Importing the libraries

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import openai as ai
```

In [ ]:
```
file = r"E:\CSV_Files\Project\Credit EDA Case Study\application_data.csv"
```

In [ ]:
```
key = "sk-xxxxxxx"
```

Create the function using OpenAI for ask question if require

In [ ]:
```
# Set your OpenAI API key here
api_key = key

# Initialize the OpenAI API client
ai.api_key = api_key


# Create the function for ask a questions.

def ask(Question):
  response = ai.Completion.create(
      engine="text-davinci-002",  # This is the GPT-3.5 engine name
      prompt=Question,  # The starting text for the model to continue
      max_tokens=100  # The maximum number of tokens to generate in the output
  )
  generated_text = response['choices'][0]['text']
  return generated_text.strip()
```

In [ ]:
```
print(ask('How are you'))
```

```
doing, Susan?

I'm doing well. How are you?
```

In [ ]:
```
data = pd.read_csv(file) # load the file
```

In [ ]:
```
df = data.copy() # create the duplicate copy for original datasets
df.head()
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN
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

Data Cleaning

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

Out[ ]:
```
(307511, 122)
```

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

Out[ ]:
```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'CODE_GENDER',
       'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY',
       ...
       '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'],
      dtype='object', length=122)
```

In [ ]:
```
pd.set_option('display.max_columns', None)  # Display all columns
```

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

Out[ ]:
```
SK_ID_CURR                      0
TARGET                          0
NAME_CONTRACT_TYPE              0
CODE_GENDER                     0
FLAG_OWN_CAR                    0
                               ...
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
Length: 122, dtype: int64
```

In [ ]:
```
null_col = df.isnull().sum() / df.shape[0] * 100
null_col
```

Out[ ]:
```
SK_ID_CURR                    0.000000
TARGET                        0.000000
NAME_CONTRACT_TYPE            0.000000
CODE_GENDER                   0.000000
FLAG_OWN_CAR                  0.000000
                                ...
AMT_REQ_CREDIT_BUREAU_DAY    13.501631
AMT_REQ_CREDIT_BUREAU_WEEK   13.501631
AMT_REQ_CREDIT_BUREAU_MON    13.501631
AMT_REQ_CREDIT_BUREAU_QRT    13.501631
AMT_REQ_CREDIT_BUREAU_YEAR   13.501631
Length: 122, dtype: float64
```

In [ ]:
```
null_col = null_col.loc[null_col.values > 50]
null_col
```

Out[ ]:
```
OWN_CAR_AGE                 65.990810
EXT_SOURCE_1                56.381073
APARTMENTS_AVG              50.749729
BASEMENTAREA_AVG            58.515956
YEARS_BUILD_AVG             66.497784
COMMONAREA_AVG              69.872297
ELEVATORS_AVG               53.295980
ENTRANCES_AVG               50.348768
FLOORSMIN_AVG               67.848630
LANDAREA_AVG                59.376738
LIVINGAPARTMENTS_AVG        68.354953
LIVINGAREA_AVG              50.193326
NONLIVINGAPARTMENTS_AVG     69.432963
NONLIVINGAREA_AVG           55.179164
APARTMENTS_MODE             50.749729
BASEMENTAREA_MODE           58.515956
YEARS_BUILD_MODE            66.497784
COMMONAREA_MODE             69.872297
ELEVATORS_MODE              53.295980
ENTRANCES_MODE              50.348768
FLOORSMIN_MODE              67.848630
LANDAREA_MODE               59.376738
LIVINGAPARTMENTS_MODE       68.354953
LIVINGAREA_MODE             50.193326
NONLIVINGAPARTMENTS_MODE    69.432963
NONLIVINGAREA_MODE          55.179164
APARTMENTS_MEDI             50.749729
BASEMENTAREA_MEDI           58.515956
YEARS_BUILD_MEDI            66.497784
COMMONAREA_MEDI             69.872297
ELEVATORS_MEDI              53.295980
ENTRANCES_MEDI              50.348768
FLOORSMIN_MEDI              67.848630
LANDAREA_MEDI               59.376738
LIVINGAPARTMENTS_MEDI       68.354953
LIVINGAREA_MEDI             50.193326
NONLIVINGAPARTMENTS_MEDI    69.432963
NONLIVINGAREA_MEDI          55.179164
FONDKAPREMONT_MODE          68.386172
HOUSETYPE_MODE              50.176091
WALLSMATERIAL_MODE          50.840783
dtype: float64
```

In [ ]:
```
df = df.drop(columns=null_col.index)
```

In [ ]:
```
null_col1 = df.isnull().sum() / df.shape[0] * 100
null_col1
```

Out[ ]:
```
SK_ID_CURR                     0.000000
TARGET                         0.000000
NAME_CONTRACT_TYPE             0.000000
CODE_GENDER                    0.000000
FLAG_OWN_CAR                   0.000000
                                  ...
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
Length: 81, dtype: float64
```

In [ ]:
```
null_col1 = null_col1.loc[null_col1.values > 0]
null_col1
```

Out[ ]:
```
AMT_ANNUITY                    0.003902
AMT_GOODS_PRICE                0.090403
NAME_TYPE_SUITE                0.420148
OCCUPATION_TYPE               31.345545
CNT_FAM_MEMBERS                0.000650
EXT_SOURCE_2                   0.214626
EXT_SOURCE_3                  19.825307
YEARS_BEGINEXPLUATATION_AVG   48.781019
FLOORSMAX_AVG                 49.760822
YEARS_BEGINEXPLUATATION_MODE  48.781019
FLOORSMAX_MODE                49.760822
YEARS_BEGINEXPLUATATION_MEDI  48.781019
FLOORSMAX_MEDI                49.760822
TOTALAREA_MODE                48.268517
EMERGENCYSTATE_MODE           47.398304
OBS_30_CNT_SOCIAL_CIRCLE       0.332021
DEF_30_CNT_SOCIAL_CIRCLE       0.332021
OBS_60_CNT_SOCIAL_CIRCLE       0.332021
DEF_60_CNT_SOCIAL_CIRCLE       0.332021
DAYS_LAST_PHONE_CHANGE         0.000325
AMT_REQ_CREDIT_BUREAU_HOUR    13.501631
AMT_REQ_CREDIT_BUREAU_DAY     13.501631
AMT_REQ_CREDIT_BUREAU_WEEK    13.501631
AMT_REQ_CREDIT_BUREAU_MON     13.501631
AMT_REQ_CREDIT_BUREAU_QRT     13.501631
AMT_REQ_CREDIT_BUREAU_YEAR    13.501631
dtype: float64
```

In [ ]:
```
df['CODE_GENDER'].isna()
```

Out[ ]:
```
0         False
1         False
2         False
3         False
4         False
          ...
307506    False
307507    False
307508    False
307509    False
307510    False
Name: CODE_GENDER, Length: 307511, dtype: bool
```

In [ ]:
```
df['CODE_GENDER'].value_counts()
```

Out[ ]:
```
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
```

In [ ]:
```
df['CODE_GENDER'] = df['CODE_GENDER'].str.replace('XNA','F')
```

In [ ]:
```
df['CODE_GENDER'].value_counts()
```

Out[ ]:
```
F    202452
M    105059
Name: CODE_GENDER, dtype: int64
```

In [ ]:
```
df['ORGANIZATION_TYPE'].value_counts()
```

Out[ ]:
```
Business Entity Type 3    67992
XNA                       55374
Self-employed             38412
Other                     16683
Medicine                  11193
Business Entity Type 2    10553
Government                10404
School                     8893
Trade: type 7              7831
Kindergarten               6880
Construction               6721
Business Entity Type 1     5984
Transport: type 4          5398
Trade: type 3              3492
Industry: type 9           3368
Industry: type 3           3278
Security                   3247
Housing                    2958
Industry: type 11          2704
Military                   2634
Bank                       2507
Agriculture                2454
Police                     2341
Transport: type 2          2204
Postal                     2157
Security Ministries        1974
Trade: type 2              1900
Restaurant                 1811
Services                   1575
University                 1327
Industry: type 7           1307
Transport: type 3          1187
Industry: type 1           1039
Hotel                       966
Electricity                 950
Industry: type 4            877
Trade: type 6               631
Industry: type 5            599
Insurance                   597
Telecom                     577
Emergency                   560
Industry: type 2            458
Advertising                 429
Realtor                     396
Culture                     379
Industry: type 12           369
Trade: type 1               348
Mobile                      317
Legal Services              305
Cleaning                    260
Transport: type 1           201
Industry: type 6            112
Industry: type 10           109
Religion                     85
Industry: type 13            67
Trade: type 4                64
Trade: type 5                49
Industry: type 8             24
Name: ORGANIZATION_TYPE, dtype: int64
```

In [ ]:
```
df = df.loc[df['ORGANIZATION_TYPE'] != 'XNA']
```

In [ ]:
```
df['ORGANIZATION_TYPE'].value_counts()
```

Out[ ]:
```
Business Entity Type 3    67992
Self-employed             38412
Other                     16683
Medicine                  11193
Business Entity Type 2    10553
Government                10404
School                     8893
Trade: type 7              7831
Kindergarten               6880
Construction               6721
Business Entity Type 1     5984
Transport: type 4          5398
Trade: type 3              3492
Industry: type 9           3368
Industry: type 3           3278
Security                   3247
Housing                    2958
Industry: type 11          2704
Military                   2634
Bank                       2507
Agriculture                2454
Police                     2341
Transport: type 2          2204
Postal                     2157
Security Ministries        1974
Trade: type 2              1900
Restaurant                 1811
Services                   1575
University                 1327
Industry: type 7           1307
Transport: type 3          1187
Industry: type 1           1039
Hotel                       966
Electricity                 950
Industry: type 4            877
Trade: type 6               631
Industry: type 5            599
Insurance                   597
Telecom                     577
Emergency                   560
Industry: type 2            458
Advertising                 429
Realtor                     396
Culture                     379
Industry: type 12           369
Trade: type 1               348
Mobile                      317
Legal Services              305
Cleaning                    260
Transport: type 1           201
Industry: type 6            112
Industry: type 10           109
Religion                     85
Industry: type 13            67
Trade: type 4                64
Trade: type 5                49
Industry: type 8             24
Name: ORGANIZATION_TYPE, dtype: int64
```

In [ ]:
```
df.dtypes.value_counts()
```

Out[ ]:
```
int64      41
float64    27
object     13
dtype: int64
```

In [ ]:
```
print(ask('how to filter the only intiger datatype columns not string datatype c
```

```
from a dataframe

This can be done using the select_dtypes function:

df.select_dtypes(include=['int'])
```

In [ ]:
```
df.select_dtypes(exclude=object)
```

Out[ ]:

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANN
0	100002	1	0	202500.0	406597.5	247
1	100003	0	0	270000.0	1293502.5	356
2	100004	0	0	67500.0	135000.0	67
3	100006	0	0	135000.0	312682.5	296
4	100007	0	0	121500.0	513000.0	218
...	...	...	...	...	...	
307504	456248	0	0	153000.0	331920.0	160
307506	456251	0	0	157500.0	254700.0	275
307508	456253	0	0	153000.0	677664.0	299
307509	456254	1	0	171000.0	370107.0	202
307510	456255	0	0	157500.0	675000.0	491

252137 rows × 68 columns

In [ ]:
```
df.select_dtypes(exclude=object).columns
```

Out[ ]:
```
Index(['SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
       'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'FLAG_EMP_PHONE',
       'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL',
       'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', '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', 'EXT_SOURCE_2',
       'EXT_SOURCE_3', 'YEARS_BEGINEXPLUATATION_AVG', 'FLOORSMAX_AVG',
       'YEARS_BEGINEXPLUATATION_MODE', 'FLOORSMAX_MODE',
       'YEARS_BEGINEXPLUATATION_MEDI', 'FLOORSMAX_MEDI', 'TOTALAREA_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'],
      dtype='object')
```

In [ ]:
```
df[df.select_dtypes(exclude=object).columns] = df[df.select_dtypes(exclude=objec
```

In [ ]:
```
df.select_dtypes(include=object)
```

Out[ ]:

	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	NAME_
0	Cash loans	M	N	Y	Una
1	Cash loans	F	N	N	
2	Revolving loans	M	Y	Y	Una
3	Cash loans	F	N	Y	Una
4	Cash loans	M	N	Y	Una
...	...	...	...	...	
307504	Cash loans	F	N	Y	Una
307506	Cash loans	M	N	N	Una
307508	Cash loans	F	N	Y	Una
307509	Cash loans	F	N	Y	Una
307510	Cash loans	F	N	N	Una

252137 rows × 13 columns

In [ ]:
```
df.select_dtypes(include=object).columns
```

Out[ ]:
```
Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE',
       'EMERGENCYSTATE_MODE'],
      dtype='object')
```

In [ ]:
```
df[df.select_dtypes(include=object).columns] = df[df.select_dtypes(include=objec
```

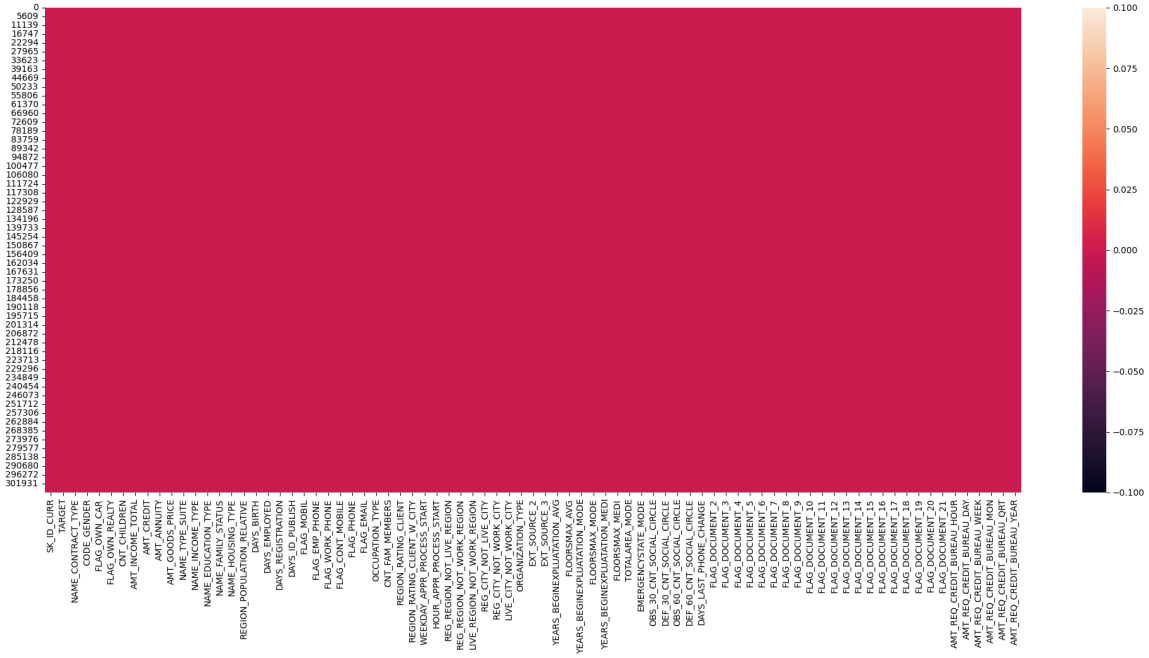
In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
SK_ID_CURR                    0
TARGET                        0
NAME_CONTRACT_TYPE            0
CODE_GENDER                   0
FLAG_OWN_CAR                  0
                             ..
AMT_REQ_CREDIT_BUREAU_DAY     0
AMT_REQ_CREDIT_BUREAU_WEEK    0
AMT_REQ_CREDIT_BUREAU_MON     0
AMT_REQ_CREDIT_BUREAU_QRT     0
AMT_REQ_CREDIT_BUREAU_YEAR    0
Length: 81, dtype: int64
```

In [ ]:
```
plt.figure(figsize=(25,10))
```

```
sns.heatmap(df.isnull())
```

Out[ ]: <Axes: >



- The chart is red so it means that there is no any null values in the datasets

Now, Finding the Outlier in Income and credit columns

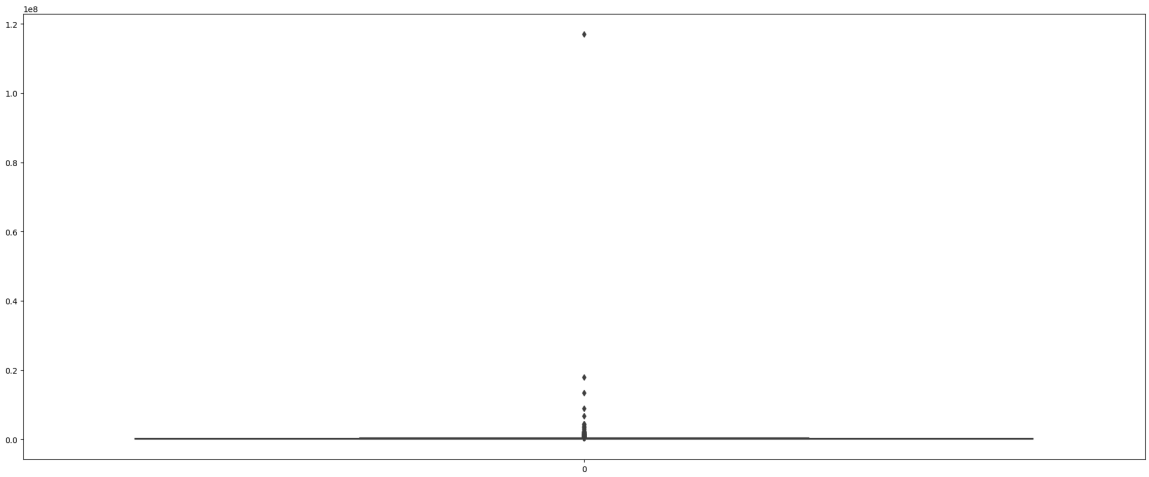
In [ ]:
```
df['AMT_INCOME_TOTAL']
```

Out[ ]:
```
0         202500.0
1         270000.0
2          67500.0
3         135000.0
4         121500.0
              ...
307504    153000.0
307506    157500.0
307508    153000.0
307509    171000.0
307510    157500.0
Name: AMT_INCOME_TOTAL, Length: 252137, dtype: float64
```

In [ ]:
```
plt.figure(figsize=(25,10))

sns.boxplot(df['AMT_INCOME_TOTAL'])
```

Out[ ]: <Axes: >



In [ ]:
```
q1, q2, q3 = np.percentile(df['AMT_INCOME_TOTAL'],[25,50,75])
```

In [ ]:
```
q1
```

Out[ ]:
```
112500.0
```

In [ ]:
```
q2
```

Out[ ]:
```
157500.0
```

In [ ]:
```
q3
```

Out[ ]:
```
211500.0
```

In [ ]:
```
iqr = q3- q1
```

In [ ]:
```
print(ask('how to find the outlier with iqr'))
```

```
rule

To find the outlier using the IQR rule, first find the interquartile range (IQ
R). The IQR is the difference between the first quartile (Q1) and the third qua
rtile (Q3).

Next, find the lower and upper bounds. The lower bound is Q1 - 1.5IQR, and the
upper bound is Q3 + 1.5IQR. Any values that are less than the lower bound or gr
eater than
```

In [ ]:
```
print(ask('can yo write the code for lower fence how to write for finding teh ou
```

```
There is no definitive answer to this question, as the code required to find ou
tliers in a dataset will vary depending on the specific dataset in question. Ho
wever, some methods for finding outliers in data include computing the mean and
standard deviation of the data, and then identifying values that are a certain
number of standard deviations away from the mean (known as a "Z-score"). Anothe
r method is to compute the median and interquartile range of the data, and then
identify values that are
```

In [ ]:
```
lower_x = q1 - 1.5 * iqr
upper_x = q3 + 1.5 * iqr
```

In [ ]:
```
lower_x
```

Out[ ]: -36000.0

In [ ]:
```
upper_x
```

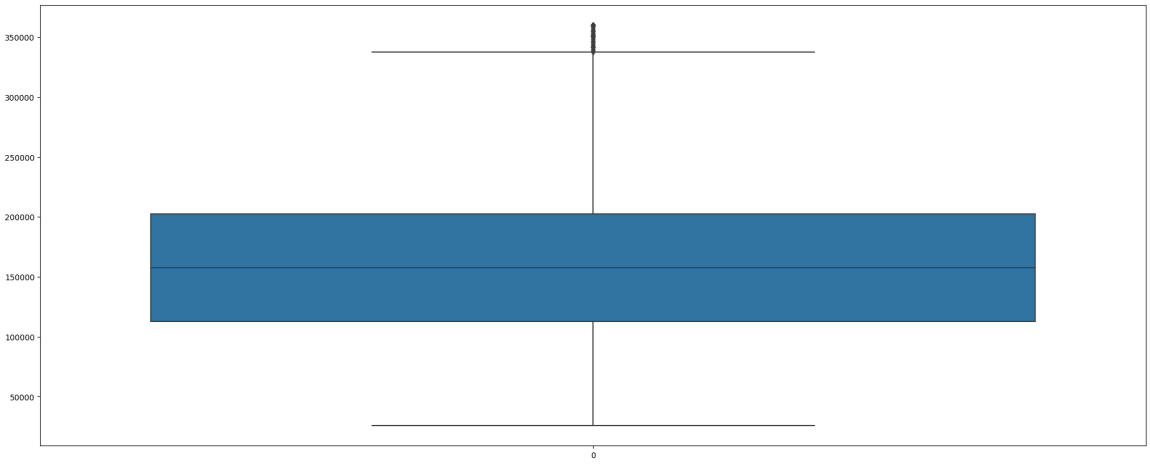
Out[ ]: 360000.0

In [ ]:
```
df = df.loc[df['AMT_INCOME_TOTAL']<= upper_x]
```

In [ ]:
```
plt.figure(figsize=(25,10))

sns.boxplot(df['AMT_INCOME_TOTAL'])
```

Out[ ]: <Axes: >



- I have delete the outlier from the income columns now the time for credit columns

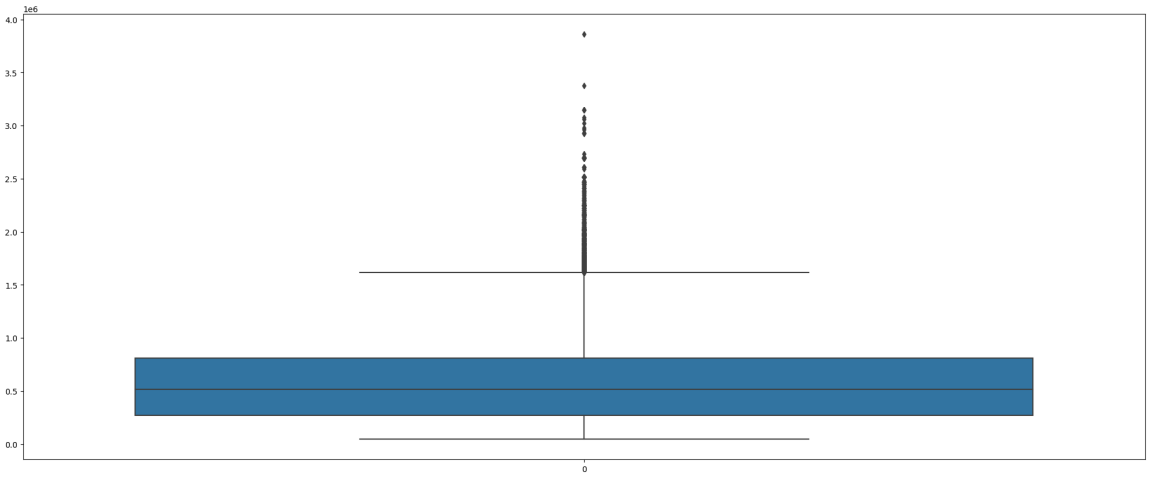
In [ ]:
```
df['AMT_CREDIT']
```

Out[ ]:
```
0          406597.5
1         1293502.5
2          135000.0
3          312682.5
4          513000.0
             ...
307504     331920.0
307506     254700.0
307508     677664.0
307509     370107.0
307510     675000.0
Name: AMT_CREDIT, Length: 243385, dtype: float64
```

In [ ]:
```
plt.figure(figsize=(25,10))

sns.boxplot(df['AMT_CREDIT'])
```

Out[ ]: <Axes: >



In [ ]:
```
df['AMT_CREDIT'].max()
```

Out[ ]: 3860019.0

In [ ]:
```
df['AMT_CREDIT'].min()
```

Out[ ]: 45000.0

In [ ]:
```
q1,q2,q3 = np.percentile(df['AMT_CREDIT'],[25,50,75])
q1
```

Out[ ]: 271066.5

In [ ]:
```
q2
```

Out[ ]: 513040.5

In [ ]:
```
q3
```

Out[ ]: 808650.0

In [ ]:
```
iqr = q3 -q1
iqr
```

Out[ ]: 537583.5

In [ ]:
```
lower_x1 = q1 - 1.5 * iqr
upper_x1 = q3 + 1.5 * iqr
```

In [ ]:
```
lower_x1
```

Out[ ]: -535308.75

In [ ]:
```
upper_x1
```

Out[ ]: 1615025.25

In [ ]:
```
df = df.loc[df['AMT_CREDIT'] <= upper_x1]
```

In [ ]:
```
df
```

Out[ ]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG
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	
...	...	...	...	...	...	
307504	456248	0	Cash loans	F	N	
307506	456251	0	Cash loans	M	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

238861 rows × 81 columns

In [ ]: