Third part

February 25, 2023

[1]:
```
import pandas as pd
import numpy as np
```

[160]:
```
#load data first
brand = pd.read_csv("/Users/wuchenghan/Desktop/Takehome_Data_January_2023/
 ↪brands.csv",sep=',')
receipt_item =  pd.read_csv("/Users/wuchenghan/Desktop/
 ↪Takehome_Data_January_2023/receipt_items.csv",sep=',')
receipt =  pd.read_csv("/Users/wuchenghan/Desktop/Takehome_Data_January_2023/
 ↪receipts.csv",sep=',')
user =  pd.read_csv("/Users/wuchenghan/Desktop/Takehome_Data_January_2023/users.
 ↪csv",sep=',')
```

[35]:
```
# find missing value function
def missing_values_table(df):
        # Total missing values
        mis_val = df.isnull().sum()

        # Percentage of missing values
        mis_val_percent = 100 * df.isnull().sum() / len(df)

        # Make a table with the results
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)

        # Rename the columns
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

        # Sort the table by percentage of missing descending
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)

        # Print some summary information
        print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")
```

```
        # Return the dataframe with missing information
        return mis_val_table_ren_columns
```

[36]:
```
# the number of categorical and numeric
categorical_variable = receipt.select_dtypes(include=['object'])
numeric_variable = receipt.select_dtypes(exclude=['object'])
```

[162]:
```
missing_values_table(receipt)
```

```
Your selected dataframe has 21 columns.
There are 15 columns that have missing values.
```

[162]:

	Missing Values	% of Total Values
PROCESSED_DATE	70601	100.0
NEEDS_FETCH_REVIEW	70276	99.5
DELETED	69733	98.8
FLAGGED_DATE	66576	94.3
REJECTED_DATE	66217	93.8
NON_POINT_EARNING_RECEIPT	8986	12.7
USER_VIEWED	6465	9.2
FINISHED_DATE	6252	8.9
PURCHASE_TIME	4947	7.0
PURCHASE_DATE	2066	2.9
STORE_NAME	1836	2.6
TOTAL_SPENT	1492	2.1
PENDING_DATE	1453	2.1
PURCHASED_ITEM_COUNT	1452	2.1
MODIFY_DATE	2	0.0

[161]:
```
missing_values_table(receipt_item)
```

```
Your selected dataframe has 12 columns.
There are 8 columns that have missing values.
```

[161]:

	Missing Values	% of Total Values
POINTS_EARNED	341425	94.7
REWARDS_GROUP	298440	82.8
BRAND_CODE	205490	57.0
BARCODE	135369	37.6
QUANTITY_PURCHASED	7756	2.2
ORIGINAL_RECEIPT_ITEM_TEXT	1681	0.5
DESCRIPTION	1091	0.3
TOTAL_FINAL_PRICE	692	0.2

[163]:
```
missing_values_table(user)
```

```
Your selected dataframe has 8 columns.
There are 1 columns that have missing values.
```

[163]:
```
                   Missing Values  % of Total Values
SIGN_UP_PLATFORM               45               27.4
```

[164]:
```
missing_values_table(brand)
```

```
Your selected dataframe has 9 columns.
There are 5 columns that have missing values.
```

[164]:
```
                    Missing Values  % of Total Values
RELATED_BRAND_IDS              243               59.9
ROMANCE_TEXT                   103               25.4
CATEGORY_CODE                   31                7.6
CATEGORY                        27                6.7
BRAND_CODE                      25                6.2
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```
# use rfm to analyze customer behavior
```

[165]:
```
# join two table first
merged_data = pd.merge(receipt, user, left_on='USER_ID', right_on='ID',␣
↪how='inner')

#change format to date and drop rows that with NA
merged_data['PURCHASE_DATE'] = pd.to_datetime(merged_data['PURCHASE_DATE'])
merged_data['DATE_SCANNED'] = pd.to_datetime(merged_data['DATE_SCANNED'])
merged_data.dropna(subset=['PURCHASE_DATE',"TOTAL_SPENT","DATE_SCANNED"],␣
↪inplace=True)
```

[179]:
```
# Calculate Recency, Frequency, and Monetary Value for each customer
recency_df = merged_data.groupby('USER_ID').agg({'PURCHASE_DATE': lambda x:␣
↪(merged_data['PURCHASE_DATE'].max() - x.max()).days,
                                                 'TOTAL_SPENT': "sum",
                                                 "DATE_SCANNED": "count"}).
↪rename(columns={'PURCHASE_DATE': 'Recency'
                                                                          ␣
↪       , 'TOTAL_SPENT': 'MonetaryValue'
                                                                          ␣
↪       , "DATE_SCANNED": "Frequency"})

# Display the resulting DataFrame
print(recency_df)
```

```
                          Recency  MonetaryValue  Frequency
USER_ID
5fef29605b73fc128b245f36        7       72941.96       1443
5ff48849291b6b12931ce51f       12       12027.23        299
5ffb49a847903912705e9a64        6      217063.93       1709
5ffdf6f6224dc11273156070        1       15245.98       1023
6001dbb3878e221317c8a065        0        2523.17        111
…                             …              …         …
61b3af1deae64d29568176d7        9         587.41         22
61b450154d2ff607dab7c0aa        0       51087.58       1180
61b8e6ab3c43881d14ba868d        1        9119.63        310
61bfce5d6655417f803b6538        1       27263.69        912
61ce28087ef94f1a020d40ca        1       38363.74       1040

[99 rows x 3 columns]
```

[ ]:
```
# Then we can use this to basically to categorize several groups based on own␣
↪guideline.
```

[180]:
```
# For example, we can use quantile to make a total score based on these three␣
↪variable.

# Calculate RFM scores for each customer
r_labels = range(4, 0, -1)
f_labels = range(1, 5)
m_labels = range(1, 5)

r_quartiles = pd.qcut(recency_df['Recency'], q=4, labels=r_labels)
f_quartiles = pd.qcut(recency_df['Frequency'], q=4, labels=f_labels)
m_quartiles = pd.qcut(recency_df['MonetaryValue'], q=4, labels=m_labels)

recency_df = recency_df.assign(R=r_quartiles.values)
recency_df = recency_df.assign(F=f_quartiles.values)
recency_df = recency_df.assign(M=m_quartiles.values)


# Display the resulting DataFrame
print(recency_df)
```

```
                          Recency  MonetaryValue  Frequency  R  F  M
USER_ID
5fef29605b73fc128b245f36        7       72941.96       1443  1  4  4
5ff48849291b6b12931ce51f       12       12027.23        299  1  2  1
5ffb49a847903912705e9a64        6      217063.93       1709  1  4  4
5ffdf6f6224dc11273156070        1       15245.98       1023  4  4  2
6001dbb3878e221317c8a065        0        2523.17        111  4  1  1
…                             …              …         … .. .. ..
61b3af1deae64d29568176d7        9         587.41         22  1  1  1
```

61b450154d2ff607dab7c0aa	0	51087.58	1180	4	4	4
61b8e6ab3c43881d14ba868d	1	9119.63	310	4	2	1
61bfce5d6655417f803b6538	1	27263.69	912	4	3	3
61ce28087ef94f1a020d40ca	1	38363.74	1040	4	4	3

[99 rows x 6 columns]