Content

Content statement of the problem. Data Description

One international company provided data for 2010 on sales of certain products in different countries where it has branches. Its task is to see what part of the buyers often use their goods.

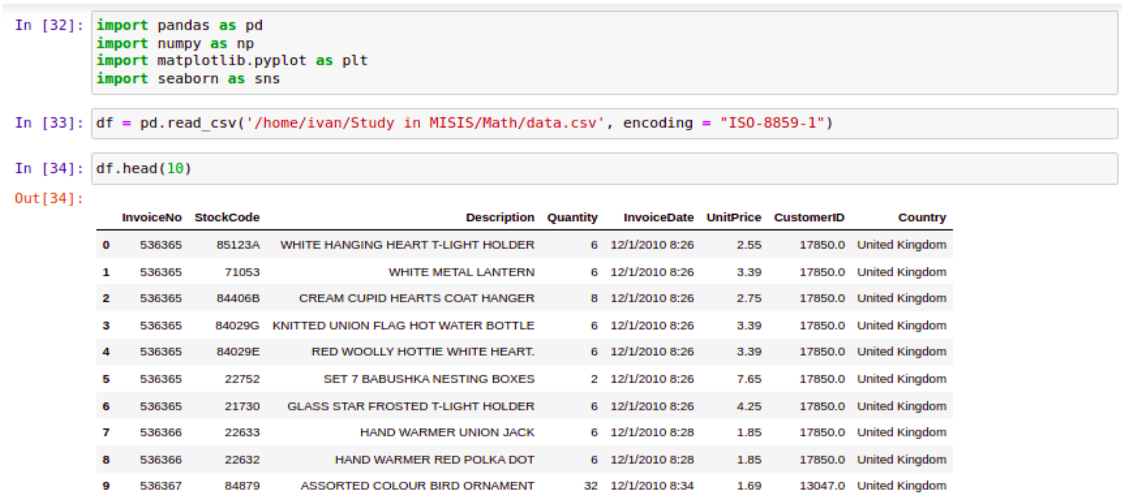
For this purpose, the company provided purchase data for a certain period. The data itself consists of 541,909 rows of records and 8 columns. Each line of the record includes information about the buyer and the product that he purchased.

1. InvoiceNo - Invoice number;
2. StockCode - stock code;
3. Description - Description;
4. Quantity - Quantity;
5. InvoiceDate - Date of purchase;
6. UnitPrice - Quantity per unit;
7. CustomerID - Customer number;
8. Country - Country of purchase;

Data source - data was taken from an open source from the site Kaggel.com

Description of work

Based on the problem statement, it was decided to resort to cohort analysis, since this method makes it easy to identify groups of buyers into the groups we need in a certain period of time, and it also allows us to analyze customer loyalty quickly and clearly.



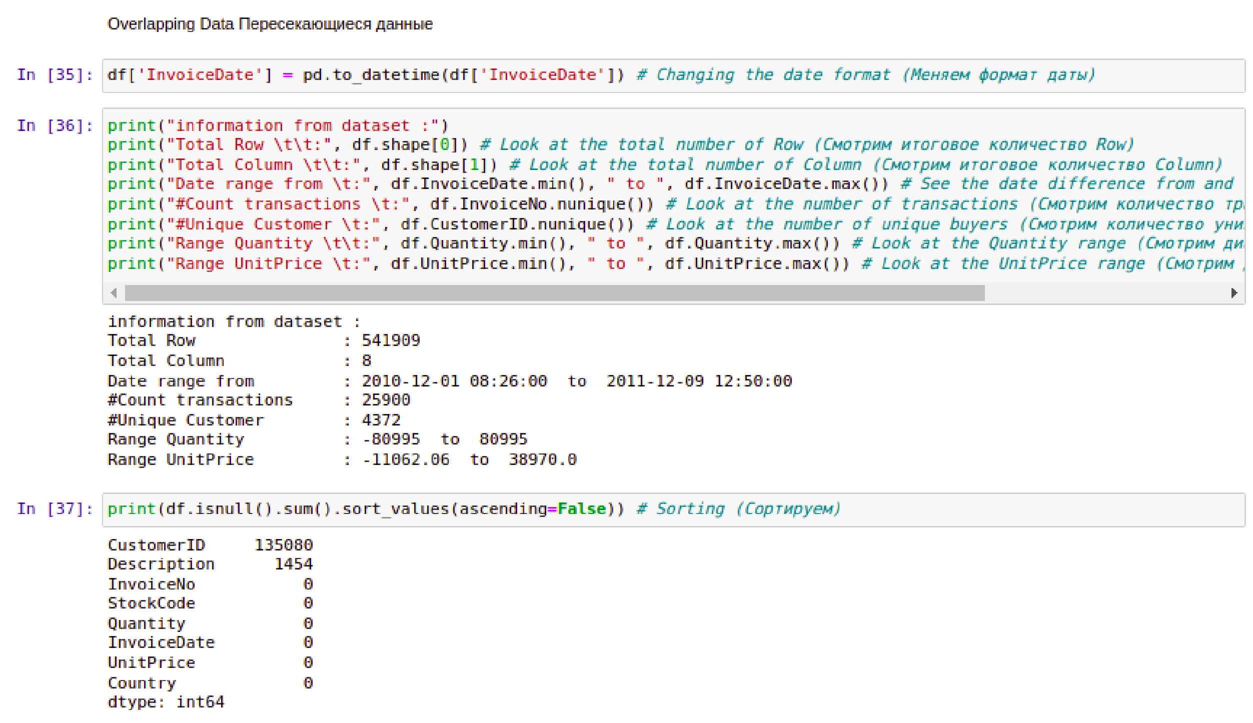
```
In [32]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns

In [33]: df = pd.read_csv('/home/ivan/Study in MISIS/Math/data.csv', encoding = "ISO-8859-1")

In [34]: df.head(10)
Out[34]:
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047.0	United Kingdom

Picture 1 – Importing and viewing data



Overlapping Data Пересекающиеся данные

```
In [35]: df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate']) # Changing the date format (Меняем формат даты)
```

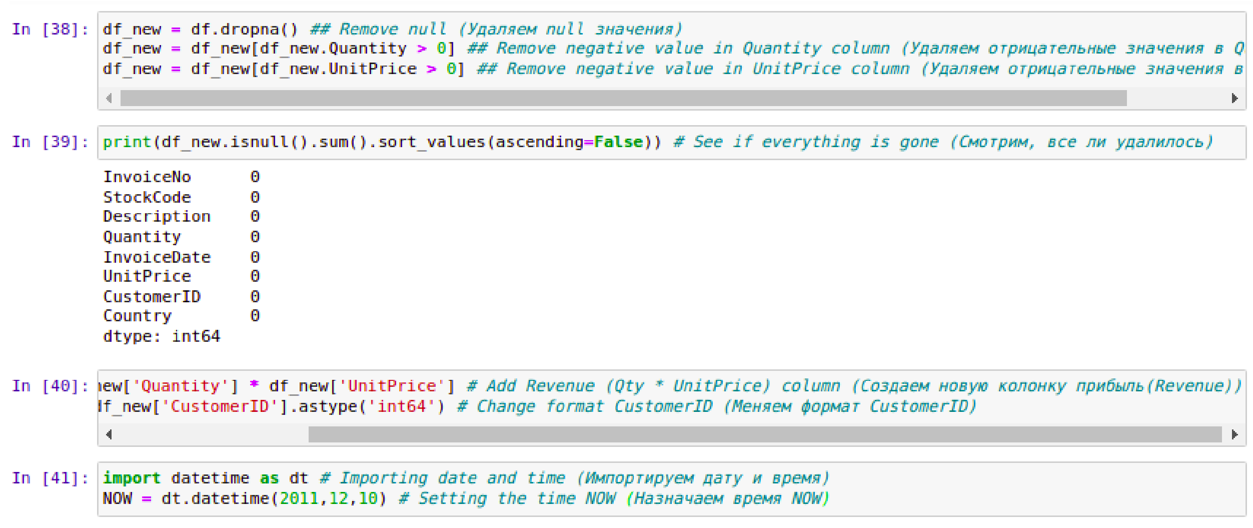
```
In [36]: print("information from dataset :")
         print("Total Row \t\t:", df.shape[0]) # Look at the total number of Row (Смотрим итоговое количество Row)
         print("Total Column \t\t:", df.shape[1]) # Look at the total number of Column (Смотрим итоговое количество Column)
         print("Date range from \t:", df.InvoiceDate.min(), " to ", df.InvoiceDate.max()) # See the date difference from and
         print("#Count transactions \t:", df.InvoiceNo.nunique()) # Look at the number of transactions (Смотрим количество тр
         print("#Unique Customer \t:", df.CustomerID.nunique()) # Look at the number of unique buyers (Смотрим количество уни
         print("Range Quantity \t\t:", df.Quantity.min(), " to ", df.Quantity.max()) # Look at the Quantity range (Смотрим ди
         print("Range UnitPrice \t:", df.UnitPrice.min(), " to ", df.UnitPrice.max()) # Look at the UnitPrice range (Смотрим
```

```
information from dataset :
Total Row                : 541909
Total Column             : 8
Date range from          : 2010-12-01 08:26:00  to  2011-12-09 12:50:00
#Count transactions      : 25900
#Unique Customer         : 4372
Range Quantity           : -80995  to  80995
Range UnitPrice          : -11062.06  to  38970.0
```

```
In [37]: print(df.isnull().sum().sort_values(ascending=False)) # Sorting (Сортируем)
```

```
CustomerID     135080
Description      1454
InvoiceNo           0
StockCode           0
Quantity            0
InvoiceDate         0
UnitPrice           0
Country             0
dtype: int64
```

Picture 2 – examine data and sort



```
In [38]: df_new = df.dropna() ## Remove null (Удаляем null значения)
         df_new = df_new[df_new.Quantity > 0] ## Remove negative value in Quantity column (Удаляем отрицательные значения в Q
         df_new = df_new[df_new.UnitPrice > 0] ## Remove negative value in UnitPrice column (Удаляем отрицательные значения в
```

```
In [39]: print(df_new.isnull().sum().sort_values(ascending=False)) # See if everything is gone (Смотрим, все ли удалилось)
```

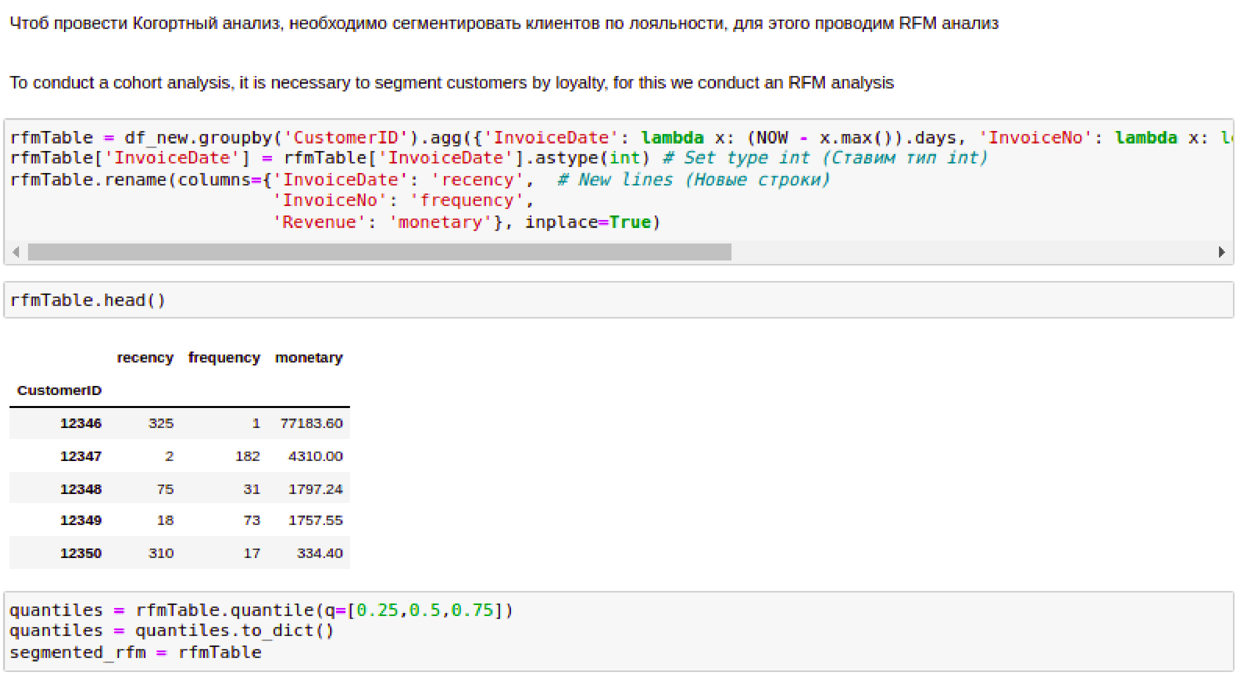
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

```
In [40]: iew['Quantity'] * df_new['UnitPrice'] # Add Revenue (Qty * UnitPrice) column (Создаем новую колонку прибыль(Revenue))
         lf_new['CustomerID'].astype('int64') # Change format CustomerID (Меняем формат CustomerID)
```

```
In [41]: import datetime as dt # Importing date and time (Импортируем дату и время)
         NOW = dt.datetime(2011,12,10) # Setting the time NOW (Назначаем время NOW)
```

Picture 3 – delete unnecessary

We also create a new profit column by simply multiplying the quantity by the price.


Чтоб провести Когортный анализ, необходимо сегментировать клиентов по лояльности, для этого проводим RFM анализ

To conduct a cohort analysis, it is necessary to segment customers by loyalty, for this we conduct an RFM analysis

```
rfmTable = df_new.groupby('CustomerID').agg({'InvoiceDate': lambda x: (NOW - x.max()).days, 'InvoiceNo': lambda x: l
rfmTable['InvoiceDate'] = rfmTable['InvoiceDate'].astype(int) # Set type int (Ставим тип int)
rfmTable.rename(columns={'InvoiceDate': 'recency',  # New lines (Новые строки)
                         'InvoiceNo': 'frequency',
                         'Revenue': 'monetary'}, inplace=True)
```

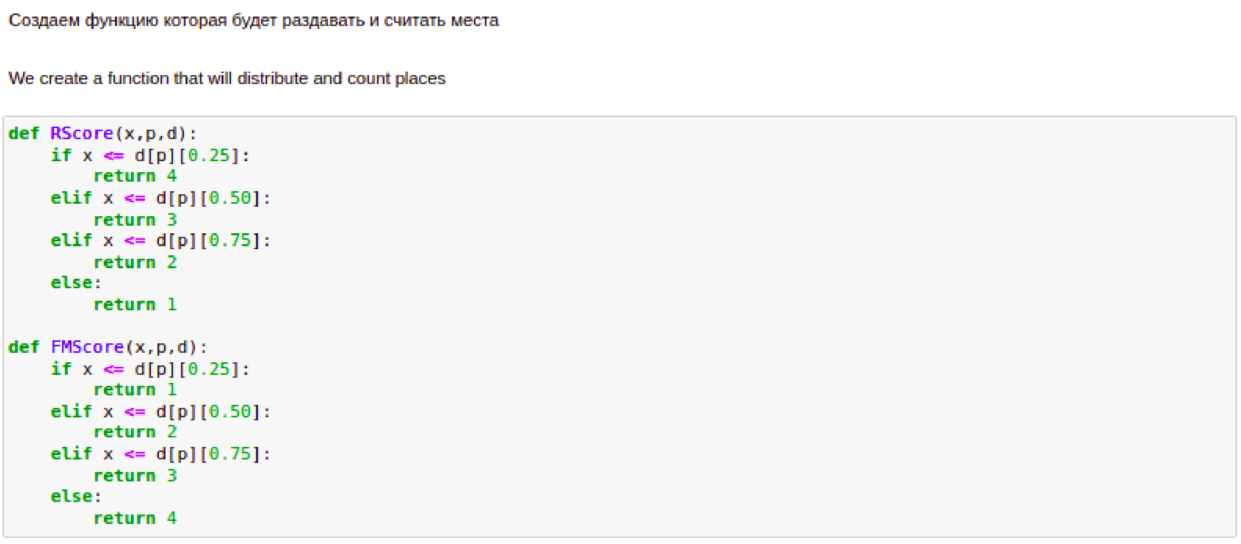
```
rfmTable.head()
```

CustomerID	recency	frequency	monetary
12346	325	1	77183.60
12347	2	182	4310.00
12348	75	31	1797.24
12349	18	73	1757.55
12350	310	17	334.40

```
quantiles = rfmTable.quantile(q=[0.25,0.5,0.75])
quantiles = quantiles.to_dict()
segmented_rfm = rfmTable
```

Picture 4 – RFM

Then, in order to conduct a loyalty analysis and build cohorts, we conduct an RFM analysis. RFM analysis is a method that allows you to segment customers by frequency, amount of purchases and identify those customers who bring more money. Where R - Recency - prescription (how long ago customers made a purchase), F - Frequency - frequency (how often they buy something from us), M - Monetary - money (total amount of purchases). Based on these very features, a cohort analysis will be carried out in the future.

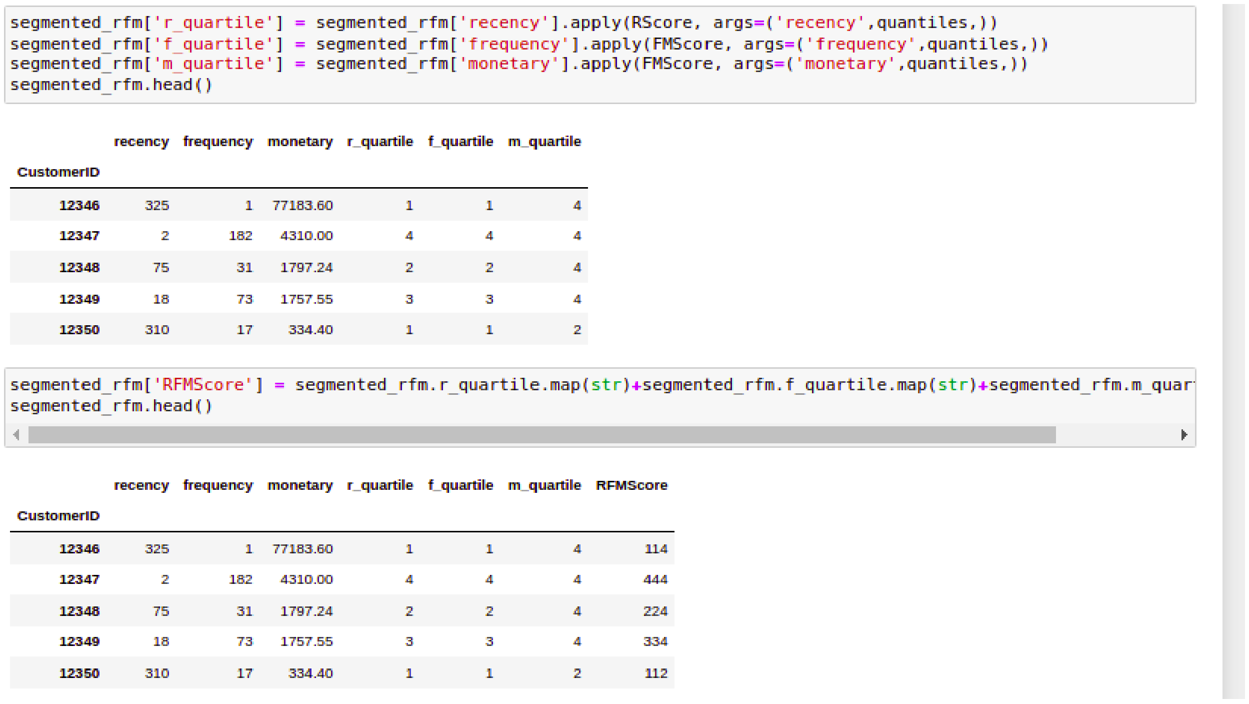

Создаем функцию которая будет раздавать и считать места

We create a function that will distribute and count places

```
def RScore(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1

def FMScore(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4
```

Picture 5 – quantile calculation function



```
segmented_rfm['r_quartile'] = segmented_rfm['recency'].apply(RScore, args=('recency',quantiles,))
segmented_rfm['f_quartile'] = segmented_rfm['frequency'].apply(FMScore, args=('frequency',quantiles,))
segmented_rfm['m_quartile'] = segmented_rfm['monetary'].apply(FMScore, args=('monetary',quantiles,))
segmented_rfm.head()
```

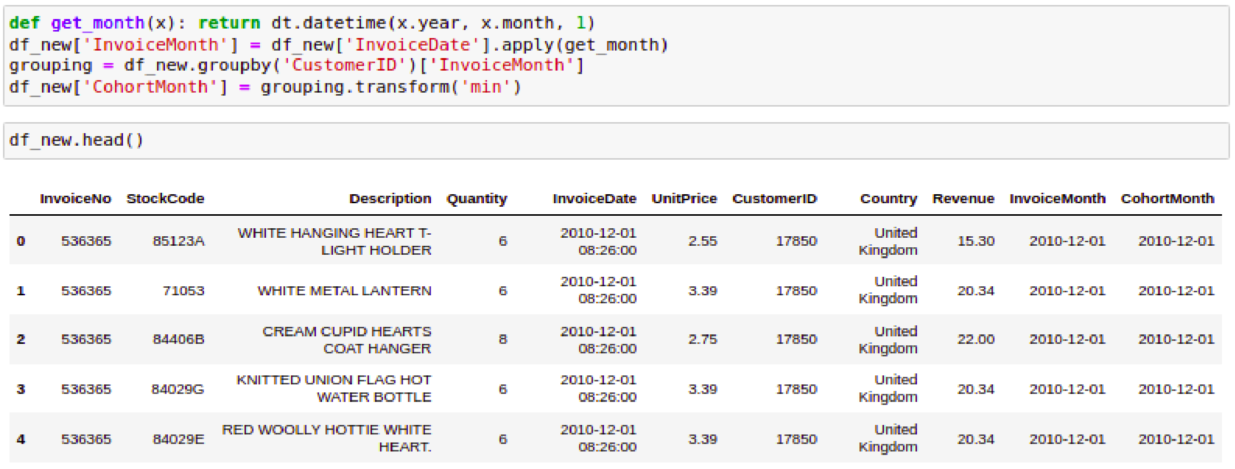
	recency	frequency	monetary	r_quartile	f_quartile	m_quartile
CustomerID						
12346	325	1	77183.60	1	1	4
12347	2	182	4310.00	4	4	4
12348	75	31	1797.24	2	2	4
12349	18	73	1757.55	3	3	4
12350	310	17	334.40	1	1	2

```
segmented_rfm['RFMScore'] = segmented_rfm.r_quartile.map(str)+segmented_rfm.f_quartile.map(str)+segmented_rfm.m_quar
segmented_rfm.head()
```

	recency	frequency	monetary	r_quartile	f_quartile	m_quartile	RFMScore
CustomerID							
12346	325	1	77183.60	1	1	4	114
12347	2	182	4310.00	4	4	4	444
12348	75	31	1797.24	2	2	4	224
12349	18	73	1757.55	3	3	4	334
12350	310	17	334.40	1	1	2	112

Picture 6 – quantiles and RFM

We then create a function that takes the month from our data column and sorts it. Then we look at what we have.



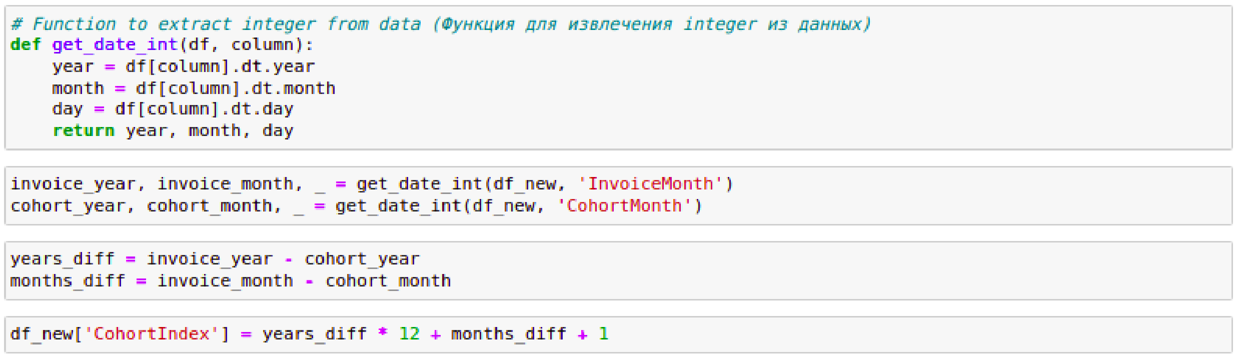
```
def get_month(x): return dt.datetime(x.year, x.month, 1)
df_new['InvoiceMonth'] = df_new['InvoiceDate'].apply(get_month)
grouping = df_new.groupby('CustomerID')['InvoiceMonth']
df_new['CohortMonth'] = grouping.transform('min')
```

```
df_new.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Revenue	InvoiceMonth	CohortMonth
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom	15.30	2010-12-01	2010-12-01
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34	2010-12-01	2010-12-01
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom	22.00	2010-12-01	2010-12-01
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34	2010-12-01	2010-12-01
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34	2010-12-01	2010-12-01

Picture 7 – month function

Then we need a date extraction function from the resulting “dataset”, which returns these dates to us. This is all done to build a table of cohorts, relative to dates.



```
# Function to extract integer from data (Функция для извлечения integer из данных)
def get_date_int(df, column):
    year = df[column].dt.year
    month = df[column].dt.month
    day = df[column].dt.day
    return year, month, day
```

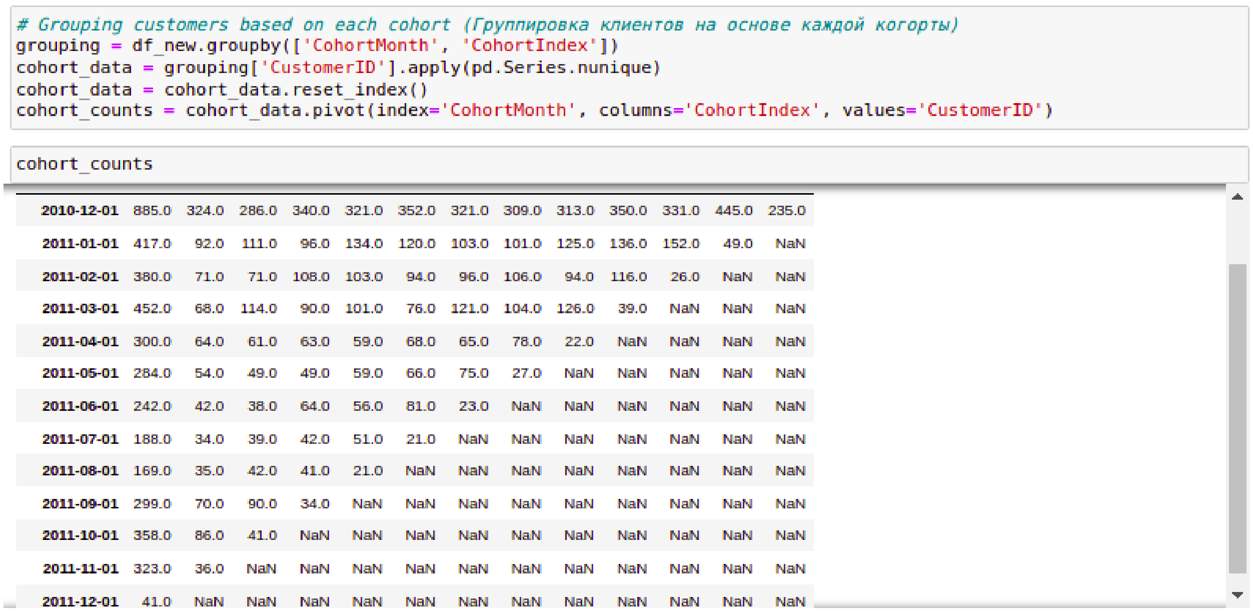
```
invoice_year, invoice_month, _ = get_date_int(df_new, 'InvoiceMonth')
cohort_year, cohort_month, _ = get_date_int(df_new, 'CohortMonth')
```

```
years_diff = invoice_year - cohort_year
months_diff = invoice_month - cohort_month
```

```
df_new['CohortIndex'] = years_diff * 12 + months_diff + 1
```

Picture 8 – function to extract int from data

And now, you can build cohorts by grouping customers based on each cohort.



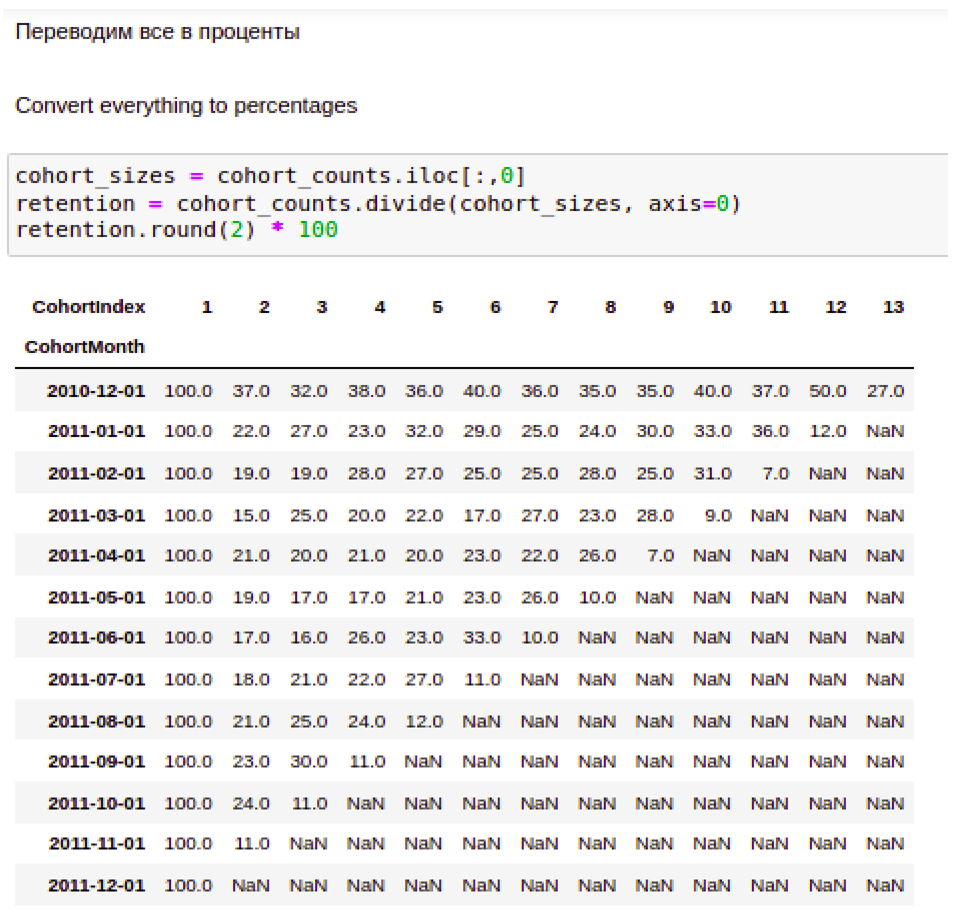
```
# Grouping customers based on each cohort (Группировка клиентов на основе каждой когорты)
grouping = df_new.groupby(['CohortMonth', 'CohortIndex'])
cohort_data = grouping['CustomerID'].apply(pd.Series.nunique)
cohort_data = cohort_data.reset_index()
cohort_counts = cohort_data.pivot(index='CohortMonth', columns='CohortIndex', values='CustomerID')
```

```
cohort_counts
```

2010-12-01	885.0	324.0	286.0	340.0	321.0	352.0	321.0	309.0	313.0	350.0	331.0	445.0	235.0
2011-01-01	417.0	92.0	111.0	96.0	134.0	120.0	103.0	101.0	125.0	136.0	152.0	49.0	NaN
2011-02-01	380.0	71.0	71.0	108.0	103.0	94.0	96.0	106.0	94.0	116.0	26.0	NaN	NaN
2011-03-01	452.0	68.0	114.0	90.0	101.0	76.0	121.0	104.0	126.0	39.0	NaN	NaN	NaN
2011-04-01	300.0	64.0	61.0	63.0	59.0	68.0	65.0	78.0	22.0	NaN	NaN	NaN	NaN
2011-05-01	284.0	54.0	49.0	49.0	59.0	66.0	75.0	27.0	NaN	NaN	NaN	NaN	NaN
2011-06-01	242.0	42.0	38.0	64.0	56.0	81.0	23.0	NaN	NaN	NaN	NaN	NaN	NaN
2011-07-01	188.0	34.0	39.0	42.0	51.0	21.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-08-01	169.0	35.0	42.0	41.0	21.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-09-01	299.0	70.0	90.0	34.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-10-01	358.0	86.0	41.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-11-01	323.0	36.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-12-01	41.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Picture 9 – formed cohorts

Further, for a more understandable display, we translate everything into percentages and build a heat map for clarity.



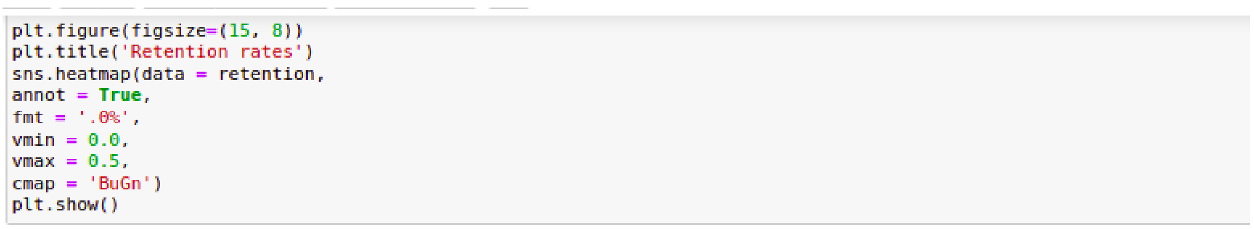
Переводим все в проценты

Convert everything to percentages

```
cohort_sizes = cohort_counts.iloc[:,0]
retention = cohort_counts.divide(cohort_sizes, axis=0)
retention.round(2) * 100
```

CohortIndex	1	2	3	4	5	6	7	8	9	10	11	12	13
CohortMonth													
2010-12-01	100.0	37.0	32.0	38.0	36.0	40.0	36.0	35.0	35.0	40.0	37.0	50.0	27.0
2011-01-01	100.0	22.0	27.0	23.0	32.0	29.0	25.0	24.0	30.0	33.0	36.0	12.0	NaN
2011-02-01	100.0	19.0	19.0	28.0	27.0	25.0	25.0	28.0	25.0	31.0	7.0	NaN	NaN
2011-03-01	100.0	15.0	25.0	20.0	22.0	17.0	27.0	23.0	28.0	9.0	NaN	NaN	NaN
2011-04-01	100.0	21.0	20.0	21.0	20.0	23.0	22.0	26.0	7.0	NaN	NaN	NaN	NaN
2011-05-01	100.0	19.0	17.0	17.0	21.0	23.0	26.0	10.0	NaN	NaN	NaN	NaN	NaN
2011-06-01	100.0	17.0	16.0	26.0	23.0	33.0	10.0	NaN	NaN	NaN	NaN	NaN	NaN
2011-07-01	100.0	18.0	21.0	22.0	27.0	11.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-08-01	100.0	21.0	25.0	24.0	12.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-09-01	100.0	23.0	30.0	11.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-10-01	100.0	24.0	11.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-11-01	100.0	11.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-12-01	100.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Picture 10 – convert to percentage



```
plt.figure(figsize=(15, 8))
plt.title('Retention rates')
sns.heatmap(data = retention,
annot = True,
fmt = '.0%',
vmin = 0.0,
vmax = 0.5,
cmap = 'BuGn')
plt.show()
```

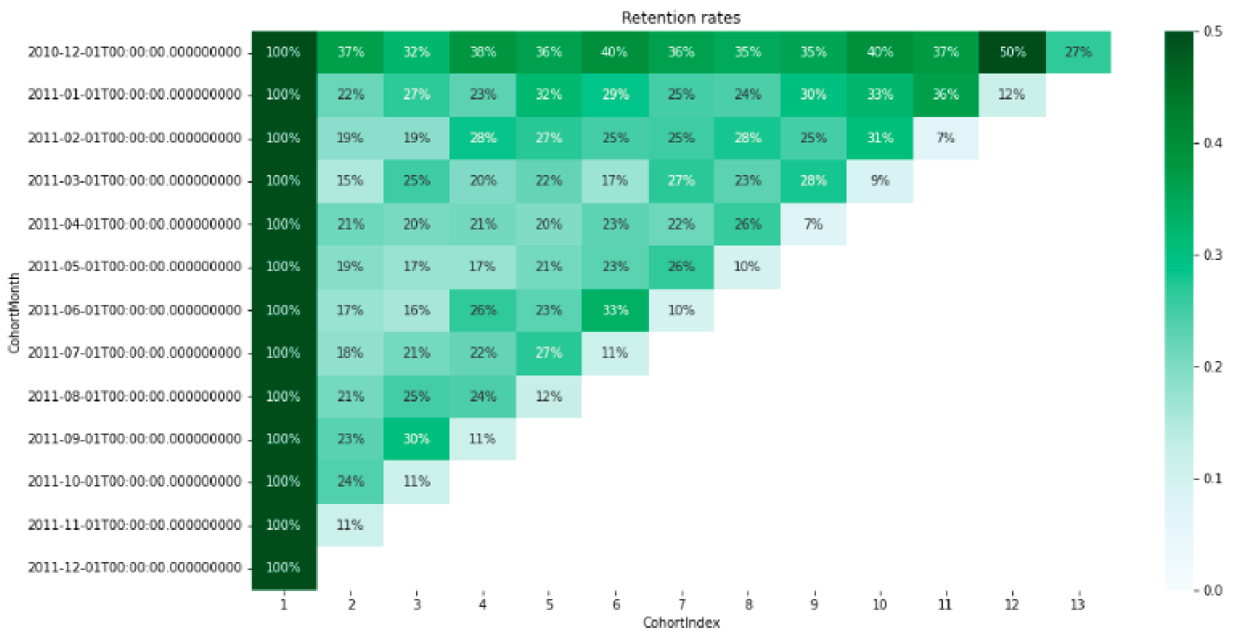


Рисунок 11 – heat map

Work results. Output

The first column of the cohort, where all values are 100%, is not taken into account, because in our new dataset for building the cohort there is no difference per month, only per year, so there is 100% everywhere.

This cohort describes how many buyers (percentage of them) are left, still buying, making a profit, and this is all broken down by month.

The first and obvious conclusion is that in December, the number of buyers and purchased goods is growing and very high, April and February were the worst months for this indicator, most of the customers make purchases at the end of the month and in the middle, and in theory, this is due to the fact that most of the people receive a salary at this particular time. In the remaining months, the number of customers and, accordingly, purchases remains approximately at the same level.