Importing necessary modules

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate
```

In [2]:

```python
credit=pd.read_csv(r"K:\Desktop\NIIT\tables\DS1_C6_S4_Credit_Data_Hackathon.csv")
credit
```

Out[2]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	GENDER	Car	House	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_GOODS_PRICE	...	DAYS_
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	351000.0	...	
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	1129500.0	...	
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	135000.0	...	
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	297000.0	...	
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	513000.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
99995	216086	0	Cash loans	F	N	Y	1	157500.0	755190.0	675000.0	...	
99996	216087	0	Cash loans	F	N	Y	1	225000.0	284400.0	225000.0	...	
99997	216088	0	Cash loans	F	Y	Y	0	135000.0	1262583.0	1102500.0	...	
99998	216089	0	Cash loans	F	Y	N	0	135000.0	225000.0	225000.0	...	
99999	216090	0	Revolving loans	M	Y	Y	0	202500.0	337500.0	337500.0	...	

100000 rows × 24 columns

Level 0 : Data Exploration

1.Visually inspect the first few and last few rows of the data

In [3]:

```python
credit.head()
```

Out[3]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	GENDER	Car	House	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_GOODS_PRICE	...	DAYS_EMP
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	351000.0	...	
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	1129500.0	...	
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	135000.0	...	
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	297000.0	...	
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	513000.0	...	

5 rows × 24 columns

In [4]:

```python
credit.tail()
```

Out[4]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	GENDER	Car	House	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_GOODS_PRICE	...	DAYS_
99995	216086	0	Cash loans	F	N	Y	1	157500.0	755190.0	675000.0	...	
99996	216087	0	Cash loans	F	N	Y	1	225000.0	284400.0	225000.0	...	
99997	216088	0	Cash loans	F	Y	Y	0	135000.0	1262583.0	1102500.0	...	
99998	216089	0	Cash loans	F	Y	N	0	135000.0	225000.0	225000.0	...	
99999	216090	0	Revolving loans	M	Y	Y	0	202500.0	337500.0	337500.0	...	

5 rows × 24 columns

2.Check the shape of the data frame

In [5]:

```python
print("Number of rows and columns = ",credit.shape)
```

```
Number of rows and columns =  (100000, 24)
```

3.Check the count of null values in each column

In [6]:
```
credit["NAME_EDUCATION_TYPE"].value_counts()
```

Out[6]:
```
Secondary / secondary special    71068
Higher education                 24399
Incomplete higher                 3270
Lower secondary                   1214
Academic degree                     49
Name: NAME_EDUCATION_TYPE, dtype: int64
```

In [7]:
```
print(credit.isnull().sum())
print()
```

```
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
```

4.Inspect all the column names and cross check with the data dictionary

In [8]:
```
credit.columns
```

Out[8]:
```
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')
```

5.Check the information of the data frame using the info() function

In [9]:
```
credit.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
```

LEVEL 1 Analysis

Identify if the type data in each column is categorical or numerical?

1. Separate out the categorical columns from the numerical types

These are the kind of analyses that can be performed on categorical data

1. Check if it is Nominal or Ordinal
2. Check how many categories are present
3. Check the Mode
4. Check for Missing values
5. Think about how the missing values could be treated
6. Think about the kind of graph/chart that can be plotted using this data

Note: We are analyzing only one column at a time (Univariate Analysis).

In [10]:

```python
def seperator(df):
    categorical=[]
    numerical=[]
    for col in df.columns:
        if(df[col].nunique()<70):
            categorical.append(col)
        else:
            numerical.append(col)
    return categorical,numerical

def bar_percentage(ax, count: "number of rows in data "):
    for bar in ax.patches:
        percentage = f"{round((bar.get_height() / count) *100, 2)}%"

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

def cat_level1(df,col):
        fig,ax=plt.subplots(1,2,figsize=(18,6))
        print("Number of Unique values present = ",df[col].nunique())
        print("NA values = ",df[col].isnull().sum())
        print("Mode = ",df[col].mode()[0])
        df[col].fillna(df[col].mode()[0],inplace=True)
        sns.countplot(x=df[col],ax=ax[0])
        ax[0]=bar_percentage(ax[0], len(df))
        percentage=df[col].value_counts()
        labels=df[col].value_counts().index
        ax[1].pie(percentage,labels = list(labels), autopct= "%0.2f%%")
        ax[1].set_title(col+" compostion")
        plt.show()

def num_level1(df,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()}")
    fig, ax = plt.subplots(1, 2, figsize= (10,5))
    sns.histplot(x = df[col], ax =ax[0], color = "blue")
    sns.boxplot(x = df[col], ax = ax[1], color = "purple",showmeans=True)
    plt.show()

def outlier_treatment(dataframe,columns):
    for item in columns:
        percentile25 = dataframe[item].quantile(0.25)
        percentile75 = dataframe[item].quantile(0.75)
        iqr=percentile75-percentile25
        upper_limit = percentile75 + 1.5 * iqr
        lower_limit = percentile25 - 1.5 * iqr
        dataframe[item] = np.where(dataframe[item] > upper_limit,upper_limit,
        np.where(dataframe[item] < lower_limit,lower_limit,dataframe[item]))
    return dataframe


```

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

In [11]:
```
for i in credit.columns:print(i)
```

```
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
```

In [12]:
```
categorical,numerical=seperator(credit)
print(tabulate({"Categorical":categorical,"continuous": numerical},headers = ["categorical", "numerical"]))
```

```
categorical          numerical
-------------------  ----------------
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
```

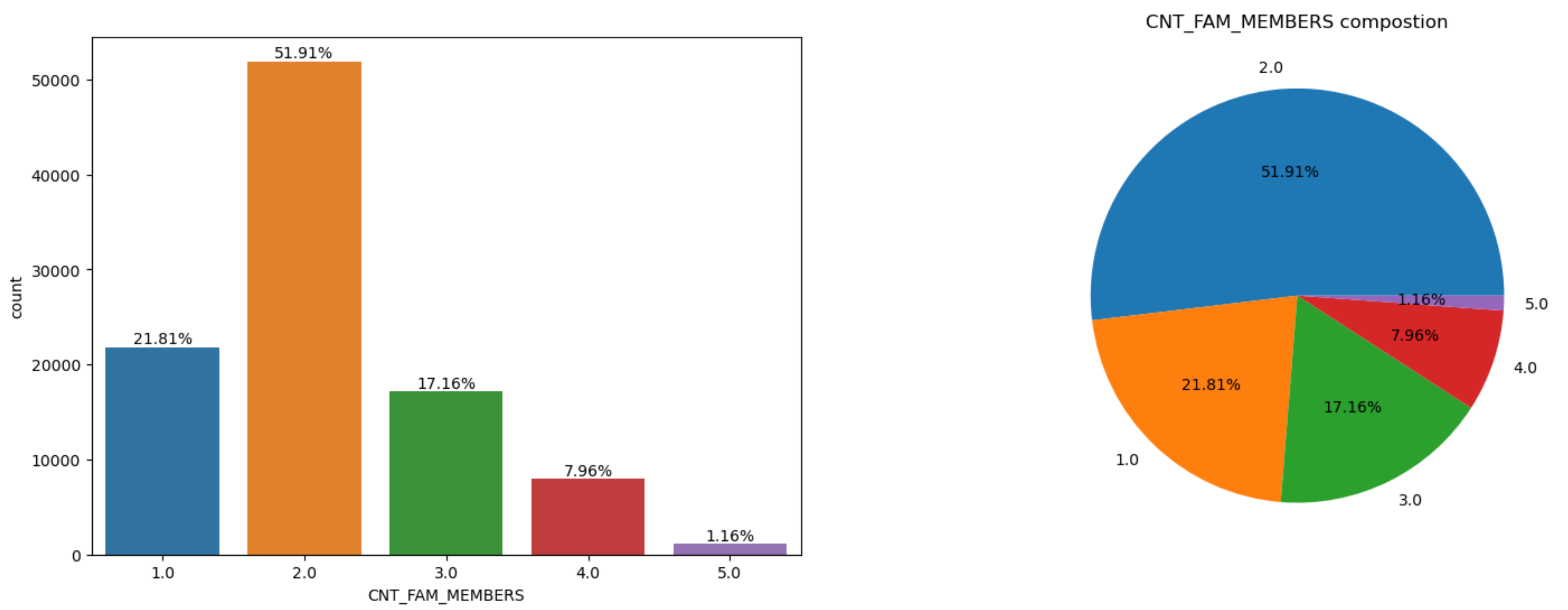
In [13]:
```
continous="AMT_INCOME_TOTAL,AMT_CREDIT,AMT_GOODS_PRICE".split(',')
categorical="CNT_FAM_MEMBERS,GENDER,TARGET,OCCUPATION_TYPE".split(',')
```

Plotting level 1 Analysis on Categorical

Count of family members for each customers

In [14]:
```python
mean = int(credit["CNT_FAM_MEMBERS"].mean())
x = credit[credit["CNT_FAM_MEMBERS"] > 5].index
for index in x:
    credit.loc[index, "CNT_FAM_MEMBERS"] = mean
x = credit[credit["CNT_FAM_MEMBERS"] > 5].index
cat_level1(credit,"CNT_FAM_MEMBERS")
```

```
Number of Unique values present =  5
NA values =  1
Mode =  2.0
```



Interpretation:

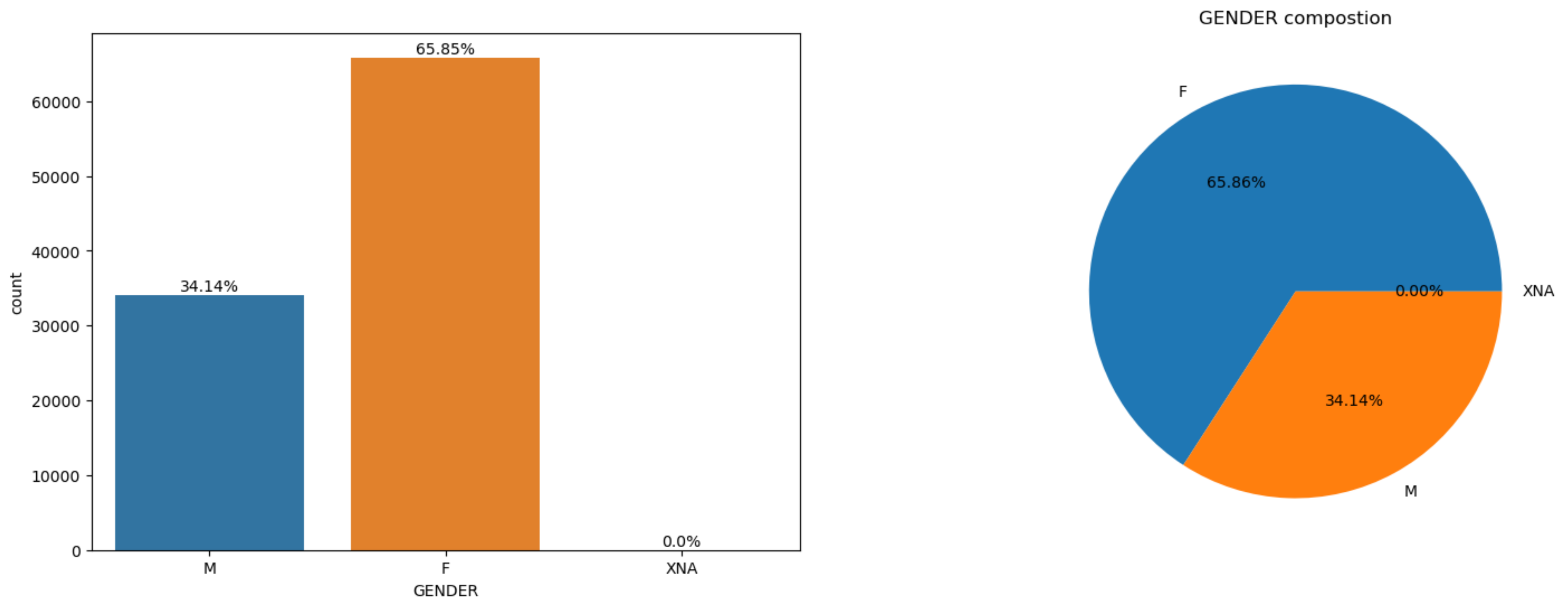
Majority of customers have only 2 or 1 family members which states thier recently married or single

Interpretation :

Majority of the homes are 2 bedroom and 3 bedroom contributing to more than 58% of all composition of homes

In [15]:
```python
cat_level1(credit,"GENDER")
```

```
Number of Unique values present =  3
NA values =  0
Mode =  F
```

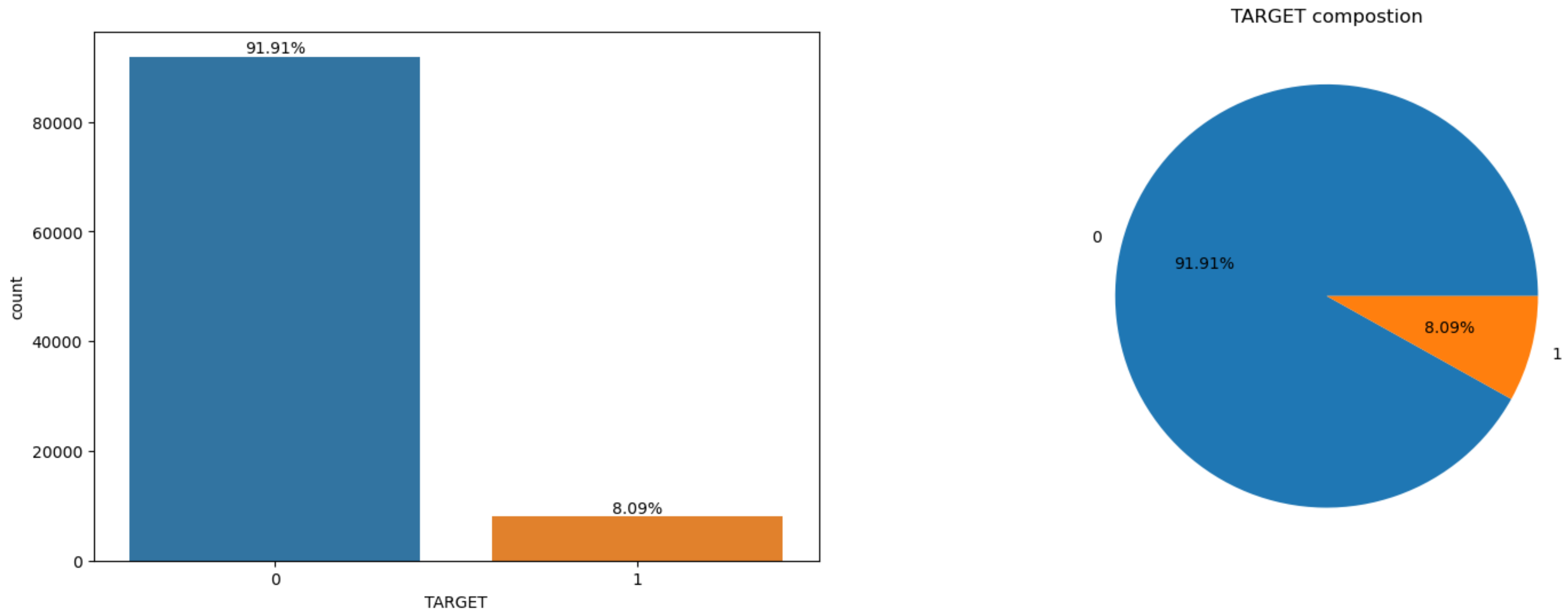


Interpretation :

Majority of customers are females and about 65% of them and 34% of males

In [16]:
```
cat_level1(credit,"TARGET")
```

```
Number of Unique values present =  2
NA values =  0
Mode =  0
```



Interpretation :

Most of the customers are being genuine in thier repayments and only 8% of customers

Level 1 Analysis for numerical data

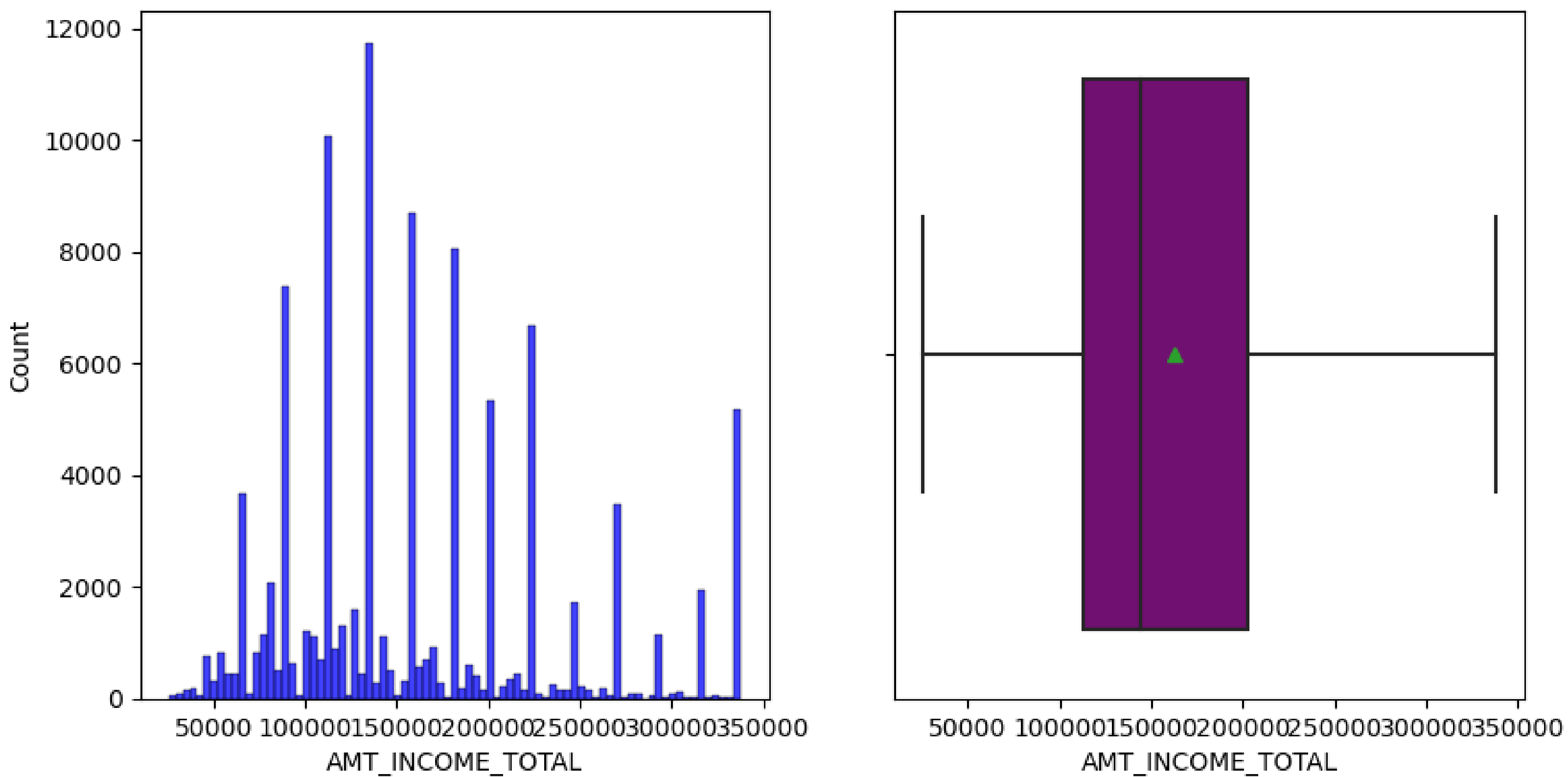
Outlier treatment for all the data in continous

In [17]:
```
credit=outlier_treatment(credit,continous)
```

The data has been cleaned of all possible outliers post outlier treatment which can be observed in below boxplots

In [18]:
```
num_level1(credit,continous[0])
```

```
The mean of the AMT_INCOME_TOTAL is 162551.20055625
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 73404.98120858667
Number of missing values in the AMT_INCOME_TOTAL is 0
```

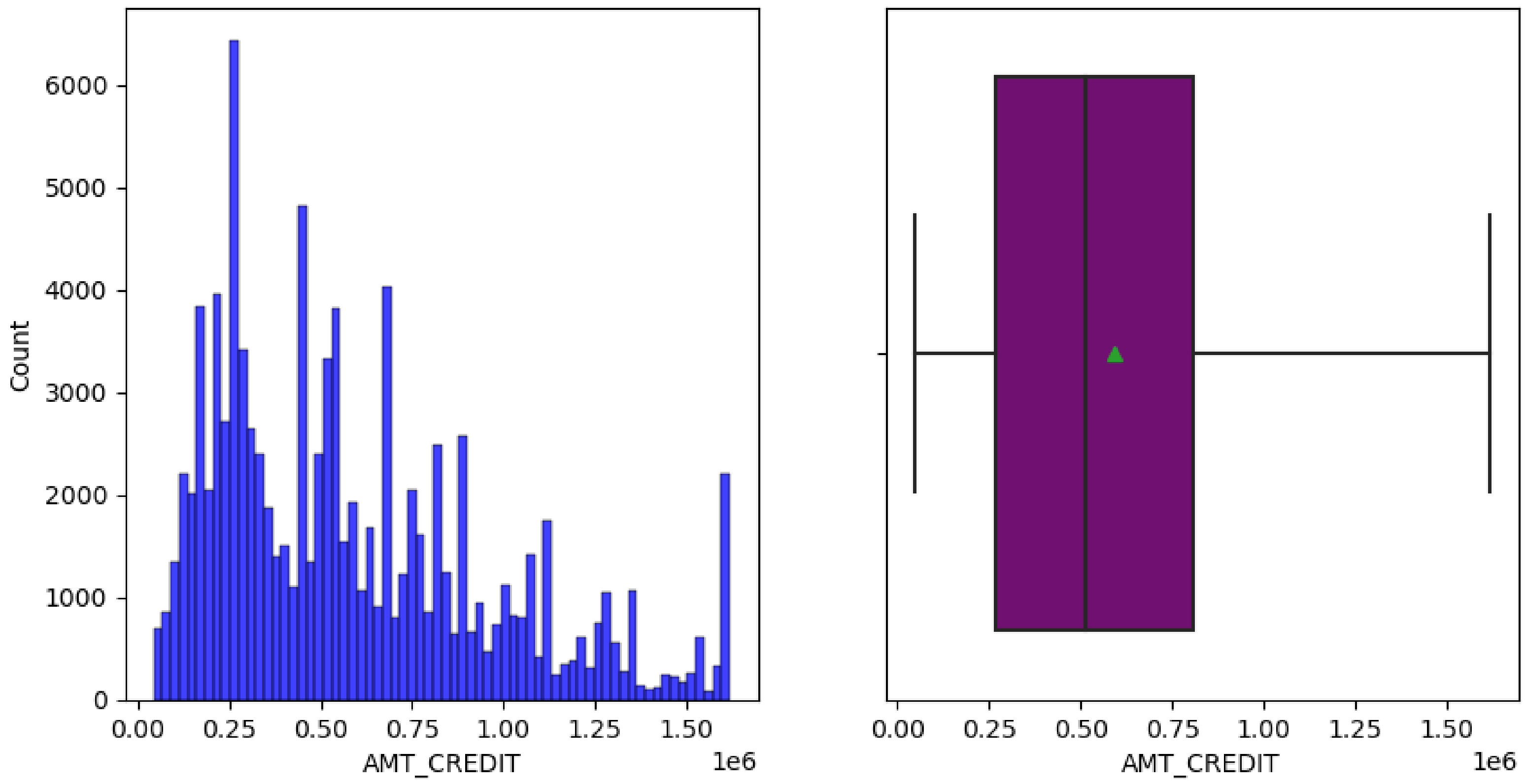


Interpretation:

We can see the boxplot looks clean of outliers and almost all incomes lie between 100000 - 250000

In [19]:
```
num_level1(credit,continous[1])
```

```
The mean of the AMT_CREDIT is 592545.253725
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 380967.40745319775
Number of missing values in the AMT_CREDIT is 0
```

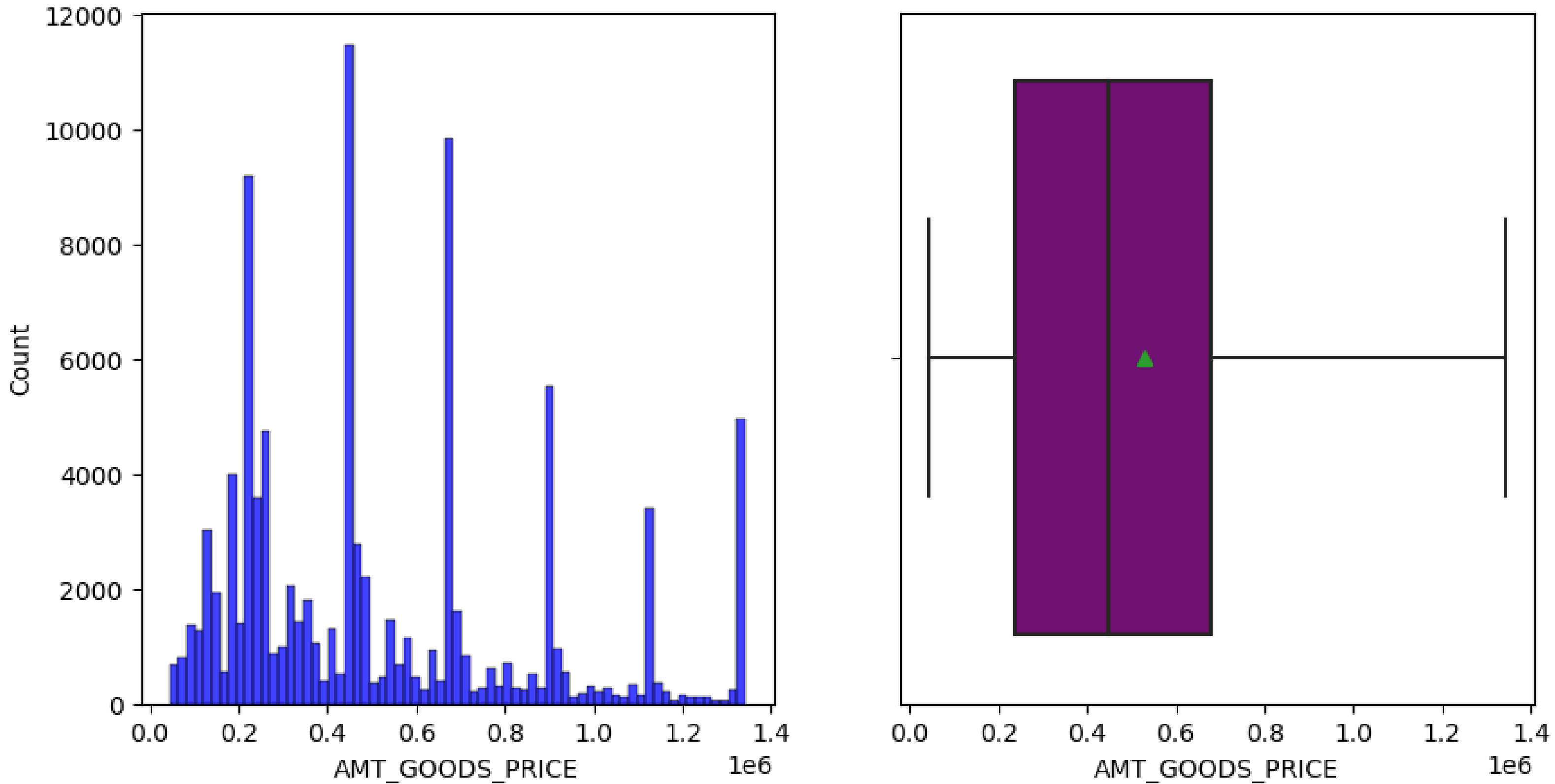


Interpretation :

From the above charts 0.25 to 0.80 is this amount_credit

In [20]:
```
num_level1(credit,continous[2])
```

```
The mean of the AMT_GOODS_PRICE is 528047.4542479408
The median of the AMT_GOODS_PRICE is 450000.0
The mode of the AMT_GOODS_PRICE is 450000.0
The standard deviation of the AMT_GOODS_PRICE is 337906.89079642127
Number of missing values in the AMT_GOODS_PRICE is 81
```



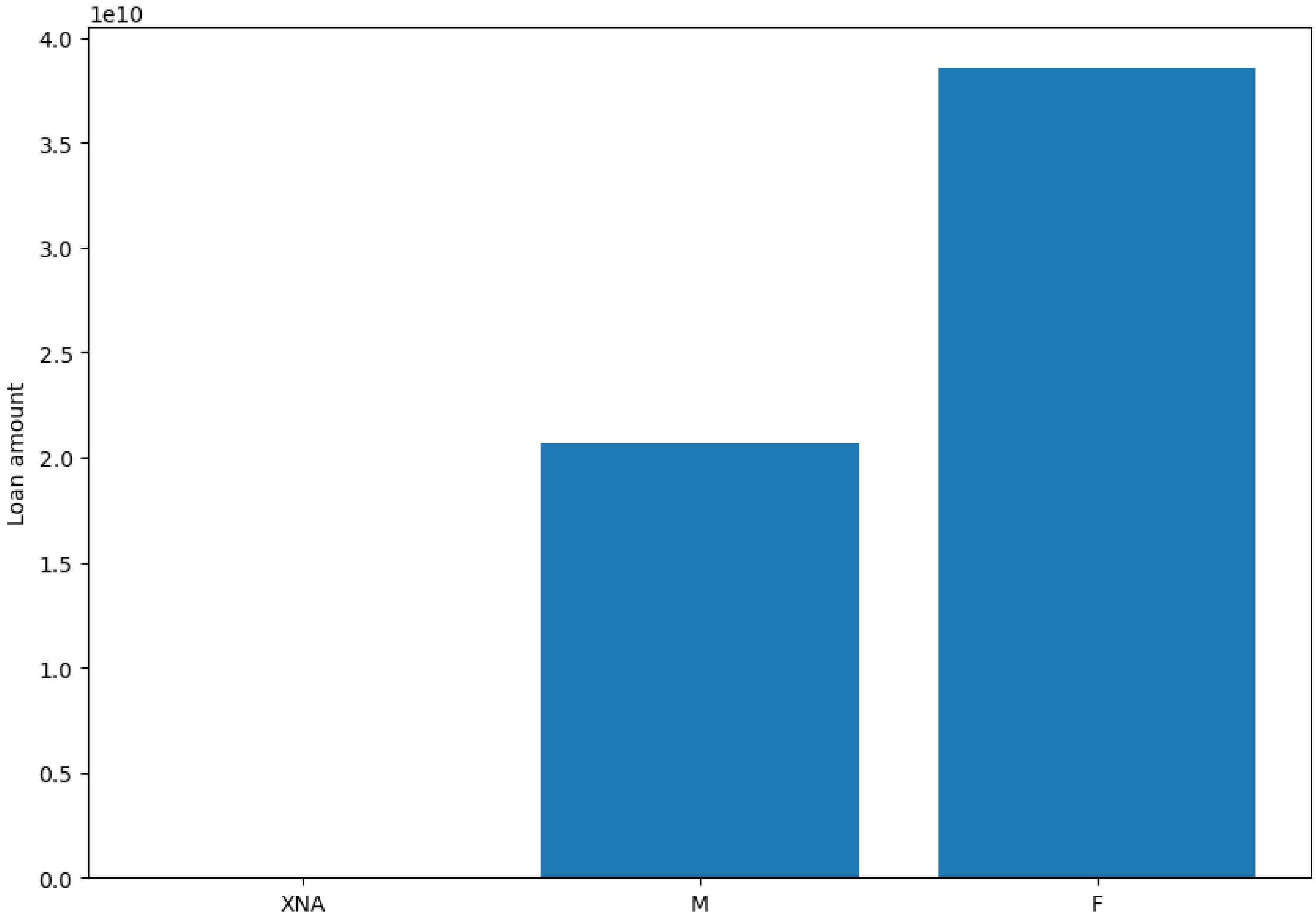
Interpretation:

The highest goods_price is around 450000 and average price of goods is 52000

Level 2: Bivariate Analysis (Getting closer to the BIG QUESTION:)

Gender vs Loan_amount

In [21]:
```
fig, ax = plt.subplots(figsize = (10, 7))
city_col=credit.groupby("GENDER").sum()["AMT_CREDIT"].sort_values()
plt.bar(city_col.index,city_col)
plt.ylabel("Loan amount")
plt.show()
```

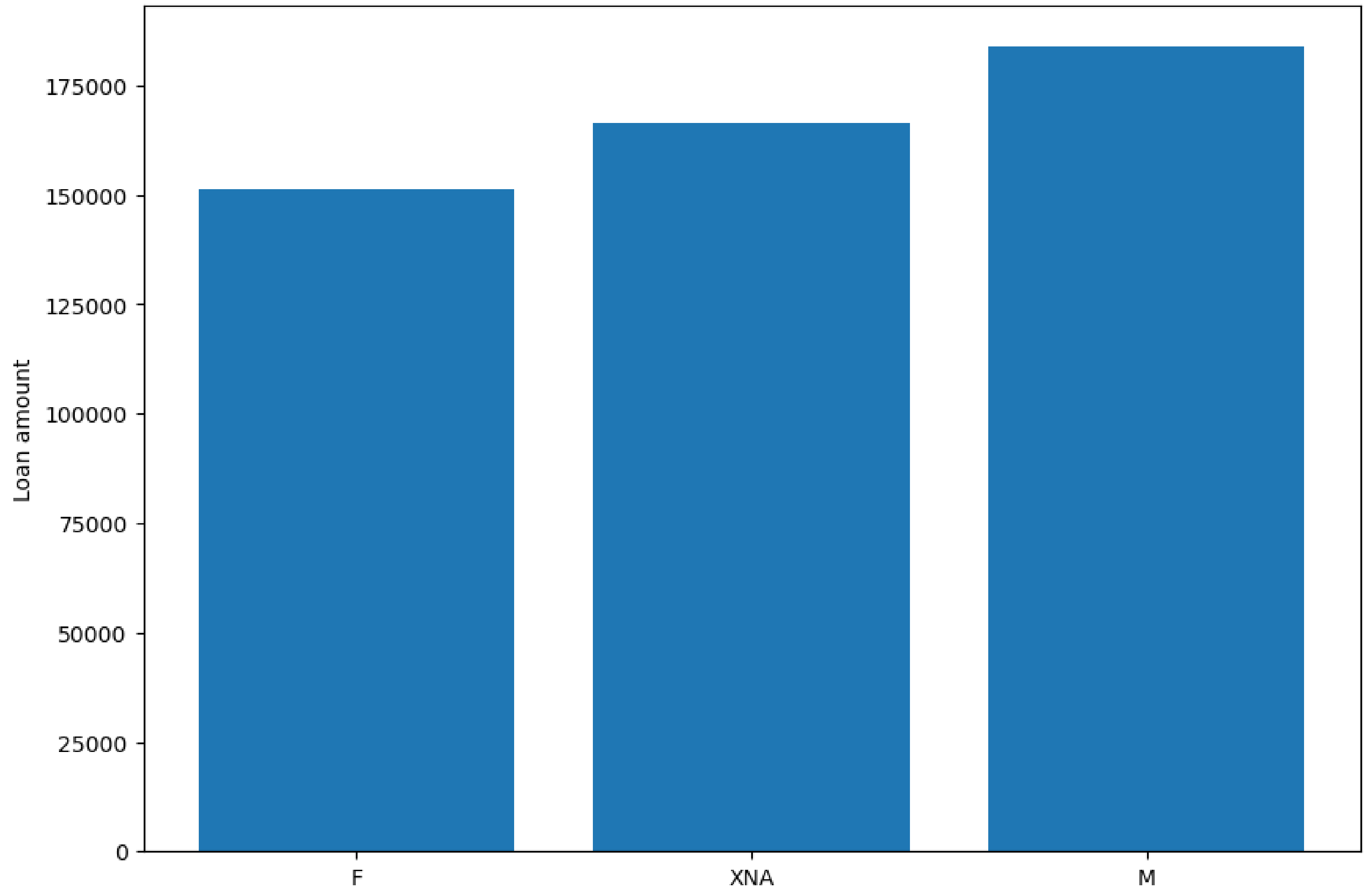


Interpretation:

From the above analysis we can see the female customers have the highest amount of loan borrowed than male customers

Income vs Gender

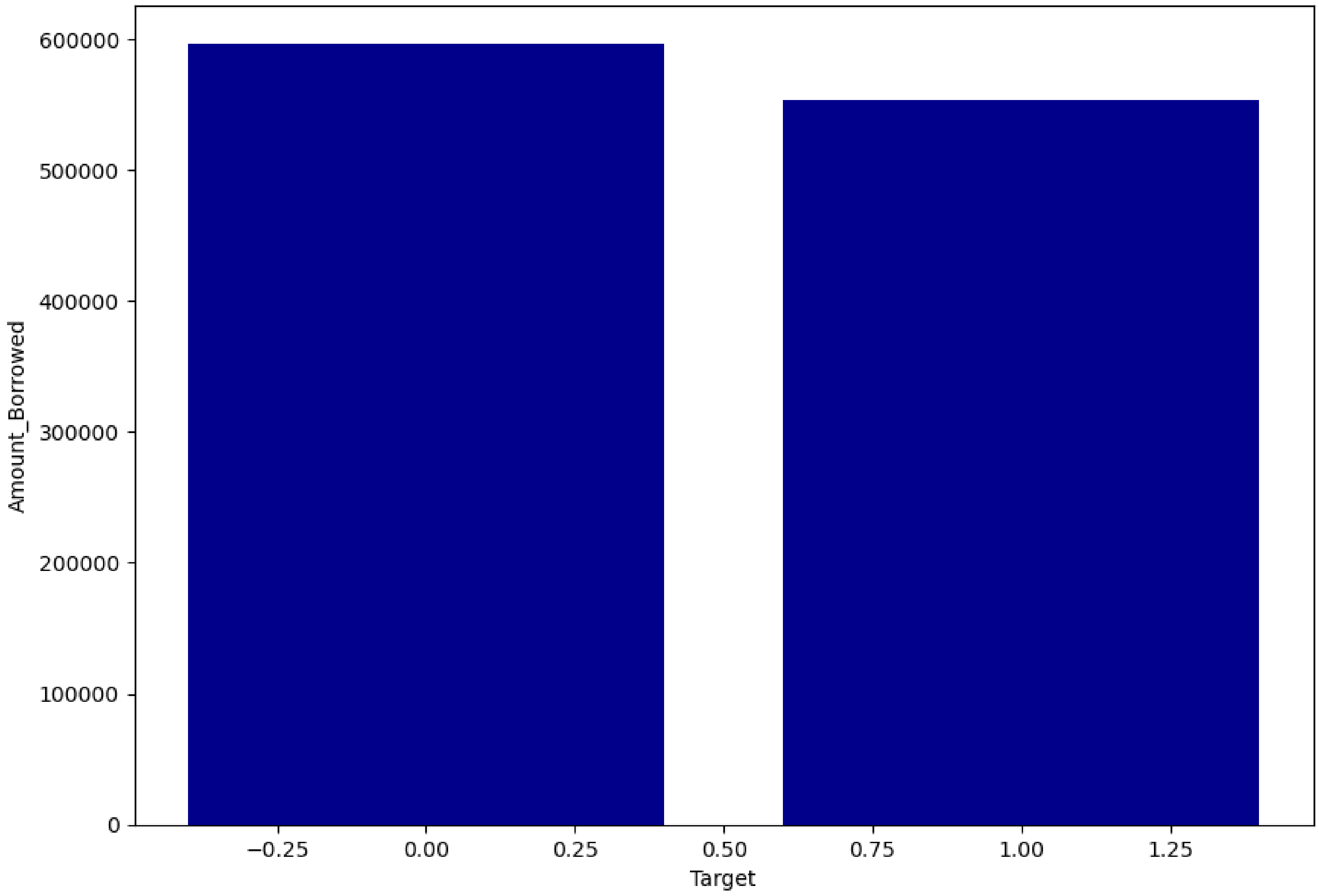
In [22]:
```
fig, ax = plt.subplots(figsize = (10, 7))
city_col=credit.groupby("GENDER").mean()["AMT_INCOME_TOTAL"].sort_values()
plt.bar(city_col.index,city_col)
plt.ylabel("Loan amount")
plt.show()
```



Interpretation:

We can see that male customers on average earn more income than female customers even though female customers are higher in numbers

In [23]:
```
fig, ax = plt.subplots(figsize = (10, 7))
fur_col=credit.groupby("TARGET").mean()["AMT_CREDIT"].sort_values()
plt.bar(fur_col.index,fur_col,color="Darkblue")
plt.xlabel("Target")
plt.ylabel("Amount_Borrowed")
plt.show()
print("Relative percentage of customers having difficulty = ",553757.684357/(595960.747636+553757.684357)*100," %")
```




```
Relative percentage of customers having difficulty =  48.164634831249835  %
```

Interpretation:

We can see close to 48% of customers have had difficulties

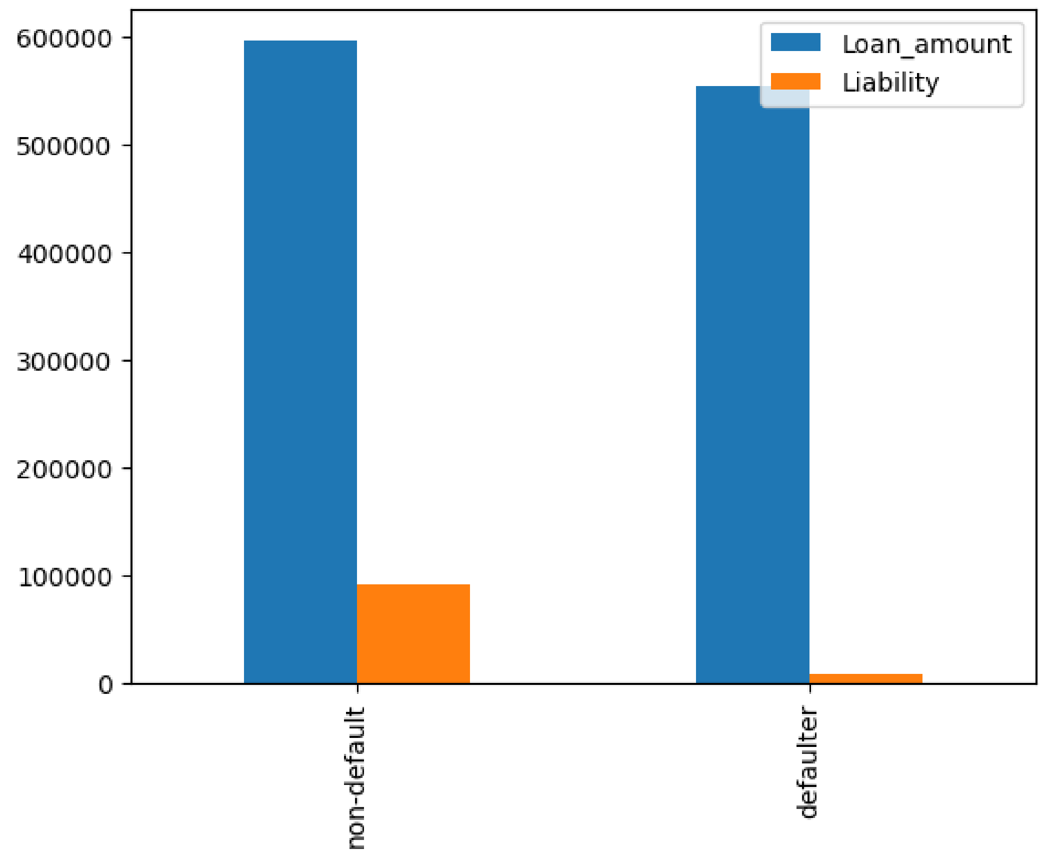
Level 3 - analysis

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.

1. How does a defaulter and genuine customer vary in terms of liability and income?
2. How does a education a factor of indicating individual's spending habits?
3. which profession of customers is the safest to loan to?

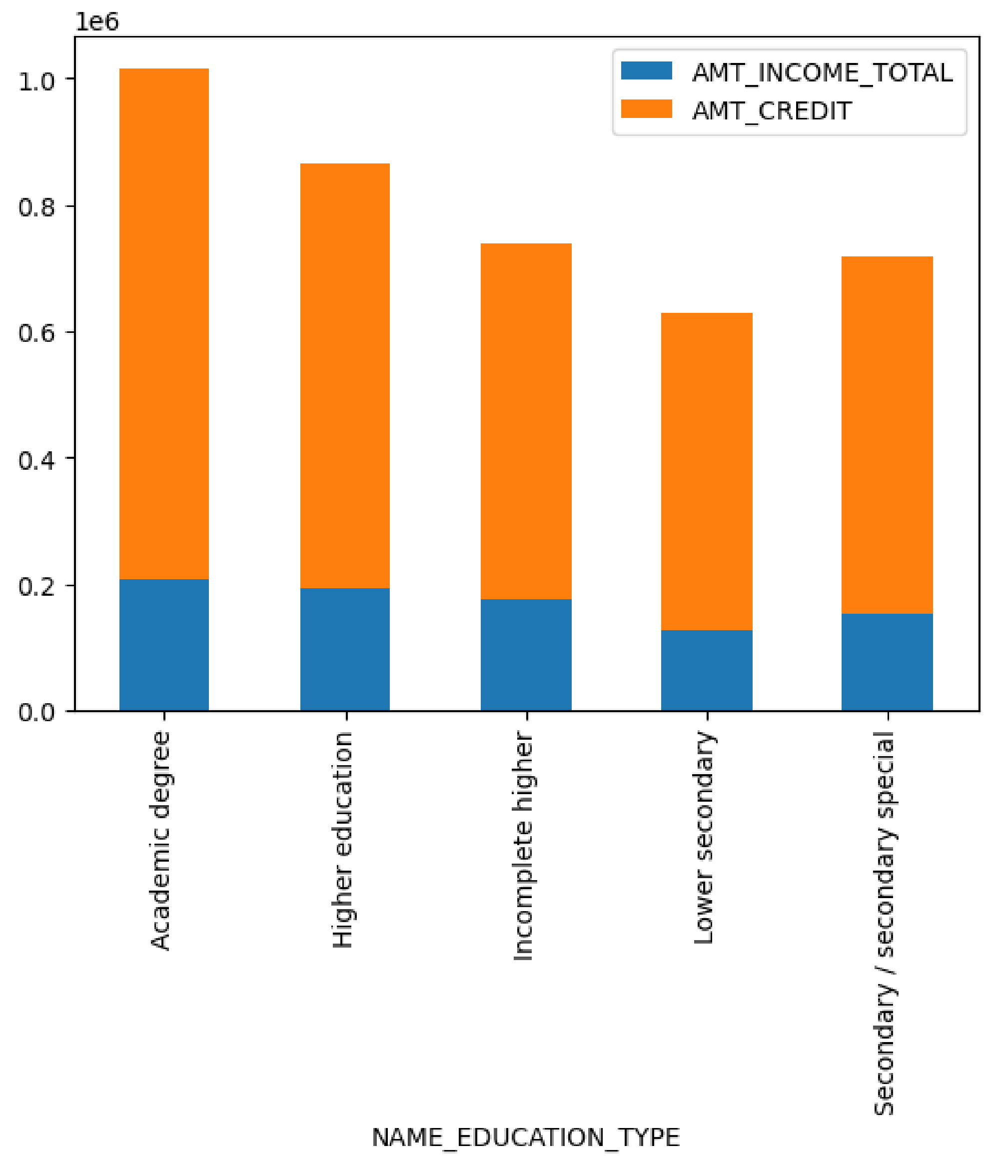
In [24]:
```
avg_loan=credit.groupby("TARGET").mean()["AMT_CREDIT"].values
lia_c=credit.groupby("TARGET")["House"].count().values
pd.DataFrame([avg_loan,lia_c],index=["Loan_amount","Liability"],columns=["non-default","defaulter"]).T.plot(kind="bar")
```

Out[24]: <AxesSubplot: >



From above graph we are able to see that most defaulters have no immovable asset like a house but have very much equal amounts of loan amounts compared to those who do...its evidental that there is higher chance that the customer might default if has no assets

In [25]:
```
m=credit.groupby("NAME_EDUCATION_TYPE").mean().loc[:,["AMT_INCOME_TOTAL","AMT_CREDIT"]].sort_index()
m.plot(kind="bar",stacked=True)
plt.show()
```



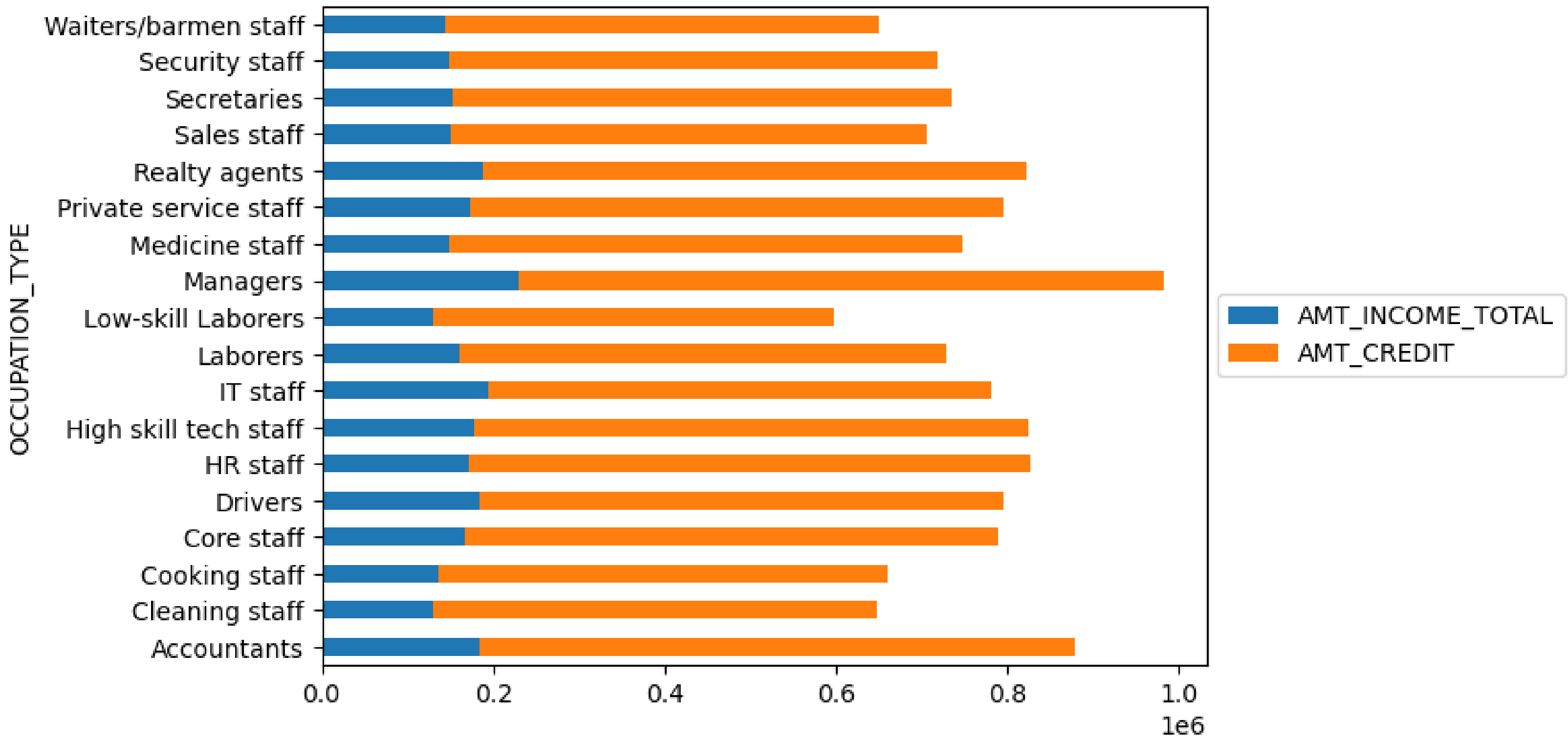
Its astonishing that lower secondary education group of customer shares very similar salary to academic degree customer although the total loan amount infered is nearly half...from this its observable that education has very minimal on income status of this group ..but in some way a person with high academic degree is tending to have more liability than other groups so the safest groups to provide a loan is secondaru special and higher education

In [26]:

```
m.to_csv("K:\Desktop\education.csv")
```

In [27]:

```
m=credit.groupby("OCCUPATION_TYPE").mean().loc[:,["AMT_INCOME_TOTAL","AMT_CREDIT"]]
#m["sum"]=m['AMT_INCOME_TOTAL']+m['AMT_CREDIT']
m.plot(kind="barh",stacked=True)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```



the people who have higher loans on average tend to be defaulters as many genuine customers on average have less overall ratio of income to loan which gives them a upper hand .. so the safest customers to target are IT stadd High skill tech stadd HR staff and others