In [2]:
```
import pandas as pd
daf = pd.read_csv('application_data.csv')
daf
```

Out[2]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OV
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	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

307511 rows × 122 columns

In [3]:
```
daf1 = pd.read_csv('previous_application.csv')
daf1
```

Out[3]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	A
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	
...	...	...	...	...	...	
1670209	2300464	352015	Consumer loans	14704.290	267295.5	
1670210	2357031	334635	Consumer loans	6622.020	87750.0	
1670211	2659632	249544	Consumer loans	11520.855	105237.0	
1670212	2785582	400317	Cash loans	18821.520	180000.0	
1670213	2418762	261212	Cash loans	16431.300	360000.0	

1670214 rows × 37 columns

In [4]:
```
#cleaning the data
# Check for missing columns
missing_columns = daf.columns[daf.isnull().any()].tolist()
```

```
if missing_columns:
    print(f"Missing columns: {missing_columns}")
else:
    print("No missing columns.")
```

```
Missing columns: ['AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_TYPE_SUITE', 'OWN_CAR_AG
E', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOU
RCE_3', 'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG', 'YEAR
S_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG',
'FLOORSMIN_AVG', 'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIV
INGAPARTMENTS_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', 'LIV
INGAPARTMENTS_MODE', 'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA
_MODE', 'APARTMENTS_MEDI', 'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI', 'Y
EARS_BUILD_MEDI', 'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMA
X_MEDI', 'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_M
EDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI', 'FONDKAPREMONT_MODE', 'HOU
SETYPE_MODE', 'TOTALAREA_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE', 'OBS_
30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DE
F_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', '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']
```

In [7]:
```
# Finding the total number of values which are missing in each column - .isnull().s
daf['AMT_ANNUITY'].isnull().sum()
daf['AMT_GOODS_PRICE'].isnull().sum()
daf['NAME_TYPE_SUITE'].isnull().sum()
daf['OWN_CAR_AGE'].isnull().sum()
daf['OCCUPATION_TYPE'].isnull().sum()
daf['CNT_FAM_MEMBERS'].isnull().sum()
daf['EXT_SOURCE_1'].isnull().sum()
daf['EXT_SOURCE_2'].isnull().sum()
```

Out[7]:
```
660
```

In [8]:
```
# understanding that there are many missing values so it is better to have a full d
# Droping the missing value - .dropna() - will drop the row where values are missin
daf.dropna()
```

Out[8]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OV
71	100083	0	Cash loans	M	Y	
124	100145	0	Cash loans	F	Y	
152	100179	0	Cash loans	F	Y	
161	100190	0	Cash loans	M	Y	
255	100295	1	Cash loans	M	Y	
...	...	...	...	...	...	
307358	456083	0	Cash loans	F	Y	
307359	456084	0	Cash loans	F	Y	
307407	456140	1	Cash loans	F	Y	
307456	456195	0	Cash loans	F	Y	
307482	456226	0	Cash loans	F	Y	

8602 rows × 122 columns

In [10]:
```
print(daf)
```

```
        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
...            ...     ...                ...         ...          ...
307506      456251       0         Cash loans           M            N
307507      456252       0         Cash loans           F            N
307508      456253       0         Cash loans           F            N
307509      456254       1         Cash loans           F            N
307510      456255       0         Cash loans           F            N

       FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  \
0                    Y             0          202500.0    406597.5
1                    N             0          270000.0   1293502.5
2                    Y             0           67500.0    135000.0
3                    Y             0          135000.0    312682.5
4                    Y             0          121500.0    513000.0
...                ...           ...               ...         ...
307506               N             0          157500.0    254700.0
307507               Y             0           72000.0    269550.0
307508               Y             0          153000.0    677664.0
307509               Y             0          171000.0    370107.0
307510               N             0          157500.0    675000.0

        AMT_ANNUITY  ...  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19 FLAG_DOCUMENT_20  \
0           24700.5  ...                 0                0                 0
1           35698.5  ...                 0                0                 0
2            6750.0  ...                 0                0                 0
3           29686.5  ...                 0                0                 0
4           21865.5  ...                 0                0                 0
...             ...  ...               ...              ...               ...
307506      27558.0  ...                 0                0                 0
307507      12001.5  ...                 0                0                 0
307508      29979.0  ...                 0                0                 0
307509      20205.0  ...                 0                0                 0
307510      49117.5  ...                 0                0                 0

       FLAG_DOCUMENT_21 AMT_REQ_CREDIT_BUREAU_HOUR AMT_REQ_CREDIT_BUREAU_DAY  \
0                     0                       0.0                       0.0
1                     0                       0.0                       0.0
2                     0                       0.0                       0.0
3                     0                       NaN                       NaN
4                     0                       0.0                       0.0
...                 ...                       ...                       ...
307506                0                       NaN                       NaN
307507                0                       NaN                       NaN
307508                0                       1.0                       0.0
307509                0                       0.0                       0.0
307510                0                       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
...                            ...                        ...
307506                         NaN                        NaN
307507                         NaN                        NaN
307508                         0.0                        1.0
307509                         0.0                        0.0
307510                         0.0                        2.0
```

```
        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
...                           ...                          ...
307506                        NaN                          NaN
307507                        NaN                          NaN
307508                        0.0                          1.0
307509                        0.0                          0.0
307510                        0.0                          1.0

[307511 rows x 122 columns]
```

In [11]:
```
#To Check for missing columns
missing_columns = daf1.columns[daf1.isnull().any()].tolist()

if missing_columns:
    print(f"Missing columns: {missing_columns}")
else:
    print("No missing columns.")
```

```
Missing columns: ['AMT_ANNUITY', 'AMT_CREDIT', 'AMT_DOWN_PAYMENT', 'AMT_GOODS_PRIC
E', 'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY', 'RATE_INTEREST_PRIVILEGED', 'NAM
E_TYPE_SUITE', 'CNT_PAYMENT', 'PRODUCT_COMBINATION', 'DAYS_FIRST_DRAWING', 'DAYS_F
IRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE', 'DAYS_TERMINATION', 'NFLA
G_INSURED_ON_APPROVAL']
```

In [12]:
```
# Droping the missing value - .dropna() - will drop the row where values are missin
daf1.dropna()
```

Out[12]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	A
598	2388655	414811	Consumer loans	14152.545	153387.0	
21366	1184010	252161	Consumer loans	3136.275	29781.0	
24027	2144692	423348	Consumer loans	2640.195	26145.0	
43927	2697394	178347	Consumer loans	10324.665	101002.5	
115115	2403906	268507	Consumer loans	13452.660	145800.0	
...	...	...	...	...	...	
1603346	1928485	386819	Consumer loans	45418.500	562500.0	
1619458	1347931	336203	Consumer loans	9207.180	113400.0	
1644524	2002593	168701	Consumer loans	3518.460	38524.5	
1645311	2396619	341729	Consumer loans	17179.380	171477.0	
1663414	1328802	105065	Consumer loans	6357.375	68553.0	

71 rows × 37 columns

In [14]:
```
daf.tail()
```

Out[14]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OV
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

5 rows × 122 columns

In [16]:
```
#General Overview
daf.head()
```

Out[16]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
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	

5 rows × 122 columns

In [17]:
```
#General Overview
daf.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
```

In [19]:
```
#merging the cell to analyse both application and previous application easily
# Concate two dataframes - .concat()
daf3=pd.concat([daf,daf1],ignore_index=True)
daf3
```

Out[19]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_O
0	100002	1.0	Cash loans	M	N	
1	100003	0.0	Cash loans	F	N	
2	100004	0.0	Revolving loans	M	Y	
3	100006	0.0	Cash loans	F	N	
4	100007	0.0	Cash loans	M	N	
...	...	...	...	...	...	
1977720	352015	NaN	Consumer loans	NaN	NaN	
1977721	334635	NaN	Consumer loans	NaN	NaN	
1977722	249544	NaN	Consumer loans	NaN	NaN	
1977723	400317	NaN	Cash loans	NaN	NaN	
1977724	261212	NaN	Cash loans	NaN	NaN	

1977725 rows × 151 columns

In [23]:
```
#Checking basic Information to understand the data
daf3.tail()
```

Out[23]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_O
1977720	352015	NaN	Consumer loans	NaN	NaN	
1977721	334635	NaN	Consumer loans	NaN	NaN	
1977722	249544	NaN	Consumer loans	NaN	NaN	
1977723	400317	NaN	Cash loans	NaN	NaN	
1977724	261212	NaN	Cash loans	NaN	NaN	

5 rows × 151 columns

In [21]:
```
daf3.head()
```

Out[21]:

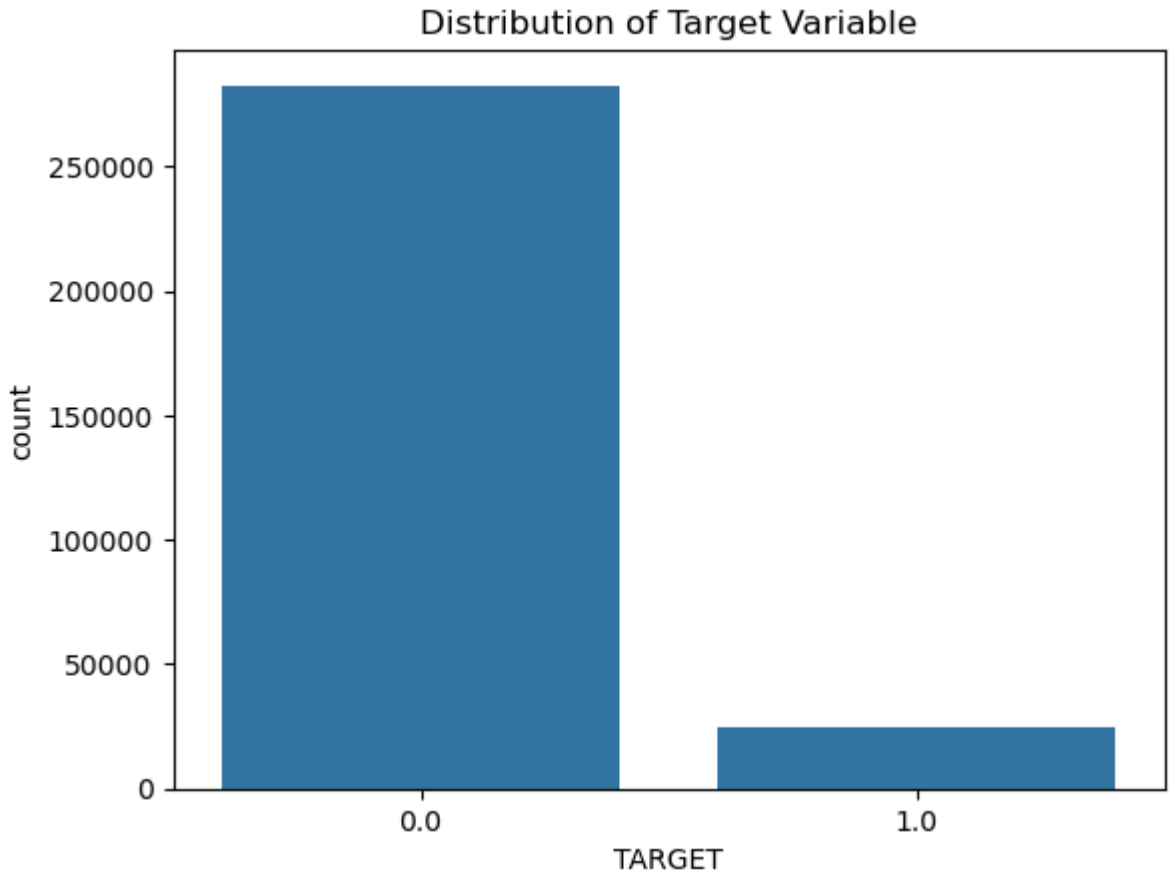
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_RE
0	100002	1.0	Cash loans	M	N	
1	100003	0.0	Cash loans	F	N	
2	100004	0.0	Revolving loans	M	Y	
3	100006	0.0	Cash loans	F	N	
4	100007	0.0	Cash loans	M	N	

5 rows × 151 columns

In [22]:
```
daf3.info()
```

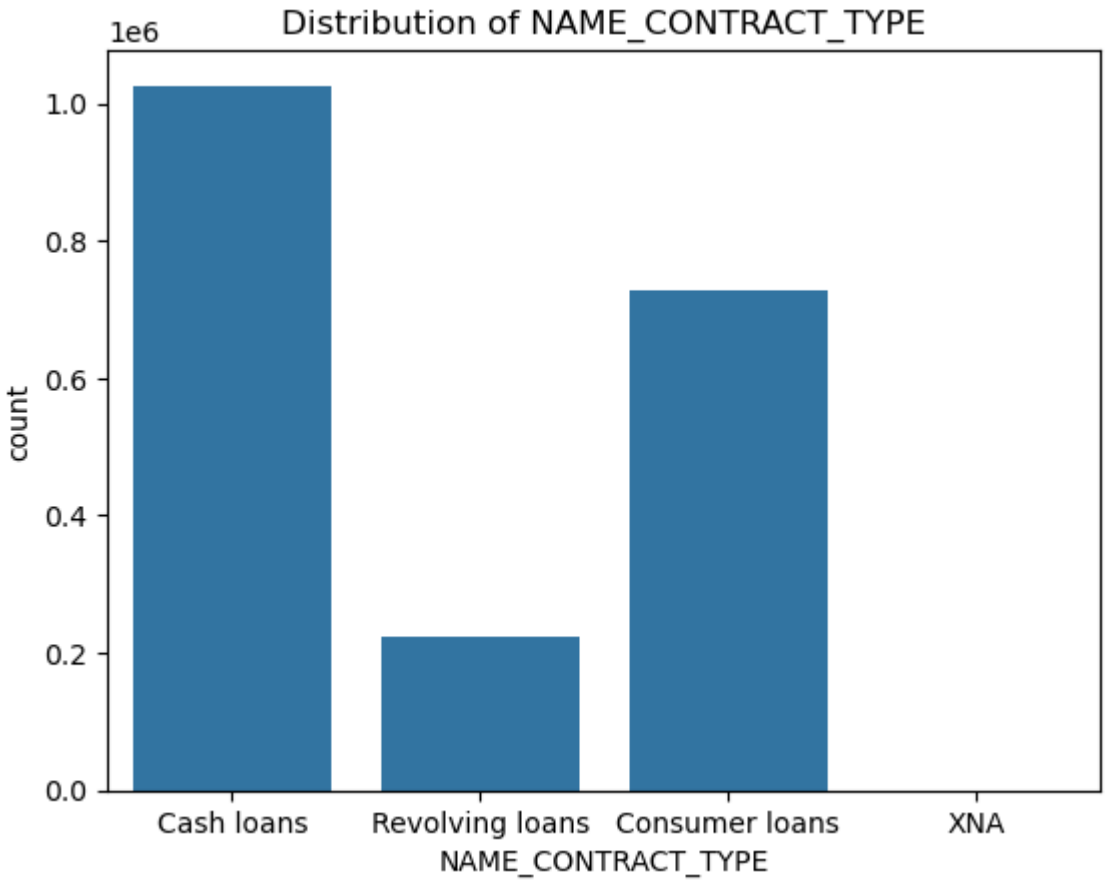
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1977725 entries, 0 to 1977724
Columns: 151 entries, SK_ID_CURR to NFLAG_INSURED_ON_APPROVAL
dtypes: float64(120), int64(2), object(29)
memory usage: 2.2+ GB
```

In [27]:
```
#Visualizing the data for better understanding
# 1 client with payment difficulties: he/she had late payment more than X days on
#at least one of the first Y installments of the loan in our sample,
#0 - all other cases
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
sns.countplot(x='TARGET', data=daf3)
plt.title('Distribution of Target Variable')
plt.show()
#Risk Assessment: Since the  problem involves identifying clients with payment diff
#the class 1 instances (payment difficulties) are of particular interest. The lower
#suggests that instances of payment difficulties are relatively less common in the
#Understanding the characteristics and patterns associated with these instances is
```

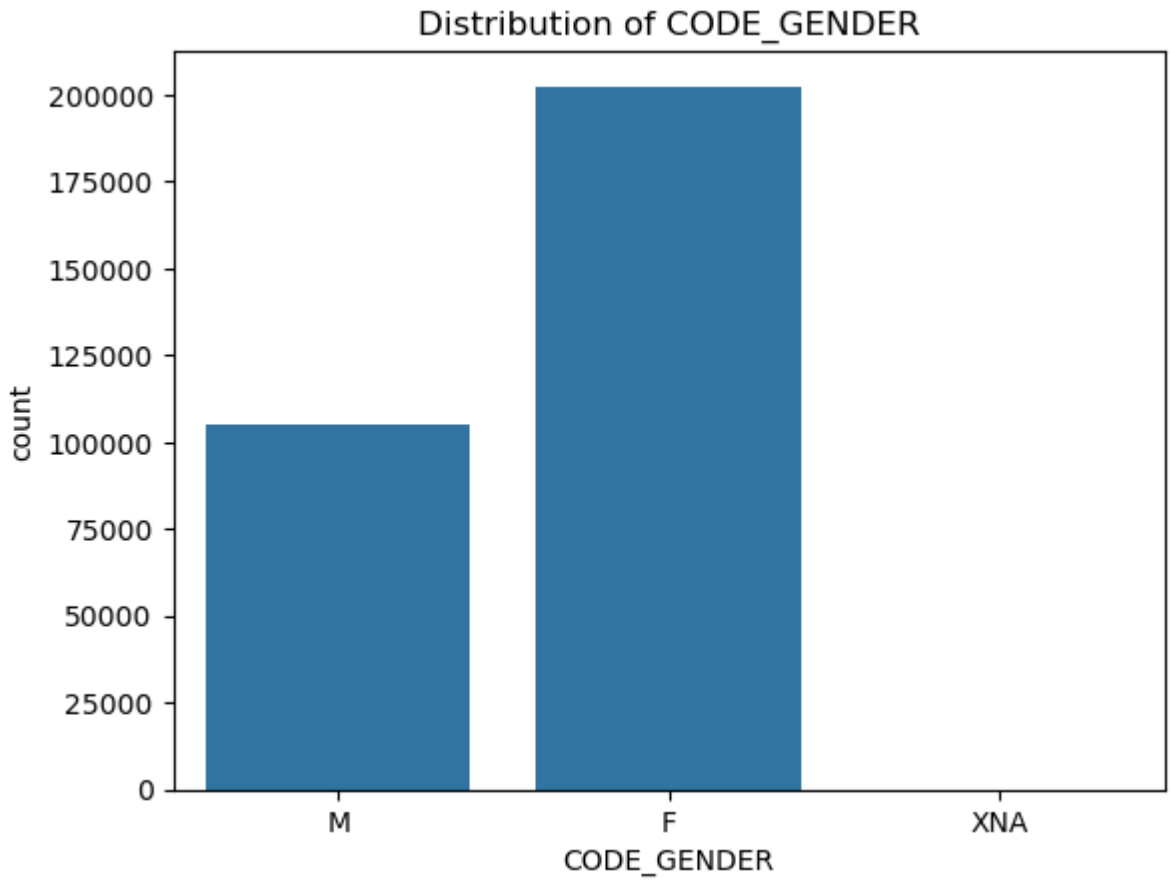


In [28]:
```
# categorical variables for analysis
categorical_columns = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_C

for column in categorical_columns:
    print(f"\nDistribution of {column}:")
    print(daf3[column].value_counts(normalize=True))
    sns.countplot(x=column, data=daf3)
    plt.title(f'Distribution of {column}')
    plt.show()
# Most of the contracts are "Cash loans," followed by "Consumer loans," and a small
#The majority of clients in the dataset are female (65.83%), while males make up a
#A significant portion of clients in the dataset owns real estate, while a smaller
#The majority of clients do not own a car, while a smaller portion owns a car.
```

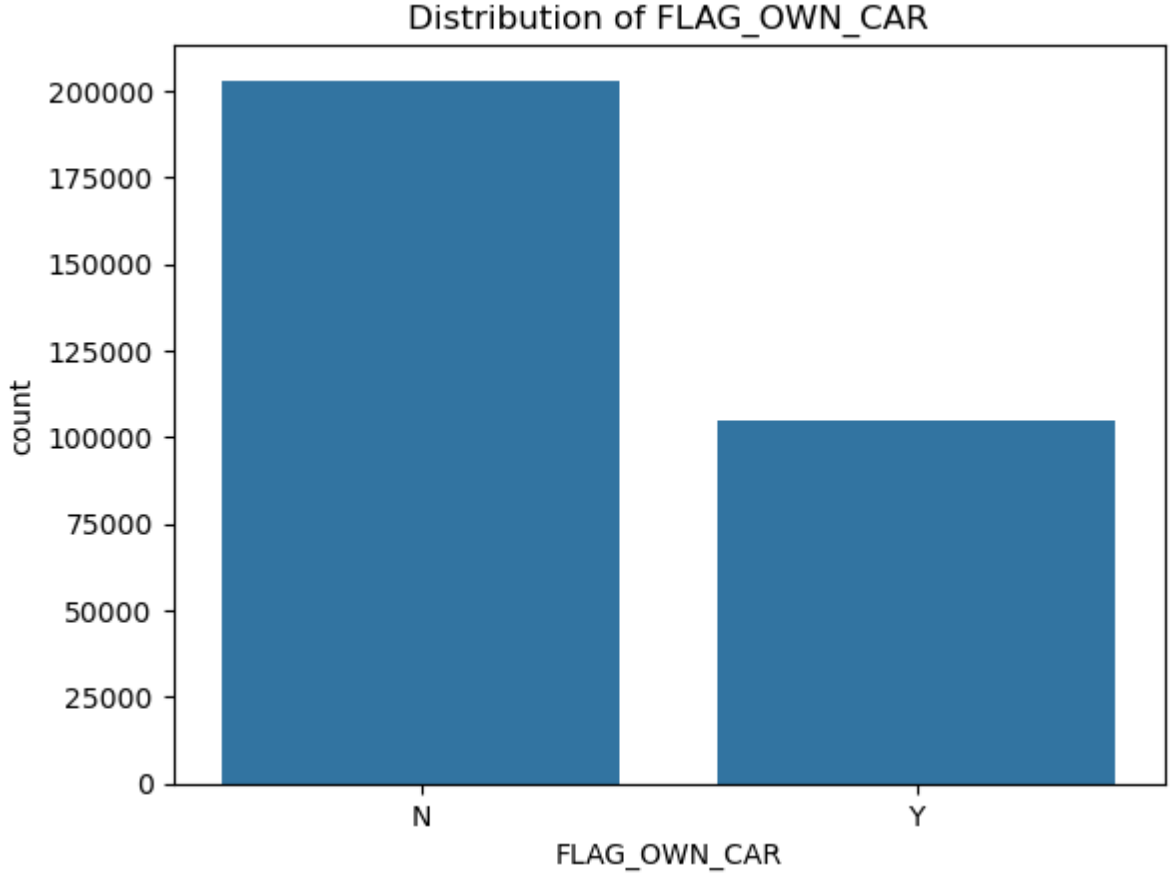
```
Distribution of NAME_CONTRACT_TYPE:
Cash loans         0.518669
Consumer loans     0.368682
Revolving loans    0.112474
XNA                0.000175
Name: NAME_CONTRACT_TYPE, dtype: float64
```



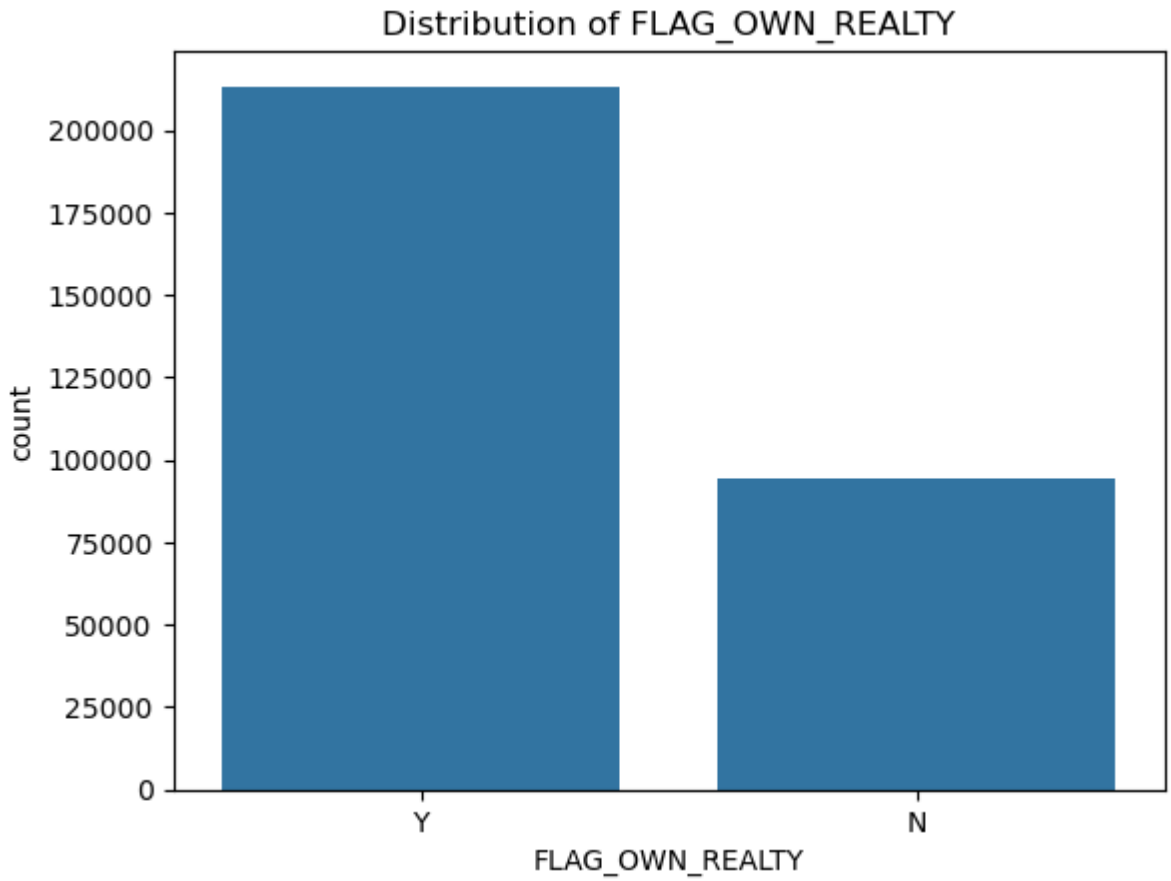

```
Distribution of CODE_GENDER:
F      0.658344
M      0.341643
XNA    0.000013
Name: CODE_GENDER, dtype: float64
```




```
Distribution of FLAG_OWN_CAR:
N    0.659892
Y    0.340108
Name: FLAG_OWN_CAR, dtype: float64
```

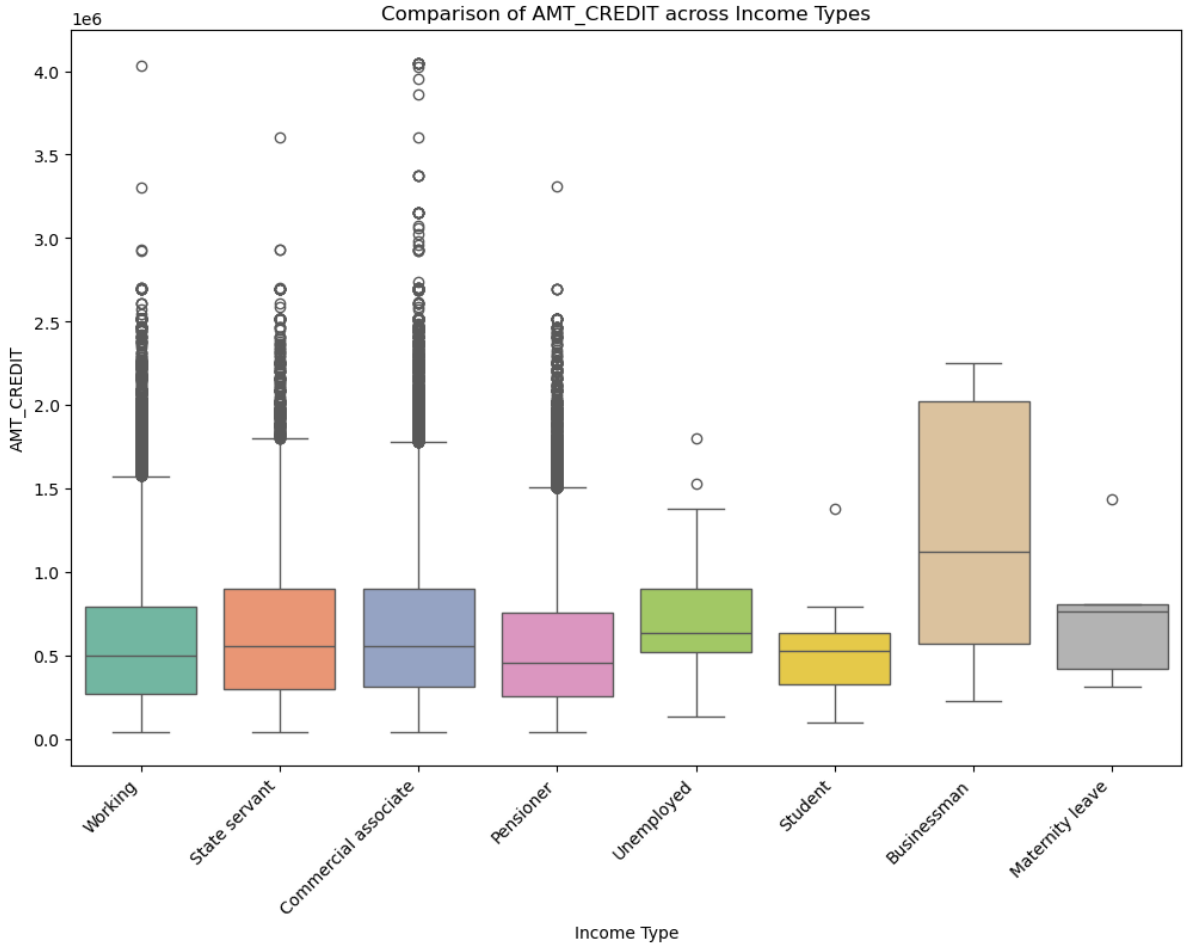



```
Distribution of FLAG_OWN_REALTY:
Y    0.693673
N    0.306327
Name: FLAG_OWN_REALTY, dtype: float64
```

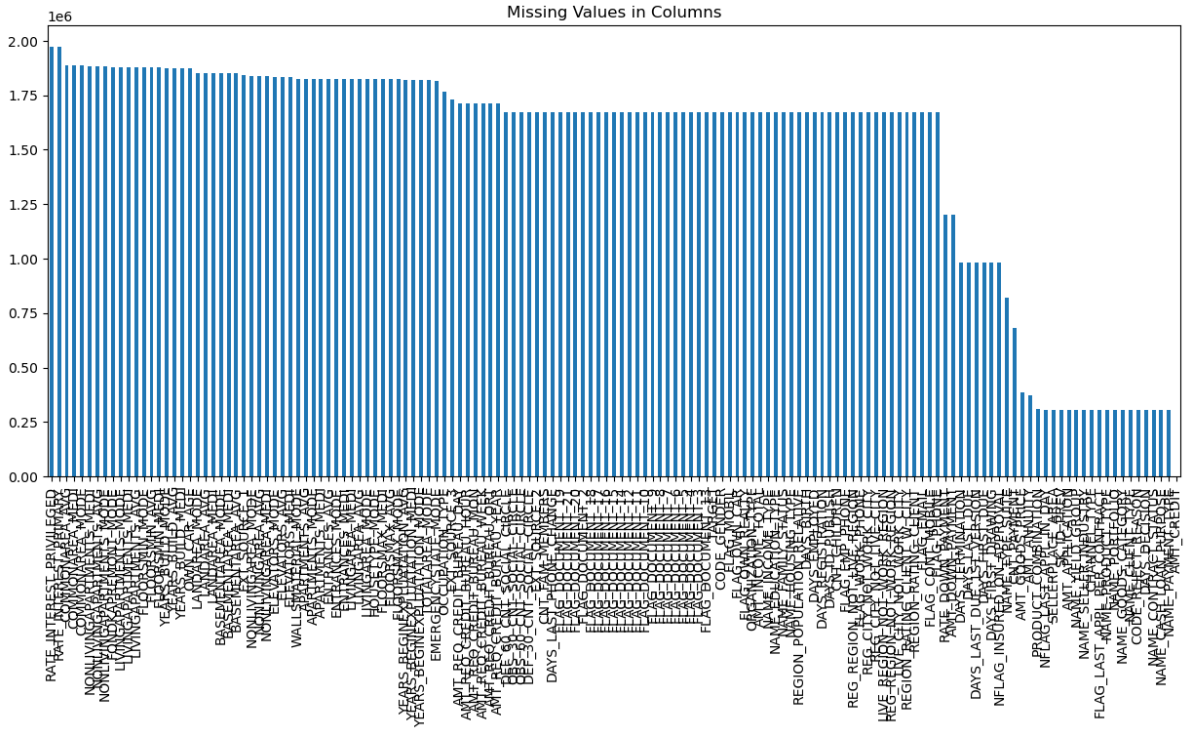


In [41]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Defining a color palette
colors = sns.color_palette("Set2")  # You can choose a different palette

# Selecting columns for the plot
columns_of_interest = ['AMT_CREDIT', 'NAME_INCOME_TYPE']

# Creating a box plot with different colors
plt.figure(figsize=(12, 8))
sns.boxplot(x='NAME_INCOME_TYPE', y='AMT_CREDIT', data=daf3, palette=colors)
plt.title('Comparison of AMT_CREDIT across Income Types')
plt.xlabel('Income Type')
plt.ylabel('AMT_CREDIT')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11304\2502307636.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.
14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='NAME_INCOME_TYPE', y='AMT_CREDIT', data=daf3, palette=colors)
```

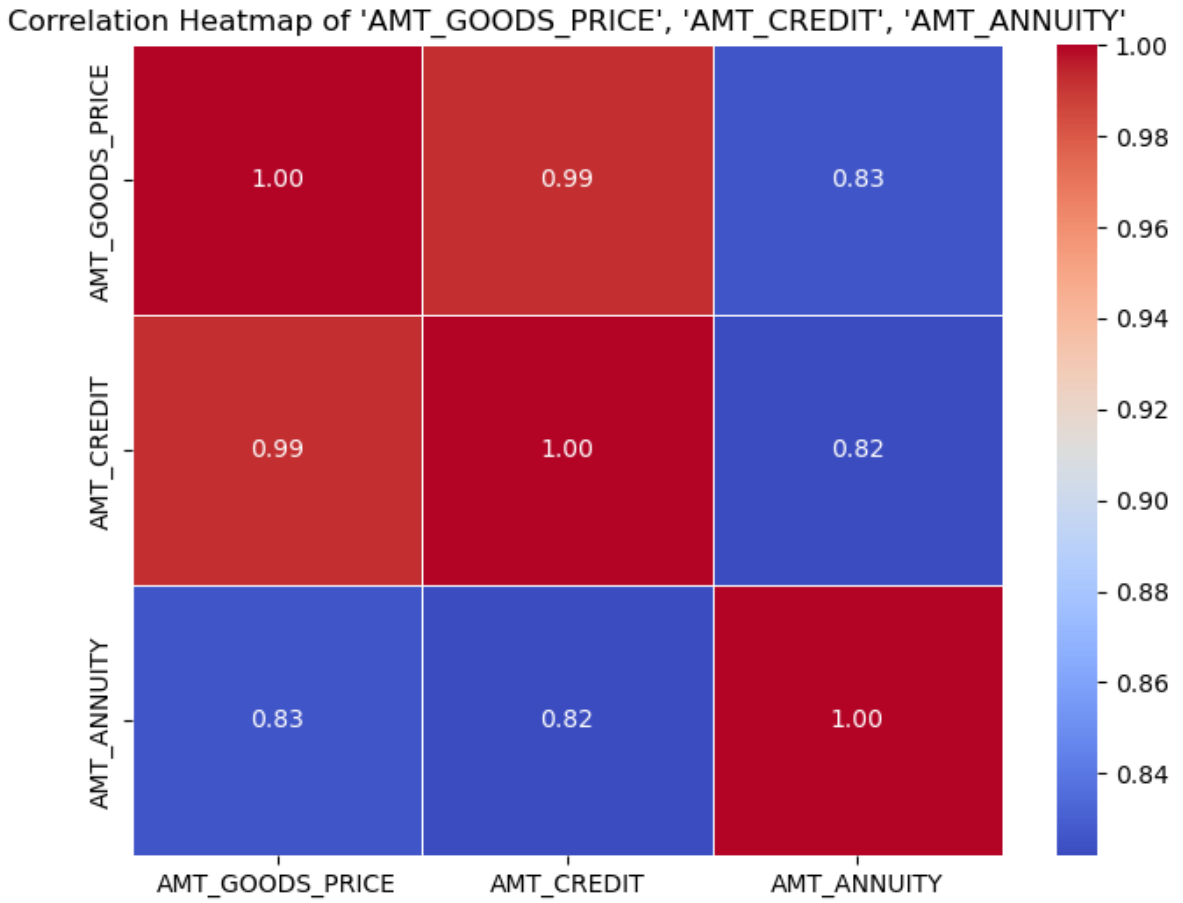


In [30]:
```
missing_values = daf3.isnull().sum()
missing_values[missing_values > 0].sort_values(ascending=False).plot(kind='bar', fi
plt.title('Missing Values in Columns')
plt.show()
```

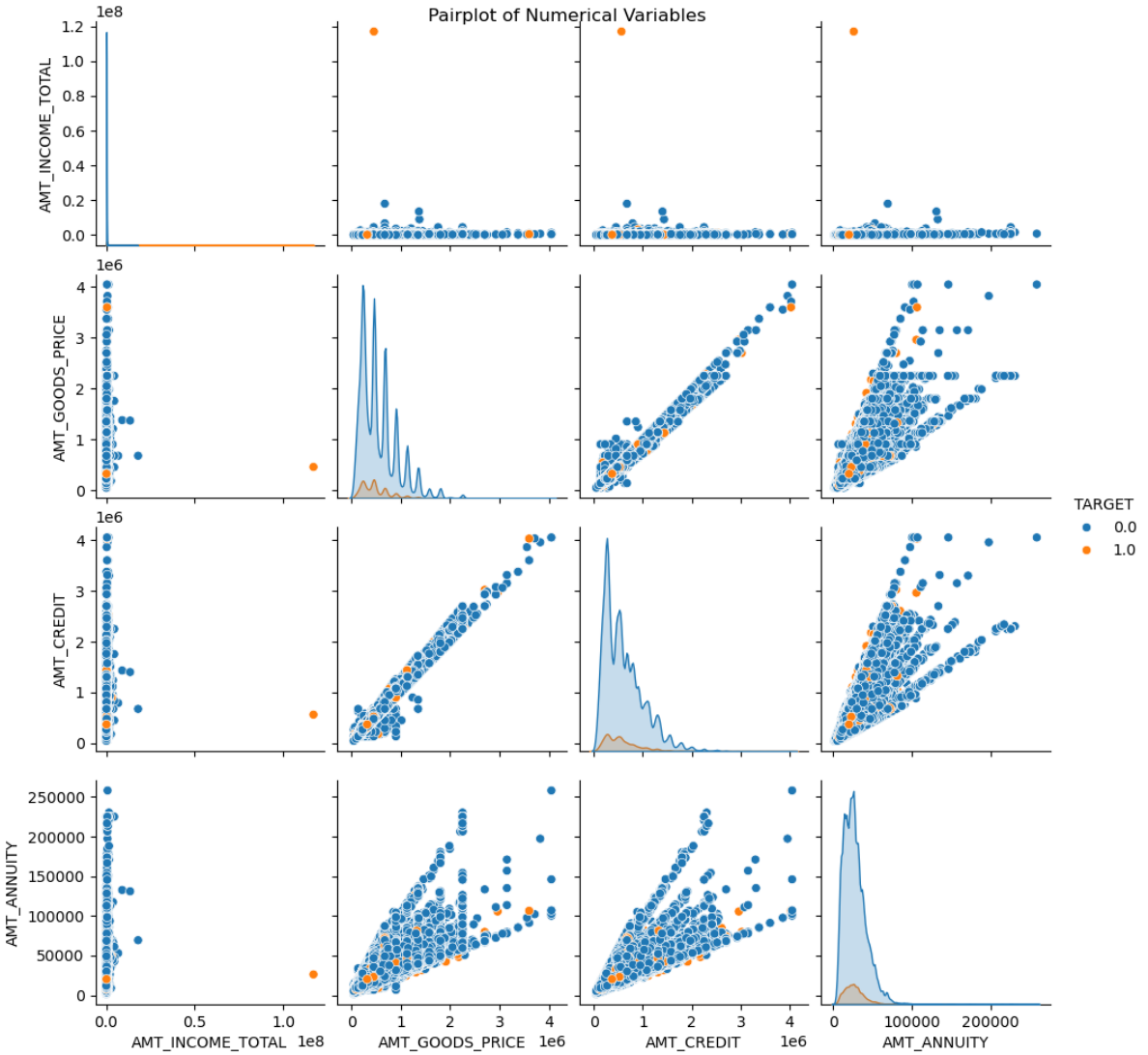


In [38]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Selecting columns for correlation analysis
columns_of_interest = ['AMT_GOODS_PRICE', 'AMT_CREDIT', 'AMT_ANNUITY']
```

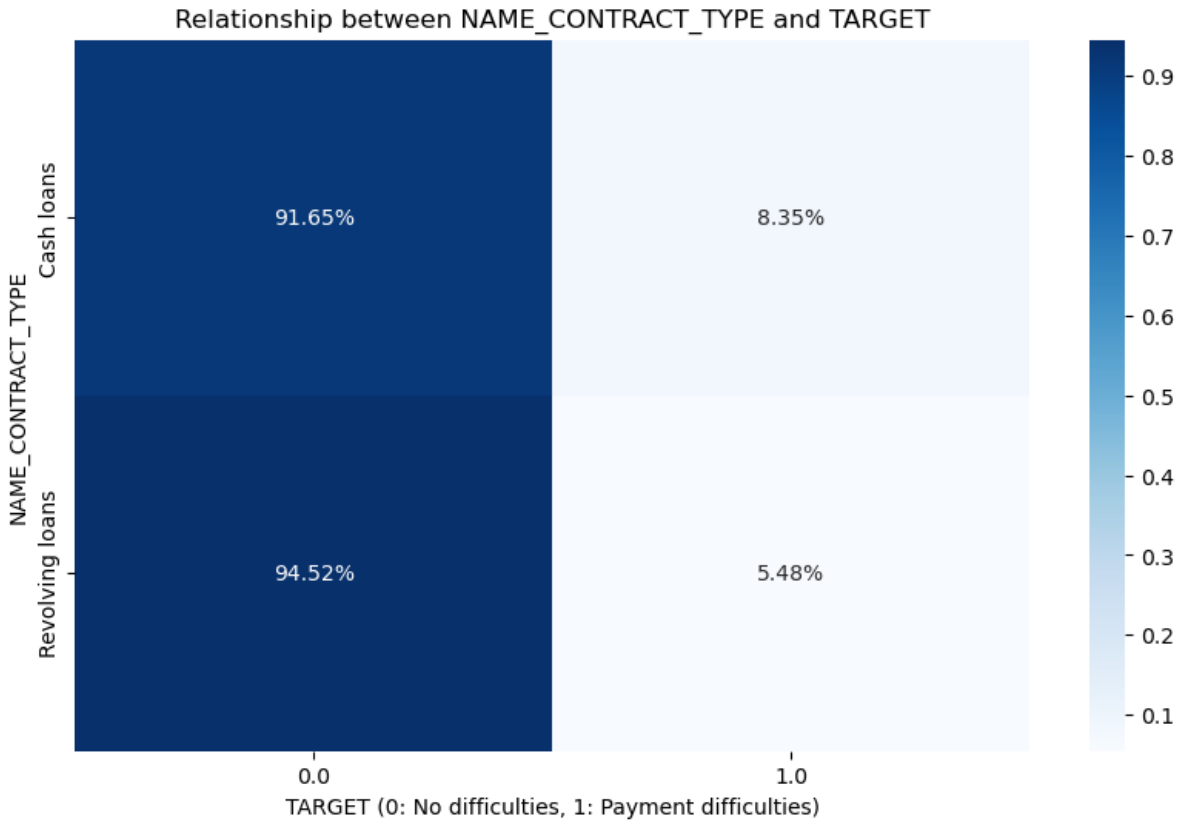
```
correlation_matrix = daf3[columns_of_interest].corr()

# Creating a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=
plt.title("Correlation Heatmap of 'AMT_GOODS_PRICE', 'AMT_CREDIT', 'AMT_ANNUITY'")
plt.show()
```

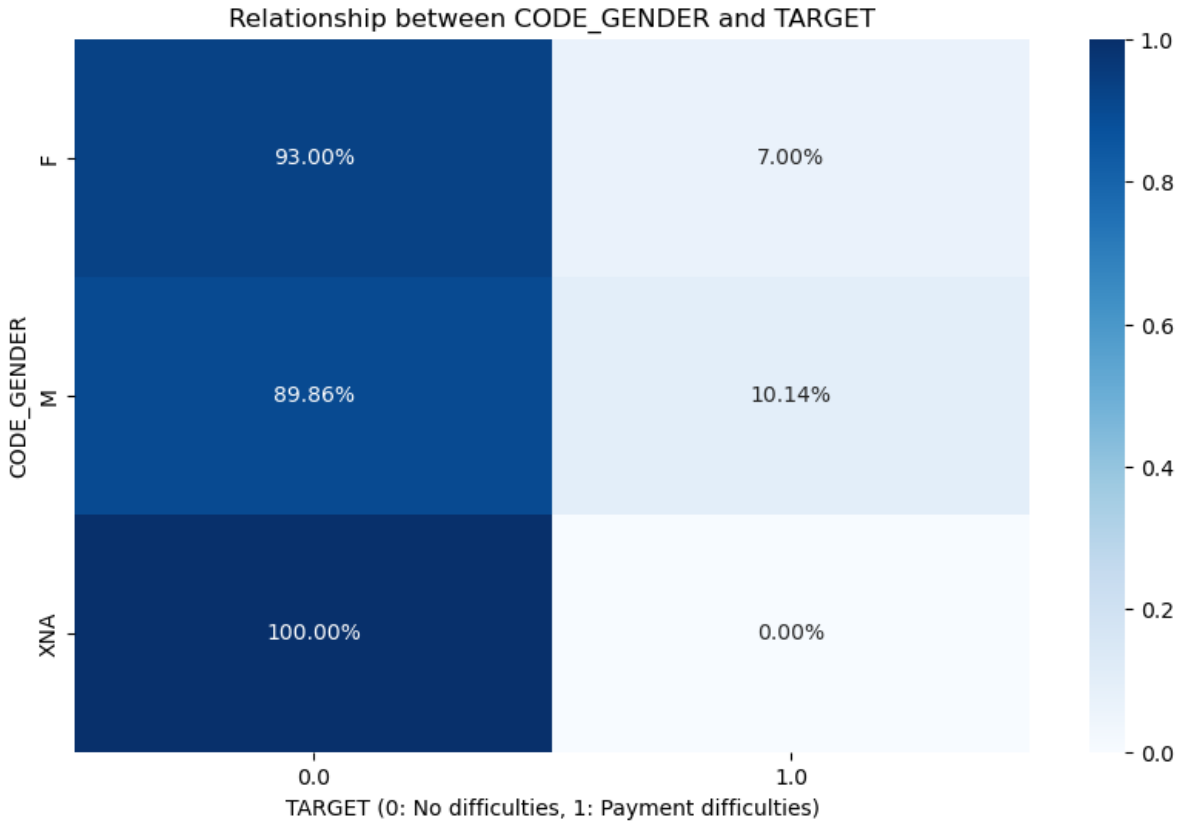


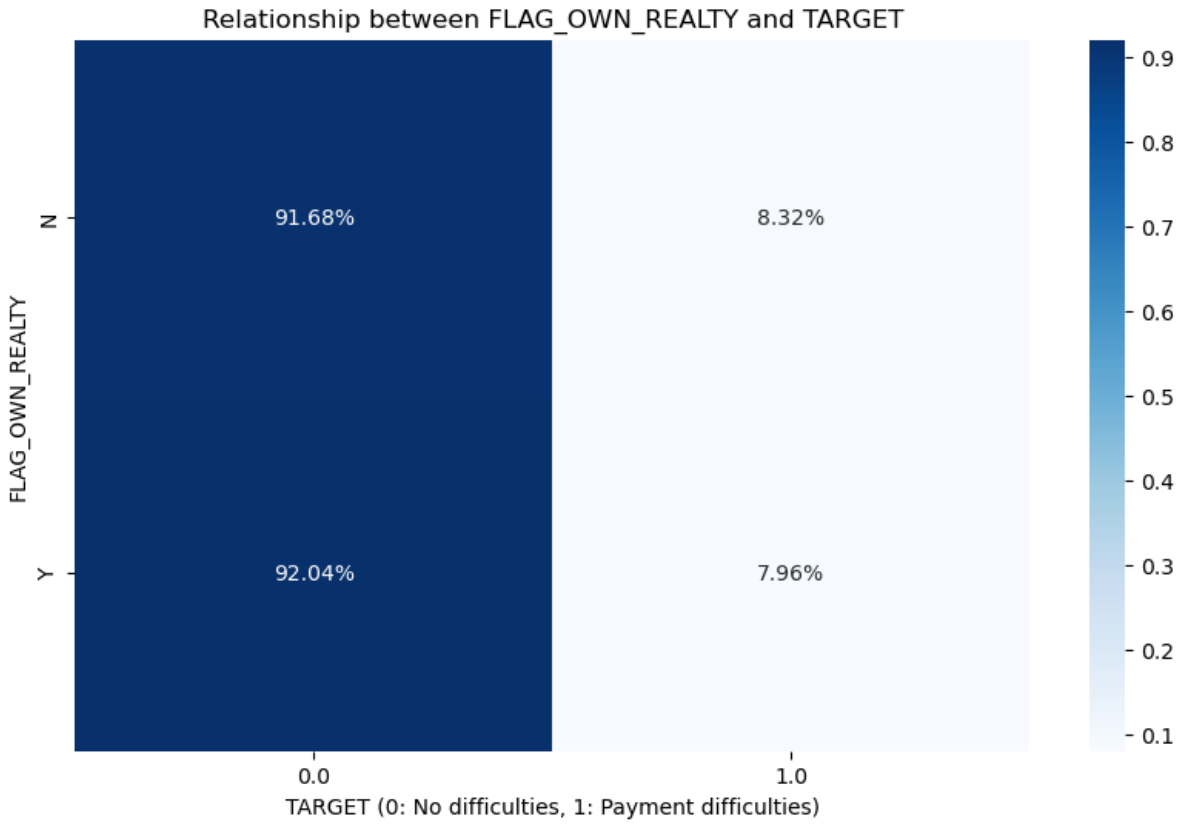
In [35]:
```
#7.1 Pairplot:
sns.pairplot(daf3[['AMT_INCOME_TOTAL','AMT_GOODS_PRICE', 'AMT_CREDIT', 'AMT_ANNUITY
plt.suptitle("Pairplot of Numerical Variables")
plt.show()
```

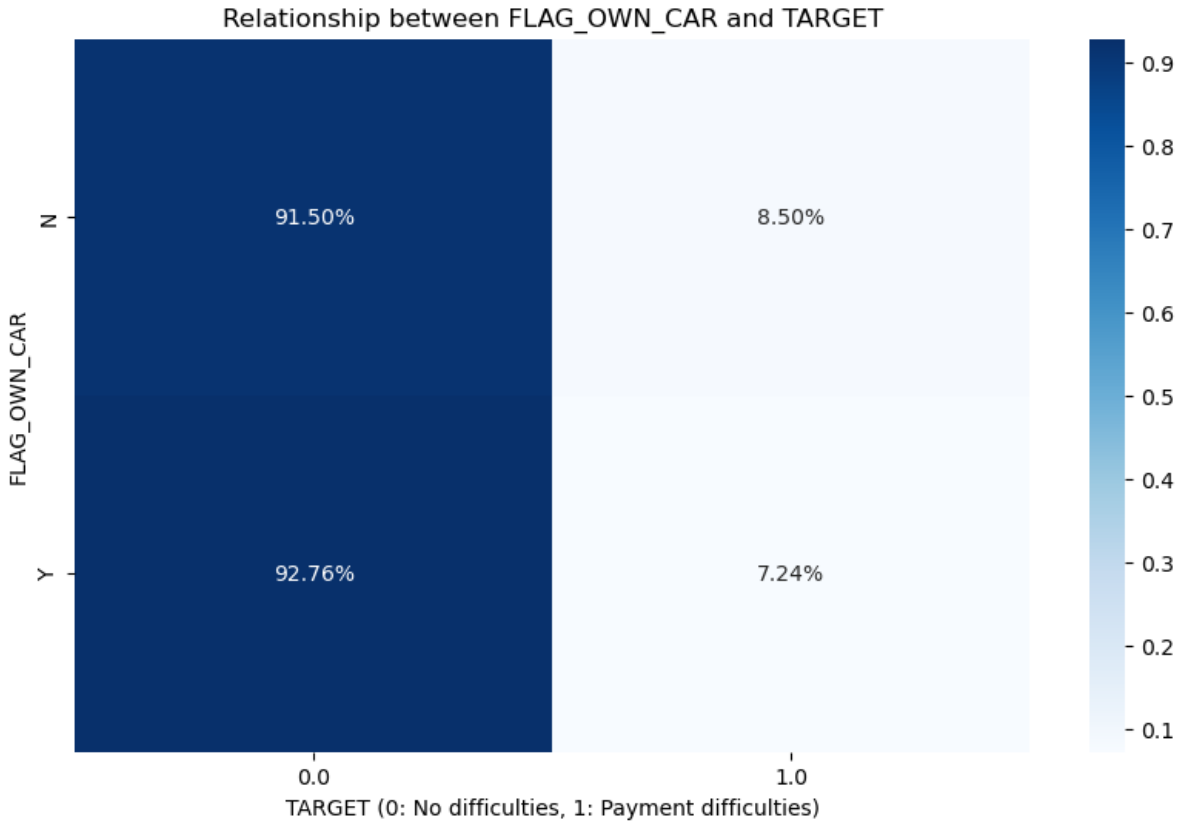


In [33]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select categorical variables of interest
categorical_columns = ['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_REALTY', 'FLA

# Iterate through categorical variables and create cross-tabulations
for column in categorical_columns:
    cross_tab = pd.crosstab(daf3[column], daf3['TARGET'], normalize='index')

    # Plot the cross-tabulation
    plt.figure(figsize=(10, 6))
    sns.heatmap(cross_tab, annot=True, fmt=".2%", cmap="Blues", cbar=True)
    plt.title(f'Relationship between {column} and TARGET')
    plt.xlabel('TARGET (0: No difficulties, 1: Payment difficulties)')
    plt.ylabel(column)
    plt.show()
```

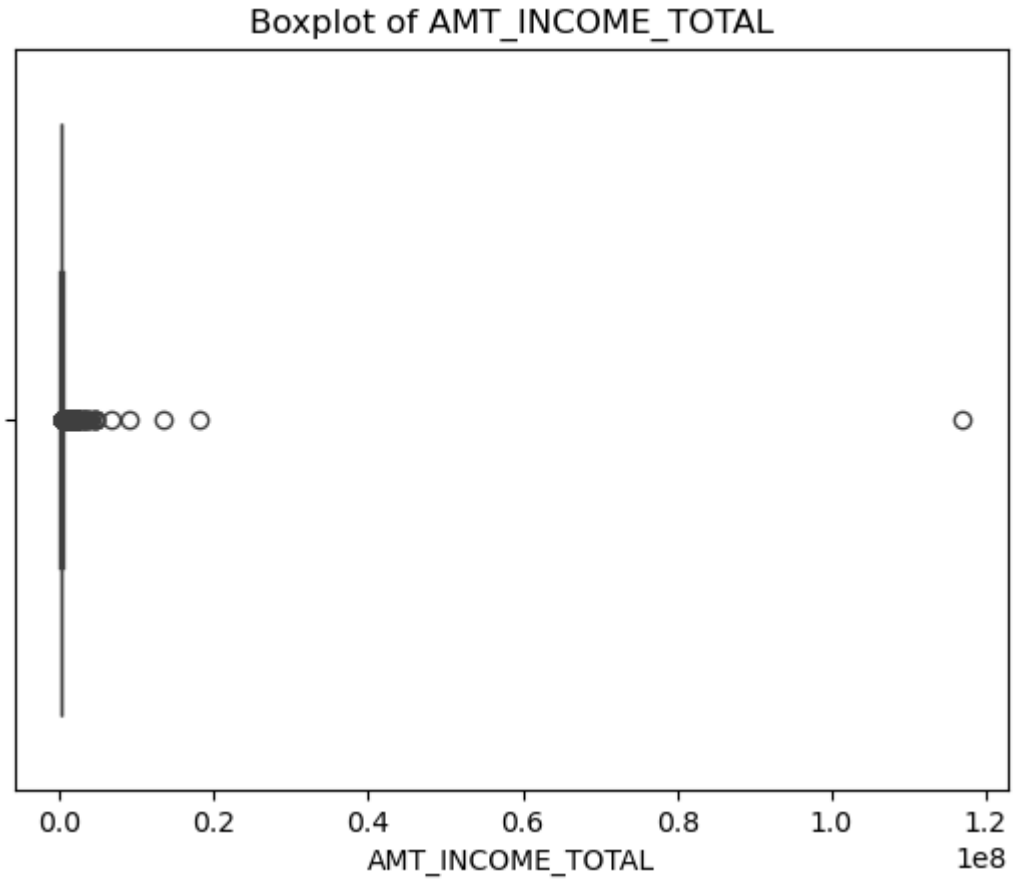

Relationship between NAME_CONTRACT_TYPE and TARGET
NAME_CONTRACT_TYPE
Cash loans
Revolving loans
91.65%
8.35%
94.52%
5.48%
0.0
1.0
TARGET (0: No difficulties, 1: Payment difficulties)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

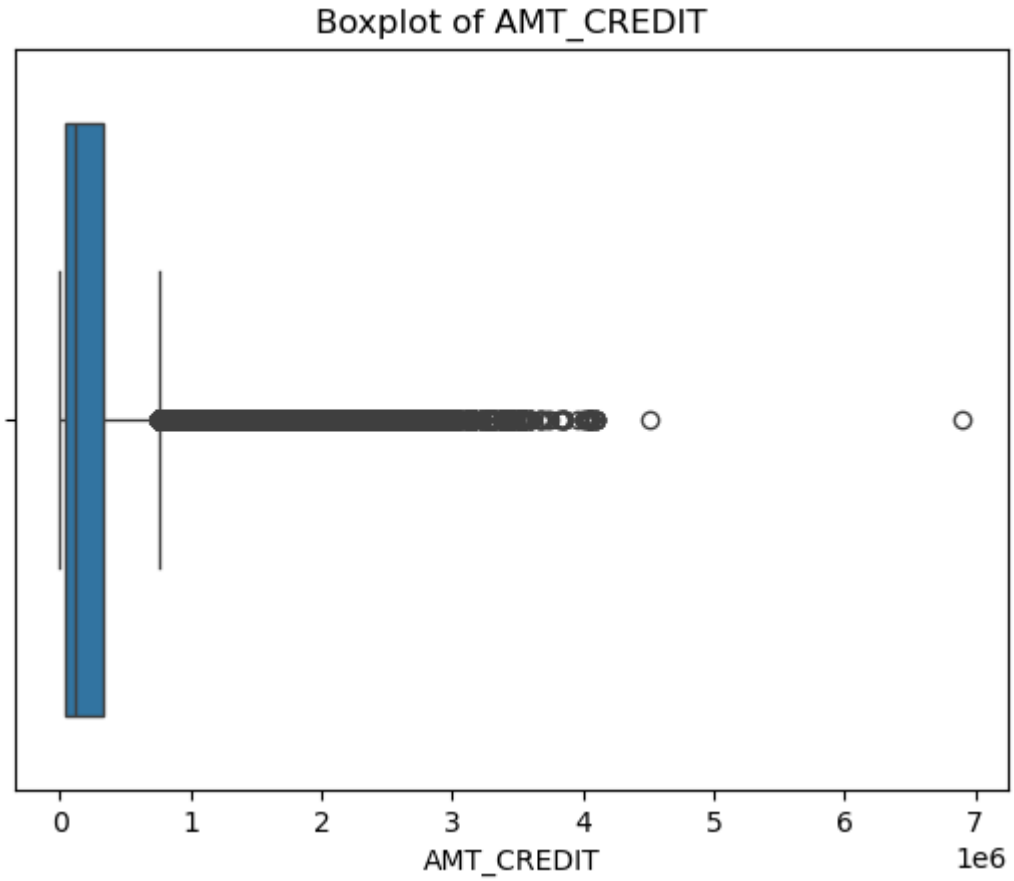

Relationship between CODE_GENDER and TARGET
CODE_GENDER
F
M
XNA
93.00%
7.00%
89.86%
10.14%
100.00%
0.00%
0.0
1.0
TARGET (0: No difficulties, 1: Payment difficulties)
1.0
0.8
0.6
0.4
0.2
0.0

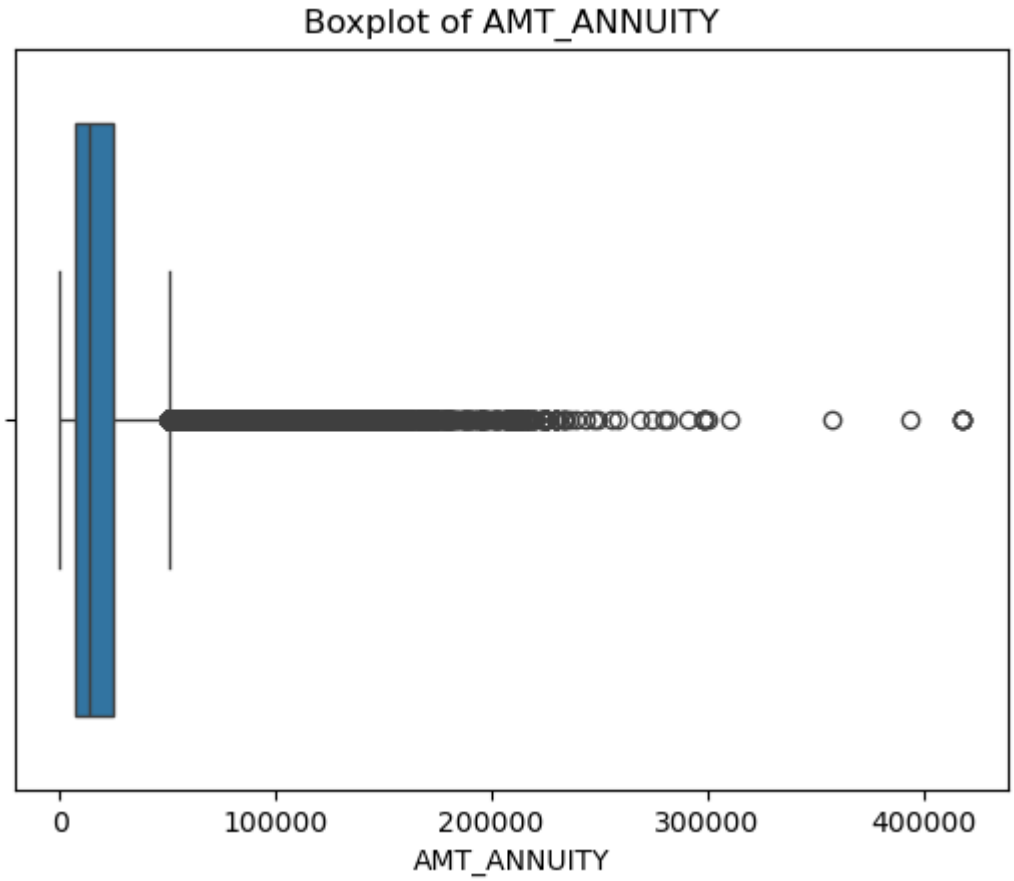




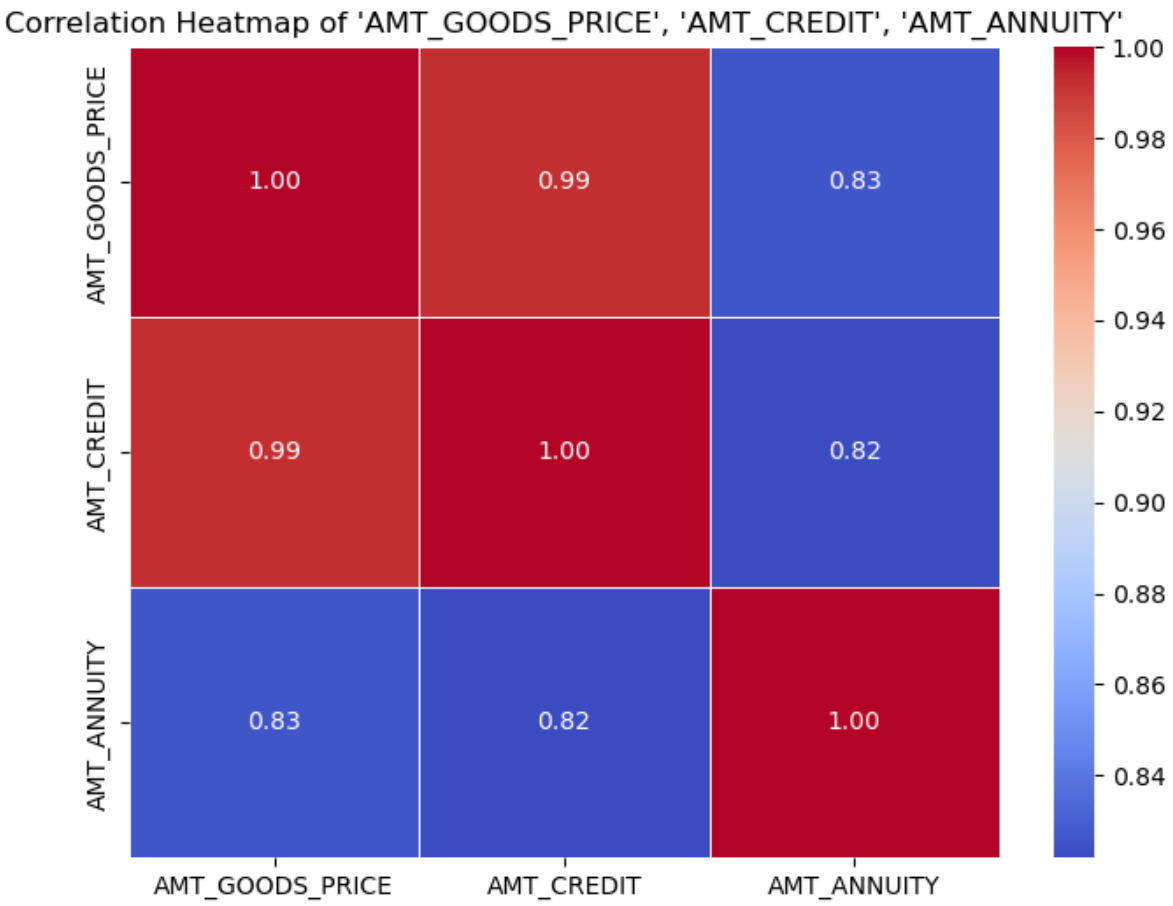
In [34]:
```
for column in numerical_columns:
    sns.boxplot(x=daf3[column])
    plt.title(f'Boxplot of {column}')
    plt.show()
```


Boxplot of AMT_INCOME_TOTAL
0.0
0.2
0.4
0.6
0.8
1.0
1.2
AMT_INCOME_TOTAL
1e8


Boxplot of AMT_CREDIT
0
1
2
3
4
5
6
7
AMT_CREDIT
1e6



In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select columns for correlation analysis
columns_of_interest = ['AMT_GOODS_PRICE', 'AMT_CREDIT', 'AMT_ANNUITY']
correlation_matrix = daf3[columns_of_interest].corr()

# Create a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=
plt.title("Correlation Heatmap of 'AMT_GOODS_PRICE', 'AMT_CREDIT', 'AMT_ANNUITY'")
plt.show()
```


Correlation Heatmap of 'AMT_GOODS_PRICE', 'AMT_CREDIT', 'AMT_ANNUITY'
AMT_GOODS_PRICE
AMT_CREDIT
AMT_ANNUITY
1.00
0.99
0.83
0.99
1.00
0.82
0.83
0.82
1.00
AMT_GOODS_PRICE
AMT_CREDIT
AMT_ANNUITY
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84