Report on Assignment (part 1)

Group: BD-2002

Team: Alimbayeva Amina, Amangeldinova Kamila, Marat Aisaya, Miras Gaukhar

Content:

Explore the dataset

In this part we download & properly open the datasets, check their size, and do descriptive statistics.

Actually we have 5 datasets, however in this part we work only with 4 of them:

- types.csv - reference of transaction types
- codes.csv - reference of transaction codes
- transactions.csv - transactional data on banking operations
- train_set.csv - training set with client gender marking (0/1 - client gender)
- test_set.csv - no need to use.

transactions.csv columns description:

- client_id - client's id
- datetime -transaction date (format - ordered day number hh:mm:ss - 421 06:33:15)
- code - transaction code
- type - transaction type
- sum - sum of transaction

According to data of sets, we may see that it is appropriate to do descriptive statistics for object data. The result's index will include count, unique, top, and freq. The top is the most common value. The freq is the most common value's frequency. If multiple object values have the highest count, then the count and top results will be arbitrarily chosen from among those with the highest count.

```
"""
Download the dataset and check the size
"""
codes = pd.read_csv('codes.csv', sep = ';')
print(codes.shape)
codes.head()
```

(184, 2)

	code	code_description
0	5944	Магазины по продаже часов, ювелирных изделий и...
1	5621	Готовые сумочные изделия
2	5697	Услуги по переделке, починке и пошиву одежды
3	7995	Транзакции по азартным играм
4	5137	Мужская, женская и детская спец-одежда

```
"""
Do descriptive statistics (count, unique, top, freq) for object type data
"""
desc_c=codes.describe(include=['object'])
desc_c
```

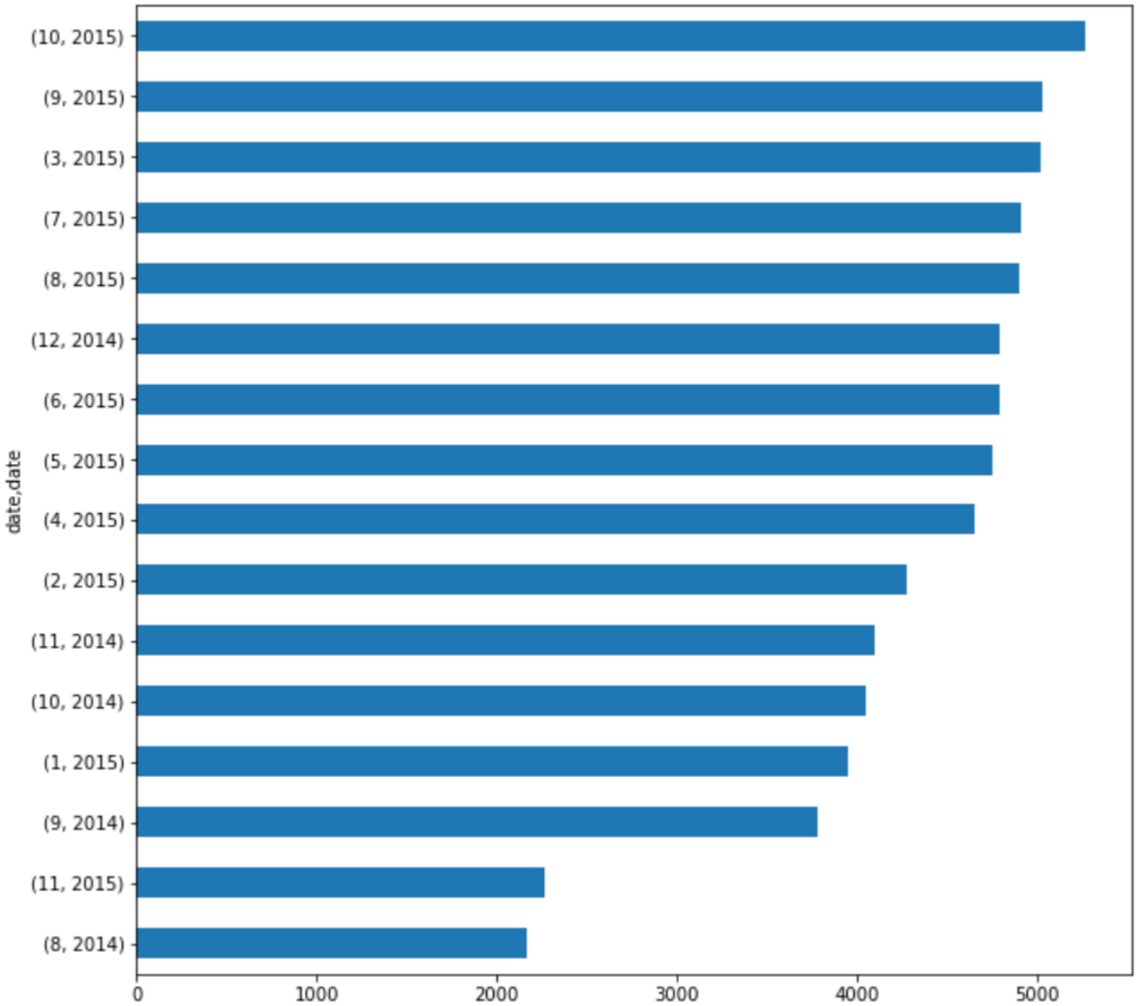
	code_description
count	184
unique	184
top	Услуги курьера — по воздуху и на земле, агентс...
freq	1

Explanatory data analysis

Here we work with visualization, explore the features of datasets.

```
transactions.index = pd.to_datetime(transactions['date'],format='%d/%m/%Y')
t=transactions.groupby(by=[transactions.index.month, transactions.index.year])["target"].aggregate('count').sort_values()
t.plot(kind='barh', figsize=(10,10))
```

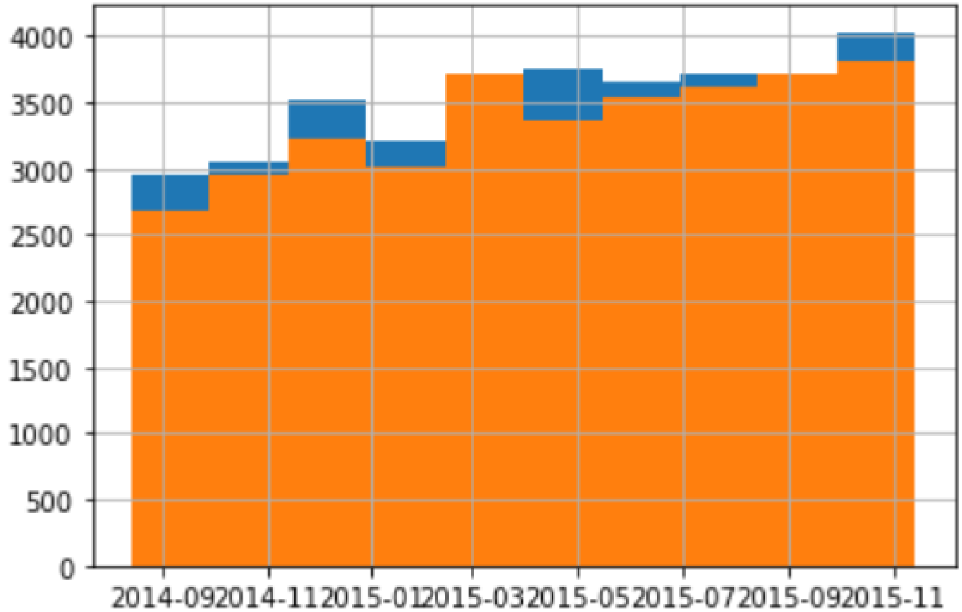
Here we group the datetime column by months and years, to make visualization for the number of transactions that were made in each month (that is available) of different years.



This histogram represents that females do more transactions than males.

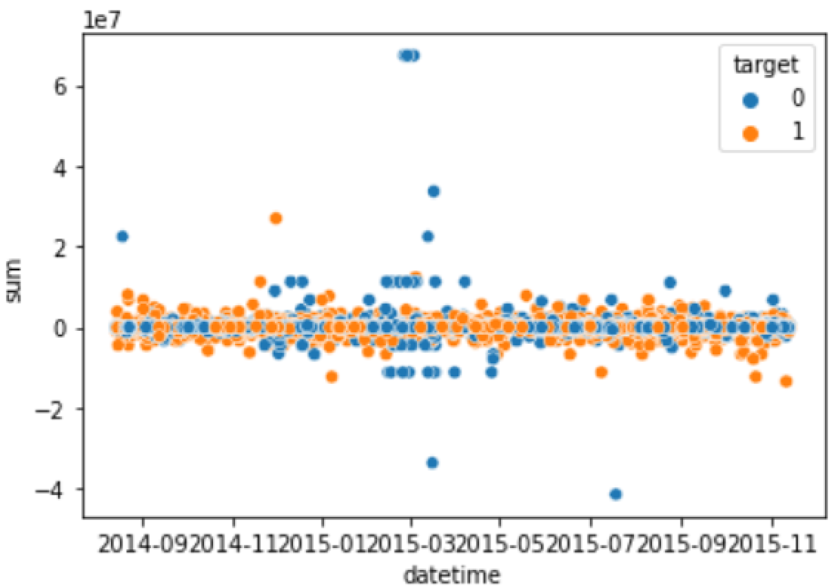
```
transactions.groupby('target').datetime.hist()
```

```
target
0    AxesSubplot(0.125,0.125;0.775x0.755)
1    AxesSubplot(0.125,0.125;0.775x0.755)
Name: datetime, dtype: object
```



Blue color (0) means male and orange color (1) means female.

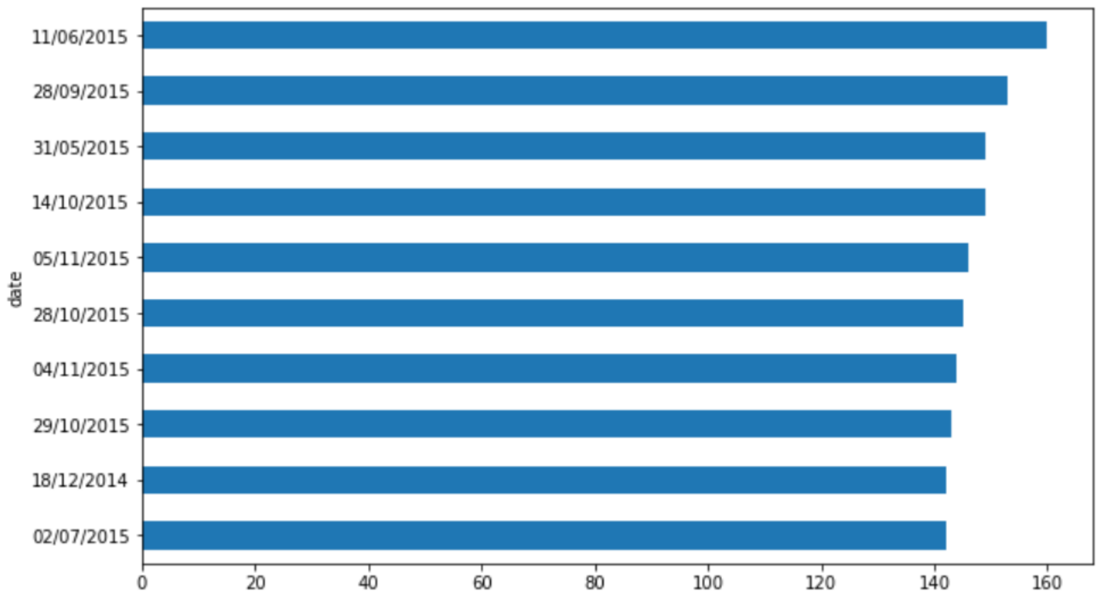
```
"""Shows that Women spend money more extreme"""
import seaborn as sns
sns.scatterplot(data=transactions, y="sum", x="datetime", hue= 'target');
```



Two graphs below illustrate the top days, when transactions were completed by females and males.

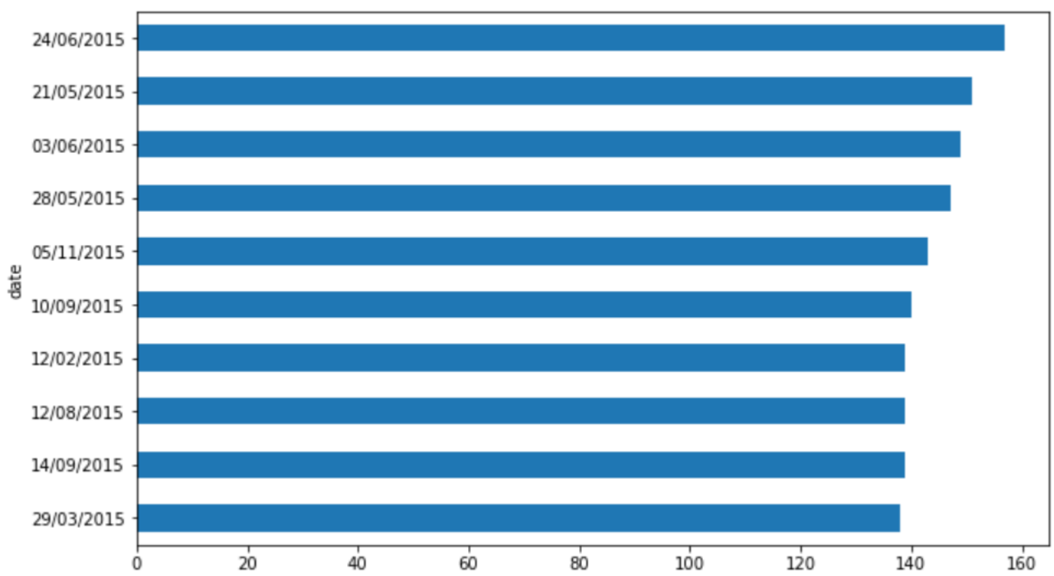
```
most_day_fem = fem_data.groupby(["date"])["client_id"].aggregate('count').sort_values().tail(10)
most_day_fem.plot(kind='barh', figsize=(10,6))
```

```
<AxesSubplot:ylabel='date'>
```




```
most_day_male = male_data.groupby(["date"])["client_id"].aggregate('count').sort_values().tail(10)
most_day_male.plot(kind='barh', figsize=(10,6))
```

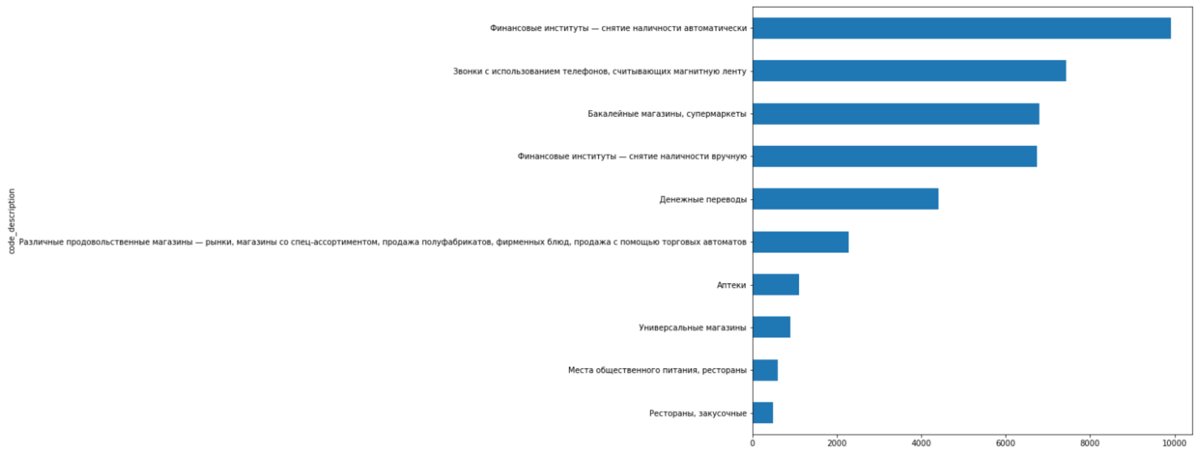
```
<AxesSubplot:ylabel='date'>
```



Here we may see which type of transaction code is the most demanded. For both genders it is automatic cash withdrawal.

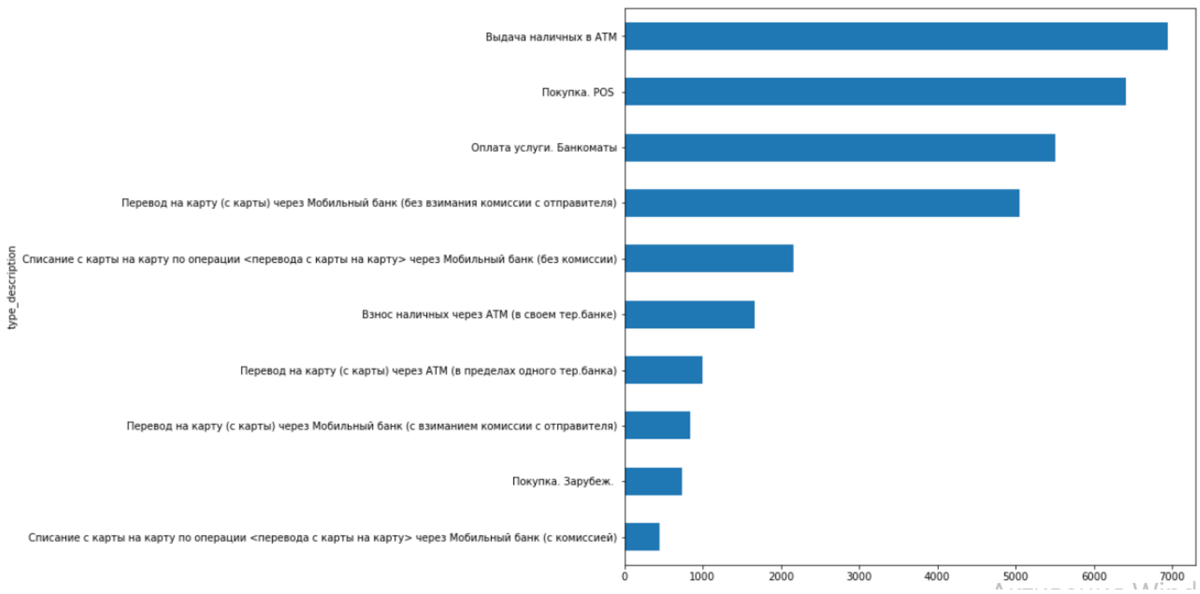
```python
most_code_fem = fem.groupby(["code_description"])["client_id"].aggregate('count').sort_values().tail(10)
most_code_fem.plot(kind='barh', figsize=(10,10))
```

```
<AxesSubplot:ylabel='code_description'>
```




```python
most_type_male = male_t.groupby(["type_description"])["client_id"].aggregate('count').sort_values().tail(10)
most_type_male.plot(kind='barh', figsize=(10,10))
```

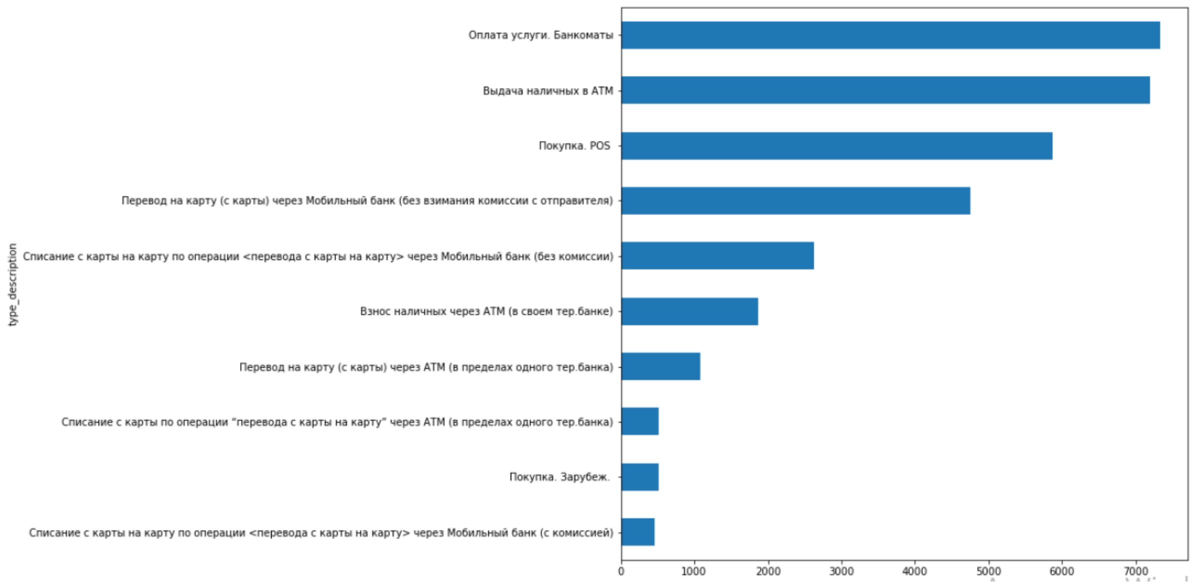
```
<AxesSubplot:ylabel='type_description'>
```



Here the most demanded type of transactions: for women is fee & ATM, for men is cash issue in ATM.

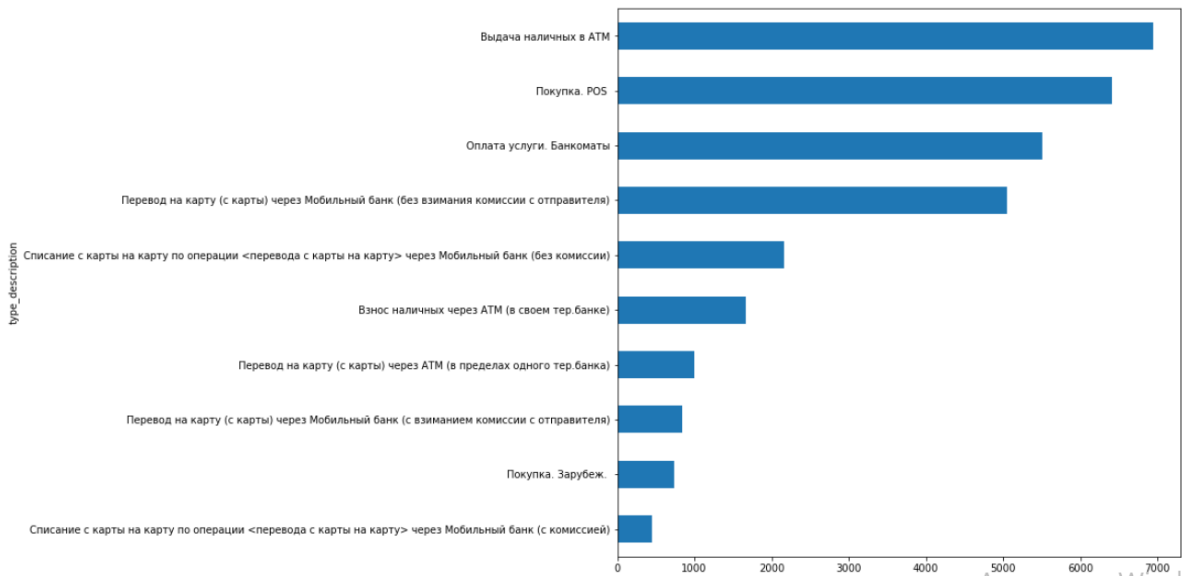
```
most_type_fem= fem_t.groupby(["type_description"])["client_id"].aggregate('count').sort_values().tail(10)
most_type_fem.plot(kind='barh', figsize=(10,10))
```

```
<AxesSubplot:ylabel='type_description'>
```




```
most_type_male = male_t.groupby(["type_description"])["client_id"].aggregate('count').sort_values().tail(10)
most_type_male.plot(kind='barh', figsize=(10,10))
```

```
<AxesSubplot:ylabel='type_description'>
```



Feature engineering

Encodings, generating the features from date-time, sum and from other columns.

For convenient further work we create additional columns, to split the datetime column into day number and time. Then we have to reformat the datetime column. First we find the start_date and end_date to identify the boundaries of date range. Then from calculations we found out that there are 457 days in set.

```python
import datetime as DT
import pandas as pd

start_date = DT.datetime(2014, 8, 14)
end_date = DT.datetime(2015, 11, 13)

res = pd.date_range(
    min(start_date, end_date),
    max(start_date, end_date)
).strftime('%d/%m/%Y').tolist()
len(res)
```

```
457
```

pandas.date_range returns a fixed frequency DatetimeIndex, the range of equally spaced time points such that they all satisfy start <= x <= end, where the first one and the last one are the first and last time points in that range that fall on the boundary of freq.

Here we decode numbers by applying the date range created by us(res)

```python
sort_date = pd.DataFrame(res, np.arange(0,457)).reset_index()

df = sort_date.rename(columns={"index":"num", 0:"date"})
df
```

	num	date
0	0	14/08/2014
1	1	15/08/2014
2	2	16/08/2014
3	3	17/08/2014
4	4	18/08/2014

Due-to this we find out the number of transactions that were made by females and males separately. As you can see, the proportions are almost equal 51% / 49% .

```python
"""Females"""
fem_data = transactions[transactions.target == 0]
fem_data
```

	client_id	datetime	code	type	sum	num	time	date	is_exists_in_train	target
0	96372458	2015-10-09 06:33:15	6011	2010	-561478.94	421	06:33:15	09/10/2015	True	0
1	96372458	2015-05-11 06:16:18	6011	7010	224591.58	270	06:16:18	11/05/2015	True	0
2	96372458	2014-12-21 05:34:06	6010	7030	224591.58	129	05:34:06	21/12/2014	True	0
3	96372458	2015-10-21 06:45:32	6011	2010	-112295.79	433	06:45:32	21/10/2015	True	0
4	96372458	2015-06-05 08:16:07	4814	1030	-11229.58	295	08:16:07	05/06/2015	True	0
...	...	...	...	...	...	...	...	...	...	...
91818	44130839	2014-09-27 11:28:18	6011	2010	-179673.26	44	11:28:18	27/09/2014	True	0
91819	43147536	2014-10-26 13:52:42	6010	7070	1122957.89	73	13:52:42	26/10/2014	True	0
91821	52382187	2015-02-27 12:33:28	6011	2010	-224591.58	197	12:33:28	27/02/2015	True	0
91824	65393099	2015-07-31 19:57:03	5921	1010	-4715.75	351	19:57:03	31/07/2015	True	0
91825	84075783	2015-07-31 08:52:31	6011	2010	-114541.70	351	08:52:31	31/07/2015	True	0

46715 rows × 10 columns

```python
"""Males"""
male_data = transactions[transactions.target == 1]
male_data
```

	client_id	datetime	code	type	sum	num	time	date	is_exists_in_train	target
13	19864270	2015-10-09 13:54:50	4814	1030	-5614.79	421	13:54:50	09/10/2015	True	1
14	19864270	2015-03-24 08:41:58	5411	1010	-14680.88	222	08:41:58	24/03/2015	True	1
15	19864270	2015-01-31 08:05:10	5499	1010	-4905.08	170	08:05:10	31/01/2015	True	1
16	19864270	2014-11-30 00:00:00	5411	1110	-17048.30	108	00:00:00	30/11/2014	True	1
17	19864270	2014-09-02 11:38:02	4814	1030	-5614.79	19	11:38:02	02/09/2014	True	1
...	...	...	...	...	...	...	...	...	...	...
91811	95055476	2014-12-16 15:04:26	4814	1030	-2245.92	124	15:04:26	16/12/2014	True	1
91817	30699304	2014-10-05 20:33:29	6010	7030	22459.16	52	20:33:29	05/10/2014	True	1
91820	18360110	2015-06-21 07:54:45	4814	1030	-2245.92	311	07:54:45	21/06/2015	True	1
91822	5188784	2015-02-27 15:59:35	6011	2010	-269509.89	197	15:59:35	27/02/2015	True	1
91823	1925153	2015-02-27 12:26:21	6011	2010	-561478.94	197	12:26:21	27/02/2015	True	1

45111 rows × 10 columns

According to this part, we may say at what time of day customers of different genders make the most number of transactions.

```python
"""Female make transactions more often these times"""
fem_data.time.value_counts().head(10)
```

```
00:00:00    4759
11:55:56       6
15:44:07       6
17:54:08       6
13:18:37       6
15:22:16       6
10:45:52       6
16:44:52       6
18:29:40       6
10:18:37       6
Name: time, dtype: int64
```

```python
"""Male make transactions more often these times"""
male_data.time.value_counts().head(10)
```

```
00:00:00    5910
14:44:47       7
13:50:37       6
12:01:13       6
10:34:09       6
14:16:40       6
16:23:34       5
10:22:57       5
16:36:52       5
18:10:07       5
Name: time, dtype: int64
```

```python
"""The sorted types dataframe shows a sequence of similar types that we can do clasterization"""
types.sort_values('type_description').head(10)
```

	type	type_description
41	999999	XXX
73	2901	Безналичное списание денежных средств со счета...
151	2320	Безналичный перевод денежных средств через POS
75	7020	Взнос наличных через POS
121	7021	Взнос наличных через POS
80	7025	Взнос наличных через POS (в других ТБ) по счет...
144	7024	Взнос наличных через POS (в своем ТБ) по счету...
69	7011	Взнос наличных через ATM
90	7015	Взнос наличных через ATM (в других ТБ) по счет...
62	7014	Взнос наличных через ATM (в своем ТБ) по счету...

While considering the sorted "types" set, we noticed that there is an unnecessary, not valuable type '999999'. That is why we may remove it.

```python
"""Deleting unnecessary values from types df"""
types.drop_duplicates(subset='type_description', inplace = True)
types = types[(types.type != 999999) & (types.type_description != 'н/д(нет данных)') & (types.type_description != 'н/д')]
types.shape
```

```
(136, 2)
```

```python
"""By this hand maded key words we can claster them into 14 groups"""
type_key_words = [
'Взнос',
'Возврат',
'Выдача',
'Зачисление Комиссия Корректировка Операции Поправки Увеличение Урегулирование Установление Безналичное Безналичный' ,
'Межфилиальные',
'Наличные',
'Оплата',
'Перевод',
'Плата',
'Платеж',
'Погашение',
'Покупка',
'Пополнение',
'Списание']
```

Doing clusterization

```
types['type_cluster'] ='null'
```

```
"""The loops for doing the clusterization"""
for k in range(14):
    for i in range(136):
        for j in types['type_description'].iloc[i].replace('.', '').replace('/', ' ').split(' '):
            for c in type_key_words[k].split(' '):
                if j == c:
                    types['type_cluster'].iloc[i] = k
```

```
types['type_cluster'] = types['type_cluster'].astype('int')
```

```
"""The clusterization of types is DONE"""
types.sort_values('type_cluster')
```

	type	type_description	type_cluster
90	7015	Взнос наличных через АТМ (в других ТБ) по счет...	0
69	7011	Взнос наличных через АТМ	0
75	7020	Взнос наличных через POS	0
62	7014	Взнос наличных через АТМ (в своем ТБ) по счету...	0
20	7010	Взнос наличных через АТМ (в своем тер.банке)	0
...	...	...	...
45	2330	Списание с карты по операции “перевода с карты...	13
44	2370	Списание с карты на карту по операции <перевод...	13
106	2342	Списание с карты по операции “перевода с карты...	13
139	8003	Списание подоходного налога с нерезидента	13
88	8100	Списание после проведения претензионной работы	13

136 rows × 3 columns

```python
"""Clusterization the transaction df"""
transactions['sum_clusters'] = 'null'
```

```python
"""Grouping into loss and profit"""
for i in range(91826 ):
    if transactions['sum'].iloc[i] > 0:
        transactions['sum_clusters'].iloc[i] = 1
    else:
        transactions['sum_clusters'].iloc[i] = 0
```

Unsupervised learning

In this part our group did the Cluster analysis, segmented the customers, did K-means, Hierarchical Clustering, visualized the clusters.
The RFM (recency, frequency, monetary value) score is a numerical score that helps you recognize all types of customers, from the best to the worst.

- Recency - How recent was the customer's last purchase? Customers who recently made a purchase will still have the product on their mind and are more likely to purchase or use the product again. Businesses often measure recency in days. But, depending on the product, they may measure it in years, weeks or even hours.
- Frequency - How often did this customer make a purchase in a given period? Customers who purchased once are often more likely to purchase again. Additionally, first time customers may be good targets for follow-up advertising to convert them into more frequent customers.
- Monetary - How much money did the customer spend in a given period? Customers who spend a lot of money are more likely to spend money in the future and have a high value to a business.

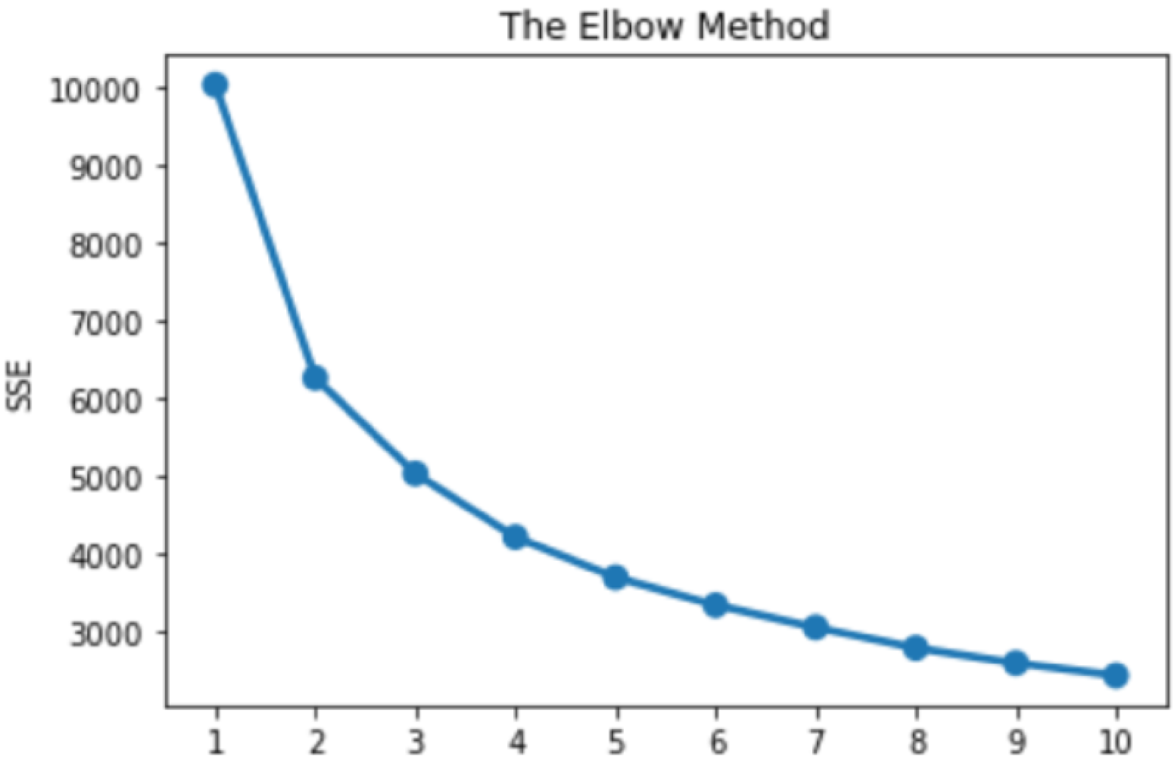
```python
###RFM score for fem data
from datetime import datetime
fem_data["datetime"] = fem_data["datetime"].dt.date


import datetime
snapshot_date = max(fem_data.datetime) + datetime.timedelta(days=1)


clients = fem_data.groupby(['client_id']).agg({
    'datetime': lambda x: (snapshot_date - x.max()).days,
    'num': 'count',
    'sum': 'sum'})


clients.rename(columns = {'datetime': 'Recency',
                          'num': 'Frequency',
                          'sum': 'MonetaryValue'}, inplace=True)
```

```python
clients
```

	Recency	Frequency	MonetaryValue
client_id			
28753	24	13	2589800.29
38084	28	26	693495.66
50940	18	3	16509.72
53395	10	1	44918.32
70680	26	10	936097.68
...	...	...	...
99876778	65	22	504212.62
99882949	64	5	256798.00
99900908	146	5	72052.13
99911226	2	12	667589.93
99985917	396	1	224591.58

3340 rows × 3 columns

Implementing “Elbow method” to determine the number of clusters in a data set, for women.



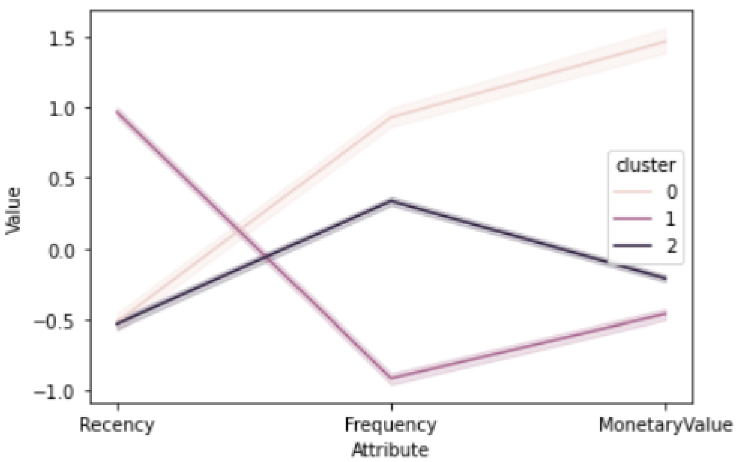
```python
clients["cluster"] = model.labels_
clients.groupby('cluster').agg({
    'Recency':'mean',
    'Frequency':'mean',
    'MonetaryValue':['mean', 'count']}).round(2)
```

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
cluster				
0	22.86	30.24	1969533.82	596
1	122.10	4.75	179564.57	1181
2	19.20	14.77	209597.94	1563

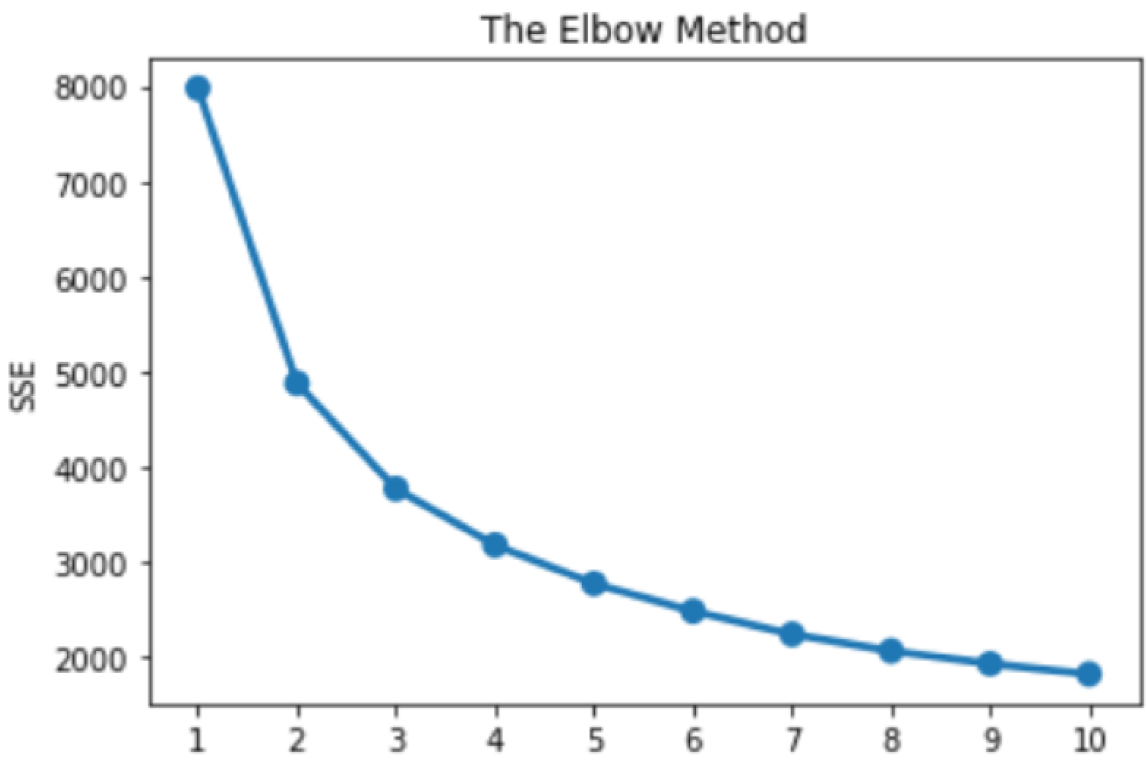
Clusterization of RFM for females

```
df_cluster = pd.DataFrame(clients_normal, columns=['Recency', 'Frequency', 'MonetaryValue'])
df_cluster['id'] = clients.index
df_cluster['cluster'] = model.labels_

df_cluster_melt = pd.melt(df_cluster.reset_index(),
                      id_vars=['id', 'cluster'],
                      value_vars=['Recency','Frequency','MonetaryValue'],
                      var_name='Attribute',
                      value_name='Value')
df_cluster_melt.head()

sns.lineplot('Attribute', 'Value', hue='cluster', data=df_cluster_melt)
```

```
C:\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following v
nly valid positional argument will be `data`, and passing other arguments without an explicit
  warnings.warn(
<AxesSubplot:xlabel='Attribute', ylabel='Value'>
```



Then we do the same for males. We added here the graphs of “Elbow method” and clusterization.



By using this plot, we know how each segment differs.

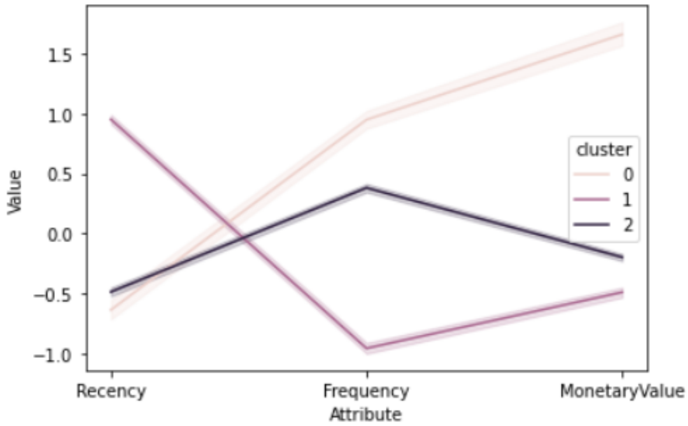
We infer that cluster 0 is the most frequent, spend most, but they do not do transactions recently. Therefore, it could be the cluster of loyal and old male clients.

Then, cluster 1 is frequent averagely, less to spend averagely, but they do not do transactions recently. Therefore, it could be the cluster of old and average class of male clients.

Finally, cluster 2 is less frequent, less to spend, but they do transactions recently. Therefore, it could be the cluster of new male clients.

```
###Clustorasation for males
df_cluster2 = pd.DataFrame(clients2_normal, columns=['Recency', 'Frequency', 'MonetaryValue'])
df_cluster2['id'] = clients2.index
df_cluster2['cluster'] = model2.labels_

df_cluster2_melt = pd.melt(df_cluster2.reset_index(),
                      id_vars=['id', 'cluster'],
                      value_vars=['Recency','Frequency','MonetaryValue'],
                      var_name='Attribute',
                      value_name='Value')
df_cluster2_melt.head()

sns.lineplot('Attribute', 'Value', hue='cluster', data=df_cluster2_melt)
```

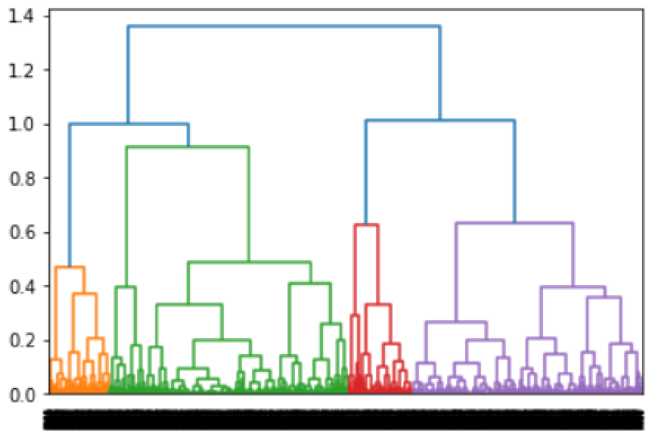
```
C:\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
nly valid positional argument will be `data`, and passing other arguments without an explicit keywo
  warnings.warn(
<AxesSubplot:xlabel='Attribute', ylabel='Value'>
```



Analyzing the results

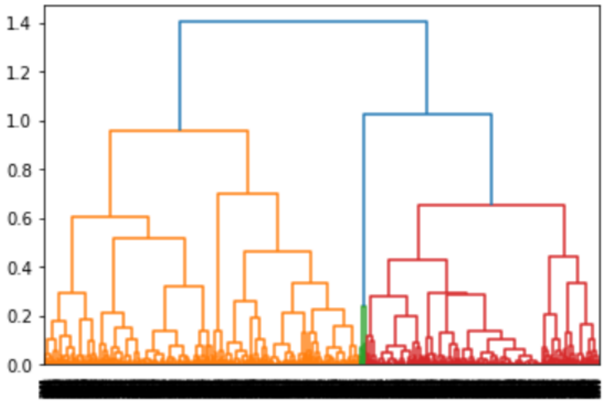
It can be seen that our HAC algorithm is divided into 3 equal clusters by average method of calculation distance. By the library scipy we have given clients_normal array (which is women data) into the dendrogram function, and we divided women data by Recency, Frequency and Transaction value into 3 categories such as in KMeans by Elbow methodology, where we can strengthen the appropriate cluster numbers by two clustering approaches.

```python
### Hierarchical clustering female data
import scipy.cluster.hierarchy as model
dend_max = model.dendrogram(model.linkage(clients_normal, method='average',metric='cosine'))
```

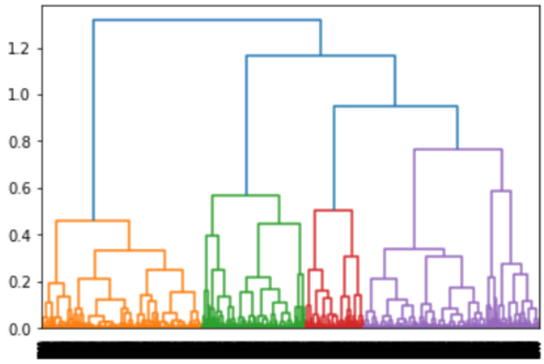


Here is a dendrogram which shows us approximated cluster numbers by HAC algorithm for male gender. It can be seen that the number of clusters is similarly 2 and we have a small cluster which is green one. By the Kmeans where we did below there by elbow method we can also choose 2 clusters, because by SSE value we have chosen similar for both values. In our situation it is SSE=5000. So for Women SSE = 5000 it is 3 clusters and for men SSE=5000 it is 2 clusters, after this type of assumption we can say that women make more transactions than men and also women make varied types of transactions than men. That is the reason why women data have more max SSE value than men.

```python
### Hierarchical clustering male data
import scipy.cluster.hierarchy as model
dend_max2=model.dendrogram(model.linkage(clients2_normal, method='average',metric='cosine'))
```



```python
### Hierarchical clustering for all data
import scipy.cluster.hierarchy as model
dend_max_total=model.dendrogram(model.linkage(total_clients_normal, method='average',metric='cosine'))
```



Conclusion

In conclusion, to reach this type of assumption at the beginning, some statistics have been made, to analyze and to see approximately what type of information gives us the data. There can be such types of feature engineering, visualizations. Moreover, our method of analysis was based on customer segmentation and identifying behavior of transactions male and female genders by using RFM methodology and clustering techniques (HAC, KMeans). After several research approaches we made some decisions regarding the data for both genders.
By decisions, we can say that women's transactions more variably than men's transactions and we had to find out behavior of the clients by gender for future data analysis.