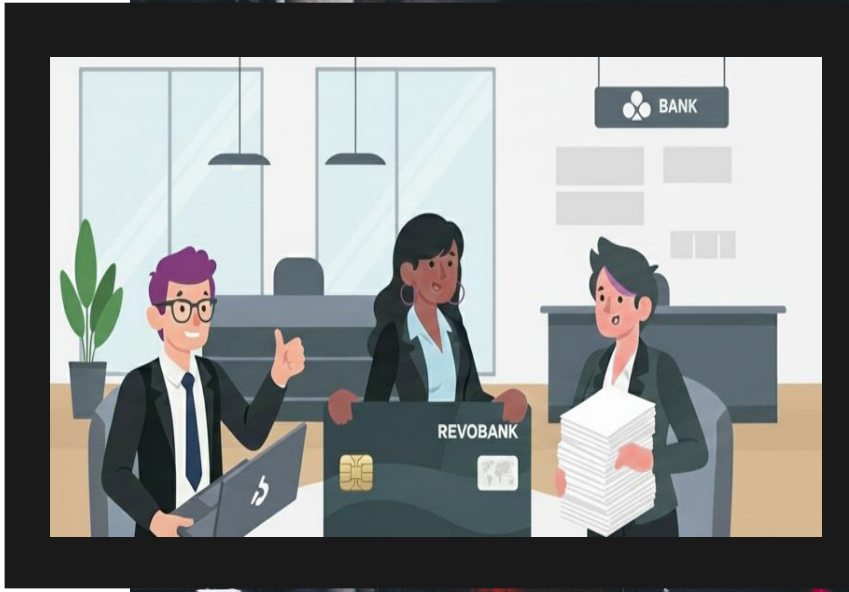
RevoBank Customer Segmentation

Fadila Utami | FSDA OCT25 | Intermediate Python | Linkedin

RevoBank

BACKGROUND

RevoBank is an Indonesia-based bank that generates revenue from credit card transactions through Merchant Discount Rate (MDR) fees. To enhance performance, the team analyzes six months of transaction data to evaluate customer behavior, transaction trends, and fraud impact on revenue.



RevoBank

OBJECTIVE

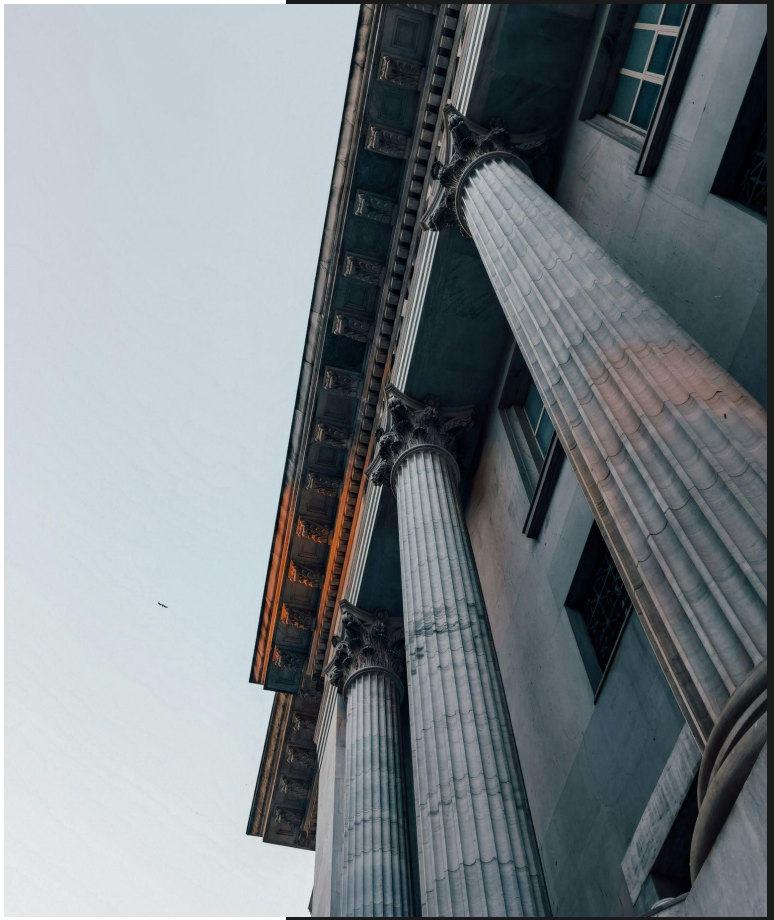
- Evaluate credit card transaction performance over the last six months
- Identify key customer segments and user personas
- Measure the impact of fraudulent transactions on net revenue

RevoBank

Dataset Overview

- **Card Dataset**: Contains credit card information including card identifiers, card brand, credit limits, transaction activity, and expiration dates, used to analyze card validity and usage patterns.

- **User Dataset**: Contains customer demographic and financial attributes such as birthdate, gender, income, debt, and retirement age, used for customer profiling and feature engineering.



Import Dataset from Spreadsheet

User and Card datasets are imported into Google Colab to begin the data cleaning and analysis process.

```
# Import dataset from spreadsheet
sheet_url ='https://docs.google.com/spreadsheets/d/1b4UKldpqgco09NIbvQ6s31YMytY6ESthLM4IJfE9QpE/edit?gid=1545334173#gid=1545334173'
sheet_url_replace = sheet_url.replace('/edit?gid=', '/export?format=csv&gid=')

# Read from the new link
df_card = pd.read_csv(sheet_url_replace)
```

```
# Import dataset from spreadsheet
sheet_url ='https://docs.google.com/spreadsheets/d/15Zu0RkZHrDPo2lgppBroZijIM9cazyfkcmUJPAVl-Ok/edit?gid=1760178867#gid=1760178867'
sheet_url_replace = sheet_url.replace('/edit?gid=', '/export?format=csv&gid=')

# Read from the new link
df_user = pd.read_csv(sheet_url_replace)
```

RevoBank

Milestone 1

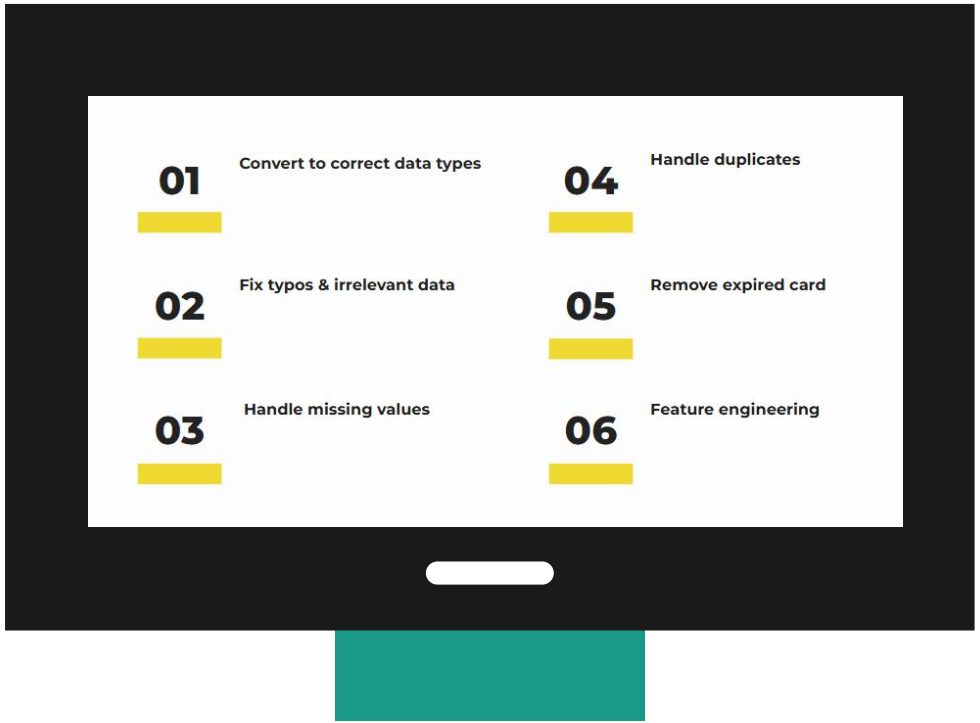
DATA CLEANING AND PREPARATION

Google Colab

RevoBank

Data Cleaning

Data cleaning ensures data accuracy and consistency by removing errors, duplicates, and inconsistencies, enabling reliable analysis and meaningful business insights



RevoBank

Check Data Type-Card

```
2 df_card_dc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5599 entries, 0 to 5598
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      5599 non-null   int64
 1   client_id               5599 non-null   int64
 2   card_brand              5599 non-null   object
 3   card_number             5599 non-null   float64
 4   expires                 5599 non-null   object
 5   cvv                     5599 non-null   int64
 6   credit_limit            5587 non-null   object
 7   acct_open_date          5599 non-null   object
 8   year_pin_last_changed   5599 non-null   int64
 9   days_since_last_trx     5599 non-null   int64
 10  count_nonfraud_trx_L6M  3707 non-null   float64
 11  amt_nonfraud_trx_L6M    3707 non-null   object
 12  count_fraud_trx_L6M     547 non-null    float64
 13  amt_fraud_trx_L6M       547 non-null    object
dtypes: float64(3), int64(5), object(6)
memory usage: 612.5+ KB
```

- id, client_id: int64 → **string**
- card_brand: object → **string**
- expires, acct_open_date: object → **datetime**
- card_number, count_nonfraud_trx_L6M, count_fraud_trx_L6M: float64 → **int**
- amt_nonfraud_trx_L6M, amt_fraud_trx_L6M: object → **float**

Convert to Correct Datatype - Card

```
<class 'pandas.core.frame.DataFrame'>
Index: 5528 entries, 0 to 5567
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      5528 non-null   string
 1   client_id               5528 non-null   string
 2   card_brand              5528 non-null   string
 3   card_number             5528 non-null   Int64
 4   expires                 5528 non-null   datetime64[ns]
 5   cvv                     5528 non-null   int64
 6   credit_limit            5528 non-null   float64
 7   acct_open_date          5528 non-null   datetime64[ns]
 8   year_pin_last_changed   5528 non-null   int64
 9   days_since_last_trx     5528 non-null   int64
 10  count_nonfraud_trx_L6M  5528 non-null   Int64
 11  amt_nonfraud_trx_L6M    5528 non-null   float64
 12  count_fraud_trx_L6M     5528 non-null   Int64
 13  amt_fraud_trx_L6M       5528 non-null   float64
dtypes: Int64(3), datetime64[ns](2), float64(3), int64(3), string(3)
memory usage: 664.0 KB
```

After the data type conversion process, the dataset has data types that are appropriately aligned with their analytical purposes.

RevoBank

Data Type Correction—Card

```
2 df_card_dc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5599 entries, 0 to 5598
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      5599 non-null   int64
 1   client_id               5599 non-null   int64
 2   card_brand              5599 non-null   object
 3   card_number             5599 non-null   float64
 4   expires                 5599 non-null   object
 5   cvv                     5599 non-null   int64
 6   credit_limit            5587 non-null   object
 7   acct_open_date          5599 non-null   object
 8   year_pin_last_changed   5599 non-null   int64
 9   days_since_last_trx     5599 non-null   int64
 10  count_nonfraud_trx_L6M  3707 non-null   float64
 11  amt_nonfraud_trx_L6M    3707 non-null   object
 12  count_fraud_trx_L6M     547 non-null    float64
 13  amt_fraud_trx_L6M       547 non-null    object
dtypes: float64(3), int64(5), object(6)
memory usage: 612.5+ KB
```

- id, client_id: int64 → **string**
- card_brand: object → **string**
- expires, acct_open_date: object → **datetime**
- card_number, count_nonfraud_trx_L6M, count_fraud_trx_L6M: float64 → **int**
- amt_nonfraud_trx_L6M, amt_fraud_trx_L6M: object → **float**

RevoBank

RevoBank

Handle or exclude Typo - User

```
# Menampilkan daftar kolom yang bertipe string dalam
df_card_dc.select_dtypes(include='string').columns
# Mencari TYPO dan inkonsistensi kategori di SEMUA kolom teks
for col in df_card_dc.select_dtypes(include=['object',
'string']).columns:
    print(f"\n=== {col} ===")
    print(df_card_dc[col].value_counts())
```

Issues Identified

- **card_brand** shows inconsistent capitalization (e.g., *"JCB"* vs *"Jcb"*), which represents the same category.
- **Date-related columns** (expires, acct_open_date) appear in string format with multiple representations (e.g., 02/2026, 11/2026), indicating formatting inconsistency.
- **Monetary columns** (credit_limit, amt_nonfraud_trx_L6M, amt_fraud_trx_L6M) contain currency symbols and thousand separators (e.g., "Rp", dots), making them non-numeric.
- **Zero-value** entries are present in credit_limit (e.g., Rp0), which are not meaningful for active card analysis.

```
# Standardize card_brand capitalization
df_card_dc['card_brand'] = df_card_dc['card_brand'].str.upper()
```

BEFORE

```
=== card_brand ===
card_brand
Mastercard    2826
Visa          2162
Amex           402
JCB            206
Jcb              3
```

AFTER

card_brand	
MASTERCARD	2826
VISA	2162
AMEX	402
JCB	209

```
1 #check missing value in the card_brand
2 df_card_dc['card_brand'].isna().sum()
3
```

```
np.int64(0)
```

After standardizing the card brand names, all values are now consistent and no longer contain capitalization-related typos.

Duplicate Check and Removal

```
1 # Check Duplicates
2 df_card_dc.duplicated().sum()
```

```
np.int64(31)
```

```
# Check duplicate records based on card_number and display all rows involved,
# sorted by card_number for easier visual inspection

df_card_dc[df_card_dc.duplicated(subset=['id'], keep=False)].sort_values('id')
```

	id	client_id	card_brand	card_number	expires
1040	1161	1891	VISA	4268017872699469	08/2031
5597	1161	1891	VISA	4268017872699469	08/2031
5584	1781	1372	VISA	4806267788873524	08/2026
1604	1781	1372	VISA	4806267788873524	08/2026
5580	1860	975	VISA	4039295566770817	05/2027
...	...	...	...	...	...
4995	5515	1422	VISA	4103248264872952	07/2031
5538	6115	921	VISA	4816027381867141	11/2029
5598	6115	921	VISA	4816027381867141	11/2029
667	749	54	VISA	4557061225558443	05/2029
5582	749	54	VISA	4557061225558443	05/2029

A total of **31 duplicate records** were identified in the card dataset and successfully removed, ensuring the dataset is free from duplicates and ready for analysis.

```
#Delete Duplicates in column card_number
df_card_dc = df_card_dc.drop_duplicates(subset=['id'])
```

Reason to Remove Duplicates

Duplicate card records were removed to prevent double counting and ensure that each card is represented only once in the analysis.

```
1 # Check Duplicates
2 df_user_dc.duplicated().sum()
```

```
np.int64(0)
```

No duplicate records were identified in the user dataset, indicating strong data integrity and eliminating the need for further deduplication prior to analysis

RevoBank

MISSING VALUE

```
1 # Check MISSING value Percolumn
2 df_card_dc.isnull().sum()
```

id	0
client_id	0
card_brand	0
card_number	0
expires	0
cvv	0
credit_limit	12
acct_open_date	0
year_pin_last_changed	0
days_since_last_trx	0
count_nonfraud_trx_L6M	0
amt_nonfraud_trx_L6M	1892
count_fraud_trx_L6M	0
amt_fraud_trx_L6M	5052

Three columns were identified with missing values and required data cleaning: credit_limit, amt_nonfraud_trx_L6M, and amt_fraud_trx_L6M.

```
# Deleting rows  whose credit_limit value is empty (NaN)
df_card_dc = df_card_dc.dropna(subset=['credit_limit'])

# Fill missing transaction values with 0
transaction_cols = ['amt_nonfraud_trx_L6M', 'amt_fraud_trx_L6M']

df_card_dc[transaction_cols] = df_card_dc[transaction_cols].fillna(0)
```

1. Rows with missing values in the credit_limit column were removed, as credit limit is a critical financial attribute and cannot be logically imputed.
2. Missing values in transaction-related columns were filled with zero, representing the absence of transaction activity rather than data quality issues.

```
1 #Check Missing Value
2 df_card_dc[trx_cols].isna().sum()
3
```

	0
count_nonfraud_trx_L6M	0
amt_nonfraud_trx_L6M	0
count_fraud_trx_L6M	0
amt_fraud_trx_L6M	0

No missing values remain in the card dataset

RevoBank

RevoBank

Remove all expired and 0-valued cards

```python
# filter invalid card ( expired & credit limit = 0)



# 2. Tentukan cutoff date
cutoff_date = pd.Timestamp('2025-05-31')

# 3. Hapus kartu yang sudah expired
df_card_dc['expires'] = pd.to_datetime(df_card_dc['expires'])


# 4. Hapus kartu dengan credit_limit = 0
df_card_dc = df_card_dc[df_card_dc['credit_limit'] > 0]

# 5. Validasi hasil
df_card_dc.shape

```

```
(5531, 14)
```

Insight

1. Cards that had expired before the analysis cutoff date (May, 31 2025) were removed from the dataset.
2. Cards with zero credit limits were excluded, as they are not usable for transactions.
3. After filtering, the dataset contains 5,531 active and usable card records, ensuring data relevance for further analysis

RevoBank

Check Data Type-User

```
1 # Check table info
2 df_user_dc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 2000 non-null   int64 
 1   retirement_age     2000 non-null   int64 
 2   birthdate          2000 non-null   object
 3   gender             2000 non-null   object
 4   per_capita_income  2000 non-null   object
 5   yearly_income      2000 non-null   object
 6   total_debt         2000 non-null   object
 7   credit_score       2000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 125.1+ KB
```

1. id and gender are converted to string
2. birthdate is converted to a datetime
3. per_capita_income, yearly_income, and total_debt are converted to float

RevoBank

Convert Data Type- User

```python
#Convert id and gender to string
df_user_dc['id'] = df_user_dc['id'].astype('string')
df_user_dc['gender'] = df_user_dc['gender'].astype('string')
```

```python
#Convert birthdate to datetime
df_user_dc['birthdate'] = pd.to_datetime(df_user_dc['birthdate'], errors='coerce')
```

```python
#Convert kolom numerik uang to float
money_cols = ['per_capita_income', 'yearly_income', 'total_debt']

for col in money_cols:
    df_user_dc[col] = (
        df_user_dc[col]
        .astype(str)
        .str.replace('Rp', '', regex=False)   # hapus Rp
        .str.replace('.', '', regex=False)    # hapus titik ribuan
        .str.replace(',', '', regex=False)    # jaga-jaga kalau ada koma
        .str.strip()
        .replace('', np.nan)
        .astype(float)
    )
```

This process ensures the datasets are analysis-ready, prevents incorrect calculations, and improves the reliability of aggregations and time-based insights.

RevoBank

Data Type Correction - User

```
1 df_user_dc.info()
2
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 2000 non-null   string
 1   retirement_age     2000 non-null   int64
 2   birthdate          2000 non-null   datetime64[ns]
 3   gender             2000 non-null   string
 4   per_capita_income  2000 non-null   float64
 5   yearly_income      2000 non-null   float64
 6   total_debt         2000 non-null   float64
 7   credit_score       2000 non-null   int64
dtypes: datetime64[ns](1), float64(3), int64(2), string(2)
memory usage: 125.1 KB
```

After the data type conversion process, the dataset has data types that are appropriately aligned with their analytical purposes.

RevoBank

Add Age, Retirement Age, and (DTI)

```python
# Define cutoff date
cutoff_date = pd.Timestamp('2025-05-31')

# Create age column
df_user_dc['age'] = (
    (cutoff_date - df_user_dc['birthdate']).dt.days // 365)

```

```python
# change age to integer
df_user_dc['age'] = df_user_dc['age'].astype('Int64')

```

```python
# Create Retired Flag
# 1= retired and 0= not yet retired

df_user_dc['retired_flag'] = np.where(df_user_dc['age'] >= df_user_dc['retirement_age'], 1, 0)

```

```python
#DTI = total_debt / yearly_income

df_user_dc['DTI'] = df_user_dc['total_debt'] / df_user_dc['yearly_income']
```

```python
df_user_dc[['age', 'retired_flag', 'DTI']].head()

```

Features Added

- **Age**: Calculated using each user's birthdate and a fixed analysis cutoff date to ensure consistent age measurement.
- **Retired Flag**: A binary indicator identifying whether a user has reached retirement age.
- **Debt-to-Income (DTI)**: A ratio comparing total debt to yearly income to represent financial burden.

RevoBank

Add Age, Retirement Age, and (DTI) Columns

	age	retired_flag	DTI
0	52	0	0.407184
1	52	0	0.471786
2	80	1	0.001117
3	62	0	0.154201
4	42	0	0.319271

The table shows the newly created features for each user.

- **Age** represents the user's age as of the analysis cutoff date.
- **Retired flag** indicates retirement status, where 0 means the user has not yet retired and 1 means the user is retired.
- **DTI (Debt-to-Income)** shows the proportion of total debt compared to yearly income.

From the sample output, users in the early 50s are not yet retired and have moderate DTI values, while older users may have higher or lower DTI depending on their financial condition. A DTI value close to or above 1 indicates that total debt is similar to or exceeds annual income, suggesting higher financial risk.

retired_flag	proportion
0	85.8
1	14.2

Insight

Most users are in the productive age group (85.8% non-retired), with only 14.2% retired. DTI levels are generally reasonable, indicating manageable debt and suitability for further risk analysis and segmentation.

SUMMARY

DATA CLEANING	ORIGINAL DATA	AFTER CLEANING	REASON
Convert to the correct data type	• id, client_id as int64 • card_brand as object • card_number, count_nonfraud_trx_L6M, and count_fraud_trx_L6M as float64 • expires and acct_open_date as object • credit_limit, amt_nonfraud_trx_L6M, and amt_fraud_trx_L6M as object	• id, client_id, card_brand convert to string • card_number, count_nonfraud_trx_L6M, and count_fraud_trx_L6M convert to int64 • expires and acct_open_date convert to datetime • credit_limit, amt_nonfraud_trx_L6M, and amt_fraud_trx_L6M convert to float	To ensure data consistency and enable accurate numerical and time-based analysis.
	• id as int64 • gender as object • birthdate as object • per_capita_income, yearly_income, and total_debt as object	• id and gender convert to string • birthdate convert to datetime • per_capita_income, yearly_income, and total_debt convert to float	To ensure data consistency and enable accurate demographic and financial analysis

SUMMARY

DATA CLEANING	ORIGINAL DATA	AFTER CLEANING	REASON
Remove Irrelevant data	Inconsistent categorical values caused by different capitalization and formatting (e.g., card_brand values such as “JCB” and “Jcb”)	card brands converted to a consistent format	To prevent category duplication and ensure accurate aggregation and analysis
Treat Missing Values	credit_limit, amt_nonfraud_trx_L6M, and amt_fraud_trx_L6M	missing critical credit_limit removed, and missing transaction amounts filled with 0	To maintain data completeness while avoiding invalid or misleading financial records
Treat duplicates	card_number	Duplicate records removed, keeping one unique record per card number	To prevent double counting and ensure each card is uniquely represented
Remove all expired cards	The dataset contained cards that had expired before the analysis cutoff date	Only cards with expiration dates after May 31, 2025 were retained	To ensure the analysis includes only valid and active cards

RevoBank

Milestone 2

EXPLORATORY DATA ANALAYST

Google Colab

Milestone 2

Evaluates exploratory data analysis skills, including net profit and fraud rate analysis with visualizations, transaction behavior by card brand, DTI comparison (retired vs. non-retired), DTI risk by credit score, and the relationship between age and credit limit—aimed at generating actionable business insights.

Net Profit Performance

```python
# Total fraud & non-fraud amount
total_fraud = df_card['amt_fraud_trx_L6M'].sum()
total_nonfraud = df_card['amt_nonfraud_trx_L6M'].sum()

# Fraud rate
fraud_rate = total_fraud / (total_fraud + total_nonfraud)

# Print result
print(f"Fraud Rate (Last 6 Months): {fraud_rate:.2%}")

```

```
Fraud Rate (Last 6 Months): 0.22%
```

The fraud rate over the last six months is **0.22%**, indicating that fraud losses account for a very small proportion of total transaction volume and remain within a safe threshold for RevoBank.

```
Total MDR Income: 6,840,847,479.00
Total Fraud Loss: 1,013,378,300.00
Total Net Profit: 5,827,469,179.00
```

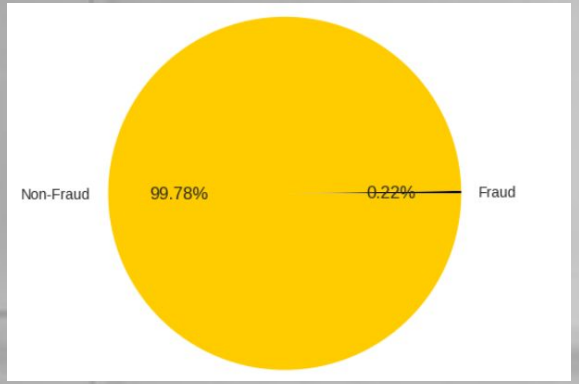
Over the past six months, RevoBank generated IDR **6.84B** in MDR fee income while incurring IDR **1.01B** in fraud losses, resulting in a total net profit of IDR **5.83B**, indicating strong profitability despite existing fraud risk.

Fraud Rate Analysis

```python
# Define the MDR rate as 1.5% based on RevoBank's profit margin
mdr_rate = 0.015

#Total Fraud Amount
total_fraud = df_card['amt_fraud_trx_L6M'].sum()

#Total Sales Amount (Non-Fraud)
total_sales_amt = df_card['amt_nonfraud_trx_L6M'].sum()


# Net Profit = (Total Sales  * MDR) - Total Fraud Loss
mdr_income = total_sales_amt * mdr_rate
total_net_profit = mdr_income - total_fraud

print(f"Total MDR Income: {mdr_income:,.2f}")
print(f"Total Fraud Loss: {total_fraud:,.2f}")
print(f"Total Net Profit: {total_net_profit:,.2f}")
```

```
Fraud Rate (Last 6 Months): 0.22%
```

Fraud vs Non-Fraud Transaction Amount



The fraud rate over the last six months is **0.22%**, indicating that fraud losses account for a very small proportion of total transaction volume and remain within a safe threshold for RevoBank.

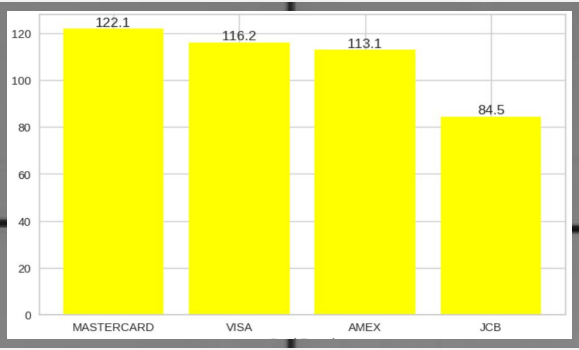
Transaction Behavior

d	avg_trx_count	avg_trx_amount
	113.06852791878173	99060459.39086294
	84.53170731707317	75211617.07317074
	122.0519018841095	80786854.74582297
	116.17249527410208	82786030.24574669

Mastercard exhibits the highest average transaction frequency (≈122 transactions over the last six months), followed by Visa (≈116) and Amex (≈113), while JCB shows the lowest usage level (≈85). Conversely, Amex records the highest average transaction amount (≈IDR 99 million), indicating a premium spending pattern, whereas Mastercard and Visa are characterized by higher transaction frequency with moderate average transaction values.

Transaction Behavior

Average Transaction Count by Card Brand



Insight:
Mastercard has the highest transaction frequency, followed by Visa and Amex, while JCB shows the lowest activity, indicating more active usage among Mastercard and Visa users.

Average Transaction Amount by Card Brand

Insight:
Amex shows the highest average transaction value, indicating premium spending, while Visa and Mastercard record moderate levels and JCB has the lowest average transaction amount.

Merged

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1938 entries, 0 to 1937
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      1938 non-null   object
 1   retirement_age          1938 non-null   int64
 2   birthdate               1938 non-null   datetime64[ns]
 3   gender                  1938 non-null   string
 4   per_capita_income       1938 non-null   float64
 5   yearly_income           1938 non-null   float64
 6   total_debt              1938 non-null   float64
 7   credit_score            1938 non-null   int64
 8   age                     1938 non-null   Int64
 9   retired_flag            1938 non-null   object
 10  DTI                     1938 non-null   float64
 11  client_id               1938 non-null   object
 12  days_since_last_trx     1938 non-null   int64
 13  count_nonfraud_trx_L6M  1938 non-null   Int64
 14  amt_nonfraud_trx_L6M    1938 non-null   float64
 15  count_fraud_trx_L6M     1938 non-null   Int64
 16  amt_fraud_trx_L6M       1938 non-null   float64
 17  credit_limit            1938 non-null   float64
```

User and card datasets were merged using an inner join at the client level, ensuring each user is represented by a single consolidated record.

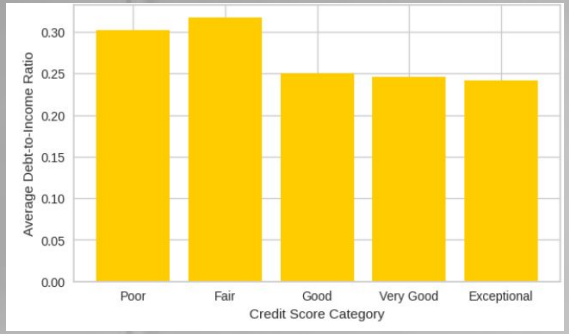
Transactional data were aggregated per client, with monetary and transaction count variables summed, while recency-related fields retained the most recent value.

DTI Retired vs Non-Retired

This analysis is conducted to examine how financial risk, measured by the Debt-to-Income (DTI) ratio, differs across credit score categories. Credit scores are first segmented into five standard groups from Poor to Exceptional to classify users based on their creditworthiness. The average DTI is then calculated for each category, enabling a clear comparison of financial risk levels across different credit score segments.

index	credit_score_category	DTI
0	Poor	0.3510773905108333
1	Fair	0.3488730950893156
2	Good	0.27797242156404645
3	Very Good	0.27285754459688216
4	Exceptional	0.26574128959148674

Average DTI by Credit Score Category

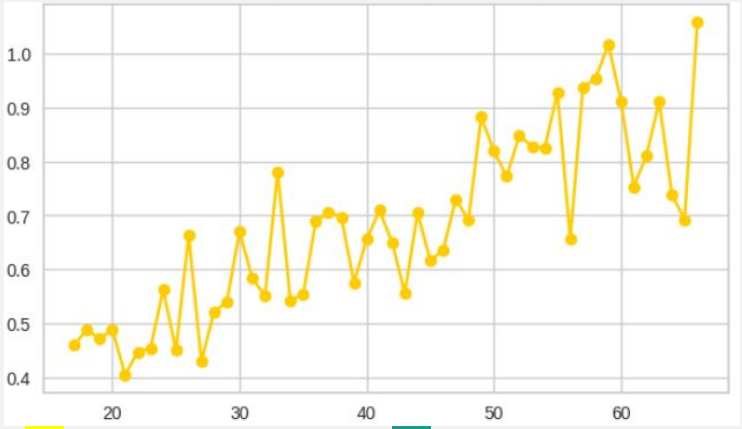


The analysis shows a clear inverse relationship between credit score category and average DTI, where higher credit scores are associated with lower debt-to-income ratios, indicating a lower financial risk profile.

RevoBank should prioritize credit expansion and promotional strategies for customers in the Good to Exceptional credit score categories, as they exhibit lower average DTI and a more manageable risk profile, while applying tighter credit controls and enhanced monitoring for Poor and Fair segments due to their higher DTI exposure.

Relationship Between User Age and Credit Limit

Average Credit Limit by Age



X
Age

Y
Average Credit limit

The analysis shows a clear positive relationship between age and average credit limit up to age 66, where limits gradually increase from approximately IDR 45–60 million in the early 20s to around IDR 80–100 million among users in their 50s–60s, indicating higher credit capacity within productive age segments.

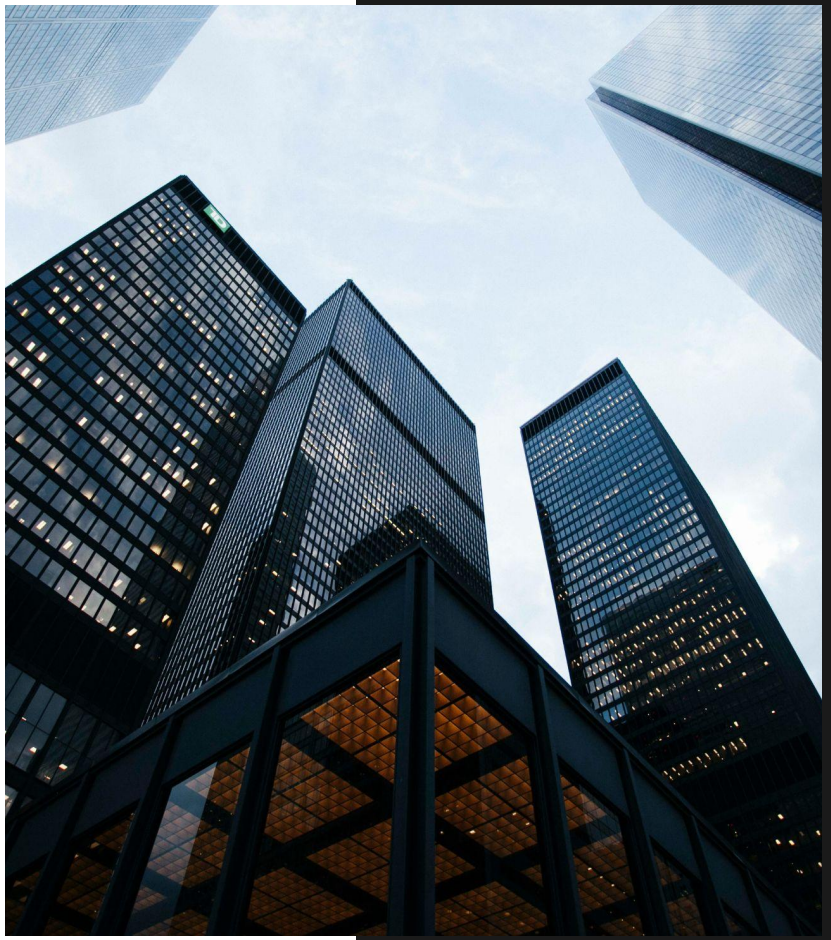
RevoBank

Milestone 3

Google Colab

Milestone 3

Identify customer segments with high potential to increase spending and contribute more to RevoBank’s profitability.



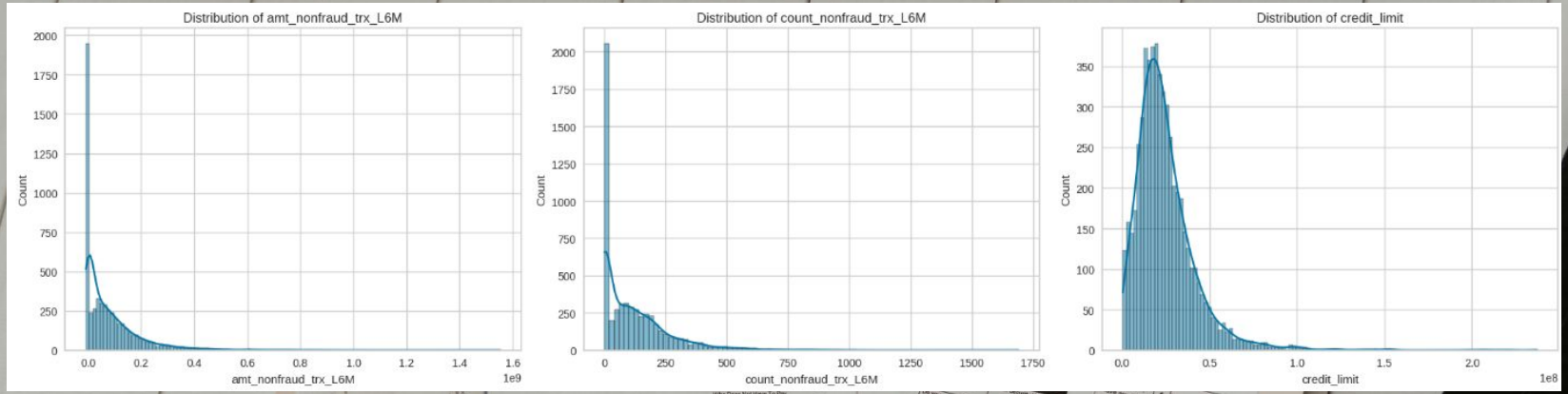
Clustering Method (K-Means)

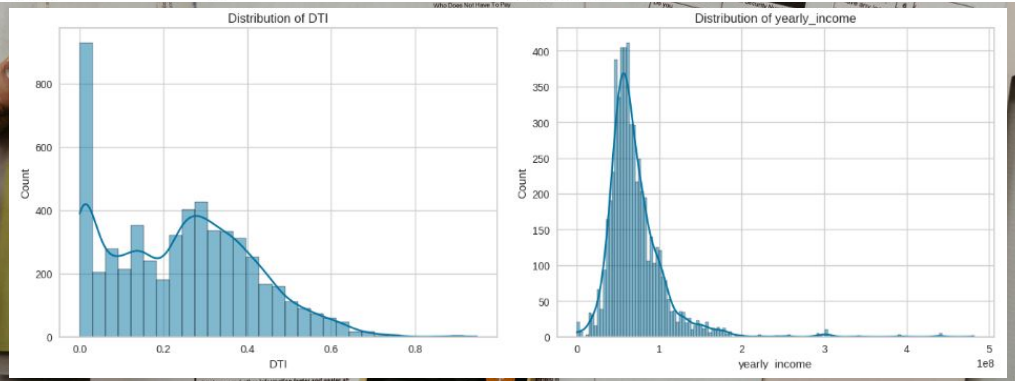
K-Means clustering is selected because it can group customers based on multiple numerical variables simultaneously, capturing not only transaction behavior but also financial capacity and risk. Compared to RFM, which focuses solely on recency, frequency, and monetary value, K-Means is more flexible and better aligned with RevoBank's objective of balancing profitability and risk.

Clustering Basis

- Transaction Value (amt_nonfraud_trx_L6M) – revenue contribution
- Transaction Frequency (count_nonfraud_trx_L6M) – activity & engagement
- Yearly_Income – underlying financial capacity and spending potential
- Credit Limit – spending capacity & growth potential
- DTI – financial risk indicator

Other numeric variables were excluded to avoid redundancy and preserve actionable, revenue-focused clusters.





The distribution analysis shows that most financial and transactional variables are heavily right-skewed with long tails, particularly transaction amount, transaction frequency, credit limit, and yearly income, indicating the presence of extreme outliers and highly heterogeneous customer behavior. In contrast, DTI exhibits a more concentrated distribution with relatively lower dispersion, suggesting more stable risk characteristics across customers. These patterns justify the use of robust scaling to reduce the influence of extreme values while preserving meaningful behavioral differences for clustering.

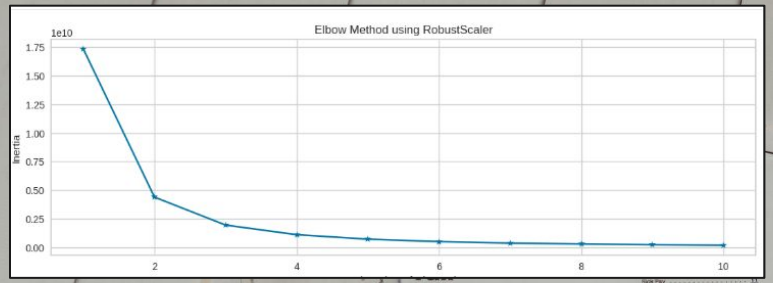
Scale the Data (RobustScaler)

RobustScaler is chosen because it is robust to skewed distributions and outliers, preserving valid extreme values while ensuring balanced feature scaling for distance-based clustering.

index	yearly_income	amt_nonfraud_trx_L6M	count_nonfraud_trx_L6M	credit_limit	DTI
count	5528.0	5528.0	5528.0	5528.0	5528.0
mean	0.25202373169252923	0.3320442258497759	0.23268439392346935	0.20315853017342642	-0.030095739744477108
std	1.1545991350842555	1.0099675609882761	0.8430706259332446	1.0179191370716594	0.6219082175861993
min	-2.0037335070340863	-0.4439156738706383	-0.4245810055865922	-1.1726473334355272	-0.9071063503024596
25%	-0.3924710256023259	-0.37170001683313836	-0.4245810055865922	-0.41979403280424754	-0.5838923716570128
50%	0.0	0.0	0.0	0.0	0.0
75%	0.6075289743976741	0.6282999831668616	0.5754189944134078	0.5802059671957525	0.4161076283429873
max	13.30231080254514	12.881851209991162	9.022346368715084	12.03645535751613	2.560426415251727

- Median values at 0 confirm successful centering using median-based scaling.
- Large gaps between the 75th percentile and maximum values highlight persistent extreme behaviors that are intentionally down-weighted rather than removed.
- DTI shows a narrower spread compared to monetary variables, indicating relatively more stable risk behavior across customers.

Cluster Selection and Validation



Elbow Method suggests k = 3 as a reasonable candidate based on the visible bend in the inertia curve

Silhouette Scores:

k = 2 → 0.4277(highest)

k = 3 → 0.3999 (Selected)

k = 4 → 0.3193

k = 5 → 0.2265

k = 3 was selected as the optimal number of clusters, as it provides the best balance between cluster separation and interpretability while achieving the highest silhouette score.

K= 3 select for the final model due to stronger cluster separation, higher stability, and better interpretability.

Business Opportunities by Customer Segment

Cluster 0 – Moderate Usage Cluster

- Encourage higher engagement through targeted promotions and usage-based incentives.
- Maintain current credit exposure while monitoring behavioral changes.
- Focus on retention programs to gradually increase transaction frequency.

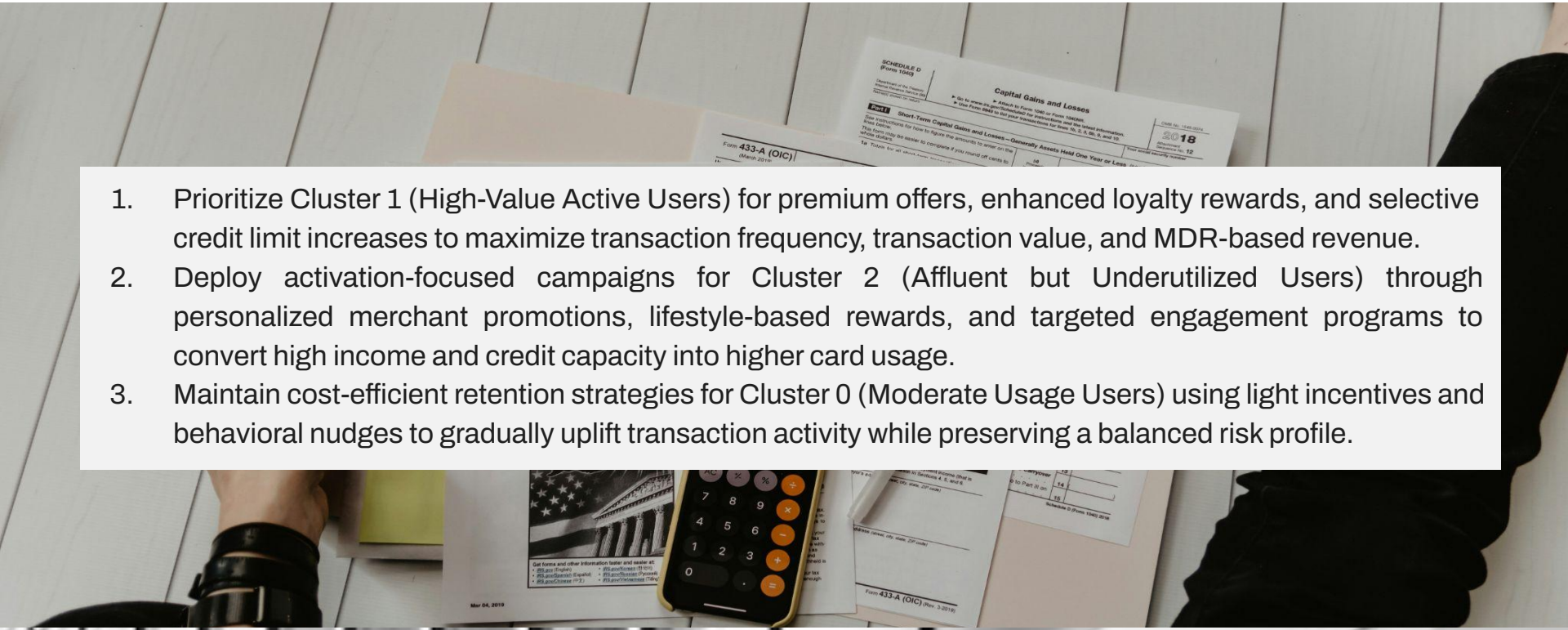
Cluster 1 – High-Value Active Cluster

- Prioritize loyalty and rewards programs to sustain high transaction intensity.
- Offer selective credit limit increases to unlock additional spending potential. Cross-sell premium products and value-added services to maximize lifetime value.

Cluster 2 – Affluent but Underutilized Cluster

- Activate spending through personalized offers and tailored merchant campaigns.
- Reassess credit utilization strategy to improve card usage efficiency. Introduce lifestyle-based rewards and premium features to convert capacity into revenue.

Recommendation



1. Prioritize Cluster 1 (High-Value Active Users) for premium offers, enhanced loyalty rewards, and selective credit limit increases to maximize transaction frequency, transaction value, and MDR-based revenue.
2. Deploy activation-focused campaigns for Cluster 2 (Affluent but Underutilized Users) through personalized merchant promotions, lifestyle-based rewards, and targeted engagement programs to convert high income and credit capacity into higher card usage.
3. Maintain cost-efficient retention strategies for Cluster 0 (Moderate Usage Users) using light incentives and behavioral nudges to gradually uplift transaction activity while preserving a balanced risk profile.

Thanks!

Appendix

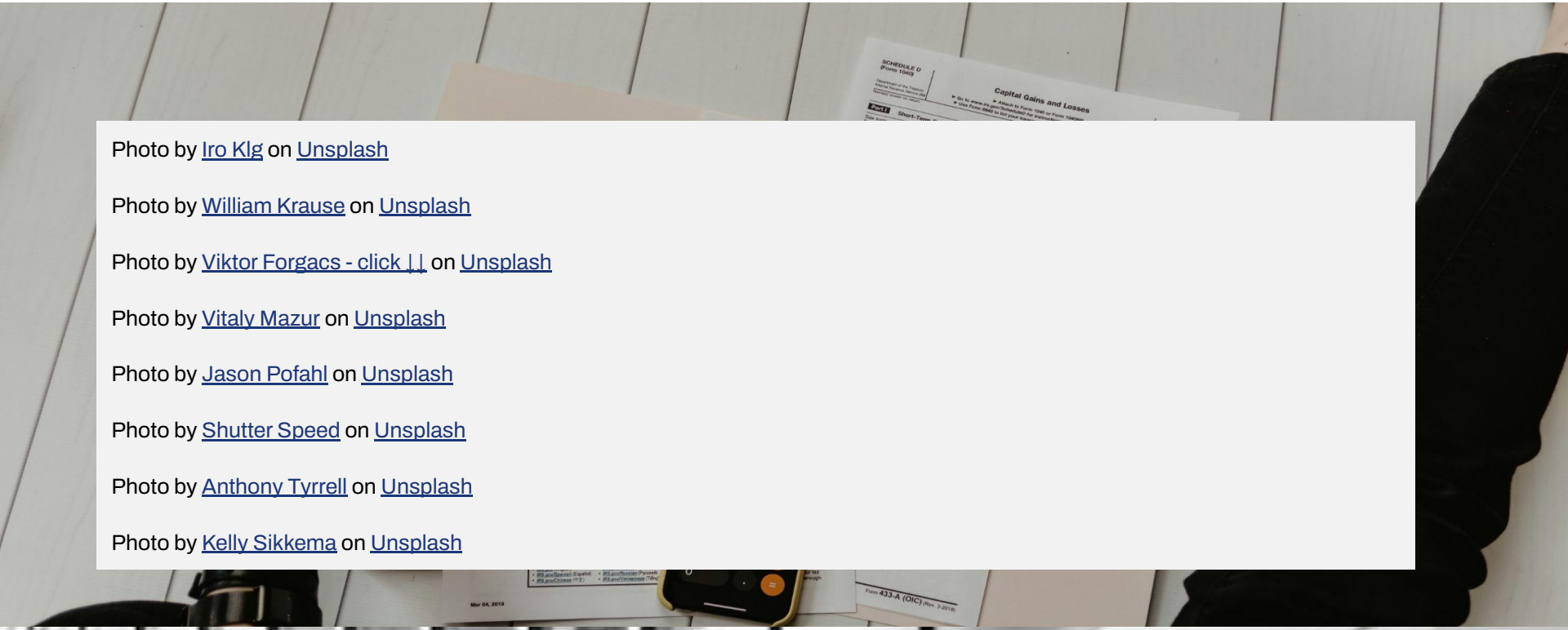


Photo by Iro Klg on Unsplash

Photo by William Krause on Unsplash

Photo by Viktor Forgacs - click ↓↓ on Unsplash

Photo by Vitaly Mazur on Unsplash

Photo by Jason Pofahl on Unsplash

Photo by Shutter Speed on Unsplash

Photo by Anthony Tyrrell on Unsplash

Photo by Kelly Sikkema on Unsplash