Bank Loan Case Study

This project involves analyzing loan application data for a finance company specializing in loans for urban customers. The company faces two significant risks: rejecting applicants who can repay and approving those who may default. The goal of the project is to understand the influence of customer and loan attributes on the likelihood of default through Exploratory Data Analysis (EDA)

Data Analytics Tasks:

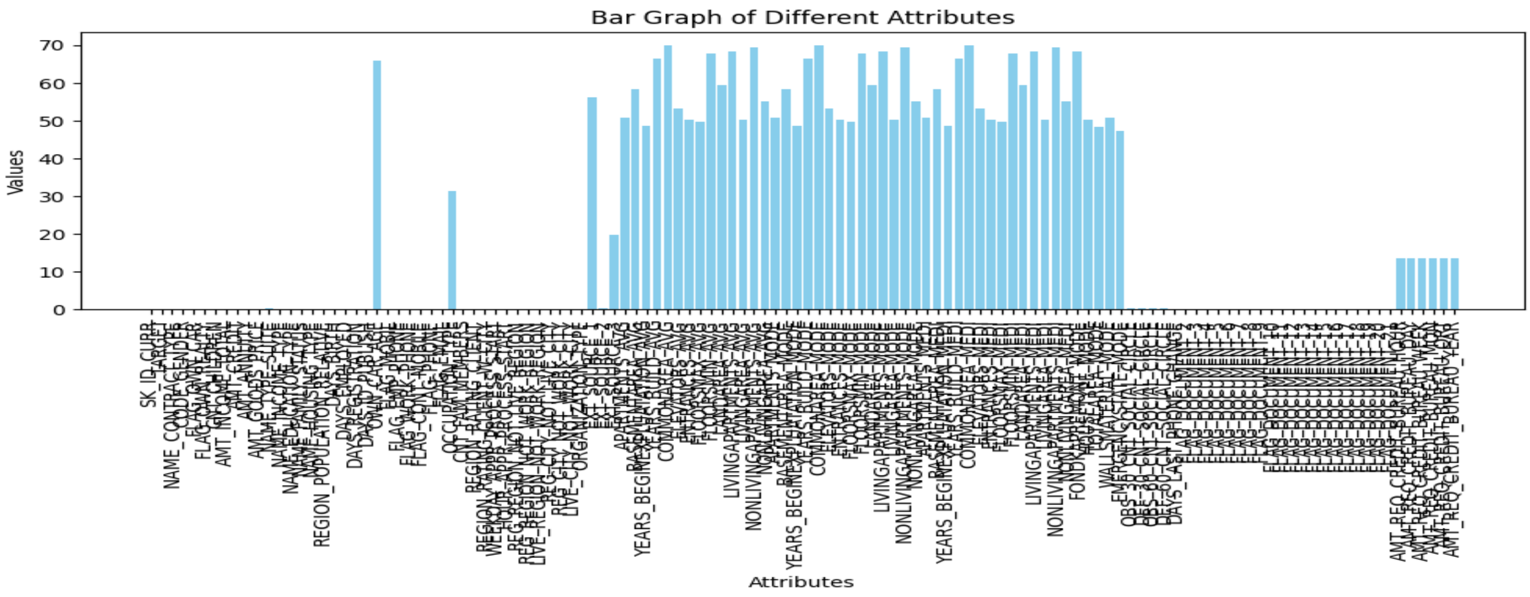
A. Identify Missing Data and Deal with it Appropriately: As a data analyst, you come across missing data in the loan application dataset. It is essential to handle missing data effectively to ensure the accuracy of the analysis.

Task: Identify the missing data in the dataset and decide on an appropriate method to deal with it using Excel built-in functions and features.

Code:

```
null_data1 = app_data.isnull().sum()/49999*100
        import matplotlib.pyplot as plt
        # Plotting the bar graph
        plt.figure(figsize=(10, 6))
        plt.bar(null_data1.keys(), null_data1.values, color='skyblue')
        plt.xticks(rotation=90)  # Rotate the x-axis labels for better readability
        plt.title('Bar Graph of Different Attributes')
        plt.xlabel('Attributes')
        plt.ylabel('Values')
        plt.tight_layout()
        # Show the plot
        plt.show()
```

Output:



Above is the bar graph that represents the percentage of missing values in each columns.when it comes to the way the data has been handled, the attributes with more than or equal to 40% of missing values has been dropped. In the next step of cleaning some of the attributes which are not necessary for the analyzation has been removed. The columns that have been removed are

'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE','FLAG_EMAIL','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', 'EXT_SOURCE_2', 'EXT_SOURCE_3'.

In the next step the columns which have negative values has been converted into positive values by using absolute function. Those columns are

'DAYS_BIRTH', 'DAYS_EMPLOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'DAYS_LAST_PHONE_CHANGE'.

There were few columns with less missing values which could be replaced so those values have been replace by there mean median or mode depending on the data type those columns are

DAYS_LAST_PHONE_CHANGE	1
CNT_FAM_MEMBERS	1
AMT_ANNUITY	1
AMT_GOODS_PRICE	38
OBS_30_CNT_SOCIAL_CIRCLE	168
DEF_30_CNT_SOCIAL_CIRCLE	168
OBS_60_CNT_SOCIAL_CIRCLE	168
DEF_60_CNT_SOCIAL_CIRCLE	168
NAME_TYPE_SUITE	192
AMT_REQ_CREDIT_BUREAU_HOUR	6734
AMT_REQ_CREDIT_BUREAU_DAY	6734
AMT_REQ_CREDIT_BUREAU_WEEK	6734
AMT_REQ_CREDIT_BUREAU_MON	6734
AMT_REQ_CREDIT_BUREAU_QRT	6734
AMT_REQ_CREDIT_BUREAU_YEAR	6734
OCCUPATION_TYPE	15654

Similar process was applied on the previous application data set also.

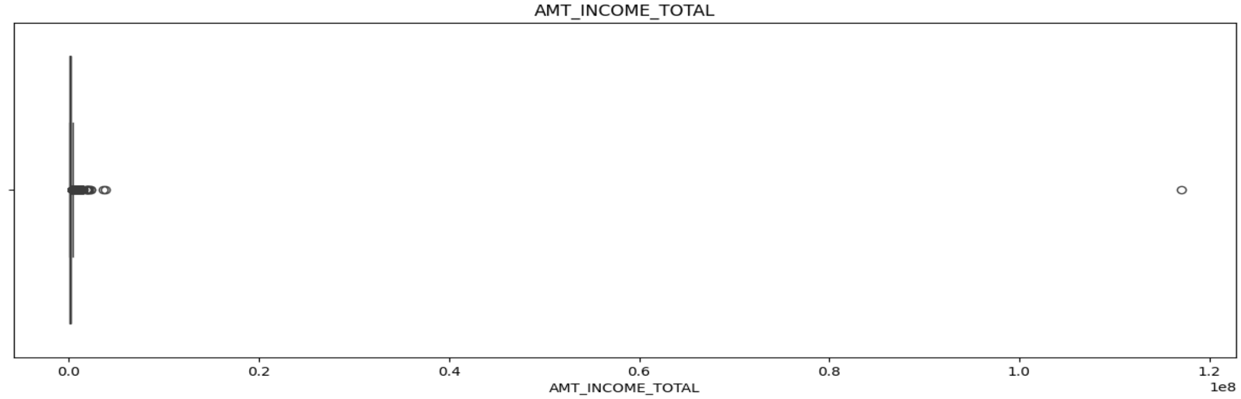
B. Identify Outliers in the Dataset: Outliers can significantly impact the analysis and distort the results. You need to identify outliers in the loan application dataset.

Task: Detect and identify outliers in the dataset using Excel statistical functions and features, focusing on numerical variables.

Code: outliers = pd.DataFrame()

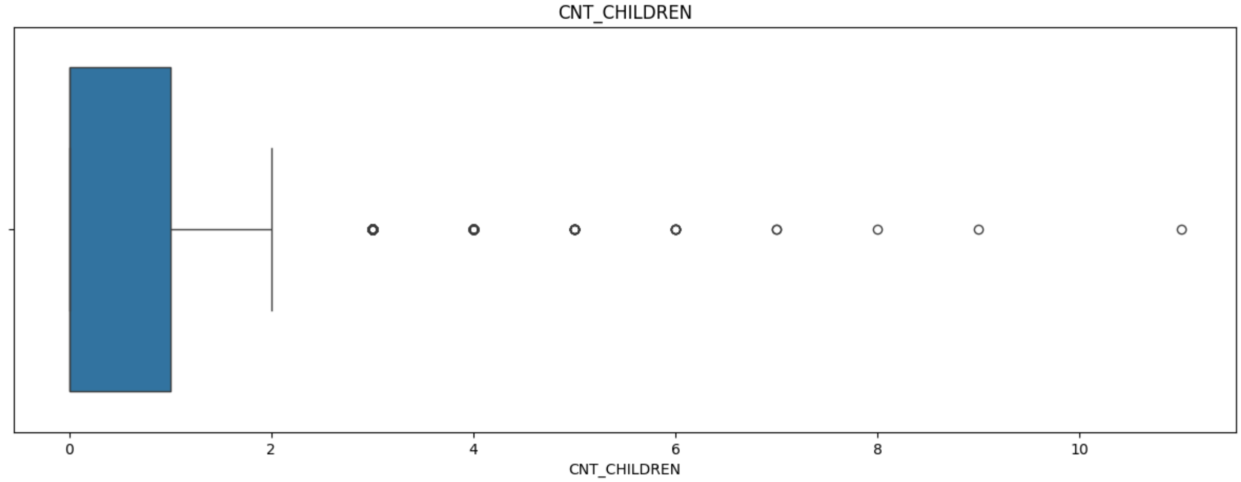
AMT_INCOME_TOTAL:

```
plt.figure(figsize = (15,5))
sns.boxplot(x = merged_final_data['AMT_INCOME_TOTAL'])
plt.title('AMT_INCOME_TOTAL')
plt.show()
```



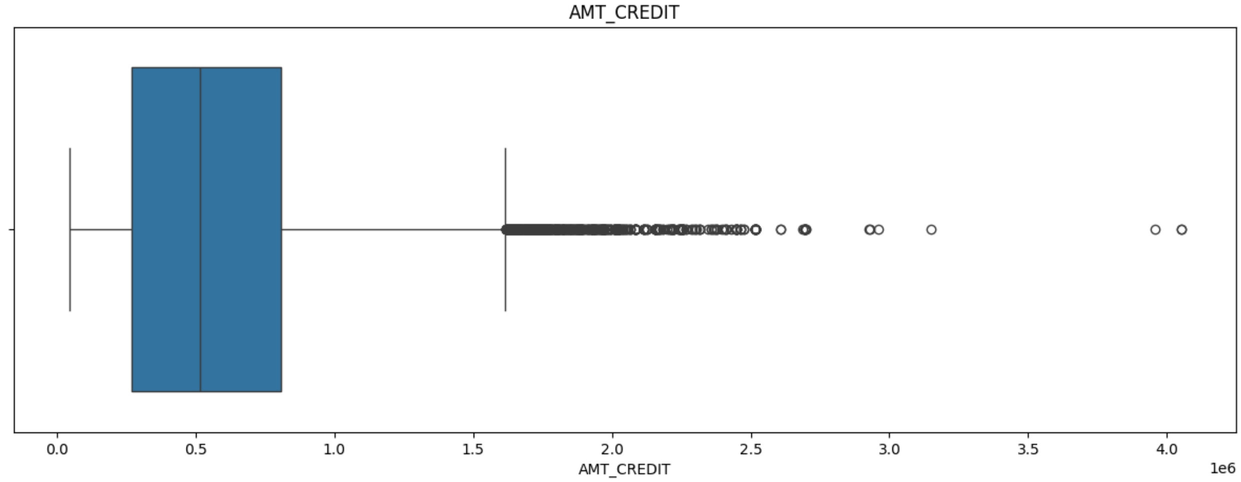
CNT_CHILDREN:

```
plt.figure(figsize = (15,5))
sns.boxplot(x = merged_final_data['CNT_CHILDREN'])
plt.title('CNT_CHILDREN')
plt.show()
```



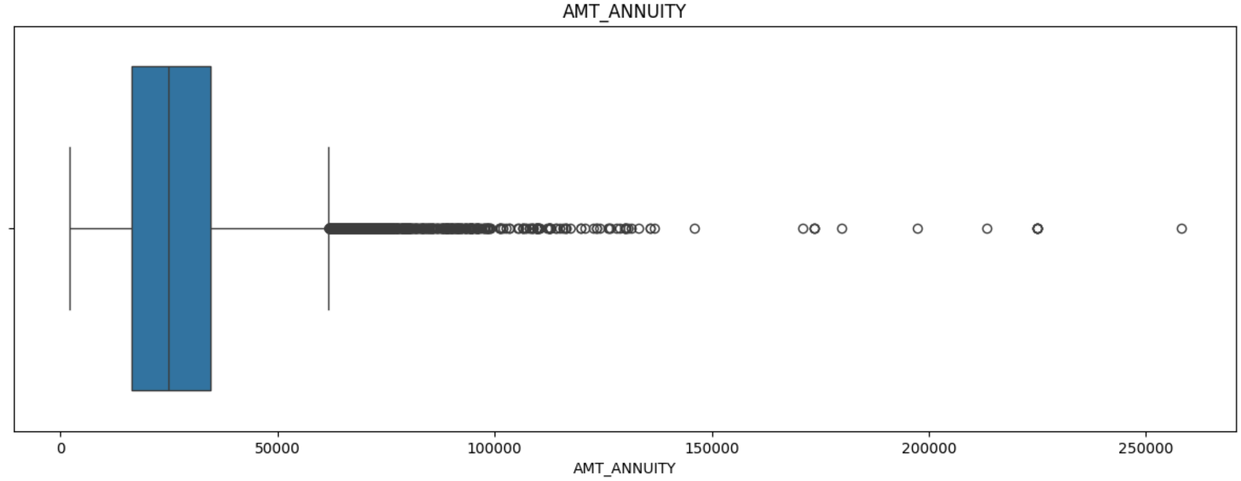
AMT_CREDIT:

```
plt.figure(figsize = (15,5))
sns.boxplot(x = merged_final_data['AMT_CREDIT'])
plt.title('AMT_CREDIT')
plt.show()
```



AMT_ANNUITY:

```
plt.figure(figsize = (15,5))
sns.boxplot(x = merged_final_data['AMT_ANNUITY'])
plt.title('AMT_ANNUITY')
plt.show()
```

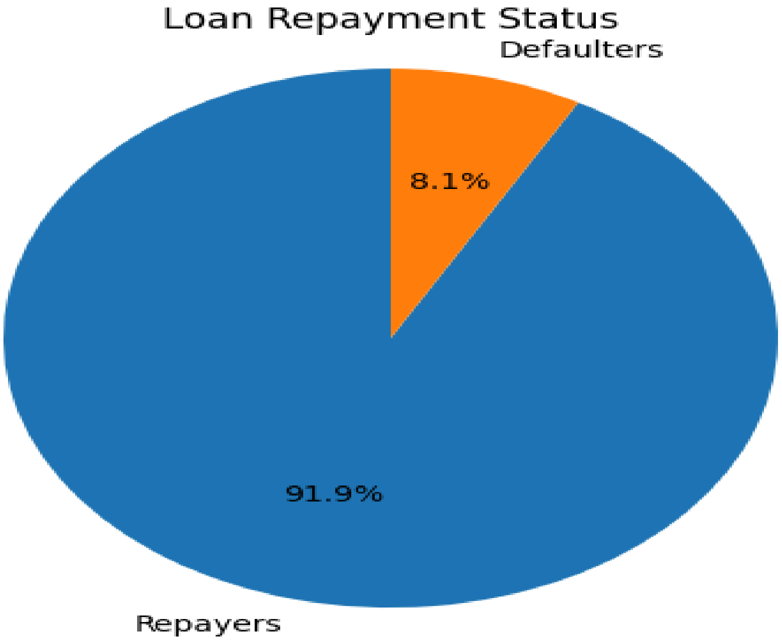


C. Analyze Data Imbalance: Data imbalance can affect the accuracy of the analysis, especially for binary classification problems. Understanding the data distribution is crucial for building reliable models.

Task: Determine if there is data imbalance in the loan application dataset and calculate the ratio of data imbalance using Excel functions.

Code:
```
Imbalance = merged_final_data["TARGET"].value_counts().reset_index()
labels = ['Repayers', 'Defaulters']
sizes = Imbalance['count']

fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title('Loan Repayment Status')
plt.show()

app_data_fil = merged_final_data['TARGET'].value_counts()
class_counts = app_data_fil
imbalance_ratio = class_counts[0] / class_counts[1]

"Imbalance Ratio: 1:{:.0f}".format(imbalance_ratio)
```

Output: Imbalance Ratio: 1:11



D. Perform Univariate, Segmented Univariate, and Bivariate Analysis: To gain insights into the driving factors of loan default, it is important to conduct various analyses on consumer and loan attributes.

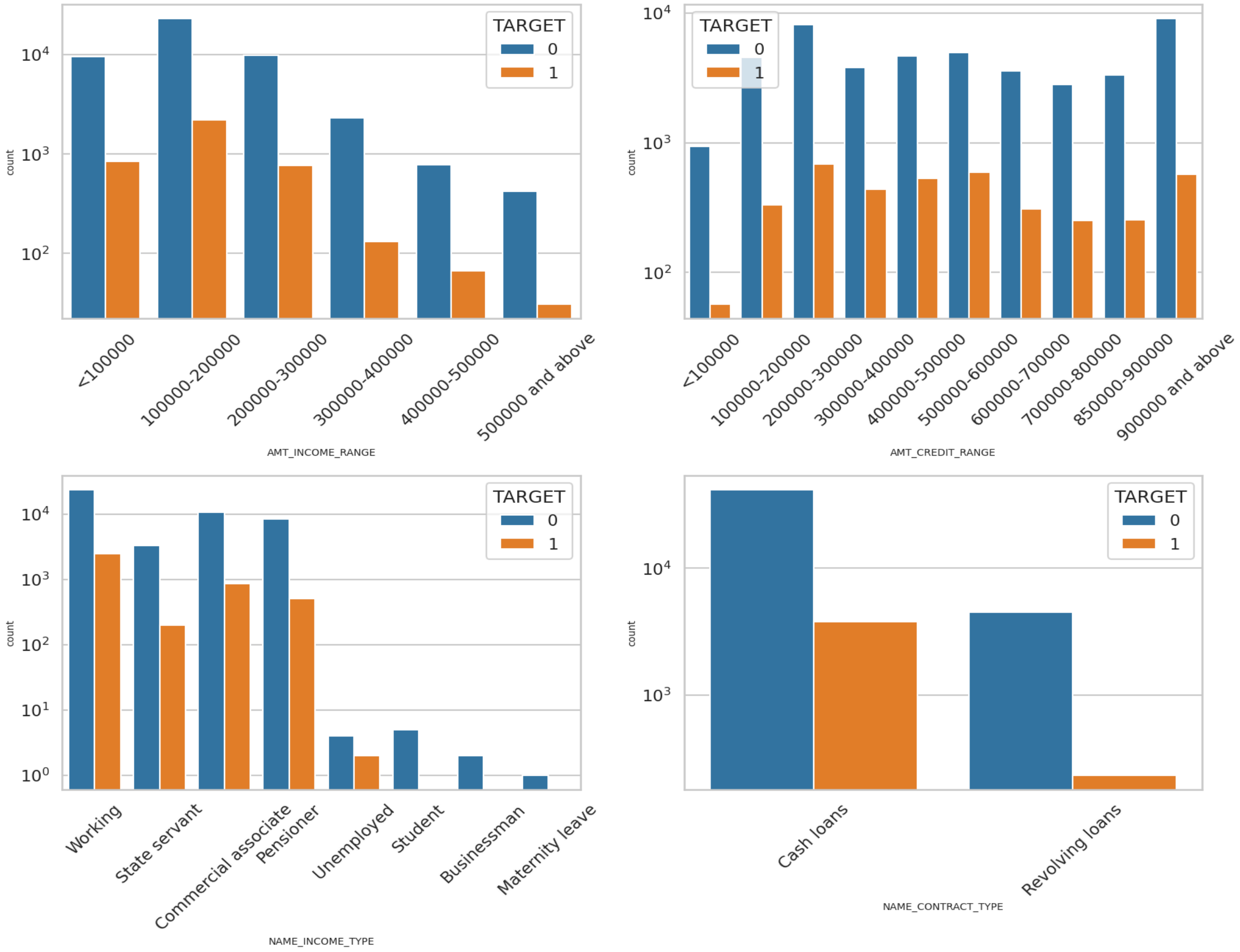
Task: Perform univariate analysis to understand the distribution of individual variables, segmented univariate analysis to compare variable distributions for different scenarios, and bivariate analysis to explore relationships between variables and the target variable using Excel functions and features.

Code:

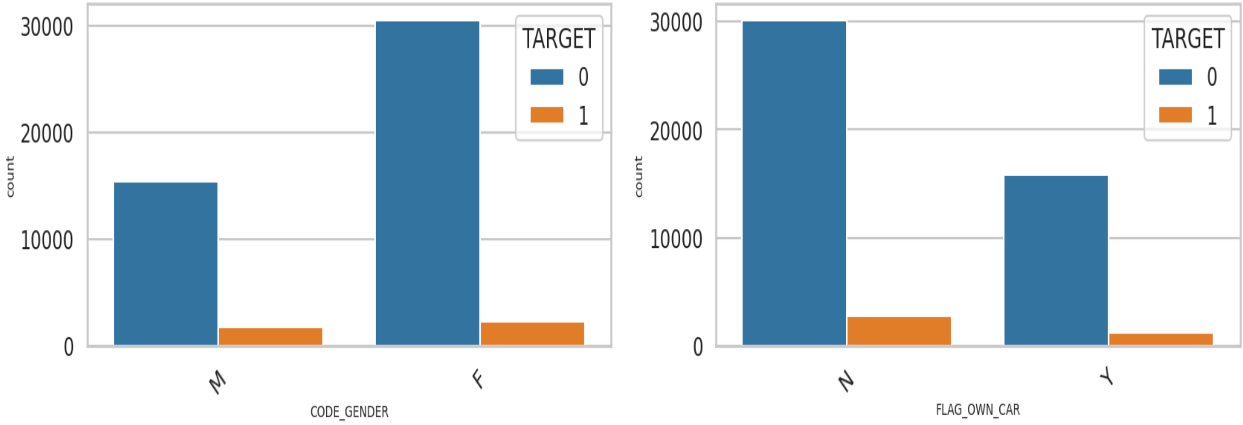
# Categoroical Univariate Analysis in logarithmic scale

```
features = ['AMT_INCOME_RANGE',
'AMT_CREDIT_RANGE','NAME_INCOME_TYPE','NAME_CONTRACT_TYPE']
plt.figure(figsize = (20, 15))

for i in enumerate(features):
   plt.subplot(2, 2, i[0]+1)
   plt.subplots_adjust(hspace=0.5)
   sns.countplot(x = i[1], hue = 'TARGET', data = merged_final_data)

   plt.rcParams['axes.titlesize'] = 16

   plt.xticks(rotation = 45)
   plt.yscale('log')
```

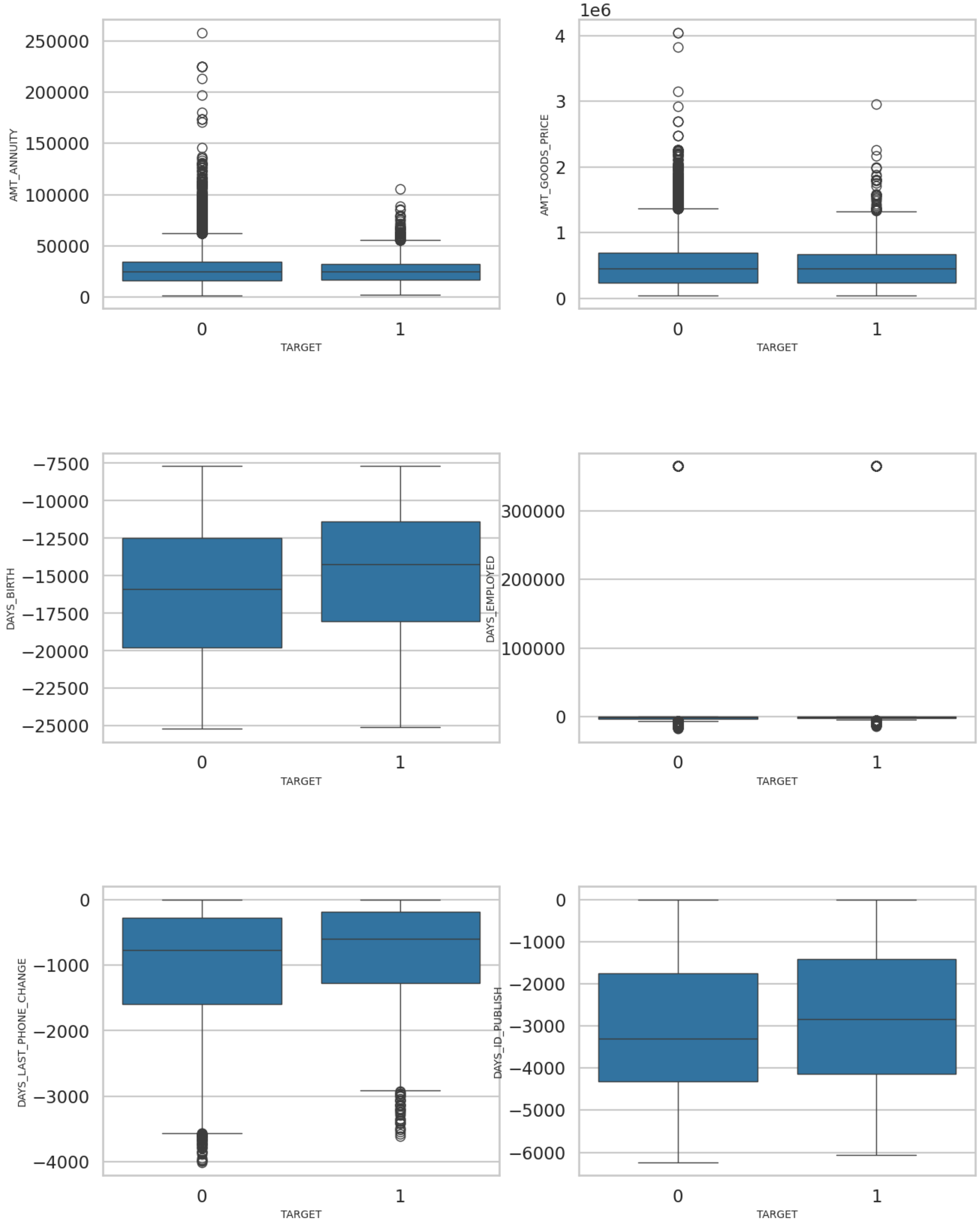
Output:



Categoroical Univariate Analysis in Value scale



Univariate Analysis for continous variable



E. Identify Top Correlations for Different Scenarios: Understanding the correlation between variables and the target variable can provide insights into strong indicators of loan default.

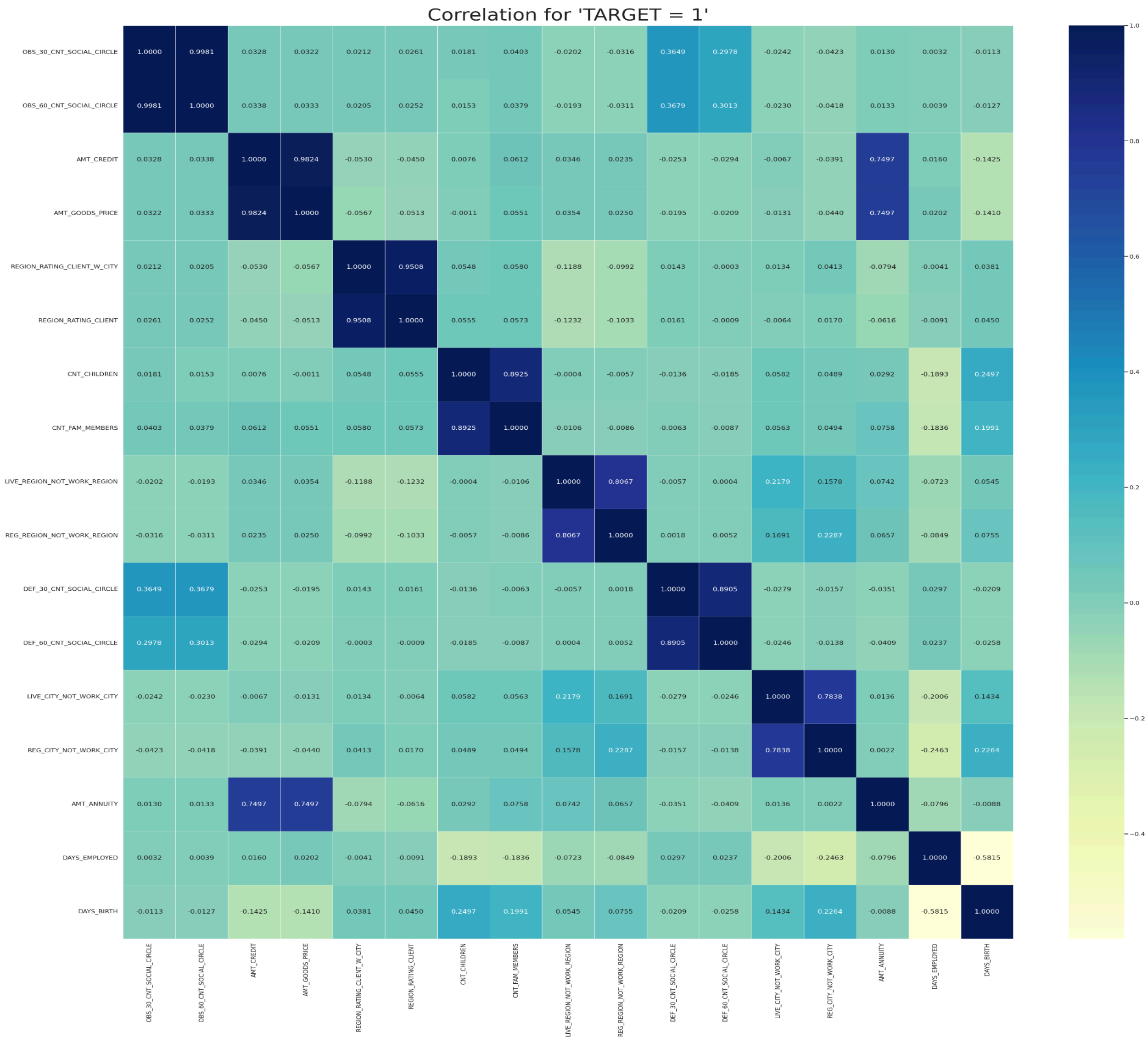
Task: Segment the dataset based on different scenarios (e.g., clients with payment difficulties and all other cases) and identify the top correlations for each segmented data using Excel functions.

Code:

```
numerical_variables = app_data_cleaned.select_dtypes(include = ['float64',
'int64']).columns
numerical_columns = app_data_cleaned[numerical_variables]
payment_difficulty = numerical_columns[numerical_columns['TARGET']==1]
all_other = numerical_columns[numerical_columns['TARGET']==0]

correlations_with_target_1 = payment_difficulty.corr()
top_10_correlations_with_target_1 =
correlations_with_target_1.unstack().sort_values(ascending=False)
top_10_correlations_with_target_1 =
top_10_correlations_with_target_1[top_10_correlations_with_target_1.index.get_level_values(0) != top_10_correlations_with_target_1.index.get_level_values(1)]
top_10_correlations_with_target_1 = top_10_correlations_with_target_1.head(20)


top_10_correlations_with_target_1

correlations_with_target_0 = all_other.corr()
top_10_correlations_with_target_0 =
correlations_with_target_0.unstack().sort_values(ascending=False)
top_10_correlations_with_target_0 =
top_10_correlations_with_target_0[top_10_correlations_with_target_0.index.get_level_values(0) != top_10_correlations_with_target_0.index.get_level_values(1)]
top_10_correlations_with_target_0 = top_10_correlations_with_target_0.head(20)

top_10_correlations_with_target_0
```

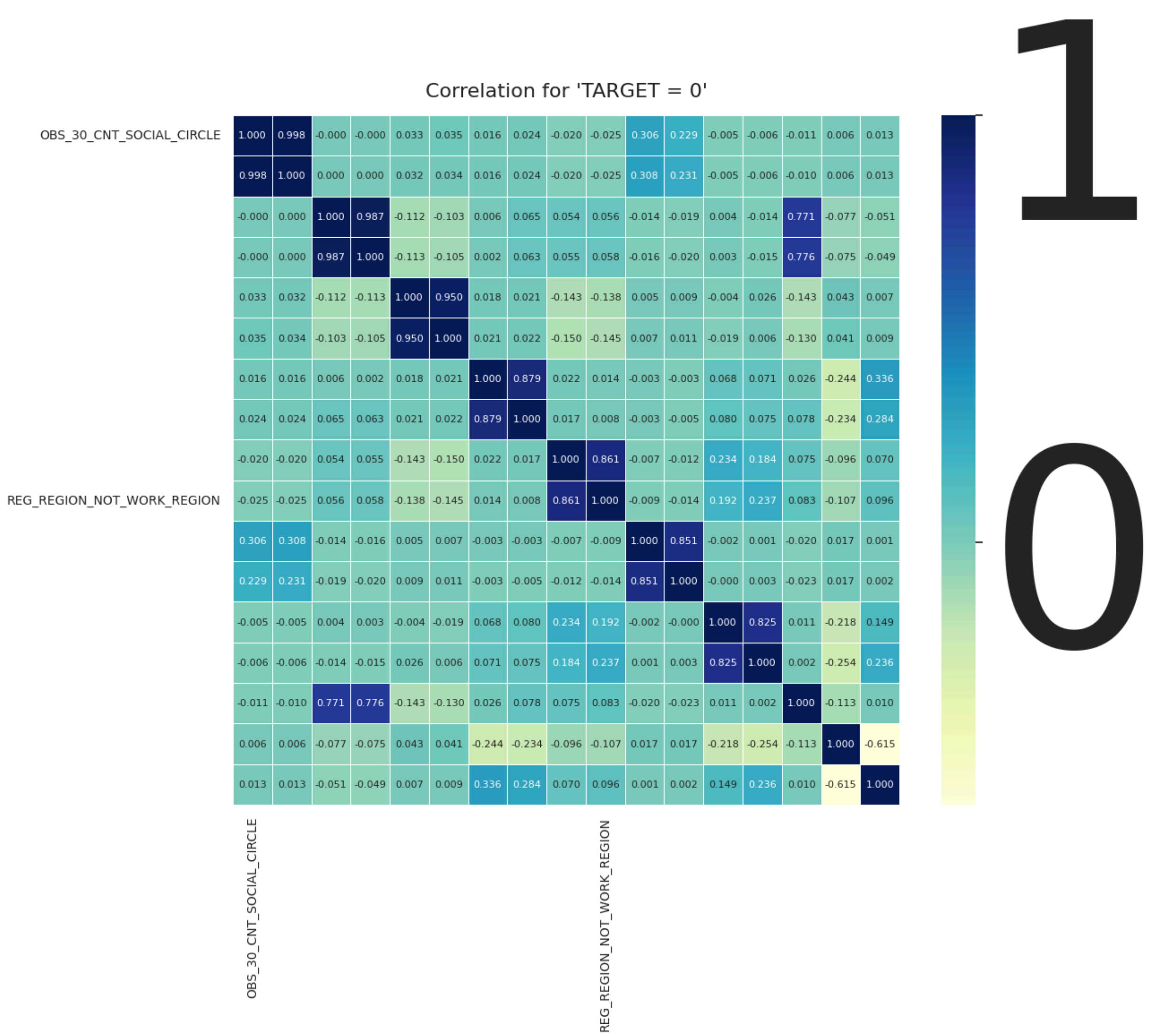
Code for correlation plot ‘Target 1’:

```
columns = ['OBS_30_CNT_SOCIAL_CIRCLE',
'OBS_60_CNT_SOCIAL_CIRCLE',
'AMT_CREDIT',
'AMT_GOODS_PRICE',
'REGION_RATING_CLIENT_W_CITY',
'REGION_RATING_CLIENT',
'CNT_CHILDREN',
'CNT_FAM_MEMBERS',
'LIVE_REGION_NOT_WORK_REGION',
'REG_REGION_NOT_WORK_REGION',
'DEF_30_CNT_SOCIAL_CIRCLE',
'DEF_60_CNT_SOCIAL_CIRCLE',
'LIVE_CITY_NOT_WORK_CITY',
'REG_CITY_NOT_WORK_CITY',
'AMT_ANNUITY',
'DAYS_EMPLOYED',
'DAYS_BIRTH']
selected_columns = payment_difficulty[columns]
correlation_matrix = selected_columns.corr()
plt.figure(figsize=(50, 50))
sns.heatmap(correlation_matrix, annot=True, cmap='YlGnBu', fmt='.4f', linewidths=0.5)
plt.title("Correlation for 'TARGET = 1'",fontsize=50)
plt.show()
```



Code for correlation plot ‘Target 0’:

```
      columns = ['OBS_30_CNT_SOCIAL_CIRCLE',
      'OBS_60_CNT_SOCIAL_CIRCLE',
      'AMT_CREDIT',
      'AMT_GOODS_PRICE',
      'REGION_RATING_CLIENT_W_CITY',
      'REGION_RATING_CLIENT',
      'CNT_CHILDREN',
      'CNT_FAM_MEMBERS',
      'LIVE_REGION_NOT_WORK_REGION',
      'REG_REGION_NOT_WORK_REGION',
      'DEF_30_CNT_SOCIAL_CIRCLE',
      'DEF_60_CNT_SOCIAL_CIRCLE',
      'LIVE_CITY_NOT_WORK_CITY',
      'REG_CITY_NOT_WORK_CITY',
      'AMT_ANNUITY',
      'DAYS_EMPLOYED',
      'DAYS_BIRTH']
selected_columns = all_other[columns]
correlation_matrix = selected_columns.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix,
        annot=True,
        cmap='YlGnBu',
        fmt='.3f',
        linewidths=0.5,
        annot_kws={"size": 8})
plt.title("Correlation for 'TARGET = 0'", fontsize=16)  # Reduce title font size
plt.xticks(fontsize=10)  # Reduce x-axis font size
plt.yticks(fontsize=10)  # Reduce y-axis font size
plt.tight_layout()  # Ensures the layout fits nicely
plt.show()
```



Drive link for colab file TRAINITY TASK 6 Bank Loan Case Study Final Project-2.ipynb