Bank Loan Case Study

Project Description: This project is about analyzing the bank loan dataset to identify customers that will potentially have difficulty repaying the installments of the loan on time (or in other words, loan defaulters).

Approach: Let's first download the dataset into our local repository and clean the dataset. Next, we'll perform some basic exploratory data analysis (EDA). Finally, we'll answer some of the questions presented to us, one by one, either graphically or in table format.

Tech-Stack Used: Google Colab (Jupyter Notebook), Python, Microsoft Excel

Guiding Questions and Insights: This is where we'll summarize the insights and knowledge gained during the project and discuss key findings and any meaningful trends or patterns discovered.

Google Colab Workbook:
https://colab.research.google.com/drive/1qMkL0y2HwWhL3M67oUtBIjLgyTI8sJk3?usp=sharing

First, import all the necessary libraries and get the data ready

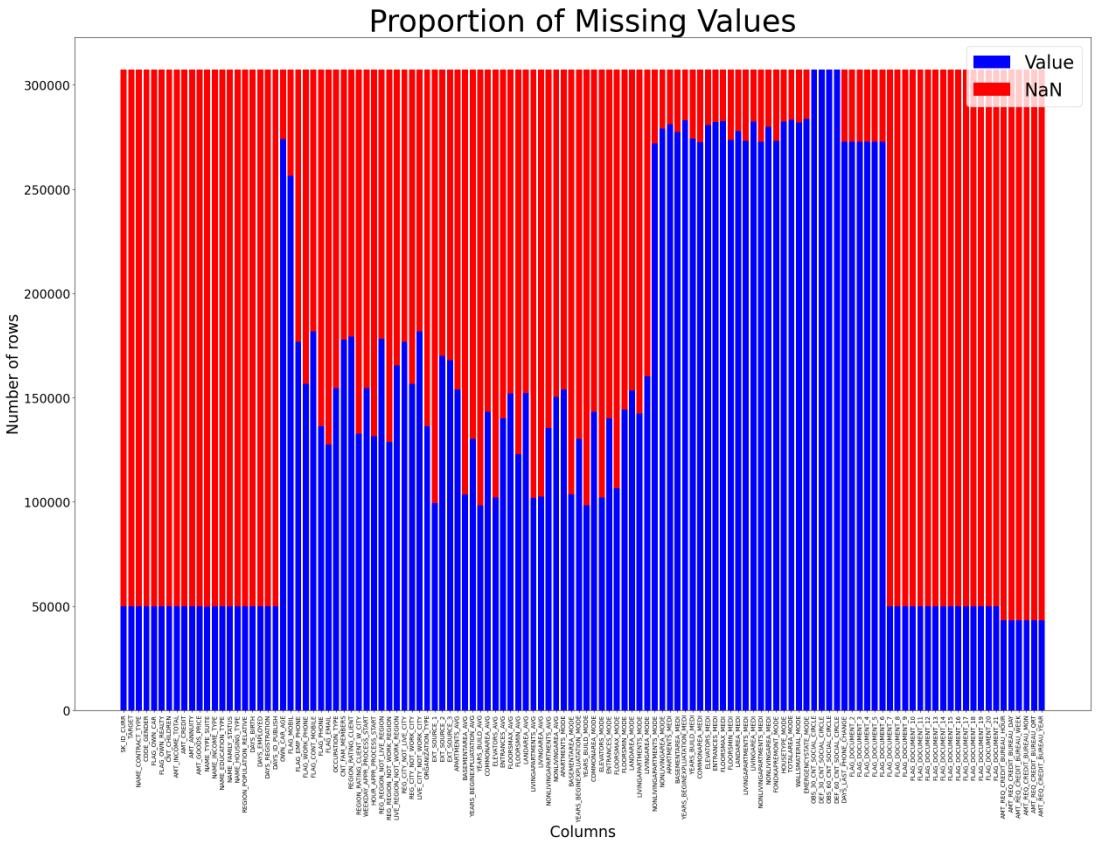
```
# Import all necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set options
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
# Import the dataset
data = pd.read_csv("drive/MyDrive/project6/application_data.csv",
                   low_memory=False)
data
```

Observe that there are 307511 rows and 122 columns. However, we'll see later that we won't need all these columns and will have to eliminate some rows before we can begin our analysis.

A. Identify Missing Data and Deal with it Appropriately

I started off by counting the number of missing values in each column and made a function to plot it on a graph to get a better understanding of the data we are working with.



Observe that the 'TARGET' column has a large number (257512) of missing values. Being a label, this column is **dependent** on all other variables, and hence randomly imputing it would lead to biases in our analysis.

Therefore, we choose to remove it. We can hope that some of the NaN values in the other columns also get eliminated when we do this.

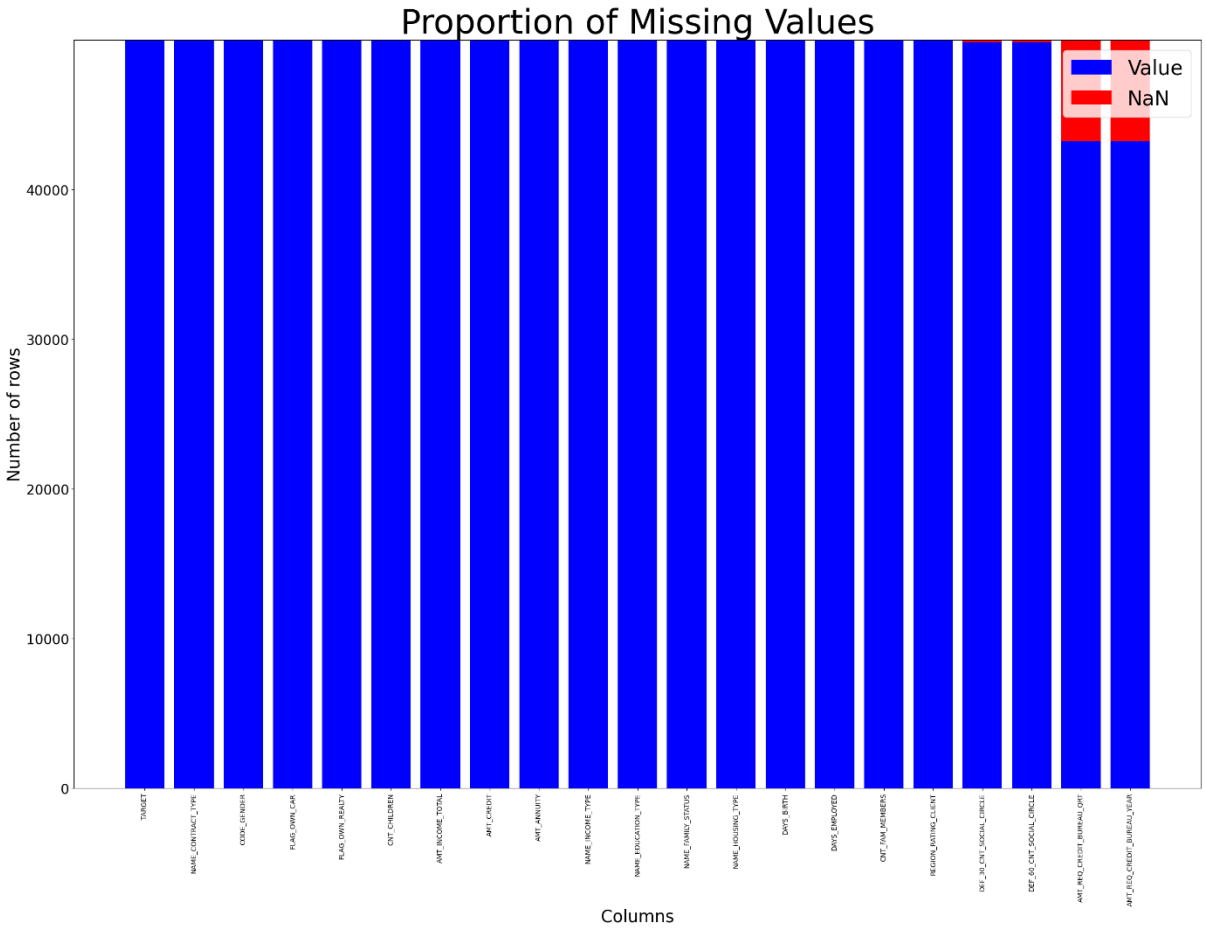
```
# Dropping rows with a NaN value in TARGET column

data = data.dropna(subset="TARGET")
```

This process removed around 250000 records but there is still a huge amount of missing data. We are going to drop some columns which aren't required for our analysis, which may bring down the NaN's.

Once we are done with that, if there is still some missing values in the independent variables (or features), we'll have to use standard imputation techniques such as average or median.

```
data.drop(drop_columns, axis=1, inplace=True)
```

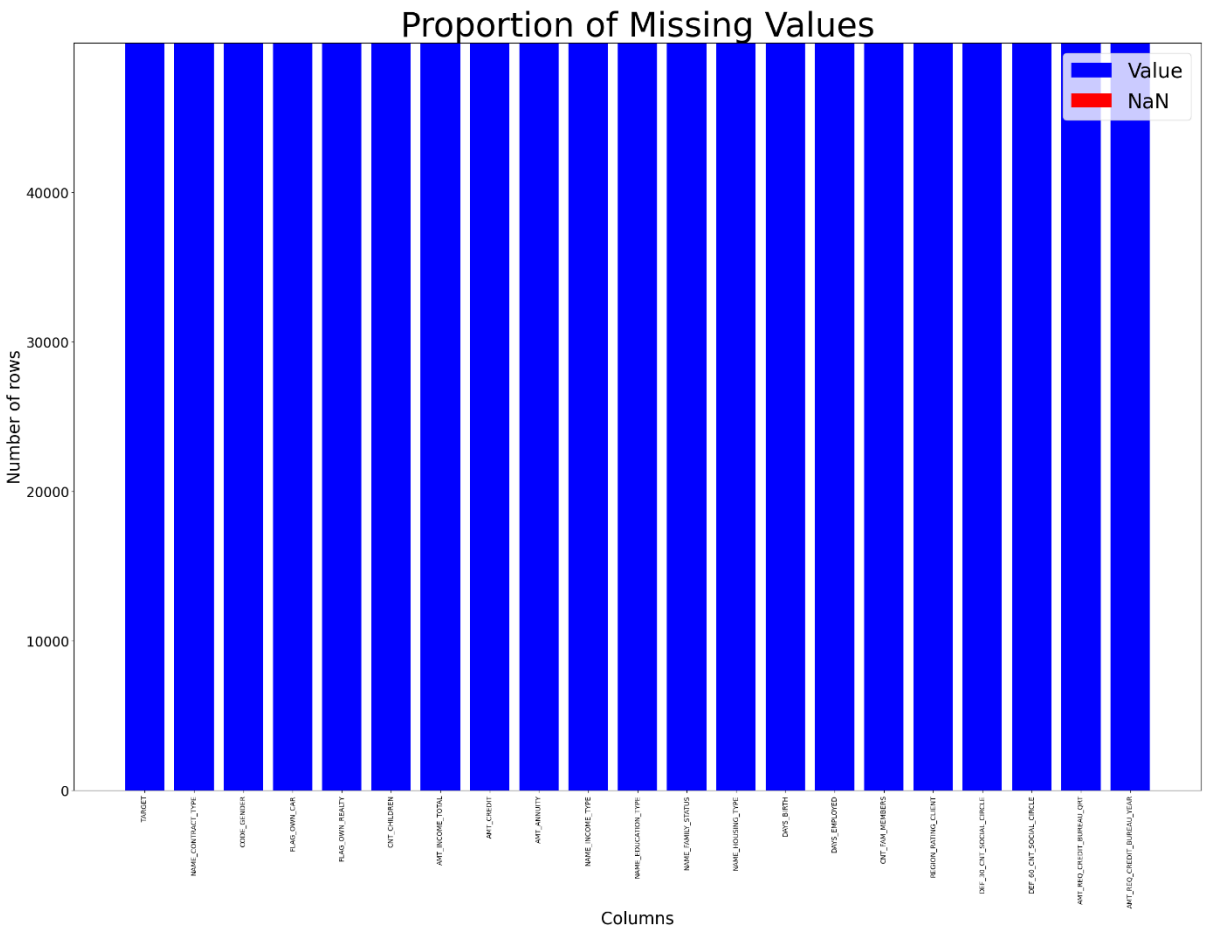
drop_columns is a list containing all the columns to drop (I mentioned it in the Colab workbook).



After dropping unnecessary columns, we now have 21 columns and 50000 records to analyze.

For the missing values in some columns, we will impute them with the median, and not the mean since median is more robust to outliers (extreme values).

```
for label, content in data.items():
    if pd.api.types.is_numeric_dtype(content) and content.isna().sum():
        data[label] = content.fillna(content.median())
```



See that there are no more missing values in our new dataset. Now, we are ready to begin our analysis on this new dataset.

B. Identify Outliers in the Dataset

On executing the following code, I got the mean, maximum, minimum and quartile range values of each column

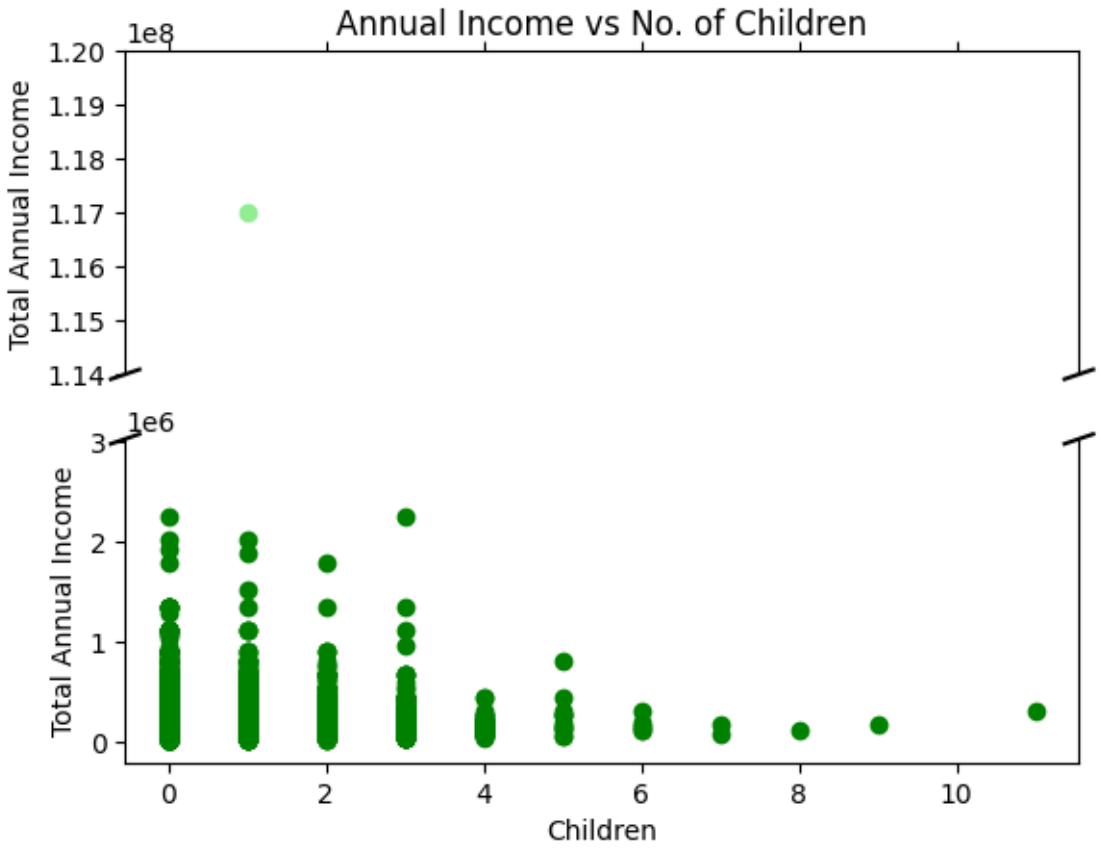
```
data.describe()
```

	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	DAYS_BIRTH	DAYS_EMPLOYED	CNT_FAM_MEMBERS	REGION_RATING_CLIENT	...
count	49999.000000	49999.000000	4.999900e+04	4.999900e+04	49999.000000	49999.000000	49999.000000	49999.000000	49999.000000	
mean	0.080522	0.419848	1.707676e+05	5.997006e+05	27107.333987	-16022.042081	63219.424488	2.158943	2.051661	
std	0.272102	0.724039	5.318191e+05	4.024154e+05	14562.802028	4361.400270	140794.605668	0.911324	0.507978	
min	0.000000	0.000000	2.565000e+04	4.500000e+04	2052.000000	-25184.000000	-17531.000000	1.000000	1.000000	
25%	0.000000	0.000000	1.125000e+05	2.700000e+05	16456.500000	-19644.000000	-2786.000000	2.000000	2.000000	
50%	0.000000	0.000000	1.458000e+05	5.147775e+05	24939.000000	-15731.000000	-1221.000000	2.000000	2.000000	
75%	0.000000	1.000000	2.025000e+05	8.086500e+05	34596.000000	-12378.500000	-292.000000	3.000000	2.000000	
max	1.000000	11.000000	1.170000e+08	4.050000e+06	258025.500000	-7680.000000	365243.000000	13.000000	3.000000	

Observe that the 'CNT_CHILDREN', 'AMT_INCOME_TOTAL' have potential outliers.

For example, the max value in 'CNT_CHILDREN' is 11 but the mean is as small as 0.42. This tells us that the max value doesn't have any effect on the mean and hence, it can be considered an outlier.

Similar is the case with 'AMT_INCOME_TOTAL'; the maximum value is $117,000,000 while the mean is $170,767.6.



```
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
ax1.scatter("CNT_CHILDREN", "AMT_INCOME_TOTAL", data = data,
color='lightgreen')
ax2.scatter("CNT_CHILDREN", "AMT_INCOME_TOTAL", data=data,
color='green')
ax1.set_title("Annual Income vs No. of Children")
ax2.set_xlabel("Children")
ax1.set_ylabel("Total Annual Income")
ax2.set_ylabel("Total Annual Income")

ax1.set_ylim(114000000, 120000000)
ax2.set_ylim(-200000, 3025000.0)
ax1.spines['bottom'].set_visible(False)
ax2.spines['top'].set_visible(False)
ax1.xaxis.tick_top()
ax1.tick_params(labeltop=False)
ax2.xaxis.tick_bottom()
d = .015
kwargs = dict(transform=ax1.transAxes, color='k', clip_on=False)
ax1.plot((-d, +d), (-d, +d), **kwargs)
ax1.plot((1 - d, 1 + d), (-d, +d), **kwargs)

kwargs.update(transform=ax2.transAxes)
ax2.plot((-d, +d), (1 - d, 1 + d), **kwargs)
ax2.plot((1 - d, 1 + d), (1 - d, 1 + d), **kwargs)
plt.show()
```

I plotted it on a graph and confirmed that there is one major outlier with an income of $117M, while all other observations are in the **$0-3M** range.

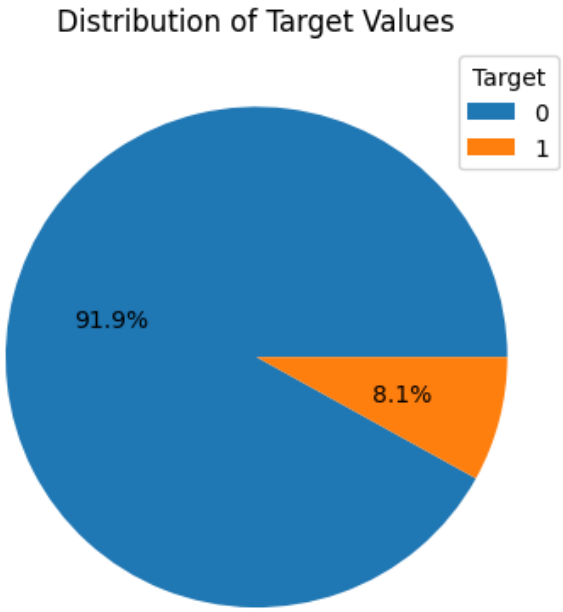
There is one data point with 11 children but a comparatively lower income. This could be considered a minor outlier since it won't make much of a difference in our analysis.

The code above has been referenced from:
https://matplotlib.org/3.1.0/gallery/subplots_axes_and_figures/broken_axis.html

C. Data Imbalance

Since we are interested in knowing if an applicant's loan has been defaulted (late payment), we'll use the 'TARGET' column as the basis for checking an imbalance.

Let's plot a pie chart for the 'TARGET' column.




```
labels = ['0', '1']
fig, ax = plt.subplots()
ax.pie(data["TARGET"].value_counts(),
       autopct='%1.1f%%')
ax.set(title="Distribution of Target Values")
ax.legend(labels=labels,
          title="Target");
```

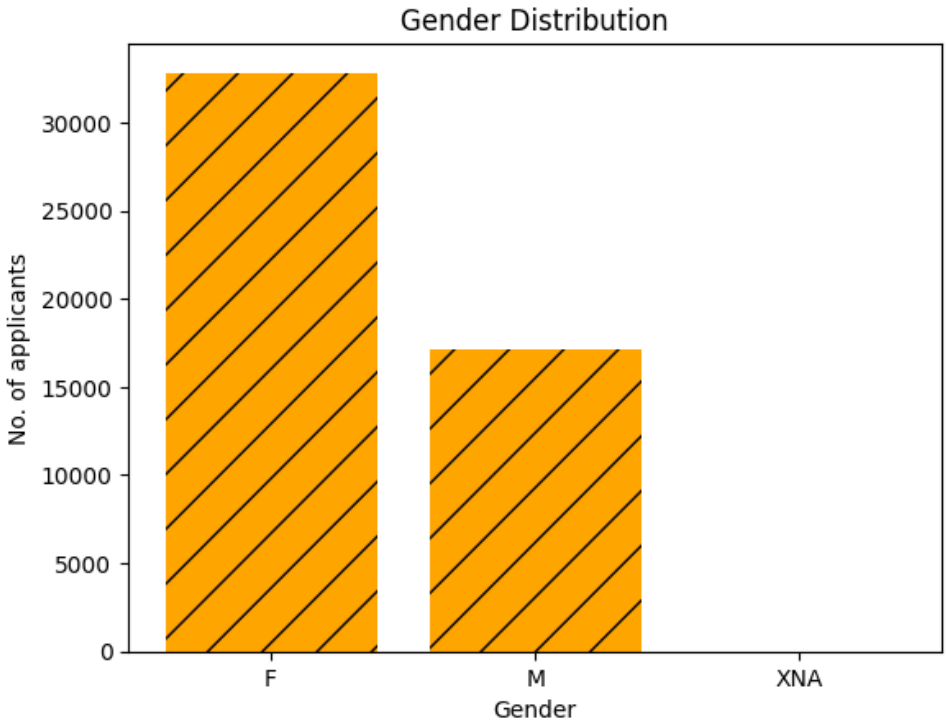
As observed, there is a huge imbalance in the number of loan defaulters (8.1%) and people who repaid their instalments on time (91.9%).

The loans that got defaulted are around 1/10th in number compared to the ones that didn't.

Data imbalance can affect the accuracy of the analysis and lead to biases, especially for binary classification problems.

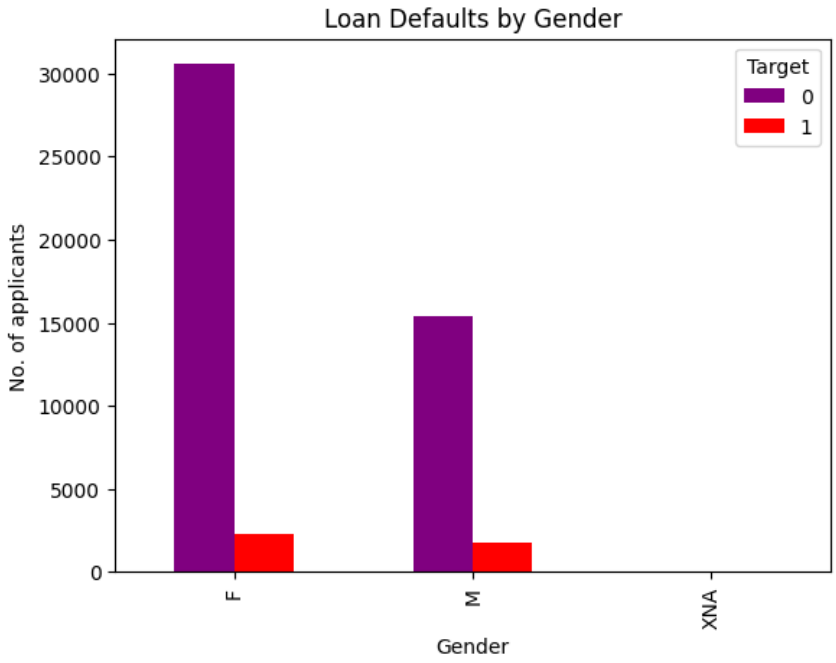
D. Perform Univariate, Segmented Univariate, and Bivariate Analysis

a) Let's first look at the distribution of gender among the applicants.



Observe that the female applicants are double in number to their male counterparts. XNA is almost negligible. This is another example of data imbalance where all elements of a column domain are not equally represented.

We are interested in loan defaults but this bar chart isn't truly reflective of what we want. Therefore, let's plot a bar chart taking 'TARGET' into consideration.

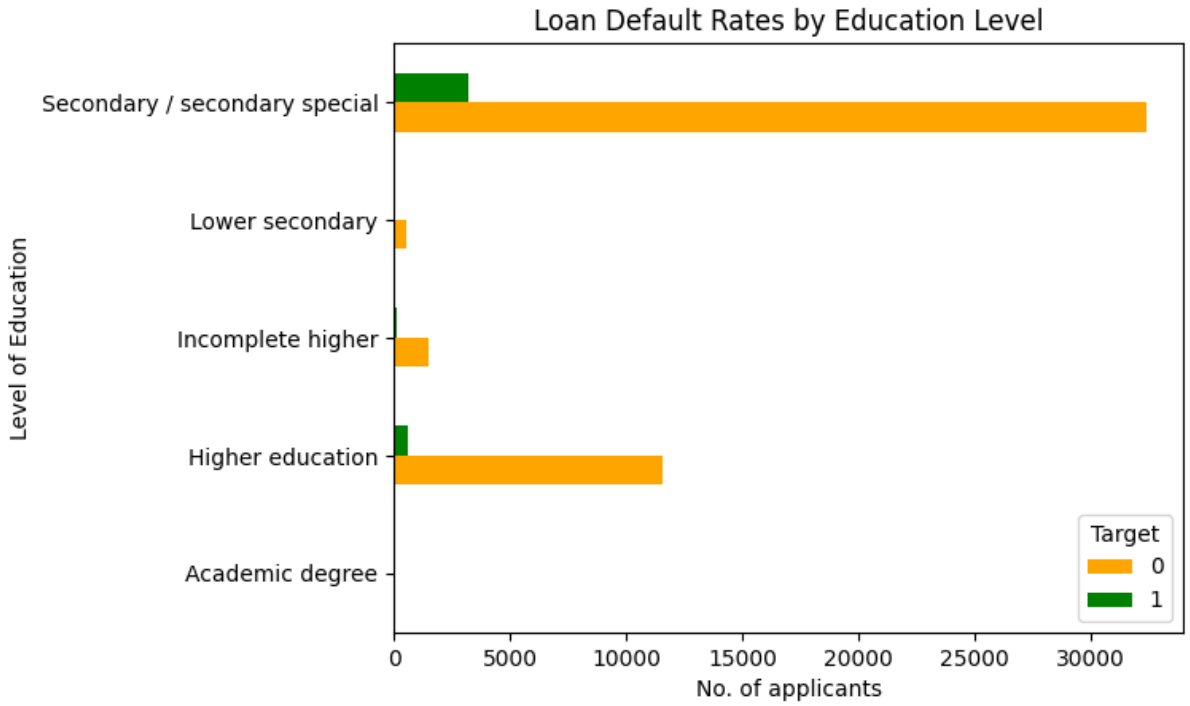


```
pd.crosstab(data["CODE_GENDER"],
            data["TARGET"]).plot(kind="bar",
                                 color=["purple", "red"],
                                 ylabel="No. of applicants",
                                 xlabel="Gender",
                                 title="Loan Defaults by Gender")
plt.legend(['0', '1'], title="Target");
```

As you can see, roughly **10%** of male applicants ended up defaulting their loans. This amount was lesser for female applicants (**7%**). This means it's important to look at male applications with even more scrutiny, before approving them.

Overall, the proportion is almost the same; hence we can conclude that gender **doesn't have** that much of a role to play when it comes to delayed loan payments.

b) Education is an important aspect in one's life. Let's look at how education plays a role, if any, in an applicant's loan being defaulted.

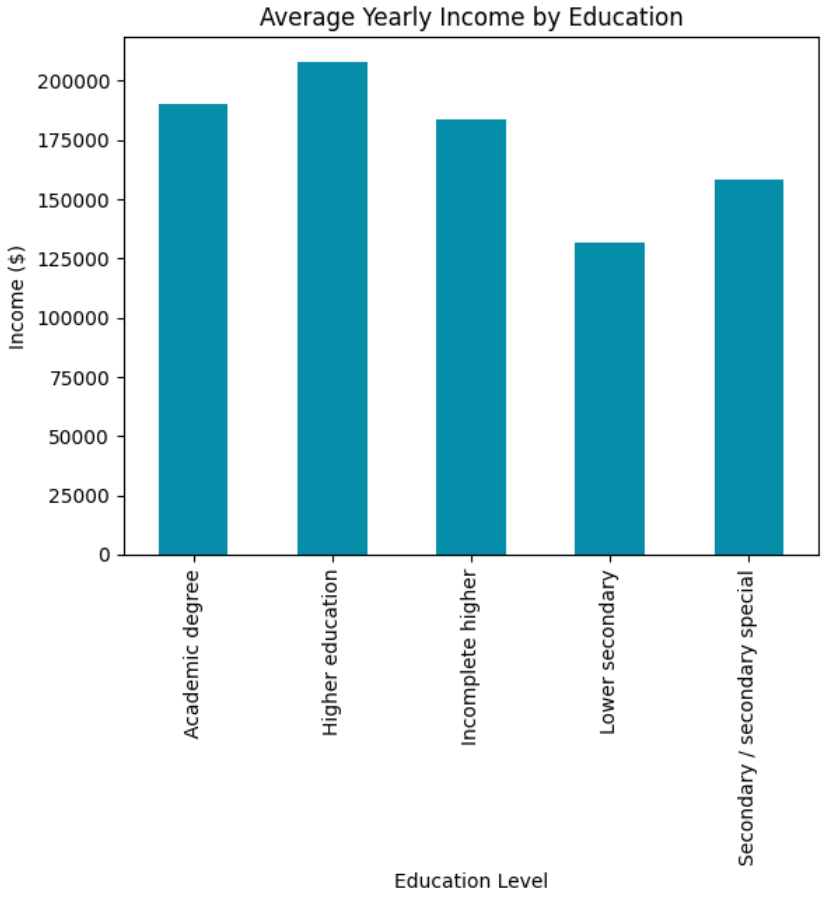



```
pd.crosstab(data["NAME_EDUCATION_TYPE"],
            data["TARGET"]).plot(kind="barh",
                                 color=["orange", "green"],
                                 xlabel="No. of applicants",
                                 ylabel="Level of Education",
                                 title="Loan Default Rates by Education
Level")
plt.legend(['0', '1'], title="Target", loc = 'lower right');
```

Insights:

- Academic degree has almost negligible records.
- Lower Secondary and Incomplete higher education levels have comparatively lesser representation.
- Percentage of loan defaults for each category:
    1. Academic degree: 0%
    2. Higher education: ~5%
    3. Incomplete higher: 8.5%
    4. Secondary / secondary special: 9%
    5. Lower secondary: 11.7%
- The grouped horizontal bar graph makes it very clear that as people get more educated, the chances of having difficulties in payment reduces. This could probably be due to higher income streams and more awareness. Hence, higher education and on-time payment go hand-in-hand.

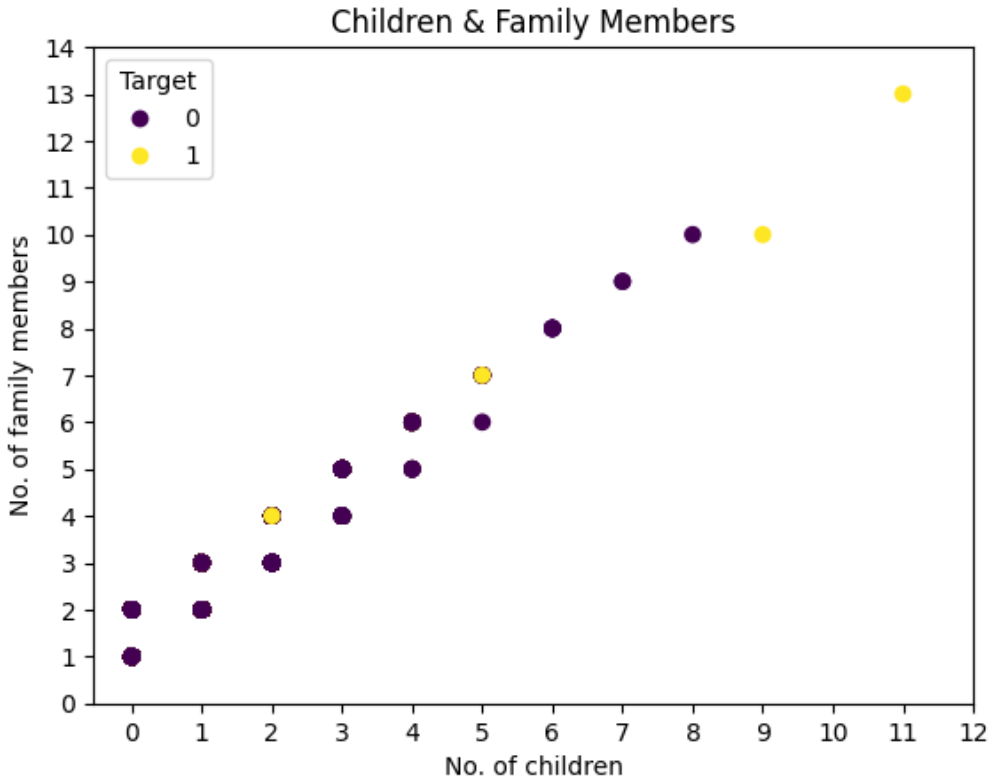
c) Talking of income, we'll look at how education levels and income streams are related. We expect it to be directly proportional.



```
pivot = data.pivot_table(index="NAME_EDUCATION_TYPE",
                         values="AMT_INCOME_TOTAL",
                         aggfunc = "mean")
pivot.plot(kind="bar",
           xlabel="Education Level",
           ylabel="Income ($)",
           title="Average Yearly Income by Education",
           color = '#068da9',
           legend=False);
```

As per our assumptions, **more the education** an applicant receives, better would be their chances of drawing **a higher mean yearly salary**.

d) Now, it's time to look at the number of family members and children for each applicant how that affects a loan applicant in repayment of their loan.




```
fig, ax = plt.subplots()
sc = ax.scatter(x=data["CNT_CHILDREN"],
           y=data["CNT_FAM_MEMBERS"],
           c=data["TARGET"])
ax.set(title="Children & Family Members",
       xlabel="No. of children",
       ylabel="No. of family members",
       xticks=[0,1,2,3,4,5,6,7,8,9,10,11,12],
       yticks=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14])
ax.legend(*sc.legend_elements(), title="Target");
```

It is quite obvious that as a person's family keeps increasing, it is harder to fend for them and hence, there is a **higher** chance of payment difficulties and loan defaults.

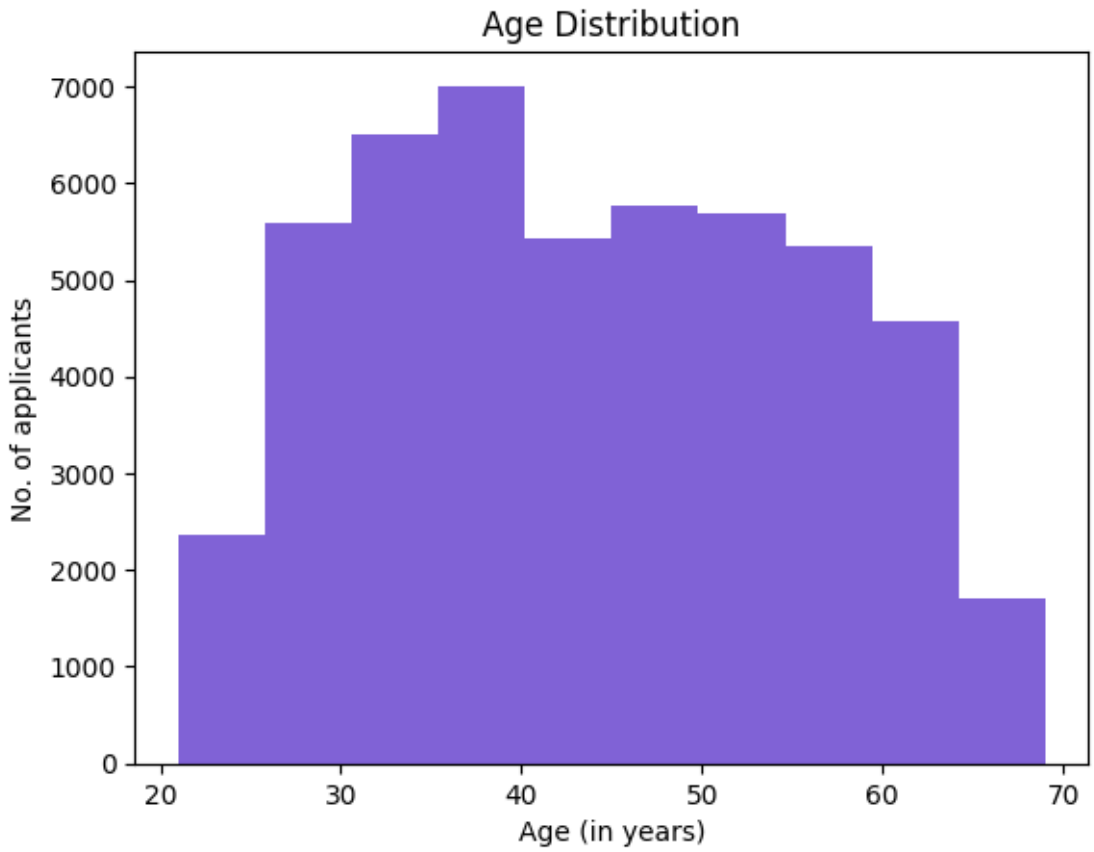
e) Now let's compare 'DAYS_BIRTH' with 'TARGET'

Observe that everything is negative; let's make them positive and calculate the age in years since that is easier to gauge.

Convert age in days to years (conversion: 1 year = 365.25 days).

```
data["AGE"] = -round(-data["DAYS_BIRTH"]/365.25)
data["AGE"]
```

Now that the columns are ready so let's look at the age distribution using a histogram.

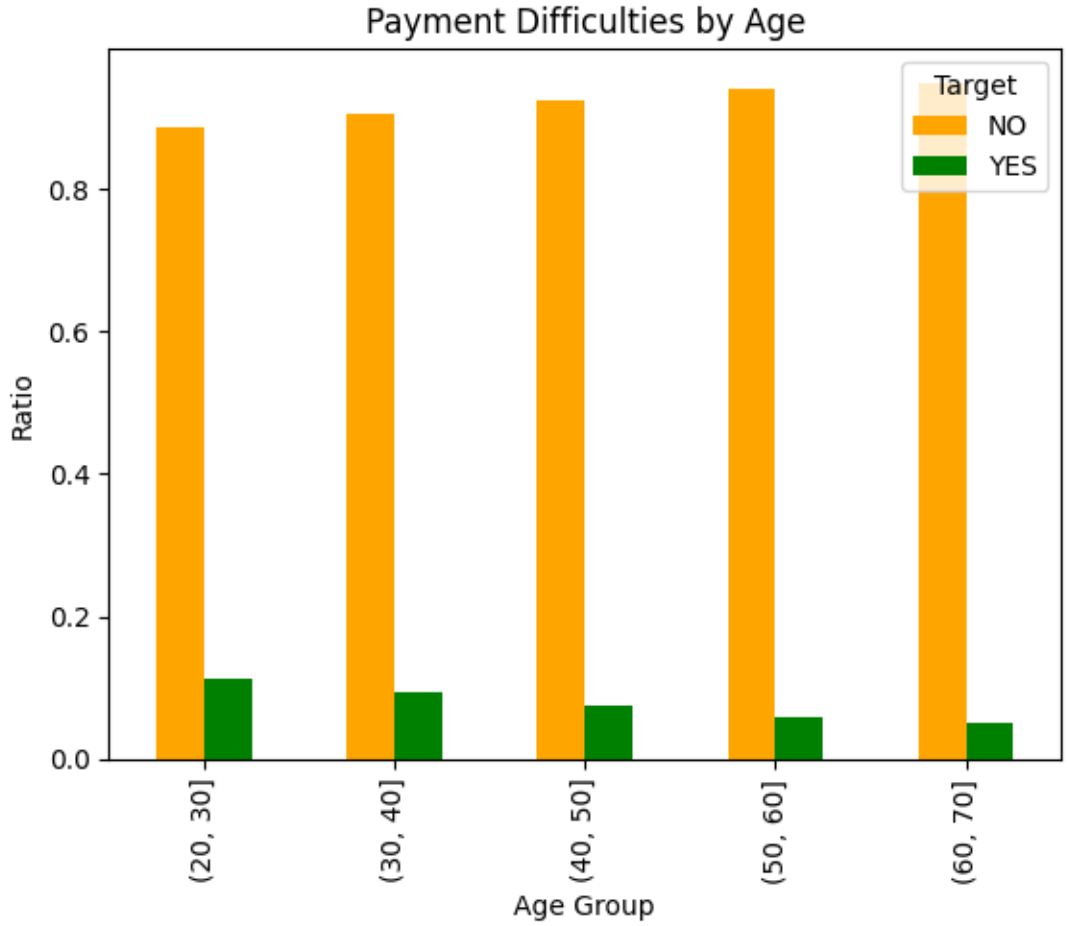



```
fig, ax = plt.subplots()
ax.hist(data["AGE"],
        color="#8062D6");
ax.set(title="Age Distribution",
       ylabel="No. of applicants",
       xlabel="Age (in years)");
```

Notice that most of the applicants are in the age group of **mid-20s to mid-60s**.

Not so surprisingly, this is also the **working age** of a normal population when they are in need of loans to build a house, finance their children's higher education, or plan for retirement.

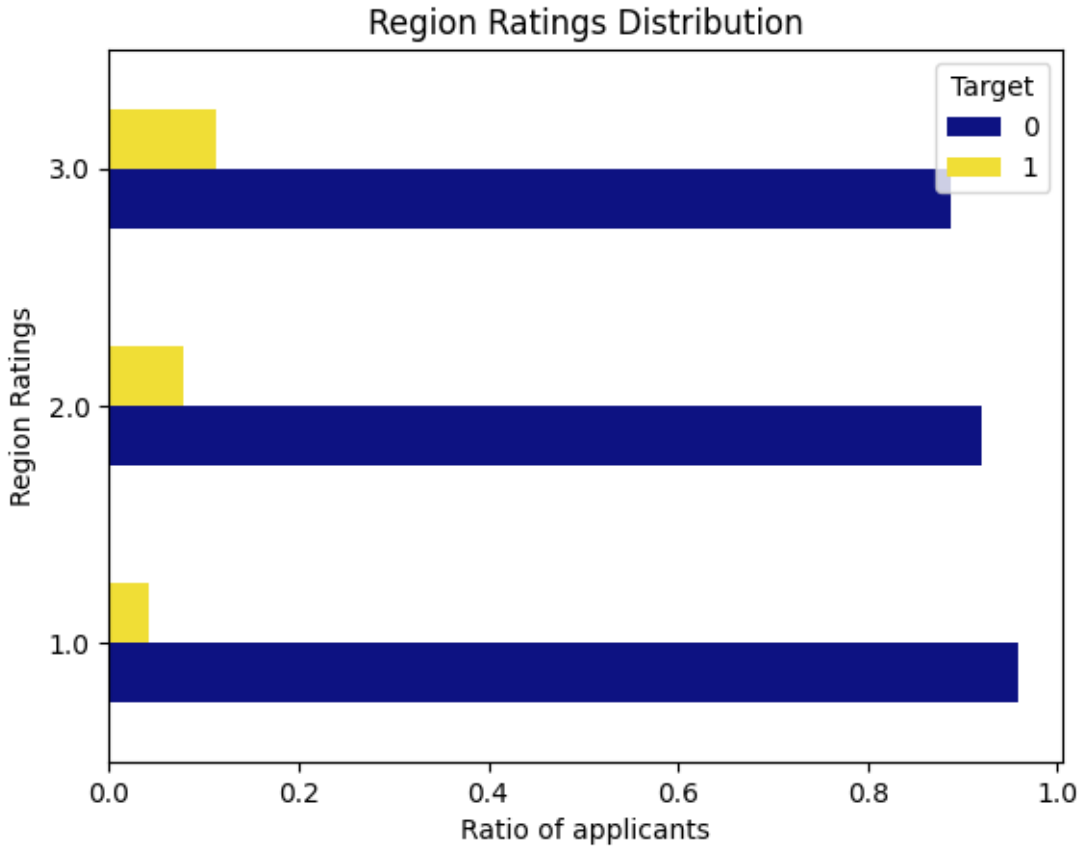
Let's compare it with the 'TARGET' column.




```
age_groups = pd.cut(data["AGE"], bins=[20, 30, 40, 50, 60, 70])

data.groupby(age_groups)["TARGET"].value_counts(normalize=True)

pd.crosstab(age_groups,
            data["TARGET"],
            normalize='index').plot(kind="bar",
                                    xlabel="Age Group",
                                    ylabel="Ratio",
                                    title="Payment Difficulties by
Age",
                                    color=["orange", "green"])
plt.legend(labels=["NO", "YES"], title="Target");
```

Observe that the ratio of delayed payments, although not significantly, keeps reducing with age. With age people get more settled in life and have a stable income to support their families. Younger people tend to not have financial literacy which could be the cause of unnecessary spending and in turn, leading to loan defaults.

f) Last but certainly not the least, it is very important to observe how pre-established 'REGION_RATING_CLIENT' affects an application leading to loan defaults.
Via a grouped bar chart, let's look at the distribution of data in the REGION_RATING_CLIENT.




```
pd.crosstab(data["REGION_RATING_CLIENT"],
            data["TARGET"],
            normalize='index').plot(kind="barh",
                                    xlabel="Ratio of applicants",
                                    ylabel="Region Ratings",
                                    title="Region Ratings
Distribution",
                                    color=["#0D1282", "#F0DE36"])
plt.legend(labels=["0", "1"], title="Target");
```

The ratings are in fact inconsistent and seem inaccurate since a region with a low rating (1.0) has lower cases of loan defaults while that with a high rating (3.0) has the highest cases. We see a direct relation between region rating and proportion of loan defaults although an opposite relation was expected.

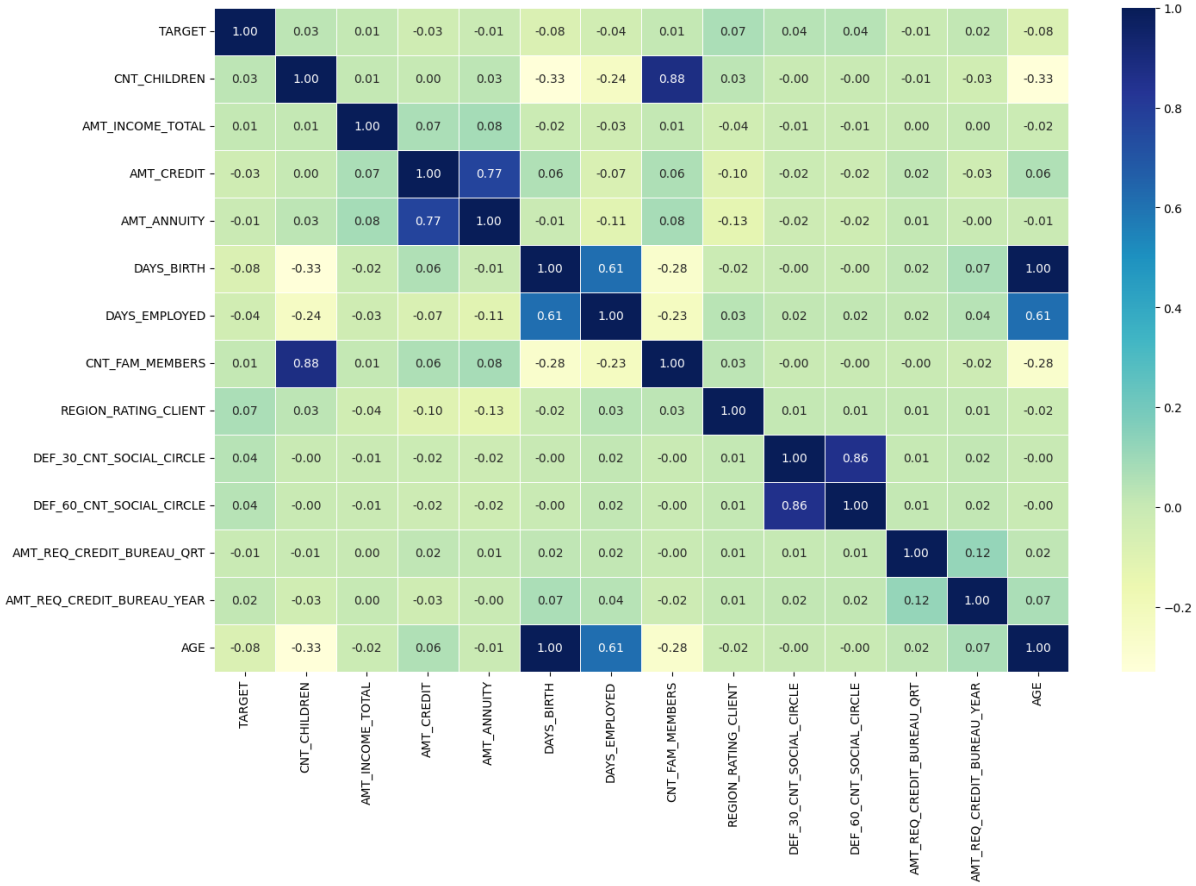
From the univariate and bivariate analysis, we draw the following conclusions & insights:

- Males are more likely to make a delayed payment compared to females.
- Higher education directly implies a lower chance of having payment difficulties.
- Higher education, more often than not, guarantees a higher annual salary.
- As family members and children keep increasing, the chances of a late payment are higher.
- Loans are common the age group of 25 - 65.
- Younger people tend to find it difficult to repay their loan.
- A client's region ratings are inaccurate and shouldn't be taken into account.

E. Identify Top Correlations for Different Scenarios

Here we'll try understanding the correlation between variables and the target variable and provide insights into strong indicators of loan default.

We shall plot a Seaborn heatmap to visualize the correlations between the different variables and the 'TARGET' variable.




```
fig, ax = plt.subplots(figsize=(16, 10))
ax = sns.heatmap(data.corr(),
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
```

Overall, there aren't many variables that possess a strong positive or negative correlation with the 'TARGET' variable.

But as stated before, AGE and DAYS_EMPLOYED are inversely proportional to the late payments 'TARGET', while 'CNT_CHILDREN' and 'REGION_RATING_CLIENT' are directly proportional (to having loan defaults).

But there are some variable pairs like (CNT_CHILDREN, CNT_FAM_MEMBERS) and (AGE, DAYS_EMPLOYED) with strong correlation.

Closing Note:

In this bank loan case study project, I undertook a comprehensive data analysis approach to determine the likelihood of loan default for loan applicants. To accomplish this, I leveraged a powerful combination of tools, including Excel and Google Colab (Jupyter Notebook), along with Python libraries such as NumPy, Pandas, Matplotlib, and Seaborn. These tools enabled me to manipulate and visualize the data effectively, gaining valuable insights into the loan application dataset. Throughout the project, I adhered to best practices in data analysis and machine learning, ensuring the accuracy and validity of the findings.

Overall, this bank loan case study project provided me with a comprehensive learning experience in data analysis and visualization, as well as financial domain applications. It equipped me with the skills and knowledge to tackle similar business challenges in the future and reinforced my commitment to leveraging data-driven insights for informed decision-making.