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.

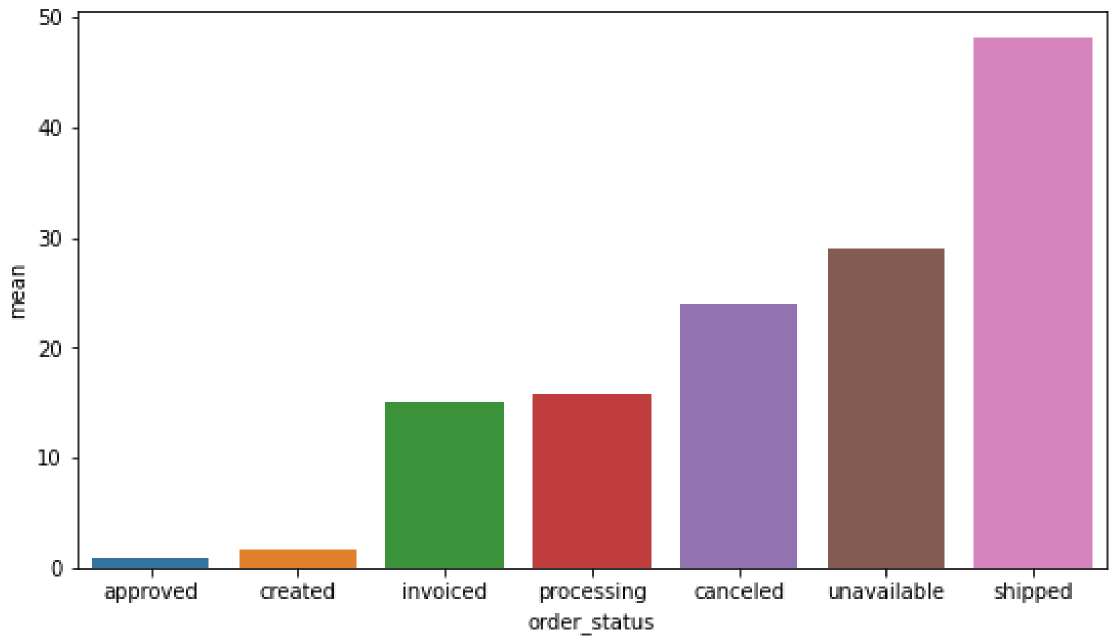


Figure 1 Bar chart of the average number of orders per month depending on the reason.

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

	product_id	count_of_day	order_approved_at
28550	aca2eb7d00ea1a7b8ebd4e68314663af	119	Tuesday
14072	53b36df67ebb7c41585e8d54d6772e08	105	Tuesday
11159	422879e10f46682990de24d770e7f83d	89	Tuesday
25493	99a4788cb24856965c36a24e339b6058	82	Tuesday
9148	368c6c730842d78016ad823897a372db	80	Thursday
9506	389d119b48cf3043d311335e499d9c6b	75	Tuesday
14039	53759a2ecddad2bb87a079a1f1519f73	73	Tuesday
34965	d1c427060a0f73f6b889a5c7c61f2ac4	63	Tuesday
10409	3dd2a17168ec895c781a9191c1e95ad7	58	Wednesday
3639	154e7e31ebfa092203795c972e5804a6	56	Tuesday
7359	2b4609f8948be18874494203496bc318	52	Wednesday
27508	a62e25e09e05e6faf31d90c6ec1aa3d1	49	Wednesday
20767	7c1bd920dbdf22470b68bde975dd3ccf	46	Tuesday
4436	19c91ef95d509ea33eda93495c4d3481	42	Tuesday

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

	customer_unique_id	Month	Year	order_id	count_of_week	mean
7316	12f5d6e1cbf93dafd9dcc19095df0b3d	01	2017	6	4.428571	1.354839
62114	a239b8e2fbce33780f1f1912e2ee5275	02	2017	4	4.000000	1.000000
69301	b4e4f24de1e8725b74e4a1f4975116ed	02	2018	4	4.000000	1.000000
23908	3e43e6105506432c953e165fb2acf44c	02	2018	4	4.000000	1.000000
67664	b08fab27d47a1eb6deda07bfd965ad43	09	2017	4	4.285714	0.933333
14471	25a560b9a6006157838aab1bdbd68624	04	2017	4	4.285714	0.933333
50560	83e7958a94bd7f74a9414d8782f87628	01	2017	4	4.428571	0.903226
76689	c8460e4251689ba205045f3ea17884a1	08	2018	4	4.428571	0.903226
14573	25f3cf83109f636d52d288fa4e797111	02	2018	3	4.000000	0.750000
71440	ba87a137c5191264841e0be40e53f4ed	02	2018	3	4.000000	0.750000
81071	d3882d7abd0c66064d740d7ed04dd1ef	02	2018	3	4.000000	0.750000
68490	b2bd387fdc3cf05931f0f897d607dc88	02	2018	3	4.000000	0.750000
35278	5bdb6f56a8fb4272b802f504bb6d1287	02	2018	3	4.000000	0.750000
88681	e78838df9c44e102b6ac84cc5eea7d5c	02	2017	3	4.000000	0.750000
71745	bb58670190dba4e9b320f84cb98317a3	06	2017	3	4.285714	0.700000
84455	dc813062e0fc23409cd255f7f53c7074	11	2017	3	4.285714	0.700000
86252	e13e8b789e5a8e6fe1445f924a4ed4f6	06	2018	3	4.285714	0.700000
66717	ae20947231d4d44dde1680a7ab14d90a	09	2017	3	4.285714	0.700000
82487	d7624d219b6ffad980e91412174db310	04	2017	3	4.285714	0.700000
85123	de34b16117594161a6a89c50b289d35a	11	2017	3	4.285714	0.700000

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.

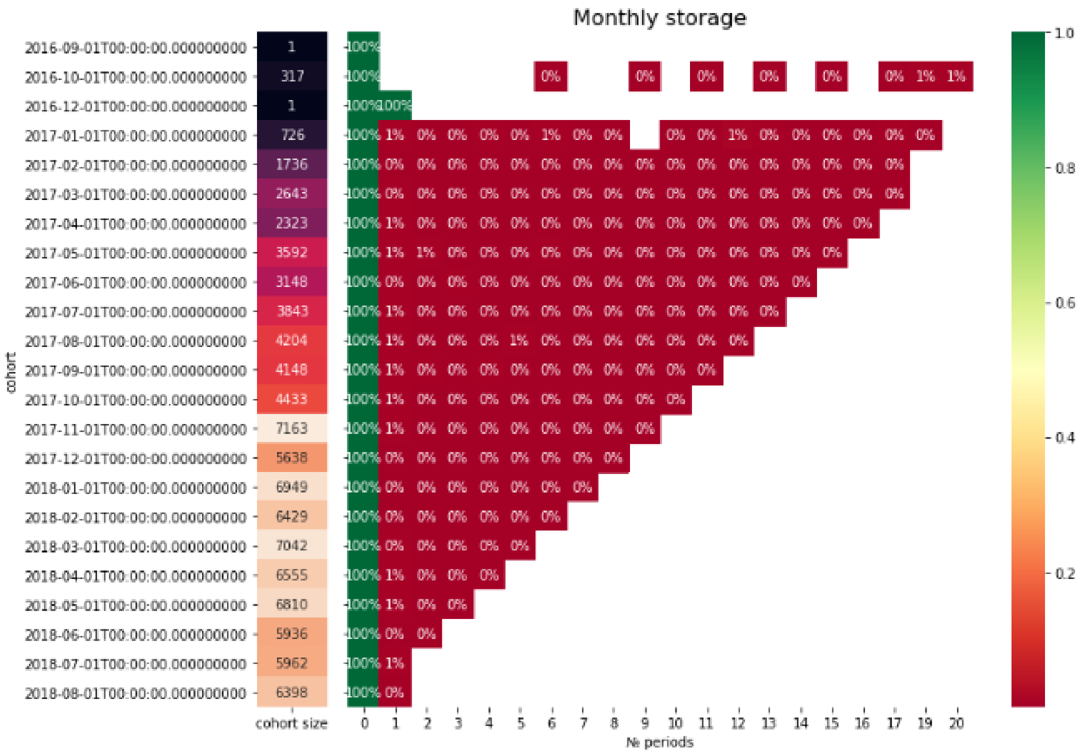


Figure 2 Retention Rate

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)

	cohort	3
0	2016-09-01	0.000000
20	2018-06-01	0.000000
21	2018-07-01	0.000000
22	2018-08-01	0.000000
1	2016-10-01	0.000000
2	2016-12-01	0.000000
12	2017-10-01	0.000902
3	2017-01-01	0.001377
13	2017-11-01	0.001675
6	2017-04-01	0.001722
4	2017-02-01	0.001728
9	2017-07-01	0.002082
19	2018-05-01	0.002203
18	2018-04-01	0.002441
10	2017-08-01	0.002617
16	2018-02-01	0.002800
17	2018-03-01	0.002840
15	2018-01-01	0.002878
14	2017-12-01	0.003193
11	2017-09-01	0.003375
5	2017-03-01	0.003405
7	2017-05-01	0.003898
8	2017-06-01	0.004130

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.

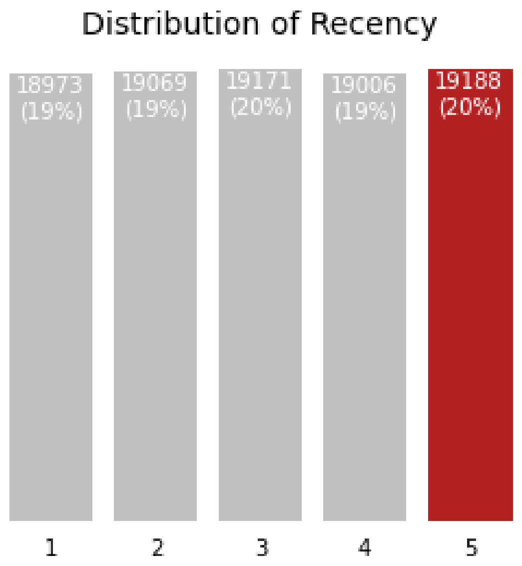
Loyal customers RFM segment -51, -52, -53, -54 (recency from 1 to 5, frequency=5, monetary from 1 to 4) has metric limits of recency from 0 to 718 days, frequency =0.16 orders per week, monetary from 0 to 3.25 units per week.

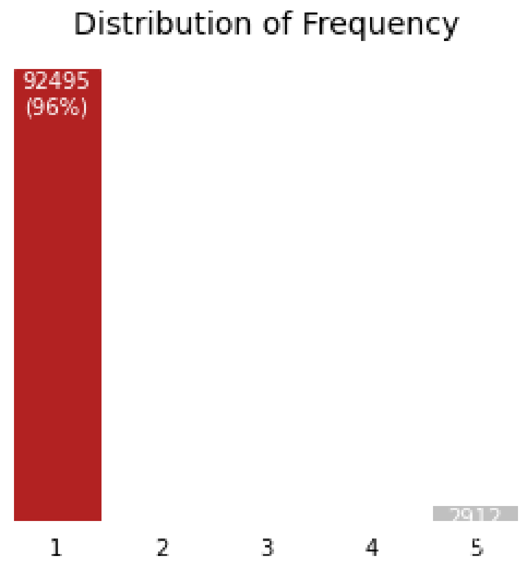
New_econom RFM segment 111, 112 (recency=1, frequency=1, monetary from 1 to 2) has metric limits of recency from 0 to 97 days, frequency=0.018018 orders per week, monetary from 0 to 2.7 units per week.

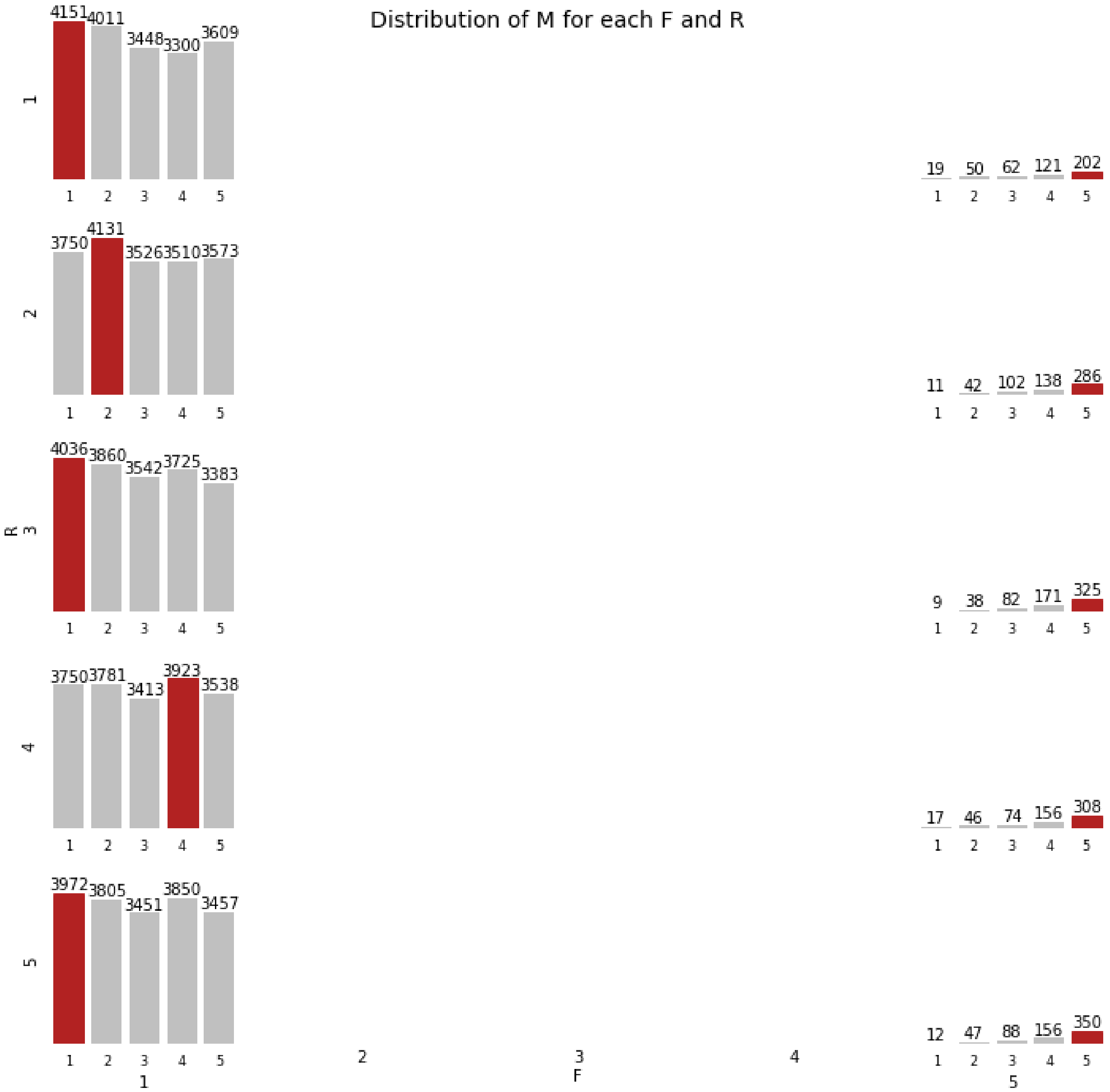
Disposable RFM segment 511, 512, 513, 514, 515 (recency=5, frequency=1, monetary from 1 to 5) has metric limits of recency from 388 to 718 days, frequency=0.018 orders per week, monetary from 0 to 119.39 units per week.

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.

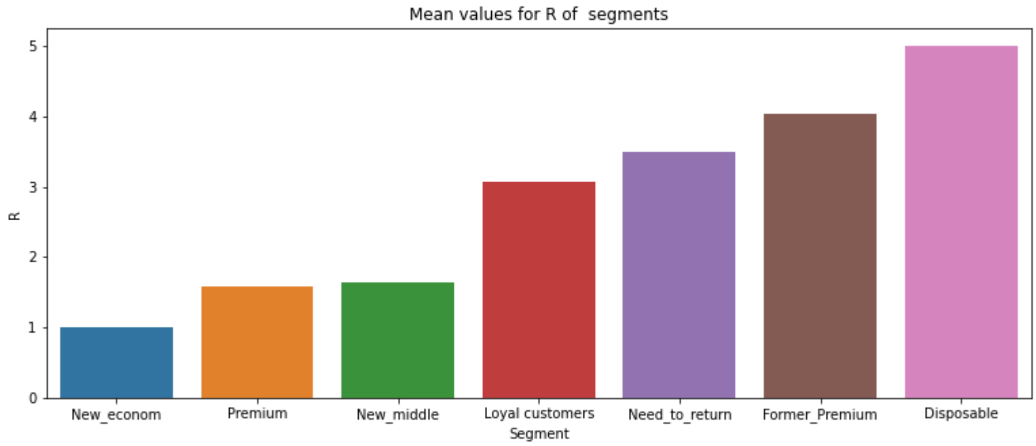
Need_to_return RFM segment 411, 412, 413, 414, 415, 311,312, 313, 314, 315 (recency from 3 to 4, frequency=1, monetary from 1 to 5) has metric limits of recency from 181 to 388 days, frequency=0.018 orders per week, monetary from 0 to 119.39 units per week.

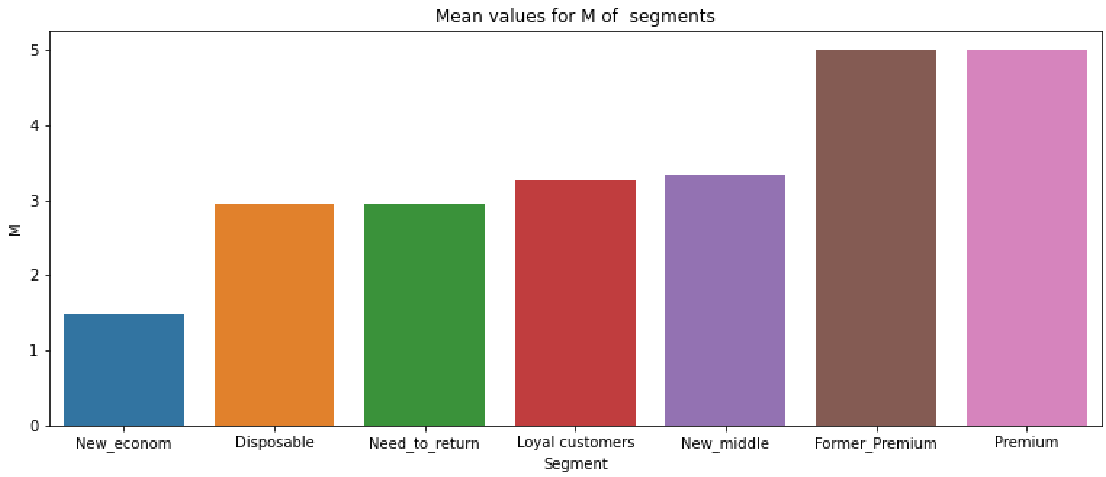

Distribution of Recency
18973 (19%)
19069 (19%)
19171 (20%)
19006 (19%)
19188 (20%)
1
2
3
4
5

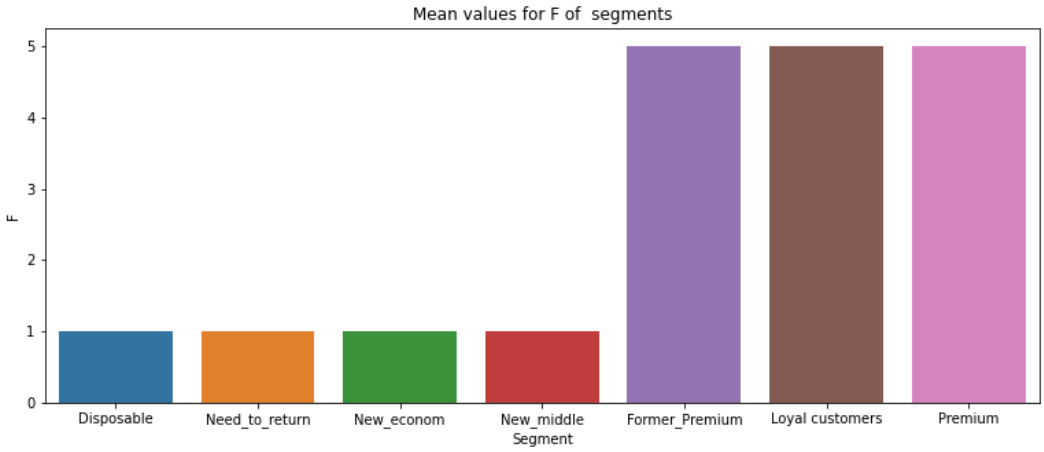

Distribution of Frequency
92495 (96%)
2912
1
2
3
4
5


Distribution of M for each F and R
R
F
1: 4151 4011 3448 3300 3609 | 19 50 62 121 202
2: 3750 4131 3526 3510 3573 | 11 42 102 138 286
3: 4036 3860 3542 3725 3383 | 9 38 82 171 325
4: 3750 3781 3413 3923 3538 | 17 46 74 156 308
5: 3972 3805 3451 3850 3457 | 12 47 88 156 350
1 2 3 4 5
1
2
3
4
5

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".