Oeson
Inspiring generation

CREDIT RISK ANALYSIS

EXPLORATORY DATA ANALYSIS(EDA) WITH PYTHON

PREPARED BY
JOSEPHINE AKUBILLA
SUPERVISOR: DR. ARITRI DEBNATH GHOSH

TABLE OF CONTENT

INTRODUCTION

This exploratory data analysis focuses on the challenges clients face in meeting their loan installments. The resulting report can furnish the company with insights into the influential factors contributing to loan difficulties.

Utilizing this information, the company can instigate changes in payment structures through risk assessments, pinpointing the most affected individuals and discerning the underlying reasons.

EDA plays a crucial role in identifying variables and patterns related to repayments and loans, forming the basis for creating hypotheses and predictive models of consumer behavior.

In the 'NAME_CONTRACT_TYPE' field of the previous_application dataset, values such as cash loans or revolving loans indicate scenarios where borrowers can spend borrowed money up to a predetermined limit, subsequently repaying and reusing the amount.

INTRODUCTION

Python is celebrated for its readability, versatility, rich library support, and vibrant community. Jupyter Lab enhances the data science experience with its interactive notebook interface and multi-language support. The collaboration between Python and Jupyter Lab is extensively applied in data science, machine learning, research, and education. This powerful combination is favored by professionals and enthusiasts, fostering innovation in diverse domains. It has emerged as a go-to solution, playing a significant role in problem-solving and advancements across various fields.

TOOLS USED

Python accessed through Jupyter Notebook

• Dictionaries:
• Numphy – Used for numerical applications
• Pandas – the most extensive dictionary used in Python for data manipulation & analysis
• Matplotlib – Data Visualization
• Seaburn – Statistical Plotting
• Scikit-Lean – Data Analysis & Mining
• Openpyxl – For connectivity through excel (importing

OBJECTIVES

The objective of this project is to identify specific patterns indicative of potential challenges in clients meeting their scheduled payments. The goal is to make informed decisions, such as rejecting loans, reducing loan amounts, or applying higher interest rates to applicants demonstrating a higher risk profile based on these patterns. Ultimately, this approach aims to mitigate risks associated with lending to borrowers who may face repayment difficulties while ensuring that deserving loan applications are not unfairly denied. Additionally, the project seeks to comprehend the primary drivers or variables that significantly contribute to loan defaults, serving as reliable predictors of potential default scenarios. Gaining insights into these crucial variables enables the business to manage its portfolio more adeptly, conduct more precise risk assessments, and make informed decisions, ultimately enhancing strategies to minimize the likelihood of loan defaults.

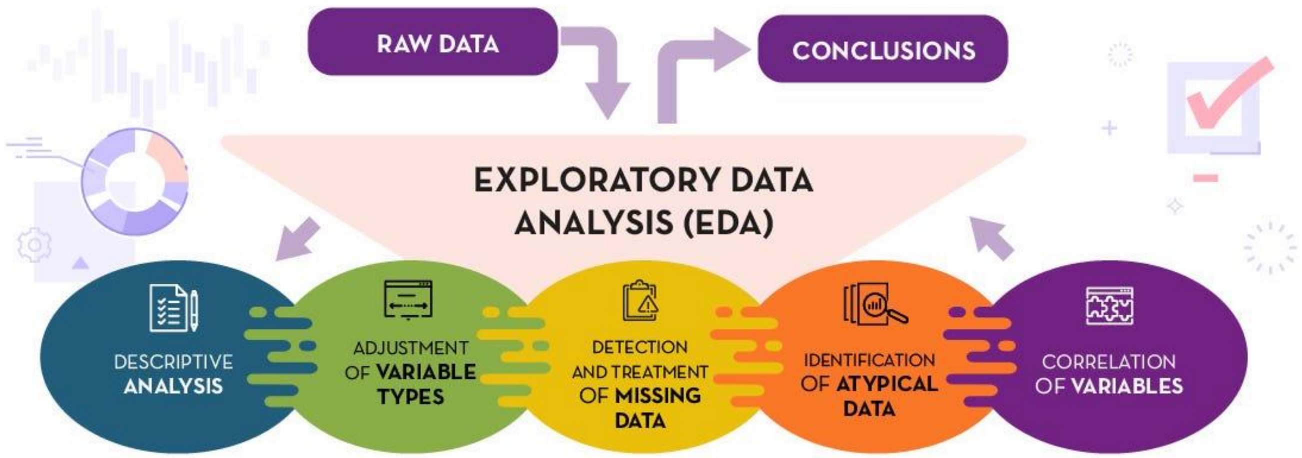
The dataset comprises three files:
1. **'application_data.csv':** This file encompasses comprehensive information about clients sourced from loan applications.
2. **'columns_description.csv':** Serving as a data dictionary, this file elucidates the significance of variables within the dataset.
3. **'previous_application.csv':** This file contains details about clients' prior loan applications, specifying whether they were approved, rejected, cancelled, or remained unused.

CREDIT RISK ANALYSIS

- Credit risk analysis helps financial institutions and lenders make informed decisions about whether to approve or deny a loan, set appropriate interest rates, and establish credit limits.
- Additionally, it plays a crucial role in portfolio management, as it helps in monitoring and managing the overall credit risk exposure of an institution.
- Due to their weak or nonexistent credit histories, loan providers find it challenging to grant loans or individuals. Because of this, some customers take advantage of it by defaulting.

EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with the help of graphical representations and statistical techniques. The primary goal of EDA is to uncover patterns, relationships, anomalies, and trends in the data, providing a better understanding of its underlying structure. During EDA, analysts explore the data visually and through summary statistics to identify patterns or insights that can guide further analysis. Techniques involved in EDA include data visualization (histograms, scatter plots, box plots), summary statistics (mean, median, standard deviation), and sometimes more advanced statistical methods. EDA is a crucial step in the data analysis process, helping analysts make informed decisions about subsequent analyses or modeling.

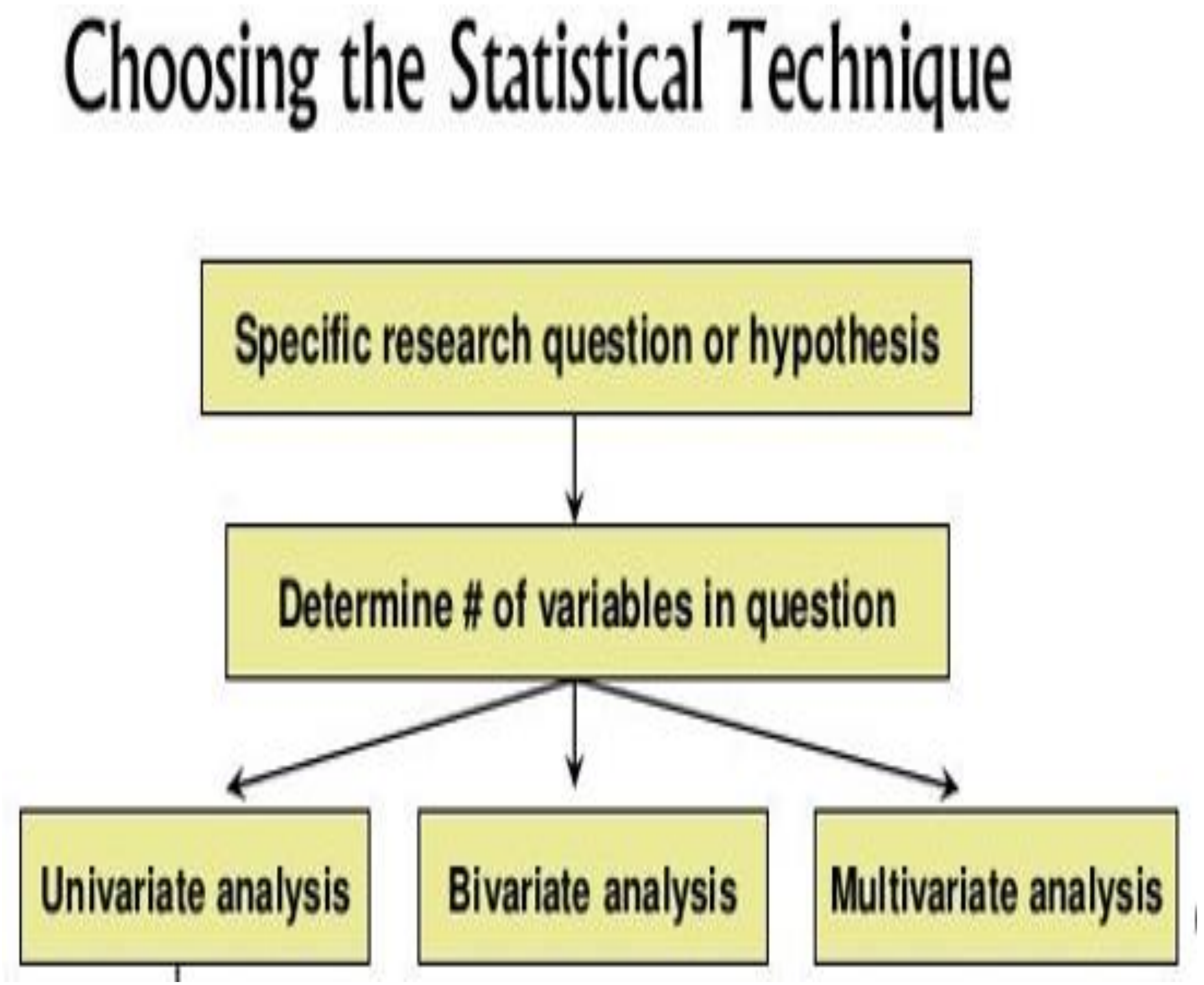


EXPLORATORY DATA ANALYSIS (EDA)

Univariate analysis involves the examination of a single variable, exploring its distribution, central tendency, and variability. It provides insights into the characteristics and patterns within that specific variable.

Bivariate analysis examines the relationship between two variables, assessing how changes in one variable correlate with changes in another. Common methods include correlation and regression analysis.

Multivariate analysis extends the exploration to three or more variables simultaneously. It allows for a comprehensive understanding of complex relationships and interactions among multiple variables, aiding in predictive modeling and decision-making processes.



PROBLEM STATEMENT, MOTIVATION & SIGNIFICANCE

Problem Statement 1: Enhancing Loan Repayment Strategies

Challenge: Clients encounter difficulties in meeting their loan installments.

Motivation: Improve company understanding of influential factors causing loan challenges.

Significance: A well-informed strategy can enhance payment structures, mitigating risks and benefiting both clients and the company.

Problem Statement 2: Targeted Risk Assessment for Payment Changes

Challenge: Determining the most affected individuals and reasons behind loan difficulties.

Motivation: Utilize data insights to initiate targeted risk assessments and strategic payment changes.

Significance: Pinpointing high-risk cases enables proactive measures, ensuring a more secure and stable loan portfolio.

Problem Statement 3: Predictive Modeling for Consumer Behavior

Challenge: Identifying variables and patterns related to repayments and loans.

Motivation: Leverage EDA to form hypotheses and predictive models of consumer behavior..

Significance: Predictive models enhance decision-making, allowing the company to anticipate and respond effectively to consumer trends

Problem Statement 4: Optimization of Cash and Revolving Loans

Challenge: Understanding the dynamics of 'NAME_CONTRACT_TYPE' values (e.g., cash loans, revolving loans).

Motivation: Analyze borrower behavior in scenarios of spending, repaying, and reusing funds.

Significance: Optimization ensures tailored lending solutions, aligning with borrower needs and fostering financial flexibility.

DATA PREPARATION

DATA CLEANING

```
In [8]: df1.duplicated()

Out[8]: 0          False
        1          False
        2          False
        3          False
        4          False
                   ...
        1670209    False
        1670210    False
        1670211    False
        1670212    False
        1670213    False
        Length: 1670214, dtype: bool
```

```
In [9]: df2.duplicated()

Out[9]: 0         False
        1         False
        2         False
        3         False
        4         False
                  ...
        307506    False
        307507    False
        307508    False
        307509    False
        307510    False
        Length: 307511, dtype: bool
```

```
In [11]: df2.isnull().sum()

Out[11]: SK_ID_CURR                     0
         TARGET                         0
         NAME_CONTRACT_TYPE             0
         CODE_GENDER                    0
         FLAG_OWN_CAR                   0
                                      ...
         AMT_REQ_CREDIT_BUREAU_DAY  41519
         AMT_REQ_CREDIT_BUREAU_WEEK 41519
         AMT_REQ_CREDIT_BUREAU_MON  41519
         AMT_REQ_CREDIT_BUREAU_QRT  41519
         AMT_REQ_CREDIT_BUREAU_YEAR 41519
         Length: 122, dtype: int64
```

```
In [10]: # Counting the number of null values
         df1.isnull().sum()

SK_ID_PREV                           0
SK_ID_CURR                           0
NAME_CONTRACT_TYPE                   0
AMT_ANNUITY                     372235
AMT_APPLICATION                      0
AMT_CREDIT                           1
AMT_DOWN_PAYMENT                895844
AMT_GOODS_PRICE                 385515
WEEKDAY_APPR_PROCESS_START           0
HOUR_APPR_PROCESS_START              0
FLAG_LAST_APPL_PER_CONTRACT          0
NFLAG_LAST_APPL_IN_DAY               0
RATE_DOWN_PAYMENT               895844
RATE_INTEREST_PRIMARY          1664263
RATE_INTEREST_PRIVILEGED       1664263
NAME_CASH_LOAN_PURPOSE               0
NAME_CONTRACT_STATUS                 0
DAYS_DECISION                        0
NAME_PAYMENT_TYPE                    0
CODE_REJECT_REASON                   0
NAME_TYPE_SUITE                 820405
NAME_CLIENT_TYPE                     0
NAME_GOODS_CATEGORY                  0
NAME_PORTFOLIO                       0
NAME_PRODUCT_TYPE                    0
CHANNEL_TYPE                         0
SELLERPLACE_AREA                     0
NAME_SELLER_INDUSTRY                 0
CNT_PAYMENT                     372230
NAME_YIELD_GROUP                     0
PRODUCT_COMBINATION                346
DAYS_FIRST_DRAWING              673065
DAYS_FIRST_DUE                  673065
DAYS_LAST_DUE_1ST_VERSION       673065
DAYS_LAST_DUE                   673065
DAYS_TERMINATION                673065
NFLAG_INSURED_ON_APPROVAL       673065
dtype: int64
```

```
In [12]: # Removing null values
         df1.dropna(subset=["AMT_ANNUITY"], inplace = True)

In [13]: # Removing null values
         df2.dropna(subset=["AMT_REQ_CREDIT_BUREAU_DAY"], inplace = True)

In [14]: # Removing null values
         df2.dropna(subset=["AMT_REQ_CREDIT_BUREAU_WEEK"], inplace = True)

In [16]: # Removing null values
         df2.dropna(subset=["AMT_REQ_CREDIT_BUREAU_MON"], inplace = True)

In [17]: # Removing null values
         df2.dropna(subset=["AMT_REQ_CREDIT_BUREAU_QRT"], inplace = True)

In [18]: # Removing null values
         df2.dropna(subset=["AMT_REQ_CREDIT_BUREAU_YEAR"], inplace = True)
```

DATA CLEANING

In [10]:
```
# Count of unique Values
df1.nunique()
```

Out[10]:
```
SK_ID_PREV                     1670214
SK_ID_CURR                      338857
NAME_CONTRACT_TYPE                   4
AMT_ANNUITY                     357959
AMT_APPLICATION                  93885
AMT_CREDIT                       86803
AMT_DOWN_PAYMENT                 29278
AMT_GOODS_PRICE                  93885
WEEKDAY_APPR_PROCESS_START           7
HOUR_APPR_PROCESS_START             24
FLAG_LAST_APPL_PER_CONTRACT          2
NFLAG_LAST_APPL_IN_DAY               2
RATE_DOWN_PAYMENT               207033
RATE_INTEREST_PRIMARY              148
RATE_INTEREST_PRIVILEGED            25
NAME_CASH_LOAN_PURPOSE              25
NAME_CONTRACT_STATUS                 4
DAYS_DECISION                     2922
NAME_PAYMENT_TYPE                    4
```

In [11]:
```
df1.sort_values(by="NAME_CONTRACT_TYPE")
```

Out[11]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE
835106	1313808	217032	Cash loans	NaN	0.000000	0.000000	NaN	NaN
909307	2147100	402177	Cash loans	NaN	0.000000	0.000000	NaN	NaN
909308	1044724	178344	Cash loans	24639.210000	337500.000000	368685.000000	NaN	337500.000000
909311	1888218	331321	Cash loans	16946.640000	454500.000000	526491.000000	NaN	454500.000000
909312	2288277	213304	Cash loans	17204.220000	157500.000000	167895.000000	NaN	157500.000000
...	...	...	...	...	...	...	...	...
885690	2385001	284394	XNA	NaN	0.000000	0.000000	NaN	NaN
778611	1325487	405555	XNA	NaN	0.000000	0.000000	NaN	NaN
348152	2059701	206926	XNA	NaN	0.000000	0.000000	NaN	NaN
1547869	1018815	103715	XNA	NaN	0.000000	0.000000	NaN	NaN
1136219	1533594	260330	XNA	NaN	0.000000	0.000000	NaN	NaN

1670214 rows × 37 columns

In [12]:
```
df1.sort_values(by="NAME_SELLER_INDUSTRY", ascending = False).head()
```

Out[12]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE
835107	1625667	399451	Cash loans	NaN	0.000000	0.000000	NaN	NaN
939619	2351388	339213	Cash loans	NaN	0.000000	0.000000	NaN	NaN
939627	2041184	195379	Revolving loans	16875.000000	337500.000000	337500.000000	NaN	337500.000000
939626	2093896	289393	Cash loans	NaN	0.000000	0.000000	NaN	NaN
939625	2370573	377164	Cash loans	8338.050000	90000.000000	98910.000000	NaN	90000.000000

5 rows × 37 columns

In [13]:
```
df2.sort_values(by="NAME_CONTRACT_TYPE", ascending = False).head()
```

Out[13]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AM
267289	409683	0	Revolving loans	F	N	Y	2	67500.000000	1800
142675	265436	0	Revolving loans	M	Y	Y	1	225000.000000	2700
178432	306764	0	Revolving loans	M	Y	Y	0	114435.000000	1350
142664	265425	0	Revolving loans	F	N	N	1	225000.000000	2700
142666	265427	0	Revolving loans	F	N	N	0	90000.000000	2475

5 rows × 122 columns

In [14]:
```
df1.duplicated()
```

Out[14]:
```
0          False
1          False
2          False
3          False
4          False
           ...
1670209    False
1670210    False
1670211    False
1670212    False
1670213    False
Length: 1670214, dtype: bool
```

In [15]:
```
df2.duplicated()
```

Out[15]:
```
0         False
1         False
2         False
3         False
4         False
          ...
307506    False
307507    False
307508    False
307509    False
307510    False
Length: 307511, dtype: bool
```

OUTLIER DETECTION

An outlier is a data point that significantly deviates from the majority in a dataset, standing as an extreme value. It falls outside the usual range and can distort statistical analyses if not properly addressed. Outliers may result from errors, natural variations, or unusual events. Identifying outliers is crucial in data analysis to ensure accurate interpretations of results. Detection methods include visual inspection, statistical techniques like z-scores and interquartile range, and machine learning algorithms. Handling outliers involves a careful decision on whether to remove them or consider them as valuable, albeit unusual, observations. Ultimately, managing outliers is essential for maintaining the integrity and reliability of statistical analyses.

AMT_INCOME_TOTAL mean = 1.71275

AMT_CREDIT mean = 6.063205

AMT_ANNUITY mean = 27174.6

AMT_GOODS_PRICE mean = 5.45

COMBINING DATA SETS (MERGING CSV FILES)

In Python, conducting a comprehensive analysis involves merging datasets to integrate information from diverse sources for a holistic perspective. Utilizing the 'merge_df' command, I combined two CSV files to create a consolidated dataset with 1,670,214 rows and 158 columns. The primary goal was to enhance analytical capabilities with a larger, well-labeled dataset for easy interpretation. The harmonization achieved through mergin facilitates effective correlation between related data points, deepenin insights. The unified structure resulting from merging streamline subsequent data manipulation and exploration tasks, promoting workflov efficiency. Furthermore, the process eliminates challenges related t handling heterogeneous data types during analysis. After merging, th column remains the same whereas the row decreases to 1413701.

```
In [26]: merged_df1=df1.merge(df2, how="inner", on="SK_ID_CURR")
```

```
In [27]: merged_df1.shape
Out[27]: (1413701, 158)
```

```
In [13]: df2 = pd.read_csv(r"C:\Users\jakubil\Downloads\application_data.csv")
df2
```

Out[13]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AM
0	100002	1	Cash loans	M	N	Y	0	202500.000000	406
1	100003	0	Cash loans	F	N	N	0	270000.000000	1293
2	100004	0	Revolving loans	M	Y	Y	0	67500.000000	135
3	100006	0	Cash loans	F	N	Y	0	135000.000000	312
4	100007	0	Cash loans	M	N	Y	0	121500.000000	513
...	...	...	...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	N	0	157500.000000	254
307507	456252	0	Cash loans	F	N	Y	0	72000.000000	269
307508	456253	0	Cash loans	F	N	Y	0	153000.000000	677
307509	456254	1	Cash loans	F	N	Y	0	171000.000000	370
307510	456255	0	Cash loans	F	N	N	0	157500.000000	675

307511 rows × 122 columns

```
In [3]: df1 = pd.read_csv(r"C:\Users\jakubil\Downloads\previous_application.csv")
df1
```

Out[3]:

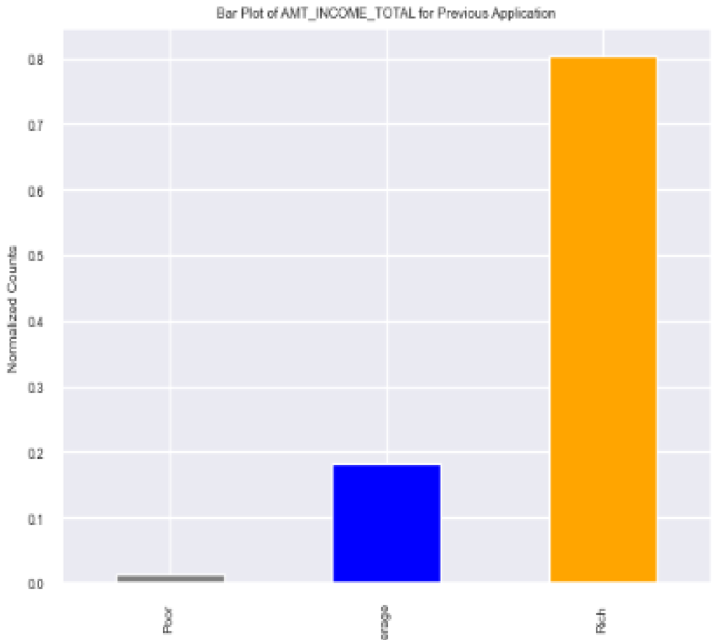
	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0	17145.0
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN	607500.0
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN	112500.0
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN	450000.0
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN	337500.0
...	...	...	...	...	...	...	...	...
1670209	2300464	352015	Consumer loans	14704.290	267295.5	311400.0	0.0	267295.5
1670210	2357031	334635	Consumer loans	6622.020	87750.0	64291.5	29250.0	87750.0
1670211	2659632	249544	Consumer loans	11520.855	105237.0	102523.5	10525.5	105237.0
1670212	2785582	400317	Cash loans	18821.520	180000.0	191880.0	NaN	180000.0
1670213	2418762	261212	Cash loans	16431.300	360000.0	360000.0	NaN	360000.0

1670214 rows × 37 columns

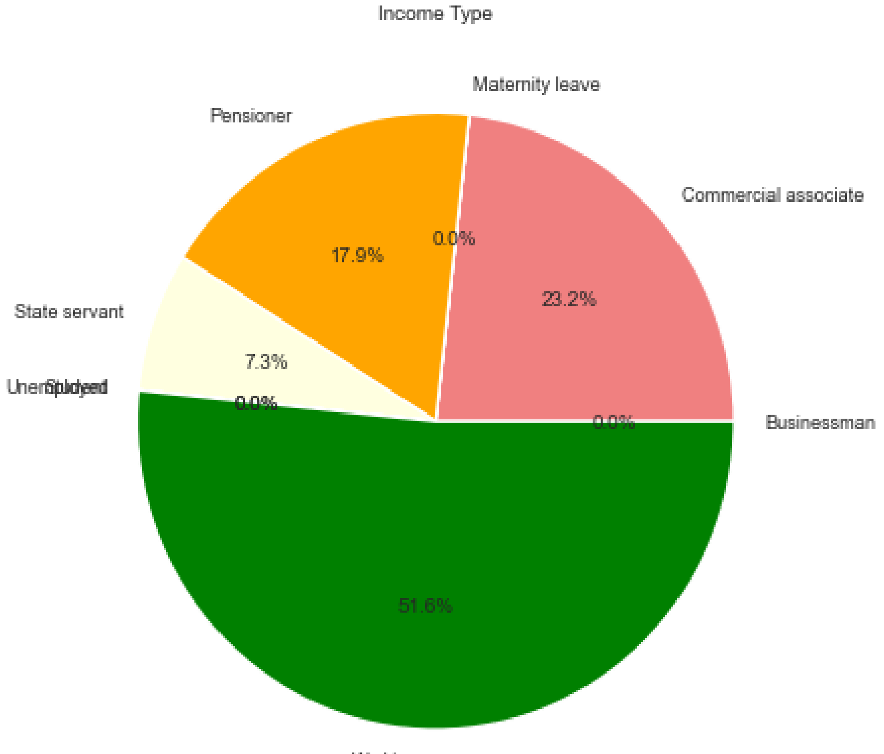
Univariate Analysis

DEMOGRAPHIC CHARACTERISTICS

In [61]:
```
import matplotlib.pyplot as plt

# Define income categories
income_bins = [0, 50000, 100000, float('inf')]
income_labels = ['Poor', 'Average', 'Rich']

# Assuming 'df2' is your DataFrame
df2['Income_Category'] = pd.cut(df2['AMT_INCOME_TOTAL'], bins=income_bins, labels=income_labels, right=False)

# Filter out rows where 'AMT_INCOME_TOTAL' is 'XNA'
filtered_df = df2[df2['AMT_INCOME_TOTAL'] != 'XNA']

# Plot the bar chart
colors = ['grey', 'blue', 'orange', 'pink']
filtered_df['Income_Category'].value_counts(normalize=True).sort_index().plot.bar(color=colors)

plt.title('Bar Plot of AMT_INCOME_TOTAL for Previous Application')
plt.xlabel('Income Category')
plt.ylabel('Normalized Counts')
plt.show()
```

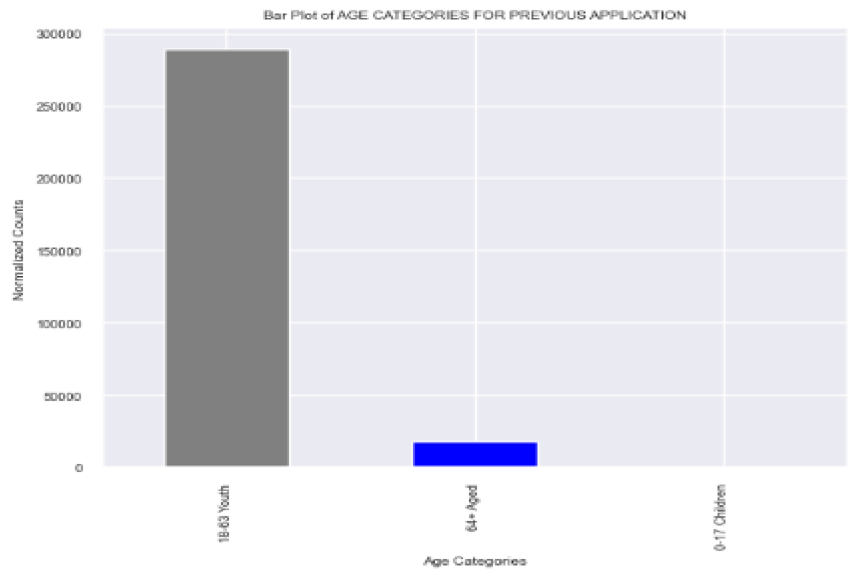


In [56]:
```
import matplotlib.pyplot as plt

# Assuming 'result' is your DataFrame
result.groupby('NAME_INCOME_TYPE').size().plot(
    kind='pie',
    autopct='%1.1f%%',
    colors=['skyblue', 'lightcoral', 'green', 'orange', 'lightyellow']
)

plt.title('Income Type')
plt.show()
```

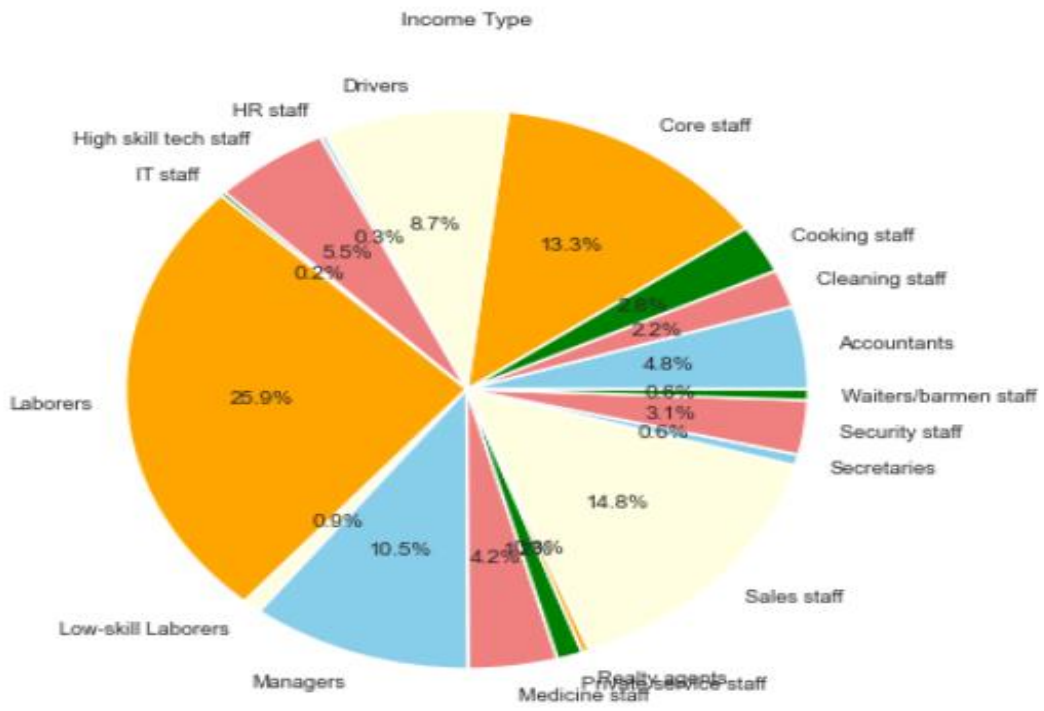


DEMOGRAPHIC CHARACTERISTICS

In [61]:
```
import matplotlib.pyplot as plt

# Filter out 'XNA' values
filtered_df = df2[df2['DAYS_BIRTH'] != 'XNA']

# Convert DAYS_BIRTH to age in years
filtered_df['AGE'] = -filtered_df['DAYS_BIRTH'] // 365

# Define age categories
bins = [0, 17, 63, float('inf')]
labels = ['0-17 Children', '18-63 Youth', '64+ Aged']

# Categorize ages
filtered_df['AGE_CATEGORY'] = pd.cut(filtered_df['AGE'], bins=bins, labels=labels, ri

# Count the occurrences of each age category
age_category_counts = filtered_df['AGE_CATEGORY'].value_counts()

# Create a bar plot
age_category_counts.plot.bar(color=['grey', 'blue', 'orange'])
plt.title('Bar Plot of AGE CATEGORIES FOR PREVIOUS APPLICATION')
plt.xlabel('Age Categories')
plt.ylabel('Normalized Counts')
plt.show()
```

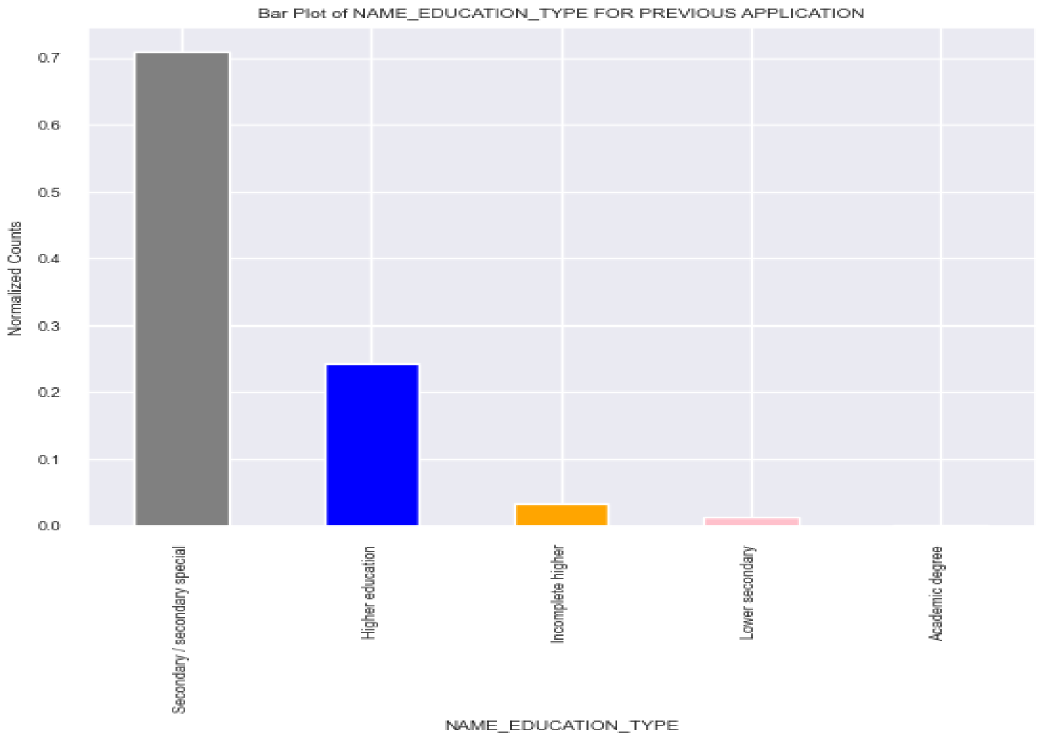


In [51]:
```
import matplotlib.pyplot as plt

# Assuming 'result' is your DataFrame
result.groupby('OCCUPATION_TYPE').size().plot(
    kind='pie',
    autopct='%1.1f%%',
    colors=['skyblue', 'lightcoral', 'green', 'orange', 'lightyellow']
)

plt.title('Occupation Type')
plt.show()
```



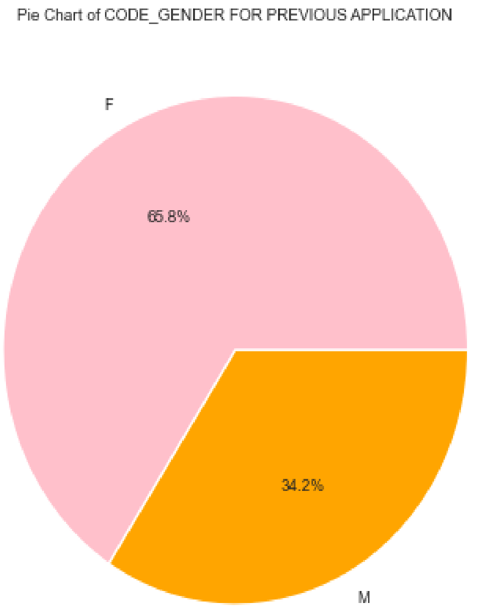
DEMOGRAPHIC CHARACTERISTICS

In [50]:
```
import matplotlib.pyplot as plt
colors = ['grey', 'blue', 'orange', 'pink']
filtered_df = df2[df2['NAME_EDUCATION_TYPE'] != 'XNA']
df2.NAME_EDUCATION_TYPE.value_counts(normalize=True).plot.bar(color=colors)
plt.title('Bar Plot of NAME_EDUCATION_TYPE FOR PREVIOUS APPLICATION')
plt.xlabel('NAME_EDUCATION_TYPE')
plt.ylabel('Normalized Counts')
plt.show()
```



DEMOGRAPHIC CHARACTERISTICS

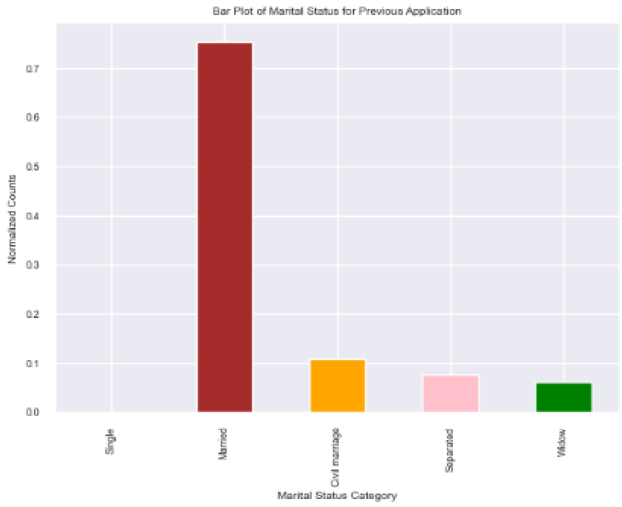
In [43]:
```python
import matplotlib.pyplot as plt

# Filter out 'XNA' values
filtered_df = df2[df2['CODE_GENDER'] != 'XNA']

# Count the occurrences of each gender
gender_counts = filtered_df['CODE_GENDER'].value_counts()

# Create a pie chart
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', colors=['pink', 'orange'])
plt.title('Pie Chart of CODE_GENDER FOR PREVIOUS APPLICATION')
plt.show()
```



In [66]:
```python
unique_family_statuses = df2['NAME_FAMILY_STATUS'].unique()
print(unique_family_statuses)
```

```
['Single / not married' 'Married' 'Widow' 'Civil marriage' 'Separated'
 'Unknown']
```

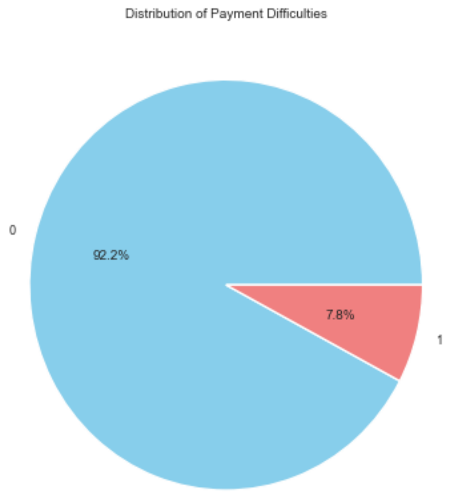
In [74]:
```python
import matplotlib.pyplot as plt

# Define family status categories
family_status_categories = ['Single', 'Married', 'Civil marriage', 'Separated', 'Widow']

# Assuming 'df2' is your DataFrame
df2['Family_Status_Category'] = pd.Categorical(df2['NAME_FAMILY_STATUS'], categories=family_status_categories, ordered=True)

# Filter out rows where 'NAME_FAMILY_STATUS' is 'XNA'
filtered_df = df2[df2['NAME_FAMILY_STATUS'] != 'XNA']

# Plot the bar chart
colors = ['red', 'brown', 'orange', 'pink', 'green']
filtered_df['Family_Status_Category'].value_counts(normalize=True).sort_index().plot.bar(color=colors)

plt.title('Bar Plot of Marital Status for Previous Application')
plt.xlabel('Marital Status Category')
plt.ylabel('Normalized Counts')
plt.show()
```



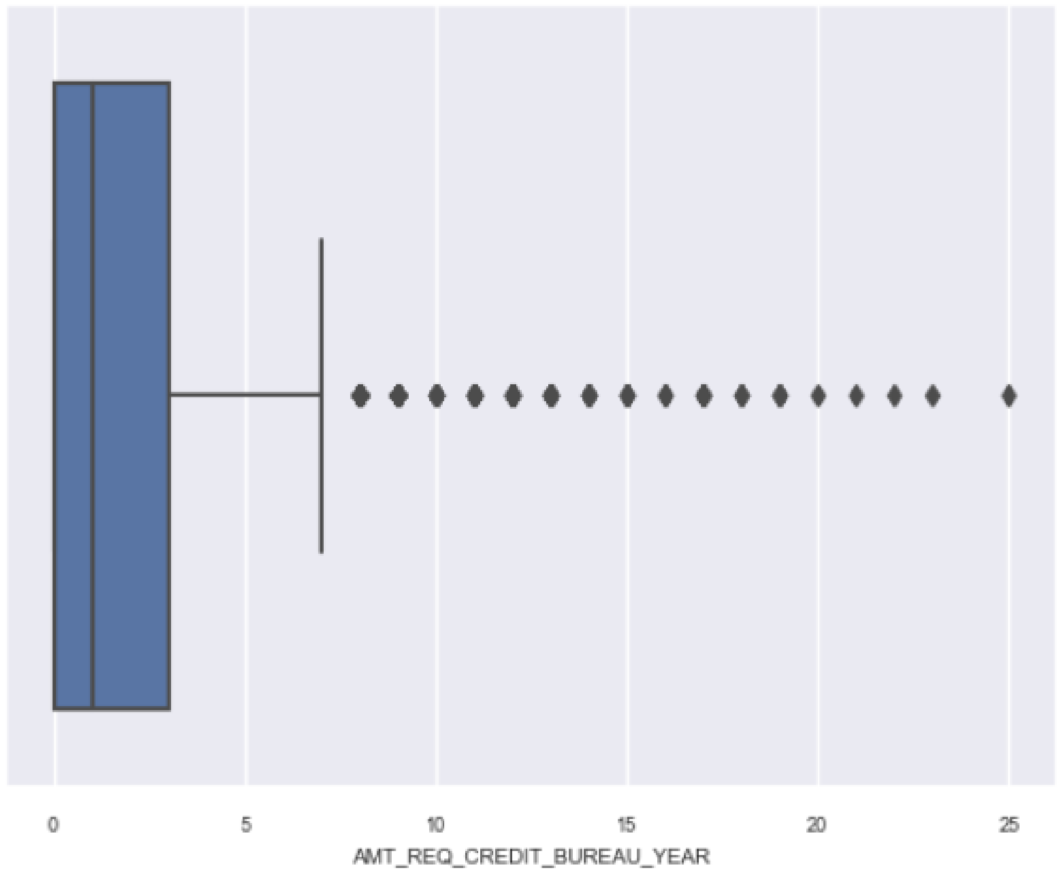
DISTRIBUTION OF PAYMENT DIFFICULTY BY CLIENT

In [42]:
```
# Dividing the dataset into two datasetof target=1(client with payment difficulties) and target=0(all other)
target0_df=df2.loc[df2['TARGET']==0]
target1_df=df2.loc[df2['TARGET']==1]

#Calculate imbalance percentage since majority is target zero and minority is target 1
round(len(target0_df)/len(target1_df),2)
```
Out[42]: 11.95

In [54]:
```
import matplotlib.pyplot as plt

# Assuming 'result' is your DataFrame
result.groupby('TARGET').size().plot(
    kind='pie',
    autopct='%1.1f%%',
    colors=['skyblue', 'lightcoral', 'green', 'orange', 'lightyellow']
)

plt.title('Distribution of Payment Difficulties by Client')
plt.show()
```



In [33]:
```
sns.boxplot(x=df2['AMT_REQ_CREDIT_BUREAU_YEAR'])
```
Out[33]: <Axes: xlabel='AMT_REQ_CREDIT_BUREAU_YEAR'>



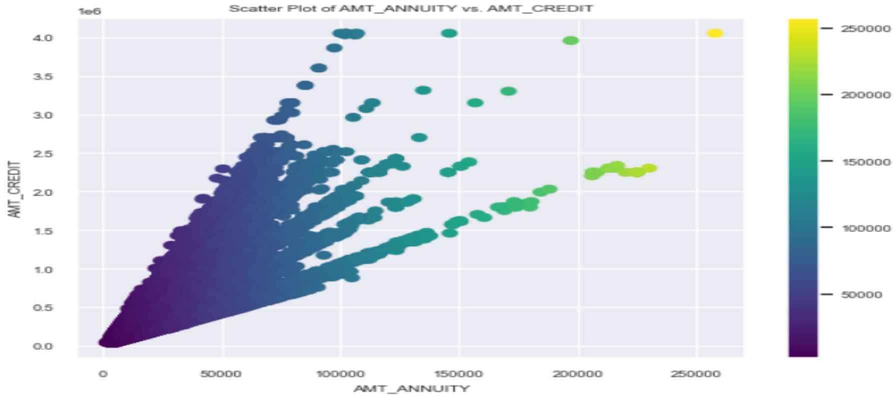
BIVARIATE ANALYSIS

CORRELATION ANALYSIS

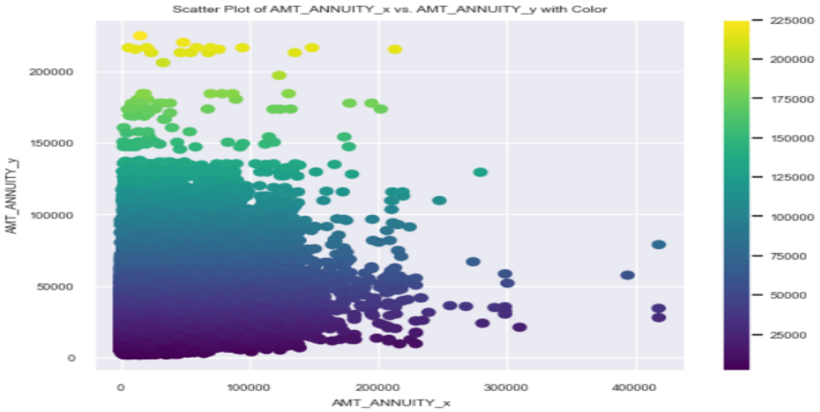
In [32]:
```
plt.scatter(df1.AMT_ANNUITY, df1.AMT_GOODS_PRICE, c=df1['AMT_ANNUITY'], cmap='viridis
plt.xlabel('AMT_ANNUITY')
plt.ylabel('AMT_GOODS_PRICE')
plt.title('Scatter Plot of AMT_ANNUITY vs. AMT_GOODS_PRICE')
plt.colorbar()  # Add a colorbar to show the mapping of colors to values
plt.show()
```

In [33]:
```
plt.scatter(df2.AMT_CREDIT, df2.AMT_INCOME_TOTAL, c=df2['AMT_CREDIT'], cmap='viridis')
plt.xlabel('AMT_CREDIT')
plt.ylabel('AMT_INCOME_TOTAL')
plt.title('Scatter Plot of AMT_CREDIT vs. AMT_INCOME_TOTAL')
plt.colorbar()  # Add a colorbar to show the mapping of colors to values
plt.show()
```

In [34]:
```
plt.scatter(df2.AMT_ANNUITY, df2.AMT_CREDIT, c=df2['AMT_ANNUITY'], cmap='viridis')
plt.xlabel('AMT_ANNUITY')
plt.ylabel('AMT_CREDIT')
plt.title('Scatter Plot of AMT_ANNUITY vs. AMT_CREDIT')
plt.colorbar()  # Add a colorbar to show the mapping of colors to values
plt.show()
```

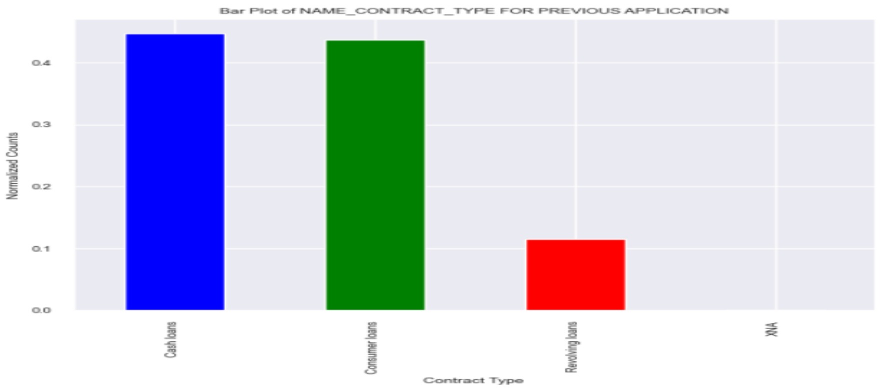


In [35]:
```
import matplotlib.pyplot as plt
merged_df = pd.merge(df1, df2, on='SK_ID_CURR', how='inner')
plt.scatter(merged_df.AMT_ANNUITY_x, merged_df.AMT_ANNUITY_y, c=merged_df['AMT_ANNUITY_y'], cmap='viridis')
plt.xlabel('AMT_ANNUITY_x')
plt.ylabel('AMT_ANNUITY_y')
plt.title('Scatter Plot of AMT_ANNUITY_x vs. AMT_ANNUITY_y with Color')
plt.colorbar()
plt.show()
```

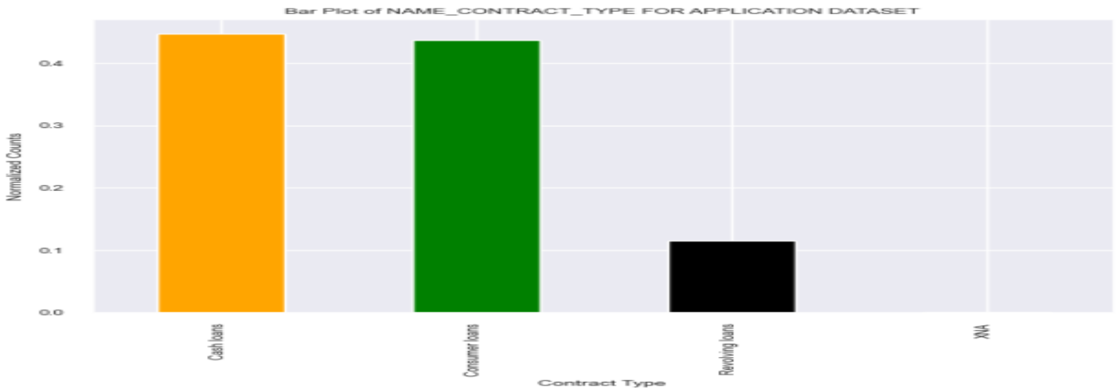


CORRELATION ANALYSIS

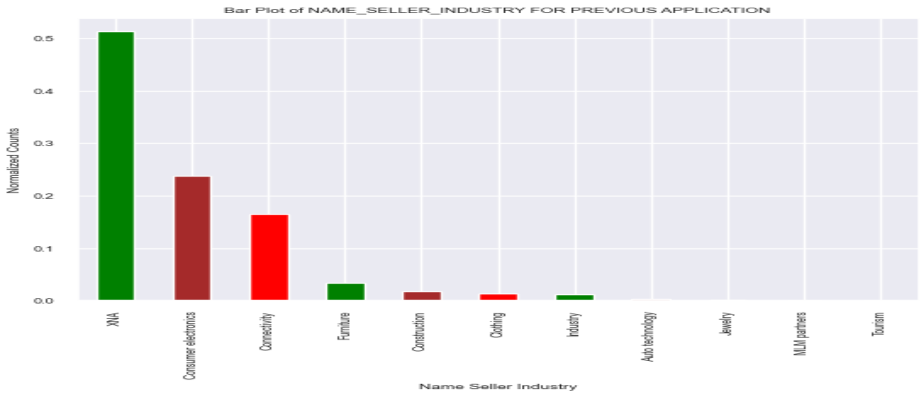
In [36]:
```
import matplotlib.pyplot as plt
colors = ['blue', 'green', 'red']
filtered_df = df1[df1['NAME_CONTRACT_TYPE'] != 'XNA']
df1.NAME_CONTRACT_TYPE.value_counts(normalize=True).plot.bar(color=colors)
plt.title('Bar Plot of NAME_CONTRACT_TYPE FOR PREVIOUS APPLICATION')
plt.xlabel('Contract Type')
plt.ylabel('Normalized Counts')
plt.show()
```



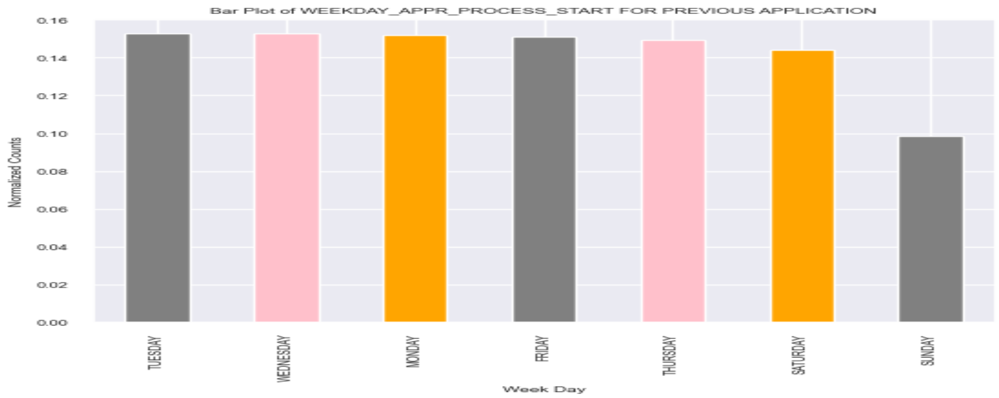
In [37]:
```
import matplotlib.pyplot as plt
colors = ['orange', 'green', 'black']
filtered_df = df2[df2['NAME_CONTRACT_TYPE'] != 'XNA']
df1.NAME_CONTRACT_TYPE.value_counts(normalize=True).plot.bar(color=colors)
plt.title('Bar Plot of NAME_CONTRACT_TYPE FOR APPLICATION DATASET')
plt.xlabel('Contract Type')
plt.ylabel('Normalized Counts')
plt.show()
```



In [38]:
```
import matplotlib.pyplot as plt
# Assuming you want different colors for each bar
colors = ['green', 'brown', 'red']
filtered_df = df1[df1['NAME_SELLER_INDUSTRY'] != 'XNA']
df1.NAME_SELLER_INDUSTRY.value_counts(normalize=True).plot.bar(color=colors)
plt.title('Bar Plot of NAME_SELLER_INDUSTRY FOR PREVIOUS APPLICATION')
plt.xlabel('Name Seller Industry')
plt.ylabel('Normalized Counts')
plt.show()
```

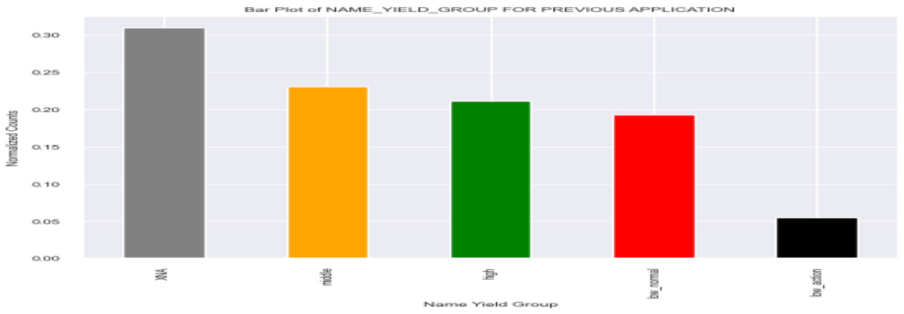


In [39]:
```
import matplotlib.pyplot as plt
colors = ['grey', 'pink', 'orange']
filtered_df = df1[df1['WEEKDAY_APPR_PROCESS_START'] != 'XNA']
df1.WEEKDAY_APPR_PROCESS_START.value_counts(normalize=True).plot.bar(color=colors)
plt.title('Bar Plot of WEEKDAY_APPR_PROCESS_START FOR PREVIOUS APPLICATION')
plt.xlabel('Week Day')
plt.ylabel('Normalized Counts')
plt.show()
```

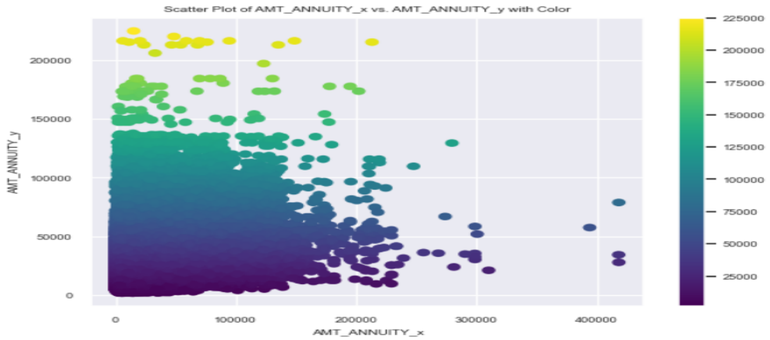


CORRELATION ANALYSIS

In [40]:
```python
import matplotlib.pyplot as plt
colors = ['grey', 'orange', 'green', 'red', 'black']
filtered_df = df1[df1['NAME_YIELD_GROUP'] != 'XNA']
df1.NAME_YIELD_GROUP.value_counts(normalize=True).plot.bar(color=colors)
plt.title('Bar Plot of NAME_YIELD_GROUP FOR PREVIOUS APPLICATION')
plt.xlabel('Name Yield Group')
plt.ylabel('Normalized Counts')
plt.show()
```



In [35]:
```python
import matplotlib.pyplot as plt
merged_df = pd.merge(df1, df2, on='SK_ID_CURR', how='inner')
plt.scatter(merged_df.AMT_ANNUITY_x, merged_df.AMT_ANNUITY_y, c=merged_df['AMT_ANNUITY_y'], cmap='viridis')
plt.xlabel('AMT_ANNUITY_x')
plt.ylabel('AMT_ANNUITY_y')
plt.title('Scatter Plot of AMT_ANNUITY_x vs. AMT_ANNUITY_y with Color')
plt.colorbar()
plt.show()
```

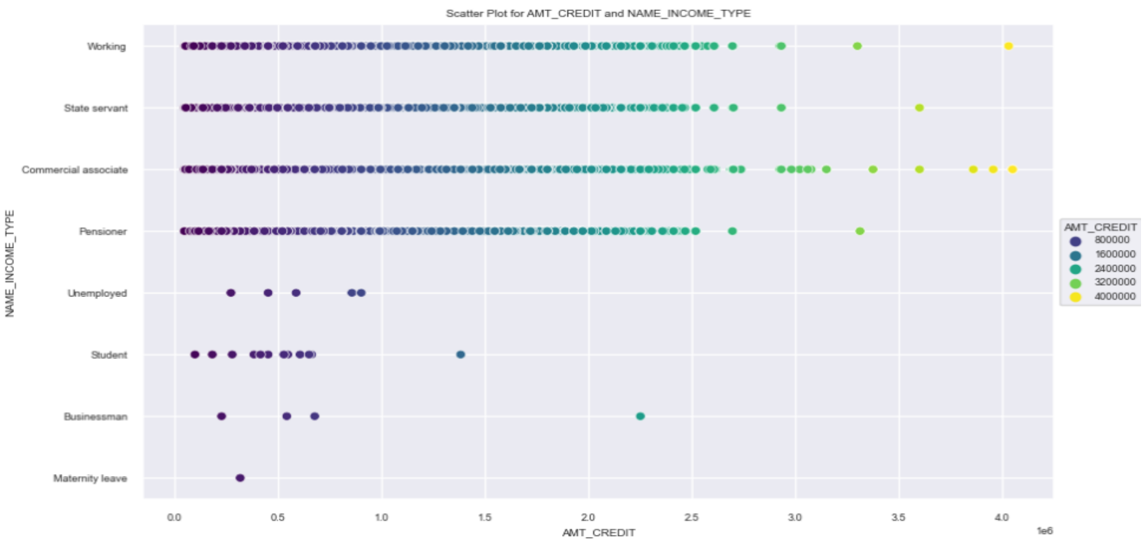


Bivariate analysis in Python is centered on examining connections between two variables. Utilizing libraries such as Pandas, Seaborn, and Matplotlib (plt), one can generate visualizations like histograms and box plots, offering a lucid portrayal of the distribution and central tendencies of the variables under consideration.

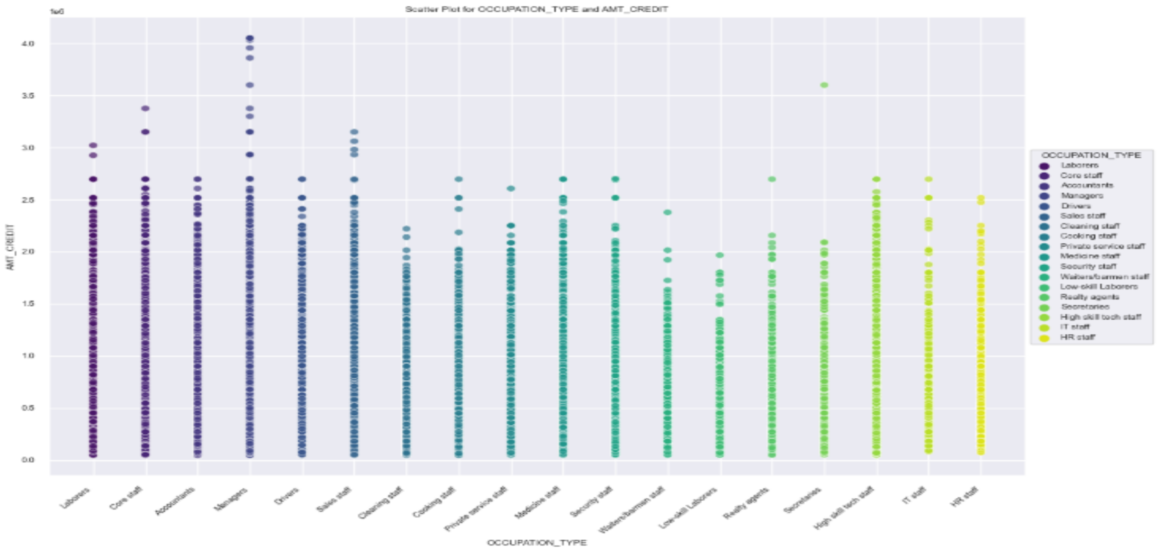
The majority of variables in the scatterplot above exhibit a positive correlation with each other. The steeper the gradient, the stronger the correlation between the variables. Specifically, the relationship between AMT_CREDIT and AMT_GOODS_PRICE displays the most linear pattern, suggesting that credit plays a significant role in influencing the purchasing behavior of high-value goods. This linearity implies that higher inflation rates may lead to increased prices of goods, while changes in interest rates can impact the cost of credit. On the other hand, AMT_INCOME_TOTAL shows a more random correlation with no distinct patterns.

EFFECT OF INCOME TYPE & OCCUPATION ON AMOUNT OF CREDIT

In [88]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'df2' is your DataFrame
plt.figure(figsize=(10, 6))
sns.scatterplot(x='AMT_CREDIT', y='NAME_INCOME_TYPE', data=df2, hue='AMT_CREDIT', palette='viridis')

plt.xlabel('AMT_CREDIT')
plt.ylabel('NAME_INCOME_TYPE')
plt.title('Scatter Plot for AMT_CREDIT and NAME_INCOME_TYPE')
plt.legend(title='AMT_CREDIT', loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```



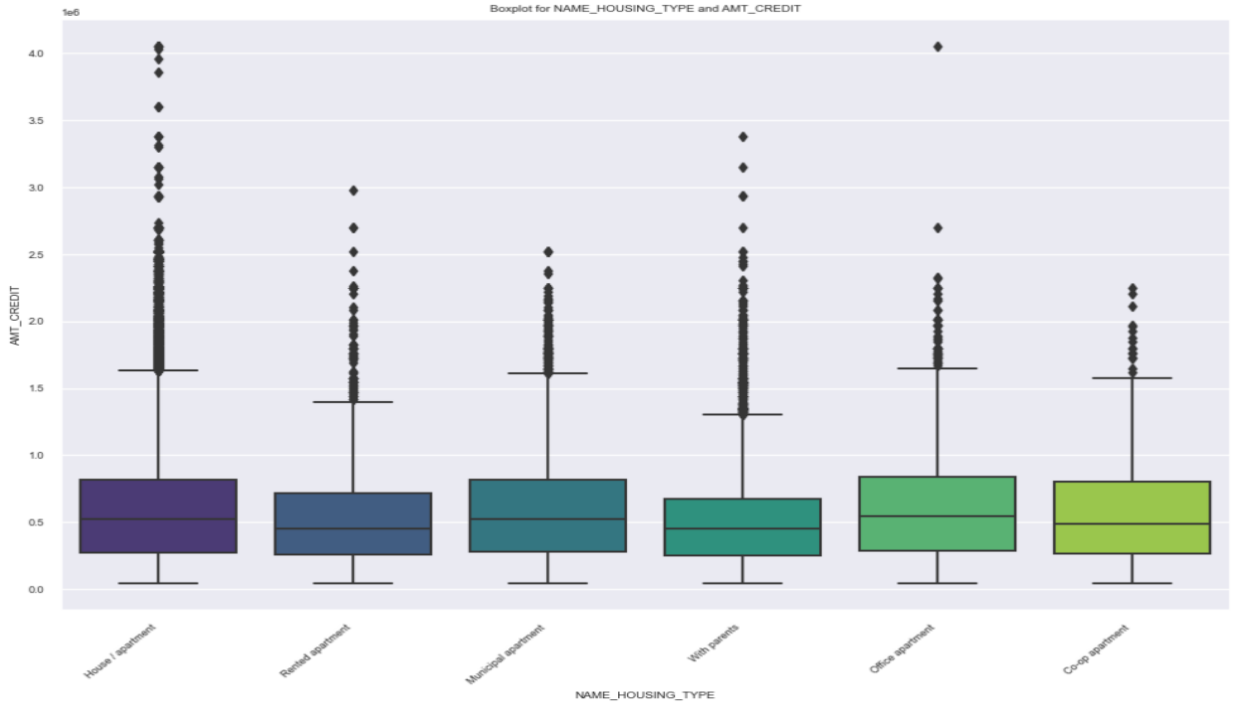
In [94]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'df2' is your DataFrame
plt.figure(figsize=(12, 8))
sns.scatterplot(x='OCCUPATION_TYPE', y='AMT_CREDIT', data=df2, hue='OCCUPATION_TYPE', palette='viridis', alpha=0.7)

plt.xlabel('OCCUPATION_TYPE')
plt.ylabel('AMT_CREDIT')
plt.title('Scatter Plot for OCCUPATION_TYPE and AMT_CREDIT')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.legend(title='OCCUPATION_TYPE', loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```



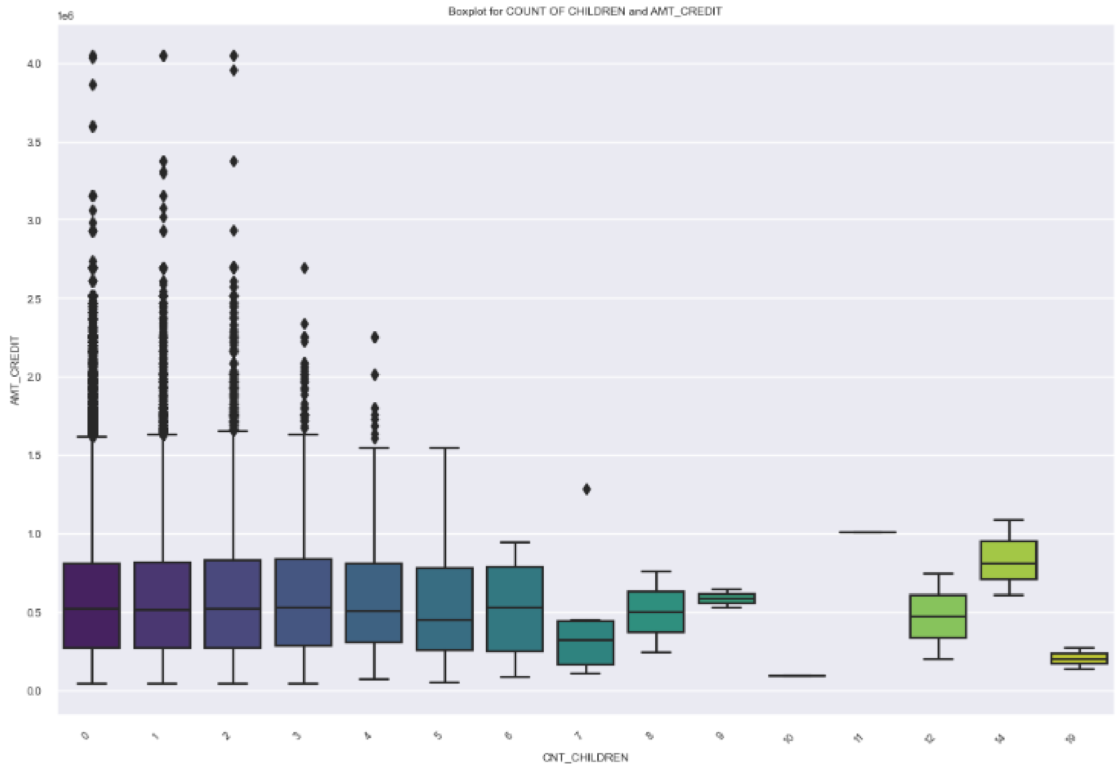
Individuals whose income type is commercial associates earn higher credit followed by individuals whose income type is working. Students, the unemployed and businessmen receive lower credit amounts. Women on maternity leave receive the lowest credit amount

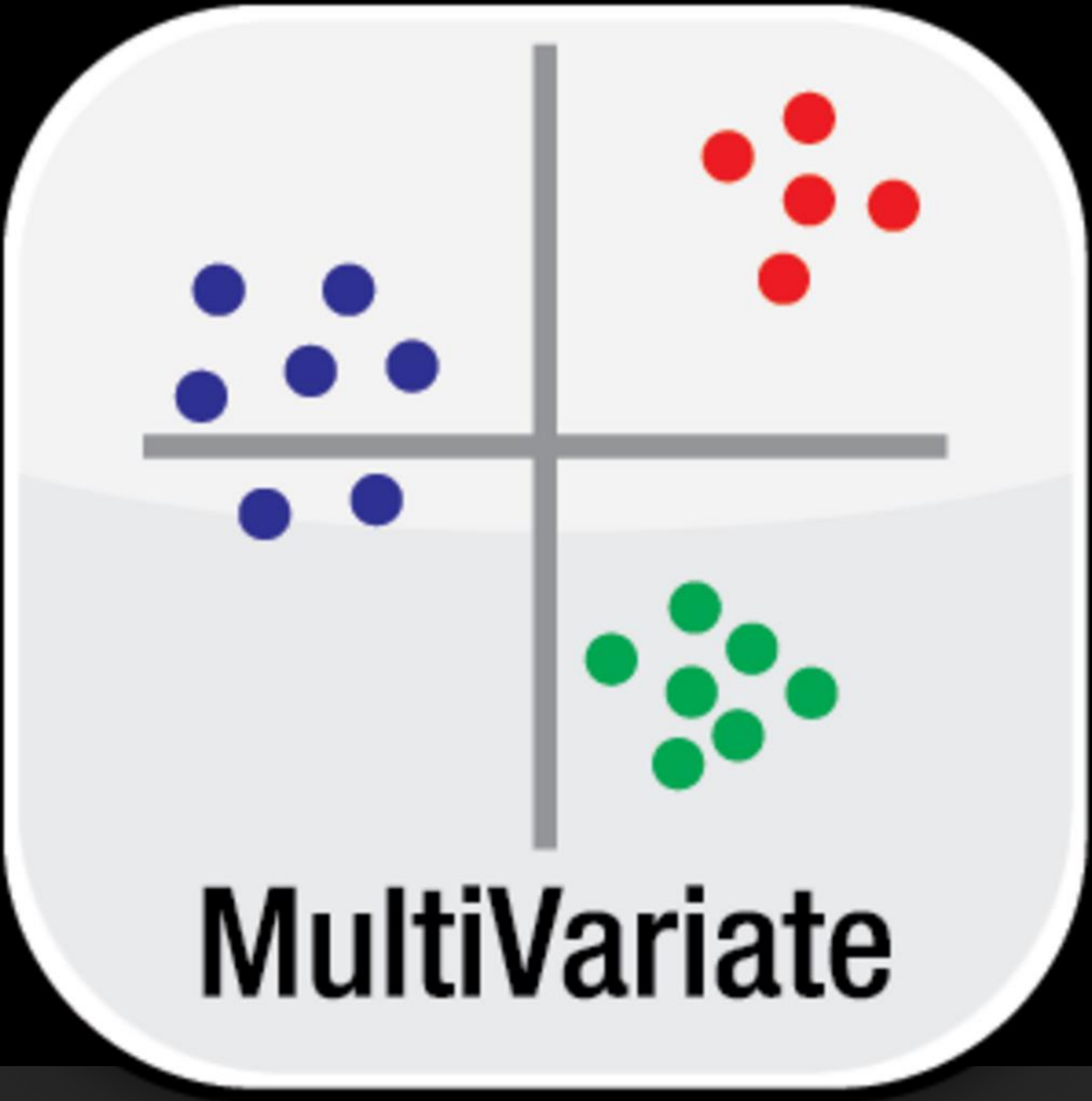
EFFECT OF HOUSING TYPE & NUMBER OF CHILDREN ON AMOUNT OF CREDIT RECEIVED

In [104]:
```
plt.figure(figsize=(12, 8))
sns.boxplot(x='NAME_HOUSING_TYPE', y='AMT_CREDIT', data=df2, palette='viridis')
plt.xlabel('NAME_HOUSING_TYPE')
plt.ylabel('AMT_CREDIT')
plt.title('Boxplot for NAME_HOUSING_TYPE and AMT_CREDIT')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.show()
```



In [105]:
```
plt.figure(figsize=(12, 8))
sns.boxplot(x='CNT_CHILDREN', y='AMT_CREDIT', data=df2, palette='viridis')
plt.xlabel('CNT_CHILDREN')
plt.ylabel('AMT_CREDIT')
plt.title('Boxplot for COUNT OF CHILDREN and AMT_CREDIT')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.show()
```





MULTIVARIATE ANALYSIS

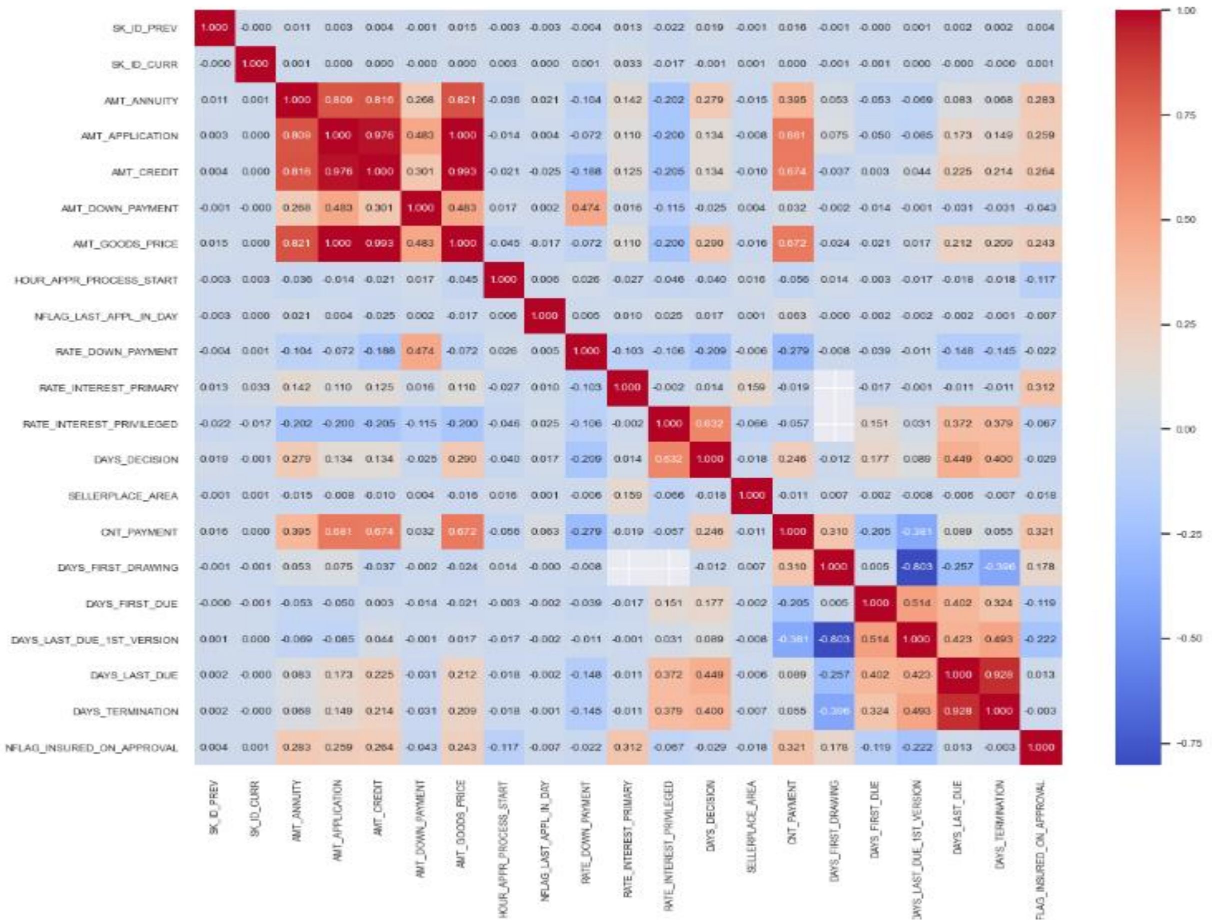
CORRELATION HEATMAP

```
In [26]: numeric_df = df1.select_dtypes(include=['float64', 'int64'])
         correlation_matrix = numeric_df.corr()
         correlation_matrix
```

Out[26]:

	SK_ID_PREV	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE
SK_ID_PREV	1.000000	-0.000321	0.011459	0.003302	0.003659	-0.001313	0.015293
SK_ID_CURR	-0.000321	1.000000	0.000577	0.000280	0.000195	-0.000063	0.000369
AMT_ANNUITY	0.011459	0.000577	1.000000	0.808872	0.816429	0.267694	0.820895
AMT_APPLICATION	0.003302	0.000280	0.808872	1.000000	0.975824	0.482776	0.999884
AMT_CREDIT	0.003659	0.000195	0.816429	0.975824	1.000000	0.301284	0.993087
AMT_DOWN_PAYMENT	-0.001313	-0.000063	0.267694	0.482776	0.301284	1.000000	0.482776
AMT_GOODS_PRICE	0.015293	0.000369	0.820895	0.999884	0.993087	0.482776	1.000000
HOUR_APPR_PROCESS_START	-0.002652	0.002842	-0.036201	-0.014415	-0.021039	0.016776	-0.045267
NFLAG_LAST_APPL_IN_DAY	-0.002828	0.000098	0.020639	0.004310	-0.025179	0.001597	-0.017100
RATE_DOWN_PAYMENT	-0.004051	0.001158	-0.103878	-0.072479	-0.188128	0.473935	-0.072479
RATE_INTEREST_PRIMARY	0.012969	0.033197	0.141823	0.110001	0.125106	0.016323	0.110001
RATE_INTEREST_PRIVILEGED	-0.022312	-0.016757	-0.202335	-0.199733	-0.205158	-0.115343	-0.199733
DAYS_DECISION	0.019100	-0.000637	0.279051	0.133660	0.133763	-0.024536	0.290422
SELLERPLACE_AREA	-0.001079	0.001265	-0.015027	-0.007649	-0.009567	0.003533	-0.015842
CNT_PAYMENT	0.015589	0.000031	0.394535	0.680630	0.674278	0.031659	0.672129
DAYS_FIRST_DRAWING	-0.001478	-0.001329	0.052839	0.074544	-0.036813	-0.001773	-0.024445

```
# Increase the figure size
plt.figure(figsize=(12, 10))

# Set a larger font scale
sns.set(font_scale=0.6)

# Plot the heatmap with annotations
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.3f')

# Show the plot
plt.show()
```



The correlation heatmaps consistently depict a correlation of 1, evident by a consistently dark green color in each row, signifying a positive correlation. Notably, AMT_ANNUITY and AMT_APPLICATION exhibited a strong correlation. The presence of multicollinearity is suggested, indicating that when multiple variables are highly correlated, it can impact regression models. In the context of positive correlation, an increase in one variable is associated with a tendency for other variables to increase as well.

CONCLUSION & SUMMARY

The bank experienced a greater influx of loan applications from female customers compared to their male counterparts. Most of the loan requests come from individuals aged between 30 and 40, with the 40-50 age group closely trailing behind. A notable proportion of applicants sought cash loans, while the demand for revolving loans was relatively modest. In terms of occupational categories, the largest portion of applicants belongs to the working class, followed by commercial associates and state servants. The prevailing educational background among loan applicants is Secondary/Secondary Special, followed by Higher Education. The majority of applicants seeking relatively larger loan amounts belong to the married category in both Defaulters and Non – Defaulters.

There was a considerable reduction in the proportion of cash loans, accompanied by a shift in the purchasing trend towards higher-value goods using cash loans in the recent dataset compared to the previous one. The average repayment term also exhibited a significant uptick between the two sets of data. The heatmap analysis revealed a negative correlation between income and credit, indicating that as income decreases, credit tends to increase. In contrast, there was a notable positive correlation between credit and annuity variables, suggesting a relationship where higher credit is associated with increased annuity. The identification and removal of outliers across various loan types aimed to enhance the accuracy and clarity of the dataset representation.

RECOMMENDATIONS

- Continuous monitoring of applications with exceptionally high annuity amounts is imperative for banks to proactively identify potential risks, based on historical data.
- The ability to identify and respond to potential risks within loan applications is foundational to a proactive risk management approach that safeguards the financial stability of the bank. In response, strategic measures such as reducing the loan amount or applying a higher interest rate should be considered by banks to mitigate potential risks associated with applicants with dependents.
- Evaluating a borrower's ability to meet financial obligations should encompass a thorough analysis of credit amount and annuity payments, referencing historical data for comprehensive insights.
- Applicants with high credit scores showcase robust creditworthiness, prompting the bank to consider approving loans for this segment, aligning with a lower-risk profile.
- Monitoring and adapting to changing risk factors is crucial for banks to make informed and dynamic decisions, ensuring the overall health of their lending portfolios.

THANK YOU