  583, 1695,  274,  955,  430, 1247,  490,
        726,  987,   42, 1626, 1470, 1739,  887,  211,  385, 1221,  753,
        324,  406, 1677, 1567,  785,  182, 1079,  184, 1771, 1048, 1069,
        561,  589, 1634,   70, 1371,  647, 1153,  239,  801, 1279, 1287,
        425, 1589,  929, 1511, 1765, 1047, 1005, 1337,  981, 1766,  805,
       1574, 1638,  186,   99,  288, 1650,  974,  996, 1595, 1594,  865,
        203, 1440,  448,  285,   94,  875,  916, 1733,  240,  330,   79,
         82,  135, 1043,  142, 1235, 1569, 1741, 1461,  560, 1551,  409,
        418, 1017,  892,  354,  124,  935,  313, 1363,  232, 1200, 1184,
       1432, 1479, 1407, 1080, 1719, 1024,  970,  761,  158,  312, 1060,
        684,  696, 1520, 1352, 1502,  936,  485,  350, 1560, 1166, 1013,
         34, 1691,  715, 1570, 1751, 1503, 1194, 1558,  411,  298,  260,
        308,  796, 1442,  912, 1256,  746,  435,  620, 1774, 1181, 1323,
        837,  307, 1430,   53, 1447,  259,  921,  328, 1034,  353,  822,
        654,  829,  609,  166,  136,  172, 1306, 1028,  808,  270, 1072,
        514,  156, 1331, 1630,  462, 1310, 1210,  265,   50, 1355,  394,
       1393,  226,  991,  368, 1018, 1037,  627, 1744,  523,  894,  944,
       1730, 1076, 1094, 1617,  449,  678, 1164,   30,  817, 1568,  607,
        709, 1230,  606, 1480,   29, 1522, 1670,   49,  626, 1641, 1712,
       1370, 1782,  119, 1137, 1277,   35,  947,  851, 1041, 1583, 1536,
        575,  196,  143,   33,  383,   57,  621,  230,  162,  352, 1504,
       1250,   39,  511, 1364,  918, 1665,  734, 1320,  316,  771,  705,
       1435,  304,  116, 1553,  424,  493, 1530, 1195, 1604,  624, 1599,
        782, 1606, 1622, 1285,  657, 1275, 1291, 1402,  420,  932,  468,
        442, 1145,   93, 1218,  283,  292,  927, 1711, 1422,  364, 1398,
        594,  789, 1350,  676,  216, 1012, 1426,  717, 1095, 1786,  850,
        777,  375,  356,  701, 1325,  290, 1775,  467, 1365, 1549,  848,
       1453,   84, 1458,  888,  770, 1591,  582,   40, 1261, 1764, 1067,
        145,  471,  476, 1128,  967,  622,  641,  539,  858,  295, 1395,
       1517, 1297, 1126,  926,  157, 1377,  993,  125,  340,  271, 1669,
        272,  979,  707,  564,  997,  447, 1578, 1465,  415,  714, 1276,
         18, 1316,  587,  245,  446,  978, 1002, 1631,  341, 1307,  524,
       1267,  103,  242, 1378, 1328, 1294, 1309,  724,  197,  999,  547,
       1100, 1456,   77, 1506, 1472,  460,   11, 1420,  218, 1253,  392,
        670, 1321,  100, 1205,  443,   92,  811,  774,  159, 1770,  972,
       1157, 1102,  217,  358, 1473,  440,  140, 1361,  481,  512, 1117,
        300, 1263, 1581,  901, 1298,  310,  637,  758, 1734, 1288,  326,
       1678, 1444, 1344, 1379,  828,  982,  104,  317, 1529, 1627,  115,
        645, 1404,  940, 1305, 1049, 1645, 1466,  175, 1165, 1419, 1031,
        581, 1580,  854,  725, 1655, 1134,  569, 1387, 1381, 1760,  360,
       1735, 1146, 1411,  969, 1281,  591,  327,  466,  667,  147,  109,
        784, 1396, 1349,  839, 1284,  802,   97,  281, 1382, 1039, 1483,
        376, 1249, 1207, 1213, 1219, 1345, 1523,  949, 1063,  983,  886,
       1151,  992, 1107, 1314,  151,   69, 1489,  695,  738,  444, 1525,
        825,  506,  518, 1077, 1354,  713,  690,  861, 1389,  968,  421,
        907,  205,  856,   72,  651,  161,  867,  971, 1078,   64, 1701,
        546, 1216,   27, 1709,  193, 1528,  957,  577,  346, 1416,  396,
       1182,  652, 1083, 1778,  680, 1754, 1544, 1703,  636,  472,  453,
        463,   75,  756,  296,  891,  813,  474,  697])
```

from the above inquiry we can se it is not right to say that a person have number of bank accounts more than 1000 or more than double digits so we can conclude them as wrong entry, so to deal with it we can use mode operation

```
data[data['Num_Bank_Accounts'] > 1000 ][['Customer_ID','Num_Bank_Accounts']]
```

	Customer_ID	Num_Bank_Accounts
267	CUS_0x4004	1414
288	CUS_0x4080	1231
356	CUS_0xaedb	1488
1057	CUS_0x1e9b	1647
1122	CUS_0x6749	1696
...	...	...
98735	CUS_0xdcc	1083
98749	CUS_0x50c4	1617
98796	CUS_0xa756	1511
99417	CUS_0xdfd	1525
99638	CUS_0x296f	1481

576 rows × 2 columns

```
data.Num_Bank_Accounts.iloc[264:273]
```

```
264       8
265       8
266       8
267    1414
268       8
269       8
270       8
271       8
```

```
272      2
Name: Num_Bank_Accounts, dtype: int64
```

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

```
data.Num_Bank_Accounts.unique()
```

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

since all Num_Bank_Accounts comes in single digits which sounds good but still there is negative value which needed to be treated seperately as shown below

```
data[data['Num_Bank_Accounts']== -1][['Customer_ID','Num_Bank_Accounts']]
```

	Customer_ID	Num_Bank_Accounts
30328	CUS_0x4f2a	-1
30329	CUS_0x4f2a	-1
30330	CUS_0x4f2a	-1
30331	CUS_0x4f2a	-1
30332	CUS_0x4f2a	-1
30333	CUS_0x4f2a	-1
30334	CUS_0x4f2a	-1
30335	CUS_0x4f2a	-1
43688	CUS_0xa878	-1
43689	CUS_0xa878	-1
43690	CUS_0xa878	-1
43691	CUS_0xa878	-1
43692	CUS_0xa878	-1
43693	CUS_0xa878	-1
43694	CUS_0xa878	-1
43695	CUS_0xa878	-1
47208	CUS_0x43bc	-1
47209	CUS_0x43bc	-1
47210	CUS_0x43bc	-1
47211	CUS_0x43bc	-1
47212	CUS_0x43bc	-1
47213	CUS_0x43bc	-1
47214	CUS_0x43bc	-1
47215	CUS_0x43bc	-1
55632	CUS_0x5993	-1
55633	CUS_0x5993	-1
55634	CUS_0x5993	-1
55635	CUS_0x5993	-1
55636	CUS_0x5993	-1
55637	CUS_0x5993	-1
55638	CUS_0x5993	-1
55639	CUS_0x5993	-1

From above observation we can say that mostly the customers which have 'Num_Bank_Accounts' as -1 they have all their entries as -1 for 'Num_Bank_Accounts' so inorder to deal with it we have to consider some other columns lets say some demographic characters like Age , Occupation and using similar in these beahviour their 'Num_Bank_Accounts' to impute this -1 value in 'Num_Bank_Accounts'

```
data[['Customer_ID','Num_Bank_Accounts']][30325:30340]
```

	Customer_ID	Num_Bank_Accounts
30325	CUS_0x510d	7
30326	CUS_0x510d	7
30327	CUS_0x510d	7
30328	CUS_0x4f2a	-1
30329	CUS_0x4f2a	-1
30330	CUS_0x4f2a	-1
30331	CUS_0x4f2a	-1
30332	CUS_0x4f2a	-1
30333	CUS_0x4f2a	-1

```
data['Num_Bank_Accounts'] = data.groupby(['Age','Occupation'])['Num_Bank_Accounts'].transform(lambda x: x.replace(-1,x.mode().iloc[0]))
```

```
data[['Customer_ID','Num_Bank_Accounts']][30325:30340]
```

	Customer_ID	Num_Bank_Accounts
30325	CUS_0x510d	7
30326	CUS_0x510d	7
30327	CUS_0x510d	7
30328	CUS_0x4f2a	3
30329	CUS_0x4f2a	3
30330	CUS_0x4f2a	3
30331	CUS_0x4f2a	3
30332	CUS_0x4f2a	3
30333	CUS_0x4f2a	3
30334	CUS_0x4f2a	3
30335	CUS_0x4f2a	3
30336	CUS_0xb7d4	1
30337	CUS_0xb7d4	1
30338	CUS_0xb7d4	1
30339	CUS_0xb7d4	1

since still -1 is left so for that we can us mode of 'Num_Bank_Accounts' for every 'Customer_ID'

```
data['Num_Bank_Accounts'] = data.groupby('Customer_ID')['Num_Bank_Accounts'].transform(lambda x: x.replace(-1,x.mode().iloc[0]))
```

```
data.isna().sum()
```

```
ID                              0
Customer_ID                     0
Month                           0
Name                            0
Age                             0
SSN                             0
Occupation                      0
Annual_Income                   0
Monthly_Inhand_Salary           0
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          0
Changed_Credit_Limit            0
Num_Credit_Inquiries            0
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         0
Payment_Behaviour               0
Monthly_Balance                 0
dtype: int64
```

Num_Credit_Card

```
data.Num_Credit_Card.unique()
```

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

from above observation we can see there are individuals who owns more than 1000 credit cards which are not possible legally so this Num_Credit_Card column should be treated accordingly as shown below

```
data[data['Num_Credit_Card'] > 1000 ][['Customer_ID','Num_Credit_Card']].shape[0]
```

```
774
```

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

```
data.Num_Credit_Card.unique()
```

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

```
data[data['Num_Credit_Card'] > 1000 ][['Customer_ID','Num_Credit_Card']].shape[0]
```

```
0
```

Now from above process the wrong entries for Num_Credit_Card where Num_Credit_Card > 1000 has been treated

Interest_Rate

```
data.Interest_Rate.unique()
```

```
array([   3,    6,    8, ..., 1347,  387, 5729])
```

```
data[data['Interest_Rate'] > 1000 ][['Customer_ID','Interest_Rate']]
```

	Customer_ID	Interest_Rate
44	CUS_0x95ee	5318
167	CUS_0x132f	5240
178	CUS_0xac86	4975
345	CUS_0xc65	1138
472	CUS_0x8f17	5261
...	...	...
99621	CUS_0xae66	2536
99753	CUS_0x4a8f	1127
99791	CUS_0x62f5	4396
99882	CUS_0x47fa	1947
99997	CUS_0x942c	5729

1681 rows × 2 columns

```
data[['Customer_ID','Interest_Rate']][165:180]
```

	Customer_ID	Interest_Rate
165	CUS_0x132f	17
166	CUS_0x132f	17
167	CUS_0x132f	5240
168	CUS_0xa16e	17
169	CUS_0xa16e	17
170	CUS_0xa16e	17
171	CUS_0xa16e	17
172	CUS_0xa16e	17
173	CUS_0xa16e	17
174	CUS_0xa16e	17
175	CUS_0xa16e	17
176	CUS_0xac86	1
177	CUS_0xac86	1
178	CUS_0xac86	4975
179	CUS_0xac86	1

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

```
data['Interest_Rate'].unique()
```

```
array([ 3,  6,  8,  4,  5, 15,  7, 12, 20,  1, 14, 32, 16, 17, 10, 31, 25,
       18, 19,  9, 24, 13, 33, 11, 21, 29, 28, 30, 23, 34,  2, 27, 26, 22])
```

from above process we can see that the interest rate column is now good to be used for further analysis

Num_of_Loan

```
data.Num_of_Loan.unique()
```

```
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 this above Num_of_Loan column there are two type of error first the some entries are non numeric in representation like "1132_" and some are > 1000 in values sow we have to deal them for this we can do as below process shown

```
data.loc[data['Num_of_Loan'].str[-1] == '_','Num_of_Loan'] = data['Num_of_Loan'].str[:-1]
data.Num_of_Loan = data.Num_of_Loan.astype(int)
data.Num_of_Loan.unique()
```

```
array([   4,    1,    3,  967, -100,    0,    2,    7,    5,    6,    8,
          9, 1464,  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,  292, 1225, 1227,
        639,  859,  243,  267,  510,  332,  996,  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, 1412,  785, 1127,  910,  538,  999,  733,  101,
        237,   87,  659,  633,  387,  447,  629,  831, 1384,  773,  621,
       1419,  289,  285, 1393,   27, 1359, 1482, 1189, 1294,  201,  579,
        814,  141,  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,  242,  415,  869,  958,   54, 1265,  656,  275,  778,
        208,  147,  350,  507,  463,  497,  927,  653,  662,  529,  635,
       1027,  897, 1039, 1345,  924, 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])
```

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

```
data.Num_of_Loan.unique()
```

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

now the data have been cleaning for Num_of_Loan variable in data as can be checked from above result

Type_of_Loan

```
data.columns
```

```
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'],
      dtype='object')
```

```
data[data.Type_of_Loan.isna()][['Customer_ID','Type_of_Loan']]
```

	Customer_ID	Type_of_Loan
32	CUS_0x1cdb	NaN
33	CUS_0x1cdb	NaN
34	CUS_0x1cdb	NaN
35	CUS_0x1cdb	NaN
36	CUS_0x1cdb	NaN
...	...	...
99939	CUS_0xad4f	NaN
99940	CUS_0xad4f	NaN
99941	CUS_0xad4f	NaN
99942	CUS_0xad4f	NaN
99943	CUS_0xad4f	NaN

11408 rows × 2 columns

to deal with these null values we can take reference of other customers demographic behaviour and banking behaviour or characteristics to predict the type of loans of customers which have null entries in it so we can take columns like 'Age', 'Occupation'.

```
data['Type_of_Loan'] = data.groupby(['Age','Occupation'])['Type_of_Loan'].transform(lambda x: x.fillna(x.value_counts().idxmax()))
```

```
data[data['Occupation']=='Developer'][['Customer_ID','Age','Occupation','Num_of_Loan','Type_of_Loan']]['Type_of_Loan'].unique()
```

```
        'Home Equity Loan, Not Specified, Home Equity Loan, Debt Consolidation Loan, Mortgage Loan, and Debt Consolidation Loan',
        'Credit-Builder Loan, Auto Loan, Student Loan, and Payday Loan',
        'Home Equity Loan, Student Loan, and Credit-Builder Loan',
        'Debt Consolidation Loan, Personal Loan, Home Equity Loan, Debt Consolidation Loan, and Student Loan',
        'Payday Loan, Mortgage Loan, Auto Loan, Credit-Builder Loan, Credit-Builder Loan, Debt Consolidation Loan, and Home Equity Loan',
        'Mortgage Loan, Personal Loan, and Debt Consolidation Loan',
        'Personal Loan, Mortgage Loan, Mortgage Loan, Home Equity Loan, Credit-Builder Loan, Auto Loan, and Not Specified',
        'Credit-Builder Loan, Credit-Builder Loan, Student Loan, Home Equity Loan, Student Loan, Personal Loan, and Personal Loan',
        'Debt Consolidation Loan, Auto Loan, Not Specified, Personal Loan, Personal Loan, Payday Loan, Student Loan, Student Loan, and Debt
Consolidation Loan',
        'Debt Consolidation Loan, Mortgage Loan, Not Specified, Credit-Builder Loan, Student Loan, and Not Specified',
        'Payday Loan, Payday Loan, Debt Consolidation Loan, Credit-Builder Loan, Not Specified, Personal Loan, Home Equity Loan, and Personal
Loan',
        'Payday Loan, and Payday Loan',
        'Credit-Builder Loan, Credit-Builder Loan, Auto Loan, and Mortgage Loan',
        'Payday Loan, Student Loan, Not Specified, Not Specified, Mortgage Loan, Personal Loan, and Not Specified',
        'Debt Consolidation Loan, Not Specified, Student Loan, Personal Loan, Not Specified, and Personal Loan',
        'Personal Loan, Not Specified, Student Loan, Personal Loan, Home Equity Loan, and Mortgage Loan',
        'Not Specified, Not Specified, Mortgage Loan, Personal Loan, Home Equity Loan, Not Specified, and Credit-Builder Loan',
        'Mortgage Loan, Personal Loan, Mortgage Loan, Home Equity Loan, Auto Loan, Home Equity Loan, Auto Loan, and Payday Loan',
        'Debt Consolidation Loan, Personal Loan, and Mortgage Loan',
        'Credit-Builder Loan, Home Equity Loan, and Mortgage Loan',
        'Not Specified, and Student Loan',
        'Home Equity Loan, Not Specified, Not Specified, Debt Consolidation Loan, and Mortgage Loan',
        'Auto Loan, Mortgage Loan, Home Equity Loan, and Debt Consolidation Loan',
        'Payday Loan, Mortgage Loan, Home Equity Loan, Home Equity Loan, Credit-Builder Loan, Not Specified, and Personal Loan',
        'Student Loan, Credit-Builder Loan, Student Loan, and Debt Consolidation Loan',
        'Personal Loan, Credit-Builder Loan, Credit-Builder Loan, and Student Loan',
        'Credit-Builder Loan, Not Specified, and Mortgage Loan',
        'Not Specified, Mortgage Loan, Credit-Builder Loan, Payday Loan, Payday Loan, and Home Equity Loan',
        'Debt Consolidation Loan, and Payday Loan',
        'Not Specified, Student Loan, Debt Consolidation Loan, Home Equity Loan, and Mortgage Loan',
        'Personal Loan, Not Specified, Student Loan, Personal Loan, Debt Consolidation Loan, Debt Consolidation Loan, and Mortgage Loan',
        'Personal Loan, Student Loan, Debt Consolidation Loan, Debt Consolidation Loan, Student Loan, Debt Consolidation Loan, Student Loan,
Personal Loan, and Credit-Builder Loan',
        'Student Loan, Debt Consolidation Loan, Auto Loan, Debt Consolidation Loan, Credit-Builder Loan, Credit-Builder Loan, and Payday Loan',
        'Home Equity Loan, Mortgage Loan, and Payday Loan',
        'Mortgage Loan, Auto Loan, Not Specified, and Home Equity Loan',
        'Credit-Builder Loan, Credit-Builder Loan, Personal Loan, and Debt Consolidation Loan',
        'Student Loan, Mortgage Loan, and Auto Loan',
        'Mortgage Loan, and Credit-Builder Loan',
        'Student Loan, Auto Loan, Payday Loan, Not Specified, Mortgage Loan, Auto Loan, Personal Loan, Debt Consolidation Loan, and Payday Loan',
```

```
data[data.Type_of_Loan.isna()][['Customer_ID','Type_of_Loan']]
```

Customer_ID	Type_of_Loan

```
data[['Customer_ID','Type_of_Loan']].loc[30:45]
```

	Customer_ID	Type_of_Loan
30	CUS_0xb891	Not Specified
31	CUS_0xb891	Not Specified
32	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
33	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
34	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
35	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
36	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
37	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
38	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
39	CUS_0x1cdb	Home Equity Loan, Auto Loan, Personal Loan, Au...
40	CUS_0x95ee	Payday Loan, and Personal Loan
41	CUS_0x95ee	Payday Loan, and Personal Loan
42	CUS_0x95ee	Payday Loan, and Personal Loan
43	CUS_0x95ee	Payday Loan, and Personal Loan
44	CUS_0x95ee	Payday Loan, and Personal Loan
45	CUS_0x95ee	Payday Loan, and Personal Loan

Delay_from_due_date

```
data.Delay_from_due_date.unique()
```

```
array([ 3, -1,  5,  6,  8,  7, 13, 10,  0,  4,  9,  1, 12, 11, 30, 31, 34,
       27, 14,  2, -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, -3, 63,
       40, 37, -5, -4, 66])
```

all values appears to be correct in terms of days

'Num_of_Delayed_Payment'

```
data.Num_of_Delayed_Payment.unique()
```

```
       '3533', '519', '2677', '2413', '4139', '2609', '4326', '4211',
       '823', '3011', '1608', '2860', '4219', '4047', '1531', '742', '52',
       '4024', '1673', '49', '2243', '1685', '1869', '2587', '3489',
       '749', '1164', '2616', '848', '4134', '1530', '1502', '4075',
       '3845', '1060', '2573', '2128', '328', '640', '2585', '2230',
       '1795', '1180', '1534', '3739', '3313', '4191', '996', '372',
       '3340', '3177', '602', '787', '4135', '3878', '4059', '1218',
       '4051', '1766', '1359', '3107', '585', '1263', '2511', '709',
       '3632', '4077', '2943', '2793', '3245', '2317', '1640', '2237',
       '3819', '252', '3978', '1498', '1833', '2737', '1192', '1481',
       '700', '271', '2286', '273', '1215', '3944', '2070', '1478',
       '3749', '871', '2508', '2959', '130', '294', '3097', '3511', '415',
       '2196', '2138', '2149', '1874', '1553', '3847', '3222', '1222',
       '2907', '3051', '98', '1598', '416', '2314', '2955', '1691',
       '1450', '2021', '1636', '80', '3708', '195', '320', '2945', '1911',
       '3416', '3796', '4159', '2255', '938', '4397', '3776', '2148',
       '1994', '853', '1178', '1633', '196', '3864', '714', '1687',
       '1034', '468', '1337', '2044', '1541', '3661', '1211', '2645',
       '2007', '102', '1891', '3162', '3142', '2766', '3881', '2728',
       '2671', '1952', '3580', '2705', '4251', '3840', '972', '3119',
       '3502', '4185', '2954', '683', '1614', '1572', '4302', '3447',
       '1852', '2131', '1900', '1699', '133', '2018', '2127', '508',
       '210', '577', '1664', '2604', '1411', '2351', '867', '1371',
       '2352', '1191', '905', '4053', '3869', '933', '3660', '3300',
       '3629', '3208', '2142', '2521', '450', '583', '876', '121', '3919',
       '2560', '2578', '2060', '813', '1236', '1489', '4360', '1154',
       '2544', '4172', '2924', '426', '4270', '2768', '3909', '3951',
       '2712', '2498', '3171', '1750', '197', '265', '4293', '887',
       '2707', '2397', '4337', '4249', '2751', '2950', '1859', '107',
       '2348', '2506', '2810', '2873', '1301', '2262', '1890', '3078',
       '3865', '3268', '2777', '3105', '1278', '3793', '2276', '2879',
       '4298', '2141', '223', '2239', '846', '1862', '2756', '1181',
       '1184', '2617', '3972', '2334', '3900', '2759', '4169', '2280',
       '2492', '2729', '3750', '1825', '309', '2431', '3099', '2080',
       '2279', '2666', '3722', '1976', '529', '1985', '3060', '4278',
       '3212', '46', '3148', '3467', '4231', '3790', '473', '1536',
       '3955', '2324', '2381', '1177', '371', '2896', '3880', '2991',
       '4319', '1061', '662', '4144', '693', '2006', '3115', '2278',
       '3751', '1861', '4262', '2913', '2615', '3492', '800', '3766',
       '384', '3407', '1087', '3329', '1086', '2216', '2457', '3522',
       '3274', '3488', '2854', '238', '351', '3706', '4280', '4095',
       '2926', '1329', '3370', '283', '1392', '1743', '2429', '974',
       '3156', '1133', '4388', '3243', '4282', '2523', '4281', '3415',
       '2001', '441', '94', '3499', '969', '3368', '106', '1004', '2638',
       '3946', '2956', '4324', '85', '4113', '819', '615', '1172', '2553',
       '1765', '3495', '2820', '4239', '4340', '1295', '2636', '4295',
       '1653', '1325', '1879', '1096', '1735', '3584', '1073', '1975',
       '3827', '2552', '3754', '2378', '532', '926', '2376', '3636',
       '3763', '778', '2621', '804', '754', '2418', '4019', '3926',
       '3861', '3574', '175', '162', '2834', '3765', '2354', '523',
       '2274', '1606', '1443', '1354', '1422', '1045', '4106', '3155',
       '666', '659', '3229', '1216', '2076', '2384', '1954', '719',
       '2534', '4002', '541', '2875', '4344', '2081', '3894', '1256',
       '676', '4178', '399', '86', '1571', '4037', '1967', '4005', '3216',
       '1150', '2591', '1801', '3721', '1775', '2260', '3707', '4292',
       '1820', '145', '1480', '1850', '430', '217', '3920', '1389',
       '1579', '3391', '2385', '3336', '3392', '3688', '221', '2047'],
      dtype=object)
```

```
np.dtype(data.Num_of_Delayed_Payment)
```

```
dtype('O')
```

```
data.Num_of_Delayed_Payment = data.Num_of_Delayed_Payment.astype(int)
```

```
data[data['Num_of_Delayed_Payment']> 1000].Num_of_Delayed_Payment
```

```
252      3318
284      3083
304      1338
409      3104
706      1106
         ...
99069    2385
99133    3336
99402    3392
99562    3688
99825    2047
Name: Num_of_Delayed_Payment, Length: 581, dtype: int64
```

from above analysis we can see there are 581 Num_of_Delayed_Payment were it goes greater than 1000 so inorder to deal with it we can use mode method in imputing where Num_of_Delayed_Payment > 1000 for Num_of_Delayed_Payment for every Customer_ID.

```
data.Num_of_Delayed_Payment.loc[245:280]
```

```
245      7
246      6
247      6
```

```
248      21
249      21
250      21
251      22
252    3318
253      21
254      21
255      18
256      14
257      17
258      20
259      15
260      18
261      17
262      20
263      17
264      17
265      20
266      20
267      23
268      20
269      20
270      20
271      20
272      10
273       8
274      10
275      10
276      10
277      10
278      10
279      10
280       8
Name: Num_of_Delayed_Payment, dtype: int64
```

```
def replace_conditionally(x):
  mode_value = x.mode().iloc[0]
  x.loc[x > mode_value] = mode_value
  return x
data['Num_of_Delayed_Payment'] = data.groupby('Customer_ID')['Num_of_Delayed_Payment'].transform(replace_conditionally)
```

```
data[data['Num_of_Delayed_Payment']> 1000].Num_of_Delayed_Payment.count()
```

```
0
```

```
data.Num_of_Delayed_Payment.loc[245:280]
```

```
245     6
246     6
247     6
248    21
249    21
250    21
251    21
252    21
253    21
254    21
255    18
256    14
257    17
258    17
259    15
260    17
261    17
262    17
263    17
264    17
265    20
266    20
267    20
268    20
269    20
270    20
271    20
272    10
273     8
274    10
275    10
276    10
277    10
278    10
279    10
280     8
Name: Num_of_Delayed_Payment, dtype: int64
```

'Changed_Credit_Limit'

```
data.Changed_Credit_Limit.unique()
```

```
array(['11.27', '_', '6.27', ..., '27.38', '25.16', '21.17'], dtype=object)
```

there is "_" in the 'Changed_Credit_Limit' column for this first we will replace it with 0 then will convert its data type then will replace 0 with the median of 'Changed_Credit_Limit' column for every customer_ID

```
data['Changed_Credit_Limit'] = data.groupby(['Customer_ID'])['Changed_Credit_Limit'].transform(lambda x: x.replace('_',"0"))
```

```
data['Changed_Credit_Limit'] = data['Changed_Credit_Limit'].astype(float)
```

```
data['Changed_Credit_Limit'] = data.groupby(['Customer_ID'])['Changed_Credit_Limit'].transform(lambda x: x.replace(0,x.median()))
```

```
data.Changed_Credit_Limit.unique()
```

```
array([11.27,  6.27,  9.27, ..., 27.38, 25.16, 21.17])
```

'Num_Credit_Inquiries'

```
data.columns
```

```
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'],
      dtype='object')
```

```
data.Num_Credit_Inquiries.unique()
```

```
array([   4.,    2.,    3., ..., 1361.,  310.,   74.])
```

```
data.Num_Credit_Inquiries = data.Num_Credit_Inquiries.astype(int)
```

```
data[data.Num_Credit_Inquiries > 50]['Num_Credit_Inquiries'].idxmin()
```

```
45924
```

```
data[['Customer_ID','Num_Credit_Inquiries']].loc[45915:45930]
```

	Customer_ID	Num_Credit_Inquiries
45915	CUS_0x7ac9	1132
45916	CUS_0x7ac9	4
45917	CUS_0x7ac9	4
45918	CUS_0x7ac9	4
45919	CUS_0x7ac9	4
45920	CUS_0x1c3b	1497
45921	CUS_0x1c3b	1
45922	CUS_0x1c3b	2
45923	CUS_0x1c3b	2
45924	CUS_0x1c3b	52
45925	CUS_0x1c3b	2
45926	CUS_0x1c3b	2
45927	CUS_0x1c3b	2
45928	CUS_0x9b22	3
45929	CUS_0x9b22	3
45930	CUS_0x9b22	3

```
data[['Customer_ID','Num_Credit_Inquiries']].loc[170:200]
```

	Customer_ID	Num_Credit_Inquiries
170	CUS_0xa16e	6
171	CUS_0xa16e	6
172	CUS_0xa16e	6
173	CUS_0xa16e	1050
174	CUS_0xa16e	6
175	CUS_0xa16e	6
176	CUS_0xac86	0
177	CUS_0xac86	0
178	CUS_0xac86	0
179	CUS_0xac86	0
180	CUS_0xac86	0
181	CUS_0xac86	0
182	CUS_0xac86	1
183	CUS_0xac86	1
184	CUS_0x5b48	7
185	CUS_0x5b48	7
186	CUS_0x5b48	7
187	CUS_0x5b48	7
188	CUS_0x5b48	7
189	CUS_0x5b48	7
190	CUS_0x5b48	7
191	CUS_0x5b48	11
192	CUS_0xa5f9	12
193	CUS_0xa5f9	1044
194	CUS_0xa5f9	17

```
def replace_conditionally(x):
  mode_value = x.mode().iloc[0]
  x.loc[x > mode_value] = mode_value
  return x
data['Num_Credit_Inquiries'] = data.groupby('Customer_ID')['Num_Credit_Inquiries'].transform(replace_conditionally)
```

```
data.Num_Credit_Inquiries.unique()
```

```
array([ 4,  2,  3,  5,  8,  6,  0,  1,  7, 12, 17,  9, 10, 11, 14, 16, 15,
       13])
```

Credit_Mix

```
data.Credit_Mix.unique()
```

```
array(['_', 'Good', 'Standard', 'Bad'], dtype=object)
```

from above observation we have "_" as wrong entries so todeal with it we will use mode the mode will be the mode of only non underscore entries of Credit_Mix for every Customer_ID as it is a categorical column.

```
def non_underscore_mode(x):
  non_underscore_values = x[x != '_']
  mode_value = non_underscore_values.mode().iloc[0]
  x[x == '_'] = mode_value
  return x

data['Credit_Mix'] = data.groupby('Customer_ID')['Credit_Mix'].transform(non_underscore_mode)
```

```
data[['Customer_ID','Credit_Mix']].head(20)
```

	Customer_ID	Credit_Mix
0	CUS_0xd40	Good
1	CUS_0xd40	Good
2	CUS_0xd40	Good
3	CUS_0xd40	Good
4	CUS_0xd40	Good
5	CUS_0xd40	Good
6	CUS_0xd40	Good
7	CUS_0xd40	Good
8	CUS_0x21b1	Good
9	CUS_0x21b1	Good
10	CUS_0x21b1	Good
11	CUS_0x21b1	Good
12	CUS_0x21b1	Good

```
data.Credit_Mix.unique()
```

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

Outstanding_Debt

18 CUS_0x2dbc Good

```
data.Outstanding_Debt.unique()
```

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

```
data.loc[data['Outstanding_Debt'].str[-1] == '_','Outstanding_Debt'] = data['Outstanding_Debt'].str[:-1]
data.Outstanding_Debt = data.Outstanding_Debt.astype(float)
data.Outstanding_Debt.unique()
```

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

Credit_Utilization_Ratio

```
data.Credit_Utilization_Ratio.sort_values(ascending= True)
```

```
15860    20.000000
54207    20.100770
3580     20.172942
14319    20.244130
63420    20.257073
           ...
87595    49.064277
62954    49.254983
17029    49.522324
68000    49.564519
9382     50.000000
Name: Credit_Utilization_Ratio, Length: 100000, dtype: float64
```

Credit_History_Age

```
data.Credit_History_Age
```

```
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
```

```
data['Credit_History_Age'].str.split(" ").str[3]
```

```
0          1
1        NaN
2          3
3          4
4          5
        ...
99995      6
```

```
99996      7
99997      8
99998      9
99999     10
Name: Credit_History_Age, Length: 100000, dtype: object
```

as these are categorical values we will convert into number of months to convert it into Credit_History_Age_month

```
data.Credit_History_Age.values
```

```
array(['22 Years and 1 Months', nan, '22 Years and 3 Months', ...,
       '31 Years and 8 Months', '31 Years and 9 Months',
       '31 Years and 10 Months'], dtype=object)
```

```
def char_to_month(x):
  if not pd.isnull(x):
    month = int(x.split(" ")[3])
    year =  int(x.split(" ")[0])
    total_month = (year*12) + month
    return int(total_month)
  else:
    return x
```

```
data['Credit_History_Age'] = data['Credit_History_Age'].apply(lambda x: char_to_month(x)).astype(float)
```

```
data['Credit_History_Age'] = data.groupby('Customer_ID')['Credit_History_Age'].transform(lambda x: x.fillna((x.shift(1) + x.shift(-1)) / 2))
data['Credit_History_Age']
```

```
0        265.0
1        266.0
2        267.0
3        268.0
4        269.0
         ...
99995    378.0
99996    379.0
99997    380.0
99998    381.0
99999    382.0
Name: Credit_History_Age, Length: 100000, dtype: float64
```

Payment_of_Min_Amount

```
data.Payment_of_Min_Amount.unique()
```

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

Total_EMI_per_month

```
data.Total_EMI_per_month.sort_values(ascending =True)
```

```
69229        0.0
52825        0.0
52826        0.0
52827        0.0
52828        0.0
           ...
51614    82193.0
3084     82204.0
29514    82236.0
15300    82256.0
87013    82331.0
Name: Total_EMI_per_month, Length: 100000, dtype: float64
```

Amount_invested_monthly

```
data.Amount_invested_monthly.sort_values(ascending =True)
```

```
84711      -1.000000
73649      -1.000000
73650      -1.000000
73654      -1.000000
20618      -1.000000
             ...
13275    1903.080048
30633    1941.237454
54018    1944.520747
62730    1961.218850
31815    1977.326102
Name: Amount_invested_monthly, Length: 100000, dtype: float64
```

there are some negative values in this column we need to treat them .

```
data[['Customer_ID','Amount_invested_monthly']].loc[73645:73670]
```

	Customer_ID	Amount_invested_monthly
73645	CUS_0x3b1f	50.391471
73646	CUS_0x3b1f	147.303648
73647	CUS_0x3b1f	357.009409
73648	CUS_0xb742	-1.000000
73649	CUS_0xb742	-1.000000
73650	CUS_0xb742	-1.000000
73651	CUS_0xb742	133.530087
73652	CUS_0xb742	123.971219
73653	CUS_0xb742	-1.000000
73654	CUS_0xb742	-1.000000
73655	CUS_0xb742	56.514450
73656	CUS_0xdcd	105.213125
73657	CUS_0xdcd	369.164780
73658	CUS_0xdcd	365.132800
73659	CUS_0xdcd	176.576593
73660	CUS_0xdcd	116.721812
73661	CUS_0xdcd	86.560408
73662	CUS_0xdcd	93.829976
73663	CUS_0xdcd	110.967469
73664	CUS_0x793	139.656620
73665	CUS_0x793	348.431719
73666	CUS_0x793	125.403129
73667	CUS_0x793	115.773774
73668	CUS_0x793	250.548814
73669	CUS_0x793	106.144418
73670	CUS_0x793	101.982203

we have to replace every negative -1 with the median value of the non negative entries of Amount_invested_monthly for every Customer_ID

```
def median_of_non_negative_entries(x):
  non_negative_Amount_invested_monthly = x[x >=  0]
  median_value = non_negative_Amount_invested_monthly.median()
  x[x < 0] = median_value
  return x

data['Amount_invested_monthly'] = data.groupby('Customer_ID')['Amount_invested_monthly'].transform(median_of_non_negative_entries)
```

rechecking

```
data[['Customer_ID','Amount_invested_monthly']].loc[73645:73670]
```

	Customer_ID	Amount_invested_monthly
73645	CUS_0x3b1f	50.391471
73646	CUS_0x3b1f	147.303648
73647	CUS_0x3b1f	357.009409
73648	CUS_0xb742	123.971219
73649	CUS_0xb742	123.971219
73650	CUS_0xb742	123.971219
73651	CUS_0xb742	133.530087
73652	CUS_0xb742	123.971219
73653	CUS_0xb742	123.971219
73654	CUS_0xb742	123.971219
73655	CUS_0xb742	56.514450
73656	CUS_0xdcd	105.213125

Payment_Behaviour

73659	CUS_0xdcd	176.576593

```
data.Payment_Behaviour.unique()
```

```
array(['High_spent_Small_value_payments',
       'Low_spent_Large_value_payments',
       'Low_spent_Medium_value_payments',
       'Low_spent_Small_value_payments',
       'High_spent_Medium_value_payments', '!@9#%8',
       'High_spent_Large_value_payments'], dtype=object)
```

from above observation we an see there is a wrong entry '!@9#%8' in Payment_Behaviour column so we need to check where it is coming

```
data[data.Payment_Behaviour == '!@9#%8']['Payment_Behaviour']
```

```
5        !@9#%8
16       !@9#%8
32       !@9#%8
47       !@9#%8
54       !@9#%8
          ...
99947    !@9#%8
99980    !@9#%8
99982    !@9#%8
99989    !@9#%8
99999    !@9#%8
Name: Payment_Behaviour, Length: 7600, dtype: object
```

```
data[['Customer_ID','Month','Payment_Behaviour']].loc[99980:99999]
```

	Customer_ID	Month	Payment_Behaviour
99980	CUS_0xaf61	May	!@9#%8
99981	CUS_0xaf61	June	Low_spent_Small_value_payments
99982	CUS_0xaf61	July	!@9#%8
99983	CUS_0xaf61	August	High_spent_Medium_value_payments
99984	CUS_0x8600	January	High_spent_Large_value_payments
99985	CUS_0x8600	February	Low_spent_Small_value_payments
99986	CUS_0x8600	March	Low_spent_Small_value_payments
99987	CUS_0x8600	April	High_spent_Large_value_payments
99988	CUS_0x8600	May	Low_spent_Small_value_payments
99989	CUS_0x8600	June	!@9#%8
99990	CUS_0x8600	July	Low_spent_Large_value_payments
99991	CUS_0x8600	August	High_spent_Large_value_payments
99992	CUS_0x942c	January	Low_spent_Small_value_payments
99993	CUS_0x942c	February	Low_spent_Medium_value_payments
99994	CUS_0x942c	March	High_spent_Medium_value_payments
99995	CUS_0x942c	April	High_spent_Large_value_payments
99996	CUS_0x942c	May	High_spent_Medium_value_payments
99997	CUS_0x942c	June	High_spent_Large_value_payments
99998	CUS_0x942c	July	Low_spent_Large_value_payments
99999	CUS_0x942c	August	!@9#%8

```
def mode_of_non_wrong_entries(x):
  non_wrong_entries = x[x != '!@9#%8']
  mode_value = non_wrong_entries.mode().iloc[0]
  x[x == '!@9#%8'] = mode_value
  return x

data['Payment_Behaviour'] = data.groupby('Customer_ID')['Payment_Behaviour'].transform(mode_of_non_wrong_entries)
```

```
data.Payment_Behaviour.unique()
```

```
array(['High_spent_Small_value_payments',
       'Low_spent_Large_value_payments',
       'Low_spent_Medium_value_payments',
       'Low_spent_Small_value_payments',
       'High_spent_Medium_value_payments',
       'High_spent_Large_value_payments'], dtype=object)
```

'Monthly_Balance'

```
data.Monthly_Balance.sort_values(ascending=True)
```

```
71453       0.007760
43200       0.088628
77405       0.095482
60346       0.131136
69129       0.366147
              ...
15878    1564.134826
17029    1566.613165
33072    1567.208309
7475     1576.288935
9376     1602.040519
Name: Monthly_Balance, Length: 100000, dtype: float64
```

$Now - All - variables - are - cleaned$

we can proceed further for analysis

```
data.columns
```

```
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'],
      dtype='object')
```

```
data.head(5)
```

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	...	Num_Credit_Inquiries
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 rows × 27 columns

Identification of variables and data types

```
data.shape
```

```
(100000, 27)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 27 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   ID                        100000 non-null  object 
 1   Customer_ID               100000 non-null  object 
 2   Month                     100000 non-null  object 
 3   Name                      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        96623 non-null   float64
 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
dtypes: float64(9), int64(8), object(10)
memory usage: 20.6+ MB
```

Analysing the basic metrics

Univariate and Bivariate

```
data.describe(include =[np.number]).shape
```

```
(8, 17)
```

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

Out of the given data we should profile the candiates for that we should select certain features from data from analysis

```
data_profile = data.iloc[:,1:14].drop(['SSN'],axis =1)
data_profile
```

```
data_profile.shape
```

```
(100000, 12)
```

```
data_profile.columns
```

```
Index(['Customer_ID', 'Month', 'Name', 'Age', 'Occupation', 'Annual_Income',
       'Monthly_Inhand_Salary', 'Num_Bank_Accounts', 'Num_Credit_Card',
       'Interest_Rate', 'Num_of_Loan', 'Type_of_Loan'],
      dtype='object')
```

```
data_profile.groupby(['Customer_ID','Annual_Income']).agg({'Num_of_Loan':'sum'}).reset_index()
```

	Customer_ID	Annual_Income	Num_of_Loan

```
data_profile_Income_and_loan = data_profile.groupby(['Customer_ID','Annual_Income']).agg({'Num_of_Loan':'sum'}).reset_index()
```

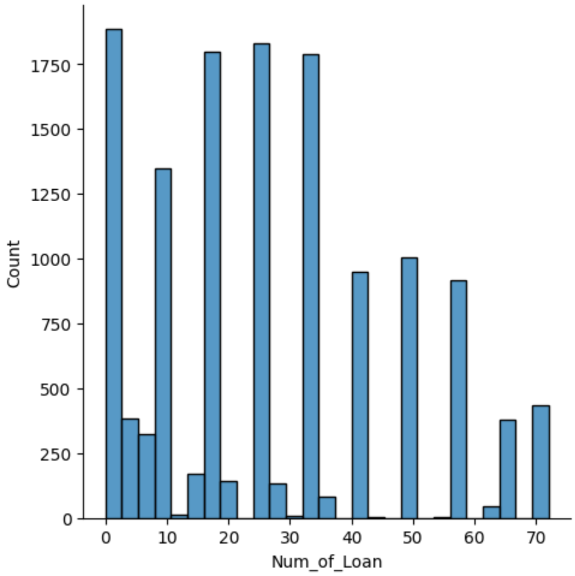
```
data_profile_Income_and_loan
```

	Customer_ID	Annual_Income	Num_of_Loan
0	CUS_0x1000	30625.940	16
1	CUS_0x1009	52312.680	32
2	CUS_0x100b	113781.390	0
3	CUS_0x1011	58918.470	24
4	CUS_0x1013	98620.980	24
...	...	...	...
13620	CUS_0xff3	17032.785	24
13621	CUS_0xff4	25546.260	40
13622	CUS_0xff6	117639.920	16
13623	CUS_0xffc	60877.170	64
13624	CUS_0xffd	41398.440	48

13625 rows × 3 columns

```
sns.displot(data=data_profile_Income_and_loan, x='Num_of_Loan')
```

```
<seaborn.axisgrid.FacetGrid at 0x7e1604117d60>
```



checking num of loans and number of bank accounts

```
data_profile.columns
```

```
Index(['Customer_ID', 'Month', 'Name', 'Age', 'Occupation', 'Annual_Income',
       'Monthly_Inhand_Salary', 'Num_Bank_Accounts', 'Num_Credit_Card',
       'Interest_Rate', 'Num_of_Loan', 'Type_of_Loan'],
      dtype='object')
```

```
pd.crosstab(index = data_profile.Num_Bank_Accounts,columns = data_profile.Num_of_Loan)
```

Num_of_Loan	0	1	2	3	4	5	6	7	8	9
Num_Bank_Accounts										

the above analysis tells the the number of loans taken according to number of banks

```
data.Annual_Income.describe()
```

```
count    1.000000e+05
mean     1.764157e+05
std      1.429618e+06
min      7.005930e+03
25%      1.945750e+04
50%      3.757861e+04
75%      7.279092e+04
max      2.419806e+07
Name: Annual_Income, dtype: float64
```

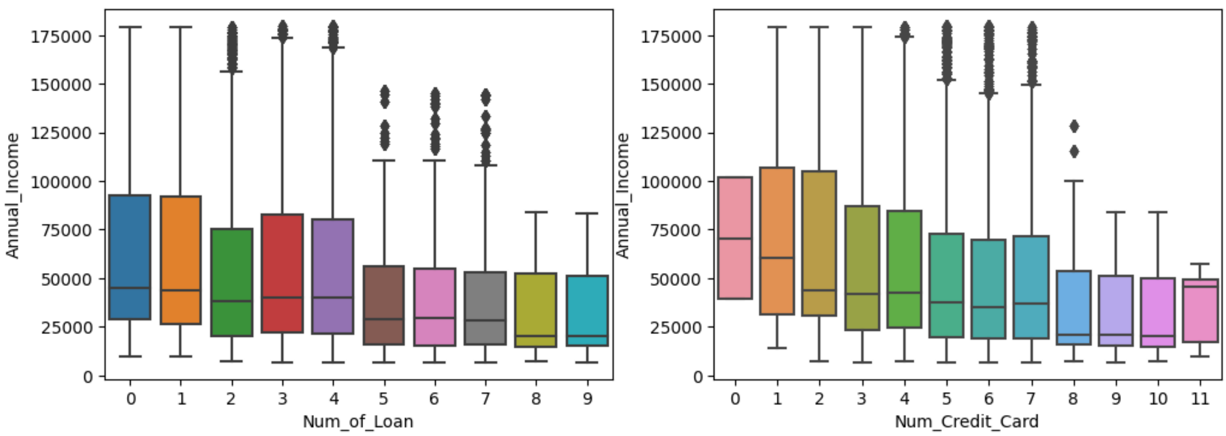
```
data.Annual_Income
```

```
0        19114.12
1        19114.12
2        19114.12
3        19114.12
4        19114.12
           ...
99995    39628.99
99996    39628.99
99997    39628.99
99998    39628.99
99999    39628.99
Name: Annual_Income, Length: 100000, dtype: float64
```

```
data_box = data[data.Annual_Income < np.percentile(data.Annual_Income,99)]
```

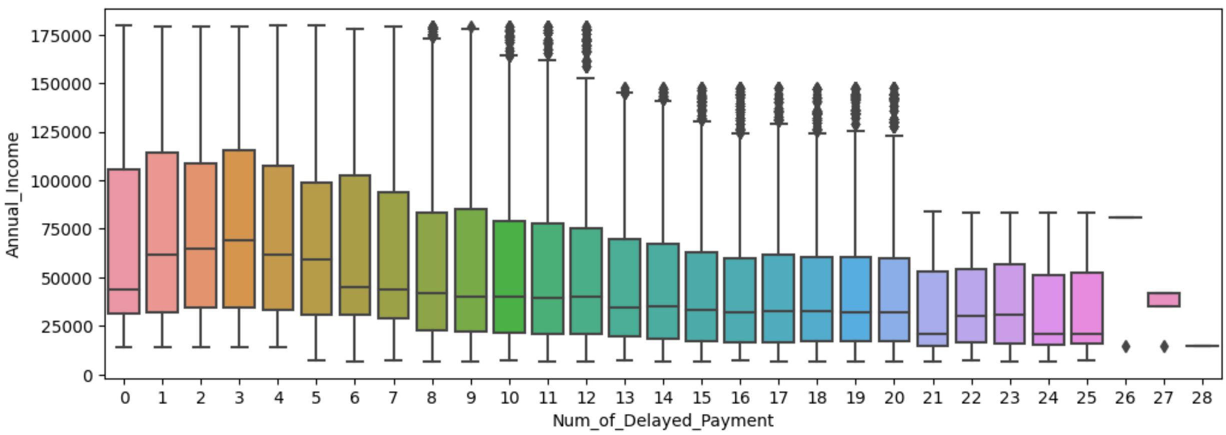
```
plt.figure(figsize = (12,4))
plt.subplot(1,2,1)
sns.boxplot(x = data_box.Num_of_Loan,y = data_box.Annual_Income)
plt.subplot(1,2,2)
sns.boxplot(x = data_box.Num_Credit_Card,y = data_box.Annual_Income)
```

```
<Axes: xlabel='Num_Credit_Card', ylabel='Annual_Income'>
```



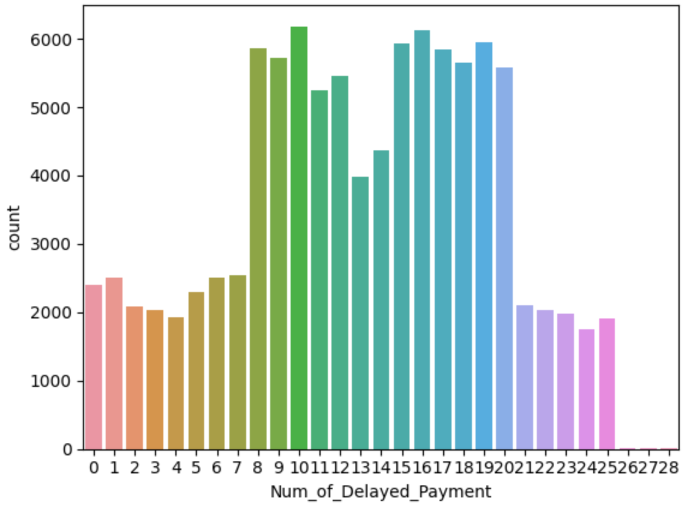

```
plt.figure(figsize = (12,4))
sns.boxplot(x = data_box.Num_of_Delayed_Payment,y = data_box.Annual_Income)
```

```
<Axes: xlabel='Num_of_Delayed_Payment', ylabel='Annual_Income'>
```



```
sns.countplot(data = data, x = data.Num_of_Delayed_Payment)
```

```
<Axes: xlabel='Num_of_Delayed_Payment', ylabel='count'>
```



Most of the people have dealyed maximum around 8- 20 days

```
sns.distplot(data.Credit_History_Age)
```

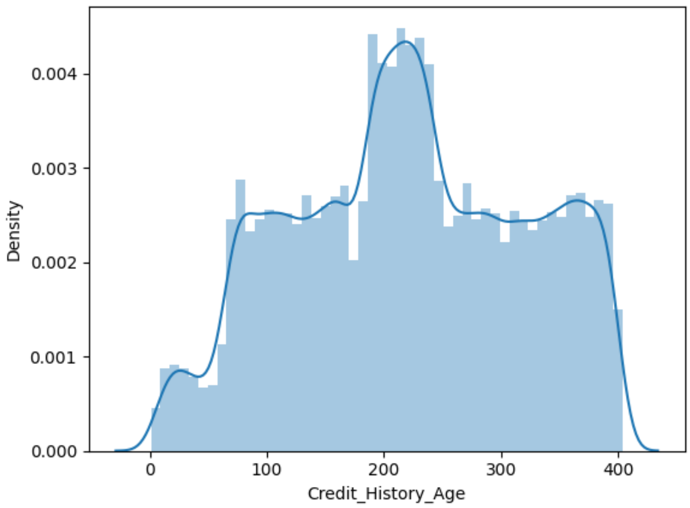
```
<ipython-input-219-717aadc6277b>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data.Credit_History_Age)
<Axes: xlabel='Credit_History_Age', ylabel='Density'>
```



from above analysis we can say the credit history month is mostly greater than 200 month and less than 250 month

checking for number of loan according to age of candidate

```
data.Age.unique()
```

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

```
pd.crosstab(index = pd.cut(data.Age, bins =[20,30,40,50,60]),columns = data.Num_of_Loan,margins=True)
```

Num_of_Loan	0	1	2	3	4	5	6	7	8	9	All
Age											
(20, 30]	3072	3016	4688	4104	4312	2488	2472	2464	928	1256	28800
(40, 50]	2808	2912	3520	3408	3480	1048	1400	1272	480	552	20880

the above analysis shows the distribution of number of loan taken w.r.t to Age of individuals

Age vs Credit Utilization ratio

for better understanding i had created

bins for credit_utilization ratio : (20, 25] (25, 30] (30, 35] (35, 40] (40, 45] (45, 50]

age: [20,30,40,50,60]

```
pd.crosstab(index = pd.cut(data.Age, bins =[20,30,40,50,60]),columns = pd.cut(data.Credit_Utilization_Ratio, bins =[20,25,30,35,40,45,50]),margins=True)
```

Credit_Utilization_Ratio	(20, 25]	(25, 30]	(30, 35]	(35, 40]	(40, 45]	(45, 50]	All
Age							
(20, 30]	2363	8168	8505	8158	1562	44	28800
(30, 40]	2468	8074	8558	8300	1545	46	28991
(40, 50]	1704	5830	6171	5867	1269	39	20880
(50, 60]	516	1890	2193	2047	545	17	7208
All	7051	23962	25427	24372	4921	146	85879

above analysis shows the distribution of credit utilization and Age column the maximum Credit_Utilization_Ratio is shown for people belonging to age within range (30 40] .the credit score inversily relates with Credit_utilization_Ratio

Deriving Feature for Credit Card score calculation for individuals

Behavioural Score card

Before proceeding we will store our process data into a new variable name inorder to differentiate between old data and new data

```
processed_data = data
```

```
processed_data.columns
```

```
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'],
      dtype='object')
```

Feature enginerring

Outstanding balance

Outstanding balance = amount due - amount paid

```
processed_data.Outstanding_Debt
```

```
0        809.98
1        809.98
2        809.98
3        809.98
4        809.98
          ...
99995    502.38
99996    502.38
99997    502.38
99998    502.38
99999    502.38
Name: Outstanding_Debt, Length: 100000, dtype: float64
```

Debt to income ratio

outstanding debt/Monthly_Inhand_Salary

or outstanding debt/Annual_Income

```
processed_data['Debt_to_income_ratio'] = processed_data['Monthly_Inhand_Salary']/processed_data['Outstanding_Debt']
```

Number of delayed payment

Represents the average number of payments delayed by a person

Payment History

weigh = 40%

This factor evaluates how consistently a borrower has made payments on their debts. A borrower who has always made on-time payments will receive a higher score than one who has missed payments.

Components that make up payment history:

Payment information on credit cards, retail accounts, installment loans, mortgages and other types of accounts

How overdue delinquent payments are today or may have become in the past

The amount of money still owed on delinquent accounts or collection items

The number of past due items on a credit report

Adverse public records (e.g., bankruptcies)

The amount of time that's passed since delinquencies, adverse public records or collection items were introduced

```
processed_data['Payment_History'] = processed_data.Num_of_Delayed_Payment
```

Amount owed

weigh = 30%

for credit score calculation we will be using Credit_Utilization_Ratio for amount owed

Credit_Utilization_Ratio: The credit utilization ratio is the percentage of a borrower’s total available credit that is currently being used. High utilization may indicate a higher risk.

Length of credit history

weigh = 15%

For this we will use

Credit History

Credit history is the ongoing documentation of your financial information, including repayment of your debts

This factor evaluates how long a borrower has had credit accounts open. A borrower who has a long history of credit accounts in good standing will receive a higher score than one who is new to credit.

```
processed_data.Credit_History_Age   #Represents the age of credit history of the person by month
```

```
0        265.0
1        266.0
2        267.0
3        268.0
4        269.0
         ...
99995    378.0
99996    379.0
99997    380.0
99998    381.0
99999    382.0
Name: Credit_History_Age, Length: 100000, dtype: float64
```

Types of credit accounts

weigh = 10%

for this feature will use

Credit mix

This factor evaluates the types of credit accounts a borrower has, such as credit cards, loans, and mortgages. A borrower who has a diverse mix of credit accounts will receive a higher score than one who only has one type of account

```
processed_data.Credit_Mix
```

```
0        Good
1        Good
2        Good
3        Good
4        Good
         ...
99995    Good
99996    Good
99997    Good
99998    Good
99999    Good
Name: Credit_Mix, Length: 100000, dtype: object
```

```
processed_data.Credit_Mix.unique()
```

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

since it is categorical so we will enocde it to for standardization

Good = 2

Standard = 1

Bad = 0

```
processed_data.Credit_Mix.replace({'Good': 2, 'Standard': 1, 'Bad': 0}, inplace=True)
```

Recent credit inquiries

weigh = 10%

This factor evaluates how frequently a borrower has applied for credit. A borrower who has made few recent credit inquiries will receive a higher score than one who has made many.

```
processed_data.Num_Credit_Inquiries
```

```
0        4
1        4
2        4
3        4
4        4
        ..
99995    3
99996    3
99997    3
99998    3
99999    3
Name: Num_Credit_Inquiries, Length: 100000, dtype: int64
```

$-----Credit-Score-calculation-----$

```
credit_data = processed_data.groupby('Customer_ID').agg({'Payment_History':'mean','Credit_Utilization_Ratio':'mean','Credit_History_Age':'max','Credit_M
```

```
credit_data = credit_data.reset_index()
```

```
credit_data
```

	Customer_ID	Payment_History	Credit_Utilization_Ratio	Credit_History_Age	Credit_Mix	Num_Credit_Inquiries
0	CUS_0x1000	24.500	33.477546	129.0	0	87

```
credit_data
```

	Customer_ID	Payment_History	Credit_Utilization_Ratio	Credit_History_Age	Credit_Mix	Num_Credit_Inquiries
0	CUS_0x1000	24.500	33.477546	129.0	0	87
1	CUS_0x1009	17.750	29.839984	372.0	1	16
2	CUS_0x100b	7.000	34.841449	190.0	2	8
3	CUS_0x1011	14.375	27.655897	190.0	1	56
4	CUS_0x1013	8.500	31.933940	214.0	2	24
...	...	...	...	...	...	...
12495	CUS_0xff3	8.375	32.889398	207.0	2	34
12496	CUS_0xff4	10.000	32.598257	225.0	1	40
12497	CUS_0xff6	3.625	33.258053	299.0	2	16
12498	CUS_0xffc	15.500	34.722108	157.0	0	99
12499	CUS_0xffd	11.500	31.894261	225.0	1	56

12500 rows × 6 columns

weigh allotment summary

Payment_History : 35%

Credit_Utilization_Ratio = 30%

Credit_History_Age : 15 %

credit mix = 10%

credit inquires = 10%

```
credit_data['credit_scores'] = (0.35*credit_data.Payment_History + 0.30*credit_data.Credit_Utilization_Ratio + 0.15*credit_data.Credit_History_Age + 0.
```

Normalizing the credit score data from 0- 1000

```
credit_data['credit_scores']  = (credit_data['credit_scores']  - credit_data['credit_scores'].min())*1000/(credit_data['credit_scores'].max()- credit_d
```

```
credit_data[['Customer_ID','credit_scores']]
```

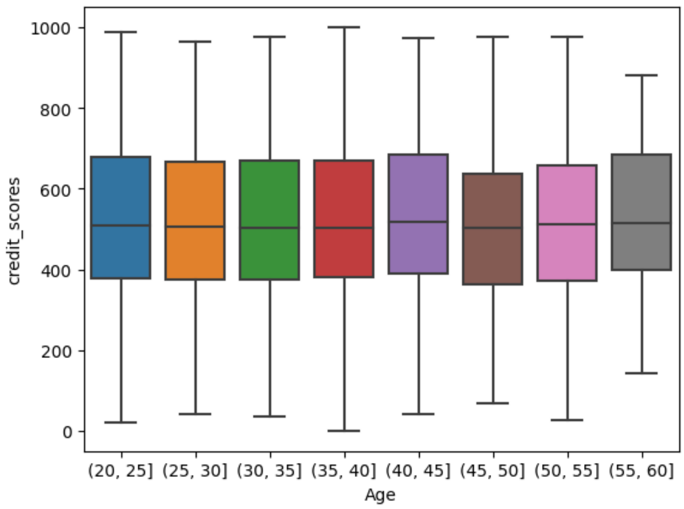
```
new_data = data.merge(credit_data,how='left',on='Customer_ID')
new_data
```

Observation around credit score

Age-Credit score

```
sns.boxplot(x=pd.cut(new_data.Age, bins =[20,25,30,35,40,45,50,55,60]),y=credit_data.credit_scores)
```

```
<Axes: xlabel='Age', ylabel='credit_scores'>
```



from above observation we can see the credit scores for all the ages lies around 400 -600 0n 1000 scale

```
sns.distplot(credit_data.credit_scores)
```

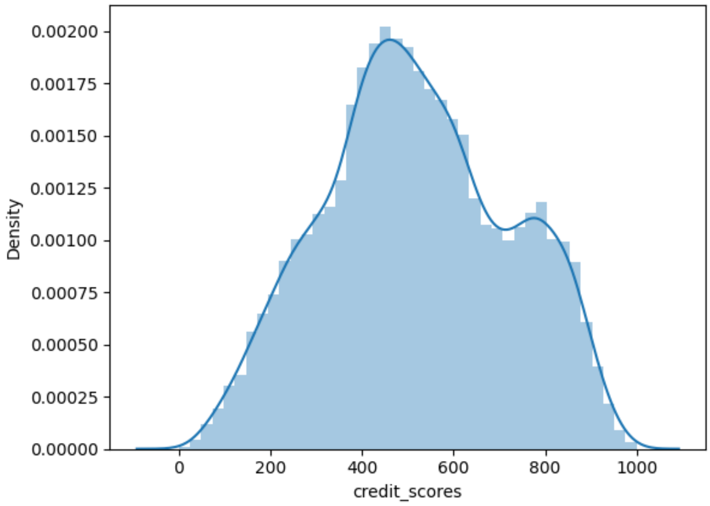
```
<ipython-input-244-614c417d09b5>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(credit_data.credit_scores)
<Axes: xlabel='credit_scores', ylabel='Density'>
```

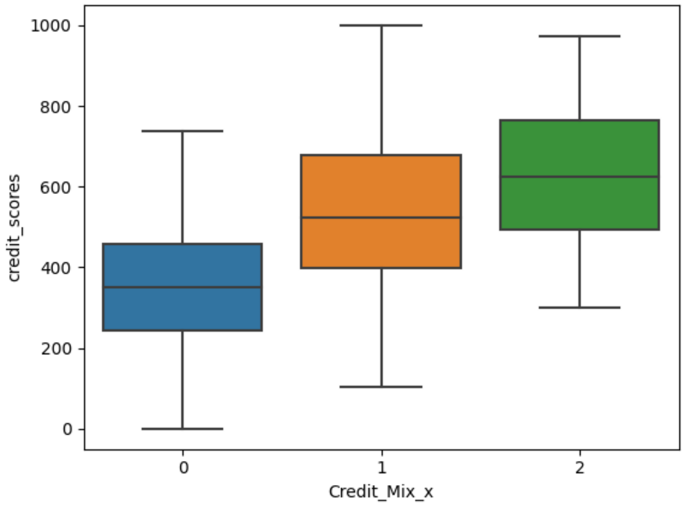


the credit score distribution for individuals appears normal distribution

credit mix and credit scores

```
sns.boxplot(x=new_data.Credit_Mix_x, y = new_data.credit_scores)
```

```
<Axes: xlabel='Credit_Mix_x', ylabel='credit_scores'>
```

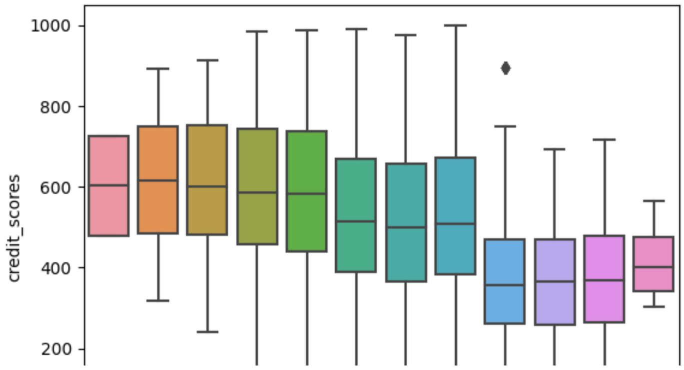


from above analysis it shows the people with good credit mix(multiple types of loan accounts individual hold) have higher credit scores as compared to standard and bad credit mix

Num of credit cards and Credit score

```
sns.boxplot(x=new_data.Num_Credit_Card, y = new_data.credit_scores)
```

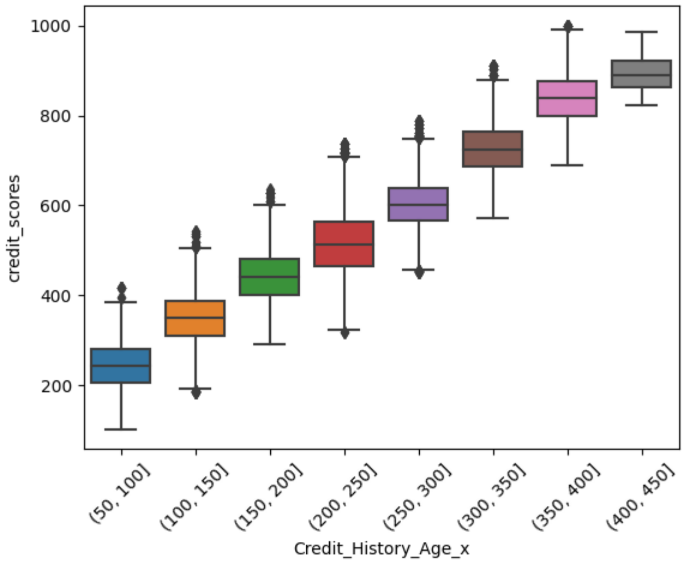
```
<Axes: xlabel='Num_Credit_Card', ylabel='credit_scores'>
```



from above analysis we can say that the people with < 5 credit cards are having gretaer credit score than people with have 5 or more credit cards

Credit History Age and Credit score

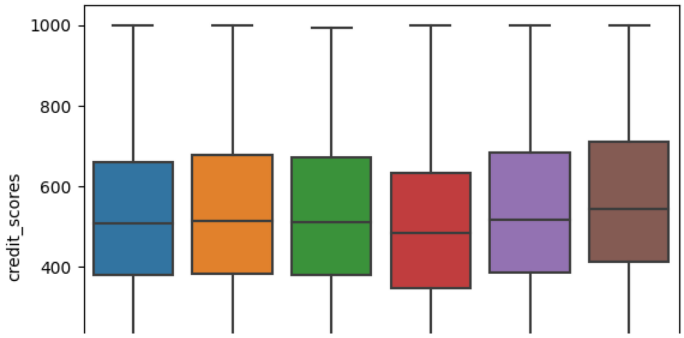
```
sns.boxplot(x = pd.cut(new_data.Credit_History_Age_x, bins =[50,100,150,200,250,300,350,400,450]), y = new_data.credit_scores)
plt.xticks(rotation = 45)
plt.show()
```



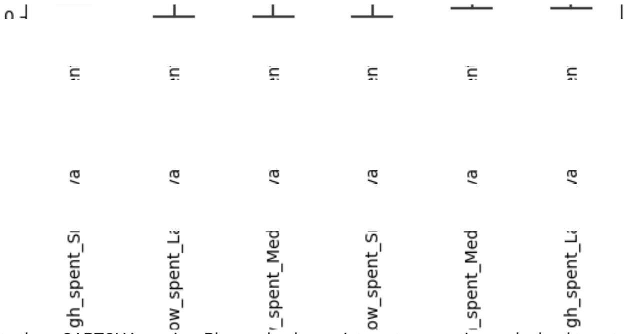
From above graph we can see as the credit history in months of an individuals increase the credit scores also increases which suggest that people who have long credit history is less likely to deafult as they have good credit score

Payment behaviour and Credit score

```
sns.boxplot(x=new_data.Payment_Behaviour, y = new_data.credit_scores)
plt.xticks(rotation = 90)
plt.show()
```



the above analysis shows the the payment behaviour is less likely effects credit score or we can say there is almost no correlation among each other



404.0

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.