E-commerce project

**Definition of purchase**

According this data purchase is an order with a unique value from column “order\_id” in the table “olist\_orders\_dataset.csv”, in which the next condition is met:

* the value from the column “order\_approved\_at” of the table “olist\_orders\_dataset.csv” does not equal NaN. So, the payment has been confirmed and it has date.

Other conditions will be redundant, because it includes the column “order\_approved\_at” != “NaN”.

**1 task**

"How many users do we have who have made a purchase only once?"

To answer this question, firstly, we need to upload data from two tables: «olist\_customers\_datase.csv» and «olist\_orders\_dataset.csv». Secondly, we merge these two tables using the "inner merge" method to the identical column "customer\_id" to get all the data from both tables into one "customers\_and\_orders" table.

Then we leave only users, which has made a purchase. To do it, we check the column “order\_approved\_at” in the table “customers\_and\_orders”. If there is no data (“NaN”) we must have flag “True”, else it is “False”. It is pandas series with name “NaN\_in”. The series must be merged with column “customer\_id” in “customers\_and\_orders” table by “inner merge” method. As a result we call this dataframe “NaN\_in\_order\_approved\_at”.

Then we must delete users, which have “NaN” in the “order\_approved\_at”. So, we merge table «customers\_and\_orders» with «NaN\_in\_order\_approved\_at» by method “inner merge”. In the received dataframe we choose only orders, which have “False”. In the end we get the datafrqme “orders\_purchase”, where all orders are purchers.

To answer the main question of the task, we calculate amount of unique orders for every unique user. It is in table “orders\_purchase”. And in the finish we choose only users with “1” in the “count\_of\_orders” column. We have 93049 users.

To sum up, 93049 users have done purchase only one time. All amount of users is 96096.

**2 task**

"On average, how many orders are not delivered per month for various reasons (display details for reasons)"?

To answer this question, we first upload data from the table  
" olist\_orders\_dataset.csv». Then we remove orders that are delivered, because we are only interested in those that are not delivered. We get the dataframe "no\_deliveres".

We change the data type in the "order\_estimated\_delivery\_date" column, which has the "object" type, to "datetime" for working with dates. Also we change this data to such a form that the year and month remain the same, and the days become unnecessary and take the value "01".

Next, we count the number of undelivered orders in each of the observation months, for each of the reasons (column "order\_status"). Then, based on this data, we calculate the average for each for reasons of orders (statuses).

We sort the resulting values and create a bar chart adjusting the chart size.

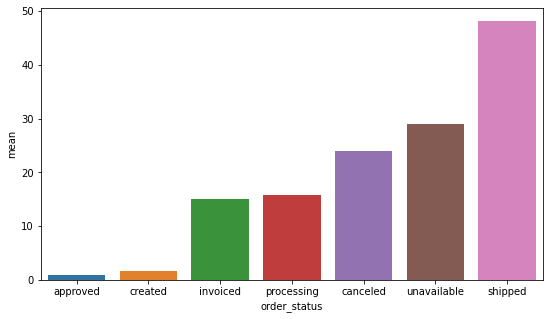


Figure 1 Bar chart of the average number of orders per month depending on the reason.

Exact values are presented in table 1.

Table 1 Average number of orders per month depending on the reason

|  |  |
| --- | --- |
| Reason | Average number per month |
| Created (only created) | 1.67 |
| Approved (only confirmed) | 1.00 |
| Invoiced (only issued an invoice) | 14.95 |
| Processing (in the process of assembling an order) | 15.84 |
| Shipped (only shipped from the warehouse) | 48.13 |
| Canceled (unavailable) | 24.04 |
| Unavailable (canceled) | 29.00 |

**3 task**

"For each product, determine on which day of the week the product is most often purchased."

Firstly, we upload data from two tables.  
«olist\_order\_items\_dataset.csv» and «olist\_orders\_dataset.csv». Then we merge these two tables using the "inner merge" on "order\_id\_id" column to get all the data from both tables into one "items\_and\_orders" table.

Secondly, we leave only users, which has made a purchase. It is like in 1 task. And we get the table “only\_purches”, where only users with purchase.

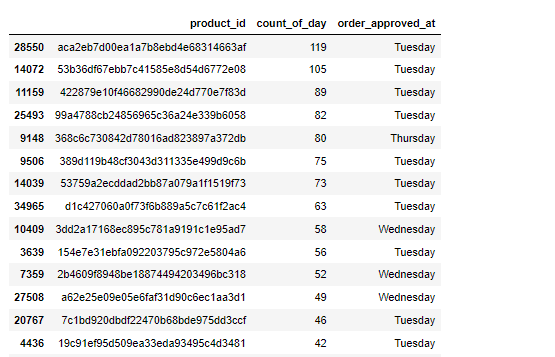
Thirdly, we change the column type «order\_approved\_at» from “object” to “datatime” and convert the date into day of the week by “dt.strftime(%A)”.

Next, we find the number of times when each item was included in the purchase for each day of the week. We write the result to the table: "day\_of\_week\_and\_product".

Then, we find the maximum value from the number of times when each product was included in purchases and write it to the table "max\_day\_of\_week\_and\_product".

We have lost the days of the week, because they were not needed in the calculation, so we attach the data by day of the week to the table "max\_day\_of\_week\_and\_product". We get the "total" table, which has the answer to the task.

Table 1 First 14 values of the table "Total" for each user and the day of the week with the highest number of purchases of this product



**4 task**

“How many purchases do each user have on average per week (by month)?”

As in the third task, we get the dataframe "only\_purchers" of orders which are purchases. Here we create column "Month", as a copy of the column "order\_approved\_at" with the data type "datetime" and the display format as the ordinal number of the month in the year. In the same way, we create the "Year" column with the year display format, so as not to confuse months from different years.

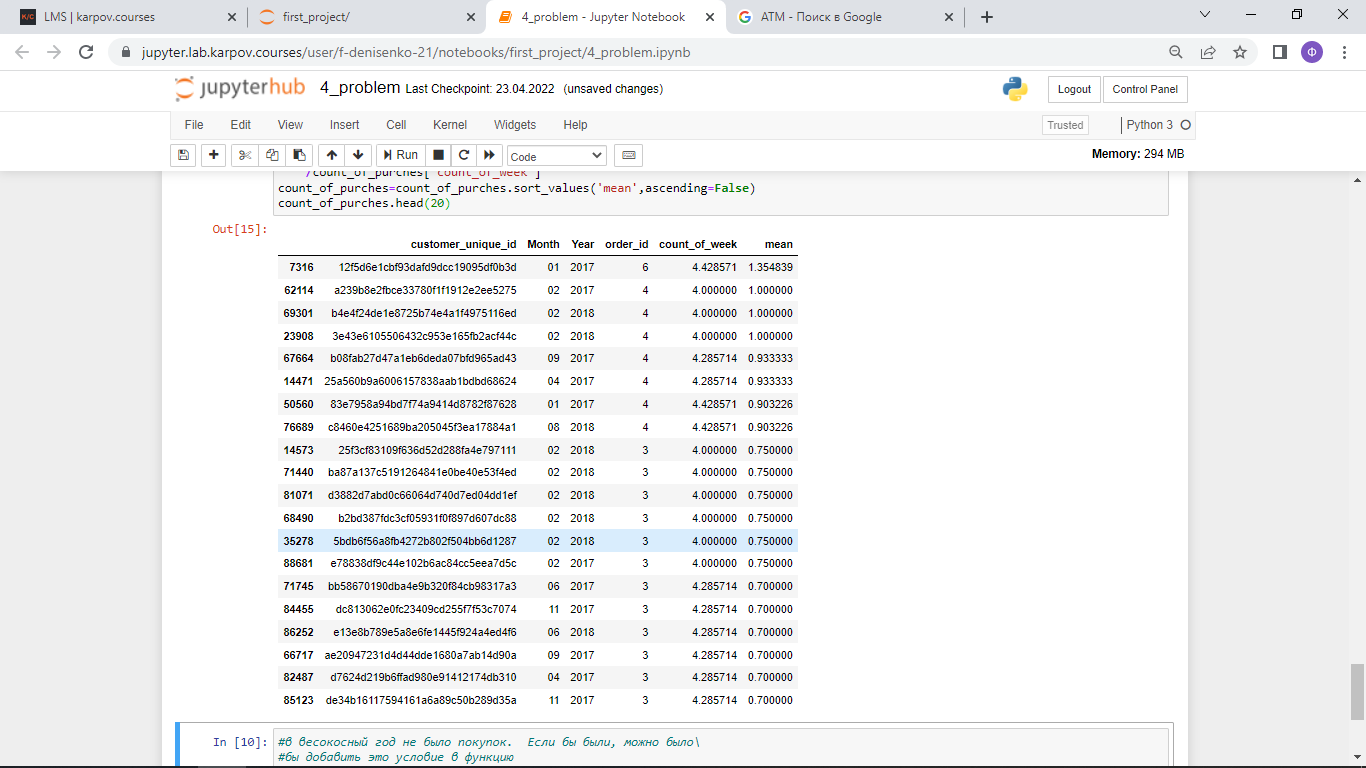
Find the number of purchases by each unique user by month and write them to the "count\_of\_purches" table.

Then we create a function that returns the number 31/7 for January and other months containing 31 days; 30/7 for April and other months with 30 days; 28/7 for February; and 29/7 for February of a leap year. These numbers form coefficients by which you can divide the number of purchases per month and get the average number of purchases per week. These coefficients show the number of weeks in a month.

Next we should create a column "count\_of\_week" in the table "count\_of\_purches" with the number of weeks in each month.

Let's include number of purchases per week by month for each user in the "mean" column. As you can see , each user can have several rows in the table, because we are looking for the average value per week in every month. It is lines for several months in which there were purchase.

Table 3 First 20 values of "Count\_of\_purches\_of\_purches" in the user table with the average number of purchases for each month



**5 task**

“Use pandas conduct a cohort analysis of users. In the period from January to December, identify the cohort with the highest retention for the 3rd month”

Firstly, as in the third task, we get the dataframe "only\_purchers\_purchers" of orders that are purchases.

Secondly, we start forming the "kogorts" dataframe. We create the "Date\_of\_purches" column and copy the "order\_approved\_at" column from "only\_purchers ". Change the data type to "datetime" and round it up to a month. In addition, we add a column of unique users "id\_customer", "order\_id” and "cohort" as the minimum date (month and year of the first purchase by a unique customer).

Next, we find the number of unique customers who made the first purchase for each month and put it into the "kogorts\_new" dataframe.

Then we create a "period\_number" column to indicate the duration between the first purchase and the next. At the same time, in the data type of this column, we use the “datetime” data type, the month format, and rounding to an integer.

Next, we create a pivot-table, where "cohort" is the index, "period\_number" is the columns, and "user\_count" is the values. We get the percentage of the users number, who continues to buy from the total number of users in the cohort. We put it in the table "retention\_matrix".

Let's create a visualization of the received data. Let's display all cohorts for 3 months and find out which has the highest percentage of users who continue to buy products.

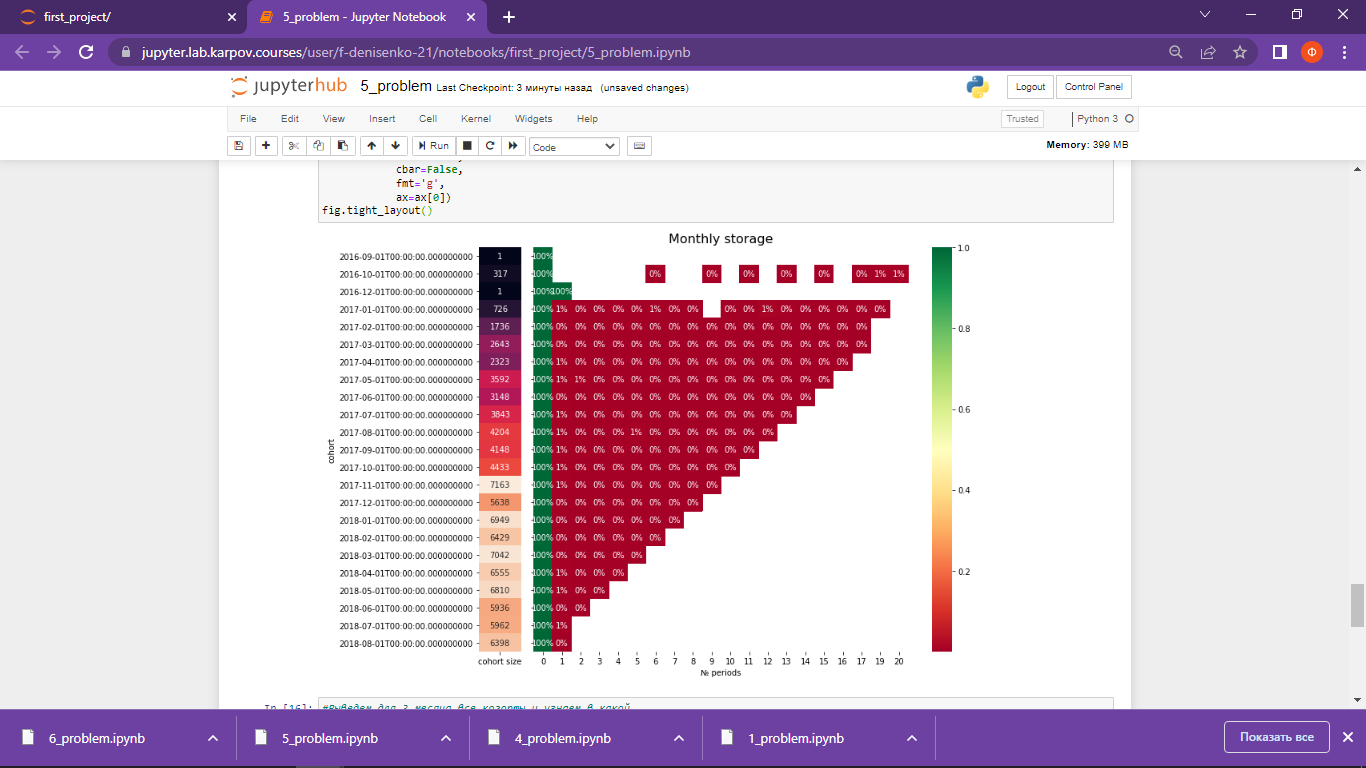
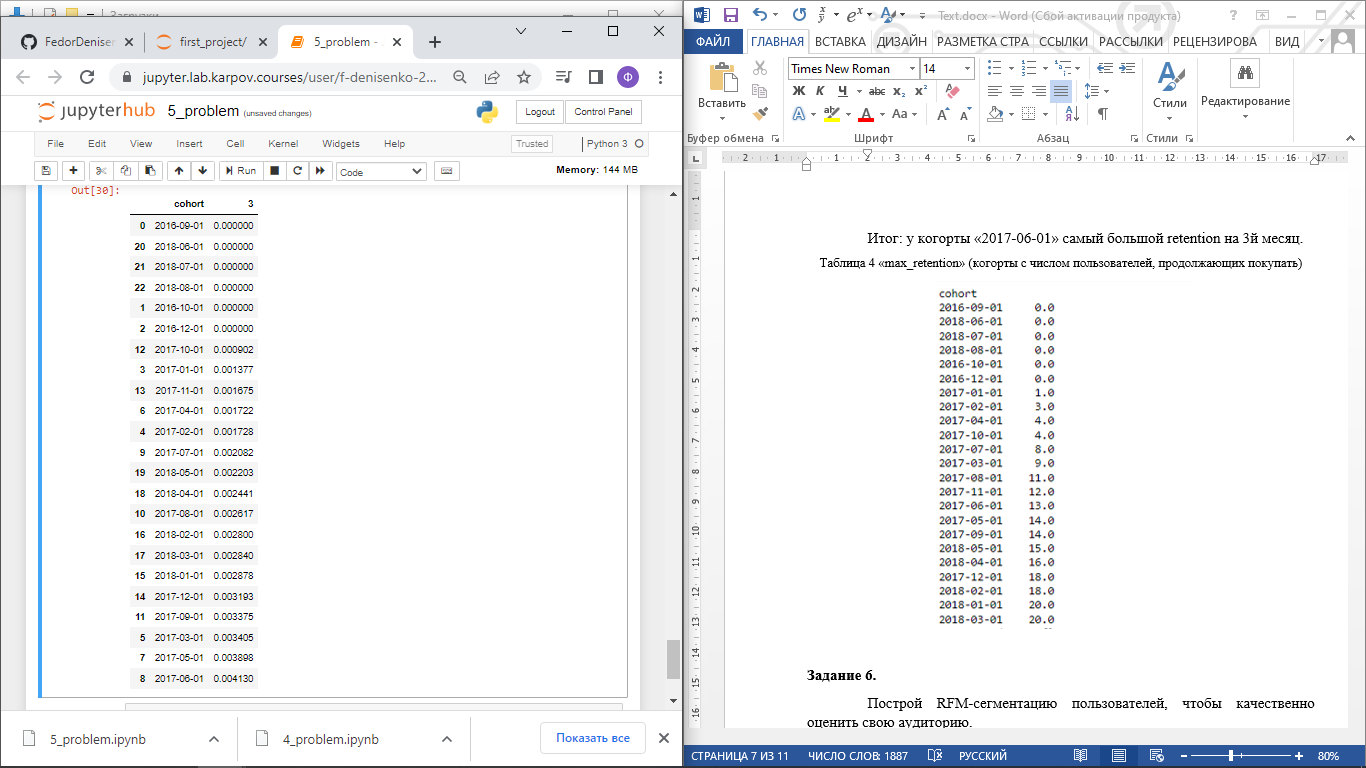


Figure 2 Retention Rate

To sum up, the cohorts "2017-06-01" have the largest retention for the 3rd month.

Table 4 "max\_retention" (cohorts with the number of users who continues to buy)



**Task 6**

“Build RFM users segmentation to estimate the audience”.

First of all, as in the third task, we get the dataframe "only\_purchers\_purchers" of orders that are purchases. We create a dataframe "sum", in which we count the amount of goods in each order and return two columns "order\_id" and "Total\_sum".

Also we delete duplicates. We merge table "sum" and "only\_purchers" with the left join in "all\_data", because there is some mistake in the data, and on the left we take the table where there is less data. The mistake is described in the end.

We create the dataframe "data", the first column will has purchase dates, which has the datetime type. The remaining columns are the cost of orders, the unique user number, and the order number.

We create the variable "NOW", which will limit the range of all order dates from the top, and the variable "start\_date", which will limit this range from the bottom and will be the date of the first order among all users.

Besides, we create a "DaysSincePurche" column that shows the number of days between the purchase and the “NOW” date (the date of the last order out of all orders). So, “NOW” is a constant.

Then we find the value for Recency, which will be equal to the minimum of the values for each user from the "DaysSincePurche" column. In other words, this is the time that has passed from the last purchase to “NOW”.

Then we find the value for Frequency, which shows how often users make purchases in the interval from “start\_date” and put it with Recency in the new dataframe "RF".

Next we create the "RFM" dataframe by adding the "Total\_sum" column from the "data" dataframe to "RF".

Also we should find the quantile values for the "RFM" dataframe that divides the data into 5 parts. We choose exactly the quantiles to have an understanding of the distributing of results among all users. To do it, we create the functions "r\_score(x)" and "fm\_score(x,c)" for distribution by ranks R, F, M.. Then we apply these functions to the columns of the "RFM" table. We get the "RFM Score" column, which shows amount of points each segment scored.

Created names for segments are put to the "segt\_map" dictionary.

Table 5 Segmentation of customers

|  |  |  |
| --- | --- | --- |
| **Segment name** | **Description** | **RFM Score** |
| Premium Description | Buy very, very often and for large amounts and recently bought. | 255, 155 |
| Former premium | We bought many very expensive things, but they have not bought for a long time. | 355, 455, 555 |
| **Segment name** | **Description** | **RFM Score** |
| Loyal customers | Very, very often buy, but the goods are inexpensive. Time from last purchase is not so important. | -51, -52, -53, -54 |
| New\_econom | They bought once, but very recently, and they need to be attracted by advertising. | 111, 112 |
| Disposable | One-time customers bought one thing a longtime ago and have never come back. | 511, 512, 513, 514, 515 |
| New middle | They bought one product that was medium-priced and even expensive not so long ago, so advertising products that are more expensive can attract them. | 211, 113, 114, 115, 212, 213, 214, 215 |
| Need to return | We bought them a long time ago, but there is a chance to return them. They bought both expensive things and cheap ones. | 411, 412, 413, 414, 415, 311,312, 313, 314, 315 |

We create a visualization of the received data in the "RFM" table.

We get bar charts. In the top left corner the Recency distribution is, In the top lright corner the Frequency distribution is. And in the bottom the Monetary value distribution is. We can see the majority of purchases were made only once by one customer and were never made again. There are very few potential loyal users. There are more new users who have recently made their first purchase, and it needs to focus on them and attract them.

Description of segments.

The Premium RFM segment 255, 155 (recency=2, frequency=5, monetary=5 and recency=1, frequency=5, monetary=5) has metric limits of recency from 0 to 181 days, frequency from 0.018 to 0.1616 orders per week, and monetary from 3.25 to 119.39 units per week.

Former premium RFM segment 355, 455, 555 (recency=3, frequency=5, monetary=5, and recency=4, frequency=5, monetary=5, and recency=5, frequency=5, monetary=5) has metric limits of recency range from 181 to 718 days, frequency from 0.018 to 0.1616 orders per week, monetary from 3.25 to 119.39 units per week.

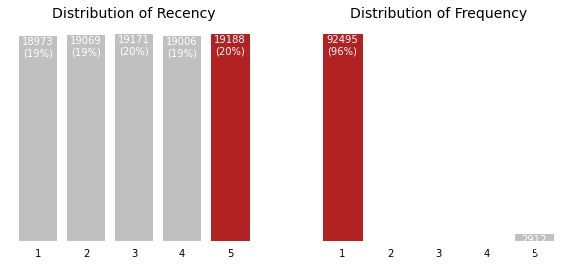
Loyal customers RFM segment -51, -52, -53, -54 (recency from 1 to 5, frequency=5, monetary from 1 to 4) has metric limits of recency from 0 to 718 days, frequency =0.16 orders per week, monetary from 0 to 3.25 units per week.

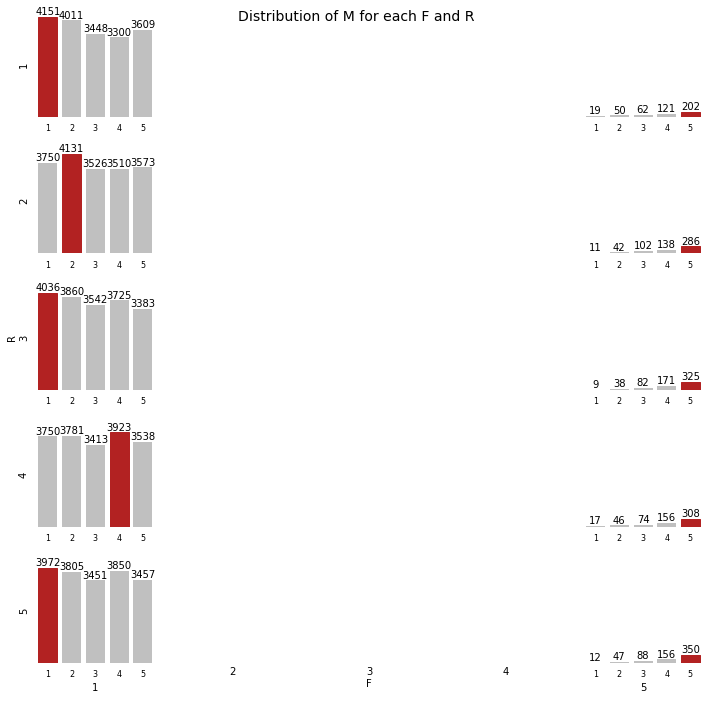
New\_econom RFM segment 111, 112 (recency=1, frequency=1, monetary from 1 to 2) has metric limits of recency from 0 to 97 days, frequency=0.018018 orders per week, monetary from 0 to 2.7 units per week.

Disposable RFM segment 511, 512, 513, 514, 515 (recency=5, frequency=1, monetary from 1 to 5) has metric limits of recency from 388 to 718 days, frequency=0.018 orders per week, monetary from 0 to 119.39 units per week.

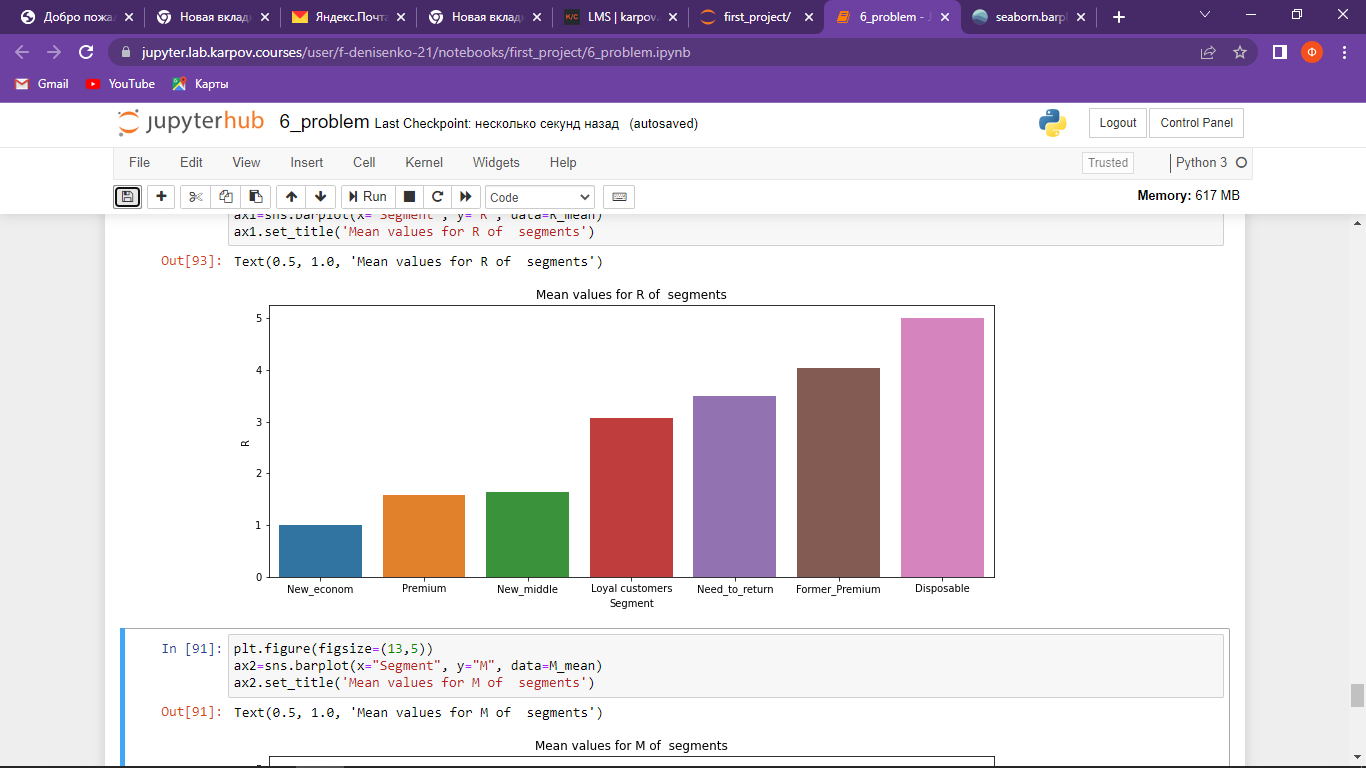
New\_middle RFM segment 211, 113, 114, 115, 212, 213, 214, 215 (recency from 1 to 2, frequency=1, monetary from 1 to 5) has metric limits of recency from 0 to 181 days, frequency=0.018 orders per week, monetary from 0 to 119.39 units per week a week.

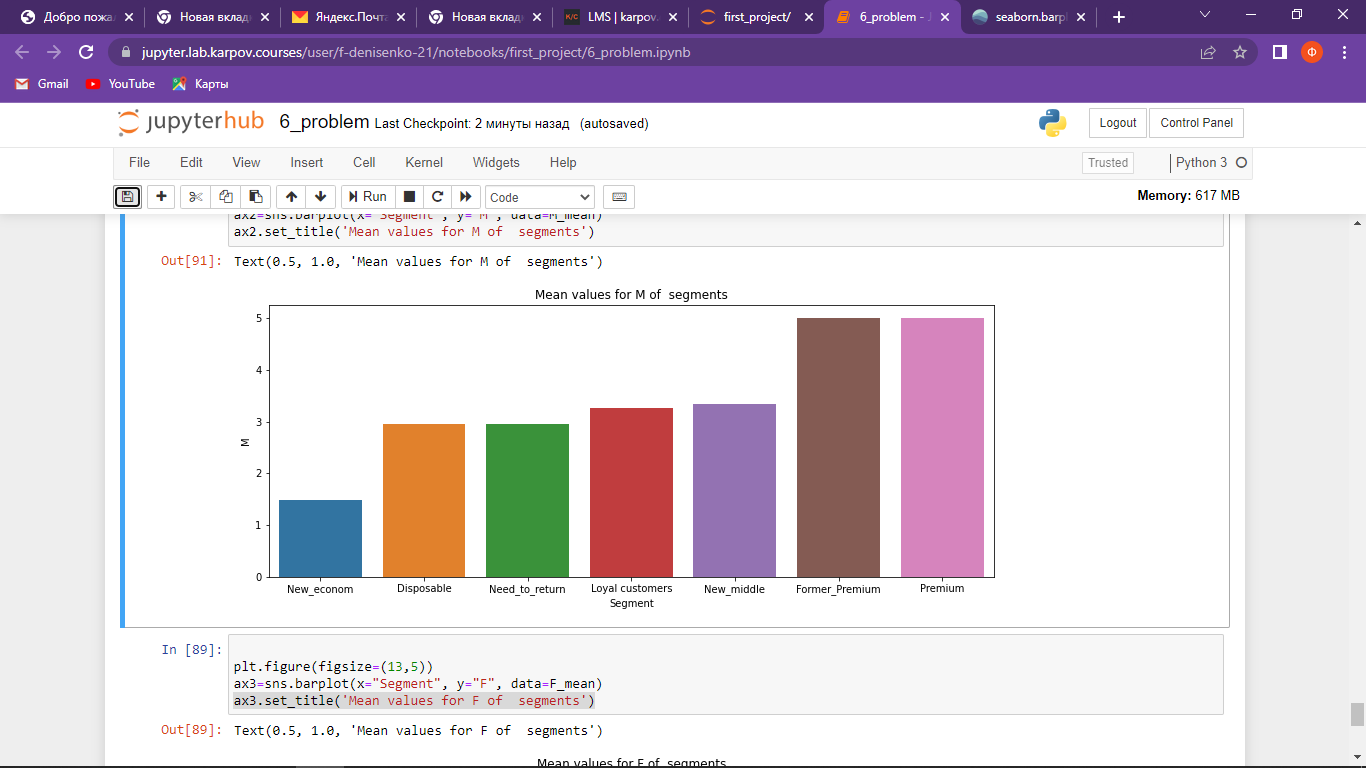
Need\_to\_return RFM segment 411, 412, 413, 414, 415, 311,312, 313, 314, 315 (recency from 3 to 4, frequency=1, monetary from 1 to 5) has metric limits of recency from 181 to 388 days, frequency=0.018 orders per week, monetary from 0 to 119.39 units per week.

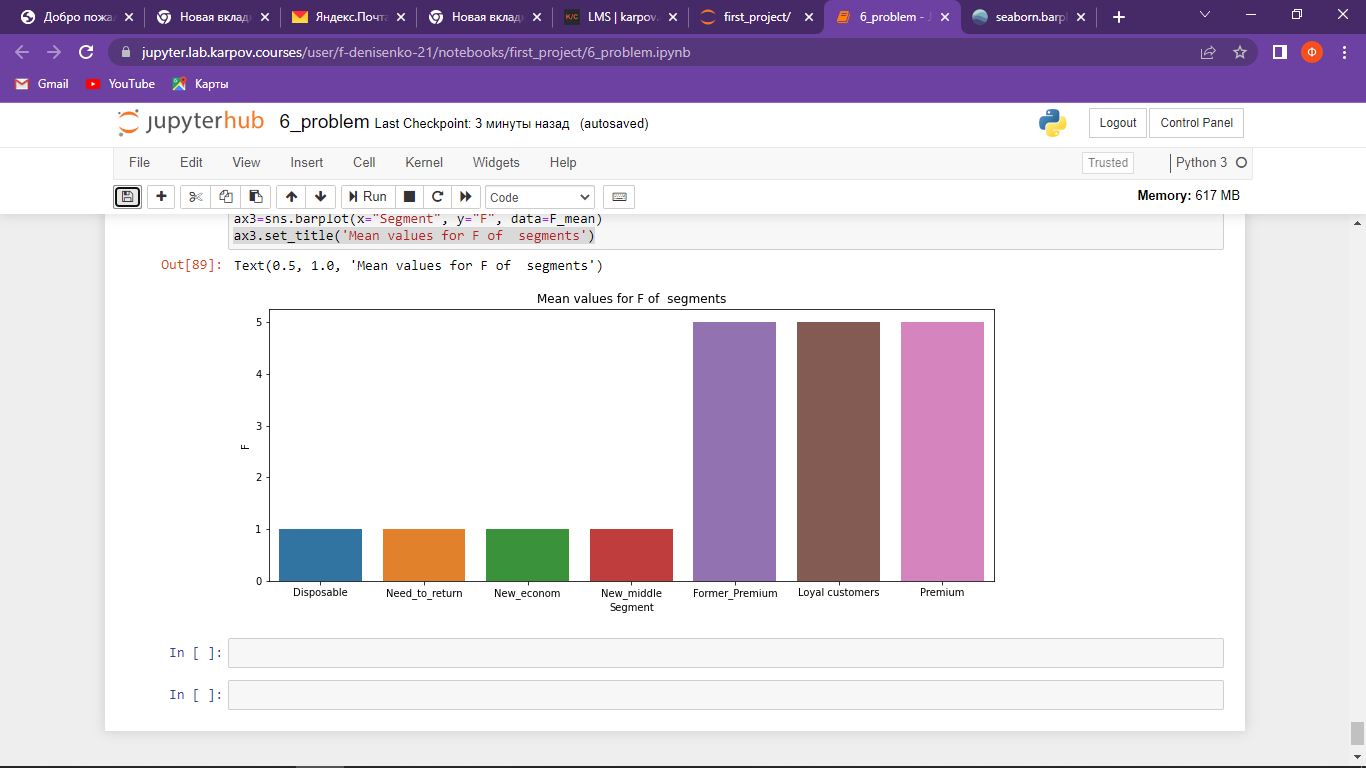




Below is the mean R, F and M values for every segment.







**Errors in the data**

The data analysis revealed the following anomalie

1). In the table "olist\_order\_items\_dataset.csv", the number of unique orders “order\_id” is 98666. In the table "olist\_orders\_dtaset.csv", the number of unique orders “order\_id” is 99441. These two numbers must match, because it is the same, unique order numbers. This means that there is no data in the table "olist\_order\_items\_dataset.csv".