Financial_Analysis

June 1, 2024

[1]:
```
import pandas as pd
import pymysql
```

[38]:
```
# Load the CSV file into a DataFrame
df_previous_application = pd.read_csv("previous_application.csv")
df_previous_application.head(5)
```

[38]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION
0	2030495	271877	Consumer loans	1730.430	17145.0
1	2802425	108129	Cash loans	25188.615	607500.0
2	2523466	122040	Cash loans	15060.735	112500.0
3	2819243	176158	Cash loans	47041.335	450000.0
4	1784265	202054	Cash loans	31924.395	337500.0

	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_START
0	17145.0	0.0	17145.0	SATURDAY
1	679671.0	NaN	607500.0	THURSDAY
2	136444.5	NaN	112500.0	TUESDAY
3	470790.0	NaN	450000.0	MONDAY
4	404055.0	NaN	337500.0	THURSDAY

	HOUR_APPR_PROCESS_START	…	NAME_SELLER_INDUSTRY	CNT_PAYMENT
0	15	…	Connectivity	12.0
1	11	…	XNA	36.0
2	11	…	XNA	12.0
3	7	…	XNA	12.0
4	9	…	XNA	24.0

	NAME_YIELD_GROUP	PRODUCT_COMBINATION	DAYS_FIRST_DRAWING
0	middle	POS mobile with interest	365243.0
1	low_action	Cash X-Sell: low	365243.0
2	high	Cash X-Sell: high	365243.0
3	middle	Cash X-Sell: middle	365243.0
4	high	Cash Street: high	NaN

	DAYS_FIRST_DUE	DAYS_LAST_DUE_1ST_VERSION	DAYS_LAST_DUE	DAYS_TERMINATION
0	-42.0	300.0	-42.0	-37.0
1	-134.0	916.0	365243.0	365243.0

```
2         -271.0                    59.0       365243.0         365243.0
3         -482.0                  -152.0         -182.0           -177.0
4            NaN                     NaN            NaN              NaN

  NFLAG_INSURED_ON_APPROVAL
0                       0.0
1                       1.0
2                       1.0
3                       1.0
4                       NaN

[5 rows x 37 columns]
```

[6]:
```
df_application_data= pd.read_csv("application_data.csv")
df_application_data.head(5)
```

[6]:
```
   SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
0      100002       1         Cash loans           M            N
1      100003       0         Cash loans           F            N
2      100004       0    Revolving loans           M            Y
3      100006       0         Cash loans           F            N
4      100007       0         Cash loans           M            N

  FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
0               Y             0          202500.0    406597.5      24700.5
1               N             0          270000.0   1293502.5      35698.5
2               Y             0           67500.0    135000.0       6750.0
3               Y             0          135000.0    312682.5      29686.5
4               Y             0          121500.0    513000.0      21865.5

   …  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
0  …                 0                0                 0                 0
1  …                 0                0                 0                 0
2  …                 0                0                 0                 0
3  …                 0                0                 0                 0
4  …                 0                0                 0                 0

  AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                        0.0                       0.0
1                        0.0                       0.0
2                        0.0                       0.0
3                        NaN                       NaN
4                        0.0                       0.0

   AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                         0.0                        0.0
1                         0.0                        0.0
```

```
2                          0.0                          0.0
3                          NaN                          NaN
4                          0.0                          0.0

   AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                        0.0                         1.0
1                        0.0                         0.0
2                        0.0                         0.0
3                        NaN                         NaN
4                        0.0                         0.0

[5 rows x 122 columns]
```

[7]:
```
df_previous_application.info
```

[7]:
```
<bound method DataFrame.info of          SK_ID_PREV  SK_ID_CURR
NAME_CONTRACT_TYPE  AMT_ANNUITY  \
0           2030495      271877     Consumer loans     1730.430
1           2802425      108129         Cash loans    25188.615
2           2523466      122040         Cash loans    15060.735
3           2819243      176158         Cash loans    47041.335
4           1784265      202054         Cash loans    31924.395
…               …           …                  …           …
1670209     2300464      352015     Consumer loans    14704.290
1670210     2357031      334635     Consumer loans     6622.020
1670211     2659632      249544     Consumer loans    11520.855
1670212     2785582      400317         Cash loans    18821.520
1670213     2418762      261212         Cash loans    16431.300

         AMT_APPLICATION  AMT_CREDIT  AMT_DOWN_PAYMENT  AMT_GOODS_PRICE  \
0                17145.0     17145.0               0.0          17145.0
1               607500.0    679671.0               NaN         607500.0
2               112500.0    136444.5               NaN         112500.0
3               450000.0    470790.0               NaN         450000.0
4               337500.0    404055.0               NaN         337500.0
…                    …           …                 …              …
1670209         267295.5    311400.0               0.0         267295.5
1670210          87750.0     64291.5           29250.0          87750.0
1670211         105237.0    102523.5           10525.5         105237.0
1670212         180000.0    191880.0               NaN         180000.0
1670213         360000.0    360000.0               NaN         360000.0

        WEEKDAY_APPR_PROCESS_START  HOUR_APPR_PROCESS_START  …  \
0                         SATURDAY                       15  …
1                         THURSDAY                       11  …
2                          TUESDAY                       11  …
3                           MONDAY                        7  …
```

```
4                        THURSDAY                         9  …
…                           …                          …  …
1670209                  WEDNESDAY                       12  …
1670210                    TUESDAY                       15  …
1670211                     MONDAY                       12  …
1670212                  WEDNESDAY                        9  …
1670213                     SUNDAY                       10  …

         NAME_SELLER_INDUSTRY  CNT_PAYMENT  NAME_YIELD_GROUP  \
0                Connectivity         12.0            middle
1                         XNA         36.0       low_action
2                         XNA         12.0             high
3                         XNA         12.0            middle
4                         XNA         24.0             high
…                         …           …                …
1670209             Furniture         30.0        low_normal
1670210             Furniture         12.0            middle
1670211  Consumer electronics         10.0        low_normal
1670212                   XNA         12.0        low_normal
1670213                   XNA         48.0            middle

                PRODUCT_COMBINATION  DAYS_FIRST_DRAWING DAYS_FIRST_DUE  \
0           POS mobile with interest            365243.0          -42.0
1                   Cash X-Sell: low            365243.0         -134.0
2                  Cash X-Sell: high            365243.0         -271.0
3                Cash X-Sell: middle            365243.0         -482.0
4                  Cash Street: high                 NaN            NaN
…                                …                  …             …
1670209   POS industry with interest            365243.0         -508.0
1670210   POS industry with interest            365243.0        -1604.0
1670211  POS household with interest            365243.0        -1457.0
1670212             Cash X-Sell: low            365243.0        -1155.0
1670213          Cash X-Sell: middle            365243.0        -1163.0

        DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE DAYS_TERMINATION  \
0                           300.0          -42.0            -37.0
1                           916.0       365243.0         365243.0
2                            59.0       365243.0         365243.0
3                          -152.0         -182.0           -177.0
4                             NaN            NaN              NaN
…                             …              …                …
1670209                     362.0         -358.0           -351.0
1670210                   -1274.0        -1304.0          -1297.0
1670211                   -1187.0        -1187.0          -1181.0
1670212                    -825.0         -825.0           -817.0
1670213                     247.0         -443.0           -423.0
```

```
         NFLAG_INSURED_ON_APPROVAL
0                              0.0
1                              1.0
2                              1.0
3                              1.0
4                              NaN
…                              …
1670209                        0.0
1670210                        0.0
1670211                        0.0
1670212                        1.0
1670213                        0.0

[1670214 rows x 37 columns]>
```

[8]:
```
df_application_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

[9]:
```
df_previous_application.describe()
```

[9]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	\
count	1.670214e+06	1.670214e+06	1.297979e+06	1.670214e+06	
mean	1.923089e+06	2.783572e+05	1.595512e+04	1.752339e+05	
std	5.325980e+05	1.028148e+05	1.478214e+04	2.927798e+05	
min	1.000001e+06	1.000010e+05	0.000000e+00	0.000000e+00	
25%	1.461857e+06	1.893290e+05	6.321780e+03	1.872000e+04	
50%	1.923110e+06	2.787145e+05	1.125000e+04	7.104600e+04	
75%	2.384280e+06	3.675140e+05	2.065842e+04	1.803600e+05	
max	2.845382e+06	4.562550e+05	4.180581e+05	6.905160e+06	

	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	\
count	1.670213e+06	7.743700e+05	1.284699e+06	
mean	1.961140e+05	6.697402e+03	2.278473e+05	
std	3.185746e+05	2.092150e+04	3.153966e+05	
min	0.000000e+00	-9.000000e-01	0.000000e+00	
25%	2.416050e+04	0.000000e+00	5.084100e+04	
50%	8.054100e+04	1.638000e+03	1.123200e+05	
75%	2.164185e+05	7.740000e+03	2.340000e+05	
max	6.905160e+06	3.060045e+06	6.905160e+06	

	HOUR_APPR_PROCESS_START	NFLAG_LAST_APPL_IN_DAY	RATE_DOWN_PAYMENT	\
count	1.670214e+06	1.670214e+06	774370.000000	
mean	1.248418e+01	9.964675e-01	0.079637	

```
std              3.334028e+00           5.932963e-02          0.107823
min              0.000000e+00           0.000000e+00         -0.000015
25%              1.000000e+01           1.000000e+00          0.000000
50%              1.200000e+01           1.000000e+00          0.051605
75%              1.500000e+01           1.000000e+00          0.108909
max              2.300000e+01           1.000000e+00          1.000000

       …  RATE_INTEREST_PRIVILEGED  DAYS_DECISION  SELLERPLACE_AREA  \
count  …               5951.000000   1.670214e+06      1.670214e+06
mean   …                  0.773503  -8.806797e+02      3.139511e+02
std    …                  0.100879   7.790997e+02      7.127443e+03
min    …                  0.373150  -2.922000e+03     -1.000000e+00
25%    …                  0.715645  -1.300000e+03     -1.000000e+00
50%    …                  0.835095  -5.810000e+02      3.000000e+00
75%    …                  0.852537  -2.800000e+02      8.200000e+01
max    …                  1.000000  -1.000000e+00      4.000000e+06

        CNT_PAYMENT  DAYS_FIRST_DRAWING  DAYS_FIRST_DUE  \
count  1.297984e+06       997149.000000   997149.000000
mean   1.605408e+01       342209.855039    13826.269337
std    1.456729e+01        88916.115834    72444.869708
min    0.000000e+00        -2922.000000    -2892.000000
25%    6.000000e+00       365243.000000    -1628.000000
50%    1.200000e+01       365243.000000     -831.000000
75%    2.400000e+01       365243.000000     -411.000000
max    8.400000e+01       365243.000000   365243.000000

       DAYS_LAST_DUE_1ST_VERSION  DAYS_LAST_DUE  DAYS_TERMINATION  \
count              997149.000000  997149.000000     997149.000000
mean                33767.774054   76582.403064      81992.343838
std                106857.034789  149647.415123     153303.516729
min                 -2801.000000   -2889.000000      -2874.000000
25%                 -1242.000000   -1314.000000      -1270.000000
50%                  -361.000000    -537.000000       -499.000000
75%                   129.000000     -74.000000        -44.000000
max                365243.000000  365243.000000     365243.000000

       NFLAG_INSURED_ON_APPROVAL
count              997149.000000
mean                    0.332570
std                     0.471134
min                     0.000000
25%                     0.000000
50%                     0.000000
75%                     1.000000
max                     1.000000
```

[8 rows x 21 columns]

[10]:
```
df_application_data.describe()
```

[10]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	\
count	307511.000000	307511.000000	307511.000000	3.075110e+05	
mean	278180.518577	0.080729	0.417052	1.687979e+05	
std	102790.175348	0.272419	0.722121	2.371231e+05	
min	100002.000000	0.000000	0.000000	2.565000e+04	
25%	189145.500000	0.000000	0.000000	1.125000e+05	
50%	278202.000000	0.000000	0.000000	1.471500e+05	
75%	367142.500000	0.000000	1.000000	2.025000e+05	
max	456255.000000	1.000000	19.000000	1.170000e+08	

	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	\
count	3.075110e+05	307499.000000	3.072330e+05	
mean	5.990260e+05	27108.573909	5.383962e+05	
std	4.024908e+05	14493.737315	3.694465e+05	
min	4.500000e+04	1615.500000	4.050000e+04	
25%	2.700000e+05	16524.000000	2.385000e+05	
50%	5.135310e+05	24903.000000	4.500000e+05	
75%	8.086500e+05	34596.000000	6.795000e+05	
max	4.050000e+06	258025.500000	4.050000e+06	

	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPLOYED	…	\
count	307511.000000	307511.000000	307511.000000	…	
mean	0.020868	-16036.995067	63815.045904	…	
std	0.013831	4363.988632	141275.766519	…	
min	0.000290	-25229.000000	-17912.000000	…	
25%	0.010006	-19682.000000	-2760.000000	…	
50%	0.018850	-15750.000000	-1213.000000	…	
75%	0.028663	-12413.000000	-289.000000	…	
max	0.072508	-7489.000000	365243.000000	…	

	FLAG_DOCUMENT_18	FLAG_DOCUMENT_19	FLAG_DOCUMENT_20	FLAG_DOCUMENT_21	\
count	307511.000000	307511.000000	307511.000000	307511.000000	
mean	0.008130	0.000595	0.000507	0.000335	
std	0.089798	0.024387	0.022518	0.018299	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	

	AMT_REQ_CREDIT_BUREAU_HOUR	AMT_REQ_CREDIT_BUREAU_DAY	\
count	265992.000000	265992.000000	
mean	0.006402	0.007000	

```
std                      0.083849                  0.110757
min                      0.000000                  0.000000
25%                      0.000000                  0.000000
50%                      0.000000                  0.000000
75%                      0.000000                  0.000000
max                      4.000000                  9.000000

       AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
count                265992.000000              265992.000000
mean                      0.034362                   0.267395
std                       0.204685                   0.916002
min                       0.000000                   0.000000
25%                       0.000000                   0.000000
50%                       0.000000                   0.000000
75%                       0.000000                   0.000000
max                       8.000000                  27.000000

       AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
count                265992.000000              265992.000000
mean                      0.265474                   1.899974
std                       0.794056                   1.869295
min                       0.000000                   0.000000
25%                       0.000000                   0.000000
50%                       0.000000                   1.000000
75%                       0.000000                   3.000000
max                     261.000000                  25.000000

[8 rows x 106 columns]
```

[21]:
```
df_previous_application.columns
```

[21]:
```
Index(['SK_ID_PREV', 'SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_ANNUITY',
       'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE',
       'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NFLAG_LAST_APPL_IN_DAY',
       'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY',
       'RATE_INTEREST_PRIVILEGED', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'DAYS_DECISION', 'NAME_PAYMENT_TYPE',
       'CODE_REJECT_REASON', 'NAME_TYPE_SUITE', 'NAME_CLIENT_TYPE',
       'NAME_GOODS_CATEGORY', 'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE',
       'CHANNEL_TYPE', 'SELLERPLACE_AREA', 'NAME_SELLER_INDUSTRY',
       'CNT_PAYMENT', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION',
       'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION',
       'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL'],
      dtype='object')
```

[12]:
```
len(df_previous_application.columns)
```

[12]: 37

[13]:
```
# Convert column names to a list and print
column_names_list = df_application_data.columns.tolist()
print(column_names_list)
print(len(column_names_list))
```

```
['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR',
'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT',
'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE',
'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE',
'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OWN_CAR_AGE', 'FLAG_MOBIL',
'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE',
'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
'REGION_RATING_CLIENT_W_CITY', 'WEEKDAY_APPR_PROCESS_START',
'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION',
'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION',
'REG_CITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY',
'ORGANIZATION_TYPE', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG',
'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG',
'FLOORSMAX_AVG', 'FLOORSMIN_AVG', 'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG',
'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG',
'APARTMENTS_MODE', 'BASEMENTAREA_MODE', 'YEARS_BEGINEXPLUATATION_MODE',
'YEARS_BUILD_MODE', 'COMMONAREA_MODE', 'ELEVATORS_MODE', 'ENTRANCES_MODE',
'FLOORSMAX_MODE', 'FLOORSMIN_MODE', 'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE',
'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE',
'APARTMENTS_MEDI', 'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI',
'YEARS_BUILD_MEDI', 'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI',
'FLOORSMAX_MEDI', 'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI',
'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI',
'FONDKAPREMONT_MODE', 'HOUSETYPE_MODE', 'TOTALAREA_MODE', 'WALLSMATERIAL_MODE',
'EMERGENCYSTATE_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7',
'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11',
'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15',
'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19',
'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21', 'AMT_REQ_CREDIT_BUREAU_HOUR',
'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
'AMT_REQ_CREDIT_BUREAU_YEAR']
122
```

1 check for null values and deal it

[14]:
```
df_previous_application.isnull().sum()
```

[14]:
```
SK_ID_PREV                        0
SK_ID_CURR                        0
NAME_CONTRACT_TYPE                0
AMT_ANNUITY                  372235
AMT_APPLICATION                   0
AMT_CREDIT                        1
AMT_DOWN_PAYMENT             895844
AMT_GOODS_PRICE              385515
WEEKDAY_APPR_PROCESS_START        0
HOUR_APPR_PROCESS_START           0
FLAG_LAST_APPL_PER_CONTRACT       0
NFLAG_LAST_APPL_IN_DAY            0
RATE_DOWN_PAYMENT            895844
RATE_INTEREST_PRIMARY       1664263
RATE_INTEREST_PRIVILEGED    1664263
NAME_CASH_LOAN_PURPOSE            0
NAME_CONTRACT_STATUS              0
DAYS_DECISION                     0
NAME_PAYMENT_TYPE                 0
CODE_REJECT_REASON                0
NAME_TYPE_SUITE              820405
NAME_CLIENT_TYPE                  0
NAME_GOODS_CATEGORY               0
NAME_PORTFOLIO                    0
NAME_PRODUCT_TYPE                 0
CHANNEL_TYPE                      0
SELLERPLACE_AREA                  0
NAME_SELLER_INDUSTRY              0
CNT_PAYMENT                  372230
NAME_YIELD_GROUP                  0
PRODUCT_COMBINATION             346
DAYS_FIRST_DRAWING           673065
DAYS_FIRST_DUE               673065
DAYS_LAST_DUE_1ST_VERSION    673065
DAYS_LAST_DUE                673065
DAYS_TERMINATION             673065
NFLAG_INSURED_ON_APPROVAL    673065
dtype: int64
```

[15]:
```
# Print the number of null values for each column
for column, nullval in df_application_data.isnull().sum().items():
    print(f'{column}: {nullval}')
```

```
SK_ID_CURR: 0
```

TARGET: 0
NAME_CONTRACT_TYPE: 0
CODE_GENDER: 0
FLAG_OWN_CAR: 0
FLAG_OWN_REALTY: 0
CNT_CHILDREN: 0
AMT_INCOME_TOTAL: 0
AMT_CREDIT: 0
AMT_ANNUITY: 12
AMT_GOODS_PRICE: 278
NAME_TYPE_SUITE: 1292
NAME_INCOME_TYPE: 0
NAME_EDUCATION_TYPE: 0
NAME_FAMILY_STATUS: 0
NAME_HOUSING_TYPE: 0
REGION_POPULATION_RELATIVE: 0
DAYS_BIRTH: 0
DAYS_EMPLOYED: 0
DAYS_REGISTRATION: 0
DAYS_ID_PUBLISH: 0
OWN_CAR_AGE: 202929
FLAG_MOBIL: 0
FLAG_EMP_PHONE: 0
FLAG_WORK_PHONE: 0
FLAG_CONT_MOBILE: 0
FLAG_PHONE: 0
FLAG_EMAIL: 0
OCCUPATION_TYPE: 96391
CNT_FAM_MEMBERS: 2
REGION_RATING_CLIENT: 0
REGION_RATING_CLIENT_W_CITY: 0
WEEKDAY_APPR_PROCESS_START: 0
HOUR_APPR_PROCESS_START: 0
REG_REGION_NOT_LIVE_REGION: 0
REG_REGION_NOT_WORK_REGION: 0
LIVE_REGION_NOT_WORK_REGION: 0
REG_CITY_NOT_LIVE_CITY: 0
REG_CITY_NOT_WORK_CITY: 0
LIVE_CITY_NOT_WORK_CITY: 0
ORGANIZATION_TYPE: 0
EXT_SOURCE_1: 173378
EXT_SOURCE_2: 660
EXT_SOURCE_3: 60965
APARTMENTS_AVG: 156061
BASEMENTAREA_AVG: 179943
YEARS_BEGINEXPLUATATION_AVG: 150007
YEARS_BUILD_AVG: 204488
COMMONAREA_AVG: 214865

ELEVATORS_AVG: 163891
ENTRANCES_AVG: 154828
FLOORSMAX_AVG: 153020
FLOORSMIN_AVG: 208642
LANDAREA_AVG: 182590
LIVINGAPARTMENTS_AVG: 210199
LIVINGAREA_AVG: 154350
NONLIVINGAPARTMENTS_AVG: 213514
NONLIVINGAREA_AVG: 169682
APARTMENTS_MODE: 156061
BASEMENTAREA_MODE: 179943
YEARS_BEGINEXPLUATATION_MODE: 150007
YEARS_BUILD_MODE: 204488
COMMONAREA_MODE: 214865
ELEVATORS_MODE: 163891
ENTRANCES_MODE: 154828
FLOORSMAX_MODE: 153020
FLOORSMIN_MODE: 208642
LANDAREA_MODE: 182590
LIVINGAPARTMENTS_MODE: 210199
LIVINGAREA_MODE: 154350
NONLIVINGAPARTMENTS_MODE: 213514
NONLIVINGAREA_MODE: 169682
APARTMENTS_MEDI: 156061
BASEMENTAREA_MEDI: 179943
YEARS_BEGINEXPLUATATION_MEDI: 150007
YEARS_BUILD_MEDI: 204488
COMMONAREA_MEDI: 214865
ELEVATORS_MEDI: 163891
ENTRANCES_MEDI: 154828
FLOORSMAX_MEDI: 153020
FLOORSMIN_MEDI: 208642
LANDAREA_MEDI: 182590
LIVINGAPARTMENTS_MEDI: 210199
LIVINGAREA_MEDI: 154350
NONLIVINGAPARTMENTS_MEDI: 213514
NONLIVINGAREA_MEDI: 169682
FONDKAPREMONT_MODE: 210295
HOUSETYPE_MODE: 154297
TOTALAREA_MODE: 148431
WALLSMATERIAL_MODE: 156341
EMERGENCYSTATE_MODE: 145755
OBS_30_CNT_SOCIAL_CIRCLE: 1021
DEF_30_CNT_SOCIAL_CIRCLE: 1021
OBS_60_CNT_SOCIAL_CIRCLE: 1021
DEF_60_CNT_SOCIAL_CIRCLE: 1021
DAYS_LAST_PHONE_CHANGE: 1
FLAG_DOCUMENT_2: 0

```
FLAG_DOCUMENT_3: 0
FLAG_DOCUMENT_4: 0
FLAG_DOCUMENT_5: 0
FLAG_DOCUMENT_6: 0
FLAG_DOCUMENT_7: 0
FLAG_DOCUMENT_8: 0
FLAG_DOCUMENT_9: 0
FLAG_DOCUMENT_10: 0
FLAG_DOCUMENT_11: 0
FLAG_DOCUMENT_12: 0
FLAG_DOCUMENT_13: 0
FLAG_DOCUMENT_14: 0
FLAG_DOCUMENT_15: 0
FLAG_DOCUMENT_16: 0
FLAG_DOCUMENT_17: 0
FLAG_DOCUMENT_18: 0
FLAG_DOCUMENT_19: 0
FLAG_DOCUMENT_20: 0
FLAG_DOCUMENT_21: 0
AMT_REQ_CREDIT_BUREAU_HOUR: 41519
AMT_REQ_CREDIT_BUREAU_DAY: 41519
AMT_REQ_CREDIT_BUREAU_WEEK: 41519
AMT_REQ_CREDIT_BUREAU_MON: 41519
AMT_REQ_CREDIT_BUREAU_QRT: 41519
AMT_REQ_CREDIT_BUREAU_YEAR: 41519
```

2 Check for the rows where there all NULL values and drop them

3 PREVIOUS_APPLICATION

[39]:
```
columns_to_check = [
    'AMT_ANNUITY', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRICE', 'RATE_DOWN_PAYMENT',
    'RATE_INTEREST_PRIMARY', 'RATE_INTEREST_PRIVILEGED', 'NAME_TYPE_SUITE',
    'CNT_PAYMENT', 'PRODUCT_COMBINATION', 'DAYS_FIRST_DRAWING',␣
 ↪'DAYS_FIRST_DUE',
    'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE', 'DAYS_TERMINATION',␣
 ↪'NFLAG_INSURED_ON_APPROVAL'
]

 # check the rows with all values in the specified columns are null
all_null = df_previous_application[columns_to_check].isnull().all(axis=1)

# Filter the DataFrame based on the all_null mask
filtered_df = df_previous_application[all_null]

filtered_df.head(5)
```

[39]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	\
6664	2515161	222844	XNA	NaN	
9029	1851920	417884	XNA	NaN	
17038	2389511	148922	XNA	NaN	
24543	2494449	366626	XNA	NaN	
24574	2781877	394843	XNA	NaN	

	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	\
6664	0.0	0.0	NaN	NaN	
9029	0.0	0.0	NaN	NaN	
17038	0.0	0.0	NaN	NaN	
24543	0.0	0.0	NaN	NaN	
24574	0.0	0.0	NaN	NaN	

	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START	…	\
6664	SATURDAY	8	…	
9029	MONDAY	13	…	
17038	TUESDAY	6	…	
24543	WEDNESDAY	13	…	
24574	SATURDAY	6	…	

	NAME_SELLER_INDUSTRY	CNT_PAYMENT	NAME_YIELD_GROUP	\
6664	Connectivity	NaN	XNA	
9029	Connectivity	NaN	XNA	
17038	Connectivity	NaN	XNA	
24543	Connectivity	NaN	XNA	
24574	Connectivity	NaN	XNA	

	PRODUCT_COMBINATION	DAYS_FIRST_DRAWING	DAYS_FIRST_DUE	\
6664	NaN	NaN	NaN	
9029	NaN	NaN	NaN	
17038	NaN	NaN	NaN	
24543	NaN	NaN	NaN	
24574	NaN	NaN	NaN	

	DAYS_LAST_DUE_1ST_VERSION	DAYS_LAST_DUE	DAYS_TERMINATION	\
6664	NaN	NaN	NaN	
9029	NaN	NaN	NaN	
17038	NaN	NaN	NaN	
24543	NaN	NaN	NaN	
24574	NaN	NaN	NaN	

	NFLAG_INSURED_ON_APPROVAL
6664	NaN
9029	NaN
17038	NaN
24543	NaN

```
24574                          NaN

[5 rows x 37 columns]
```

[40]:
```
df_previous_application.drop(filtered_df.index, inplace=True)
```

[36]:
```
# # DROP NULL VALUES FROM FOLLOWING COLUMNS

# df_previous_application.dropna(subset=['AMT_CREDIT', 'AMT_ANNUITY',␣
 ↪'DAYS_FIRST_DRAWING', 'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION',
#                                         'DAYS_LAST_DUE', 'DAYS_TERMINATION',␣
 ↪'NFLAG_INSURED_ON_APPROVAL'], inplace=True)
```

[41]:
```
# CONVERT CNT_PAYMENT '0' VALUES TO NULL AND DROP THEM
import numpy as np
df_previous_application['CNT_PAYMENT'] = df_previous_application['CNT_PAYMENT'].
 ↪replace('0', np.nan)

df_previous_application.dropna(subset=['CNT_PAYMENT'], inplace=True)
```

4 APPLICATION_DATA

[42]:
```
df_application_data.dropna(subset=['AMT_ANNUITY', 'CNT_FAM_MEMBERS',␣
 ↪'DAYS_LAST_PHONE_CHANGE'], inplace = True)
```

[ ]:
```

```

5 FILL NULL VALUES WITH SUITABLE METHODS

6 PREVIOUS_APPLICATION

[46]:
```
# FILL WITH '0'
df_previous_application['AMT_DOWN_PAYMENT'] =␣
 ↪df_previous_application['AMT_DOWN_PAYMENT'].fillna(0)


# Fill only NaN values in the 'AMT_GOODS_PRICE' column with the corresponding␣
 ↪values from the 'AMT_CREDIT' column
df_previous_application.loc[df_previous_application['AMT_GOODS_PRICE'].
 ↪isnull(), 'AMT_GOODS_PRICE'] = df_previous_application['AMT_CREDIT']

# # FILL NULL VALUES WITH SIMPLE IMPUTER
# from sklearn.impute import SimpleImputer
```

```
# df_previous_application['RATE_INTEREST_PRIVILEGED'] =␣
 ↪SimpleImputer(strategy='mean').
 ↪fit_transform(df_previous_application['RATE_INTEREST_PRIVILEGED'].values.
 ↪reshape(-1, 1))
# df_previous_application['DAYS_FIRST_DRAWING'] =␣
 ↪SimpleImputer(strategy='mean').
 ↪fit_transform(df_previous_application['DAYS_FIRST_DRAWING'].values.
 ↪reshape(-1, 1))


# FILL NULL VALUES WITH A SPECIFIC CHOICE

df_previous_application['NAME_TYPE_SUITE'].fillna('Unaccompanied', inplace=True)
```

[47]:
```
from scipy.optimize import fsolve
def calculate_monthly_rate(annuity, credit_amount, n_payments):
    # Define the equation for the annuity
    def annuity_equation(r):
        return credit_amount * r / (1 - (1 + r) ** -n_payments) - annuity

    # Use fsolve to solve for r, starting with an initial guess of 0.01 (1%)
    monthly_rate = fsolve(annuity_equation, 0.01)[0]
    return monthly_rate
# FILL NULL WITH CALCULATED VALUES
null_rate_rows =␣
 ↪df_previous_application[df_previous_application['RATE_INTEREST_PRIMARY'].
 ↪isnull()]

for idx, row in null_rate_rows.iterrows():
    if pd.notnull(row['AMT_ANNUITY']) and pd.notnull(row['AMT_CREDIT']) and pd.
 ↪notnull(row['CNT_PAYMENT']):
        n_payments = row['CNT_PAYMENT']
        annuity = row['AMT_ANNUITY']
        credit_amount = row['AMT_CREDIT']

        if n_payments > 0:
            monthly_rate = calculate_monthly_rate(annuity, credit_amount,␣
 ↪n_payments)
            annual_rate = ((1 + monthly_rate) ** 12) - 1
            df_previous_application.at[idx, 'RATE_INTEREST_PRIMARY'] =␣
 ↪annual_rate

# Display the DataFrame to verify the updated RATE_INTEREST_PRIMARY column
print(df_previous_application[['SK_ID_PREV', 'RATE_INTEREST_PRIMARY']].head())
```

```
   SK_ID_PREV  RATE_INTEREST_PRIMARY
0     2030495               0.182832
```

1	2802425	0.216953
2	2523466	0.718134
3	2819243	0.410797
4	1784265	0.991558

[48]:
```
# FILL NULL VALUES BY CALCULATIONS
# \text{RATE_DOWN_PAYMENT} = \frac{\text{AMT_DOWN_PAYMENT}}{\text{AMT_CREDIT}}␣
 ↪\times 100
df_previous_application['RATE_DOWN_PAYMENT'] =␣
 ↪(df_previous_application['AMT_DOWN_PAYMENT'] /␣
 ↪df_previous_application['AMT_CREDIT']) * 100
```

[49]:
```
# FILL NA WITH MEAN/median
df_previous_application = df_previous_application.
 ↪fillna(df_previous_application.median(numeric_only=True))

for column in df_previous_application.select_dtypes(include=['object']).columns:
    mode_value = df_previous_application[column].mode()[0]
    df_previous_application[column].fillna(mode_value, inplace=True)
```

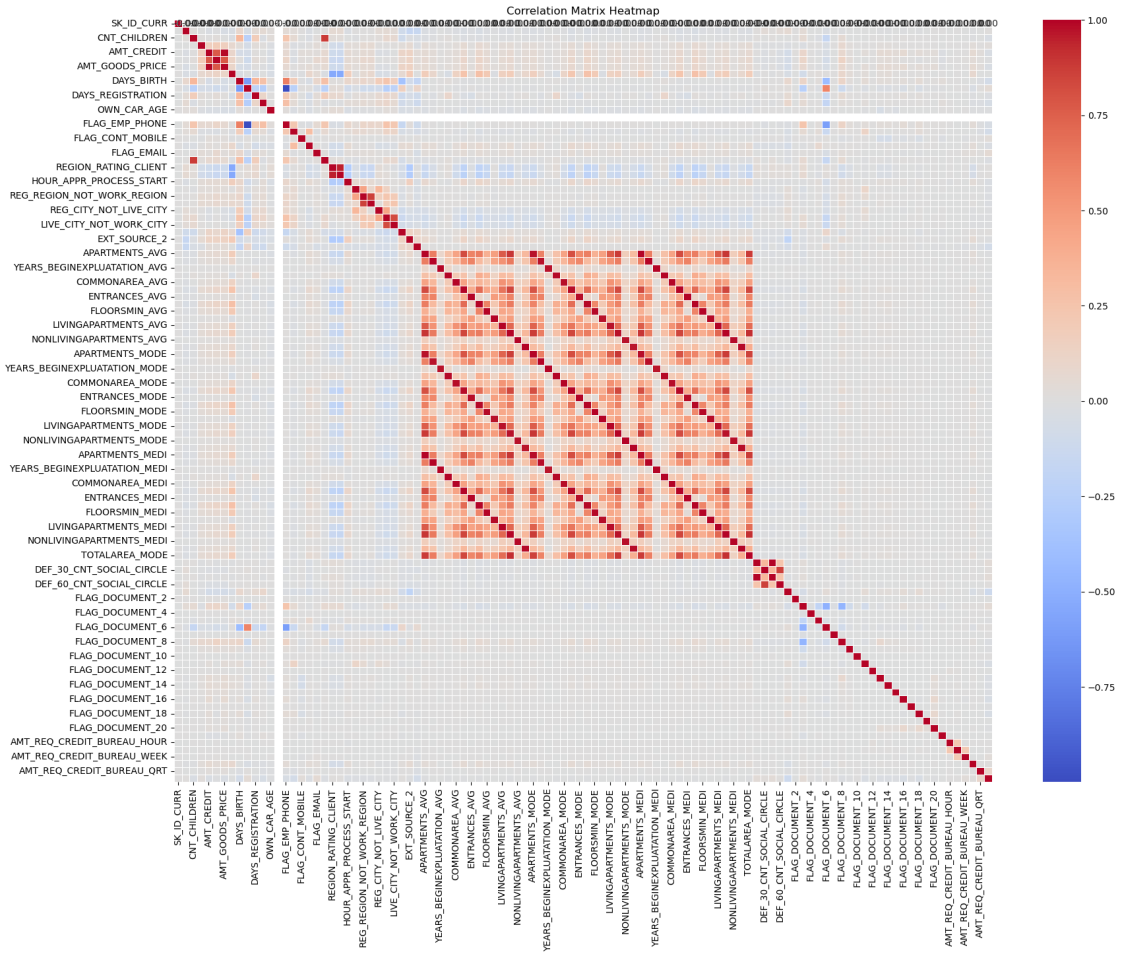
7 APPLICATION_DATA

[50]:
```
# FILL NULL VALUES WITH APPROPRIATE METHODS:
# Fill null values in 'AMT_GOODS_PRICE' with values from 'AMT_ANNUITY'
df_application_data['AMT_GOODS_PRICE'].
 ↪fillna(df_application_data['AMT_CREDIT'], inplace=True)


# FILL NULL VALUES WITH A SPECIFIC CHOICE
df_application_data['OCCUPATION_TYPE'].fillna('Not Mentioned', inplace=True)
df_application_data['NAME_TYPE_SUITE'].fillna('Unaccompanied', inplace=True)


# FILL NA WITH MEAN/median
df_application_data = df_application_data.fillna(df_application_data.
 ↪median(numeric_only=True))

for column in df_application_data.select_dtypes(include=['object']).columns:
    mode_value = df_application_data[column].mode()[0]
    df_application_data[column].fillna(mode_value, inplace=True)
```

8 EXPLORATORY DATA ANALYSIS

[56]:
```
# select numeric data
df_application_data_num = df_application_data.select_dtypes(include=['number'])

# plot correlation matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix
corr_matrix = df_application_data_num.corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(20, 15))  # Adjust the size as needed
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
C:\Users\HP\Anaconda\Lib\site-packages\seaborn\matrix.py:260: FutureWarning:
Format strings passed to MaskedConstant are ignored, but in future may error or
produce different behavior
  annotation = ("{:" + self.fmt + "}").format(val)
```



[58]:
```
# Calculate the correlation matrix
corr_matrix = df_application_data_num.corr()

# Define the threshold for "closely related"
threshold = 0.8

closely_related_cols = []

for col in corr_matrix.columns:
    related_cols = corr_matrix.index[corr_matrix[col] > threshold].tolist()
    # Ensure the column itself is in the list before attempting to remove it
    if col in related_cols:
        related_cols.remove(col)  # Remove the column itself
    if related_cols:
        closely_related_cols.append((col, related_cols))

# Print closely related columns
```

```
for col, related_cols in closely_related_cols:
    print(f"Column '{col}' is closely related to columns: {related_cols}")
```

```
Column 'CNT_CHILDREN' is closely related to columns: ['CNT_FAM_MEMBERS']
Column 'AMT_CREDIT' is closely related to columns: ['AMT_GOODS_PRICE']
Column 'AMT_GOODS_PRICE' is closely related to columns: ['AMT_CREDIT']
Column 'CNT_FAM_MEMBERS' is closely related to columns: ['CNT_CHILDREN']
Column 'REGION_RATING_CLIENT' is closely related to columns:
['REGION_RATING_CLIENT_W_CITY']
Column 'REGION_RATING_CLIENT_W_CITY' is closely related to columns:
['REGION_RATING_CLIENT']
Column 'REG_REGION_NOT_WORK_REGION' is closely related to columns:
['LIVE_REGION_NOT_WORK_REGION']
Column 'LIVE_REGION_NOT_WORK_REGION' is closely related to columns:
['REG_REGION_NOT_WORK_REGION']
Column 'REG_CITY_NOT_WORK_CITY' is closely related to columns:
['LIVE_CITY_NOT_WORK_CITY']
Column 'LIVE_CITY_NOT_WORK_CITY' is closely related to columns:
['REG_CITY_NOT_WORK_CITY']
Column 'APARTMENTS_AVG' is closely related to columns: ['ELEVATORS_AVG',
'LIVINGAREA_AVG', 'APARTMENTS_MODE', 'ELEVATORS_MODE', 'LIVINGAREA_MODE',
'APARTMENTS_MEDI', 'ELEVATORS_MEDI', 'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'BASEMENTAREA_AVG' is closely related to columns: ['BASEMENTAREA_MODE',
'BASEMENTAREA_MEDI']
Column 'YEARS_BEGINEXPLUATATION_AVG' is closely related to columns:
['YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BEGINEXPLUATATION_MEDI']
Column 'YEARS_BUILD_AVG' is closely related to columns: ['YEARS_BUILD_MODE',
'YEARS_BUILD_MEDI']
Column 'COMMONAREA_AVG' is closely related to columns: ['COMMONAREA_MODE',
'COMMONAREA_MEDI']
Column 'ELEVATORS_AVG' is closely related to columns: ['APARTMENTS_AVG',
'LIVINGAREA_AVG', 'ELEVATORS_MODE', 'LIVINGAREA_MODE', 'APARTMENTS_MEDI',
'ELEVATORS_MEDI', 'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'ENTRANCES_AVG' is closely related to columns: ['ENTRANCES_MODE',
'ENTRANCES_MEDI']
Column 'FLOORSMAX_AVG' is closely related to columns: ['FLOORSMAX_MODE',
'FLOORSMAX_MEDI']
Column 'FLOORSMIN_AVG' is closely related to columns: ['FLOORSMIN_MODE',
'FLOORSMIN_MEDI']
Column 'LANDAREA_AVG' is closely related to columns: ['LANDAREA_MODE',
'LANDAREA_MEDI']
Column 'LIVINGAPARTMENTS_AVG' is closely related to columns:
['LIVINGAPARTMENTS_MODE', 'LIVINGAPARTMENTS_MEDI']
Column 'LIVINGAREA_AVG' is closely related to columns: ['APARTMENTS_AVG',
'ELEVATORS_AVG', 'APARTMENTS_MODE', 'ELEVATORS_MODE', 'LIVINGAREA_MODE',
'APARTMENTS_MEDI', 'ELEVATORS_MEDI', 'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'NONLIVINGAPARTMENTS_AVG' is closely related to columns:
['NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAPARTMENTS_MEDI']
```

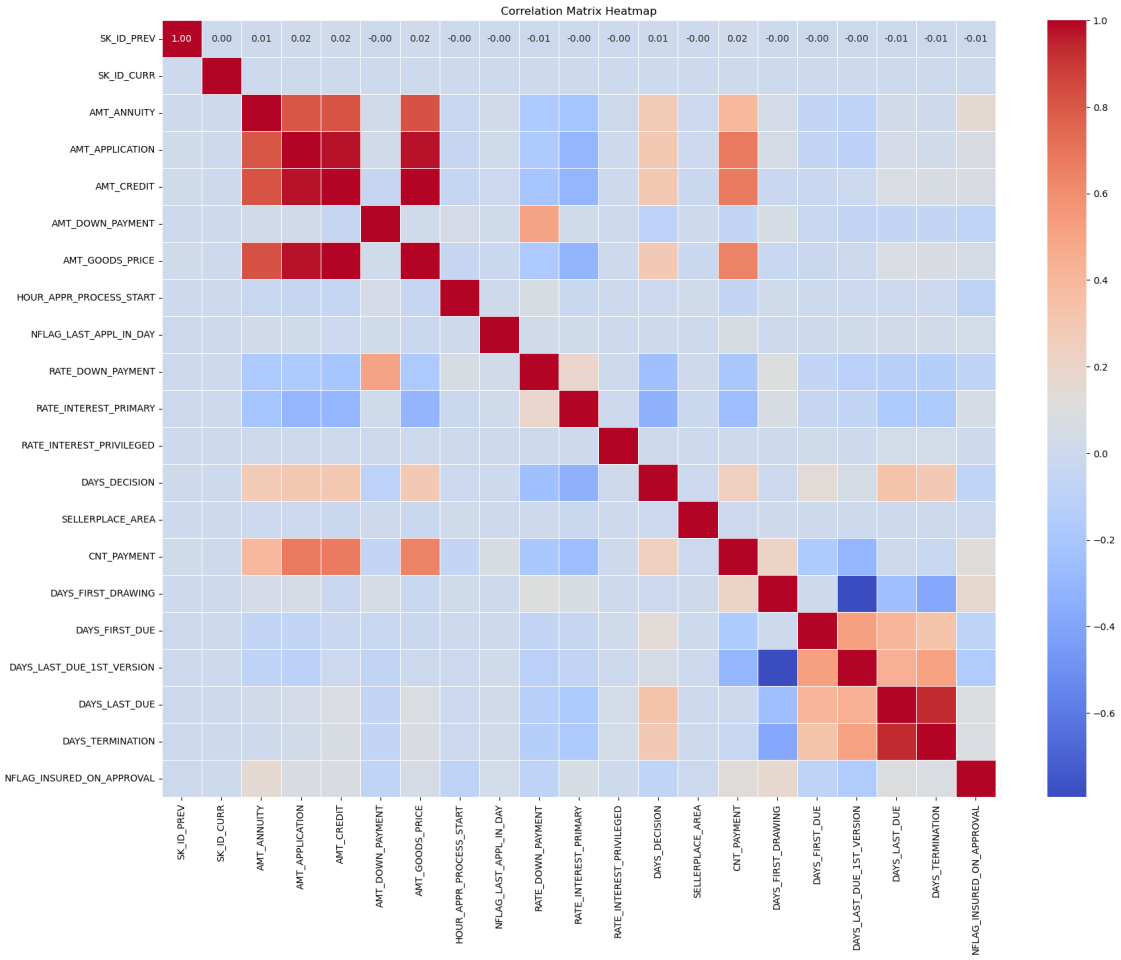
```
Column 'NONLIVINGAREA_AVG' is closely related to columns: ['NONLIVINGAREA_MODE',
'NONLIVINGAREA_MEDI']
Column 'APARTMENTS_MODE' is closely related to columns: ['APARTMENTS_AVG',
'LIVINGAREA_AVG', 'ELEVATORS_MODE', 'LIVINGAREA_MODE', 'APARTMENTS_MEDI',
'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'BASEMENTAREA_MODE' is closely related to columns: ['BASEMENTAREA_AVG',
'BASEMENTAREA_MEDI']
Column 'YEARS_BEGINEXPLUATATION_MODE' is closely related to columns:
['YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BEGINEXPLUATATION_MEDI']
Column 'YEARS_BUILD_MODE' is closely related to columns: ['YEARS_BUILD_AVG',
'YEARS_BUILD_MEDI']
Column 'COMMONAREA_MODE' is closely related to columns: ['COMMONAREA_AVG',
'COMMONAREA_MEDI']
Column 'ELEVATORS_MODE' is closely related to columns: ['APARTMENTS_AVG',
'ELEVATORS_AVG', 'LIVINGAREA_AVG', 'APARTMENTS_MODE', 'LIVINGAREA_MODE',
'APARTMENTS_MEDI', 'ELEVATORS_MEDI', 'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'ENTRANCES_MODE' is closely related to columns: ['ENTRANCES_AVG',
'ENTRANCES_MEDI']
Column 'FLOORSMAX_MODE' is closely related to columns: ['FLOORSMAX_AVG',
'FLOORSMAX_MEDI']
Column 'FLOORSMIN_MODE' is closely related to columns: ['FLOORSMIN_AVG',
'FLOORSMIN_MEDI']
Column 'LANDAREA_MODE' is closely related to columns: ['LANDAREA_AVG',
'LANDAREA_MEDI']
Column 'LIVINGAPARTMENTS_MODE' is closely related to columns:
['LIVINGAPARTMENTS_AVG', 'LIVINGAPARTMENTS_MEDI']
Column 'LIVINGAREA_MODE' is closely related to columns: ['APARTMENTS_AVG',
'ELEVATORS_AVG', 'LIVINGAREA_AVG', 'APARTMENTS_MODE', 'ELEVATORS_MODE',
'APARTMENTS_MEDI', 'ELEVATORS_MEDI', 'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'NONLIVINGAPARTMENTS_MODE' is closely related to columns:
['NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAPARTMENTS_MEDI']
Column 'NONLIVINGAREA_MODE' is closely related to columns: ['NONLIVINGAREA_AVG',
'NONLIVINGAREA_MEDI']
Column 'APARTMENTS_MEDI' is closely related to columns: ['APARTMENTS_AVG',
'ELEVATORS_AVG', 'LIVINGAREA_AVG', 'APARTMENTS_MODE', 'ELEVATORS_MODE',
'LIVINGAREA_MODE', 'ELEVATORS_MEDI', 'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'BASEMENTAREA_MEDI' is closely related to columns: ['BASEMENTAREA_AVG',
'BASEMENTAREA_MODE']
Column 'YEARS_BEGINEXPLUATATION_MEDI' is closely related to columns:
['YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BEGINEXPLUATATION_MODE']
Column 'YEARS_BUILD_MEDI' is closely related to columns: ['YEARS_BUILD_AVG',
'YEARS_BUILD_MODE']
Column 'COMMONAREA_MEDI' is closely related to columns: ['COMMONAREA_AVG',
'COMMONAREA_MODE']
Column 'ELEVATORS_MEDI' is closely related to columns: ['APARTMENTS_AVG',
'ELEVATORS_AVG', 'LIVINGAREA_AVG', 'ELEVATORS_MODE', 'LIVINGAREA_MODE',
'APARTMENTS_MEDI', 'LIVINGAREA_MEDI', 'TOTALAREA_MODE']
Column 'ENTRANCES_MEDI' is closely related to columns: ['ENTRANCES_AVG',
```

```
'ENTRANCES_MODE']
Column 'FLOORSMAX_MEDI' is closely related to columns: ['FLOORSMAX_AVG',
'FLOORSMAX_MODE']
Column 'FLOORSMIN_MEDI' is closely related to columns: ['FLOORSMIN_AVG',
'FLOORSMIN_MODE']
Column 'LANDAREA_MEDI' is closely related to columns: ['LANDAREA_AVG',
'LANDAREA_MODE']
Column 'LIVINGAPARTMENTS_MEDI' is closely related to columns:
['LIVINGAPARTMENTS_AVG', 'LIVINGAPARTMENTS_MODE']
Column 'LIVINGAREA_MEDI' is closely related to columns: ['APARTMENTS_AVG',
'ELEVATORS_AVG', 'LIVINGAREA_AVG', 'APARTMENTS_MODE', 'ELEVATORS_MODE',
'LIVINGAREA_MODE', 'APARTMENTS_MEDI', 'ELEVATORS_MEDI', 'TOTALAREA_MODE']
Column 'NONLIVINGAPARTMENTS_MEDI' is closely related to columns:
['NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAPARTMENTS_MODE']
Column 'NONLIVINGAREA_MEDI' is closely related to columns: ['NONLIVINGAREA_AVG',
'NONLIVINGAREA_MODE']
Column 'TOTALAREA_MODE' is closely related to columns: ['APARTMENTS_AVG',
'ELEVATORS_AVG', 'LIVINGAREA_AVG', 'APARTMENTS_MODE', 'ELEVATORS_MODE',
'LIVINGAREA_MODE', 'APARTMENTS_MEDI', 'ELEVATORS_MEDI', 'LIVINGAREA_MEDI']
Column 'OBS_30_CNT_SOCIAL_CIRCLE' is closely related to columns:
['OBS_60_CNT_SOCIAL_CIRCLE']
Column 'DEF_30_CNT_SOCIAL_CIRCLE' is closely related to columns:
['DEF_60_CNT_SOCIAL_CIRCLE']
Column 'OBS_60_CNT_SOCIAL_CIRCLE' is closely related to columns:
['OBS_30_CNT_SOCIAL_CIRCLE']
Column 'DEF_60_CNT_SOCIAL_CIRCLE' is closely related to columns:
['DEF_30_CNT_SOCIAL_CIRCLE']
```

[ ]:
```
import seaborn as sns
import warnings

# Create a pair plot of 'AMT_ANNUITY' with 'AMT_APPLICATION', 'AMT_CREDIT', and␣
 ↪'AMT_GOODS_PRICE'
sns.pairplot(df_previous_application_num[['AMT_ANNUITY', 'AMT_APPLICATION',␣
 ↪'AMT_CREDIT', 'AMT_GOODS_PRICE']])
```

[55]:
```
# select numeric data
df_previous_application_num = df_previous_application.
 ↪select_dtypes(include=['number'])

# plot correlation matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix
corr_matrix = df_previous_application_num.corr()
```

```
# Create a heatmap of the correlation matrix
plt.figure(figsize=(20, 15))  # Adjust the size as needed
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



[59]:
```
# Calculate the correlation matrix
corr_matrix = df_previous_application_num.corr()

# Define the threshold for "closely related"
threshold = 0.8

closely_related_cols = []

for col in corr_matrix.columns:
    related_cols = corr_matrix.index[corr_matrix[col] > threshold].tolist()
    # Ensure the column itself is in the list before attempting to remove it
```

```
    if col in related_cols:
        related_cols.remove(col)  # Remove the column itself
    if related_cols:
        closely_related_cols.append((col, related_cols))

# Print closely related columns
for col, related_cols in closely_related_cols:
    print(f"Column '{col}' is closely related to columns: {related_cols}")
```

```
Column 'AMT_ANNUITY' is closely related to columns: ['AMT_APPLICATION',
'AMT_CREDIT', 'AMT_GOODS_PRICE']
Column 'AMT_APPLICATION' is closely related to columns: ['AMT_ANNUITY',
'AMT_CREDIT', 'AMT_GOODS_PRICE']
Column 'AMT_CREDIT' is closely related to columns: ['AMT_ANNUITY',
'AMT_APPLICATION', 'AMT_GOODS_PRICE']
Column 'AMT_GOODS_PRICE' is closely related to columns: ['AMT_ANNUITY',
'AMT_APPLICATION', 'AMT_CREDIT']
Column 'DAYS_LAST_DUE' is closely related to columns: ['DAYS_TERMINATION']
Column 'DAYS_TERMINATION' is closely related to columns: ['DAYS_LAST_DUE']
```

9 LOAD CLEANED DATA TO MYSQL

[1]:
```
import pymysql
from sqlalchemy import create_engine

# Connection parameters
connection_params = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Nahid123',
}

# Establish initial connection to create the database
connection = pymysql.connect(**connection_params)
cursor = connection.cursor()

# Database name
database_name = 'Financial_Analysis'

# Create the database if it doesn't exist
cursor.execute(f"CREATE DATABASE IF NOT EXISTS {database_name}")

connection.close()

# Reconnect to the newly created database
connection = pymysql.connect(host='localhost', user='root',␣
 ↪password='Nahid123', database=database_name)
```

```
cursor = connection.cursor()

# Create SQLAlchemy engine
engine = create_engine(f'mysql+pymysql://root:Nahid123@localhost/
 ↪{database_name}', echo=False)
```

[63]:
```
# Load the DataFrame into the MySQL table
table_name = 'previous_application'
df_previous_application.to_sql(name=table_name, con=engine, if_exists='append',␣
 ↪index=False)
```

[63]: 1297984

[64]:
```
# Load the DataFrame into the MySQL table
table_name = 'application_data'
df_application_data.to_sql(name=table_name, con=engine, if_exists='append',␣
 ↪index=False)
```

[64]: 307496

10 CONVERT THE CLEANED DATA TO CSV

[68]:
```
df_application_data.to_csv('cleaned_application_data.csv', index=False)
```

[69]:
```
df_previous_application.to_csv('cleaned_previous_application.csv', index=False)
```

[ ]:
```

```