Base Table Creation using Python Language

Financial Programming Group 7

SERHAN Yasmina

YUAN Quan

PERAN Mathieu

Table of Contents

Introduction

In the course of preparing our data for future analysis for the financial institutions as well as the clients, we meticulously performed various operations to enhance its quality, relevance, and suitability for future possible modeling. The data preparation process involved cleaning, transforming, and aggregating information from different tables to construct a comprehensive and detailed basetable for subsequent analysis.

Data Preparation

In the process of data preparation for our small tables before merging them into the base table, several key operations were performed to ensure the quality and relevance of the data. The following steps outline the data correction and transformation methods applied in the Jupyter Notebook table by table:

Account

- Created an ‘opening_year’ column extracted from ‘date’
- To ensure that we have sufficient information for our independent variables (a full year in 1996), the table was filtered to only include the accounts that were opened before 1996.
- Created ‘lor’ which is the length of relationship with 1996 as the reference year
- Checked the different categories in frequency (3) and mapped them into more meaningful values
- Created and fixed the type of ‘opening_date’ extracted from opening year

Client

- Extracted ‘birth_year’, ‘birth_day’ and ‘birth_month’ from ‘birth_number’ and changed the type to integer
- Extracted ‘gender’ from ‘birth_number’ where the gender was ‘M’ if the month was below 50 and ‘F’ otherwise
- Consequently, ‘birth_month’ was corrected for the female clients by subtracting 50 from it.
- Dropped ‘birth_number’
- Created ‘age’ from ‘birth_year’ with 1996 as the reference year

- Created ‘age_group’ from ‘age’

Disposition

- Filtered the data of this table to only include ‘OWNER’ from ‘type’\

Transaction

- When checking for the categories in ‘type’, it was noticed that there are 3 categories whereas the data description specifies only 2 (credit or withdrawal). To further inspect this, the value in ‘operation’ was checked for all the rows with ‘type’ ‘VYBER’ (the third category). The results returned that the operation for all of these rows was also ‘VYBER’ (withdrawal in cash), so a decision was made to merge the 2 withdrawal categories in ‘type’ into 1. Thus, ‘type’ was mapped into 2 categories.
- The categories in ‘operation’ (5 categories) were mapped to numbers. The missing values in this column were mapped into a 6th category to be considered as ‘other’.
- Similarly for ‘k_symbol’, the 7 of its categories were mapped to numbers and the missing values were mapped to an 8th category to be considered as ‘other’.
- In an attempt to deal with the missing values of ‘bank’ and ‘account’, a theory was tested to check whether a row with a missing value in ‘bank’ has a credit transaction ‘type’. This would justify not having a bank of the partner. However, the theory was incorrect, and for the purposes of this project, the values were left as NaN because only the unique count was required from ‘bank’ and ‘account’.
- Created ‘trans_date’ from ‘date’
- Filtered the data of this table by only including transactions that occurred in the year 1996
- Initiated a dataframe that will be merged with the base table and created new variables which are a result of aggregation of the transaction table on account_id. These variables will be explained in the base table description at the end.

Order

- To filter the order table in a way to prevent data leakage, the aim was to only keep the account_id-amount-bank-account groupings that are also found in trans_1996. Since the

order table has no information regarding date, the filtering for this data was a bit trickier. If we only left join to the trans_1996 table, the data would be filtered to include only accounts who have had transactions in 1996. However, this would not prevent orders that were newly placed post 1996 from being included. To clarify, an account_id 123 that exists in the trans_1996 table could have a new permanent order in 1997 related to insurance payment, loan payment etc. This will be considered a form of data leakage. Thus, the preferred method here was to filter based on groupings that were found in 1996 (account_id-amount-bank-account). This method ensured that the kept orders took place in 1996 (they might have also taken place before that year and even in the future, the important thing is to not include orders that have no relation to 1996).

- Renamed columns to 'bank_to' and 'account_to' for the recipients
- Mapped the 4 'k_symbol' categories into more meaningful values, where the 4th category was an empty string and was replaced by 'OTHER'
- Initiated a dataframe that will be merged with the base table and created new variables which are a result of aggregation of the filtered_order table on account_id. These variables will be explained in the base table description at the end.

Loan

- Created 'year' from 'date'
- Filtered the data for this table by keeping only the loans granted before 1997. The logic behind this is to keep track of the status of loans granted for each account, as this is something to consider for the dependent variables.
- Dropped 'status' since this data extracted is from 1999 and including it will be considered data leakage. Another 'status' will be calculated with 31 December 1996 as the reference date.
- Checked if an account could have been granted more than 1 loan, it is not the case.
- Calculated 'total_expected_paid_amount' to calculate how much of the loan should be covered by 31 December 1996.
- Calculated 'loan_actual_tot_paid' to see how much was actually covered by 31 December 1996. To achieve this, the main transaction table was filtered for all data before 1997 and having a k_symbol equivalent to loan payment. Then, each account was checked to see if it has a unique loan payment (i.e check if the amount in the transaction is the same for each account). There is one account_id with different transaction amounts.

Comparing the data for this account in the loan and transactions table, it was noticed that only 1 out of 7 transactions is different, with NaN value for ‘bank’. Thus, it was safe to consider this as a different type of loan payment and it was dropped for this step. Then, the number of loan transactions ‘all_loan_payment_trans’ was obtained and this was used to calculate ‘loan_actual_tot_paid’ by multiplying it by ‘loan_payments’.

- Created a ‘status’ column identical to the one in the original loan table, but with 31 December 1996 as a reference.

Card

- Used the .rename() function to rename columns that were unclear to avoid misunderstandings in the final base table. (e.g. card1996.rename(columns={'type': 'card_type'})
- Fixed date format by pd.to_datetime() to keep it consistent with other dates in the base table.
- Created 'card_issue_weekday' column as a new variable by using the function .dt.weekday, and then replaced the numbers with a mapping of text of the weekdays for clearer expression of the base table.
- Applied filtering based on time window requirements, filtering cards issued before 1997 to have more data points support the future modeling.
- Checked the missing values of the card table by .isnull().sum() which there is none.
- Checked there are no duplicates for 'disp_id' by the function .duplicated().sum(), and merged with the data frame df2 by left join.

District

- Renamed all the columns to have a readable column name by the information provided by .rename().
- Dropped columns 'unemploy_rate95' and 'no_crimes95' because the time window of our possible independent variables should be in 1996.
- Checked the missing values of the card table by .isnull().sum() which there is none.

- Detected outliers by +/- 3 standard deviations as lower and upper bounds, and there was nothing printed out which means that there are no outliers in numerical variables of district1996 dataset.
- Merged the cleaned and filtered district1996 table with data frame df2 by left join and renamed the other dictrict_id as 'bank_district_id' and 'client_district_id' for clearer structure and better understanding to avoid confusion for the final base table df2.

Old and New Variables in the Final Base Table

List of Potential Numerical IVs

'birth_year', 'birth_day', 'birth_month', 'age', 'age_group', 'opening_year', 'lor', 'tot_transactions', 'withdrawals_percentage', 'avg_monthly_trans', 'avg_monthly_credits', 'avg_monthly_withdrawals', 'tot_amount', 'avg_monthly_amount', 'tot_amount_credits', 'avg_monthly_amount_credits', 'tot_amount_withdrawals', 'avg_monthly_amount_withdrawal', 'max_amount_credits', 'min_amount_credits', 'max_amount_withdrawals', 'min_amount_withdrawals', 'tot_op_creditcard_withdrawal', 'tot_amount_op_creditcard_withdrawal', 'tot_op_credit_cash', 'tot_amount_op_credit_cash', 'tot_op_collection_otherbank', 'tot_amount_op_collection_otherbank', 'tot_op_withdrawal_cash', 'tot_amount_op_withdrawal_cash', 'tot_op_remittance_otherbank', 'tot_amount_op_remittance_otherbank', 'tot_op_other', 'tot_amount_op_other', 'per_op_creditcard_withdrawal', 'per_op_credit_cash', 'per_op_collection_otherbank', 'per_op_withdrawal_cash', 'per_op_remittance_otherbank', 'per_op_other', 'balance_after_first_trans', 'balance_after_last_trans', 'activity_duration_days', 'account_growth', 'tot_k_insurrance_payment', 'tot_amount_k_insurrance_payment', 'tot_k_statement_payment', 'tot_amount_k_statement_payment', 'tot_k_interest_credited', 'tot_amount_k_interest_credited', 'tot_k_sanction_interest', 'tot_amount_k_sanction_interest', 'tot_k_household', 'tot_amount_k_household', 'tot_k_oldage_pension', 'tot_amount_k_oldage_pension', 'tot_k_loan_payment', 'tot_amount_k_loan_payment', 'tot_k_other', 'tot_amount_k_other', 'per_k_insurrance_payment', 'per_k_statement_payment', 'per_k_interest_credited', 'per_k_sanction_interest', 'per_k_household', 'per_k_oldage_pension', 'per_k_loan_payment', 'per_k_other', 'banks_count', 'tot_orders', 'order_tot_insurrance_payment', 'order_avg_amount_insurrance_payment', 'order_tot_household_payment', 'order_avg_amount_household_payment', 'order_tot_loan_payment', 'order_avg_amount_loan_payment', 'order_tot_other', 'order_avg_amount_other', 'loan_amount', 'loan_duration', 'loan_payments', 'loan_total_expected_paid_amount', 'loan_num_trans', 'loan_actual_tot_paid', 'no_of_inhabitants', 'no_of_inhabitants<499', 'no_of_inhabitants_500-1999', 'no_of_inhabitants_2000-9999', 'no_of_inhabitants>10000',

'no_of_cities', 'ratio_urban_inhabitants', 'avg_salary', 'unemploy_rate96', 'no_enterp_per_1000_inhabitants', 'no_crimes96'

List of Date Variables

'opening_date', 'first_trans_date', 'last_trans_date', 'loan_date', 'loan_expected_end_date', 'card_issued'

List of Primary Keys (IDs)

'client_id', 'client_district_id', 'account_id', 'bank_district_id', 'disp_id', 'loan_id', 'card_id'

List of Potential Categorical IVs

gender', 'frequency', 'loan_status', 'card_type', 'card_issue_weekday', 'district_name', 'region'

List of Dependent Variables/ Target Variables

TV1: loan granted in 1997

TV2: credit card issued in 1997

Visualization and Statistics

In order to have a clearer structure and better understanding of the important data of the final base table, as well as to discover the potential hypotheses for the future modeling, visualizing data and organizing the descriptive statistics of variables are crucial to data processing. Our visualizations and statistical tables are generated as following:

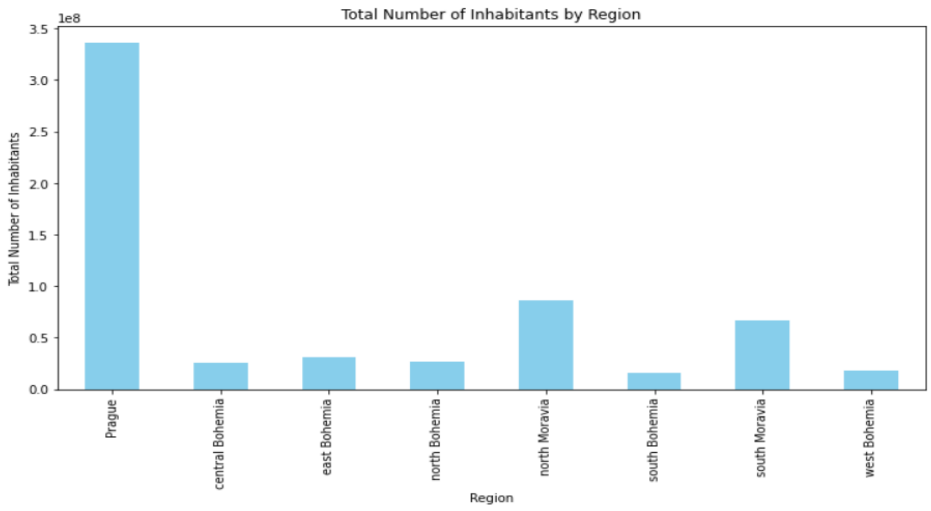
Graphs and Statistics of District and Financial Information

Table 1: Aggregated Crucial Data by Region

	region	no_of_inhabitants	no_of_cities	ratio_urban_inhabitants	unemploy_rate96	no_crimes96	avg_salary
0	Prague	1204953	1	100.000000	0.430000	99107	12541.000000
1	central Bohemia	1105234	66	52.441667	2.878333	43447	9357.250000
2	east Bohemia	1234781	86	62.945455	2.983636	30646	8611.181818
3	north Bohemia	1178977	70	80.200000	5.857000	48380	9334.200000
4	north Moravia	1970302	60	65.145455	5.697273	67917	9049.181818
5	south Bohemia	700595	48	61.050000	2.813750	18173	8831.500000
6	south Moravia	2054989	82	54.864286	3.828571	53841	8728.500000
7	west Bohemia	859306	69	65.690000	2.646000	25863	9015.400000

We used .agg(agg_dict).reset_index() to have the total number of inhabitants, cities, crimes and the average of the ratio of urban inhabitants, unemployment rate in 1996 as well as the average salary of each region in order to have a general view of the demographics of 8 regions.

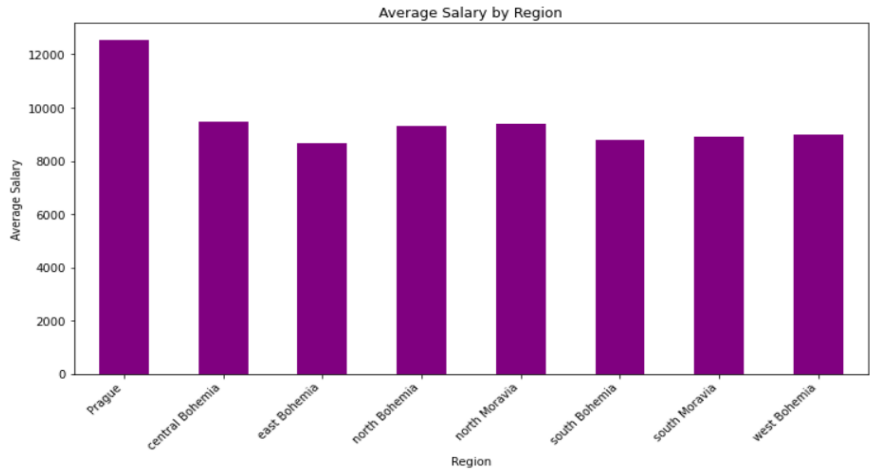
Graph 1: Total Number of Inhabitants by Region



We used .groupby() and .sum() function for variable 'region' and 'no_of_inhabitants' to plot a bar chart by using .plot(kind='bar', color='skyblue') in order to take a look at the total inhabitants by

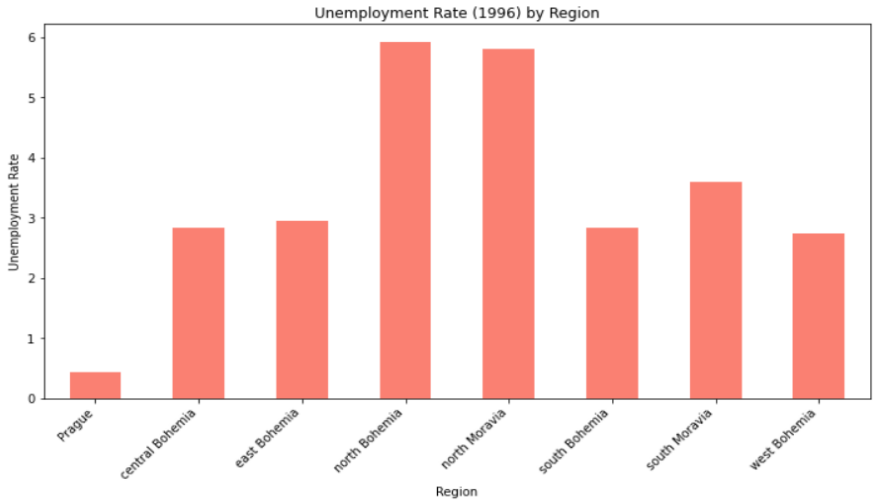
region. And the graph indicates that Prague has dramatically more inhabitants than other regions which indicates clients who apply for loans, cards and accounts are more likely from Prague.

Graph 2: Average Salary by Region



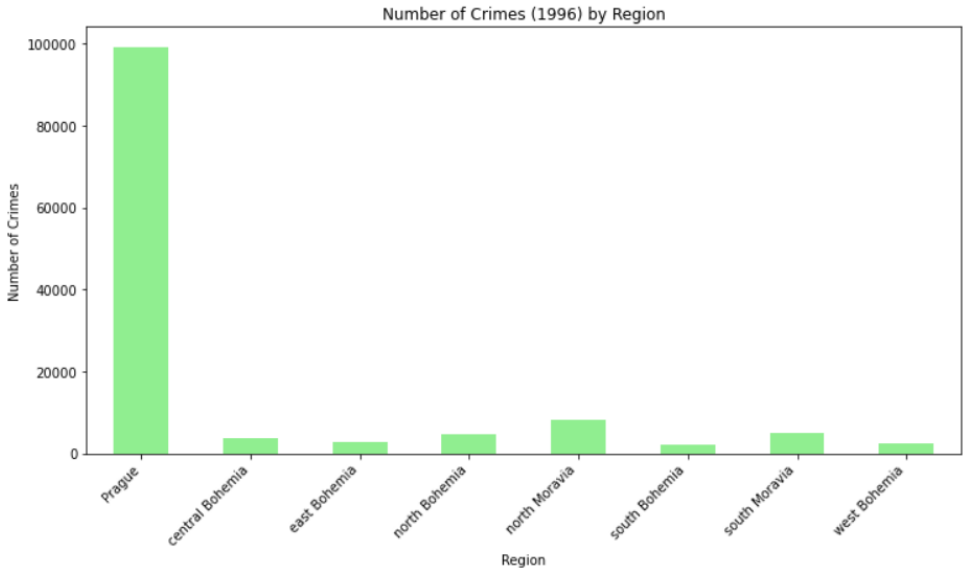
For this graph we also used .groupby() , .mean() and .plot(kind = 'bar') to plot a bar chart of average salary by region. It shows that Prague has notably higher average salary than other regions which means it can be a hypothesis that clients from Prague might have lower default risk.

Graph 3: Unemployment Rate in 1996 by Region



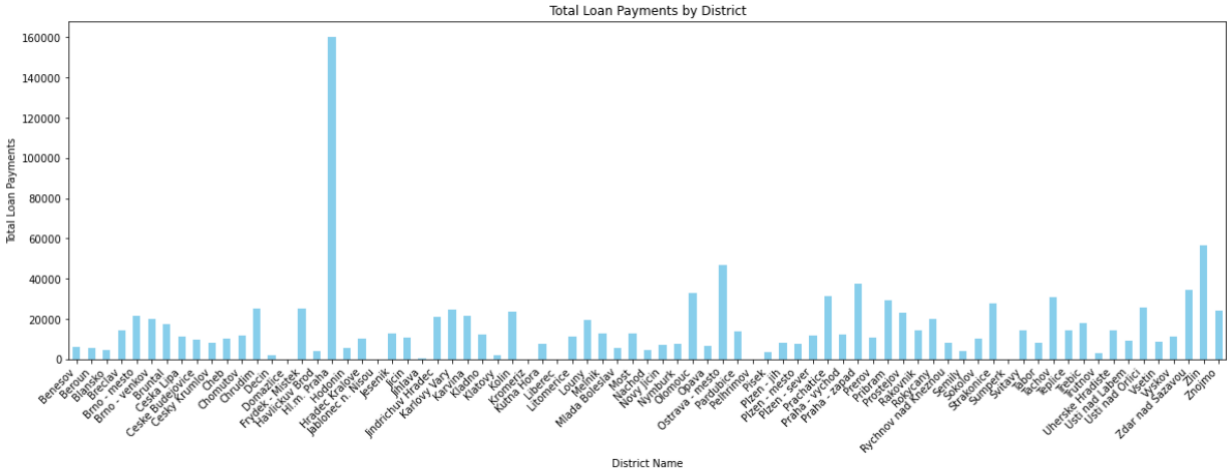
North Bohemia and north Moravia had the highest unemployment rate in 1996, while Prague had the lowest, which means that clients from Prague might have lower default risk.

Graph 4: Number of Crimes in 1996 by Region



Prague region had the highest number of crimes in 1996 that is dramatically higher than other regions with almost 100000 cases and hypothetically it can be a reason leading to high default risks.

Graph 5: Total Loan Payments by District



This graph is grouped by district_name instead of region, and it can be seen that Praha district has the highest amount of loan payments before 1997 that is dramatically higher than other

regions with almost 160000 euros. We can guess that it was probably due to the bigger population.

Table 2: Total Amount of Loan Payments by District

district_name	loan_payments
Hl.m. Praha	160152.0
Zlin	56538.0
Ostrava - mesto	46517.0
Praha - zapad	37548.0
Zdar nad Sazavou	34204.0
Olomouc	33047.0
Prachatice	31440.0
Tachov	30740.0
Pribram	29470.0
Strakonice	27856.0
Usti nad Orlici	25674.0
Chrudim	25237.0
Frydek - Mistek	25075.0
Karlovy Vary	24588.0
Znojmo	24324.0
Kolin	23548.0
Prostejov	23144.0
Brno - mesto	21569.0
Karvina	21251.0
Jindrichuv Hradec	20923.0
Rokycany	20045.0
Brno - venkov	19822.0
Louny	19172.0
Trebic	17670.0
Bruntal	17514.0
Uherske Hradiste	14440.0
Breclav	14338.0
Rakovnik	14331.0
Svitavy	14198.0
Teplice	14170.0
Pardubice	13883.0
Melnik	12907.0
Jesenik	12586.0
Most	12515.0
Praha - vychod	12235.0
Kladno	12048.0
Chomutov	11762.0

Use pd.DataFrame to show the table, and use .sort_values(by = ['loan_payments'], ascending = False) to check from the highest amount to the lowest. Also, using pd.set_option('display.max_rows', None) and pd.set_option('display.max_columns', None) to show all the rows and columns.

Table 3: Region vs. Loan Granted Possibility

```
region
west Bohemia       0.101382
east Bohemia       0.098113
south Moravia      0.095718
north Bohemia      0.092920
north Moravia      0.092269
south Bohemia      0.086486
Prague             0.068100
central Bohemia    0.063197
Name: tv1_loan_granted_97, dtype: float64
```

Using df2.groupby('region')['tv1_loan_granted_97'].agg('mean').sort_values(ascending=False) to show the default probability of each region. It seems that west Bohemia had a slightly higher possibility of loaning than other regions.

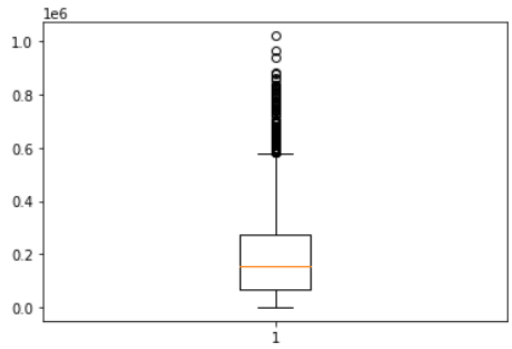
Table 4: Region vs. Issuance Probability of Credit Card

```
region
south Moravia      0.120907
central Bohemia    0.118959
north Moravia      0.117207
north Bohemia      0.110619
east Bohemia       0.109434
Prague             0.096774
west Bohemia       0.087558
south Bohemia      0.081081
Name: tv2_card_issued_97, dtype: float64
```

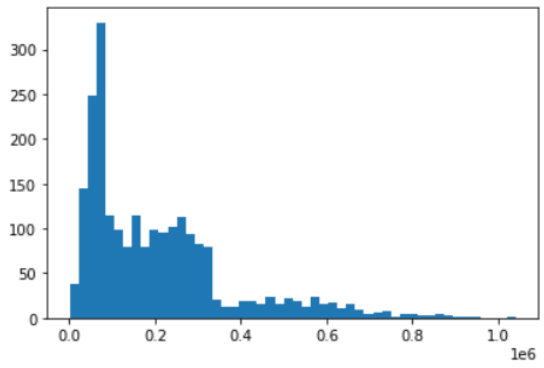
Using df2.groupby('region')['tv2_card_issued_97'].agg('mean').sort_values(ascending=False) to show the issuance probability of each region. It indicates that clients from south Moravia had a slightly higher chance to be issued with credit cards.

Graphs and Statistics of Clients and Financial Information

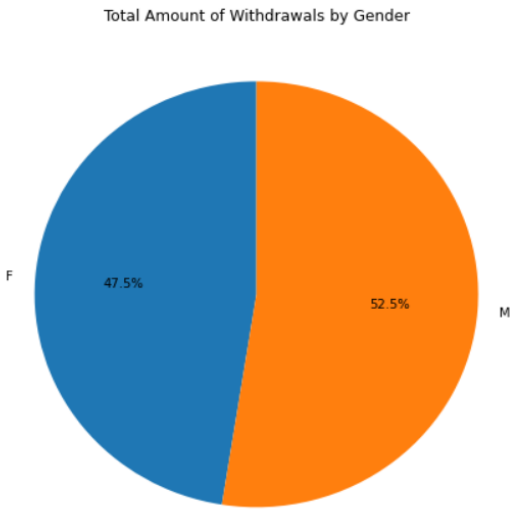
Graph 6: Box Plot of Total Amount of Withdrawals



Graph 7: Histogram of Total Amount of Credits



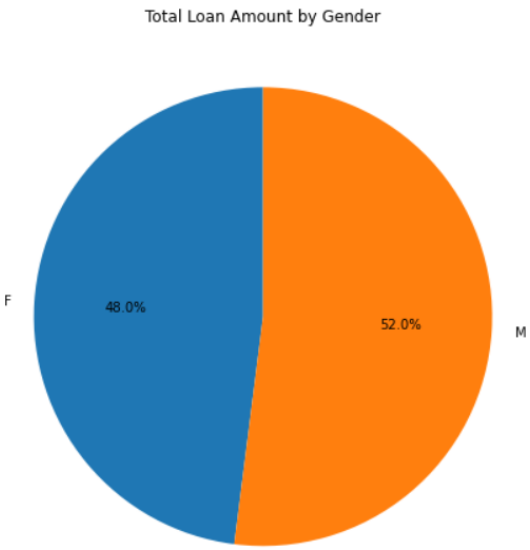
Graph 8: Pie Chart of Total Amount of Withdrawals by Gender



gender	tot_amount_withdrawals
F	215146804.3
M	237697185.2

We used .groupby(), .sum() and plt.pie() to plot this pie chart, and it shows that males withdrew slightly more money than females in 1996.

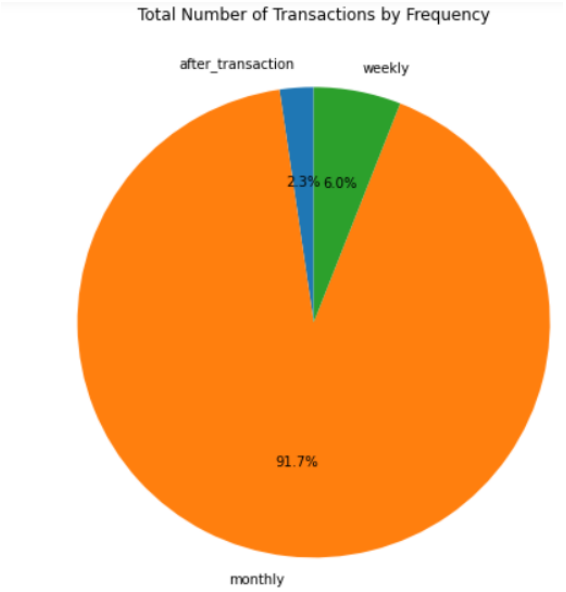
Graph 9: Pie Chart of Total Loan Amount by Gender



gender	loan_amount
F	2.075381e+07
M	2.248944e+07

We used .groupby(), .sum() and plt.pie() to plot this pie chart, and it shows that males loaned slightly more money than females before 1997.

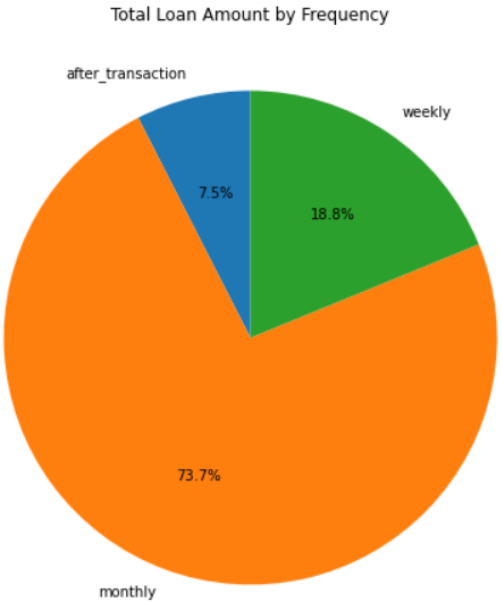
Graph 10: Pie Chart of Total Number of Transactions by Frequency



frequency	tot_transactions
after_transaction	3786
monthly	150273
weekly	9772

The pie chart shows clients with monthly issuance statements did most of the transactions in 1996.

Graph 11: Pie Chart of Total Loan Amount by Frequency



frequency	loan_amount
after_transaction	3.250716e+06
monthly	3.185232e+07
weekly	8.140211e+06

It indicates that clients with monthly issuance statements loan more money than other two groups with 73.7% of the total loan amount.

Descriptive Statistics: Age, Total Transactions and Average Monthly Amount of Withdrawal

```
count    2239.000000
mean       42.056722
std        17.102929
min        14.000000
25%        27.000000
50%        41.000000
75%        55.000000
max        78.000000
Name: age, dtype: float64
count    2239.000000
mean       73.171505
std        17.699459
min         5.000000
25%        63.000000
50%        72.000000
75%        83.000000
max       142.000000
Name: tot_transactions, dtype: float64
count     2239.000000
mean     16925.283951
std      14236.075605
min        438.333333
25%       5885.350000
50%      13168.266667
75%      22837.775000
max      85080.850000
Name: avg_monthly_amount_withdrawal, dtype: float64
```

Table 5: Gender vs. Loan Granted Possibility

```
gender
F    0.098708
M    0.077056
Name: tv1_loan_granted_97, dtype: float64
```

Using df2.groupby('gender')['tv1_loan_granted_97'].agg('mean').sort_values(ascending=False) to show the default probability by gender, and it seems that female clients are more likely to be granted with their loans.

Table 6: Gender vs. Issuance Probability of Credit Card

```
gender
F    0.130996
M    0.086580
Name: tv2_card_issued_97, dtype: float64
```

Using df2.groupby('gender')['tv2_card_issued_97'].agg('mean').sort_values(ascending=False) to show the issuance probability of males and females, and it indicates females had a higher chance to get credit cards.

Table 7: Frequency vs.Loan Granted Possibility

```
frequency
weekly               0.368852
after_transaction    0.354167
monthly              0.112615
Name: granted_loan, dtype: float64
```

Using df2.groupby('frequency')['granted_loan'].agg('mean').sort_values(ascending=False) to show the default probability by frequency, and it seems that clients had weekly issuance statements and had that after transactions are more likely to apply their loans successfully.

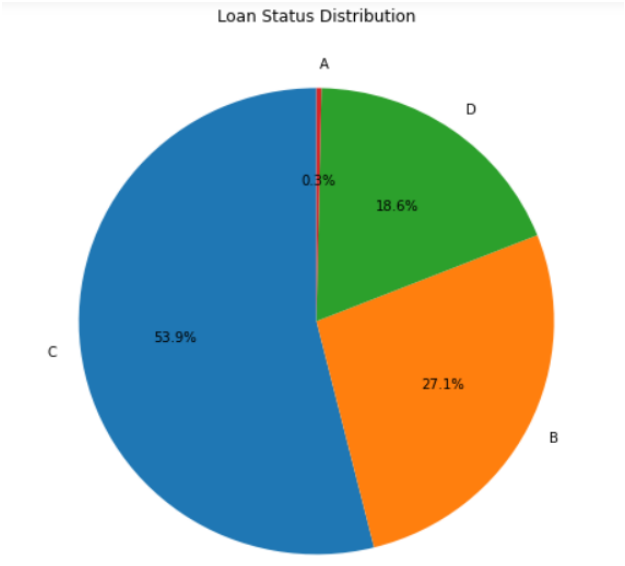
Table 8: Frequency vs. Issuance Probability of Credit Card

```
frequency
after_transaction    0.145833
weekly               0.131148
monthly              0.081199
Name: credit_card_issued, dtype: float64
```

Using df2.groupby('frequency')['credit_card_issued'].agg('mean').sort_values(ascending=False) to show the issuance probability of frequency of the issuance statements, and it indicates t clients had weekly issuance statements and had that after transactions had a higher chance to get credit cards.

<u>Graphs and Statistics of Loans and Cards Information</u>

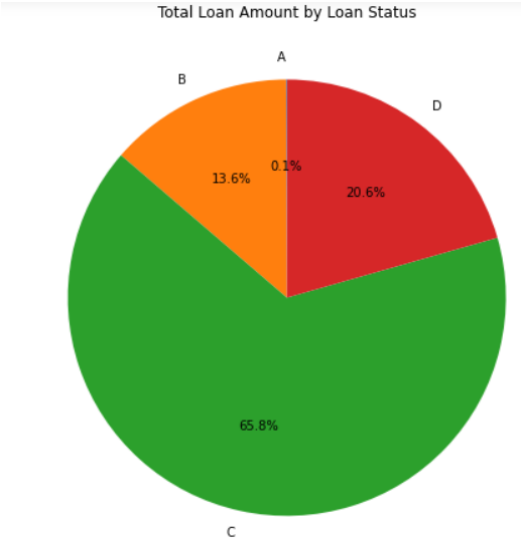
Graph 12: Loan Status Distribution



loan_status	count
C	159
B	80
D	55
A	1

Most of the clients are in status C which means 'running contract, OK so far' by 31/12/1996 with 53.9%, 159 people. Only one client which stands for 0.3% of the whole group that reached status A: contract finished, no problems, by 31/12/1996.

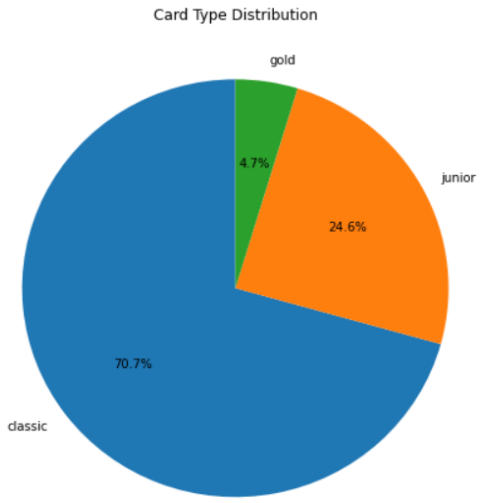
Graph 13: Total Loan Amount by Loan Status



loan_status	loan_amount
A	2.798400e+04
B	5.886420e+06
C	2.843593e+07
D	8.892911e+06

It shows that clients in status C tended to loan more money than the other status with 65.8% of the total loan amount.

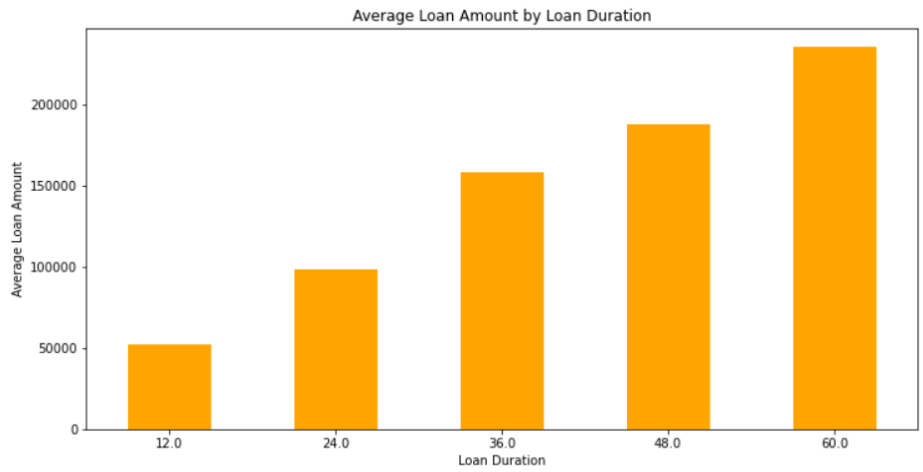
Graph 14: Card Type Distribution



card_type	count
classic	135
junior	47
gold	9

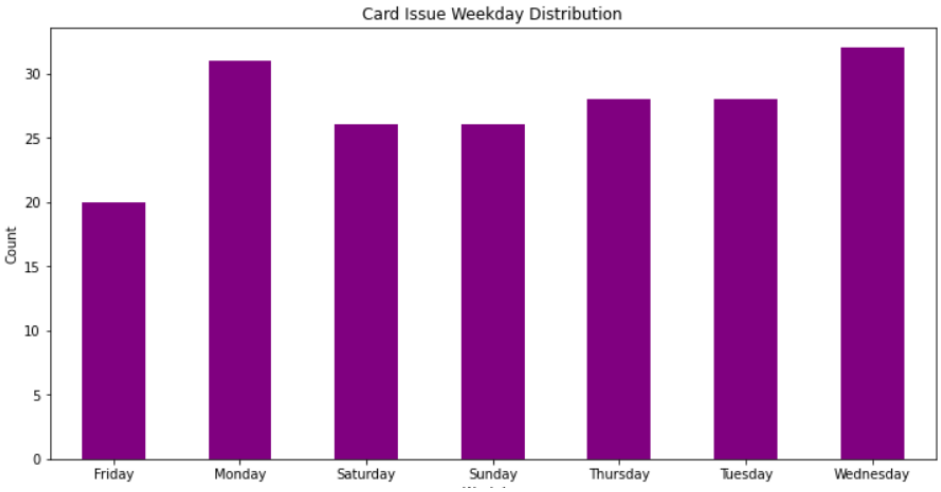
It shows that Most of the account owners are issued with classic cards with 70.7% of the group before 1997, and around 24.6% of clients holding junior cards within people who had been issue credit cards.

Graph 15: Bar Chart of Average Loan Amount by Loan Duration



The graph indicates obviously that the longer its loan duration, the higher the average amount of the loan, which can also be a hypothesis for future modeling.

Graph 16: Card Issue Weekday Distribution



It can be seen that the cards were issued quietly on average among weekdays except for Friday, there were less credit cards issued with only 20 cards before 1997.

Descriptive Statistics: Loan Amount, Loan Payments and Total Expected Paid Amount of Loan

```
count       295.000000
mean     146587.262051
std      105552.172585
min        4980.000000
25%       68616.000000
50%      115992.000000
75%      203394.000000
max      461050.576244
Name: loan_amount, dtype: float64
count     295.000000
mean     4138.064407
std      2168.100378
min       319.000000
25%      2410.500000
50%      3883.000000
75%      5892.500000
max      9689.000000
Name: loan_payments, dtype: float64
count       295.000000
mean      83118.111864
std       69034.276611
min           0.000000
25%       28256.000000
50%       62118.000000
75%      124096.000000
max      337386.000000
Name: loan_total_expected_paid_amount, dtype: float64
```

Table 9: Card Type vs. Loan Granted Possibility

```
card_type
classic    0.125926
gold       0.111111
junior     0.042553
Name: tv1_loan_granted_97, dtype: float64
```

Used df2.groupby('card_type')['tv1_loan_granted_97'].agg('mean').sort_values(ascending=False) to show the default probability by card type, and it seems that clients had classic and gold cards are more likely to be granted with their loans.

Table 10: Loan Duration vs. Issuance Probability of Credit Card

```
loan_duration
24.0    0.154930
12.0    0.150943
48.0    0.113208
60.0    0.111111
36.0    0.109091
Name: tv2 card issued 97, dtype: float64
```

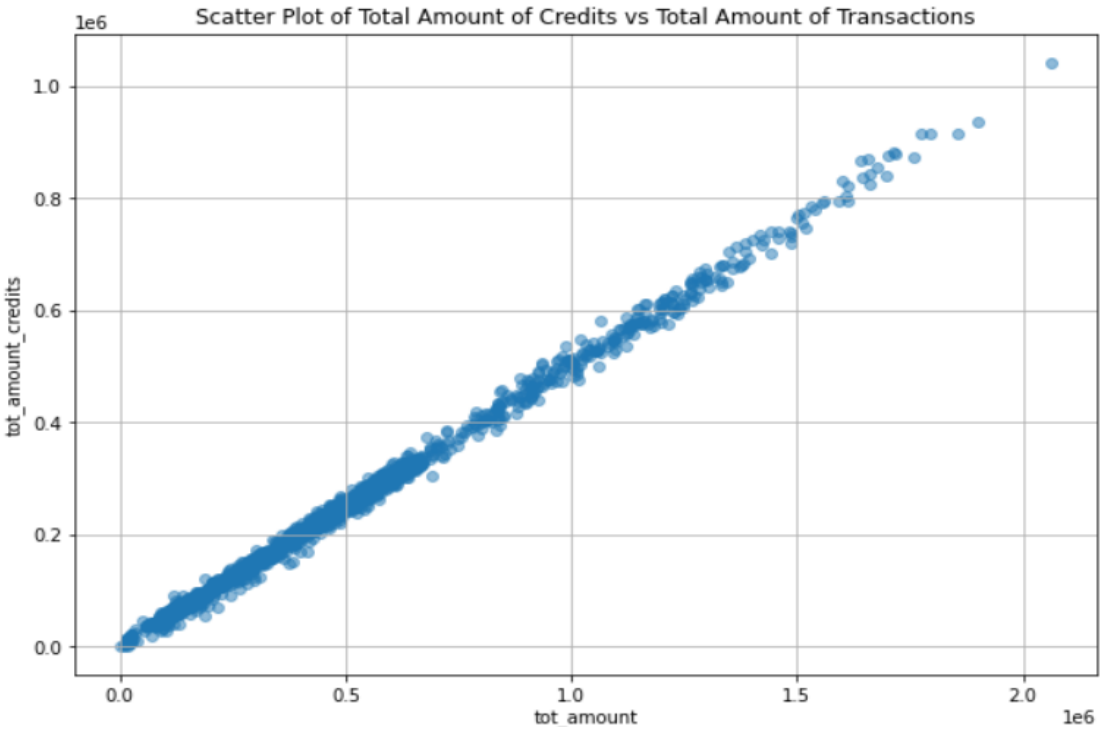
Used df2.groupby('loan_duration')['tv2_card_issued_97'].agg('mean') to show the issuance probability of loan duration, and it indicates the clients with 24 and 12 months of loan duration had a higher chance to get credit cards.

Table 11: Loan Status vs. Issuance Probability of Credit Card

```
loan_status
B    0.212500
C    0.113208
D    0.054545
A    0.000000
Name: tv2_card_issued_97, dtype: float64
```

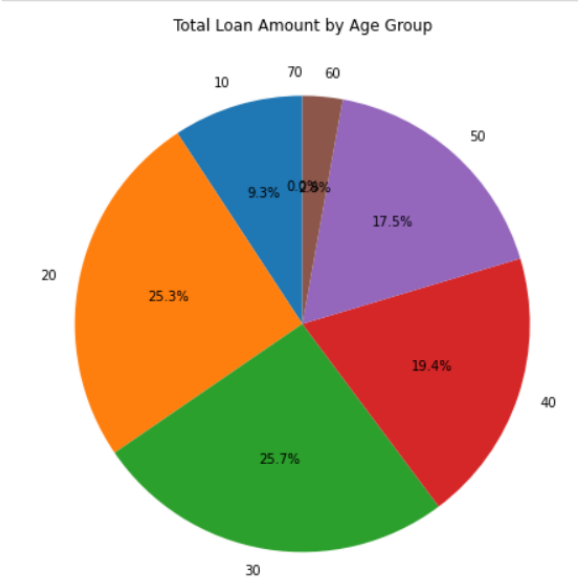
Used df2.groupby('loan_status')['tv2_card_issued_97'].agg('mean').sort_values(ascending=False) to show the issuance probability of loan status, and it indicates the clients with status B had a higher chance to get credit cards.

Graph 17: Scatter Plot of Total Amount of Credits and Total Amount of Transactions



Since there is a high correlation between tot_amount and tot_amount_credits, plot a scatter plot to check, and the graph means that there is a positive linear relationship between Total Amount of Credits and Total Amount of Transactions which is logical and can be a hypothesis for future modeling.

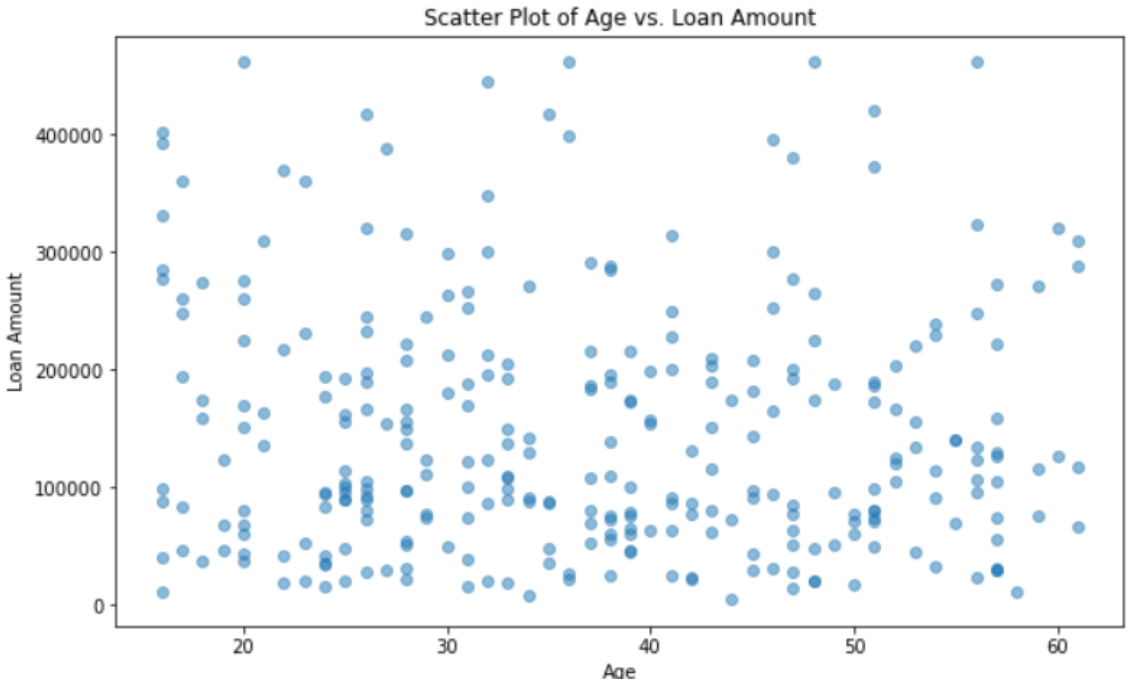
Graph 18: Pie Chart of Total Loan Amount by Age Group



age_group	loan_amount
10	4.001124e+06
20	1.094095e+07
30	1.110294e+07
40	8.389019e+06
50	7.579739e+06
60	1.229472e+06
70	0.000000e+00

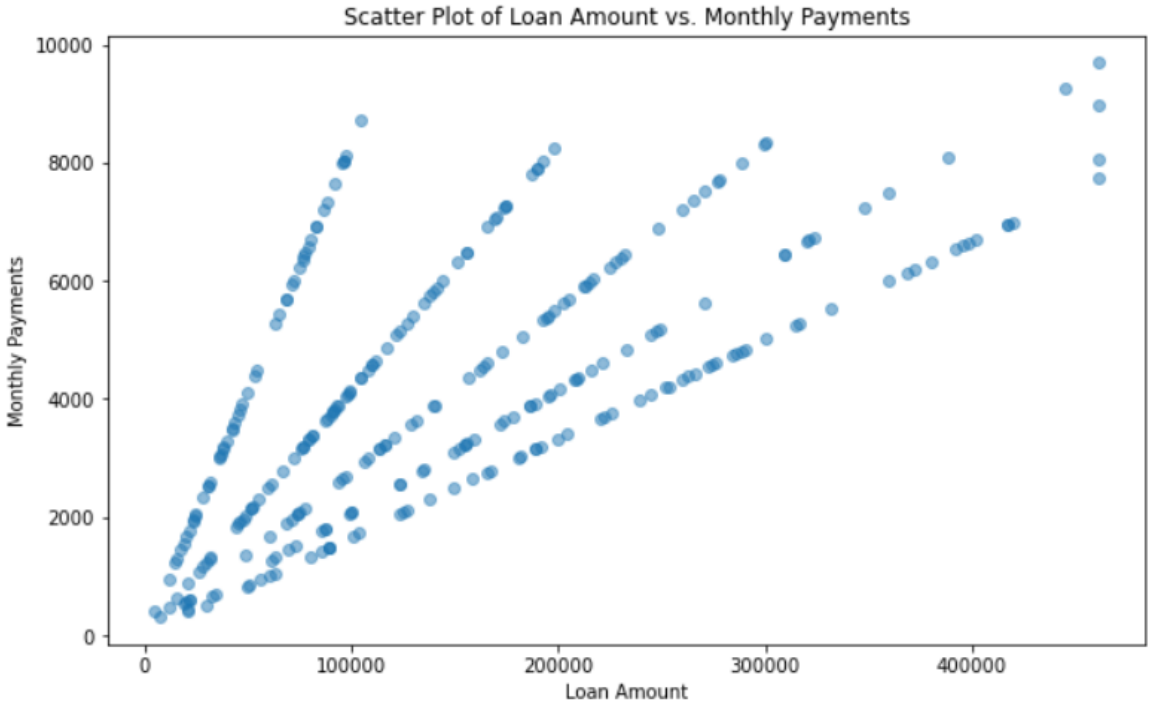
The majority of the total loan amount is concentrated in the age groups between 20 and 50 years old.

Graph 19: Scatter Plot of Age vs. Loan Amount



The range of loan amount does not seem to depend on the age of the client.

Graph 20: Scatter Plot Monthly Payments vs. Loan Amounts



Clients who have relatively lower loan amounts are not necessarily making smaller monthly payments.

Data Preprocessing

Once we described and visualized both the independent and dependent variables of our basetable, it is time to handle the different errors we could spot. To do that, the first step was to create subsets of the dataframe. As a matter of fact, we would not detect and treat errors the same way depending on the data type.

We first created a list of independent variables, excluding the two target variables from the subset. Then, we created new subsets by filtering by data type. Thus, we created three lists of variables: categories, date, and numeric.

When starting to explore these lists, we spotted that our 'id' columns were included in our list of numeric variables, which is normal since they were from type integers. Thus, we excluded them from our 'iv_num_list', and created a fourth and last list for 'id's.

After checking that all variables were included in one list (the sum of these lists was equal to the list of total independent variables), we could move to the next step.

Error Correction

Missing Values

The first step of the process was to spot the missing values from our dataframe. To do that, we used a for loop that would check the sum of missing values within each column of our subset. We also created a variable that calculates the percentage of missing values within its column. It enabled us to have a better overview and understanding of our independent variables, and then to be able to make the best decision on how to handle them.

We applied this method for our three subsets.

- For the numeric variables, the number of missing values was huge in many columns, representing more than 80% of the value count of a lot of them. However, we did not worry because these variables were advanced independent variables that we created. It is perfectly normal that not all clients perform every kind of transaction, or grant a loan. Because we know the explanation behind these missing values, we can easily decide to replace them by 0. As a matter of fact, they are numeric and not issued from an error. It simply means that the client has no activity in this field, so 0 fits perfectly for that case.

To do that, we downloaded a new python library named 'U feature engine', which is for feature engineering for Machine Learning. We found it from the slides of last session. From this library, we imported 'Add Missing Indicator' and 'Arbitrary Number imputer', since we wanted to replace null values by 0. These two transformers, runned together, ensured to keep track of the imputations.

- For categorical variables, we found three columns with missing values. Again, the percentage of missing value was higher than 80%, but it made sense because the variables were related to card or loan. As we know, only a minority of the dataset did grant a loan, or own a card. Because the null values made perfect sense, we decided to replace them by "NaN" and keep them like that.

- Finally, we had to take care of the date variables, and found three columns with the same quantity of missing values. They were related to loan and card as well, so we decided to leave them as missing.

Outliers

The next step in error correction is the detection of outliers. To do that, we calculated the lower and upper boundaries based on mean +/- 3*sd for each numerical variable in our list. We used a for loop again. As a result, we have 70 columns with outliers. Since, we should replace outliers

only in case it improves our model, we compared the result of our model with and without replacing the outliers.

To check that, we used a quick test with Decision Tree and 5-fold CV, taking into account only numerical variables since we have not transformed the categorical variables into dummies yet.

Firstly, we ran it for our basetable with no outliers replacement, and got an Accuracy (5 fold CV) of 0.82, which is satisfying.

Then we ran the same test on the other dataframe in which we replaced the outliers. To do that, we used the 'Winsorizer' package from the feature engine library.

As a result of the test, we obtained a very similar accuracy of 0.82.

We repeated the same process for the second target variable. For the model with no outliers replacement, we got an accuracy of 0.81, while we got 0.80 for the winsorized model.

Therefore, it was not a good decision to replace the outliers, and we kept our base table without handling the outliers.

Value Transformation

We checked the class frequency distribution of each of the categorical variables. Given the result, we decided not to do any remapping, and to encode them into dummies for the next part of the project later.

Similarly, we decided not to discretize the continuous variables yet. In fact, some algorithms can handle continuous variables efficiently without discretization. Decision tree is one of them, and since it is the only algorithm running our model so far, we do not need to apply any more data transformation.

Conclusion

The meticulously prepared basetable now stands as a consolidated dataset, ready for in-depth analysis and modeling. Also, the generated visualizations and statistics on the dependent and independent variables will be helpful for the clients, the operators of the bank to have a better understanding as well as analysis on their records and financial information. The harmonization of diverse data sources into a cohesive structure provides a solid foundation for extracting meaningful insights and making informed decisions.

Index

Base table columns

item	meaning
client_id	**client identifier**
client_district_id	**address of the client**
birth_year	
birth_day	
birth_month	
gender	
age	
age_group	**10 20 30 40 …**
account_id	**identification of the account**
bank_district_id	**location of the branch**
frequency	**frequency of issuance of statement**
opening_year	**opening year of the account**
lor	**length of relationship (as of 1996)**
opening_date	**opening date of the account**

disp_id	**record identifier**
tot_transactions	**total number of transactions (in 1996)**
first_trans_date	**first transaction date (in 1996)**
last_trans_date	**last transaction date (in 1996)**
withdrawals_percentage	**withdrawals percentage of total transactions 96**
avg_monthly_trans	**average number of transactions per month 96**
avg_monthly_credits	**average number of credit transactions per month 96**
avg_monthly_withdrawals	**average number of withdrawal transactions per month 96**
tot_amount	**total amount (in volume) transacted 96**
avg_monthly_amount	**average amount (in volume) transacted per month 96**
tot_amount_credits	**total amount and average amount per month transacted for each credit/withdrawal transactions 96**
avg_monthly_amount_credits	**average amount (in volume) of credit transactions per month 96**
tot_amount_withdrawals	**average amount (in volume) of credit transactions per month 96**
avg_monthly_amount_withdrawal	**average amount (in volume) of withdrawal transactions per month 96**

max_amount_credits	**maximum and minimum amount transacted for each credit/withdrawal transactions 96**
min_amount_credits	
max_amount_withdrawals	
min_amount_withdrawals	
tot_op_creditcard_withdrawal	**total number of transactions and total amount transacted for each operation category 96**
tot_amount_op_creditcard_withdrawal	
tot_op_credit_cash	
tot_amount_op_credit_cash	
tot_op_collection_otherbank	
tot_amount_op_collection_otherbank	
tot_op_withdrawal_cash	
tot_amount_op_withdrawal_cash	
tot_op_remittance_otherbank	
tot_amount_op_remittance_otherbank	
tot_op_other	
tot_amount_op_other	

per_op_creditcard_withdrawal	**operation type percentages of total transactions 96**
per_op_credit_cash	
per_op_collection_otherbank	
per_op_withdrawal_cash	
per_op_remittance_otherbank	
per_op_other	
balance_after_first_trans	**balance after first transaction 96**
balance_after_last_trans	**balance after last transaction 96**
activity_duration_days	**days elapsed from first and last transaction dates 96**
account_growth	**percentage difference between balance after last and first transactions**
tot_k_insurrance_payment	**total number of transactions and total amount transacted for each k_symbol category 96**
tot_amount_k_insurrance_payment	
tot_k_statement_payment	
tot_amount_k_statement_payment	
tot_k_interest_credited	

tot_amount_k_interest_credited	
tot_k_sanction_interest	
tot_amount_k_sanction_interest	
tot_k_household	
tot_amount_k_household	
tot_k_oldage_pension	
tot_amount_k_oldage_pension	
tot_k_loan_payment	
tot_amount_k_loan_payment	
tot_k_other	
tot_amount_k_other	
per_k_insurrance_payment	**k_symbol type percentages of total transactions 96**
per_k_statement_payment	
per_k_interest_credited	
per_k_sanction_interest	
per_k_household	
per_k_oldage_pension	

per_k_loan_payment	
per_k_other	
banks_count	**number of different banks dealt with 96**
tot_orders	**total number of permanent orders 96**
order_tot_insurrance_payment	**total number of permanent orders and average amount per order for each category 96**
order_avg_amount_insurrance_payment	
order_tot_household_payment	
order_avg_amount_household_payment	
order_tot_loan_payment	
order_avg_amount_loan_payment	
order_tot_other	
order_avg_amount_other	
loan_id	**record identifier**
loan_date	**loan granting date**
loan_amount	**amount of money**
loan_duration	**duration of the loan in months**

loan_payments	**monthly payments**
loan_expected_end_date	**expected end date of the loan**
loan_total_expected_paid_amount	**expected paid amount by 31 December 1996**
loan_num_trans	**total number of loan transactions (pre 1997)**
loan_actual_tot_paid	**actual paid amount by 31 December 1996**
loan_status	**status of paying off the loan (as of 31 December 1996)**
card_id	**record identifier**
card_type	**type of card**
card_issued	**issue date**
card_issue_weekday	**day of the week in which the card was issued**
district_name	
region	
no_of_inhabitants	
no_of_inhabitants<499	
no_of_inhabitants_500-1999	
no_of_inhabitants_2000-9999	
no_of_inhabitants>10000	

no_of_cities	
ratio_urban_inhabitants	
avg_salary	
unemploy_rate96	
no_enterp_per_1000_inhabitants	
no_crimes96	
tv1_loan_granted_97	**target variable 1: loan granted in 1997**
tv2_card_issued_97	**target variable 2: card issued in 1997**