Exercise 1 Analysis of client portfolio

In [1]:
```
## Importing the necessary libraries for the analysis
import numpy as np # linear algebra
import pandas as pd # for data manipulation & data frame
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns

## Loading the data set
df = pd.read_csv("Portfolio_for_test6.csv")
df.fillna(0,inplace=True)
df.head()
```

Out[1]:

	CLIENTID	CLIENTNAME	GROUPNAME	MERCHANT NAME	CLIENTNAME	MCC	ONLINEPOS	SCHEME	TRX COUNT	TRX_AMOUNT	MSC	MSC_LTM	NET MSC	NET MSC_LTM	ATV	ID	REFUND_C
0	Group_JD5826	Client_JD12962	Group_JD5826	Merchant5158	Client1435	5977	POS	Visa	1032	36042.95	164.4513	174751.75720	10.829076	1264.64461	18.496762	1981952	
1	Group_JC161	Client_JD1470	Group_JC161	Merchant5181	Client1383	5299	Online	Other_CC	0	0.00	0.0000	1.379428	0.000000	0.483200	27.500000	1967069	
2	Group_JD2968	Client_JD9830	Group_JD2968	Merchant5170	Client5125	5977	POS	Visa	0	0.00	0.0000	849.784410	0.000000	67.831484	29.947444	1980027	
3	Group_JD4209	Client_JD10750	Group_JD4209	Merchant5996	Client9631	5977	POS	MC	0	0.00	0.0000	14015.680340	0.000000	2181.874021	23.287567	1980027	
4	Group_JD4064	Client_JD11710	Group_JD4064	Merchant7915	Client13032	5921	Online	Visa	1	33.21	0.2760	5.292904	0.150512	3.206020	26.799000	1991204	

5 rows × 21 columns

In [2]:
```
# Check the shape of the dataset
df.shape # 1990 rows and 26 columns
```

Out[2]:
```
(1990, 26)
```

In [3]:
```
## Data Understanding and Cleaning
## Check for null values
df.isnull().sum() # 4 columns are empty so I will drop them
```

Out[3]:
```
GROUPID            0
CLIENTID           0
GROUPNAME          0
MERCHANT NAME      0
CLIENTNAME         0
MCC                0
ONLINEPOS          0
SCHEME             0
TRX COUNT          0
TRX_AMOUNT         0
TRX_COUNT_LTM      0
TRX_AMOUNT_LTM     0
MSC                0
MSC_LTM            0
NET MSC            0
NET MSC_LTM        0
ATV                0
ID                 0
REFUND_COUNT       0
REFUND_AMOUNT      0
REFUND_COUNT_LTM   0
REFUND_AMOUNT_LTM  0
dtype: int64
```

In [4]:
```
# Delete the 4 columns because they all have missing data
df.drop(['CB_COUNT', 'CB_AMOUNT', 'FRAUD_COUNT', 'FRAUD_AMOUNT'], axis=1, inplace=True)
```

In [5]:
```
df.isnull().sum() # no missing values
```

Out[5]:
```
GROUPID            0
CLIENTID           0
GROUPNAME          0
MERCHANT NAME      0
CLIENTNAME         0
MCC                0
ONLINEPOS          0
SCHEME             0
TRX COUNT          0
TRX_AMOUNT         0
TRX_COUNT_LTM      0
TRX_AMOUNT_LTM     0
MSC                0
MSC_LTM            0
NET MSC            0
NET MSC_LTM        0
ATV                0
ID                 0
REFUND_COUNT       0
REFUND_AMOUNT      0
REFUND_COUNT_LTM   0
REFUND_AMOUNT_LTM  0
dtype: int64
```

In [6]:
```
# Check data types of all columns
df.dtypes # 7 columns are categorical
```

Out[6]:
```
GROUPID            object
CLIENTID           object
GROUPNAME          object
MERCHANT NAME      object
CLIENTNAME         object
MCC                int64
ONLINEPOS          object
SCHEME             object
TRX COUNT          int64
TRX_AMOUNT         float64
TRX_COUNT_LTM      int64
TRX_AMOUNT_LTM     float64
MSC                float64
MSC_LTM            float64
NET MSC            float64
NET MSC_LTM        float64
ATV                float64
ID                 int64
REFUND_COUNT       float64
REFUND_AMOUNT      float64
REFUND_COUNT_LTM   int64
REFUND_AMOUNT_LTM  float64
dtype: object
```

In [7]:
```
# The first 5 columns contain categorical values that are not really relevant for the analysis, so I will drop them
df.drop(['GROUPID', 'CLIENTID', 'GROUPNAME', 'MERCHANT NAME', 'CLIENTNAME'], axis=1, inplace=True)
```

In [8]:
```
# Since the ONLINEPOS is a categorical column that contains only 2 categories (binary), we can use One Hot Encoding, that will assign 1 for POS and 0 for Online
# Apply the lambda function to the ONLINEPOS column and create a new column
df['ONLINEPOS_encoded'] = df['ONLINEPOS'].apply(lambda x:1 if x== 'POS' else 0)
```

In [9]:
```
# Delete the ONLINEPOS column
df.drop(['ONLINEPOS'], axis=1, inplace=True)
print(df)
```

```
      MCC  SCHEME  TRX COUNT  TRX_AMOUNT  TRX_COUNT_LTM  TRX_AMOUNT_LTM  \
0    5977    Visa       1032    36042.95             24       164.4513   
1    5299  Other_CC         0        0.00             11        1.3794   
2    5977      MC         0        0.00           6187       849.7844   
3    5977    Visa         0        0.00          18876      14015.6803   
4    5921    Visa         1       33.21             21         5.2929   

      MSC  MSC_LTM  NET MSC  NET MSC_LTM    ATV       ID  \
0  164.45  174751.  10.829  1264.64461  18.49  1981952   
1    0.00       1.  0.0000       0.4832  27.50  1967069   
2    0.00     849.  0.0000      67.8314  29.94  1980027   
3    0.00   14015.  0.0000    2181.8740  23.28  1980027   
4    0.27       5.  0.1505       3.2060  26.79  1991204   

   REFUND_COUNT  REFUND_AMOUNT  REFUND_COUNT_LTM  REFUND_AMOUNT_LTM  \
0             0            0.0                 0                0.0   
1             0            0.0                 0                0.0   
2             0            0.0                 0                0.0   
3             0            0.0                 0                0.0   
4             0            0.0                 0                0.0   

   ONLINEPOS_encoded  
0                  1  
1                  0  
2                  1  
3                  1  
4                  0  
[1990 rows x 17 columns]
```

In [10]:
```
# View the SCHEME column
df['SCHEME'].values
```

Out[10]:
```
array(['Visa', 'Other_CC', 'Visa', ..., 'Visa', 'Other_CC', 'MC'],
      dtype=object)
```

In [11]:
```
# To convert the categorical column SCHEME into numerical values, we will use label encoding
# With label encoding, an unique integer will be assigned for each category

# Importing the LabelEncoder from sklearn library
from sklearn.preprocessing import LabelEncoder

# Initialize the label Encoder
label_encoder = LabelEncoder()

# Apply label encoding to the SCHEME column
df['SCHEME_coded'] = label_encoder.fit_transform(df['SCHEME'])
```

In [12]:
```
# Drop the SCHEME column
df.drop(['SCHEME'], axis=1, inplace=True)
print(df)
```

```
      MCC  TRX COUNT  TRX_AMOUNT  TRX_COUNT_LTM  TRX_AMOUNT_LTM     MSC  \
0    5977       1032    36042.95             24       164.4513  164.45.  
1    5299          0        0.00             11        1.3794  0.00000  
2    5977          0        0.00           6187       849.7844  0.00000  
3    5977          0        0.00          18876      14015.6803  0.00000  
4    5921          1       33.21             21         5.2929  0.270000  

      MSC_LTM  NET MSC  NET MSC_LTM    ATV       ID  REFUND_COUNT  \
0  174751.75720  10.829  1264.64461  18.49  1981952             0  
1       1.379428  0.0000       0.4832  27.50  1967069             0  
2     849.784410  0.0000      67.8314  29.94  1980027             0  
3   14015.680340  0.0000    2181.8740  23.28  1980027             0  
4       5.292904  0.150512    3.206020  26.79  1991204             0  

   REFUND_AMOUNT  REFUND_COUNT_LTM  REFUND_AMOUNT_LTM  ONLINEPOS_encoded  \
0            0.0                 0                0.0                  1  
1            0.0                 0                0.0                  0  
2            0.0                 0                0.0                  1  
3            0.0                 0                0.0                  1  
4            0.0                 0                0.0                  0  

   SCHEME_coded  
0             2  
1             1  
2             2  
3             2  
4             2  

[1990 rows x 17 columns]
```

In [13]:
```
# Descriptive Statistics for the value of gross profit in the last year (MSC_LTM) to understand the profitability of the merchant
df['MSC_LTM'].describe().round(2)
# The average revenue that a merchant client has a gross profit in the last year of $19,505.89.
# The standard deviation has a high value of $24,600.02, meaning that the values are widely spread out from the mean of the gross profit in the last year.
# The gross profit in the last year had a minimum value of $0.40 and a maximum value of $161,760.2.
# 25% of the values of the gross profit in the last year are less than or equal to $1,144.32 (25th percentile).
# 50% of the values of the gross profit in the last year are less than or equal to $2,921.97 (median or 50th percentile).
# 75% of the values of the gross profit in the last year are less than or equal to $15,727.80 (75th percentile).
```

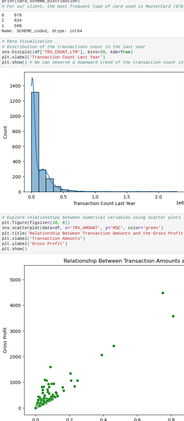
Out[14]:
```
count    1990.00
mean    19505.89
std     24600.02
min         0.45
25%      1144.32
50%      2921.97
75%     15727.95
max    161760.87
Name: MSC_LTM, dtype: float64
```

In [15]:
```
# Descriptive Statistics for the net value of the profit in the last year (NET_MSC_LTM) to understand the net profitability of the merchant after expenses and deductions.
df['NET_MSC_LTM'].describe().round(2)
# The average revenue that a merchant client has a gross profit in the last year is approximately $2,599.18.
# The standard deviation has a high value of $5,459.61, indicating considerable variability in net profits among merchants.
# The net profit in the last year had a minimum value of -$1,058.87 and a maximum value of $104,473.35.
# 25% of the values of the net profit in the last year are less than or equal to $92.18 (25th percentile).
# 50% of the values of the net profit in the last year are less than or equal to $297.17 (median or 50th percentile).
# 75% of the values of the net profit in the last year are less than or equal to $1,934.05 (75th percentile).
```

Out[15]:
```
count    1990.00
mean     2599.18
std      5459.61
min     -1058.87
25%        92.18
50%       297.17
75%      1934.05
max    104473.35
Name: NET_MSC_LTM, dtype: float64
```

In [16]:
```
# Distribution of the net profit in the last year
sns.histplot(df['NET_MSC_LTM'], bins=20, kde=True)
plt.xlabel('Net profit of the client in the last year')
plt.show()
```

Out[16]:



In [17]:
```
# Compute the frequency distribution of online and physical transactions to assess the prevalence of each type.
print(df.ONLINEPOS_encoded.value_counts())
print(online_physical_distribution)
# For our client, there are 1748 online transactions and 182 physical transactions.
```

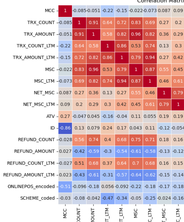
```
1    1748
0     182
Name: ONLINEPOS_encoded, dtype: int64
```

In [18]:
```
# Analyzed distribution of different card types used for transactions
card_scheme_distribution = df['SCHEME_coded'].value_counts(normalize=True) # 0 = MC, 1 - Other_CC, 2 - Visa
print(card_scheme_distribution)
# For our client, the most frequent type of card used is MasterCard (67%), followed by Visa (63%) and 598 use other types of cards.
```

```
0    0.676
2    0.326
1    0.598
Name: SCHEME_coded, dtype: int64
```

In [19]:
```
# Data visualization
# Distribution of the transactions count in the last year
sns.histplot(df['TRX_COUNT_LTM'], bins=20, kde=True)
plt.xlabel('Transaction Count Last Year')
plt.show() # we can observe a downward trend of the transaction count in the last year
```

Out[19]:



In [20]:
```
# Explore relationships between numerical variables using scatter plots
plt.figure(figsize=(10, 5))
sns.scatterplot(data=df, x='TRX_AMOUNT', y='MSC', color='green')
plt.title('Relationship Between Transaction Amounts and the Gross Profit')
plt.xlabel('Transaction Amounts')
plt.ylabel('Gross Profit')
plt.show()
```

Out[20]:

In [21]:
```
# Compute and visualize correlation matrix
correlation_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidth=0.5)
plt.title('Correlation Matrix of Variables')
plt.show()
# we can see that there is a positive and high correlation (0.86) between the gross profit (MSC) and the amount of the transactions (TRX_AMOUNT). Also, between the gross profit (
# and the amount of the transactions of the last year (0.94). This tells us that there is a direct and positive relationship between the gross profit and the value of the transacti
# Also, there are some negative correlations between the value of the refunds and the number of the refunds, indicating that there is a indirect relationship between them.
```

Out[21]: