Level 0:

In [1]:
```
import matplotlib.pyplot as plt                 # to visualize
from tabulate import tabulate                   # to print the table
import matplotlib as mat                        # to visualize
import seaborn as sns                           # to visualize
import pandas as pd                             # for data reading
import numpy as np                              # for numerical computation
```

In [2]:
```
df = pd.read_csv("D:\Data Science\Course 6\DS1_C6_S4_Credit_Data_Hackathon.c
df.head()
```

Out[2]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	GENDER	Car	House	CNT_CHILDREN	AMT_
0	100002	1	Cash loans	M	N	Y	0	
1	100003	0	Cash loans	F	N	N	0	
2	100004	0	Revolving loans	M	Y	Y	0	
3	100006	0	Cash loans	F	N	Y	0	
4	100007	0	Cash loans	M	N	Y	0	

5 rows × 24 columns

In [3]:
```
print(df.columns)
```

```
Index(['SK_ID_CURR', 'TARGET', 'NAME_CONTRACT_TYPE', 'GENDER', 'Car', 'House',
       'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_GOODS_PRICE',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'DAYS_EMPLOYED', 'MOBILE', 'WORK_PHONE',
       'HOME_PHONE', 'MOBILE_REACHABLE', 'FLAG_EMAIL', 'OCCUPATION_TYPE',
       'CNT_FAM_MEMBERS', 'APPLICATION_DAY', 'TOTAL_DOC_SUBMITTED'],
      dtype='object')
```

In [4]:
```
print(df.shape)
```

```
(100000, 24)
```

In [5]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 24 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   SK_ID_CURR           100000 non-null  int64  
 1   TARGET               100000 non-null  int64  
 2   NAME_CONTRACT_TYPE   100000 non-null  object 
 3   GENDER               100000 non-null  object 
 4   Car                  100000 non-null  object 
 5   House                100000 non-null  object 
 6   CNT_CHILDREN         100000 non-null  int64  
 7   AMT_INCOME_TOTAL     100000 non-null  float64
 8   AMT_CREDIT           100000 non-null  float64
 9   AMT_GOODS_PRICE      99919 non-null   float64
 10  NAME_TYPE_SUITE      99595 non-null   object 
 11  NAME_INCOME_TYPE     100000 non-null  object 
 12  NAME_EDUCATION_TYPE  100000 non-null  object 
 13  NAME_FAMILY_STATUS   100000 non-null  object 
 14  DAYS_EMPLOYED        100000 non-null  int64  
 15  MOBILE               100000 non-null  int64  
 16  WORK_PHONE           100000 non-null  int64  
 17  HOME_PHONE           100000 non-null  int64  
 18  MOBILE_REACHABLE     100000 non-null  int64  
 19  FLAG_EMAIL           100000 non-null  int64  
 20  OCCUPATION_TYPE      68776 non-null   object 
 21  CNT_FAM_MEMBERS      99999 non-null   float64
 22  APPLICATION_DAY      100000 non-null  object 
 23  TOTAL_DOC_SUBMITTED  100000 non-null  int64  
dtypes: float64(4), int64(10), object(10)
memory usage: 18.3+ MB
```

In [6]:
```
1 df.isnull().sum()
```

Out[6]:
```
SK_ID_CURR                0
TARGET                    0
NAME_CONTRACT_TYPE        0
GENDER                    0
Car                       0
House                     0
CNT_CHILDREN              0
AMT_INCOME_TOTAL          0
AMT_CREDIT                0
AMT_GOODS_PRICE          81
NAME_TYPE_SUITE         405
NAME_INCOME_TYPE          0
NAME_EDUCATION_TYPE       0
NAME_FAMILY_STATUS        0
DAYS_EMPLOYED             0
MOBILE                    0
WORK_PHONE                0
HOME_PHONE                0
MOBILE_REACHABLE          0
FLAG_EMAIL                0
OCCUPATION_TYPE       31224
CNT_FAM_MEMBERS           1
APPLICATION_DAY           0
TOTAL_DOC_SUBMITTED       0
dtype: int64
```

In [7]:
```
def seprate_data_types(df):
    categorical = []
    continuous = []
    for column in df.columns:
        if df[column].nunique() < 100:

            categorical.append(column)
        else:
            continuous.append(column)

    return categorical, continuous


categorical, continuous = seprate_data_types(df)

# # Tabulate is a package used to print the list, dict or any data sets in a
from tabulate import tabulate
table = [categorical, continuous]
print(tabulate({"Categorical":categorical,
                "continuous": continuous}, headers = ["categorical", "contin
```

```
categorical          continuous
-------------------  ----------------
TARGET               SK_ID_CURR
NAME_CONTRACT_TYPE   AMT_INCOME_TOTAL
GENDER               AMT_CREDIT
Car                  AMT_GOODS_PRICE
House                DAYS_EMPLOYED
CNT_CHILDREN
NAME_TYPE_SUITE
NAME_INCOME_TYPE
NAME_EDUCATION_TYPE
NAME_FAMILY_STATUS
MOBILE
WORK_PHONE
HOME_PHONE
MOBILE_REACHABLE
FLAG_EMAIL
OCCUPATION_TYPE
CNT_FAM_MEMBERS
APPLICATION_DAY
TOTAL_DOC_SUBMITTED
```

In [8]:
```
def info_of_cat(col):
    print(f"Unique values in {col} are: {df[col].unique()}")
    print(f"Mode of {col} is {df[col].mode()[0]}")
    print(f"Number of missing values in {col} is {df[col].isnull().sum()}")
```

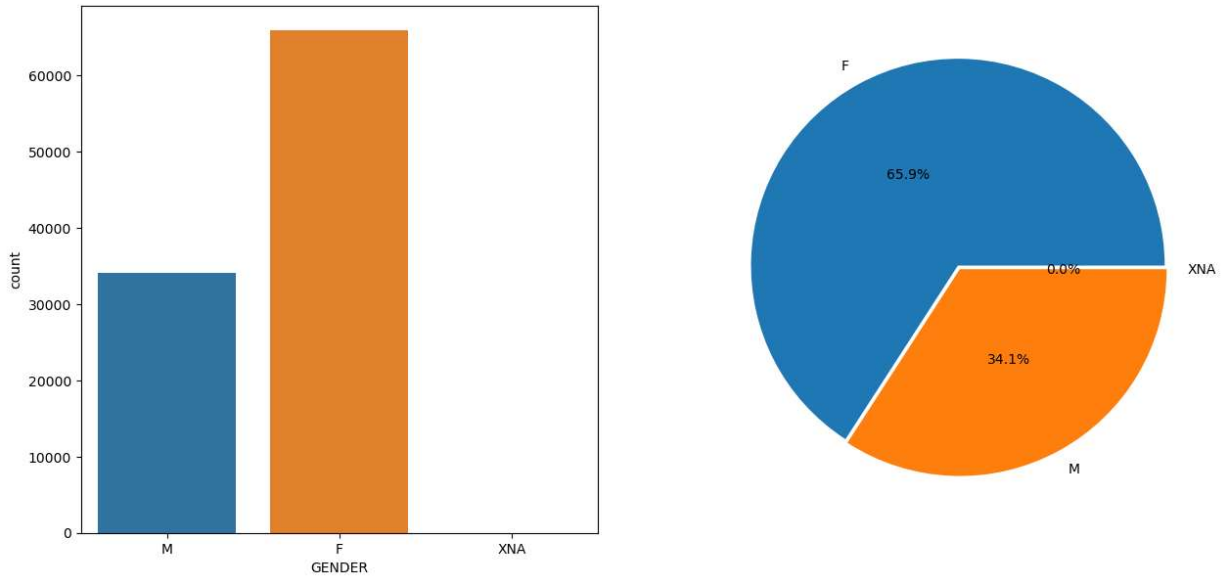
Level 1

Question 1 :

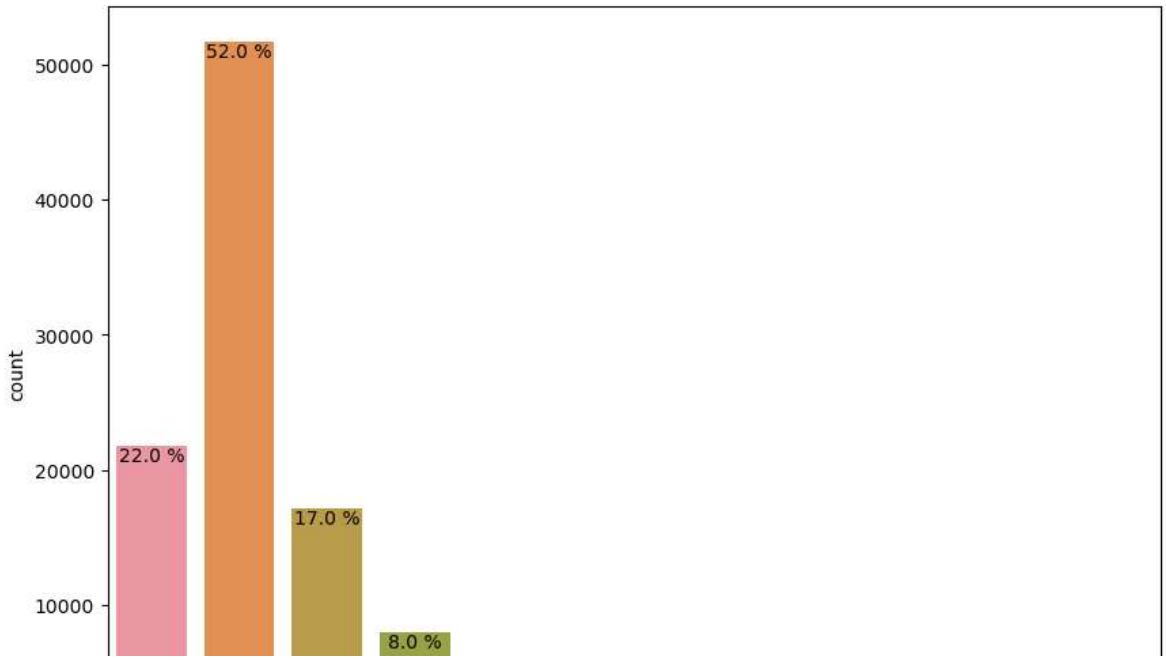
In [9]:
```
1 info_of_cat("GENDER")
```

```
Unique values in GENDER are: ['M' 'F' 'XNA']
Mode of GENDER is F
Number of missing values in GENDER is 0
```

In [10]:
```
1 fig, ax = plt.subplots(1, 2, figsize = (15, 7))
2 data = df["GENDER"].value_counts()
3 labels =data.keys()
4
5 sns.countplot(x = df["GENDER"], ax = ax[0])
6 plt.pie(x = data, autopct = "%.1f%%", explode = [0.02, 0.0, 0], labels = lab
7
8 plt.show()
9
```



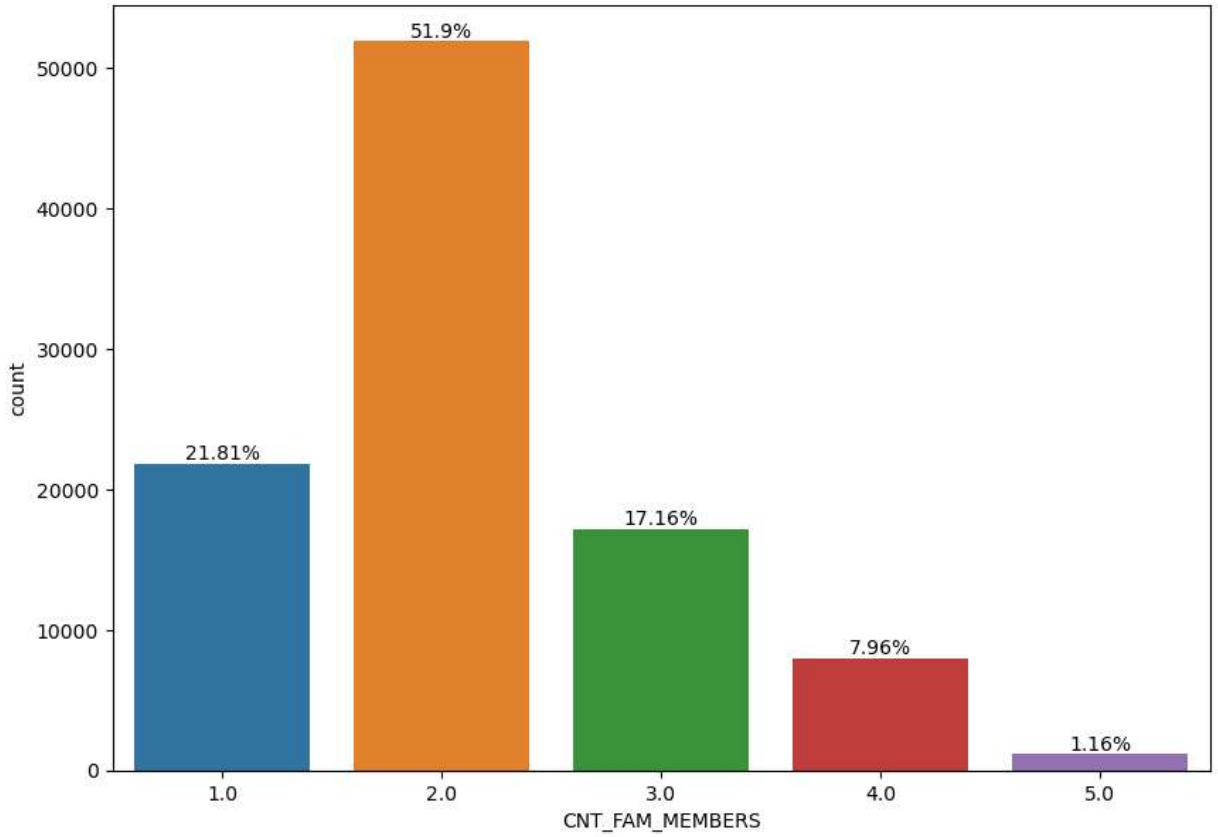
In [11]:
```
fig, ax = plt.subplots(figsize= (10, 7))
ax = sns.countplot(x = df["CNT_FAM_MEMBERS"])
ax.set_xticklabels(ax.get_xticklabels(), rotation = 40 , ha = "right")



count = len(df["CNT_FAM_MEMBERS"])
for bar in ax.patches:
    percentage = f"{round(bar.get_height()/count, 2)*100} %"

    x = bar.get_x() + bar.get_width() /2
    y = bar.get_height()
    ax.annotate(percentage, (x, y), ha = "center", va = "top")

```



In [12]:
```
mean = int(df.CNT_FAM_MEMBERS.mean())
# accessing the data higher than 5
x = df[df["CNT_FAM_MEMBERS"] > 5].index
for index in x:
    df.loc[index, "CNT_FAM_MEMBERS"] = mean

```

In [13]:

```
fig, ax = plt.subplots(figsize= (10, 7))
ax = sns.countplot(x = df["CNT_FAM_MEMBERS"])
for bar in ax.patches:
    percentage = f"{round(bar.get_height()*100 /len(df), 2)}%"

    x = bar.get_x() + bar.get_width() /2
    y = bar.get_height()
    ax.annotate(percentage, (x, y), va = "bottom", ha ="center" )

plt.show()
```



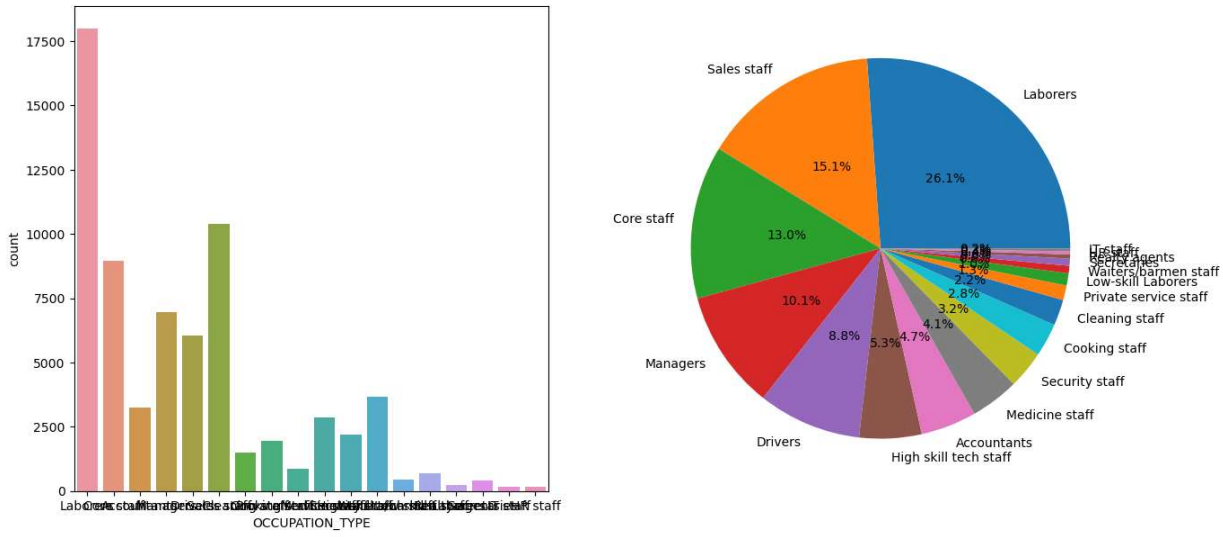
In [14]:
```
1 info_of_cat("OCCUPATION_TYPE")
```

```
Unique values in OCCUPATION_TYPE are: ['Laborers' 'Core staff' 'Accountants' 'M
anagers' nan 'Drivers'
 'Sales staff' 'Cleaning staff' 'Cooking staff' 'Private service staff'
 'Medicine staff' 'Security staff' 'High skill tech staff'
 'Waiters/barmen staff' 'Low-skill Laborers' 'Realty agents' 'Secretaries'
 'IT staff' 'HR staff']
Mode of OCCUPATION_TYPE is Laborers
Number of missing values in OCCUPATION_TYPE is 31224
```

In [15]:
```
1 TYPE_mode = df.OCCUPATION_TYPE.mode()
2 print("OCCUPATION_TYPE")
3 df["OCCUPATION_TYPE"].fillna(TYPE_mode, inplace = True)
4 OCCUPATION= df["OCCUPATION_TYPE"]
```

```
OCCUPATION_TYPE
```

In [16]:
```
1 fig, ax = plt.subplots(1, 2, figsize = (15, 7))
2 data = OCCUPATION.value_counts()
3 labels =data.keys()
4
5 sns.countplot(x = OCCUPATION, ax = ax[0])
6 plt.pie(x = data, autopct = "%.1f%%", labels = labels, pctdistance = 0.5)
7
8 plt.show()
9
```



Continuous

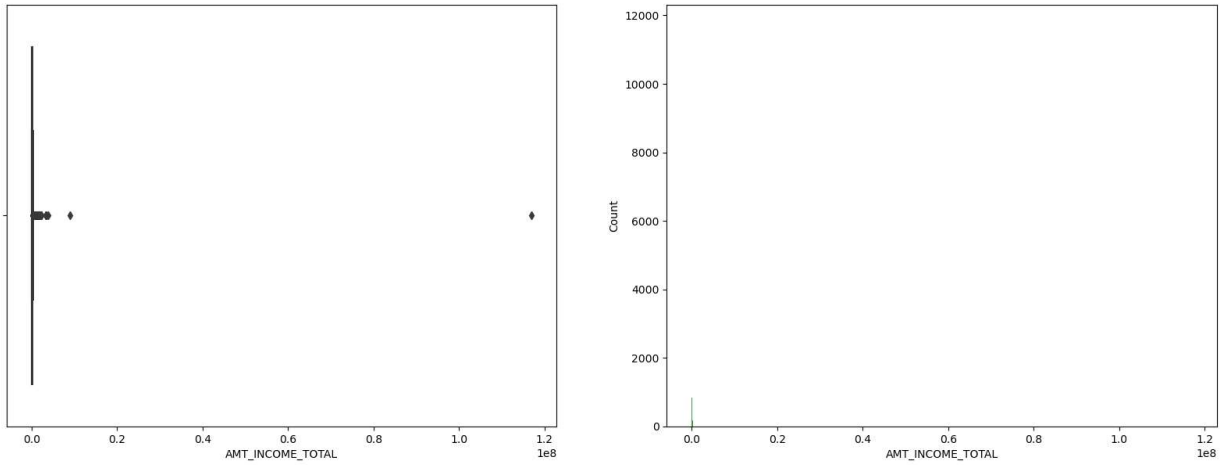
In [17]:
```
1 #AMT_INCOME_TOTAL, AMT_CREDIT, AMT_GOODS_PRICE
```

In [18]:
```
def info_of_numerical(col):
    print(f"The mean of the {col} is {df[col].mean()}")
    print(f"The median of the {col} is {df[col].median()}")
    print(f"The mode of the {col} is {df[col].mode()[0]}")
    print(f"The standard deviation of the {col} is {df[col].std()}")
    print(f"Number of missing values in the {col} is {df[col].isnull().sum()
```

In [19]:
```
info_of_numerical("AMT_INCOME_TOTAL")
```

```
The mean of the AMT_INCOME_TOTAL is 169426.07027325002
The median of the AMT_INCOME_TOTAL is 144000.0
The mode of the AMT_INCOME_TOTAL is 135000.0
The standard deviation of the AMT_INCOME_TOTAL is 383500.74427718896
Number of missing values in the AMT_INCOME_TOTAL is 0
```

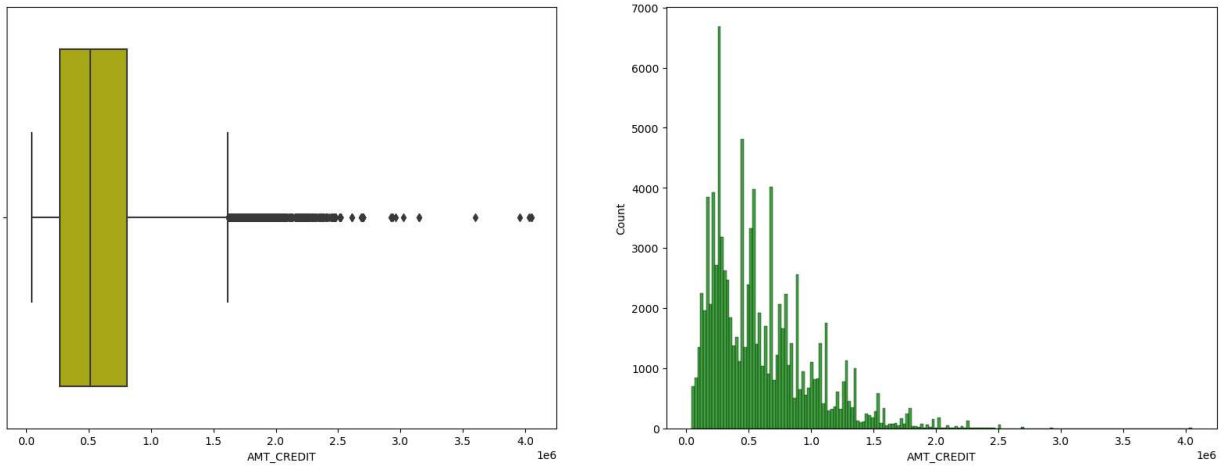
In [20]:
```
fig, ax = plt.subplots(1, 2, figsize= (20, 7))
sns.histplot(df["AMT_INCOME_TOTAL"], ax = ax[1], color= "g")     # xaxis: it
sns.boxplot(x = df['AMT_INCOME_TOTAL'], ax = ax[0], color = "y") # color = y
plt.show()
```



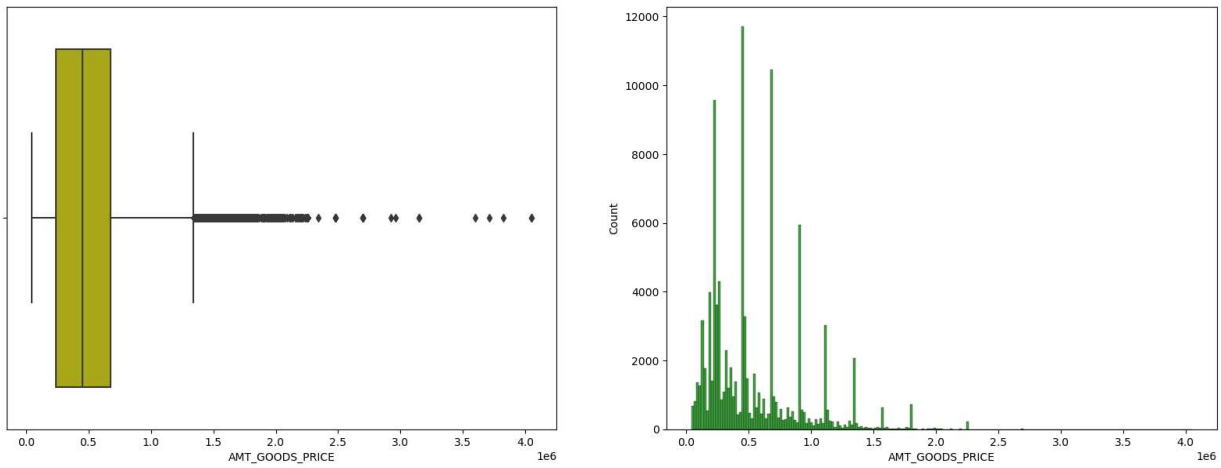
In [21]:
```
info_of_numerical("AMT_CREDIT")
```

```
The mean of the AMT_CREDIT is 599003.4465
The median of the AMT_CREDIT is 513040.5
The mode of the AMT_CREDIT is 450000.0
The standard deviation of the AMT_CREDIT is 402051.9591213264
Number of missing values in the AMT_CREDIT is 0
```

In [22]:
```
fig, ax = plt.subplots(1, 2, figsize= (20, 7))
sns.histplot(df["AMT_CREDIT"], ax = ax[1], color= "g")     # xaxis: its a do
sns.boxplot(x = df['AMT_CREDIT'], ax = ax[0], color = "y") # color = y:  yel
plt.show()
```



In [23]:
```
fig, ax = plt.subplots(1, 2, figsize= (20, 7))
sns.histplot(df["AMT_GOODS_PRICE"], ax = ax[1], color= "g")     # xaxis: its
sns.boxplot(x = df['AMT_GOODS_PRICE'], ax = ax[0], color = "y") # color = y:
plt.show()
```



Level 2

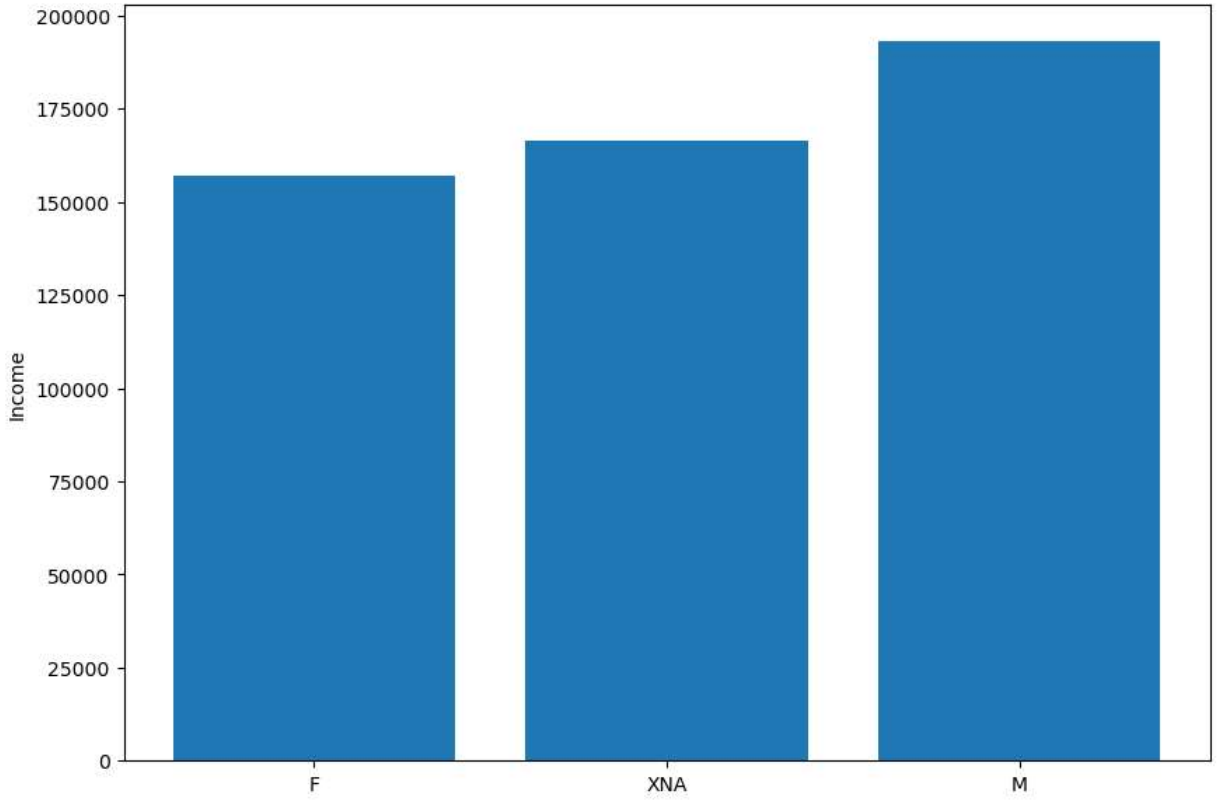
In [24]:
```
print(tabulate({"Categorical":categorical,
                "continuous": continuous}, headers = ["categorical", "contin
```

```
categorical          continuous
-------------------  ---------------
TARGET               SK_ID_CURR
NAME_CONTRACT_TYPE   AMT_INCOME_TOTAL
GENDER               AMT_CREDIT
Car                  AMT_GOODS_PRICE
House                DAYS_EMPLOYED
CNT_CHILDREN
NAME_TYPE_SUITE
NAME_INCOME_TYPE
NAME_EDUCATION_TYPE
NAME_FAMILY_STATUS
MOBILE
WORK_PHONE
HOME_PHONE
MOBILE_REACHABLE
FLAG_EMAIL
OCCUPATION_TYPE
CNT_FAM_MEMBERS
APPLICATION_DAY
TOTAL_DOC_SUBMITTED
```

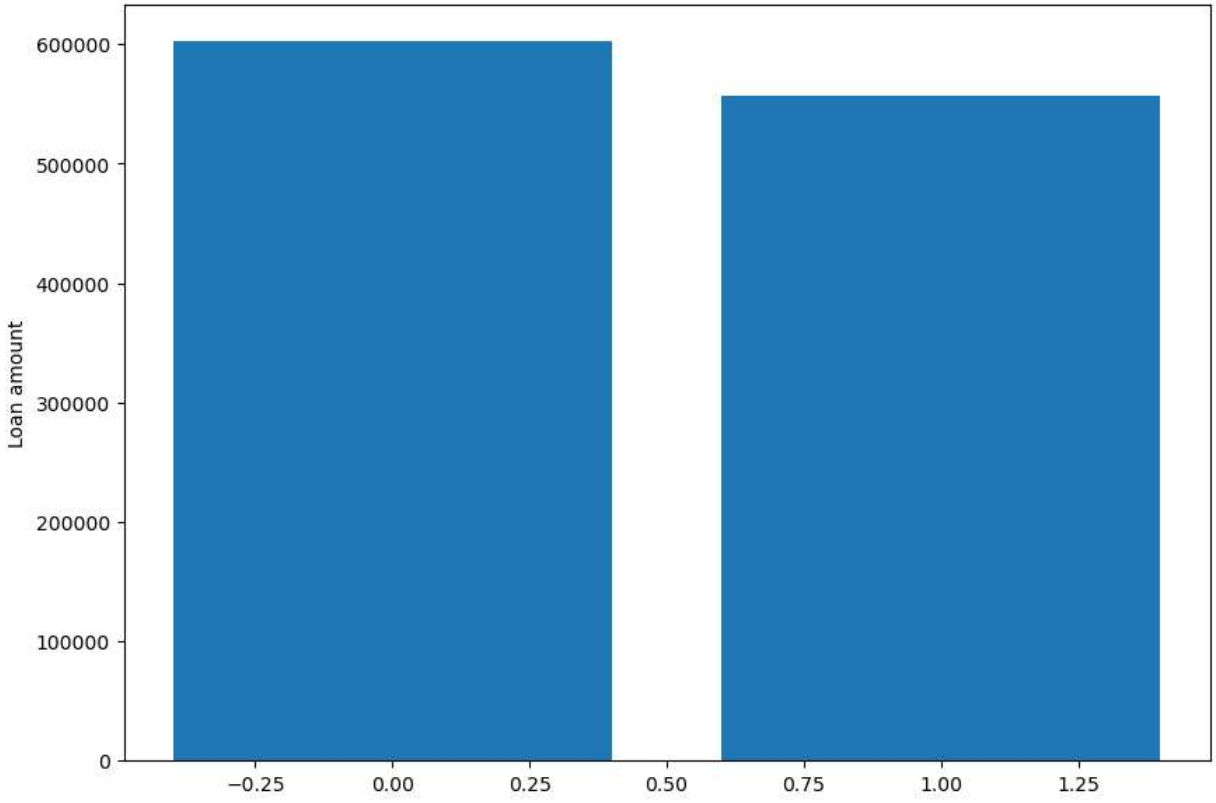
In [25]:
```
## Interpretation:
### From the above analysis we can see the female customers have the highest
```

In [29]:

```
fig, ax = plt.subplots(figsize = (10, 7))
city_col=df.groupby("GENDER").mean()["AMT_INCOME_TOTAL"].sort_values()
plt.bar(city_col.index,city_col)
plt.ylabel("Income")
plt.show()
```



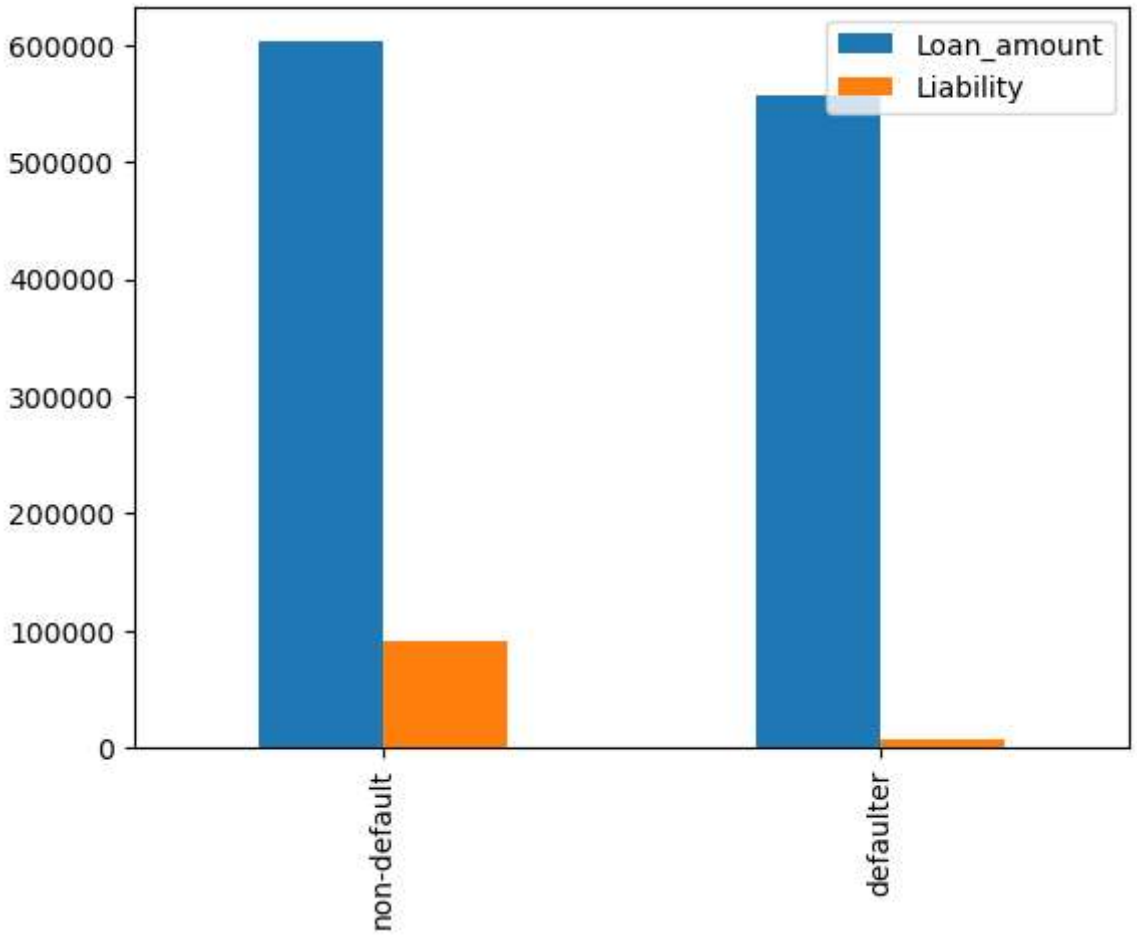
In [30]:
```
fig, ax = plt.subplots(figsize = (10, 7))
city_col=df.groupby("TARGET").mean()["AMT_CREDIT"].sort_values()
plt.bar(city_col.index,city_col)
plt.ylabel("Loan amount")
plt.show()
```



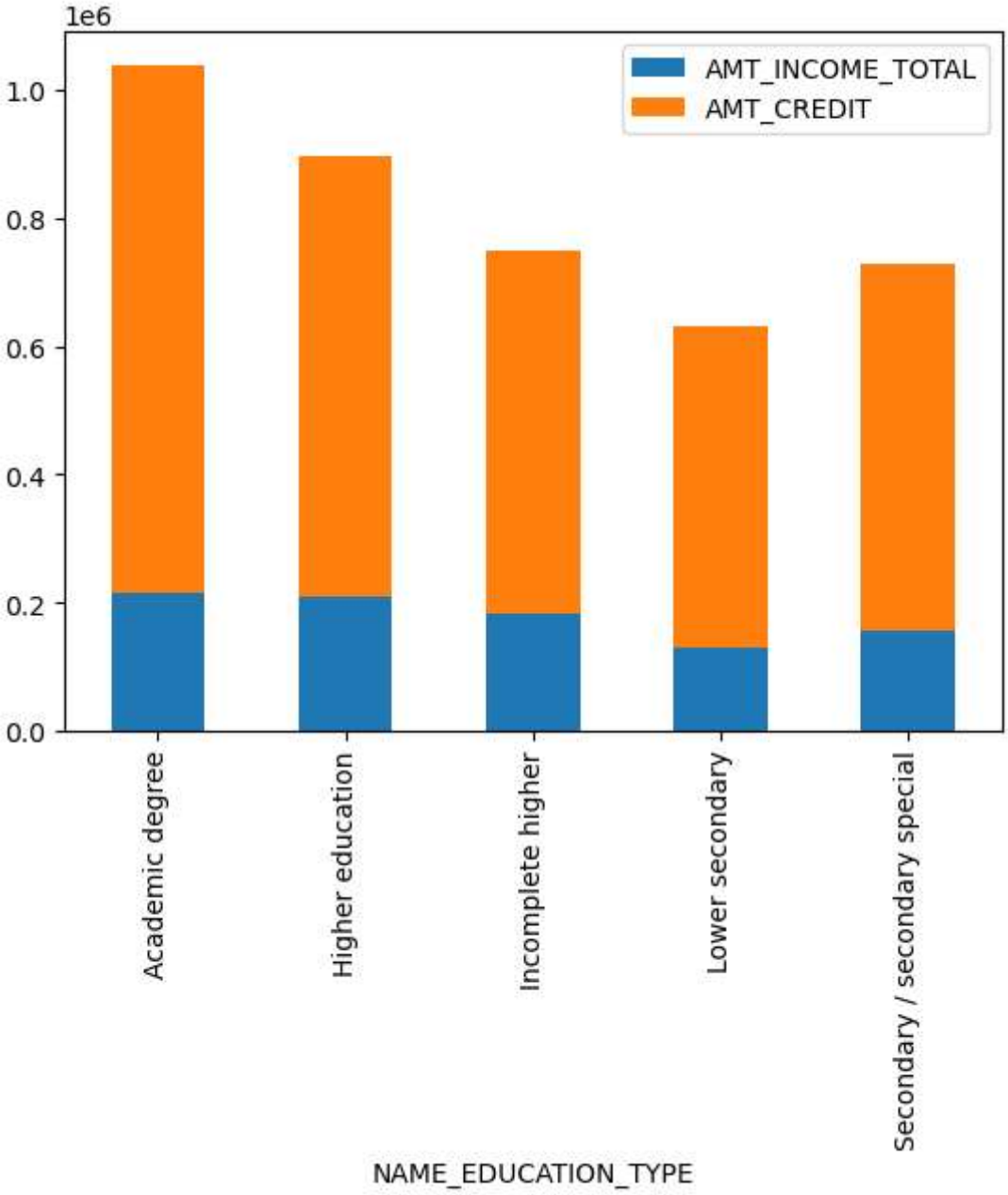

```
Level 3 - analysis
One could consider analyzing all the above columns for the customers who have left and having 2 or 3 dependents.
However it could be a meaningless visualization, hence it is better to consult the domain expert to choose the
appropriate columns for further analysis.
```

In [31]:
```
avg_loan=df.groupby("TARGET").mean()["AMT_CREDIT"].values
lia_c=df.groupby("TARGET")["House"].count().values
pd.DataFrame([avg_loan,lia_c],index=["Loan_amount","Liability"],columns=["no
```

Out[31]: <AxesSubplot:>



In [32]:
```
m=df.groupby("NAME_EDUCATION_TYPE").mean().loc[:,["AMT_INCOME_TOTAL","AMT_CR
m.plot(kind="bar",stacked=True)
plt.show()
```



In [33]:

```
m=df.groupby("OCCUPATION_TYPE").mean().loc[:,["AMT_INCOME_TOTAL","AMT_CREDIT

m.plot(kind="barh",stacked=True)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```