CREDIT EDA ASSIGNMENT

SOURAV DEY

Data Import

- First we need to import the data to the system so that we can work with that data set

- So for that we use the command
- from google.colab import drive
- drive.mount('/content/drive’)

- Once the data is loaded we import the different headers
- These header help us to do further detailed work in Python

- Once the data and all the values are loaed we will be cleaning the data and removing all bad values from the data set so that we can move ahead with the analysis of the data

- There are two major data set
- Previous application data
- Application data

- Once the data is organized and ordered we can start with the different functions and requirements that were necessary

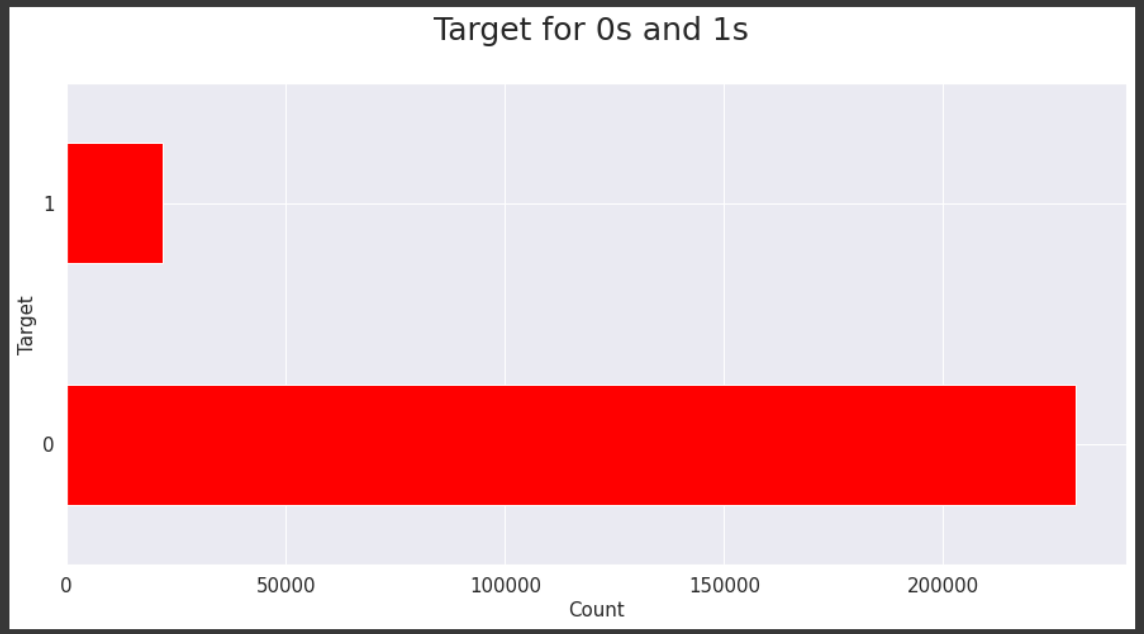
```
[94] bins_income = [0,25000,50000,75000,100000,125000,150000,175000,200000,225000,250000,275000,300000,325000,350000,375000,400000,425000,450000,475000,500000,10000000000]
     slot_income = ['0-25000', '25000-50000','50000-75000','75000,100000','100000-125000', '125000-150000', '150000-175000','175000-200000',
            '200000-225000','225000-250000','250000-275000','275000-300000','300000-325000','325000-350000','350000-375000',
            '375000-400000','400000-425000','425000-450000','450000-475000','475000-500000','500000 and above']

     df_appdata['AMT_INCOME_RANGE']=pd.cut(df_appdata['AMT_INCOME_TOTAL'],bins_income,labels=slot_income)
```

```
     bins_credit = [0,150000,200000,250000,300000,350000,400000,450000,500000,550000,600000,650000,700000,750000,800000,850000,900000,1000000000]
     slots_credit = ['0-150000', '150000-200000','200000-250000', '250000-300000', '300000-350000', '350000-400000','400000-450000',
             '450000-500000','500000-550000','550000-600000','600000-650000','650000-700000','700000-750000','750000-800000',
             '800000-850000','850000-900000','900000 and above']

     df_appdata['AMT_CREDIT_RANGE']=pd.cut(df_appdata['AMT_CREDIT'],bins_credit,labels=slots_credit)
```

- We start with setting the bin values and the slot values for the different ranges

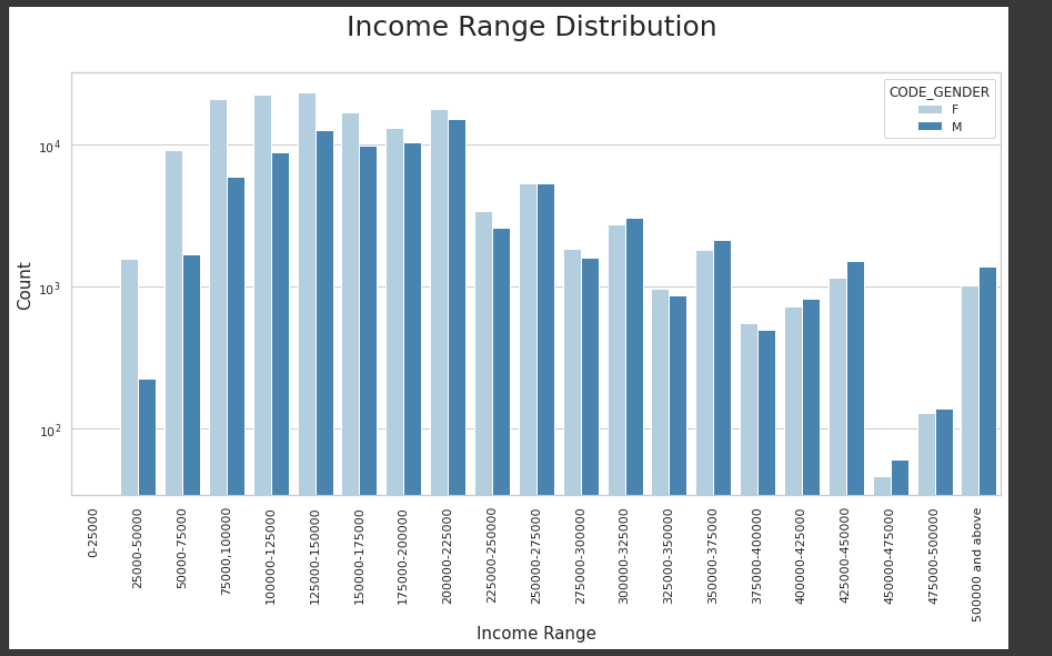
- Once that is done we can start to check values for target 1 and 0

```
plt.figure(figsize=[15,7])
df_appdata.TARGET.value_counts().plot.barh(color ='Red')

plt.title('Target for 0s and 1s \n', fontsize =25)
plt.xlabel('Count', fontsize = 15)
plt.ylabel('Target', fontsize = 15)
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)

plt.show()
```



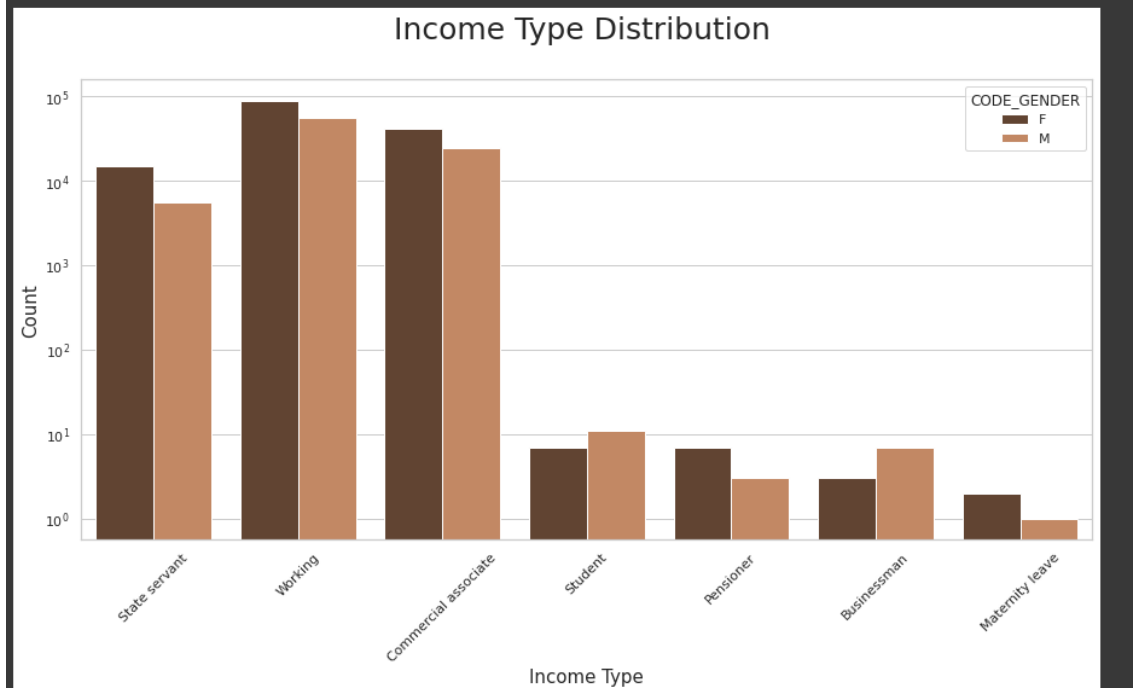
Once the data is organized we get the income distribution range and the different values that are assigned to it and a visual representation of the same

```
plt.figure(figsize=[15,7])
sns.set_style('whitegrid')
sns.countplot(data= target_0, x='AMT_INCOME_RANGE',hue='CODE_GENDER', palette='Blues')
plt.xticks(rotation= 90)

plt.title('Income Range Distribution \n', fontsize =25)
plt.xlabel('Income Range', fontsize = 15)
plt.ylabel('Count', fontsize = 15)
plt.yscale('log')
plt.show()
```



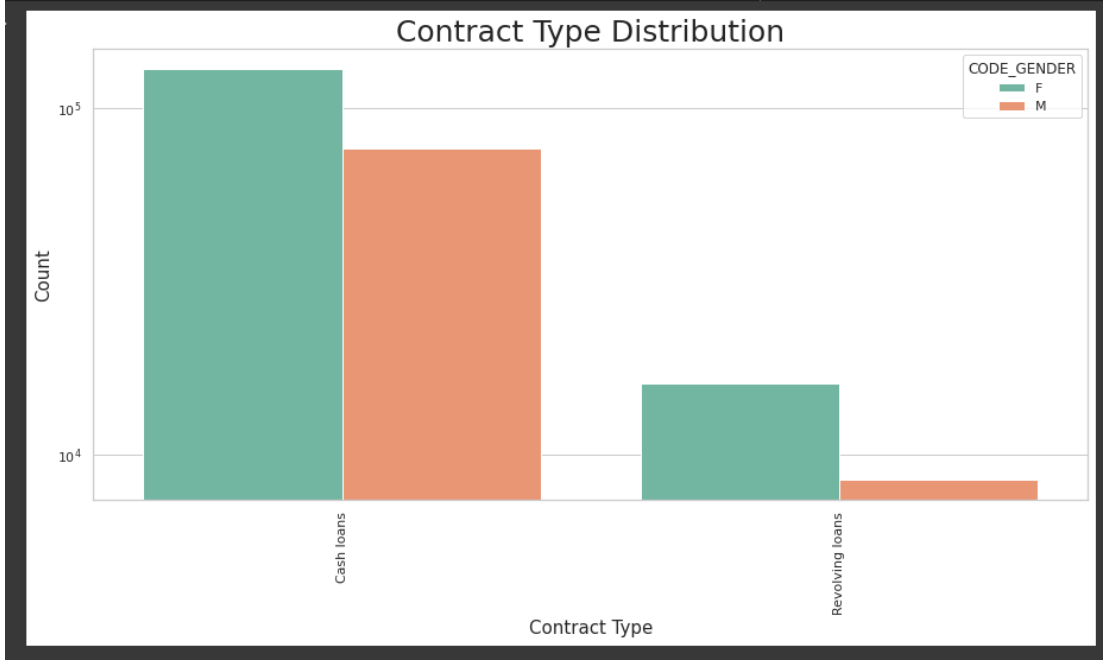
And also the income distribution range of the value

```
plt.figure(figsize=[15,7])
sns.set_style('whitegrid')
sns.countplot(data= target_0, x='NAME_INCOME_TYPE',hue='CODE_GENDER', palette='copper')
plt.xticks(rotation= 45)

plt.title('Income Type Distribution \n', fontsize =25)
plt.xlabel('Income Type', fontsize = 15)
plt.ylabel('Count', fontsize = 15)
plt.yscale('log')
plt.show()
```



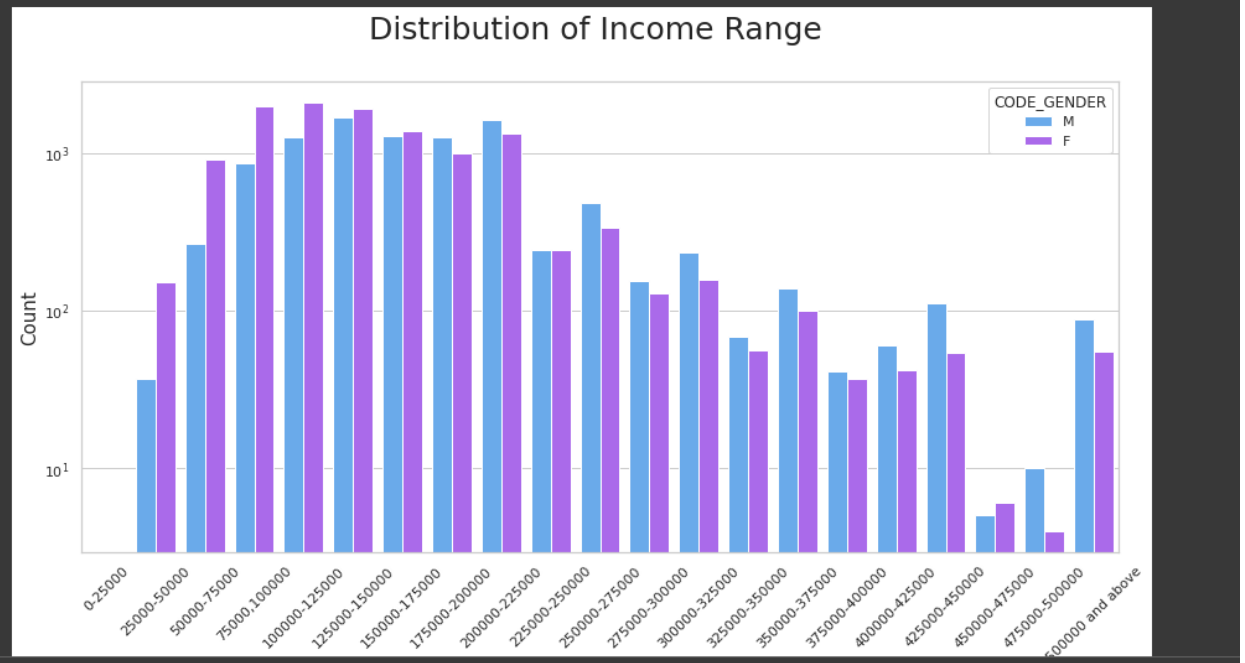
The contract distribution type

```
plt.figure(figsize=[15,7])
sns.set_style('whitegrid')
sns.countplot(data= target_0, x='NAME_CONTRACT_TYPE',hue='CODE_GENDER', palette='Set2')
plt.xticks(rotation= 90)

plt.title('Contract Type Distribution', fontsize =25)
plt.xlabel('Contract Type', fontsize = 15)
plt.ylabel('Count', fontsize = 15)
plt.yscale('log')
plt.show()
```



Distribution of Income range from the given data is derived and now we have a visual of how the income is segregated among male and female

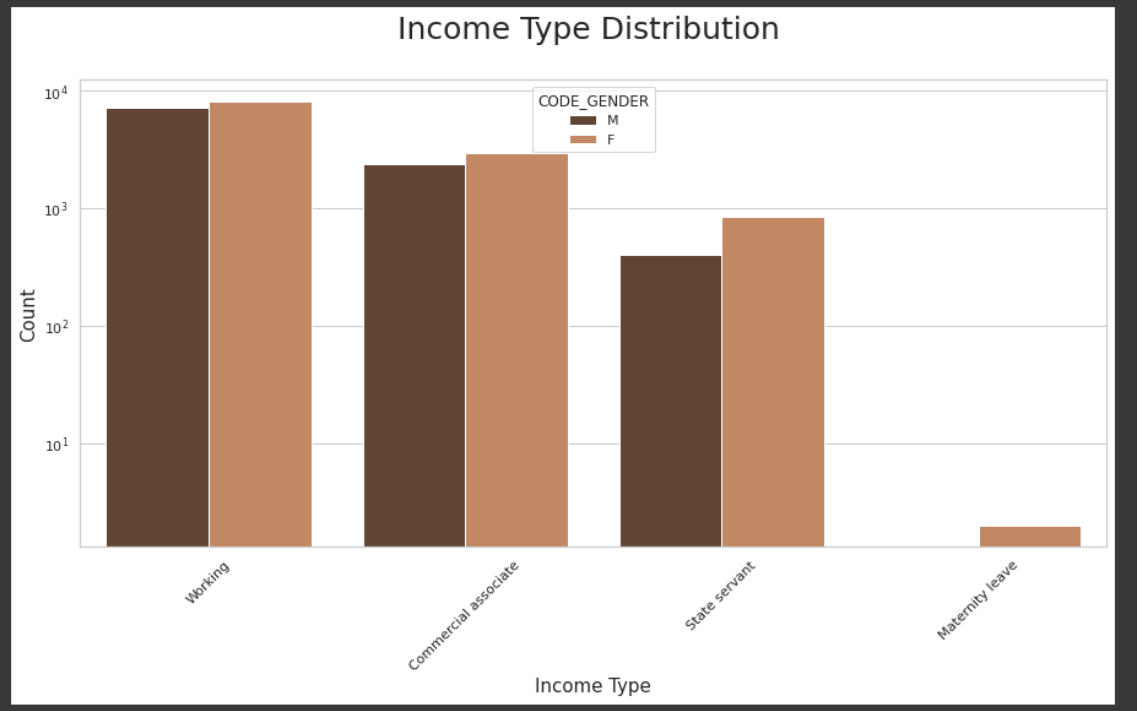
```
[118] plt.figure(figsize=[15,7])
      sns.set_style('whitegrid')
      sns.countplot(data= target_1, x='AMT_INCOME_RANGE',hue='CODE_GENDER',palette='cool')
      plt.title('Distribution of diffrent org type \n', fontsize =25)
      plt.xticks(rotation= 45)

      plt.title('Distribution of Income Range \n', fontsize =25)
      plt.xlabel('Income Range', fontsize = 15)
      plt.ylabel('Count', fontsize = 15)
      plt.yscale('log')
      plt.show()
```



✓ 2s completed at 8:09 PM

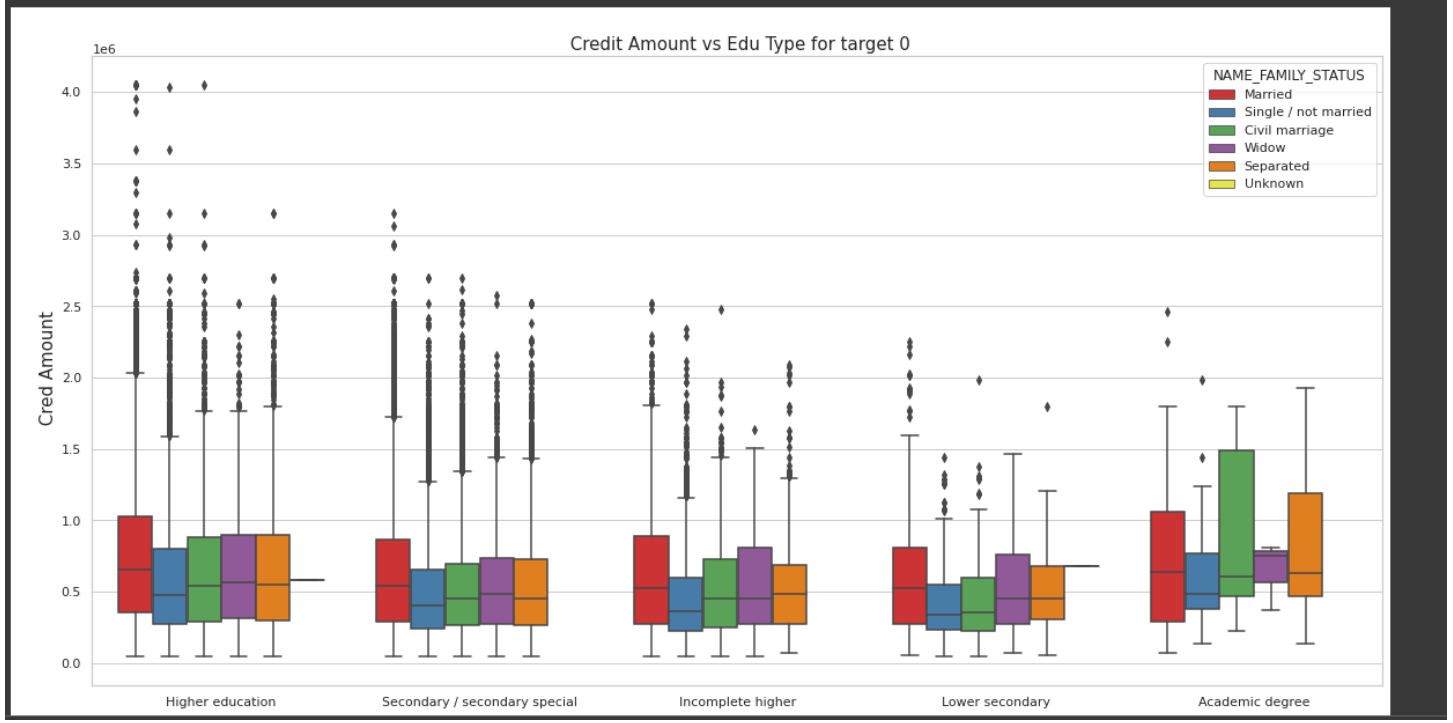
And the income distribution

```
plt.figure(figsize=[15,7])
sns.set_style('whitegrid')
sns.countplot(data= target_1, x='NAME_INCOME_TYPE',hue='CODE_GENDER', palette='copper')
plt.xticks(rotation= 45)

plt.title('Income Type Distribution \n', fontsize =25)
plt.xlabel('Income Type', fontsize = 15)
plt.ylabel('Count', fontsize = 15)
plt.yscale('log')
plt.show()
```



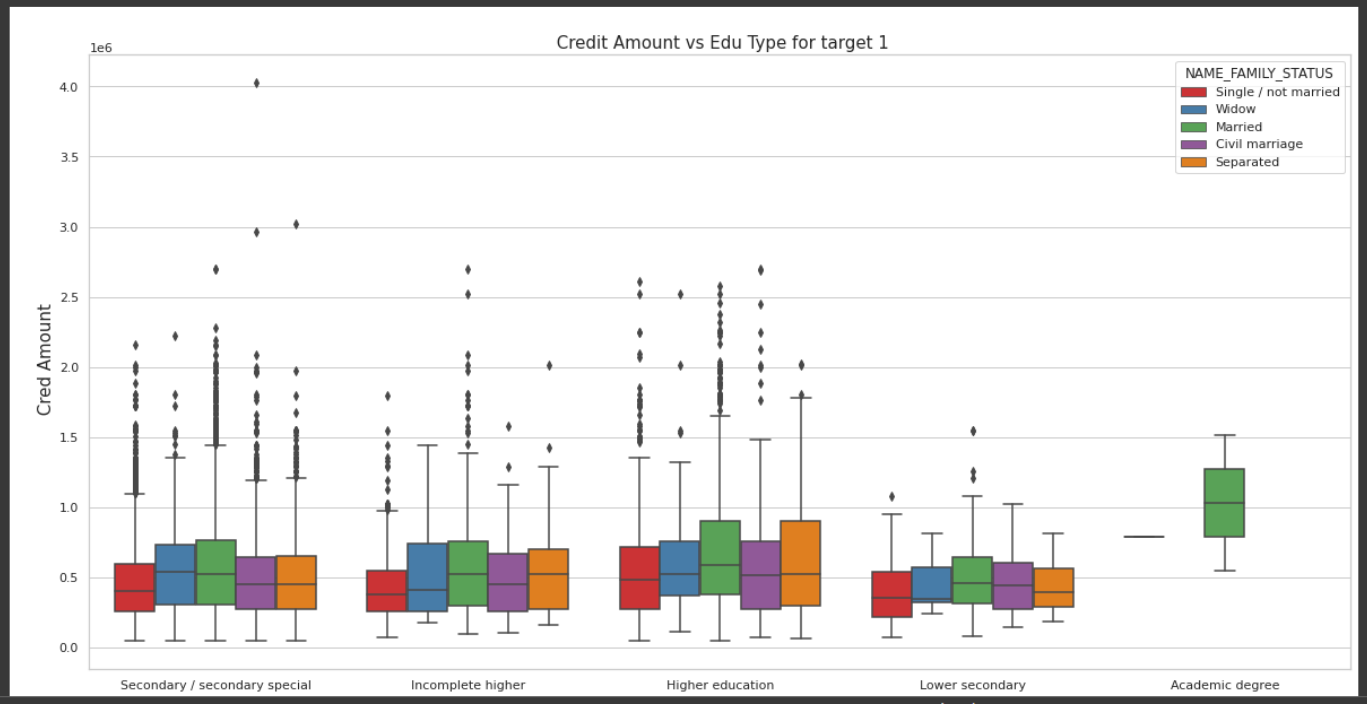
Below we have the credit amount and the education type for target 0

```
plt.figure(figsize=[20,10] )

sns.boxplot(data= target_0,x='NAME_EDUCATION_TYPE', y='AMT_CREDIT', hue='NAME_FAMILY_STATUS', orient='v',palette='Set1')
plt.xticks(rotation=0)
plt.xlabel('\n Eductaion Type',fontsize=15)
plt.ylabel('\n Cred Amount',fontsize=15)
plt.title('\n Credit Amount vs Edu Type for target 0', fontsize=15)
plt.show()
```



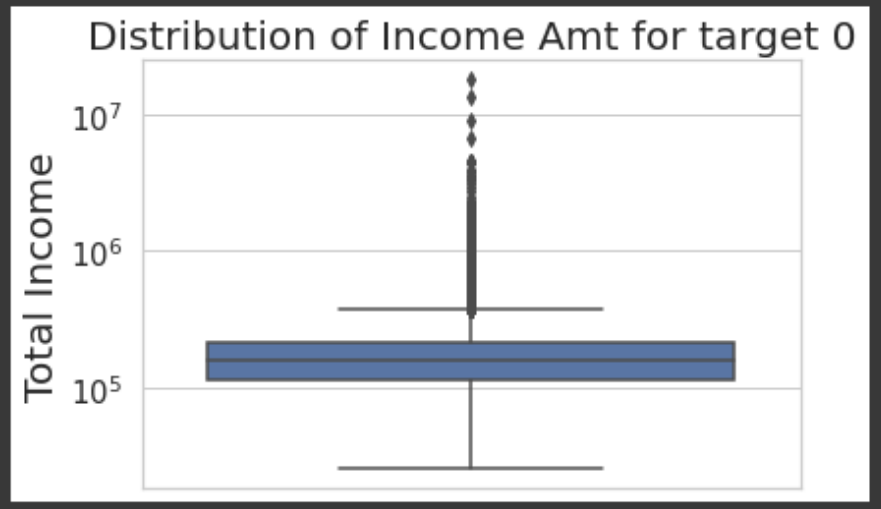
Below we have the credit amount and the education type for target 1

```
[165] plt.figure(figsize=[20,10] )

      sns.boxplot(data= target_1,x='NAME_EDUCATION_TYPE', y='AMT_CREDIT', hue='NAME_FAMILY_STATUS', orient='v',palette='Set1')
      plt.xticks(rotation=0)
      plt.xlabel('\n Eductaion Type',fontsize=15)
      plt.ylabel('\n Cred Amount',fontsize=15)
      plt.title('\n Credit Amount vs Edu Type for target 1', fontsize=15)
      plt.show()
```



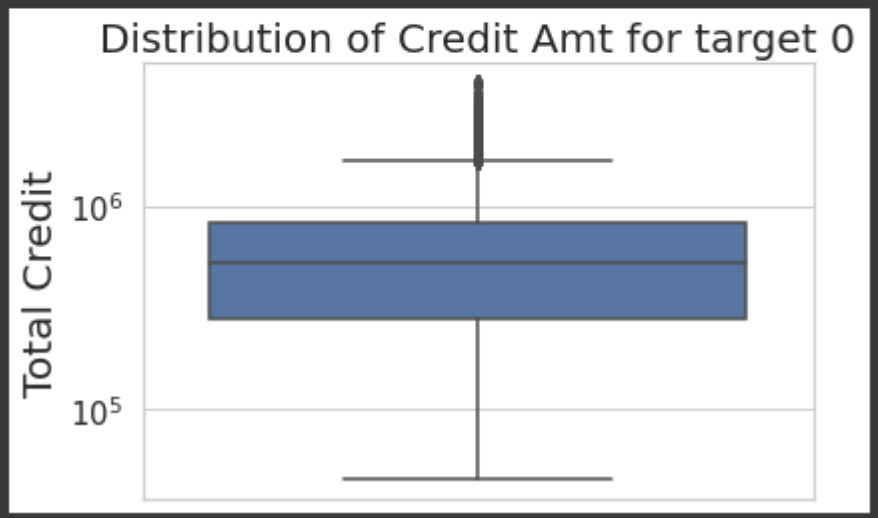
Data segregation and distribution of income amt for target value 0

```
[152] sns.set_style('whitegrid')

      sns.boxplot(data= target_0, y='AMT_INCOME_TOTAL')
      plt.yscale('log')
      plt.yticks(fontsize=15)
      plt.ylabel("Total Income", fontsize=20)
      plt.title("Distribution of Income Amt for target 0", fontsize=20)
      plt.show()
```



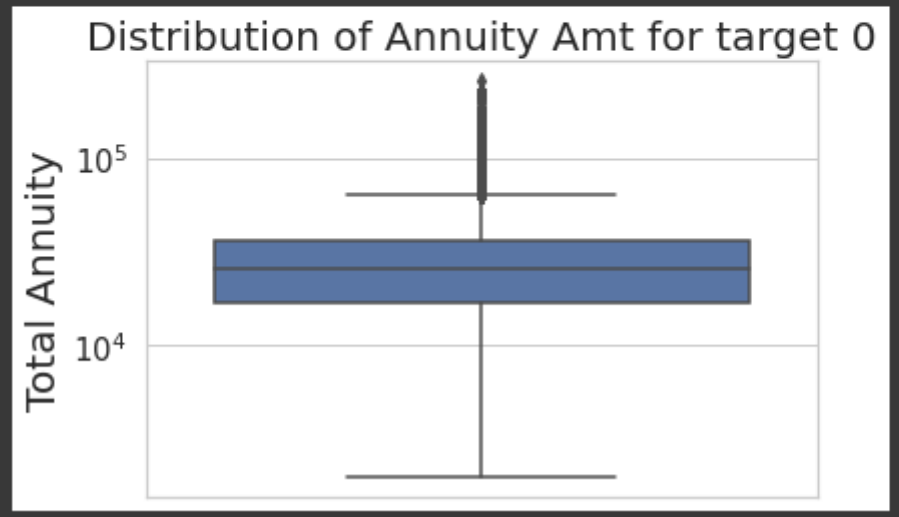
Data segregation and distribution of credit amt for target value 0

```
sns.set_style('whitegrid')

sns.boxplot(data= target_0, y='AMT_CREDIT')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel("Total Credit", fontsize=20)
plt.title("Distribution of Credit Amt for target 0", fontsize
plt.show()
```



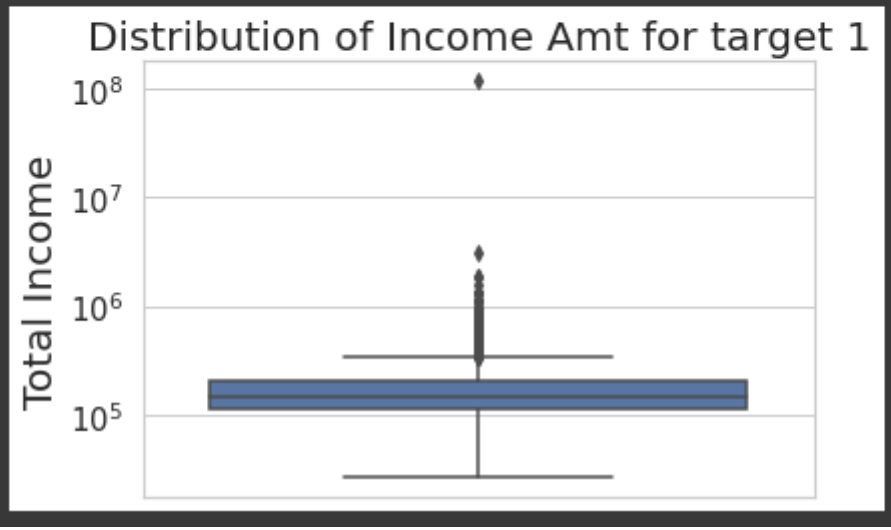
Data segregation and distribution of annuity amt for target value 0

```
[156] sns.set_style('whitegrid')

      sns.boxplot(data= target_0, y='AMT_ANNUITY')
      plt.yscale('log')
      plt.yticks(fontsize=15)
      plt.ylabel("Total Annuity", fontsize=20)
      plt.title("Distribution of Annuity Amt for target 0", fontsize=20)
      plt.show()
```



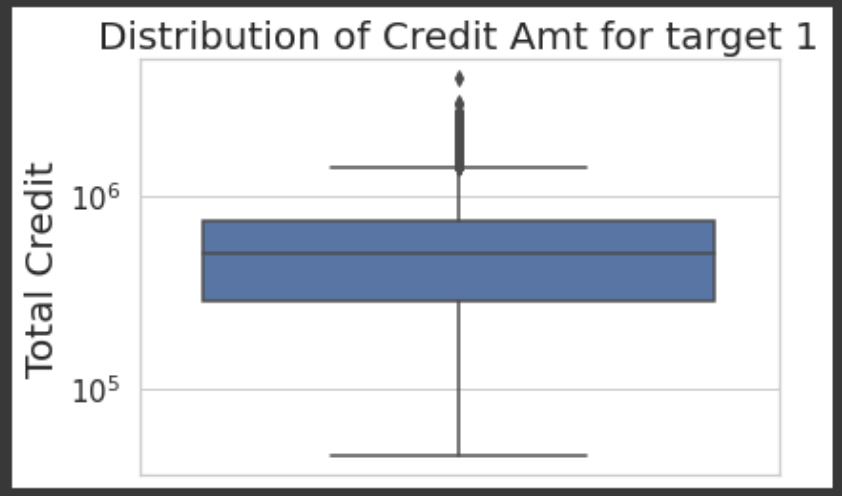
Data segregation and distribution of annuity amt for target value 1

```
sns.set_style('whitegrid')

sns.boxplot(data= target_1, y='AMT_INCOME_TOTAL')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel("Total Income", fontsize=20)
plt.title("Distribution of Income Amt for target 1", fontsize=20)
plt.show()
```



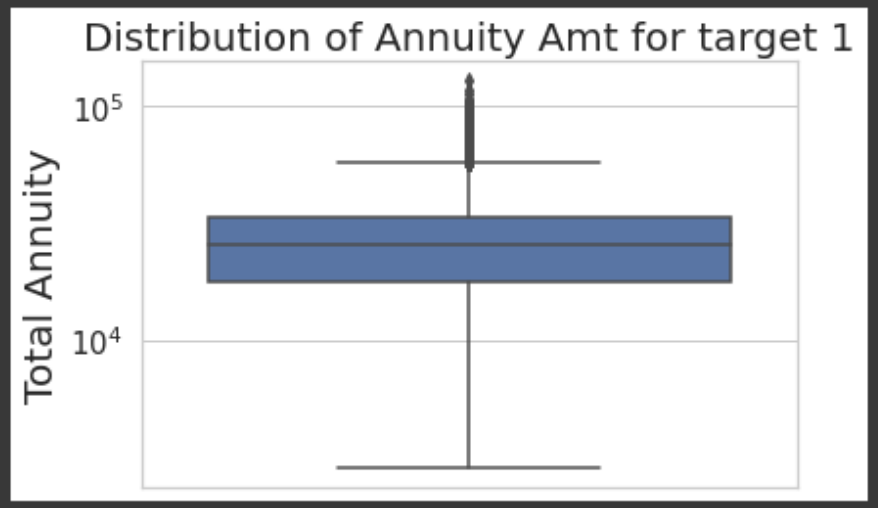
Data segregation and distribution of credit amt for target value 1

```
sns.set_style('whitegrid')

sns.boxplot(data= target_1, y='AMT_CREDIT')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel("Total Credit", fontsize=20)
plt.title("Distribution of Credit Amt for target 1", fontsize=20)
plt.show()
```



Data segregation and distribution of annuity amt for target value 1

```
sns.set_style('whitegrid')

sns.boxplot(data= target_1, y='AMT_ANNUITY')
plt.yscale('log')
plt.yticks(fontsize=15)
plt.ylabel("Total Annuity", fontsize=20)
plt.title("Distribution of Annuity Amt for target 1", fontsize=20
plt.show()
```



Once we have data as such we can also aggregate the tables to get more valuable outputs and understands a better way as to which customer is more likely to get a loan

These data helps us make the structed decision which in turn makes the utilization of the application and resource more meaningful.

Thank You