BANK LOAN CASE STUDY

A. PROJECT DESCRIPTION : This project aims understand a cohort of people who have difficulties paying back loans to make better business decisions as well as ensure that capable loan applicants are not rejected. It involves the use of Explanatory Data Analysis (EDA) to analyze patterns in the data and find a solution to challenges faced by a financial company.

B. APPROACH : The approach for this project is defined in the following steps:

i. RESEARCH, it was done to understand the dataset, the values and the questions for proper Exploratory Data Analysis.

ii. Then, the dataset was loaded into the tech-stack that I wanted to use. It was prepared by cleaning it – removing empty and unnecessary values, filtering out rows and columns not important to my analysis.

iii. Then, analysis was done on this cleaned data.

NOTE: APPROACH FOR EVERY INDIVIDUAL TASK IS ALSO DISCUSSED WITHIN THE INSIGHTS SECTION.

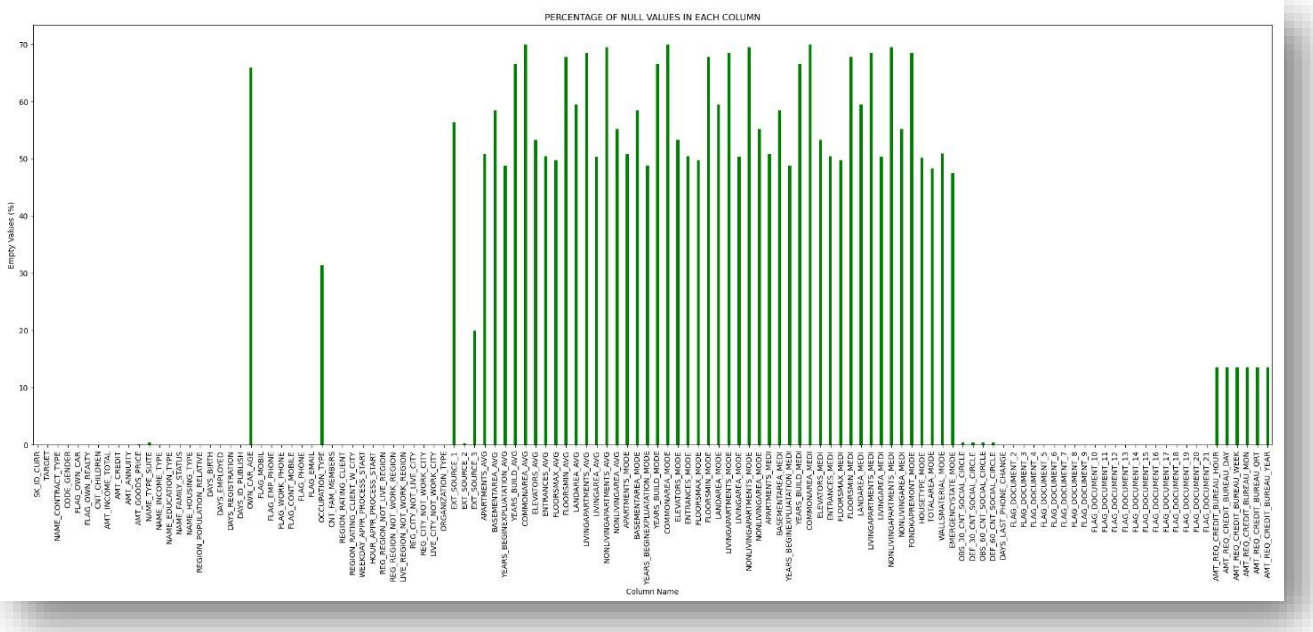
C. TECH-STACK USED : Python Programming Language ( ), Jupyter Notebook (jupyter), Pandas Module ( ), Matplotlib Module ( ), Seaborn Module ( ), MS Excel (X) and MS Word (W) were used to execute this project.

A large part of the analysis was done using Python, in Jupyter Notebook, Python modules - Pandas, Matplotlib and Seaborn were used for the analysis and visualisation. Excel was used to visualise and analyze data. MS Word was used to present all this analysis and insight.

NOTE: THE CLEANED DATASET USED FOR ANALYSIS ALONG WITH THE CODE IS PRESENT AT THE VERY END AFTER THE RESULT SECTION.

D. INSIGHTS:

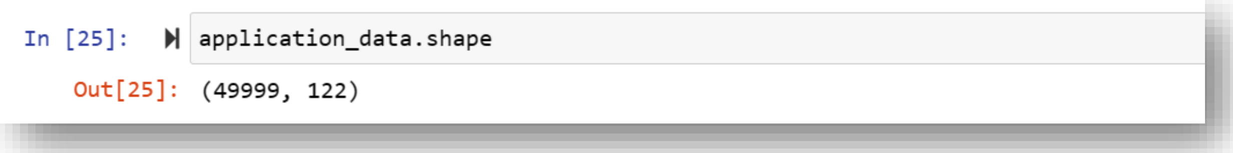
- ***TASK A**: IDENTIFY THE MISSING DATA IN A DATASET AND DECIDE ON AN APPROPRIATE METHOD TO DEAL WITH IT.*



The given Bar Graph presents the percentage of **null values** across all the columns of the **Data Frame** named **application_data**.

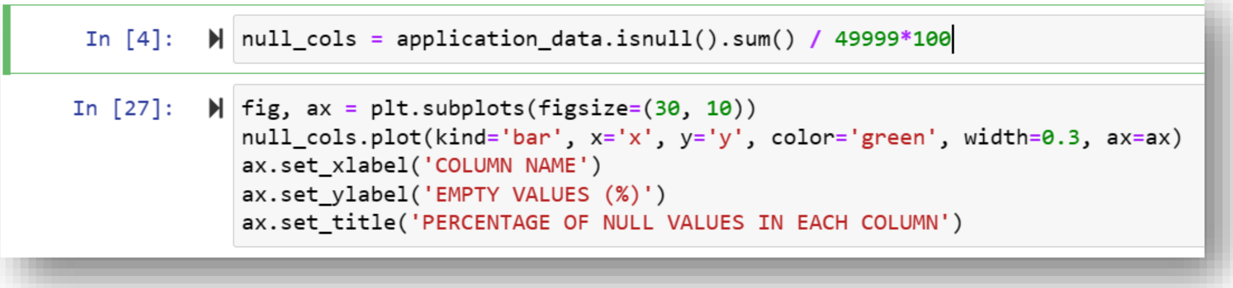
APPROACH:

1) To calculate the percentage of null values, the rows and columns of the **application_data** DataFrame were calculated using the **.shape function.**



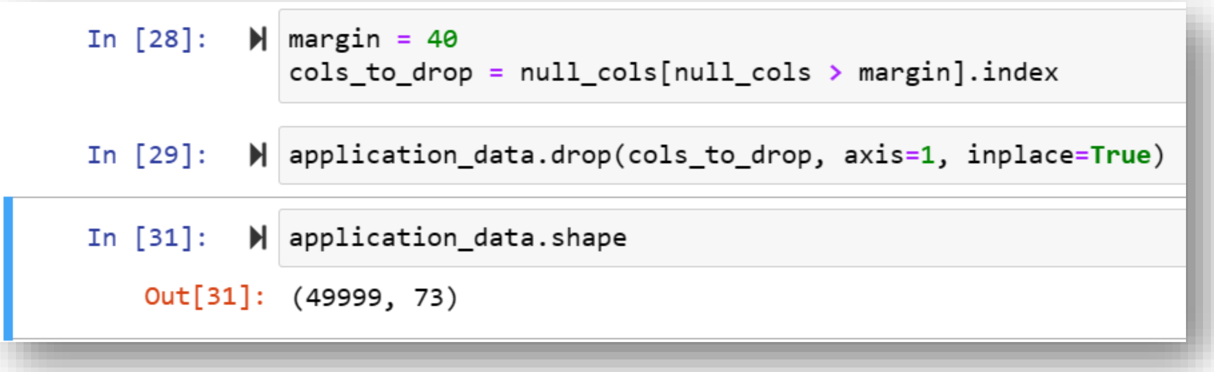
```
In [25]:    application_data.shape

Out[25]: (49999, 122)
```

2) I **used these values to find the percentage of null values** across all columns and **plotted** a Bar graph using the**. plot ()** function.



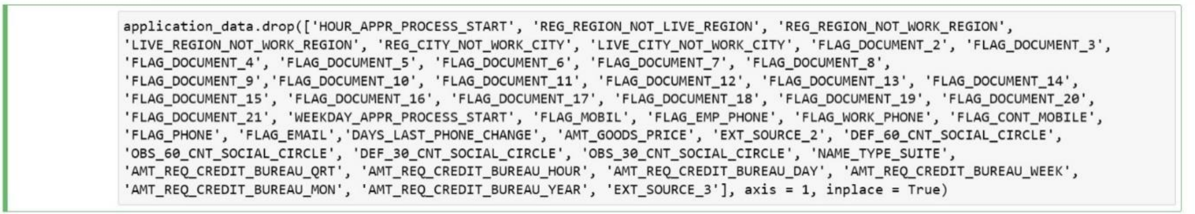
```
In [4]:    null_cols = application_data.isnull().sum() / 49999*100

In [27]:   fig, ax = plt.subplots(figsize=(30, 10))
           null_cols.plot(kind='bar', x='x', y='y', color='green', width=0.3, ax=ax)
           ax.set_xlabel('COLUMN NAME')
           ax.set_ylabel('EMPTY VALUES (%)')
           ax.set_title('PERCENTAGE OF NULL VALUES IN EACH COLUMN')
```

3) Then, columns with **more than 40% of empty values** were then removed.

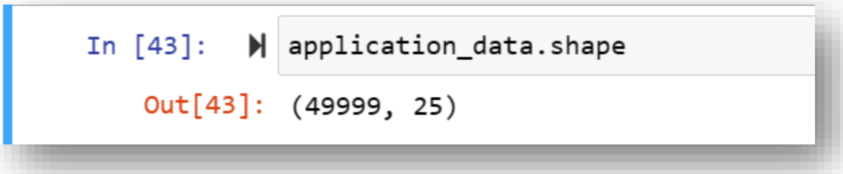


```
In [28]:  margin = 40
          cols_to_drop = null_cols[null_cols > margin].index

In [29]:  application_data.drop(cols_to_drop, axis=1, inplace=True)

In [31]:  application_data.shape

Out[31]:  (49999, 73)
```

4) Then, **columns with factors irrelevant for our analysis** were also **dropped out** for enhanced viewing and better analysis bringing the number of columns down to 25. The criteria of which columns to drop were decided by looking at the **columns_description** dataset.

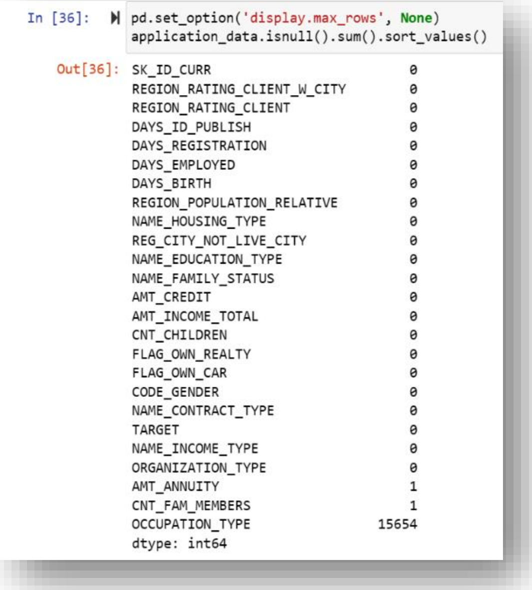


```
application_data.drop(['HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3',
'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', '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', 'WEEKDAY_APPR_PROCESS_START', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE',
'FLAG_PHONE', 'FLAG_EMAIL','DAYS_LAST_PHONE_CHANGE', 'AMT_GOODS_PRICE', 'EXT_SOURCE_2', 'DEF_60_CNT_SOCIAL_CIRCLE',
'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_30_CNT_SOCIAL_CIRCLE', 'NAME_TYPE_SUITE',
'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_YEAR', 'EXT_SOURCE_3'], axis = 1, inplace = True)
```

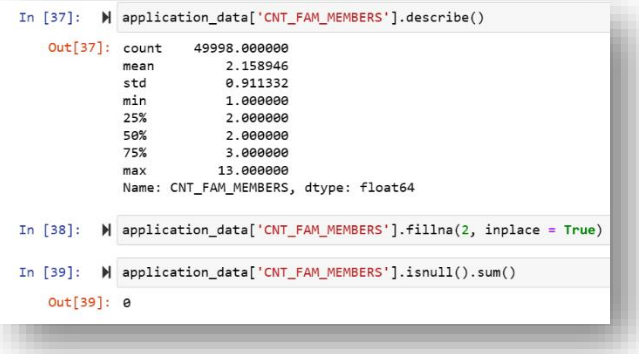


```
In [43]:  application_data.shape

Out[43]:  (49999, 25)
```

5) After filtering out these columns, **null values for the left-over** columns were calculated. **Those null values were dealt with** by using the **.describe()** function.



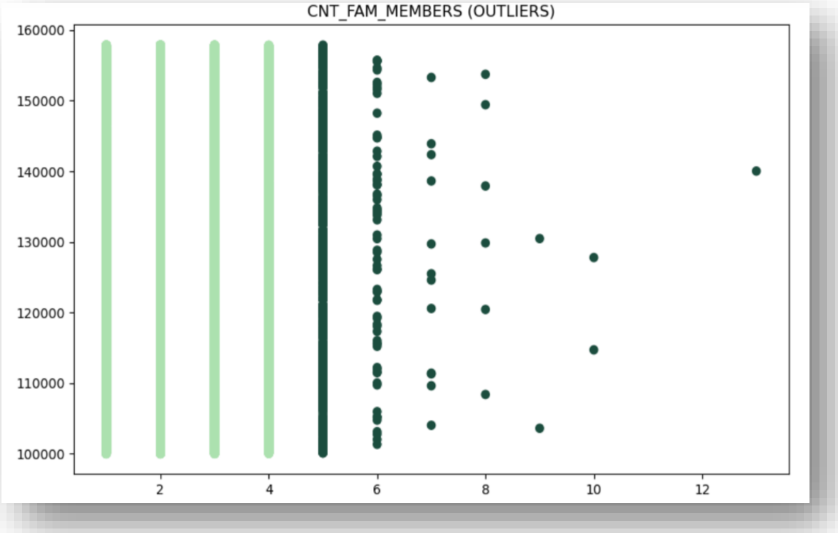
```
In [36]:  pd.set_option('display.max_rows', None)
          application_data.isnull().sum().sort_values()

Out[36]:  SK_ID_CURR                        0
          REGION_RATING_CLIENT_W_CITY       0
          REGION_RATING_CLIENT              0
          DAYS_ID_PUBLISH                   0
          DAYS_REGISTRATION                 0
          DAYS_EMPLOYED                     0
          DAYS_BIRTH                        0
          REGION_POPULATION_RELATIVE        0
          NAME_HOUSING_TYPE                 0
          REG_CITY_NOT_LIVE_CITY            0
          NAME_EDUCATION_TYPE               0
          NAME_FAMILY_STATUS                0
          AMT_CREDIT                        0
          AMT_INCOME_TOTAL                  0
          CNT_CHILDREN                      0
          FLAG_OWN_REALTY                   0
          FLAG_OWN_CAR                      0
          CODE_GENDER                       0
          NAME_CONTRACT_TYPE                0
          TARGET                            0
          NAME_INCOME_TYPE                  0
          ORGANIZATION_TYPE                 0
          AMT_ANNUITY                       1
          CNT_FAM_MEMBERS                   1
          OCCUPATION_TYPE               15654
          dtype: int64
```



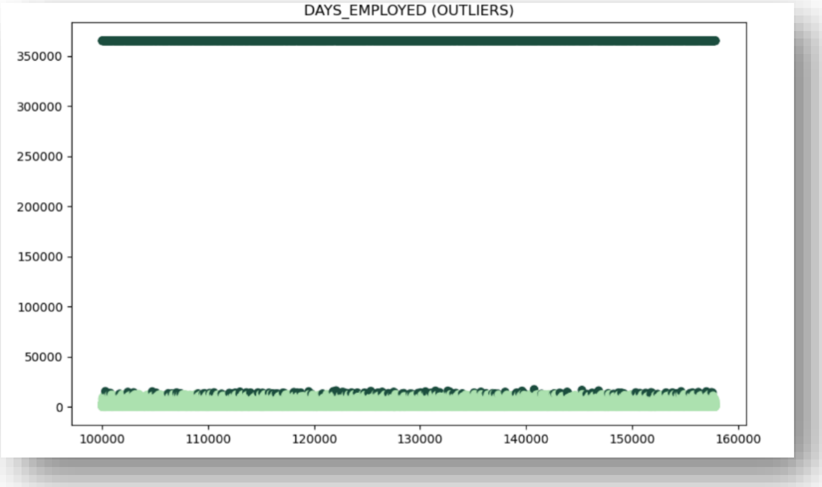
```
In [37]:  application_data['CNT_FAM_MEMBERS'].describe()

Out[37]:  count    49998.000000
          mean         2.158946
          std          0.911332
          min          1.000000
          25%          2.000000
          50%          2.000000
          75%          3.000000
          max         13.000000
          Name: CNT_FAM_MEMBERS, dtype: float64

In [38]:  application_data['CNT_FAM_MEMBERS'].fillna(2, inplace = True)

In [39]:  application_data['CNT_FAM_MEMBERS'].isnull().sum()

Out[39]:  0
```

- ***TASK B**: DETECT AND IDENTIFY OUTLIERS USING STATISTICAL FUCNTIONS AND FEATURES, FOCUSING ON NUMERICAL VALUES.*

FOLLOWING ARE THE OUTLIERS DETECTED IN THE DATASET:

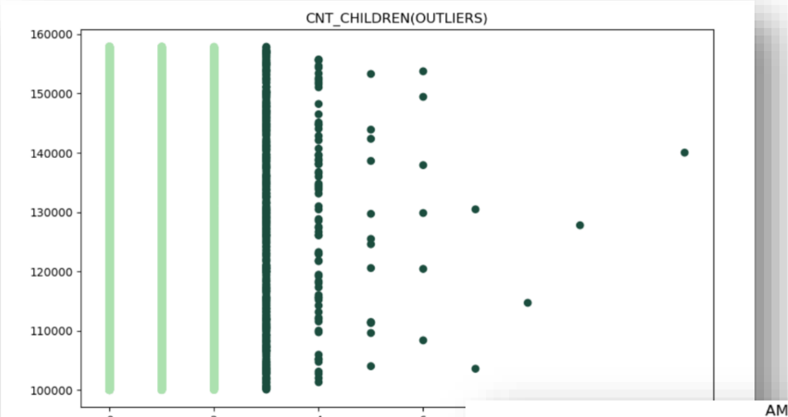


This Scatter Plot represents the outliers for the column **'CNT_FAM_MEMBERS'**, which represents the **number of family members** an applicant has.

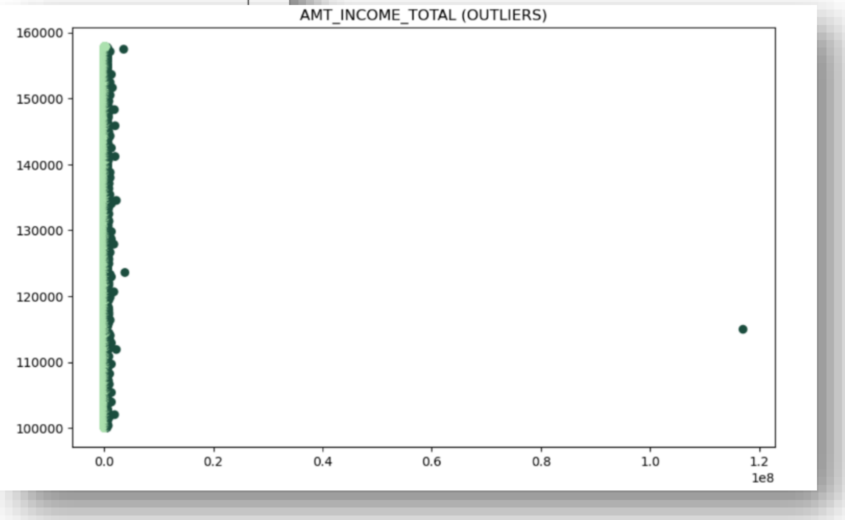


This scatter plot illustrates the outliers in the **'DAYS_EMPLOYED'** column, which denotes the **number of days** an applicant has been **employed.**

This scatter plot showcases the outliers in the



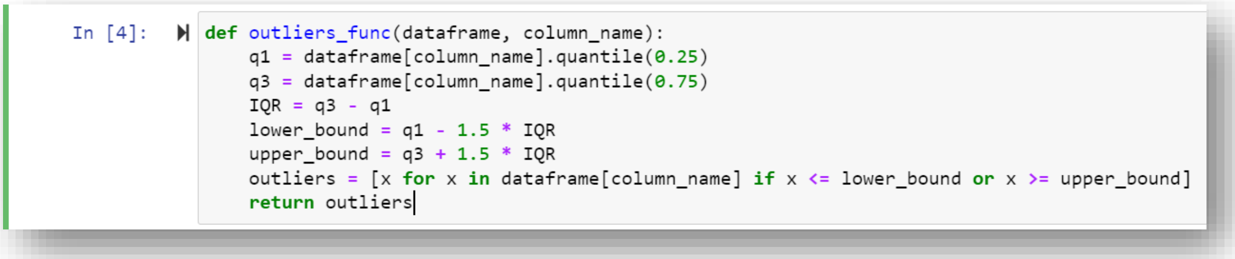
‘CNT_CHILDREN’ column, signifying the **number of children** the applicant has under their care.



This scatter plot depicts the outliers in the ‘AMT_INCOME_TOTAL’ column, which corresponds to the **applicant’s total income.**

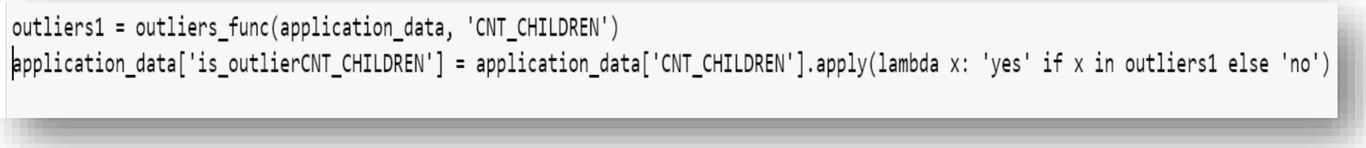
APPROACH:

1) I first created **a function** that gives me the **outliers** in a column using the **IQR method**.



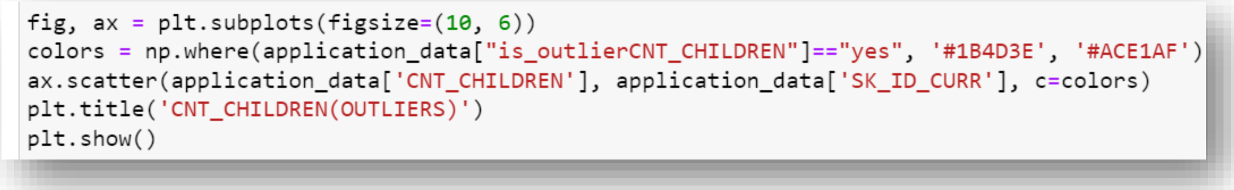
```
In [4]:  def outliers_func(dataframe, column_name):
             q1 = dataframe[column_name].quantile(0.25)
             q3 = dataframe[column_name].quantile(0.75)
             IQR = q3 - q1
             lower_bound = q1 - 1.5 * IQR
             upper_bound = q3 + 1.5 * IQR
             outliers = [x for x in dataframe[column_name] if x <= lower_bound or x >= upper_bound]
             return outliers
```

2) This function was **applied to the columns** across the table in the given manner, **creating another column** with the value **‘yes’ for the data being an outlier** and the value ‘**no’ for the data not being an outlier.**



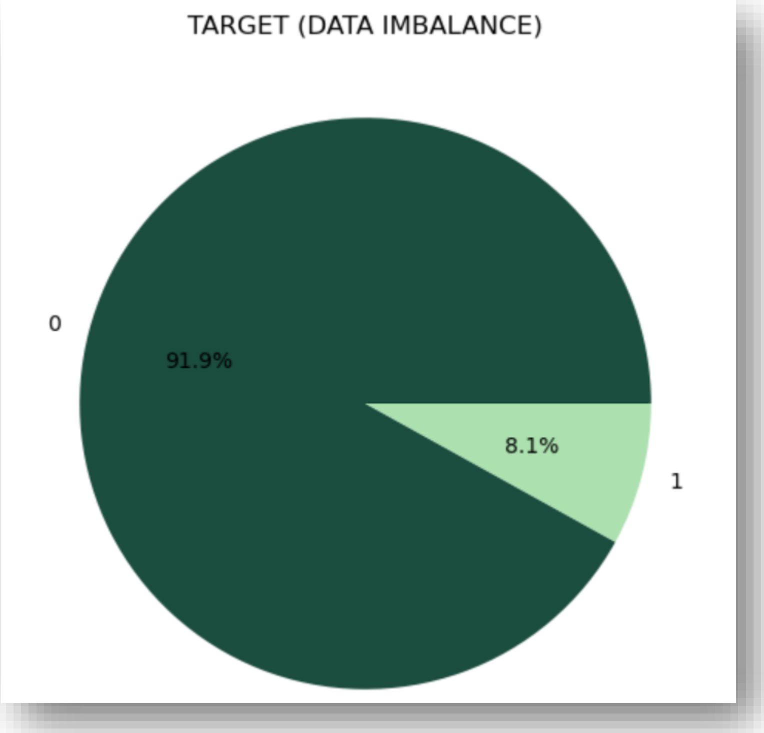
```
outliers1 = outliers_func(application_data, 'CNT_CHILDREN')
application_data['is_outlierCNT_CHILDREN'] = application_data['CNT_CHILDREN'].apply(lambda x: 'yes' if x in outliers1 else 'no')
```

3) This column's values were then used to create the scatter plot for the visualization of the outliers.



```
fig, ax = plt.subplots(figsize=(10, 6))
colors = np.where(application_data["is_outlierCNT_CHILDREN"]=="yes", '#1B4D3E', '#ACE1AF')
ax.scatter(application_data['CNT_CHILDREN'], application_data['SK_ID_CURR'], c=colors)
plt.title('CNT_CHILDREN(OUTLIERS)')
plt.show()
```

- ***TASK C**: DETERMINE IF THERE IS DATA IMBALANCE IN THE LOAN APPLICATION DATASET AND CALCULATE THE RATIO OF DATA IMBALANCE USING EXCEL FUNCTIONS.*

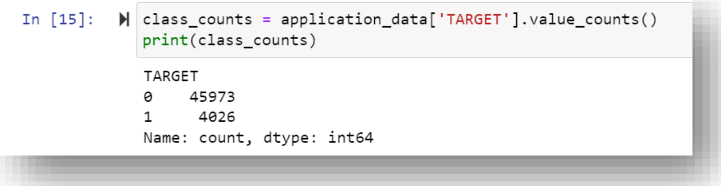


The given pie chart **illustrates the data imbalance** present in **'TARGET'** column. It is a column where **value 1** portrays that **the applicant has payment difficulties** and **value 0 portrays that the applicant does not have payment difficulties**.

The data imbalance presented in the target value shows **that the majority (91.9%)** of the applicants **do not exhibit payment difficulties**.

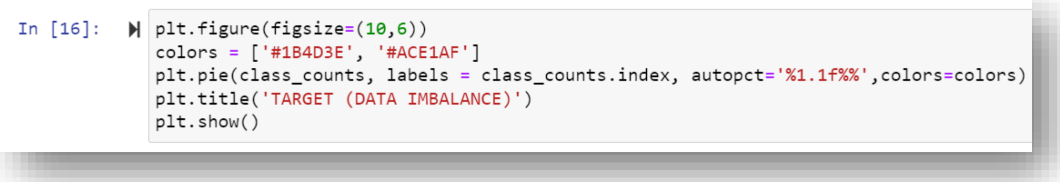
APPROACH:

1) First, **a count** of the number of values in the **'TARGET'** variable was done.



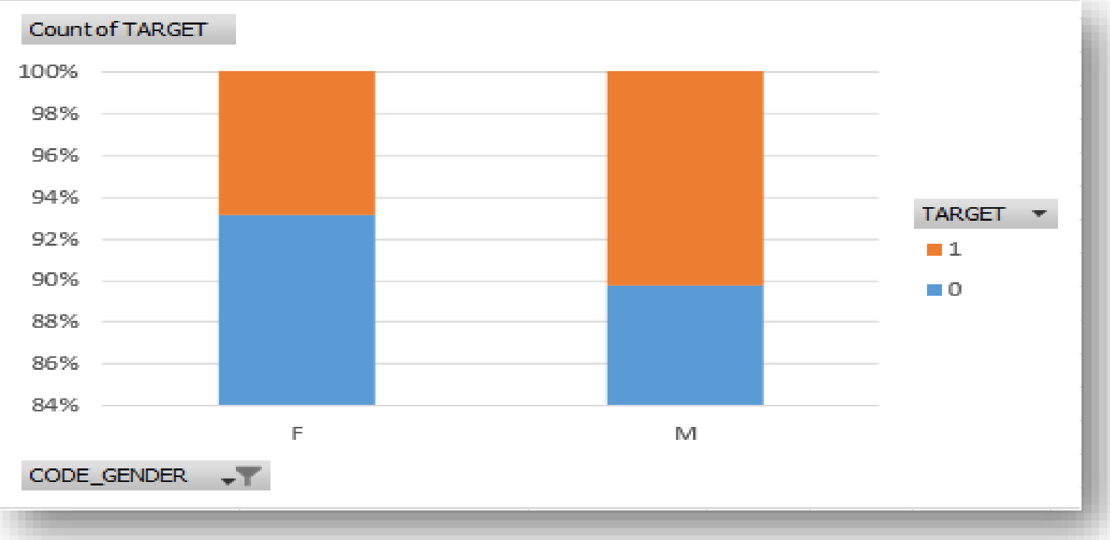
```
In [15]:  class_counts = application_data['TARGET'].value_counts()
          print(class_counts)

          TARGET
          0    45973
          1     4026
          Name: count, dtype: int64
```

2) The output was then visualized into a Pie Chart.

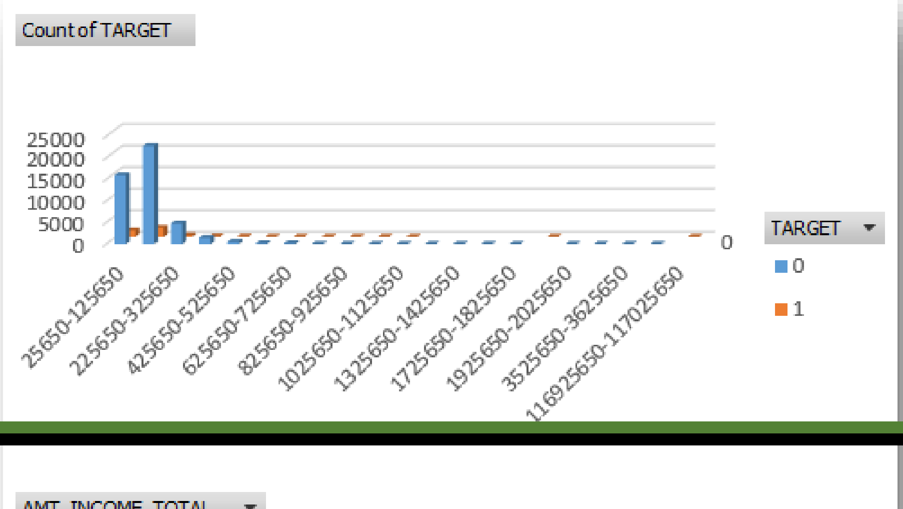


```
In [16]:  plt.figure(figsize=(10,6))
          colors = ['#1B4D3E', '#ACE1AF']
          plt.pie(class_counts, labels = class_counts.index, autopct='%1.1f%%',colors=colors)
          plt.title('TARGET (DATA IMBALANCE)')
          plt.show()
```

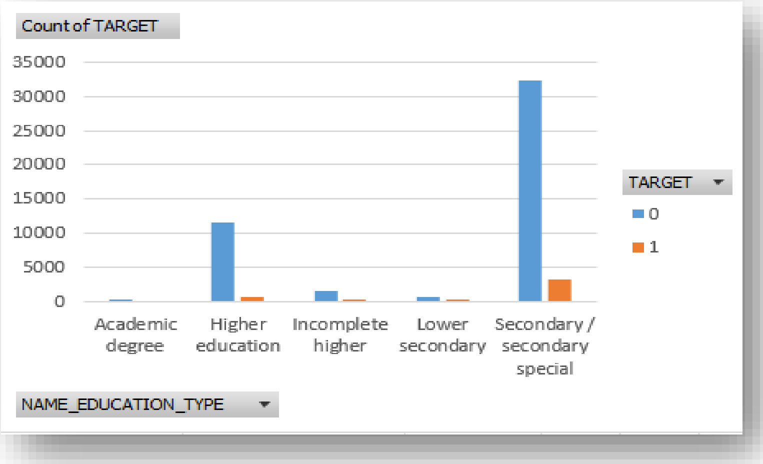
- ***TASK D**: PERFORM UNIVARIATE ANALYSIS TO UNDERSTAND THE DISTRIBUTION OF INDIVIDUAL VARIABLES, SEGEMNTED UNIVARIATE ANALYSIS TO COMPARE VARIABLE DISTRIBUTIONS FOR DIFFERENT SCENARIOS, AND BIVARIATE ANALYSIS TO EXPLORE RELATIONSHIPS BETWEEN VARIABLES AND THE TARGET VARIABLES.*



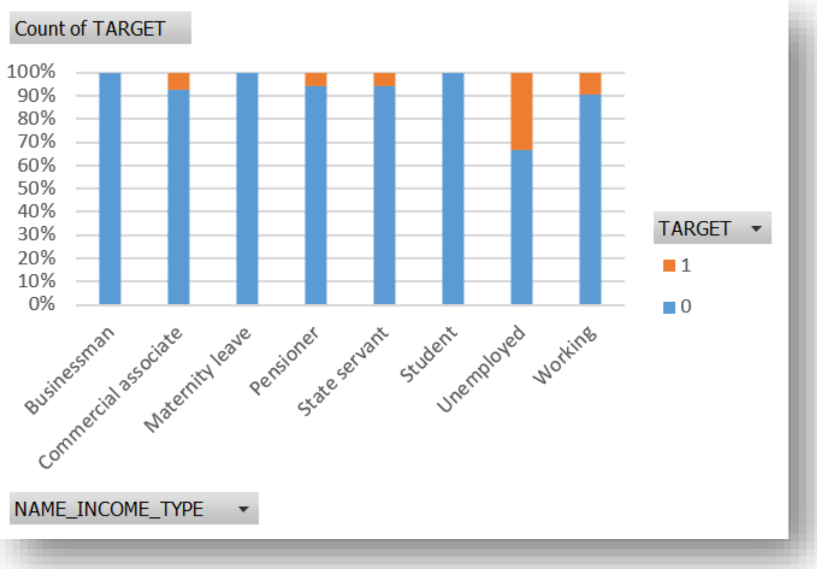
The given Pivot Bar Chart illustrates **payment difficulties** across **male and female applicants**.



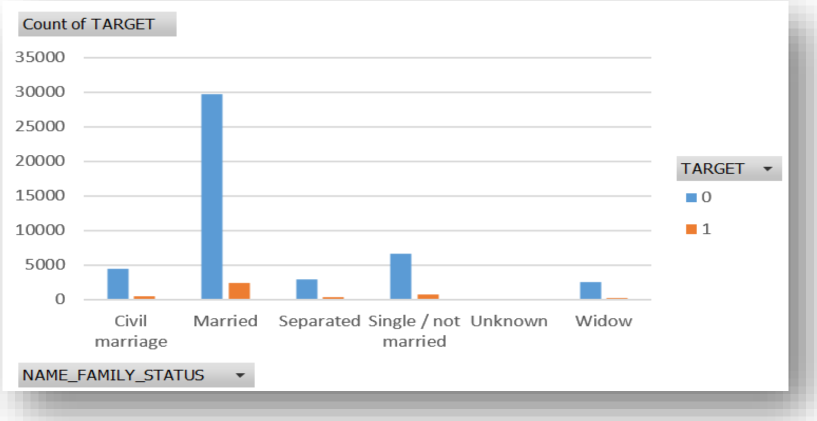
This Pivot Bar Chart explores the **payment difficulties across different income groups**. We can see that **applicants in lower income groups exhibit more payment difficulties**.



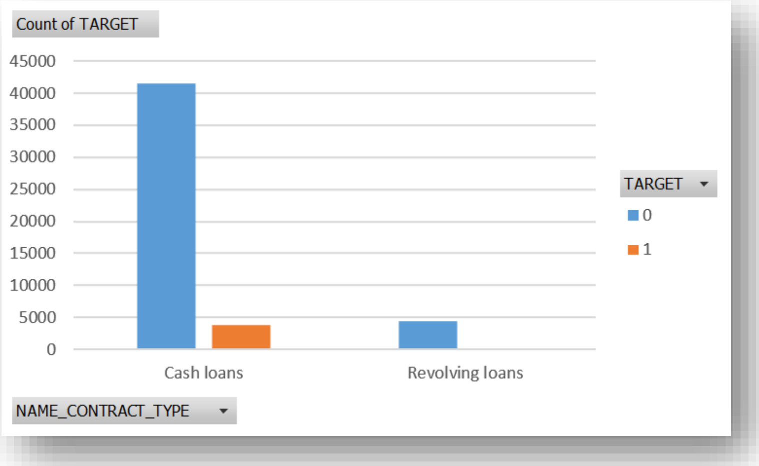
This Pivot Bar Chart shows **the education background** of applicant along with their **payment behavior.**



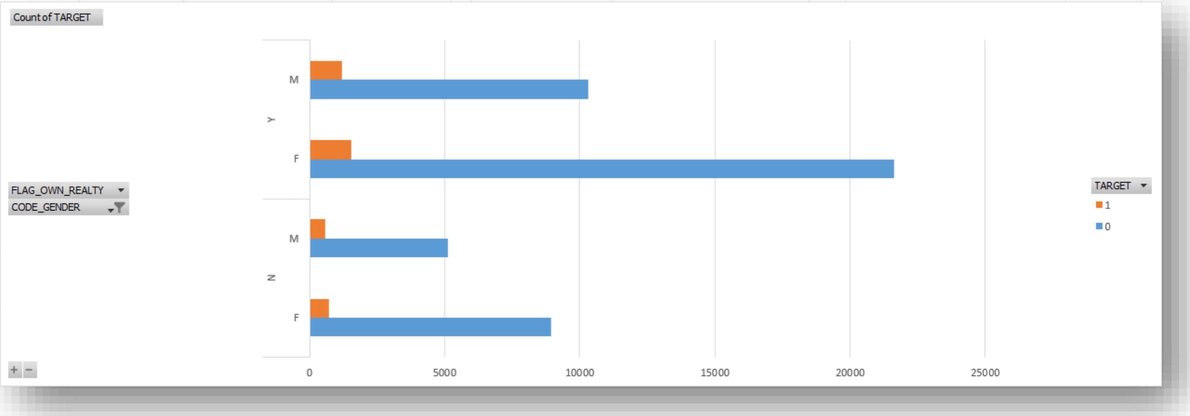
This Pivot Bar Chart illustrates **applicants with different income types.** Through this, we observe that **unemployed applicants exhibit the most payment difficulties**.



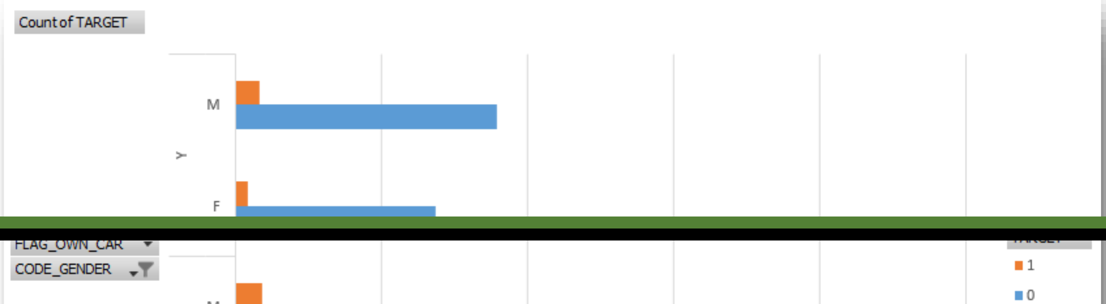
This Pivot Bar Chart expresses **payment behaviors** across applicants with different **Family Status'.**



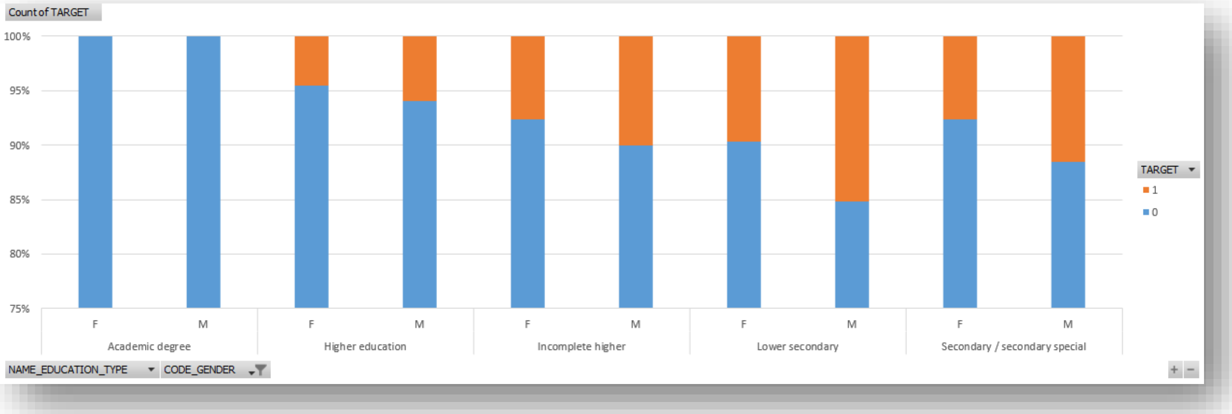
This Pivot Bar Chart exhibits the **payment behaviors of applicants with different contract type**. It can be observed that applicants with **Cash Loans show more payment difficulties** than applicants with Revolving loans.



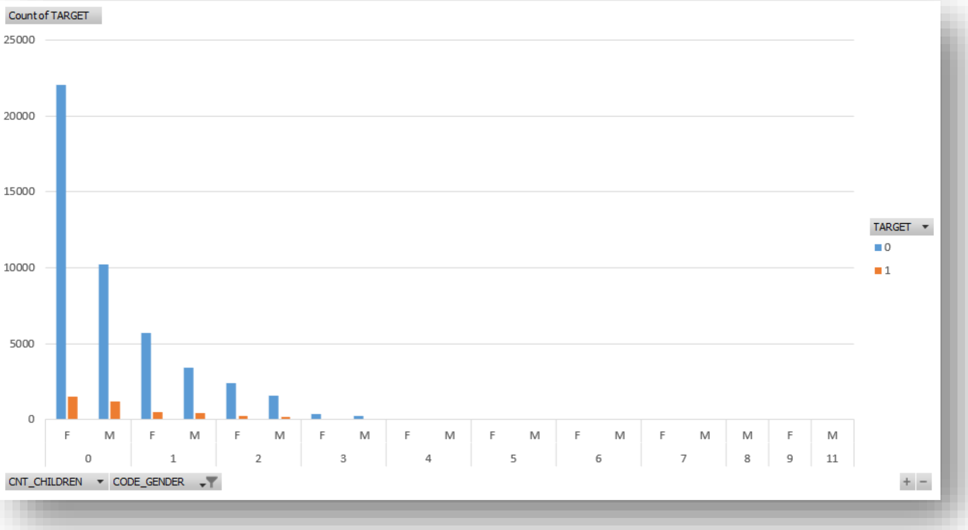
This Pivot Bar Chart shows the **Payment behaviors of Male and Female applicants** with respect to **owning realty.**



This Pivot Bar Chart shows the **Payment behaviors of Male and Female applicants** with respect to **owning a car.**



This Pivot Bar Chart shows **the Payment behaviors of Male and Female** applicants with respect to **their educational background**.

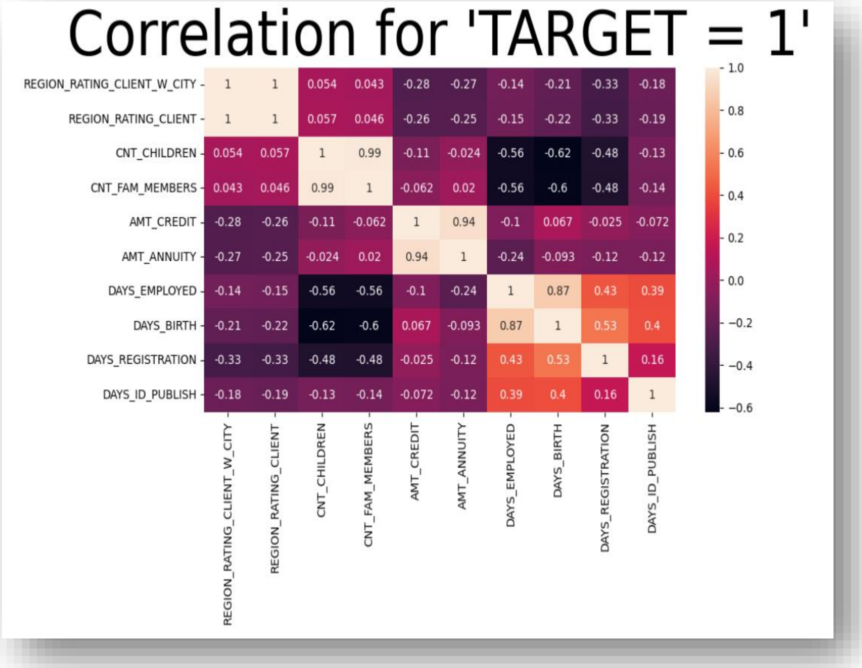


This Pivot Bar Chart shows the **Payment behaviors of Male and Female applicants** with respect **to the number of children under their care**.

APPROACH:

This task was done in **MS Excel** using **Pivot Tables.**

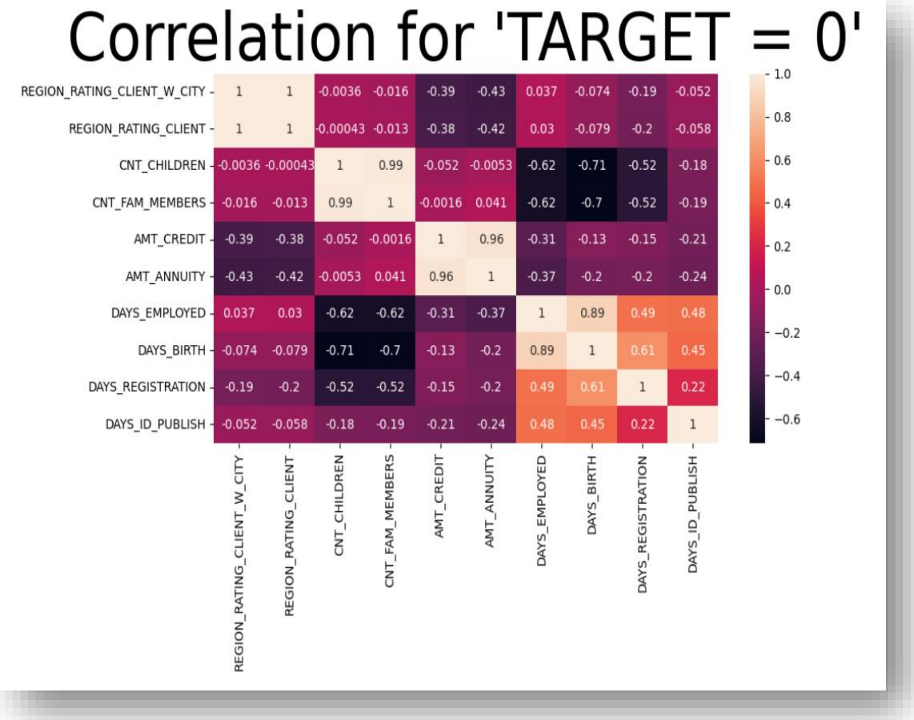
- ***TASK E:*** *SEGMENT THE DATASET BASED ON DIFFERENT SCENARIOS (E.G. CLIENTS WITH PAYMENT DIFFICULTIES AND ALL OTHER CASES) AND IDENTIFY THE TOP CORRELATIONS FOR EACH SEGMENT.*


Correlation for 'TARGET = 1'

	REGION_RATING_CLIENT_W_CITY	REGION_RATING_CLIENT	CNT_CHILDREN	CNT_FAM_MEMBERS	AMT_CREDIT	AMT_ANNUITY	DAYS_EMPLOYED	DAYS_BIRTH	DAYS_REGISTRATION	DAYS_ID_PUBLISH
REGION_RATING_CLIENT_W_CITY	1	1	0.054	0.043	-0.28	-0.27	-0.14	-0.21	-0.33	-0.18
REGION_RATING_CLIENT	1	1	0.057	0.046	-0.26	-0.25	-0.15	-0.22	-0.33	-0.19
CNT_CHILDREN	0.054	0.057	1	0.99	-0.11	-0.024	-0.56	-0.62	-0.48	-0.13
CNT_FAM_MEMBERS	0.043	0.046	0.99	1	-0.062	0.02	-0.56	-0.6	-0.48	-0.14
AMT_CREDIT	-0.28	-0.26	-0.11	-0.062	1	0.94	-0.1	0.067	-0.025	-0.072
AMT_ANNUITY	-0.27	-0.25	-0.024	0.02	0.94	1	-0.24	-0.093	-0.12	-0.12
DAYS_EMPLOYED	-0.14	-0.15	-0.56	-0.56	-0.1	-0.24	1	0.87	0.43	0.39
DAYS_BIRTH	-0.21	-0.22	-0.62	-0.6	0.067	-0.093	0.87	1	0.53	0.4
DAYS_REGISTRATION	-0.33	-0.33	-0.48	-0.48	-0.025	-0.12	0.43	0.53	1	0.16
DAYS_ID_PUBLISH	-0.18	-0.19	-0.13	-0.14	-0.072	-0.12	0.39	0.4	0.16	1

This **correlation matrix** presents the **top correlations for Target value 1**, i.e. correlations of top factors of applicants with payment difficulties.

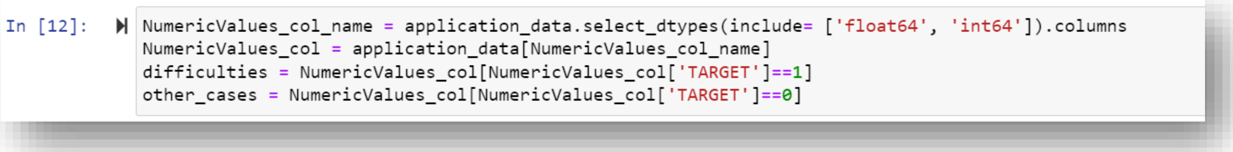
This **correlation matrix** presents **the top correlations for Target value 0,** i.e. correlations of top factors of every other case.



	REGION_RATING_CLIENT_W_CITY	REGION_RATING_CLIENT	CNT_CHILDREN	CNT_FAM_MEMBERS	AMT_CREDIT	AMT_ANNUITY	DAYS_EMPLOYED	DAYS_BIRTH	DAYS_REGISTRATION	DAYS_ID_PUBLISH
REGION_RATING_CLIENT_W_CITY	1	1	-0.0036	-0.016	-0.39	-0.43	0.037	-0.074	-0.19	-0.052
REGION_RATING_CLIENT	1	1	-0.00043	-0.013	-0.38	-0.42	0.03	-0.079	-0.2	-0.058
CNT_CHILDREN	-0.0036	-0.00043	1	0.99	-0.052	-0.0053	-0.62	-0.71	-0.52	-0.18
CNT_FAM_MEMBERS	-0.016	-0.013	0.99	1	-0.0016	0.041	-0.62	-0.7	-0.52	-0.19
AMT_CREDIT	-0.39	-0.38	-0.052	-0.0016	1	0.96	-0.31	-0.13	-0.15	-0.21
AMT_ANNUITY	-0.43	-0.42	-0.0053	0.041	0.96	1	-0.37	-0.2	-0.2	-0.24
DAYS_EMPLOYED	0.037	0.03	-0.62	-0.62	-0.31	-0.37	1	0.89	0.49	0.48
DAYS_BIRTH	-0.074	-0.079	-0.71	-0.7	-0.13	-0.2	0.89	1	0.61	0.45
DAYS_REGISTRATION	-0.19	-0.2	-0.52	-0.52	-0.15	-0.2	0.49	0.61	1	0.22
DAYS_ID_PUBLISH	-0.052	-0.058	-0.18	-0.19	-0.21	-0.24	0.48	0.45	0.22	1

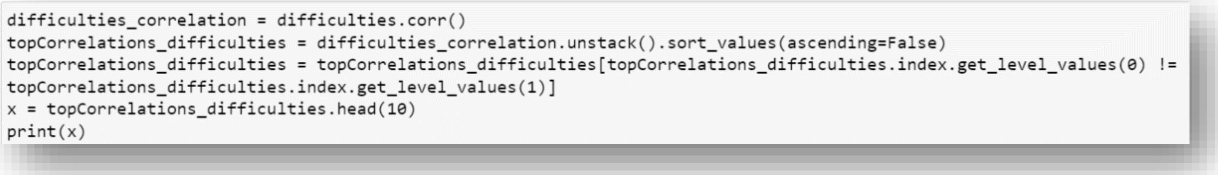
APPROACH:

1) First, **another table** was created storing **only 'int64'** and **'float64'** values. This was **further filtered to create two tables**, one **for Target Value 1** (applicants with payment difficulties) and the other **for Target Value 0** (every other case).



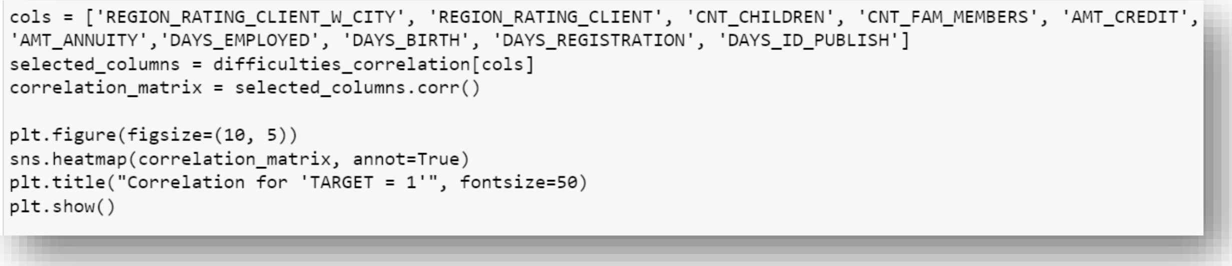
```
In [12]:  NumericValues_col_name = application_data.select_dtypes(include= ['float64', 'int64']).columns
          NumericValues_col = application_data[NumericValues_col_name]
          difficulties = NumericValues_col[NumericValues_col['TARGET']==1]
          other_cases = NumericValues_col[NumericValues_col['TARGET']==0]
```

2) These correlations were then **sorted to give the top correlations**. Values that showed **something related to itself** were **filtered out** as they might skew the analysis. Then, these **columns were printed** out so that we can **pick columns for correlation**.



```
difficulties_correlation = difficulties.corr()
topCorrelations_difficulties = difficulties_correlation.unstack().sort_values(ascending=False)
topCorrelations_difficulties = topCorrelations_difficulties[topCorrelations_difficulties.index.get_level_values(0) !=
topCorrelations_difficulties.index.get_level_values(1)]
x = topCorrelations_difficulties.head(10)
print(x)
```

3) After **picking columns to avoid repetition of correlations**, they were **filtered out from the original correlation index** that was obtained. This filtered out index was then **visualized as a heatmap** to visualize the correlation matrix.



```
cols = ['REGION_RATING_CLIENT_W_CITY', 'REGION_RATING_CLIENT', 'CNT_CHILDREN', 'CNT_FAM_MEMBERS', 'AMT_CREDIT',
'AMT_ANNUITY','DAYS_EMPLOYED', 'DAYS_BIRTH', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH']
selected_columns = difficulties_correlation[cols]
correlation_matrix = selected_columns.corr()

plt.figure(figsize=(10, 5))
sns.heatmap(correlation_matrix, annot=True)
plt.title("Correlation for 'TARGET = 1'", fontsize=50)
plt.show()
```

E) RESULT : Working on this project was a challenge driven experience for me. I completed it having learned a new tech-stack and understanding the technicalities of analysis in the financial sector.

It was a project full of effortful tasks that demanded my growth. My skills as a Data Analyst improved significantly as a result of the valuable experience I gained from this project.

THE CODE (PDF):

TASK A : https://drive.google.com/file/d/1LqSWYe3NWwatrVTXtXMbttJ-wqpXaYQs/view?usp=sharing

ALL OTHER TASKS : https://drive.google.com/file/d/1gKEeCVczYpSrdYitecb8gxTzsiV331KC/view?usp=drive_link