Credit_EDA_and_Scoring

October 9, 2024

1 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
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
!pip install pandas_profiling
```

[ ]:
```
from ydata_profiling import ProfileReport
```

[ ]:
```
!gdown 1pljm6_3nxcFS9UMIFm124HBsjNZP6ACA
```

```
Downloading…
From: https://drive.google.com/uc?id=1pljm6_3nxcFS9UMIFm124HBsjNZP6ACA
To: /content/Credit_score.csv
100% 27.4M/27.4M [00:00<00:00, 68.3MB/s]
```

[ ]:
```
df=pd.read_csv('/content/Credit_score.csv')
```

Downloading the dataset

[ ]:
```
df.shape
```

[ ]: (100000, 27)

Dataset contains 100000 rows and 27 columns

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

[ ]:
```
       ID Customer_ID     Month           Name    Age          SSN Occupation  \
0  0x1602   CUS_0xd40   January  Aaron Maashoh     23  821-00-0265  Scientist
```

```
1  0x1603   CUS_0xd40  February  Aaron Maashoh    23  821-00-0265  Scientist
2  0x1604   CUS_0xd40     March  Aaron Maashoh  -500  821-00-0265  Scientist
3  0x1605   CUS_0xd40     April  Aaron Maashoh    23  821-00-0265  Scientist
4  0x1606   CUS_0xd40       May  Aaron Maashoh    23  821-00-0265  Scientist

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

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

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

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

[5 rows x 27 columns]
```

First five rows of the dataset

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

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

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

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

[ ]:
```
0
```

The dataset contains no duplicate values.

[ ]:
```
df.isnull().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
```

Columns Name, Monthly Inhand Salary, Type of loan, Num of delayed payment, Num credit Inquiries, Credit History Age, Amount invested monthly and Monthly balance contains null values

[ ]:
```
ProfileReport(df)
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

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

[ ]:

[ ]:
```
df['Month']=df['Month'].astype('category')
df['Name']=df['Name'].astype('category')
df['Credit_Mix']=df['Credit_Mix'].astype('category')
df['Payment_of_Min_Amount']=df['Payment_of_Min_Amount'].astype('category')
df['Occupation']=df['Occupation'].astype('category')
```

Columns like Month, Name, Credit Mix, Payment of min amount and occupation are converted to category datatype

1.1 Data Cleaning

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

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

[ ]:
```
df['Annual_Income'].unique()
```

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

[ ]:
```
df['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)
```

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

[ ]:
```
array(['7', nan, '4', '8_', '6', '1', '-1', '3_', '0', '8', '5', '3', '9',
       '12', '15', '17', '10', '2', '2_', '11', '14', '20', '22', '13',
       '13_', '14_', '16', '12_', '18', '19', '23', '24', '21', '3318',
       '3083', '22_', '1338', '4_', '26', '11_', '3104', '21_', '25',
       '10_', '183_', '9_', '1106', '834', '19_', '24_', '17_', '23_',
       '2672', '20_', '2008', '-3', '538', '6_', '1_', '16_', '27', '-2',
       '3478', '2420', '15_', '707', '708', '26_', '18_', '3815', '28',
       '5_', '1867', '2250', '1463', '25_', '7_', '4126', '2882', '1941',
       '2655', '2628', '132', '3069', '306', '0_', '3539', '3684', '1823',
       '4128', '1946', '827', '2297', '2566', '904', '182', '929', '3568',
       '2503', '1552', '2812', '1697', '3764', '851', '3905', '923', '88',
       '1668', '3253', '808', '2689', '3858', '642', '3457', '1402',
       '1732', '3154', '847', '3037', '2204', '3103', '1063', '2056',
       '1282', '1841', '2569_', '211', '793', '3484', '411', '3491',
       '2072', '3050', '1049', '2162', '3402', '2753', '27_', '1718',
       '1014', '3260', '3855', '84', '2311', '3251', '1832', '4069',
       '3010', '733', '4241', '166', '2461', '1749', '3200', '663_',
       '2185', '4161', '3009', '359', '2015', '1523', '594', '1079',
       '1199', '186', '1015', '1989', '281', '559', '2165', '1509',
       '3545', '779', '192', '4311', '-2_', '2323', '1471', '1538',
       '3529', '439', '3456', '3040', '2697', '3179', '1332', '3175',
       '3112', '829', '4022', '3870', '4023', '531', '1511', '3092',
       '3191', '2400', '3621', '3536', '544', '1864', '28_', '142',
       '2300', '264', '72', '497', '398', '2222', '3960', '1473', '3043',
       '4216', '2903', '2658', '-1_', '4042', '1323_', '2184', '921',
       '1328', '3404', '2438', '809', '47', '1996', '4164', '1370',
       '1204', '2167', '4011', '2590', '2594', '2533', '1663', '1018',
       '2919', '3458', '3316', '2589', '2801', '3355', '2529', '2488',
       '4266', '1243', '739', '845', '4107', '1884', '337', '2660', '290',
       '674', '2450', '3738', '1792', '2823', '2570', '775', '960', '482',
       '1706', '2493', '3623', '3031', '2794_', '2219_', '758_', '1849',
       '3559', '4096', '3726', '1953', '2657', '4043', '2938', '4384',
       '1647', '2694', '3533', '519', '2677', '2413', '-3_', '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', '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)
```

[ ]:
```
df['Changed_Credit_Limit'].unique()
```

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

[ ]:
```
df['Outstanding_Debt'].unique()
```

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

[ ]:
```
df['Amount_invested_monthly'].unique()
```

[ ]:
```
array(['80.41529544', '118.2802216', '81.69952126', …, '24.02847745',
       '251.6725822', '167.1638652'], dtype=object)
```

[ ]:
```
df['Monthly_Balance'].unique()
```

[ ]:
```
array(['312.4940887', '284.6291625', '331.2098629', …, 516.8090833,
       319.1649785, 393.6736956], dtype=object)
```

[ ]:
```
df.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')
```

[ ]:
```
def remove_underscore(value):
      # Check if the value is a string
    if isinstance(value, str):
        # Remove underscores from the string and return the modified value
        return value.replace('_', '') if value != 'nan' else float('nan')
    else:
        # If it's already a float or NaN, return it as is
        return value


def modify_data(columns):
```

```
    for each_column in columns:
        # Apply the remove_underscore function to the column and convert 'nan'␣
 ↪strings to NaN
        df[each_column] = df[each_column].apply(remove_underscore)

        # Convert valid numeric strings to float and handle conversion errors
        df[each_column] = pd.to_numeric(df[each_column], errors='coerce')

# Specify columns to modify
columns_to_modify = ['Age',␣
 ↪'Annual_Income','Num_of_Loan','Num_of_Delayed_Payment','Changed_Credit_Limit','Outstanding_
                     'Amount_invested_monthly','Monthly_Balance']
modify_data(columns_to_modify)
```

Removed "_" and converted some columns to float

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 27 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   ID                        100000 non-null  object
 1   Customer_ID               100000 non-null  object
 2   Month                     100000 non-null  category
 3   Name                      90015 non-null   category
 4   Age                       100000 non-null  int64
 5   SSN                       100000 non-null  object
 6   Occupation                100000 non-null  category
 7   Annual_Income             100000 non-null  float64
 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  int64
 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   float64
 16  Changed_Credit_Limit      97909 non-null   float64
 17  Num_Credit_Inquiries      98035 non-null   float64
 18  Credit_Mix                100000 non-null  category
 19  Outstanding_Debt          100000 non-null  float64
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        90970 non-null   object
 22  Payment_of_Min_Amount     100000 non-null  category
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   95521 non-null   float64
```

```
 25  Payment_Behaviour         100000 non-null  object
 26  Monthly_Balance           98800 non-null   float64
dtypes: category(5), float64(10), int64(6), object(6)
memory usage: 17.7+ MB
```

[ ]:
```python
df.isnull().sum().sort_values(ascending=False).to_frame('Missing_Values')
```

[ ]:

	Missing_Values
Monthly_Inhand_Salary	15002
Type_of_Loan	11408
Name	9985
Credit_History_Age	9030
Num_of_Delayed_Payment	7002
Amount_invested_monthly	4479
Changed_Credit_Limit	2091
Num_Credit_Inquiries	1965
Monthly_Balance	1200
Num_Bank_Accounts	0
Num_Credit_Card	0
Payment_Behaviour	0
Month	0
Total_EMI_per_month	0
Payment_of_Min_Amount	0
Age	0
Credit_Utilization_Ratio	0
Outstanding_Debt	0
Credit_Mix	0
SSN	0
Occupation	0
Annual_Income	0
Delay_from_due_date	0
Customer_ID	0
Num_of_Loan	0
Interest_Rate	0
ID	0

[ ]:
```python
df.describe(include='number').T
```

[ ]:

	count	mean	std	min	\
Age	100000.0	1.106497e+02	6.862447e+02	-5.000000e+02	
Annual_Income	100000.0	1.764157e+05	1.429618e+06	7.005930e+03	
Monthly_Inhand_Salary	84998.0	4.194171e+03	3.183686e+03	3.036454e+02	
Num_Bank_Accounts	100000.0	1.709128e+01	1.174048e+02	-1.000000e+00	
Num_Credit_Card	100000.0	2.247443e+01	1.290574e+02	0.000000e+00	
Interest_Rate	100000.0	7.246604e+01	4.664226e+02	1.000000e+00	
Num_of_Loan	100000.0	3.009960e+00	6.264788e+01	-1.000000e+02	
Delay_from_due_date	100000.0	2.106878e+01	1.486010e+01	-5.000000e+00	

```
Num_of_Delayed_Payment      92998.0  3.092334e+01  2.260319e+02 -3.000000e+00
Changed_Credit_Limit        97909.0  1.038903e+01  6.789496e+00 -6.490000e+00
Num_Credit_Inquiries        98035.0  2.775425e+01  1.931773e+02  0.000000e+00
Outstanding_Debt           100000.0  1.426220e+03  1.155129e+03  2.300000e-01
Credit_Utilization_Ratio   100000.0  3.228517e+01  5.116875e+00  2.000000e+01
Total_EMI_per_month        100000.0  1.403118e+03  8.306041e+03  0.000000e+00
Amount_invested_monthly     95521.0  6.374130e+02  2.043319e+03  0.000000e+00
Monthly_Balance             98800.0 -3.036437e+22  3.181295e+24 -3.333333e+26

                                 25%           50%           75%  \
Age                        24.000000     33.000000     42.000000
Annual_Income           19457.500000  37578.610000  72790.920000
Monthly_Inhand_Salary    1625.568229   3093.745000   5957.448333
Num_Bank_Accounts           3.000000      6.000000      7.000000
Num_Credit_Card             4.000000      5.000000      7.000000
Interest_Rate               8.000000     13.000000     20.000000
Num_of_Loan                 1.000000      3.000000      5.000000
Delay_from_due_date        10.000000     18.000000     28.000000
Num_of_Delayed_Payment      9.000000     14.000000     18.000000
Changed_Credit_Limit        5.320000      9.400000     14.870000
Num_Credit_Inquiries        3.000000      6.000000      9.000000
Outstanding_Debt          566.072500   1166.155000   1945.962500
Credit_Utilization_Ratio   28.052567     32.305784     36.496663
Total_EMI_per_month        30.306660     69.249473    161.224249
Amount_invested_monthly    74.534002    135.925681    265.731733
Monthly_Balance           270.092209    336.719190    470.220186

                                 max
Age                     8.698000e+03
Annual_Income           2.419806e+07
Monthly_Inhand_Salary   1.520463e+04
Num_Bank_Accounts       1.798000e+03
Num_Credit_Card         1.499000e+03
Interest_Rate           5.797000e+03
Num_of_Loan             1.496000e+03
Delay_from_due_date     6.700000e+01
Num_of_Delayed_Payment  4.397000e+03
Changed_Credit_Limit    3.697000e+01
Num_Credit_Inquiries    2.597000e+03
Outstanding_Debt        4.998070e+03
Credit_Utilization_Ratio 5.000000e+01
Total_EMI_per_month     8.233100e+04
Amount_invested_monthly 1.000000e+04
Monthly_Balance         1.602041e+03
```

Insights:

Age Distribution: The average age is unexpectedly high (110.65), and the standard deviation

(686.24) is unusually large, indicating possible outliers or data entry errors (e.g., negative values and a maximum of 8,698 years). Minimum age is -500, which is unrealistic, suggesting data quality issues.

Annual Income: The mean annual income is 176,415.70 USD, but the high standard deviation (1.42 million) indicates significant variability in income levels, with a maximum of over 24 million USD. The 25th percentile shows 19,457 USD, and the median is around $37,578, suggesting a skewed income distribution.

Monthly In-hand Salary: The average in-hand salary is 5,743.26 USD, but the standard deviation of 45,814.69 USD shows a large variation. The max value of nearly $2 million is unusually high, indicating possible outliers.

Number of Bank Accounts: The average number of bank accounts is 17.09, with a standard deviation of 117.40. This large variation suggests the presence of extreme values (max 1,798 accounts). The minimum value is -1, which points to incorrect data entries.

Number of Credit Cards: The average number of credit cards is 22.47, with a wide range (maximum of 1,499). The absence of negative values, but significant variability, may suggest outliers or misreporting.

Interest Rate: The mean interest rate is 72.47%, with a very high standard deviation (466.42%), indicating large discrepancies in interest rates. The max value is 5,797%, suggesting potential data entry errors.

Number of Loans: The average number of loans is 3.01, but with a minimum of -100 and a maximum of 1,496, this highlights potential outliers or incorrect data entries.

Delayed Payments: The average number of delayed payments is 29.74, with significant variability (standard deviation of 218). The maximum of 4,397 delayed payments suggests the presence of extreme cases.

Changed Credit Limit: On average, the credit limit was changed by 10.39 units, with the maximum change being 36.97. Negative values (minimum -6.49) could indicate reductions in credit limits.

Credit Inquiries: The average number of credit inquiries is 27.33, with a maximum of 2,597. The high standard deviation suggests that some individuals have a significantly higher number of inquiries.

Outstanding Debt: The mean outstanding debt is 1,426.22 USD, with a fairly high standard deviation ($1,155.13). The debt range (from 0.23 to $4,998) indicates a wide variety of debt loads.

Credit Utilization Ratio: The average credit utilization ratio is 32.29%, which falls within the typical 30-40% range for responsible credit usage.

Total EMI per Month: The average EMI per month is 1,403.12 USD, with a maximum of $82,331, suggesting significant variation in the amount of debt repayments.

Monthly Balance: The extreme negative mean value (-3e+22) and large variation (max 1.6k, min -3.33e+26) indicate major issues with the data quality in this column.

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

[ ]:
```
                      count  unique                              top   freq
ID                   100000  100000                           0x1602      1
Customer_ID          100000   12500                        CUS_0xd40      8
SSN                  100000   12501                        #F%$D@*&8   5572
Type_of_Loan          88592    6260                    Not Specified   1408
Credit_History_Age    90970     404          15 Years and 11 Months     446
Payment_Behaviour    100000       7  Low_spent_Small_value_payments  25513
```

[ ]:
```
df.describe(include='category').T
```

[ ]:
```
                        count unique       top   freq
Month                  100000      8     April  12500
Name                    90015  10139    Langep     44
Occupation             100000     16   _______   7062
Credit_Mix             100000      4  Standard  36479
Payment_of_Min_Amount  100000      3       Yes  52326
```

[ ]:
```
pd.set_option('display.max_columns',None)
```

[ ]:
```
df.head(3)
```

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

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

   Interest_Rate  Num_of_Loan  \
0              3            4
1              3            4
2              3            4

                                        Type_of_Loan  Delay_from_due_date  \
0  Auto Loan, Credit-Builder Loan, Personal Loan,…                     3
1  Auto Loan, Credit-Builder Loan, Personal Loan,…                    -1
2  Auto Loan, Credit-Builder Loan, Personal Loan,…                     3

  Num_of_Delayed_Payment  Changed_Credit_Limit  Num_Credit_Inquiries  \
0                    7.0                 11.27                   4.0
1                    NaN                 11.27                   4.0
2                    7.0                   NaN                   4.0

 Credit_Mix  Outstanding_Debt  Credit_Utilization_Ratio  \
```

```
0          _              809.98                26.822620
1       Good              809.98                31.944960
2       Good              809.98                28.609352

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

   Amount_invested_monthly                Payment_Behaviour  Monthly_Balance
0                80.415295  High_spent_Small_value_payments       312.494089
1               118.280222   Low_spent_Large_value_payments       284.629163
2                81.699521  Low_spent_Medium_value_payments       331.209863
```

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

[ ]: 0

[ ]:
```
correlation_matrix=df.select_dtypes(include='number').corr()
```

[ ]:
```
correlation_matrix
```

[ ]:
```
                              Age  Annual_Income  Monthly_Inhand_Salary  \
Age                      1.000000      -0.000533               0.003029
Annual_Income           -0.000533       1.000000               0.030519
Monthly_Inhand_Salary    0.003029       0.030519               1.000000
Num_Bank_Accounts        0.002466      -0.004627              -0.010819
Num_Credit_Card         -0.001791       0.001250              -0.005049
Interest_Rate           -0.005667      -0.002908              -0.006422
Num_of_Loan             -0.001273       0.001464              -0.010398
Delay_from_due_date     -0.009681      -0.010631              -0.250100
Num_of_Delayed_Payment  -0.002545       0.001180               0.000917
Changed_Credit_Limit    -0.003048       0.000615              -0.174933
Num_Credit_Inquiries    -0.002022       0.003153              -0.008867
Outstanding_Debt        -0.001995      -0.003706              -0.269727
Credit_Utilization_Ratio  0.002774      0.010316               0.173192
Total_EMI_per_month      0.000662      -0.000248               0.007264
Amount_invested_monthly  0.003797      -0.005318               0.061485
Monthly_Balance          0.001052       0.000818              -0.000806

                          Num_Bank_Accounts  Num_Credit_Card  Interest_Rate  \
Age                                0.002466        -0.001791      -0.005667
Annual_Income                     -0.004627         0.001250      -0.002908
Monthly_Inhand_Salary             -0.010819        -0.005049      -0.006422
Num_Bank_Accounts                  1.000000        -0.002216      -0.003998
Num_Credit_Card                   -0.002216         1.000000      -0.004012
Interest_Rate                     -0.003998        -0.004012       1.000000
```

Num_of_Loan	-0.000679	0.001421	0.000614
Delay_from_due_date	0.015966	0.008715	0.009792
Num_of_Delayed_Payment	-0.003619	0.004876	0.002669
Changed_Credit_Limit	0.008010	0.005599	0.000887
Num_Credit_Inquiries	-0.001683	-0.003479	-0.001681
Outstanding_Debt	0.015548	0.005626	0.010721
Credit_Utilization_Ratio	-0.001365	-0.003341	-0.000359
Total_EMI_per_month	-0.001433	0.000768	0.002517
Amount_invested_monthly	0.003233	-0.002072	-0.001203
Monthly_Balance	0.001065	0.001276	0.001256

	Num_of_Loan	Delay_from_due_date \
Age	-0.001273	-0.009681
Annual_Income	0.001464	-0.010631
Monthly_Inhand_Salary	-0.010398	-0.250100
Num_Bank_Accounts	-0.000679	0.015966
Num_Credit_Card	0.001421	0.008715
Interest_Rate	0.000614	0.009792
Num_of_Loan	1.000000	0.012625
Delay_from_due_date	0.012625	1.000000
Num_of_Delayed_Payment	0.010218	0.012657
Changed_Credit_Limit	0.015865	0.293697
Num_Credit_Inquiries	-0.000497	0.011508
Outstanding_Debt	0.023772	0.571713
Credit_Utilization_Ratio	-0.003967	-0.063796
Total_EMI_per_month	0.001554	-0.003889
Amount_invested_monthly	-0.000861	-0.012557
Monthly_Balance	-0.000033	0.004454

	Num_of_Delayed_Payment	Changed_Credit_Limit \
Age	-0.002545	-0.003048
Annual_Income	0.001180	0.000615
Monthly_Inhand_Salary	0.000917	-0.174933
Num_Bank_Accounts	-0.003619	0.008010
Num_Credit_Card	0.004876	0.005599
Interest_Rate	0.002669	0.000887
Num_of_Loan	0.010218	0.015865
Delay_from_due_date	0.012657	0.293697
Num_of_Delayed_Payment	1.000000	0.007136
Changed_Credit_Limit	0.007136	1.000000
Num_Credit_Inquiries	0.002837	0.006562
Outstanding_Debt	0.010121	0.458549
Credit_Utilization_Ratio	0.000866	-0.048862
Total_EMI_per_month	0.001120	-0.001840
Amount_invested_monthly	0.001280	-0.011704
Monthly_Balance	0.000931	-0.002182

	Num_Credit_Inquiries	Outstanding_Debt \
Age	-0.002022	-0.001995
Annual_Income	0.003153	-0.003706
Monthly_Inhand_Salary	-0.008867	-0.269727
Num_Bank_Accounts	-0.001683	0.015548
Num_Credit_Card	-0.003479	0.005626
Interest_Rate	-0.001681	0.010721
Num_of_Loan	-0.000497	0.023772
Delay_from_due_date	0.011508	0.571713
Num_of_Delayed_Payment	0.002837	0.010121
Changed_Credit_Limit	0.006562	0.458549
Num_Credit_Inquiries	1.000000	0.013387
Outstanding_Debt	0.013387	1.000000
Credit_Utilization_Ratio	0.000145	-0.071165
Total_EMI_per_month	-0.007031	-0.004856
Amount_invested_monthly	-0.000614	-0.015877
Monthly_Balance	0.001052	0.001346

	Credit_Utilization_Ratio	Total_EMI_per_month \
Age	0.002774	0.000662
Annual_Income	0.010316	-0.000248
Monthly_Inhand_Salary	0.173192	0.007264
Num_Bank_Accounts	-0.001365	-0.001433
Num_Credit_Card	-0.003341	0.000768
Interest_Rate	-0.000359	0.002517
Num_of_Loan	-0.003967	0.001554
Delay_from_due_date	-0.063796	-0.003889
Num_of_Delayed_Payment	0.000866	0.001120
Changed_Credit_Limit	-0.048862	-0.001840
Num_Credit_Inquiries	0.000145	-0.007031
Outstanding_Debt	-0.071165	-0.004856
Credit_Utilization_Ratio	1.000000	0.001389
Total_EMI_per_month	0.001389	1.000000
Amount_invested_monthly	0.005161	0.000172
Monthly_Balance	-0.004242	-0.000258

	Amount_invested_monthly	Monthly_Balance
Age	0.003797	0.001052
Annual_Income	-0.005318	0.000818
Monthly_Inhand_Salary	0.061485	-0.000806
Num_Bank_Accounts	0.003233	0.001065
Num_Credit_Card	-0.002072	0.001276
Interest_Rate	-0.001203	0.001256
Num_of_Loan	-0.000861	-0.000033
Delay_from_due_date	-0.012557	0.004454
Num_of_Delayed_Payment	0.001280	0.000931
Changed_Credit_Limit	-0.011704	-0.002182

```
Num_Credit_Inquiries                    -0.000614         0.001052
Outstanding_Debt                        -0.015877         0.001346
Credit_Utilization_Ratio                 0.005161        -0.004242
Total_EMI_per_month                      0.000172        -0.000258
Amount_invested_monthly                  1.000000         0.001725
Monthly_Balance                          0.001725         1.000000
```

[ ]:
```
categorical_columns = df.select_dtypes(include='category').columns
object_columns = df.select_dtypes(include='object').columns
```

[ ]:
```
categorical_columns
```

[ ]:
```
Index(['Month', 'Name', 'Occupation', 'Credit_Mix', 'Payment_of_Min_Amount'],
dtype='object')
```

[ ]:
```
object_columns
```

[ ]:
```
Index(['ID', 'Customer_ID', 'SSN', 'Type_of_Loan', 'Credit_History_Age',
       'Payment_Behaviour'],
      dtype='object')
```

[ ]:
```
df['Occupation']=df['Occupation'].replace('_______','Other')
```

Null values in Occupation is replaced by other

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

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

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

[ ]:
```
Credit_Mix
Standard    36479
Good        24337
_           20195
Bad         18989
Name: count, dtype: int64
```

[ ]:
```
df['Credit_Mix']=df['Credit_Mix'].replace('_','Unknown')
```

Null values in Credit Mix is replaced by Unknown

[ ]:
```
df.isnull().sum().sort_values(ascending=False).to_frame('Missing_Values')
```

[ ]:

	Missing_Values
Monthly_Inhand_Salary	15002
Type_of_Loan	11408
Name	9985
Credit_History_Age	9030
Num_of_Delayed_Payment	7002
Amount_invested_monthly	4479
Changed_Credit_Limit	2091
Num_Credit_Inquiries	1965
Monthly_Balance	1200
Num_Bank_Accounts	0
Num_Credit_Card	0
Payment_Behaviour	0
Month	0
Total_EMI_per_month	0
Payment_of_Min_Amount	0
Age	0
Credit_Utilization_Ratio	0
Outstanding_Debt	0
Credit_Mix	0
SSN	0
Occupation	0
Annual_Income	0
Delay_from_due_date	0
Customer_ID	0
Num_of_Loan	0
Interest_Rate	0
ID	0

[ ]:
```
df['Monthly_Inhand_Salary'].mean()
```

[ ]:
```
4194.170849592996
```

[ ]:
```
df['Monthly_Inhand_Salary']=df['Monthly_Inhand_Salary'].
↪fillna(df['Annual_Income']/12)
```

Null values in Monthly Inhand Salary is replaaced by Annual_Income column

[ ]:
```
df['Monthly_Inhand_Salary'].isnull().sum()
```

[ ]:
```
0
```

[ ]:
```
df['Type_of_Loan']=df['Type_of_Loan'].astype('object')
```

[ ]:
```
df['Type_of_Loan']=df['Type_of_Loan'].fillna('Unknown')
```

Null values in Type of loan column is replaced by Unknown

[ ]:
```
df['Type_of_Loan'].isnull().sum()
```

[ ]:
```
0
```

[ ]:
```
df["Name"]=df["Name"].astype('object')
df["Name"]=df["Name"].fillna("Unknown")
```

Null values in Name column is replaced by Unknown

[ ]:
```
df["Name"].isnull().sum()
```

[ ]:
```
0
```

[ ]:
```
df['Credit_History_Age']=df['Credit_History_Age'].fillna("Unknown")
```

Null values in Credit_History_Age column is replaced by Unknown

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

[ ]:
```
            ID Customer_ID Month    Name  Age          SSN Occupation  \
67284  0x1a03e   CUS_0x6d63   May   Mattx   29  609-98-6953     Lawyer

       Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  \
67284        8393.96             780.496667                  5

       Num_Credit_Card  Interest_Rate  Num_of_Loan  \
67284                5             32            2

                                            Type_of_Loan  Delay_from_due_date  \
67284  Debt Consolidation Loan, and Debt Consolidatio…                   14

       Num_of_Delayed_Payment  Changed_Credit_Limit  Num_Credit_Inquiries  \
67284                    17.0                 18.84                  12.0
```

```
      Credit_Mix  Outstanding_Debt  Credit_Utilization_Ratio  \
67284   Standard           1402.78                 29.420349

          Credit_History_Age Payment_of_Min_Amount  Total_EMI_per_month  \
67284  14 Years and 2 Months                    NM             8.025108

       Amount_invested_monthly                 Payment_Behaviour  \
67284                22.791812  High_spent_Medium_value_payments

       Monthly_Balance
67284       297.232747
```

[ ]:
```
df['Num_of_Delayed_Payment'].median()
```

[ ]:
```
14.0
```

[ ]:
```
df['Num_of_Delayed_Payment']=df['Num_of_Delayed_Payment'].
    ↪fillna(df['Num_of_Delayed_Payment'].median())
```

Null values in Num_of_Delayed_Payment column is replaced by Median value of Num_of_Delayed_Payment

[ ]:
```
df['Amount_invested_monthly'].median()
```

[ ]:
```
135.9256815
```

[ ]:
```
df['Amount_invested_monthly']=df['Amount_invested_monthly'].
    ↪fillna(df['Amount_invested_monthly'].median())
```

Null values in Amount_invested_monthly column is replaced by Median value of Amount_invested_monthly

[ ]:
```
df['Changed_Credit_Limit'].unique()
```

[ ]:
```
array([11.27,   nan,  6.27, …, 27.38, 25.16, 21.17])
```

[ ]:
```
df['Changed_Credit_Limit']=df['Changed_Credit_Limit'].
    ↪fillna(df['Changed_Credit_Limit'].mean())
```

Null values in Changed_Credit_Limit column is replaced by Mean value of Changed_Credit_Limit

[ ]:
```
df['Num_Credit_Inquiries'].median()
```

[ ]:
```
6.0
```

[ ]:
```
df['Num_Credit_Inquiries']=df['Num_Credit_Inquiries'].
    ↪fillna(df['Num_Credit_Inquiries'].median())
```

Null values in Num_Credit_Inquiries column is replaced by Median value of Num_Credit_Inquiries

[ ]:
```
df['Monthly_Balance'].median()
```

[ ]:
```
336.7191898
```

[ ]:
```
df['Monthly_Balance']=df['Monthly_Balance'].fillna(df['Monthly_Balance'].
 ↪median())
```

Null values in Monthly_Balance column is replaced by Median value of Monthly_Balance

[ ]:
```
df.isnull().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                0
Delay_from_due_date         0
Num_of_Delayed_Payment      0
Changed_Credit_Limit        0
Num_Credit_Inquiries        0
Credit_Mix                  0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Credit_History_Age          0
Payment_of_Min_Amount       0
Total_EMI_per_month         0
Amount_invested_monthly     0
Payment_Behaviour           0
Monthly_Balance             0
dtype: int64
```

2 Outliers

[ ]:
```
numeric_columns = df.select_dtypes(include='number').columns
```

[ ]:
```
numeric_columns
```

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

[ ]:
```
# Detecting Outliers using IQR method

for i in numeric_columns:
  Q1=np.quantile(df[i],0.25)
  Q3=np.quantile(df[i],0.75)
  IQR=Q3-Q1
  upper_limit=Q3+(1.5*IQR)
  lower_limit=Q1-(1.5*IQR)
  Outliers=df[(df[i]>upper_limit) | (df[i]<lower_limit)]
   # Print column and outlier information
  print(f'Column: {i}')
  print(f'Q1: {Q1}')
  print(f'Q3: {Q3}')
  print(f'IQR: {IQR}')
  print(f'Lower Bound (LB): {lower_limit}')
  print(f'Upper Bound (UB): {upper_limit}')
  print(f'Number of outliers: {Outliers.shape[0]}')
  print()  # Blank line for better readability between columns
```

```
Column: Age
Q1: 24.0
Q3: 42.0
IQR: 18.0
Lower Bound (LB): -3.0
Upper Bound (UB): 69.0
Number of outliers: 2781

Column: Annual_Income
Q1: 19457.5
Q3: 72790.92
IQR: 53333.42
Lower Bound (LB): -60542.630000000005
Upper Bound (UB): 152791.05
Number of outliers: 2783

Column: Monthly_Inhand_Salary
Q1: 1625.793333
Q3: 5971.78000025
```

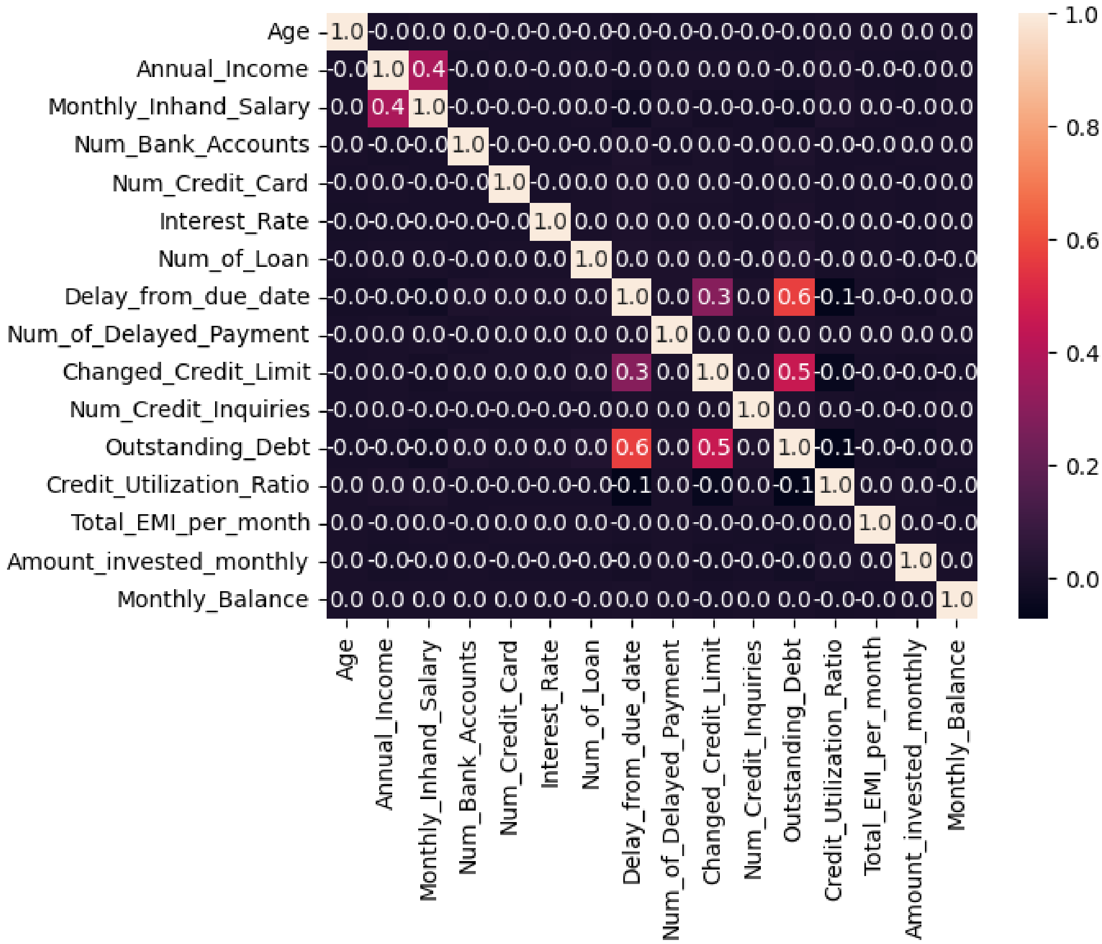
```
IQR: 4345.986667249999
Lower Bound (LB): -4893.186667874999
Upper Bound (UB): 12490.760001125
Number of outliers: 2141

Column: Num_Bank_Accounts
Q1: 3.0
Q3: 7.0
IQR: 4.0
Lower Bound (LB): -3.0
Upper Bound (UB): 13.0
Number of outliers: 1315

Column: Num_Credit_Card
Q1: 4.0
Q3: 7.0
IQR: 3.0
Lower Bound (LB): -0.5
Upper Bound (UB): 11.5
Number of outliers: 2271

Column: Interest_Rate
Q1: 8.0
Q3: 20.0
IQR: 12.0
Lower Bound (LB): -10.0
Upper Bound (UB): 38.0
Number of outliers: 2034

Column: Num_of_Loan
Q1: 1.0
Q3: 5.0
IQR: 4.0
Lower Bound (LB): -5.0
Upper Bound (UB): 11.0
Number of outliers: 4348

Column: Delay_from_due_date
Q1: 10.0
Q3: 28.0
IQR: 18.0
Lower Bound (LB): -17.0
Upper Bound (UB): 55.0
Number of outliers: 4002

Column: Num_of_Delayed_Payment
Q1: 9.0
Q3: 18.0
```

```
IQR: 9.0
Lower Bound (LB): -4.5
Upper Bound (UB): 31.5
Number of outliers: 736

Column: Changed_Credit_Limit
Q1: 5.42
Q3: 14.66
IQR: 9.24
Lower Bound (LB): -8.44
Upper Bound (UB): 28.52
Number of outliers: 1177

Column: Num_Credit_Inquiries
Q1: 3.0
Q3: 9.0
IQR: 6.0
Lower Bound (LB): -6.0
Upper Bound (UB): 18.0
Number of outliers: 1650

Column: Outstanding_Debt
Q1: 566.0725
Q3: 1945.9625
IQR: 1379.89
Lower Bound (LB): -1503.7625
Upper Bound (UB): 4015.7975
Number of outliers: 5272

Column: Credit_Utilization_Ratio
Q1: 28.05256656
Q3: 36.4966630525
IQR: 8.444096492499998
Lower Bound (LB): 15.386421821250002
Upper Bound (UB): 49.162807791249996
Number of outliers: 4

Column: Total_EMI_per_month
Q1: 30.30666049
Q3: 161.2242491
IQR: 130.91758861
Lower Bound (LB): -166.069722425
Upper Bound (UB): 357.600632015
Number of outliers: 6795

Column: Amount_invested_monthly
Q1: 77.01741385
Q3: 255.03869785
```

```
IQR: 178.021284
Lower Bound (LB): -190.01451215
Upper Bound (UB): 522.07062385
Number of outliers: 10866

Column: Monthly_Balance
Q1: 270.89342695
Q3: 467.67059704999997
IQR: 196.77717009999998
Lower Bound (LB): -24.272328200000004
Upper Bound (UB): 762.8363522
Number of outliers: 7873
```

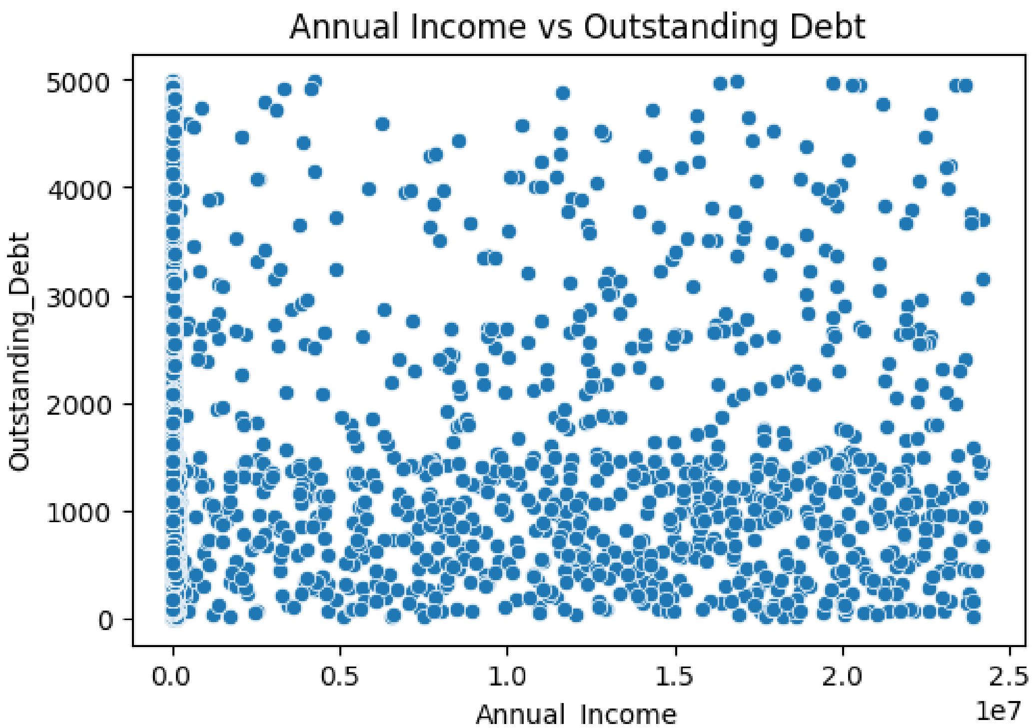
Insight : Outliers may reflect important variations within the population, it's generally recommended to retain them in the dataset rather than remove them.

[ ]:
```
sns.heatmap(df[numeric_columns].corr(),annot=True,fmt='.1f')
plt.show()
```



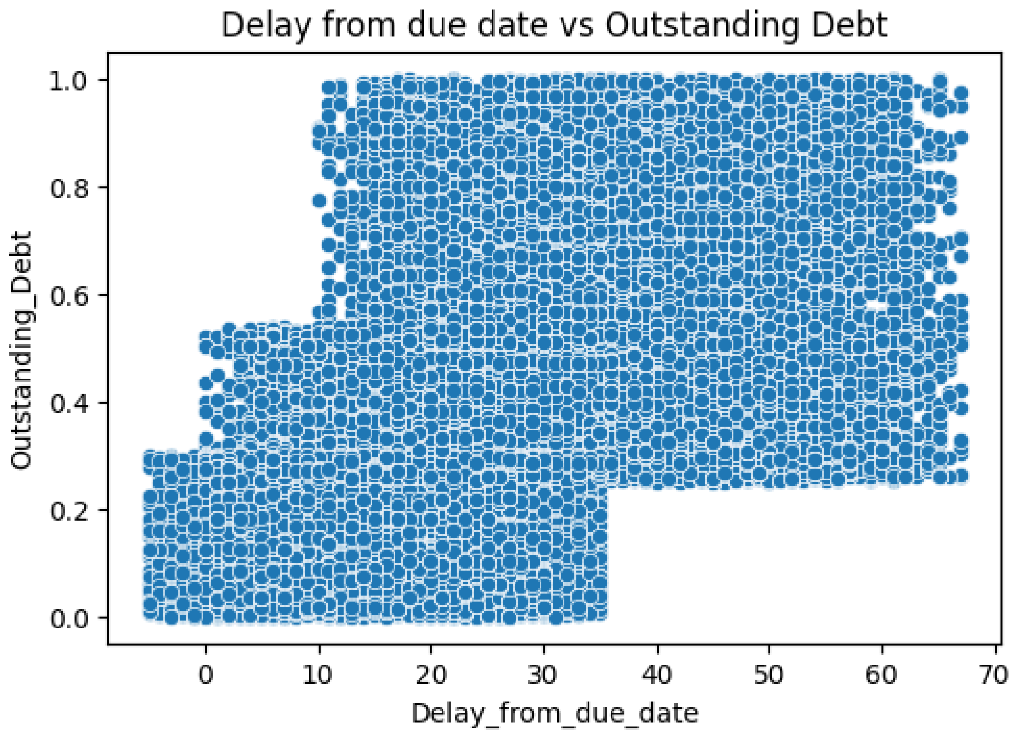
Insights: * Monthly Inhand Salary is highly correlated with Annual Income * Outstanding Debt is correlated with delay from due date and changed credit limit

[ ]:
```
plt.figure(figsize=(6,4))
sns.scatterplot(x='Annual_Income', y='Outstanding_Debt', data=df)
plt.title('Annual Income vs Outstanding Debt')
plt.show()
```



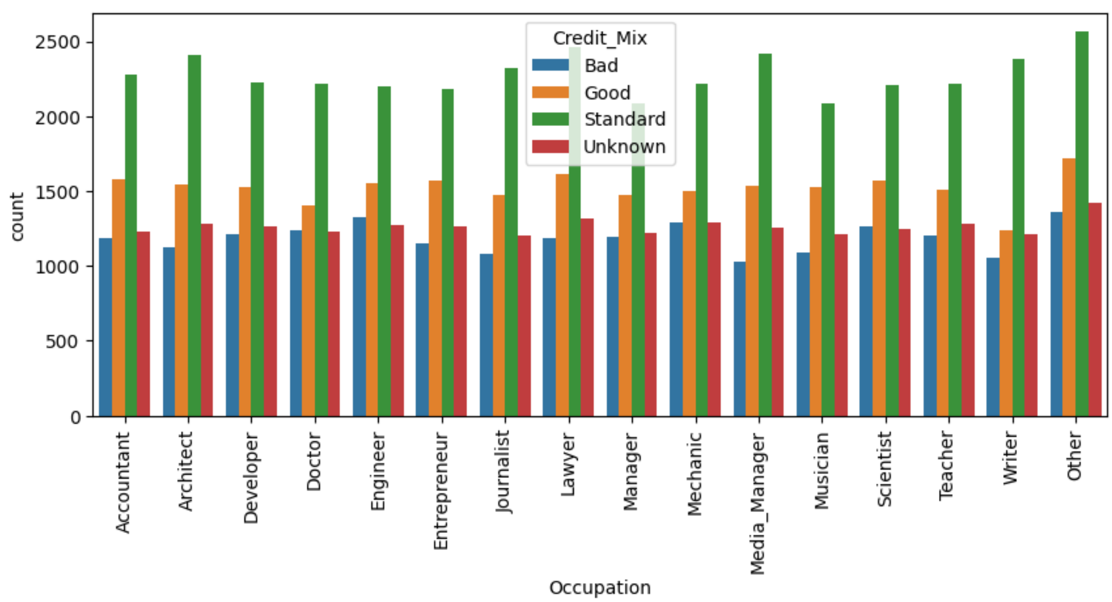
There is no correlation between Annual Income and Outstanding Debt

[ ]:
```
plt.figure(figsize=(6,4))
sns.scatterplot(x='Delay_from_due_date', y='Outstanding_Debt', data=df)
plt.title('Delay from due date vs Outstanding Debt')
plt.show()
```



There is a correlation between Delay from due date and Outstanding Debt

[ ]:
```
plt.figure(figsize=(10,4))
sns.countplot(x='Occupation',data=df,hue='Credit_Mix')
plt.xticks(rotation=90)
plt.show()
```

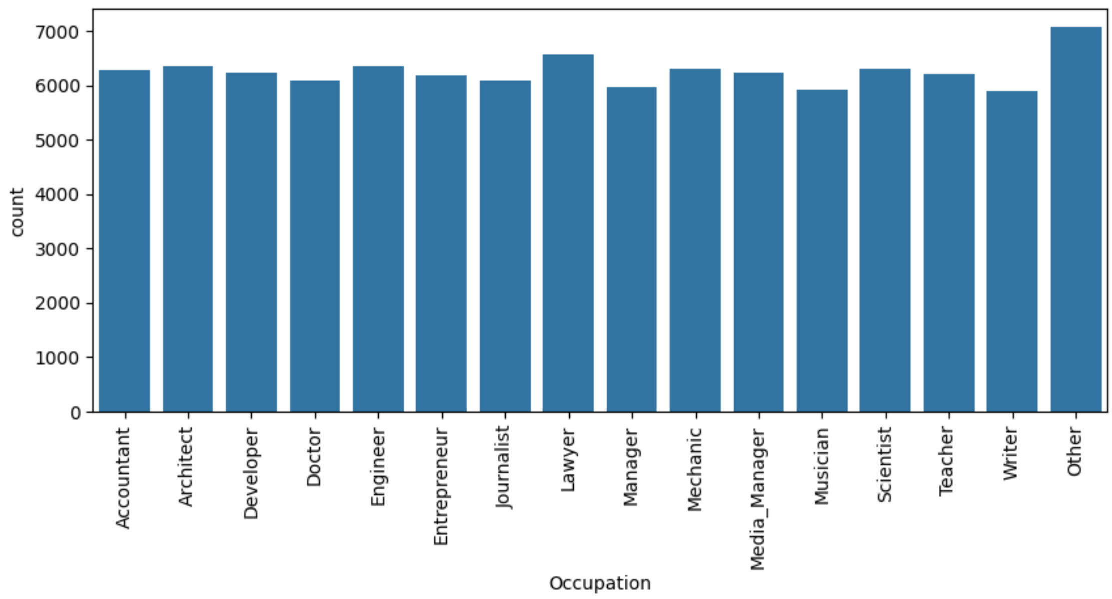


Customers with “Unknown” occupation have the most standard or balanced credit mix, followed closely by customers in the “Lawyer” occupation category.

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

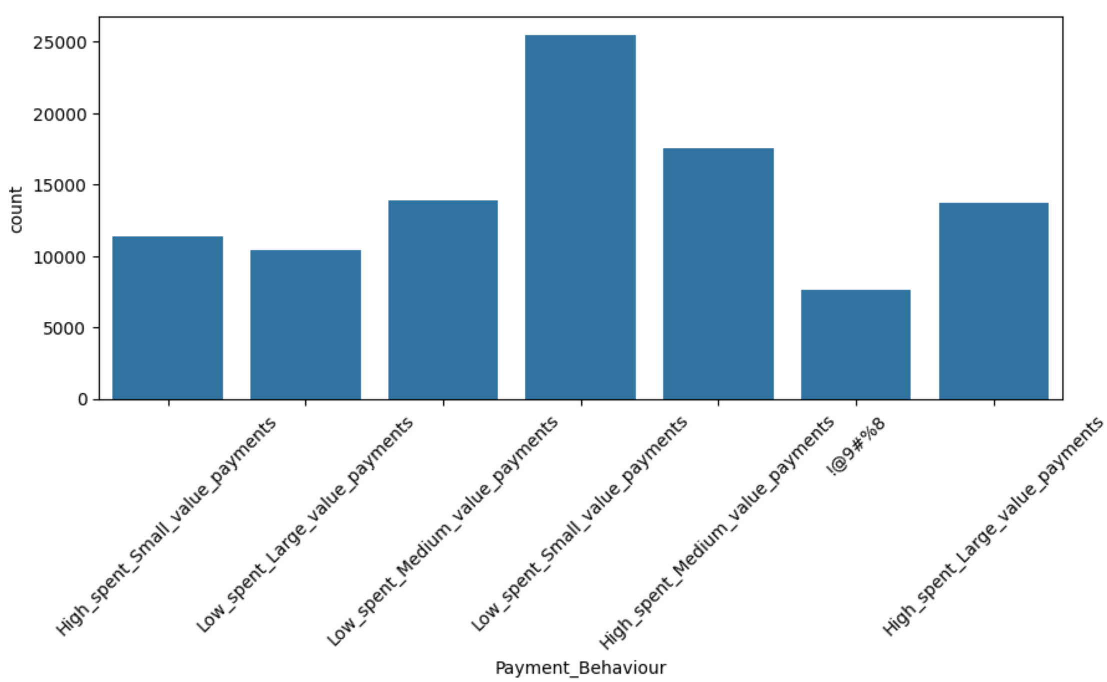
[ ]:
```
Occupation
Other            7062
Lawyer           6575
Architect        6355
Engineer         6350
Scientist        6299
Mechanic         6291
Accountant       6271
Developer        6235
Media_Manager    6232
Teacher          6215
Entrepreneur     6174
Doctor           6087
Journalist      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

[ ]:
```
plt.figure(figsize=(10,4))
sns.countplot(x='Occupation',data=df)
plt.xticks(rotation=90)
plt.show()
```



dataset contains a large number of entries with "Unknown" occupation (7062) and "Lawyer" occupation (6575)

[ ]:
```
plt.figure(figsize=(10,4))
sns.countplot(x='Payment_Behaviour',data=df)
plt.xticks(rotation=45)
plt.show()
```



Insights: Small-value payments with low spending impact payment behavior One important finding in the dataset is that consumers who frequently make small-value, low-spend payments have a big influence on how people pay overall. These clients regularly pay on time, which helps to foster good credit behavior.

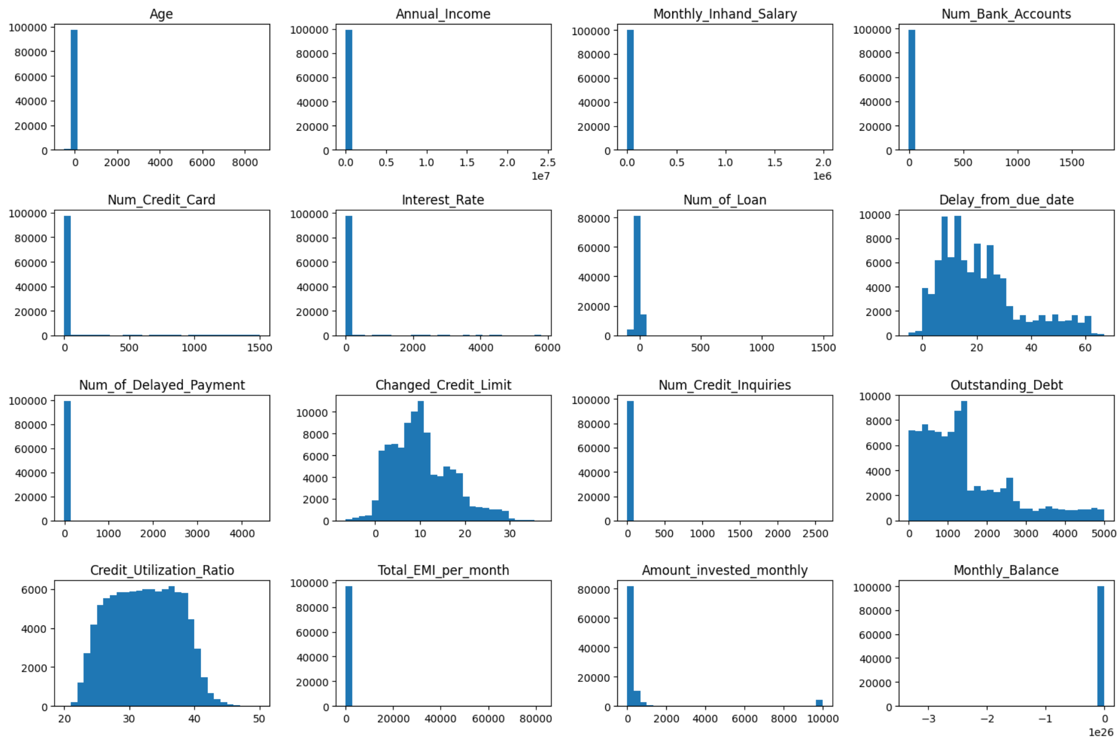
High-Spend Medium-Value Payments Affect Payment Patterns as Well: Customers that make high-spending, medium-value payments are likewise a significant category, even if they do so less frequently than small-value payments. They frequently exhibit a responsible credit utilization and payback pattern. Their payment habit is often regular and modest, which serves to bolster their good creditworthiness.

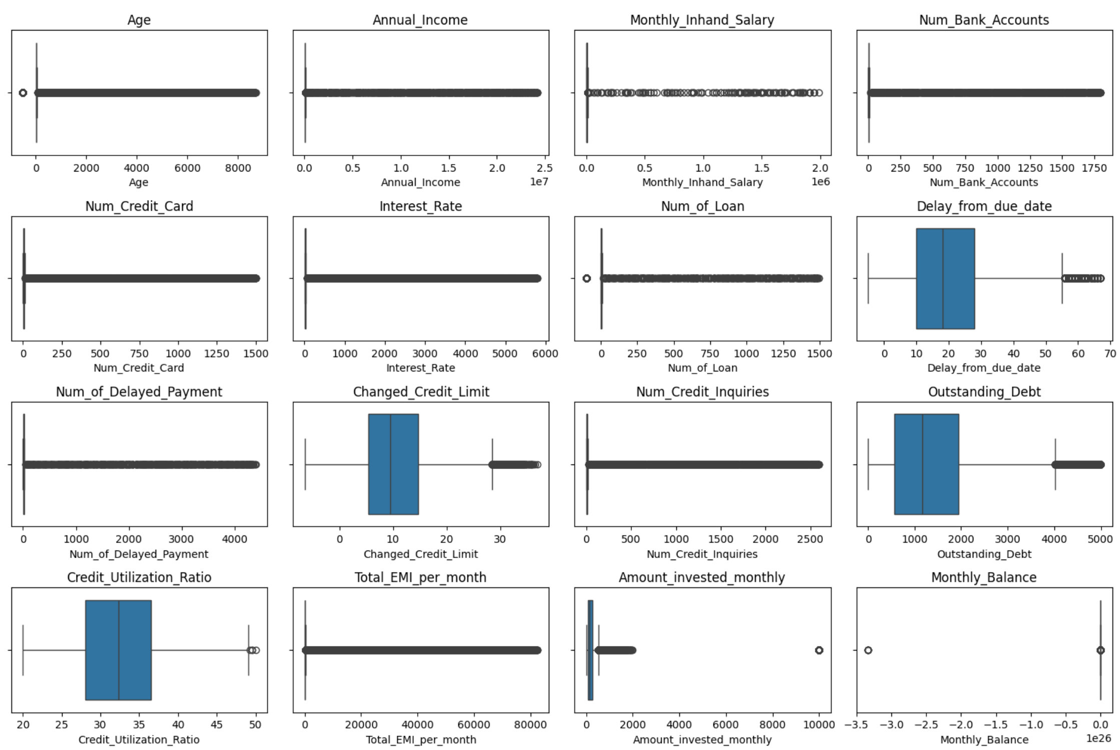
Effect on Credit Score: Regular small-value payments show reliable payment practices, which lowers the risk of late payments. Conversely, clients paying for medium-value items, nonetheless

[ ]:
```
df['Payment_Behaviour'].value_counts().head(6)
```

[ ]:
```
Payment_Behaviour
Low_spent_Small_value_payments      25513
High_spent_Medium_value_payments    17540
Low_spent_Medium_value_payments     13861
High_spent_Large_value_payments     13721
High_spent_Small_value_payments     11340
Low_spent_Large_value_payments      10425
Name: count, dtype: int64
```

[ ]:
```
# Histograms for numeric columns
numeric_columns = df.select_dtypes(include='number').columns

df[numeric_columns].hist(figsize=(15, 10), bins=30, grid=False)
plt.tight_layout()
plt.show()

# Box plots for outlier detection
plt.figure(figsize=(15, 10))
for i, column in enumerate(numeric_columns, 1):
    plt.subplot(4, 4, i)
    sns.boxplot(x=df[column])
    plt.title(column)
plt.tight_layout()
plt.show()
```


Age
Annual_Income
Monthly_Inhand_Salary
Num_Bank_Accounts
Num_Credit_Card
Interest_Rate
Num_of_Loan
Delay_from_due_date
Num_of_Delayed_Payment
Changed_Credit_Limit
Num_Credit_Inquiries
Outstanding_Debt
Credit_Utilization_Ratio
Total_EMI_per_month
Amount_invested_monthly
Monthly_Balance


Age
Annual_Income
Monthly_Inhand_Salary
Num_Bank_Accounts
Num_Credit_Card
Interest_Rate
Num_of_Loan
Delay_from_due_date
Num_of_Delayed_Payment
Changed_Credit_Limit
Num_Credit_Inquiries
Outstanding_Debt
Credit_Utilization_Ratio
Total_EMI_per_month
Amount_invested_monthly
Monthly_Balance

Credit Utilization Ratio Exhibits a Normal Distribution and Other Numerical Features Are Skewed.

3 Feature Engineering:

[ ]:
```
# This ratio measures the portion of a customer's income used to pay debts. A␣
↪lower value indicates good credit behavior.

df['Debt_to_Income_Ratio'] = df['Outstanding_Debt'] / df['Annual_Income']
```

[ ]:
```
# This feature sums the EMI and monthly investment to reflect the total␣
↪outgoing payments.

df['Avg_Monthly_Debt_Payments'] = df['Total_EMI_per_month'] +␣
↪df['Amount_invested_monthly']
```

[ ]:
```
# Frequent credit inquiries combined with high interest rates can be a sign of␣
↪credit risk.

df['Credit_Inquiry_Impact'] = df['Num_Credit_Inquiries'] * df['Interest_Rate']
```

[ ]:
```
# This measures how many loans a customer has compared to their income.

df['Loan_to_Income_Ratio'] = df['Num_of_Loan'] / df['Annual_Income']
```

[ ]:
```
# Function to convert 'Years and Months' to total months
def convert_to_months(value):
    if value == 'Unknown':
        return np.nan  # Replace 'Unknown' with NaN
    else:
        # Split the value into years and months
        parts = value.split('and')
        years = int(parts[0].strip().split()[0])  # Extract years part
        months = int(parts[1].strip().split()[0])  # Extract months part
        total_months = years * 12 + months  # Convert to total months
        return total_months

# Apply the function to the 'Credit_History_Age' column
df['Credit_History_Age_Months'] = df['Credit_History_Age'].
↪apply(convert_to_months)
```

[ ]:
```
df['Credit_History_Age_Months'].isnull().sum()
```

[ ]: 9030

[ ]:
```
df['Credit_History_Age_Months']=df['Credit_History_Age_Months'].
↪fillna(df['Credit_History_Age_Months'].median())
```

[ ]:
```
df['Credit_History_Age_Months'].isnull().sum()
```

[ ]: 0

3.1 Standardizing the Features

[ ]:
```
from sklearn.preprocessing import MinMaxScaler

# Select relevant columns for standardization
features_to_scale = ['Num_of_Delayed_Payment', 'Credit_Utilization_Ratio',␣
 ↪'Credit_History_Age_Months',
                     'Num_Credit_Inquiries', 'Outstanding_Debt',␣
 ↪'Annual_Income', 'Debt_to_Income_Ratio']

# Initialize scaler
scaler = MinMaxScaler()

# Apply scaling
df[features_to_scale] = scaler.fit_transform(df[features_to_scale])
```

Explanation of the Methodology

Payment History: Delayed payments are a critical indicator of poor credit behavior. A customer with fewer delayed payments will receive a higher score.

Credit Utilization Ratio: High credit utilization suggests reliance on credit, which is risky. Keeping this ratio below 30% is considered good.

Credit History Length: A longer history demonstrates experience managing credit.

Credit Inquiries: Frequent credit inquiries can signal financial distress, so fewer inquiries lead to a higher score.

Debt-to-Income Ratio: A lower debt-to-income ratio means the customer can manage their debt better, improving the score.

[ ]:
```
# Define weighting scheme
weights = {
    'Num_of_Delayed_Payment': 0.35,
    'Credit_Utilization_Ratio': 0.30,
    'Credit_History_Age_Months': 0.15,
    'Num_Credit_Inquiries': 0.05,
    'Debt_to_Income_Ratio': 0.05
}

# Calculate credit score
df['Credit_Score'] = (
    df['Num_of_Delayed_Payment'] * weights['Num_of_Delayed_Payment'] +
    df['Credit_Utilization_Ratio'] * weights['Credit_Utilization_Ratio'] +
```

```
    df['Credit_History_Age_Months'] * weights['Credit_History_Age_Months'] +
    df['Num_Credit_Inquiries'] * weights['Num_Credit_Inquiries'] +
    df['Debt_to_Income_Ratio'] * weights['Debt_to_Income_Ratio']
)

# Scale the credit score to a 300-850 range
df['Credit_Score'] = df['Credit_Score'] * (850 - 300) + 300
```

Justification for the weightage allocation in the credit score calculation:

Num_of_Delayed_Payment (35%): This is one of the most significant factors because delayed payments are strong indicators of poor credit behavior. A higher number of delayed payments suggests difficulty in managing debts and is typically a red flag for lenders. Thus, it receives the highest weight, aligning with real-world credit scoring systems where payment history often has a large impact.

Credit_Utilization_Ratio (30%): This ratio reflects how much of the available credit is being used. Higher credit utilization can indicate over-reliance on credit, which is a risk for lenders. A well-managed credit utilization (typically below 30%) suggests financial discipline, which is why it has a significant weight. It's a crucial measure of how responsibly a customer handles their available credit.

Credit_History_Age_Months (15%): The length of credit history is important because it shows how experienced the customer is with managing credit. A longer history typically suggests more reliability and stability. Although important, it receives a lower weight than payment history and utilization, as it doesn't directly indicate current financial behavior.

Num_Credit_Inquiries (5%): Credit inquiries can signal credit-seeking behavior. A high number of inquiries in a short period may indicate financial stress, but this alone doesn't necessarily indicate poor credit behavior. Thus, it receives a lower weight since it's more of a cautionary indicator rather than a decisive factor.

Debt_to_Income_Ratio (5%): The debt-to-income ratio compares a customer's monthly debt payments to their monthly income. A lower ratio suggests better ability to manage debt. While important, it's given a smaller weight since it's often seen as a supplemental factor that provides context to other key metrics, such as payment history and credit utilization.

[ ]:
```
df['Credit_Score'].unique()
```

[ ]:
```
array([393.75449389, 412.81674286, 403.99095137, …, 495.4279587 ,
       454.0874448 , 456.99056013])
```

[ ]:
```
# Step 1: Aggregate data by Customer_ID
customer_level_data = df.groupby('Customer_ID').agg({
    'Num_of_Delayed_Payment': 'sum',  # or 'mean', depending on how you want to
 ↪aggregate
    'Credit_Utilization_Ratio': 'mean',
    'Credit_History_Age_Months': 'mean',
    'Num_Credit_Inquiries': 'sum',  # or 'mean'
    'Debt_to_Income_Ratio': 'mean'   # or another aggregation method
```

```python
}).reset_index()

# Step 2: Define weighting scheme
weights = {
    'Num_of_Delayed_Payment': 0.35,
    'Credit_Utilization_Ratio': 0.30,
    'Credit_History_Age_Months': 0.15,
    'Num_Credit_Inquiries': 0.05,
    'Debt_to_Income_Ratio': 0.05
}

# Step 3: Calculate the credit score for each customer
customer_level_data['Credit_Score'] = (
    customer_level_data['Num_of_Delayed_Payment'] * 
 ↪weights['Num_of_Delayed_Payment'] +
    customer_level_data['Credit_Utilization_Ratio'] * 
 ↪weights['Credit_Utilization_Ratio'] +
    customer_level_data['Credit_History_Age_Months'] * 
 ↪weights['Credit_History_Age_Months'] +
    customer_level_data['Num_Credit_Inquiries'] * 
 ↪weights['Num_Credit_Inquiries'] +
    customer_level_data['Debt_to_Income_Ratio'] * 
 ↪weights['Debt_to_Income_Ratio']
)

# Step 4: Scale the credit score to a 300-850 range
# Assuming the unscaled scores range is normalized between 0 and 1 for 
 ↪calculation
customer_level_data['Credit_Score'] = (
    customer_level_data['Credit_Score'] * (850 - 300) + 300
)

# Display the final customer-level data with credit scores
print(customer_level_data[['Customer_ID', 'Credit_Score']])
```

	Customer_ID	Credit_Score
0	CUS_0x1000	412.388700
1	CUS_0x1009	508.987295
2	CUS_0x100b	448.679415
3	CUS_0x1011	407.812976
4	CUS_0x1013	413.730389
…	…	…
12495	CUS_0xff3	420.165701
12496	CUS_0xff4	421.261051
12497	CUS_0xff6	436.422762
12498	CUS_0xffc	424.316112
12499	CUS_0xffd	436.954438

```
[12500 rows x 2 columns]
```

[ ]:
```
df['Month'].unique()
```

[ ]:
```
['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August']
Categories (8, object): ['April', 'August', 'February', 'January', 'July',
'June', 'March', 'May']
```

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

[ ]:
```
data['Month'] = data['Month'].astype(str)
```

[ ]:
```
data['Month'] = pd.to_datetime(data['Month'] + ' 2020', format='%B %Y')
```

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

[ ]:
```
<DatetimeArray>
['2020-01-01 00:00:00', '2020-02-01 00:00:00', '2020-03-01 00:00:00',
 '2020-04-01 00:00:00', '2020-05-01 00:00:00', '2020-06-01 00:00:00',
 '2020-07-01 00:00:00', '2020-08-01 00:00:00']
Length: 8, dtype: datetime64[ns]
```

[ ]:
```
# Define a reference date
reference_date = pd.to_datetime('2020-08-01')

# Calculate the last 3 months and last 6 months
last_3_months_start = reference_date - pd.DateOffset(months=3)
last_6_months_start = reference_date - pd.DateOffset(months=6)

#Filter the DataFrame for last 3 months
last_3_months_data = data[(data['Month'] >= last_3_months_start) &␣
 ↪(data['Month'] <= reference_date)]

# Filter the DataFrame for last 6 months
last_6_months_data = data[(data['Month'] >= last_6_months_start) &␣
 ↪(data['Month'] <= reference_date)]



# Step 3: Aggregate the last 3 months data
last_3_months_agg = last_3_months_data.groupby('Customer_ID').agg({
    'Num_of_Delayed_Payment': 'sum',  # Or 'mean', depending on how you want to␣
 ↪aggregate
    'Credit_Utilization_Ratio': 'mean',
    'Credit_History_Age_Months': 'mean',
    'Num_Credit_Inquiries': 'sum',  # Or 'mean'
    'Debt_to_Income_Ratio': 'mean'   # Or another aggregation method
```

```python
}).reset_index()

# Calculate the Credit Score for last 3 months
last_3_months_agg['Credit_Score'] = (
    last_3_months_agg['Num_of_Delayed_Payment'] *␣
 ↪weights['Num_of_Delayed_Payment'] +
    last_3_months_agg['Credit_Utilization_Ratio'] *␣
 ↪weights['Credit_Utilization_Ratio'] +
    last_3_months_agg['Credit_History_Age_Months'] *␣
 ↪weights['Credit_History_Age_Months'] +
    last_3_months_agg['Num_Credit_Inquiries'] * weights['Num_Credit_Inquiries']␣
 ↪+
    last_3_months_agg['Debt_to_Income_Ratio'] * weights['Debt_to_Income_Ratio']
)

# Scale the Credit Score to a 300-850 range for last 3 months
last_3_months_agg['Credit_Score_3M'] = (
    last_3_months_agg['Credit_Score'] * (850 - 300) + 300
)

# Step 4: Aggregate the last 6 months data
last_6_months_agg = last_6_months_data.groupby('Customer_ID').agg({
    'Num_of_Delayed_Payment': 'sum',
    'Credit_Utilization_Ratio': 'mean',
    'Credit_History_Age_Months': 'mean',
    'Num_Credit_Inquiries': 'sum',
    'Debt_to_Income_Ratio': 'mean'
}).reset_index()

# Calculate the Credit Score for last 6 months
last_6_months_agg['Credit_Score'] = (
    last_6_months_agg['Num_of_Delayed_Payment'] *␣
 ↪weights['Num_of_Delayed_Payment'] +
    last_6_months_agg['Credit_Utilization_Ratio'] *␣
 ↪weights['Credit_Utilization_Ratio'] +
    last_6_months_agg['Credit_History_Age_Months'] *␣
 ↪weights['Credit_History_Age_Months'] +
    last_6_months_agg['Num_Credit_Inquiries'] * weights['Num_Credit_Inquiries']␣
 ↪+
    last_6_months_agg['Debt_to_Income_Ratio'] * weights['Debt_to_Income_Ratio']
)

# Scale the Credit Score to a 300-850 range for last 6 months
last_6_months_agg['Credit_Score_6M'] = (
    last_6_months_agg['Credit_Score'] * (850 - 300) + 300
)
```

[ ]:
```
#Step 6: Combine results
# Merge the original DataFrame with the last 3 and last 6 months aggregated data
final_results = customer_level_data[['Customer_ID', 'Credit_Score']].
 ↪drop_duplicates()
final_results = final_results.merge(
    last_3_months_agg[['Customer_ID', 'Credit_Score']],
    on='Customer_ID',
    suffixes=('', '_last_3_months'),
    how='left'
).rename(columns={'Credit_Score_last_3_months': 'Credit_Score_Last_3_Months'})

final_results = final_results.merge(
    last_6_months_agg[['Customer_ID', 'Credit_Score']],
    on='Customer_ID',
    suffixes=('', '_last_6_months'),
    how='left'
).rename(columns={'Credit_Score_last_6_months': 'Credit_Score_Last_6_Months'})
```

[ ]:
```
final_results.head(5)
```

[ ]:
```
  Customer_ID  Credit_Score  Credit_Score_Last_3_Months  \
0  CUS_0x1000    412.388700                    0.212094
1  CUS_0x1009    508.987295                    0.282936
2  CUS_0x100b    448.679415                    0.198754
3  CUS_0x1011    407.812976                    0.189901
4  CUS_0x1013    413.730389                    0.215571

   Credit_Score_Last_6_Months
0                    0.211917
1                    0.248571
2                    0.265420
3                    0.195085
4                    0.191743
```

[ ]:
```
# Sort by original credit score in descending order and select the top 5␣
 ↪customers
top_5_customers_by_credit_score = final_results.sort_values(by='Credit_Score',␣
 ↪ascending=False).head(5)

# Sort by recency-based credit score for the last 3 months and select the top 5␣
 ↪customers
top_5_customers_by_last_3_months = final_results.
 ↪sort_values(by='Credit_Score_Last_3_Months', ascending=False).head(5)

# Sort by recency-based credit score for the last 6 months and select the top 5␣
 ↪customers
```

```
top_5_customers_by_last_6_months = final_results.
 ↪sort_values(by='Credit_Score_Last_6_Months', ascending=False).head(5)

# Display the results
print("Top 5 Customers by Original Credit Score:")
print(top_5_customers_by_credit_score[['Customer_ID', 'Credit_Score']])

print("\nTop 5 Customers by Credit Score (Last 3 Months):")
print(top_5_customers_by_last_3_months[['Customer_ID',␣
 ↪'Credit_Score_Last_3_Months']])

print("\nTop 5 Customers by Credit Score (Last 6 Months):")
print(top_5_customers_by_last_6_months[['Customer_ID',␣
 ↪'Credit_Score_Last_6_Months']])
```

```
Top 5 Customers by Original Credit Score:
     Customer_ID  Credit_Score
1795  CUS_0x2c60    771.019115
6889  CUS_0x7755    749.730082
5191  CUS_0x5e84    746.997485
7534  CUS_0x80fd    732.436785
3274  CUS_0x4314    716.520887

Top 5 Customers by Credit Score (Last 3 Months):
      Customer_ID  Credit_Score_Last_3_Months
3274   CUS_0x4314                    0.760554
7534   CUS_0x80fd                    0.737873
2509   CUS_0x37be                    0.625469
11190  CUS_0xb6e0                    0.621415
4155   CUS_0x4f15                    0.620414

Top 5 Customers by Credit Score (Last 6 Months):
     Customer_ID  Credit_Score_Last_6_Months
5191  CUS_0x5e84                    0.804387
7534  CUS_0x80fd                    0.791502
3274  CUS_0x4314                    0.755245
990   CUS_0x205a                    0.696073
2856  CUS_0x3d40                    0.629621
```

Insights:

- Dataset contains 100000 rows and 27 columns
- Age Distribution: The average age is unexpectedly high (110.65), and the standard deviation (686.24).
- Annual Income: The mean annual income is 176,415.70 USD, but the high standard deviation (1.42 million) indicates significant variability in income levels.
- Monthly In-hand Salary: The average in-hand salary is 5,743.26 USD, but the standard

deviation of 45,814.69 USD shows a large variation.

- Number of Bank Accounts: The average number of bank accounts is 17.09, with a standard deviation of 117.40.
- Number of Credit Cards: The average number of credit cards is 22.47, with a wide range (maximum of 1,499).
- Interest Rate: The mean interest rate is 72.47%, with a very high standard deviation (466.42%).
- Delayed Payments: The average number of delayed payments is 29.74, with significant variability (standard deviation of 218). The maximum of 4,397 delayed payments suggests the presence of extreme cases.
- Changed Credit Limit: On average, the credit limit was changed by 10.39 units, with the maximum change being 36.97. Negative values (minimum -6.49) could indicate reductions in credit limits.
- Credit Inquiries: The average number of credit inquiries is 27.33, with a maximum of 2,597. The high standard deviation suggests that some individuals have a significantly higher number of inquiries.
- Outstanding Debt: The mean outstanding debt is 1,426.22 USD, with a fairly high standard deviation (1,155.13).
- Credit Utilization Ratio: The average credit utilization ratio is 32.29%, which falls within the typical 30-40% range for responsible credit usage.
- Total EMI per Month: The average EMI per month is 1,403.12 USD, with a maximum of $82,331, suggesting significant variation in the amount of debt repayments.
- Monthly Inhand Salary is highly correlated with Annual Income
- Outstanding Debt is correlated with delay from due date and changed credit limit
- Small-value payments with low spending impact payment behavior One important finding in the dataset is that consumers who frequently make small-value, low-spend payments have a big influence on how people pay overall. These clients regularly pay on time, which helps to foster good credit behavior.
- High-Spend Medium-Value Payments Affect Payment Patterns as Well: Customers that make high-spending, medium-value payments are likewise a significant category, even if they do so less frequently than small-value payments. They frequently exhibit a responsible credit utilization and payback pattern. Their payment habit is often regular and modest, which serves to bolster their good creditworthiness.
- Customers with "Unknown" occupation have the most standard or balanced credit mix, followed closely by customers in the "Lawyer" occupation category.

Recommendations:

- Encourage consumers to make frequent, small-value payments by promoting services or products with small recurring fees.

- Offer targeted incentives or rewards for timely payments, especially for small-to-medium transactions, as these contribute to overall credit health.
- Investigate why certain individuals have an unusually high number of inquiries, which may indicate fraud or reporting issues.
- Educate customers on maintaining a credit utilization ratio below 30% to improve their creditworthiness.
- Provide personalized payment schedules or financial management tools to help customers balance spending and debt repayments.
- Offer debt consolidation or financial counseling services to customers with a high correlation between outstanding debt and delayed payments to help improve their financial health.