Материалы

- Ссылка на презентацию
- Ссылка на дашборд

Описание проекта

Ритейл — Анализ программы лояльности

Менеджер магазина строительных материалов "Строили, строили и наконец построили", отвечающий за программу лояльности клиентов, хочет оценить её эффективность.

Цель исследования — необходимо проанализировать программу лояльности магазина.

Задачи исследования:

Провести исследовательский анализ данных;
Провести анализ программы лояльности;
Сформулировать и проверить статистические гипотезы.

Ход исследования:
Шаг 1. Загрузка данных и предобработка данных
Шаг 2. Анализ данных
Шаг 3. Анализ программы лояльности
Шаг 4. Проверка гипотез
Выводы и рекомендации

Описание данных

Файл retail_dataset.csv:

- `purchaseId` — id чека;
- `item_ID` — id товара;
- `purchasedate` — дата покупки;
- `Quantity` — количество товара;
- `CustomerID` — id покупателя;
- `ShopID` — id магазина;
- `loyalty_program` — участвует ли покупатель в программе лояльности;

Файл product_codes.csv:

- `productID` — id товара;
- `price_per_one` — стоимость одной единицы товара;

Шаг 1. Загрузка данных и предобработка данных

Загрузка данных и изучение общей информации

In [1]:
```
#Импорт библиотек
import pandas as pd
import numpy as np
import datetime as dt
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
from scipy import stats as st
import statsmodels.api as sm
warnings.filterwarnings('ignore')
import math
import plotly.express as px
from plotly import graph_objects as go

import requests
from urllib.parse import urlencode
```

In [2]:
```
#Загрузка данных файл product_codes.csv
try:
    # используем api
    base_url = 'https://cloud-api.yandex.net/v1/disk/public/resources/download?'
    public_key = 'https://disk.yandex.ru/d/KkyaJb76T2hFdQ'

    # получаем url
    final_url = base_url + urlencode(dict(public_key=public_key))
    response = requests.get(final_url)
    download_url = response.json()['href']

    # загружаем файл в df
    download_response = requests.get(download_url)
    product_codes = pd.read_csv(download_url)
    print('Данные загружены')
except:
        try:
            product_codes = pd.read_csv('/datasets/product_codes.csv')
            print('Данные загружены из локального источника')
        except:
            print('Данные не загружены')
```

```
Данные загружены
```

In [3]:
```
#Загрузка данных retail_dataset.csv
try:
    # используем api
    base_url = 'https://cloud-api.yandex.net/v1/disk/public/resources/download?'
    public_key = 'https://disk.yandex.ru/d/q0niUM12x0lsAg'

    # получаем url
    final_url = base_url + urlencode(dict(public_key=public_key))
    response = requests.get(final_url)
    download_url = response.json()['href']

    # загружаем файл в df
    download_response = requests.get(download_url)
    retail_dataset = pd.read_csv(download_url)
    print('Данные загружены')
except:
        try:
            retail_dataset = pd.read_csv('/datasets/retail_dataset.csv')
            print('Данные загружены из локального источника')
```

```
        except:
            print('Данные не загружены')
```

```
Данные загружены
```

In [4]:

```
# Случайные пять строк датафрейма
product_codes.sample(5)
```

Out[4]:

	productID	price_per_one
7651	22364	5.79
1575	21109	13.57
171	22960	3.75
678	22429	4.25
6101	47594A	1.66

In [5]:

```
#информация о датафрейме
product_codes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9969 entries, 0 to 9968
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   productID      9969 non-null   object 
 1   price_per_one  9969 non-null   float64
dtypes: float64(1), object(1)
memory usage: 155.9+ KB
```

In [6]:

```
#Значения столбца price_per_one
product_codes['price_per_one'].describe().to_frame()
```

Out[6]:

	price_per_one
count	9969.000000
mean	19.503697
std	330.880754
min	0.000000
25%	1.250000
50%	2.550000
75%	5.510000
max	16888.020000

In [7]:

```
# Случайные пять строк датафрейма
retail_dataset.head(5)
```

Out[7]:

	purchaseid	item_ID	Quantity	purchasedate	CustomerID	ShopID	loyalty_program
0	538280	21873	11	2016-12-10 12:50:00	18427.0	Shop 0	0.0
1	538862	22195	0	2016-12-14 14:11:00	22389.0	Shop 0	1.0
2	538855	21239	7	2016-12-14 13:50:00	22182.0	Shop 0	1.0

	purchaseid	item_ID	Quantity	purchasedate	CustomerID	ShopID	loyalty_program
3	543543	22271	0	2017-02-09 15:33:00	23522.0	Shop 0	1.0
4	543812	79321	0	2017-02-13 14:40:00	23151.0	Shop 0	1.0

In [8]:
```
#информация о датафрейме
retail_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105335 entries, 0 to 105334
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   purchaseid       105335 non-null  object 
 1   item_ID          105335 non-null  object 
 2   Quantity         105335 non-null  int64  
 3   purchasedate     105335 non-null  object 
 4   CustomerID       69125 non-null   float64
 5   ShopID           105335 non-null  object 
 6   loyalty_program  105335 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 5.6+ MB
```

In [9]:
```
#Значения столбца Quantity
retail_dataset['Quantity'].describe().to_frame()
```

Out[9]:

	Quantity
count	105335.000000
mean	7.821218
std	327.946695
min	-74216.000000
25%	0.000000
50%	2.000000
75%	7.000000
max	74214.000000

In [10]:
```
#Значения столбца purchasedate
retail_dataset['purchasedate'].describe().to_frame()
```

Out[10]:

	purchasedate
count	105335
unique	4430
top	2016-12-06 16:57:00
freq	675

In [11]:
```
#Значения столбца CustomerID
retail_dataset['CustomerID'].describe().to_frame()
```

Out[11]:

	CustomerID
count	69125.000000
mean	21019.302047
std	1765.444679
min	18025.000000
25%	19544.000000
50%	20990.000000
75%	22659.000000
max	23962.000000

In [12]:
```
#Значения столбца loyalty_program
retail_dataset['loyalty_program'].value_counts()
```

Out[12]:
```
0.0    81493
1.0    23842
Name: loyalty_program, dtype: int64
```

Общее кол-во записей в датасете `retail_dataset` 105335. В датасете `product_codes` 9969.

Пропуски присутсвуют в столбце : `CustomerID`

Следут привести название всех столбцов к нижнему регистру.

Столбец `purchasedate` преобразовать в "datetime64". Столбцы `CustomerID` и `loyalty_program` преобразовать "int64".

Проверка наименований колонок

In [13]:
```
#Переименуем столбцы
product_codes.rename(columns={'productID': 'product_id'}, inplace=True)

retail_dataset.rename(columns={\
                               'purchaseid': 'purchase_id', 'item_ID': 'item_id',\
                               'Quantity': 'quantity', 'purchasedate': 'purchase_date'
                               'CustomerID': 'customer_id', 'ShopID': 'shop_id'\
                              }, inplace=True)
```

Заменили название столбцов на удобные

Проверка дубликатов

In [14]:
```
#Проверим на дубликаты
retail_dataset.duplicated().sum()
```

Out[14]:
```
1033
```

In [15]:
```
round(retail_dataset.duplicated().sum()/retail_dataset.shape[0], 2)
```

Out[15]:
```
0.01
```

In [16]:
```
#Удалим дубликаты
retail_dataset = retail_dataset.drop_duplicates().reset_index(drop=True)
```

In [17]:
```
#Проверим на дубликаты
retail_dataset.duplicated().sum()
```

Out[17]: 0

Дубликатов в датасете `retail_dataset` оказалось 1033, что состовляет 1%.
Удалили все полные дубликаты.

In [18]:
```
#Проверим на дубликаты
product_codes.duplicated().sum()
```

Out[18]: 0

In [19]:
```
#Вывод столбеца product_id
product_codes['product_id'].value_counts().to_frame()
```

Out[19]:

	product_id
DOT	174
M	59
S	29
POST	15
D	13
...	...
16012	1
90037C	1
84247N	1
84985A	1
21226	1

3159 rows × 1 columns

В таблице присутствуют значения item_id, которым соответствует несколько указаний с ценой- это как числовые, так и буквенные идентификаторы.

Проверим на уникальность.

In [20]:
```
#Проверка на уникальность
product_codes.groupby('product_id')['price_per_one'].nunique().reset_index()\
.query('price_per_one > 1').sort_values('price_per_one', ascending=False).head(15)
```

Out[20]:

	product_id	price_per_one
3150	DOT	174
3151	M	59

	product_id	price_per_one
3153	S	29
3152	POST	15
3139	D	13
2185	79321	11
2277	84406B	10
2024	47566	10
1007	22111	9
1008	22112	9
705	21673	9
2203	82484	9
703	21671	9
3136	AMAZONFEE	9
1248	22378	9

Для некоторых товаров есть несколько вариантов цен. Посчитаем их количество

In [21]:
```
#Количество товаров с несколько вариантов цен
product_codes.groupby('product_id')['price_per_one'].nunique().reset_index().query('pr
```

Out[21]: 2494

2494 - большое значение.

Лучше заменить на медийное значение.

In [22]:
```
#Заменим на медианное значение
product_codes = product_codes.pivot_table(index = 'product_id', values = 'price_per_or
```

Заменили на медийное значение товары с несколькими вариантами цен.

Проверка пропущенные значений

In [23]:
```
#Функция поиска пропущенных значений в датасете
def show_nan(data):
  df_nan = data.isna().sum().to_frame()
  df_nan[1] = data.isna().mean()
  df_nan.columns = ['Количество', '%']
  return df_nan.style.background_gradient('Reds').format({'%':'{:.2%}'})

show_nan(retail_dataset)
```

Out[23]:

	Количество	%
purchase_id	0	0.00%
item_id	0	0.00%
quantity	0	0.00%
purchase_date	0	0.00%

	Количество	%
customer_id	36148	34.66%
shop_id	0	0.00%
loyalty_program	0	0.00%

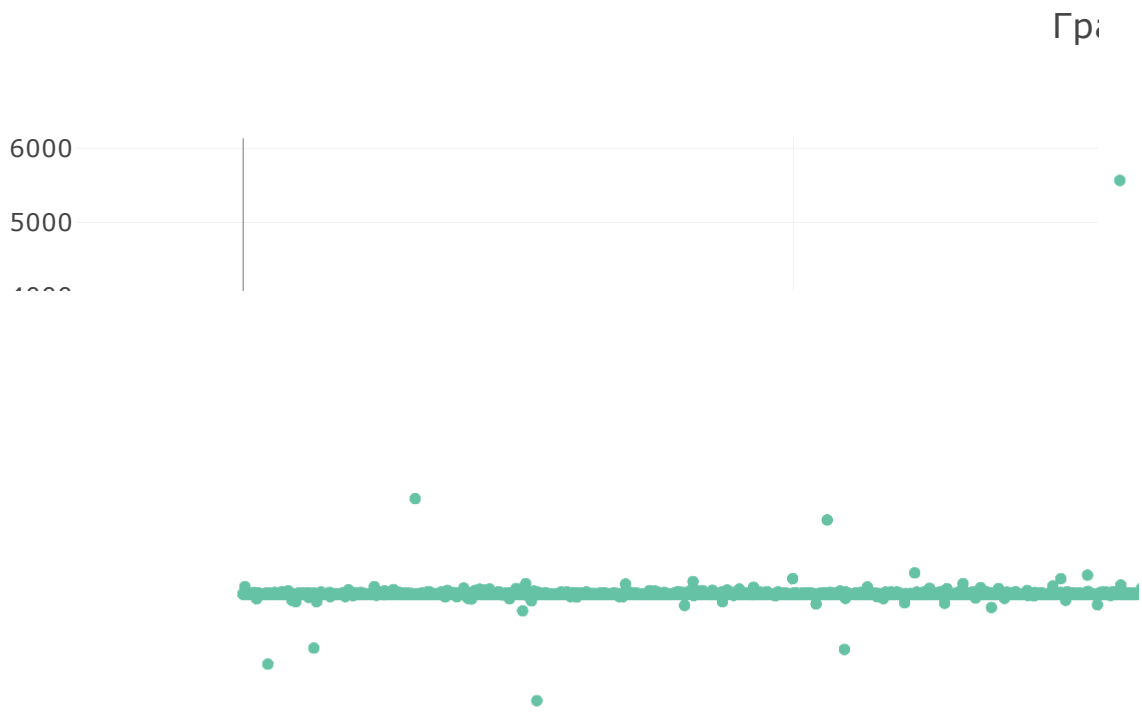
В столбце `customer_id` присутствуют 36 тысяч пропущенных значений или 34.66%.

In [24]:
```
retail_dataset[retail_dataset['customer_id'].isnull()].\
query('loyalty_program == 1')['customer_id'].count()
```

Out[24]: 0

Количество покупателей, участвующих в программе лояльности среди покупателей с пропущенным значением равно нулю. То есть эти покупатели не является участником программы лояльности.

In [25]:
```
fig = px.scatter(retail_dataset[retail_dataset['customer_id'].isnull()],
             y='quantity',
             title='График количества товаров у покупателей с пропущенным значением',
             template='none',
             labels={
                 'quantity':'Количество товаров',
             },
             color_discrete_sequence=px.colors.qualitative.Set2
             )

fig.update_layout(title_pad_l=300)

fig.show()
```



Количества покупок варьируются в очень большом диапазоне от 5567 до -2601. Выявить взаимосвязь между пропущенным значением и количеством покупок на данном этапе не представляется возможным.

Тогда заменим пропуски на нули.

In [26]:
```
#Замена пропусков на нули
retail_dataset = retail_dataset.fillna(0)
```

Заменили пропуски на нули.

Проверка соответствие типов

In [27]:
```
#Изменение типов данных
retail_dataset['purchase_date'] = pd.to_datetime(retail_dataset['purchase_date'], form
retail_dataset['customer_id'] = retail_dataset['customer_id'].astype('int64')
retail_dataset['loyalty_program'] = retail_dataset['loyalty_program'].astype('int64')
```

Изменили типы данных у столбца `purchase_date` на "datetime64", а у столбцов `customer_id` и `loyalty_program` на "int64"

Добавления столбцов

In [28]:
```
#создадим столбцы с годом, месяцем, неделей и датой
retail_dataset['purchase_year']  = retail_dataset['purchase_date'].dt.year
retail_dataset['purchase_month'] = retail_dataset['purchase_date'].dt.month
retail_dataset['purchase_week']  = retail_dataset['purchase_date'].dt.week
retail_dataset['purchase_day'] = retail_dataset['purchase_date'].dt.date
```

Добавили столбцы с годом, месяцем, неделей и датой

In [29]:
```
#добавим цены на товары к основному датасету
retail = retail_dataset.merge(product_codes, how='left', left_on='item_id', right_on='
```

Объединили две таблицы в одну с названием `retail`

In [30]:
```
#добавим столбец с общей суммой покупки
retail['revenue'] = retail['quantity'] * retail['price_per_one']
```

Добавили столбц с общей суммой покупки

In [31]:
```
retail.sample(5)
```

Out[31]:

	purchase_id	item_id	quantity	purchase_date	customer_id	shop_id	loyalty_program	purchas
24845	538635	85199S	5	2016-12-13 13:32:00	22982	Shop 0	1	
29077	543530	22459	0	2017-02-09	0	Shop 0	0	

	purchase_id	item_id	quantity	purchase_date	customer_id	shop_id	loyalty_program	purchas
				12:46:00				
10429	543015	20725	9	2017-02-02 13:46:00	19867	Shop 0	0	
71247	544998	22814	11	2017-02-25 11:56:00	20149	Shop 0	0	
38275	540458	22046	24	2017-01-07 12:28:00	18180	Shop 4	0	

Вывод

Заменили название столбцов на удобные.

Дубликатов в датасете `retail_dataset` оказалось 1033, что состовляет 1%.
Удалили все полные дубликаты.

В таблице присутствуют значения item_id, которым соответствует несколько указаний с ценой- это как числовые, так и буквенные идентификаторы.
Заменили на медийное значение товары с несколькими вариантами цен.

В столбце `customer_id` присутствуют 36 тысяч пропущенных значений или 34.66%.
Заменили пропуски на нули.

Изменили типы данных у столбца `purchase_date` на "datetime64", а у столбцов `customer_id` и `loyalty_program` на "int64".

Добавили столбцы с годом, месяцем, неделей и датой.
Добавили столбец с общей суммой покупки.

Объединил две таблицы в одну с названием `retail` .

Шаг 2. Анализ данных

Анализ выбросов данных

Посмотрим данные на выбросы, в том числе на наличие нулевых и отрицательных значений.

In [32]:
```
# Количество нулевых значений в столбце 'quantity'
retail.query('quantity == 0').shape[0]
```

Out[32]: 32362

In [33]:
```
# Количество нулевых значений в столбце 'price_per_one'
retail.query('price_per_one == 0').shape[0]
```

Out[33]: 59

Нулевые значения присутствуют в столбцах - `quantity` , `price_per_one` .

In [34]:
```
# Количество нулевых значений в столбце 'quantity' относительно лояльности
retail.query('quantity == 0')['loyalty_program'].value_counts()
```

Out[34]:
```
0    26776
1     5586
Name: loyalty_program, dtype: int64
```

Товары с нулевым количеством появляются не только в чеках с картой лояльности.

In [35]:
```
# Количество уникальных значений
retail.query('quantity == 0').nunique()
```

Out[35]:
```
purchase_id        1739
item_id            2801
quantity              1
purchase_date      1676
customer_id         837
shop_id              17
loyalty_program       2
purchase_year         2
purchase_month        3
purchase_week        13
purchase_day         68
price_per_one       484
revenue               1
dtype: int64
```

Количество уникальных товаров с нуливым значением в чеке дастоточное большое.

In [36]:
```
# Количество нулевых значений в столбце 'price_per_one' относительно лояльности
retail.query('price_per_one == 0')['loyalty_program'].value_counts()
```

Out[36]:
```
0    59
Name: loyalty_program, dtype: int64
```

In [37]:
```
# Количество уникальных значений в столбце 'item_id'
retail.query('price_per_one == 0').nunique()
```

Out[37]:
```
purchase_id        59
item_id            57
quantity           24
purchase_date      43
customer_id         1
shop_id             1
loyalty_program     1
purchase_year       2
purchase_month      3
purchase_week       8
purchase_day       15
price_per_one       1
revenue             1
dtype: int64
```

Нулевые значения в столбце `price_per_one` присутствуют только у покупателей без карты и у не большого количества товаров.

Нулевые значения указаны для различных позиций, это не одинаковые позиции, количество их также разное, вероятнее всего это акционные товары, которые даются бонусом к покупке, однако это также может быть некий сбой программы

Рассмотрим отрицательны значения

In [38]:
```
# Количество отрицательных значений в столбце 'quantity'
retail.query('quantity < 0').shape[0]
```

Out[38]: 2076

Отрицательные значения присутствуют только в столбце `quantity` .

Предположим, что отрицательные значения это возварты.

In [39]:
```
# Отрицательные значения и буква C
retail[(retail['quantity'] < 0) & (retail['purchase_id'].str.contains('C'))]
```

Out[39]:

	purchase_id	item_id	quantity	purchase_date	customer_id	shop_id	loyalty_program	purcha
64	C539944	22776	-2	2016-12-23 11:38:00	20239	Shop 0	0	
109	C542910	20726	-2	2017-02-01 15:38:00	23190	Shop 0	1	
112	C542426	22418	-25	2017-01-28 09:32:00	19825	Shop 0	0	
253	C539726	22791	-11	2016-12-21 14:24:00	22686	Shop 0	1	
344	C544034	21878	-2	2017-02-15 11:28:00	20380	Shop 0	0	
...	...	...	...	...	...	...	...	
104132	C541650	M	-2	2017-01-20 11:44:00	0	Shop 0	0	
104143	C540246	79320	-2	2017-01-05 15:43:00	18760	Shop 0	0	
104180	C539467	22801	-2	2016-12-19 12:46:00	20723	Shop 0	0	
104217	C540847	22197	-3	2017-01-11 17:35:00	19137	Shop 0	0	
104267	C540164	21144	-13	2017-01-05 12:02:00	20590	Shop 6	0	

1862 rows × 13 columns

In [40]:
```
# Отрицательные значения без буквы C
retail[(retail['quantity'] < 0) & (~retail['purchase_id'].str.contains('C'))]
```

Out[40]:

	purchase_id	item_id	quantity	purchase_date	customer_id	shop_id	loyalty_program	purcha
468	537032	21275	-31	2016-12-03 16:50:00	0	Shop 0	0	
503	540119	22865	-61	2017-01-05 10:07:00	0	Shop 0	0	
910	540241	35957	-940	2017-01-05 15:17:00	0	Shop 0	0	

	purchase_id	item_id	quantity	purchase_date	customer_id	shop_id	loyalty_program	purcha
1784	537009	84534B	-81	2016-12-03 15:38:00	0	Shop 0	0	
1928	540010	22501	-101	2017-01-04 11:13:00	0	Shop 0	0	
...	...	...	...	...	...	...	...	
102027	542225	85096	-60	2017-01-26 13:10:00	0	Shop 0	0	
102531	540558	21258	-30	2017-01-10 10:04:00	0	Shop 0	0	
103566	541487	85118	-36	2017-01-18 13:19:00	0	Shop 0	0	
103934	540564	22617	-2601	2017-01-10 10:36:00	0	Shop 0	0	
104111	542572	85064	-2	2017-01-28 14:54:00	0	Shop 0	0	

214 rows × 13 columns

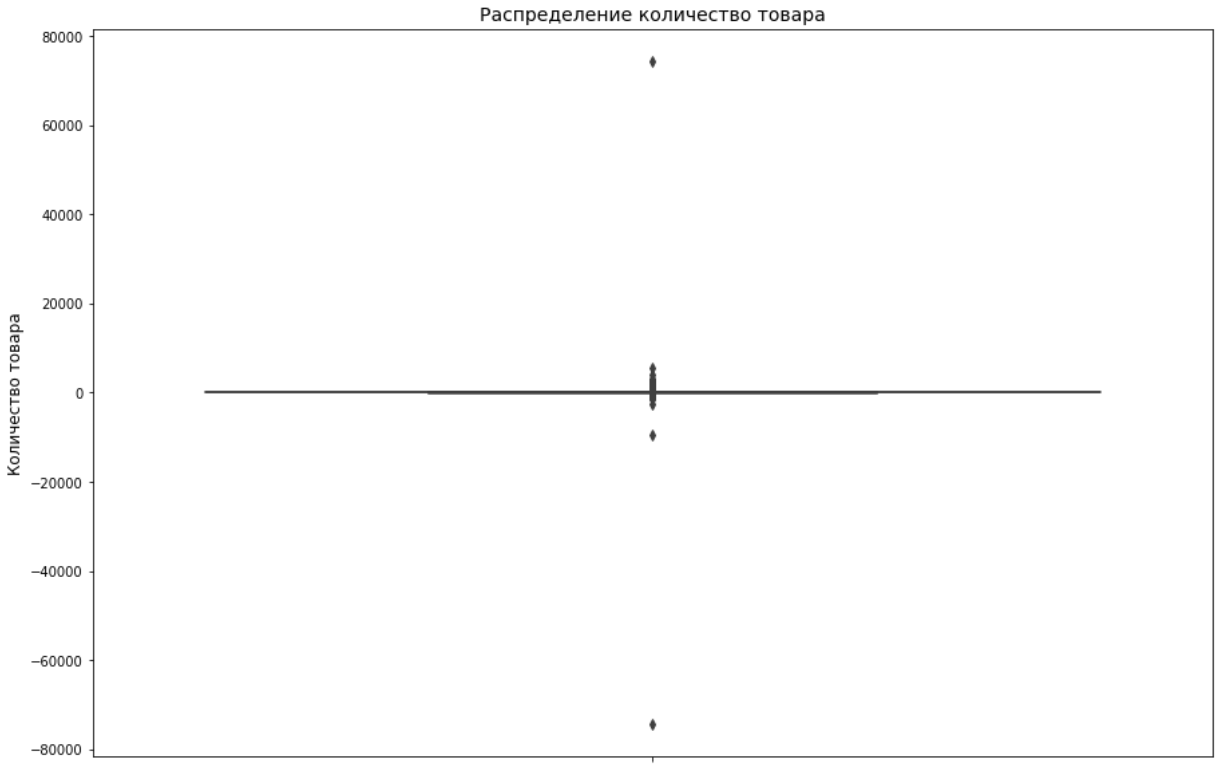
In [41]:
```
retail[(retail['quantity'] < 0) & (~retail['purchase_id'].str.contains('C') & (retail[
```

Out[41]: 214

Среды отрицательных значений столбца `quantity` чаще всего встречается дополнительное значение в столбце `purchase_id` в виде буквы `C`, что скорей всего означает возврат товара.

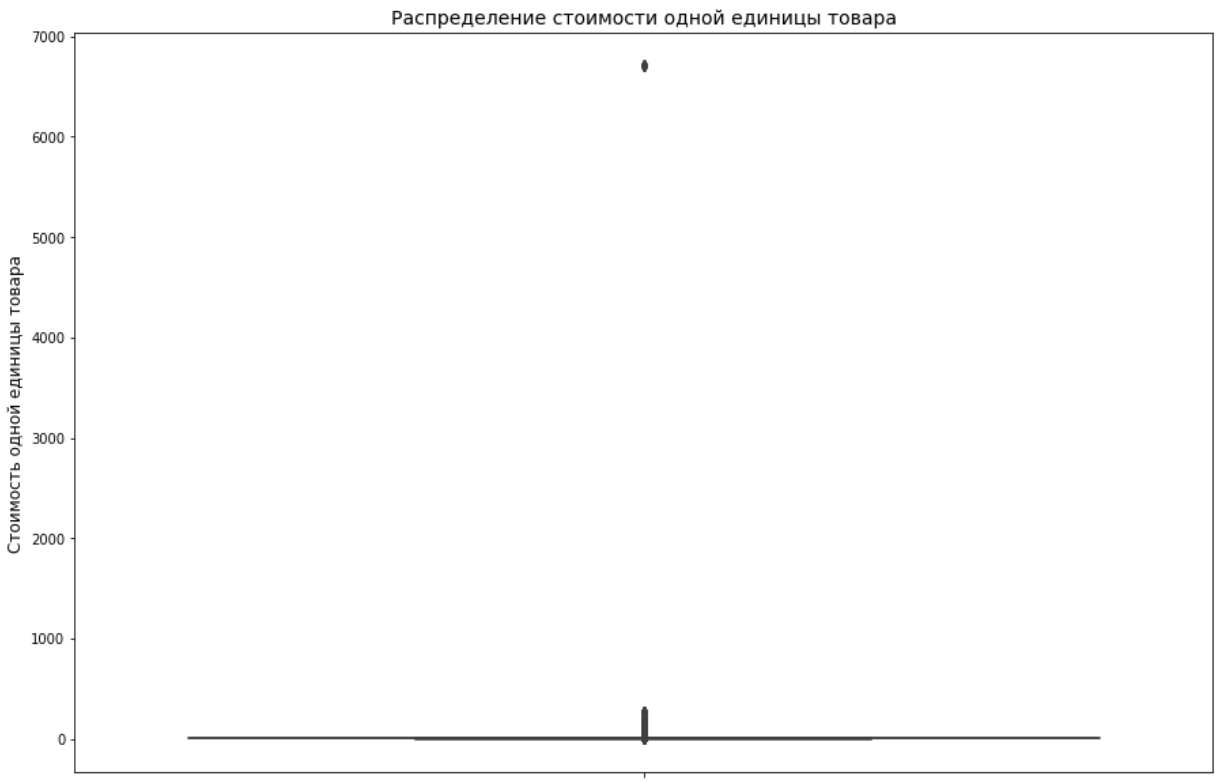
Построим графики распределения признаков.

In [42]:
```
#Построение графика
plt.figure(figsize=(15, 10))
sns.boxplot(y='quantity', data=retail)
plt.title('Распределение количество товара', fontsize=14)
plt.ylabel('Количество товара', fontsize=12)
plt.show()
```



In [43]:

```
#Построение графика
plt.figure(figsize=(15, 10))
sns.boxplot(y='price_per_one', data=retail)
plt.title('Распределение стоимости одной единицы товара', fontsize=14)
plt.ylabel('Стоимость одной единицы товара', fontsize=12)
plt.show()
```



На графике `quantity` присутствуют единичные выбросы для значений количества товаров более 20000 и -20000 шт, так как таких значений немного - мы их отбросим. Также на графике `price_per_one` присутствует незначительное количество товаорв ценой более 1000 - возможно это дорогостоящие элементы, однако при анализе

целесообразно их отбросить, так как они будут давать сильное искажение по общей сумме товара и затруднят прогнозирование.

In [44]:
```
retail.shape[0]
```

Out[44]: 104302

In [45]:
```
np.percentile(retail['quantity'], [1, 99])
```

Out[45]: array([-3., 99.])

In [46]:
```
np.percentile(retail['price_per_one'], [1, 99])
```

Out[46]: array([0.42, 19.96])

In [47]:
```
retail = retail[(retail['quantity'] < np.percentile(retail['quantity'], 99))\
                & (retail['quantity'] > np.percentile(retail['quantity'], 1))]
```

In [48]:
```
retail = retail[(retail['price_per_one'] < np.percentile(retail['quantity'], 99))]
```

In [49]:
```
retail.shape[0]
```

Out[49]: 101672

Удалили по 1 перцентилю с каждой стороны в столбце `quantity` и 1 перцентиль верхний границы в столбце `price_per_one`

После удаления число выбросов общее количество записей 101672, то есть менее чем 3% записей исчез в результате удаления выбросов

Анализ магазинов

In [50]:
```
# посмотрим на распределение наблюдений по магазинам
retail['shop_id'].value_counts().to_frame().head(10)
```

Out[50]:

	shop_id
Shop 0	94284
Shop 4	1633
Shop 1	1513
Shop 6	938
Shop 8	553
Shop 3	324
Shop 7	311
Shop 10	291
Shop 12	270

	shop_id
Shop 18	249

In [51]:
```
#Построение графика
shops = pd.DataFrame(retail['shop_id'].value_counts()).reset_index()

fig = px.bar(shops,
             x='index',
             y='shop_id',
             title='Количество наблюдений во всех магазинах, кроме Shop 0',
             labels={
                 'index': 'id магазина',
                 'shop_id': 'Количество наблюдений'
             },
             color_discrete_sequence=px.colors.qualitative.Set2,
             template='simple_white')

fig.update_layout(title_pad_l=300)

fig.show()
```

In [52]:
```
#Построение графика
fig = px.bar(shops.query('index != "Shop 0"'),
             x='index',
             y='shop_id',
             title='Количество наблюдений во всех магазинах, кроме Shop 0',
             labels={
```

```
                'index': 'id магазина',
                'shop_id': 'Количество наблюдений'
            },
            color_discrete_sequence=px.colors.qualitative.Set2,
            template='simple_white')

fig.update_layout(title_pad_l=300)

fig.show()
```

Shop 0 явно самый крупный магазин сети. Это либо интернет магазин, либо оптовый склад. В тройку лидеров входят магазины номер 4, 1 и 6. Есть магазинов, где продали менее 100 позиций.

Период вы располагаемых данных

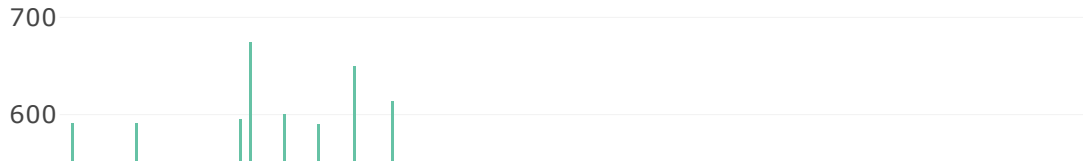
In [53]:
```
retail['purchase_date'].describe()
```

Out[53]:
```
count                  101672
unique                   3895
top       2016-12-06 16:57:00
freq                      674
first     2016-12-01 08:26:00
last      2017-02-28 17:01:00
Name: purchase_date, dtype: object
```

In [54]:
```
retail['purchase_date'].value_counts()
```

Out[54]:
```
2016-12-06 16:57:00    674
2016-12-09 14:09:00    650
2016-12-10 14:59:00    613
2016-12-07 15:28:00    600
2016-12-06 09:58:00    595
                      ...
2016-12-09 16:11:00      1
2016-12-17 13:44:00      1
2017-01-06 12:32:00      1
2016-12-02 11:56:00      1
2017-01-05 16:06:00      1
Name: purchase_date, Length: 3895, dtype: int64
```

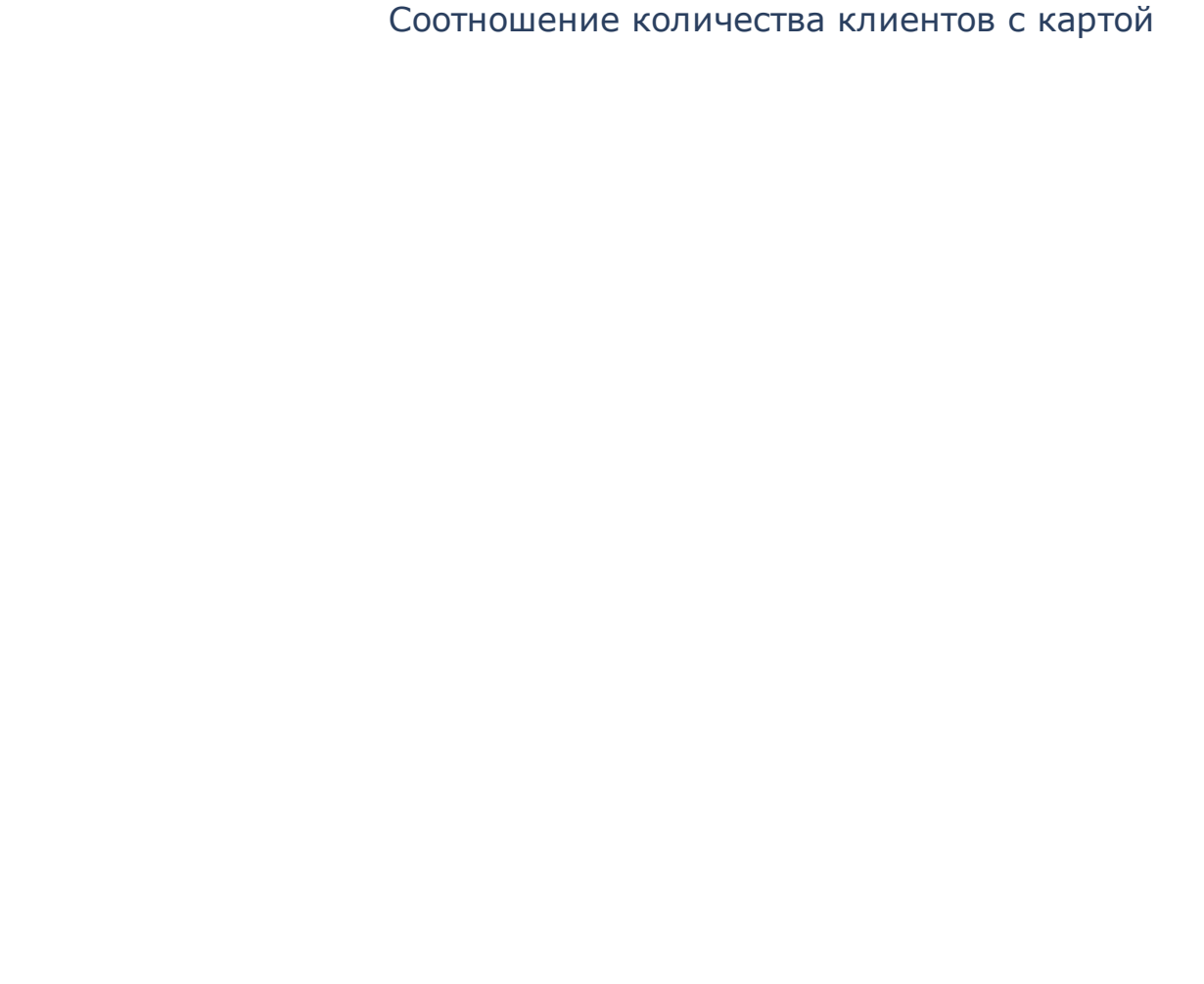
In [55]:
```
p_date = pd.DataFrame(retail['purchase_date'].value_counts()).reset_index()

fig = px.bar(p_date,
             x='index',
             y='purchase_date',
             title='Количество наблюдений за весь период',
             labels={
                 'index': 'Дата',
                 'purchase_date': 'Количество наблюдений'
             },
             color_discrete_sequence=px.colors.qualitative.Set2,
             template='none')

fig.update_layout(title_pad_l=300)

fig.show()
```



Наблюдения распределены по временной шкале примерно равномерно, делать срез данных за какой-то определенный адекватный период не требуется.
В датасете отсутсвуют данные за период с 24 декабря 2016 по 3 января 2017.

Соотношение покупателей с картой и без нее

In [56]:
```
loyalty_program = pd.DataFrame(retail['loyalty_program'].value_counts())\
    .rename(index={0: 'Без карты', 1: 'С картой'})\
    .reset_index()

fig = go.Figure(data=[
    go.Pie(labels=loyalty_program['index'],
           values=loyalty_program['loyalty_program'],
           hole=.5)
])

fig.update_layout(title='Соотношение количества клиентов с картой лояльности',
                  title_pad_l=108,
                  )

fig.show()
```



Клиентов с картой лояльности в 3 раза меньше, чем обычных покупателей.

Вывод

Были удалены выбросы, общее количество записей стало 104289.

Shop 0 явно самый крупный магазин сети. Это либо интернет магазин, либо оптовый склад. В тройку лидеров входят магазины номер 4, 1 и 6. Есть магазинов, где продали менее 100 позиций.

Наблюдения распределены по временной шкале примерно равномерно, делать срез данных за какой-то определенный адекватный период не требуется.
В датасете отсутсвуют данные за период с 24 декабря 2016 по 3 января 2017.

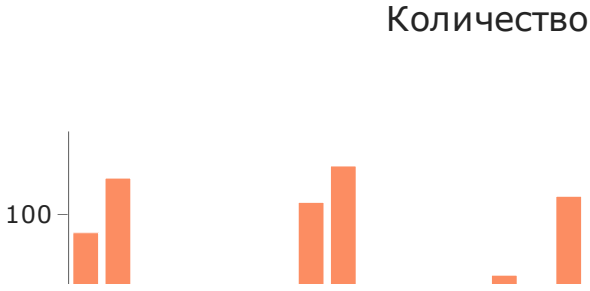
Клиентов с картой лояльности в 3 раза меньше, чем обычных покупателей.

Шаг 3. Анализ программы лояльности

Количество уникальных покупателей в день, неделю и месяц

In [57]:
```
#Изменим столбец loyalty_program на более понятный
retail['loyalty_program'] = retail['loyalty_program'].replace({0: 'нет карты лояльност
```

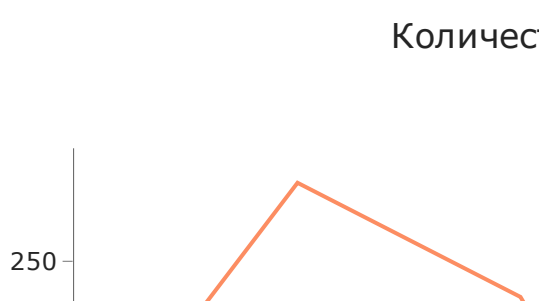
In [58]:
```
dau = retail.groupby(['purchase_day', 'loyalty_program']).agg({'customer_id': 'nunique

fig = px.bar(dau,
             x='purchase_day',
             y='customer_id',
             color = 'loyalty_program',
             title='Количество уникальных покупателей в день',
             template='simple_white',
             labels={
                 'purchase_day': 'Дата',
                 'customer_id': 'Уникальные покупатели'
             },
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(title_pad_l=150, legend=dict(title=' '))

fig.show()
```



Ежедневных уникальный покупателей без карты лояльности регулярно больше.

In [59]:

```
wau = retail.groupby(['purchase_year', 'purchase_week', 'loyalty_program']).agg({'cust

wau['week_yr'] = pd.to_datetime(wau['purchase_year'].astype(str) + ' ' + wau['purchase
                                format='%Y %U %w').astype('str')

fig = px.line(wau,
              x='week_yr',
              y='customer_id',
              color = 'loyalty_program',
              title='Количество уникальных покупателей в неделю',
              template='simple_white',
              labels={
                  'week_yr': 'Дата',
                  'customer_id': 'Уникальные покупатели'
              },
              color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(title_pad_l=150, legend=dict(title=' '))
fig.update_traces(textposition="top right")

fig.show()
```



Уникальный покупателей в неделю без карты лояльности так же больше.

In [60]:

```
mau = retail.groupby(['purchase_year', 'purchase_month', 'loyalty_program']).agg({'cus

mau['purchase_month'] = mau['purchase_month'].replace({12: 'декабрь', 1: 'январь', 2:

fig = px.bar(mau,
             x='purchase_month',
             y='customer_id',
             color = 'loyalty_program',
             text='customer_id',
             title='Количество уникальных покупателей в месяц',
             template='simple_white',
             labels={
                 'purchase_month': 'Месяц',
                 'customer_id': 'Уникальные покупатели'
             },
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(title_pad_l=150, legend=dict(title=' '))
#fig.update_traces(textposition="top right")

fig.show()
```

По месяцам количество уникальных пользователей без карты лояльности больше, чем с ней примерно в два раза.

Большинство покупателей предпочитают не участвовать в программе лояльности магазина.

Юнит-экономика

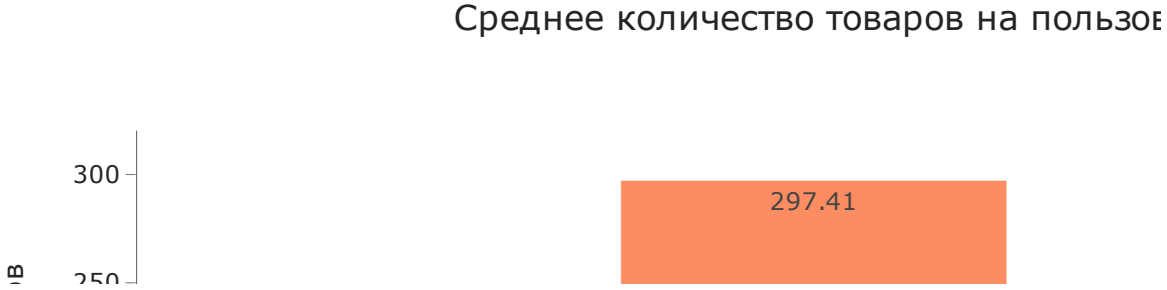
In [61]:
```
check_by_month = retail.groupby(by=['loyalty_program', 'purchase_year', 'purchase_mont
                .agg({'purchase_id': 'count', 'customer_id': 'nunique'}).reset_index()

check_by_month['purchase_month'] = check_by_month['purchase_month'].replace({12: 'дека
check_by_month.rename(columns={"purchase_id": "total_income", "customer_id": "number_c

check_by_month['income_per_customer'] = round(check_by_month['total_income'] / check_b
```

In [62]:
```
fig = px.bar(check_by_month,
             x='purchase_month',
             y='income_per_customer',
             color = 'loyalty_program',
             text='income_per_customer',
             title='Среднее количество покупок на пользователя в месяц',
             template='simple_white',
             barmode='group',
             labels={
                 'purchase_month': 'Месяц',
                 'income_per_customer': 'Среднее количество покупок'
             },
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(title_pad_l=150, legend=dict(title=' '))
#fig.update_traces(textposition="top right")

fig.show()
```

Покупатели без карты лояльности чаще совершают покупки.

In [63]:
```
quantity_by_month = retail.groupby(by=['loyalty_program', 'purchase_year', 'purchase_m
                .agg({'quantity': 'sum', 'customer_id': 'nunique'}).reset_index()

quantity_by_month['purchase_month'] = quantity_by_month['purchase_month'].replace({12:
quantity_by_month.rename(columns={"quantity": "total_income", "customer_id": "number_c

quantity_by_month['income_per_customer'] = round(quantity_by_month['total_income'] / c
```

In [64]:
```
fig = px.bar(quantity_by_month,
             x='purchase_month',
             y='income_per_customer',
             color = 'loyalty_program',
             text='income_per_customer',
             title='Среднее количество товаров на пользователя в месяц',
             template='simple_white',
             barmode='group',
             labels={
                 'purchase_month': 'Месяц',
                 'income_per_customer': 'Среднее количество товаров'
             },
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(title_pad_l=150, legend=dict(title=' '))
#fig.update_traces(textposition="top right")

fig.show()
```

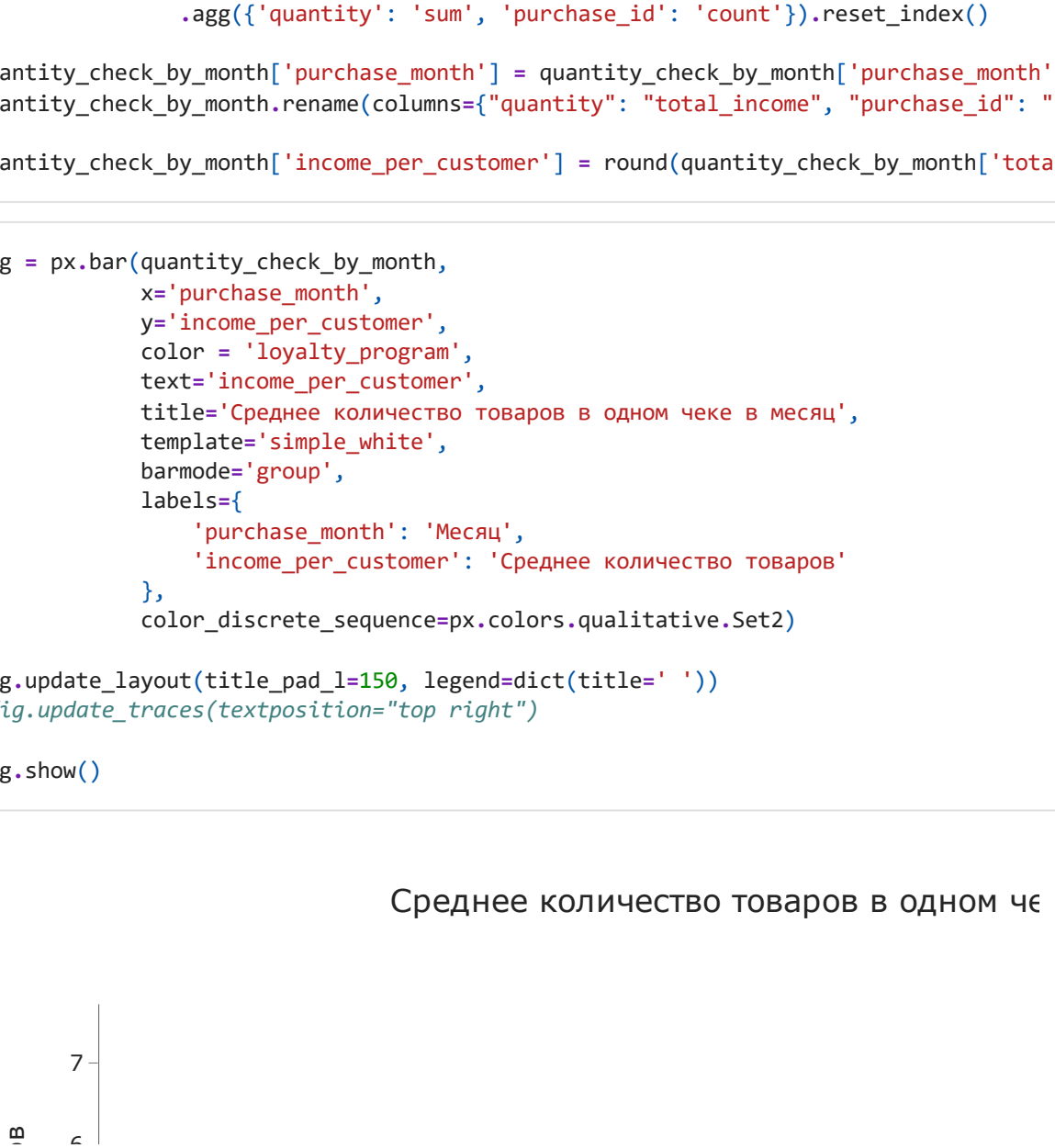


Среднее количество товаров на пользователя в месяц без карты лояльности выше, чем с ней.

Стоит отметить, что покупатели с картой каждый месяц покупают все больше товаров.

In [65]:
```
quantity_check_by_month = retail.groupby(by=['loyalty_program', 'purchase_year', 'purc
                .agg({'quantity': 'sum', 'purchase_id': 'count'}).reset_index()

quantity_check_by_month['purchase_month'] = quantity_check_by_month['purchase_month'].
quantity_check_by_month.rename(columns={"quantity": "total_income", "purchase_id": "nu

quantity_check_by_month['income_per_customer'] = round(quantity_check_by_month['total_
```

In [66]:
```
fig = px.bar(quantity_check_by_month,
             x='purchase_month',
             y='income_per_customer',
             color = 'loyalty_program',
             text='income_per_customer',
             title='Среднее количество товаров в одном чеке в месяц',
             template='simple_white',
             barmode='group',
             labels={
                 'purchase_month': 'Месяц',
                 'income_per_customer': 'Среднее количество товаров'
             },
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(title_pad_l=150, legend=dict(title=' '))
#fig.update_traces(textposition="top right")

fig.show()
```



Среднее количество товаров в одном чеке в месяц с картой лояльности выше, чем с без нее.

Программа лояльности стимулирует покупать больше товаров.

Средний чек

In [67]:
```
income_by_month = retail.groupby(by=['loyalty_program', 'purchase_year', 'purchase_mor
                .agg({'revenue': 'sum', 'customer_id': 'nunique'}).reset_index()

income_by_month['purchase_month'] = income_by_month['purchase_month'].replace({12: 'де
income_by_month.rename(columns={"revenue": "total_income", "customer_id": "number_of_c

income_by_month['income_per_customer'] = round(income_by_month['total_income'] / incom
```

In [68]:
```
fig = px.bar(income_by_month,
             x='purchase_month',
             y='income_per_customer',
             color = 'loyalty_program',
             text='income_per_customer',
             title='Средней чек в месяц',
             template='simple_white',
             barmode='group',
             labels={
                 'purchase_month': 'Месяц',
                 'income_per_customer': 'Средней чек'
             },
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(title_pad_l=150, legend=dict(title=' '))
#fig.update_traces(textposition="top right")

fig.show()
```

Средний чек без карты лояльности на графике по месяцам выше, чем с ней на протяжении всех месяцев наблюдений.

LVT

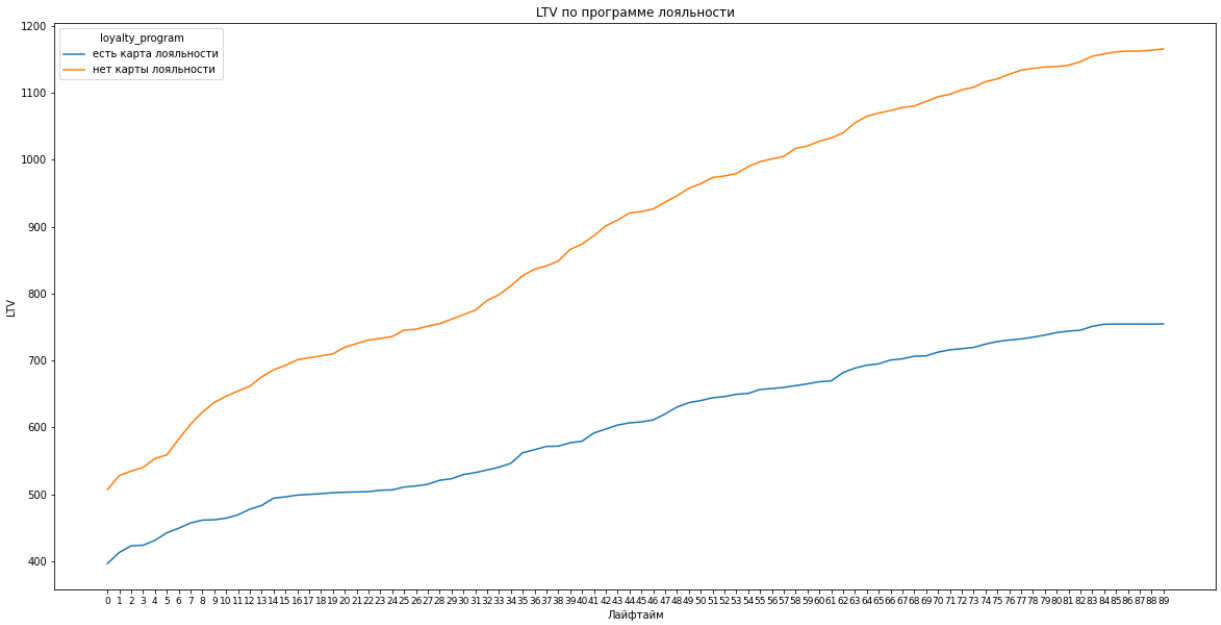
In [69]:
```
# Формирование когорт покупателей по дате первого заказа
first_order_date_by_customers = retail.groupby('customer_id')['purchase_date'].min()
first_order_date_by_customers.name = 'first_order_date'

# Объединение с основной таблицей
retail = retail.join(first_order_date_by_customers,on='customer_id')
```

In [70]:
```
retail = retail.sort_values(by=['customer_id', 'first_order_date'])
```

In [71]:
```
retail['lifetime'] = (retail['purchase_date'] - retail['first_order_date']).dt.days
```

In [72]:
```
dims = 'loyalty_program'

result = retail.pivot_table(index=dims, columns='lifetime', values='revenue',aggfunc='

result = result.fillna(0).cumsum(axis=1)

cohort_sizes = (retail.groupby(dims).agg({'customer_id': 'nunique'}).rename(columns={'

result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)

result = result.div(result['cohort_size'], axis=0)
```

In [73]:
```
report = result.drop(columns=['cohort_size'])
report.T.plot(grid=False, figsize=(20, 10), xticks=list(report.columns.values))
plt.tick_params(axis='x', which='both', labelsize=9)
plt.title('LTV по программе лояльности')
plt.ylabel('LTV')
plt.xlabel('Лайфтайм')
#plt.legend(['нет карты лояльности','есть карта лояльности'], loc=2)
plt.show()
```



В рамках когортного анализа расчёта LTV выручка покупателей с картой лояльности меньше, чем у покупателей без карты лояльности.

In [74]:
```
#количество уникальных покупателей с картой лояльности каждый месяц
customer_loyalty = retail.query('loyalty_program == "есть карта лояльности"')\
.groupby('purchase_month').agg({'customer_id': 'nunique'})
# Умножаем на 200 рублей
customer_loyalty['customer_revenue'] = customer_loyalty['customer_id'] * 200
customer_loyalty
```

Out[74]:

purchase_month	customer_id	customer_revenue
1	232	46400
2	257	51400
12	327	65400

Выручка от продажи карт лояльности на каждый месяц

In [75]:
```
# Таблица выручки
result_card = retail.pivot_table(index=dims, columns='lifetime', values='revenue',aggf
result_card
```

Out[75]:

loyalty_program	lifetime	0	1	2	3	4	5	6	7
есть карта лояльности	223580.83	9333.050	5603.850	468.380	4095.035	6343.925	3888.165	4403.170	24
нет карты лояльности	570831.70	23657.585	7479.835	6016.975	15050.625	6056.780	26469.300	24978.625	205

2 rows × 90 columns

In [76]:
```
# Прибавляем выручка от продаж карт в начало каждого месяца
```

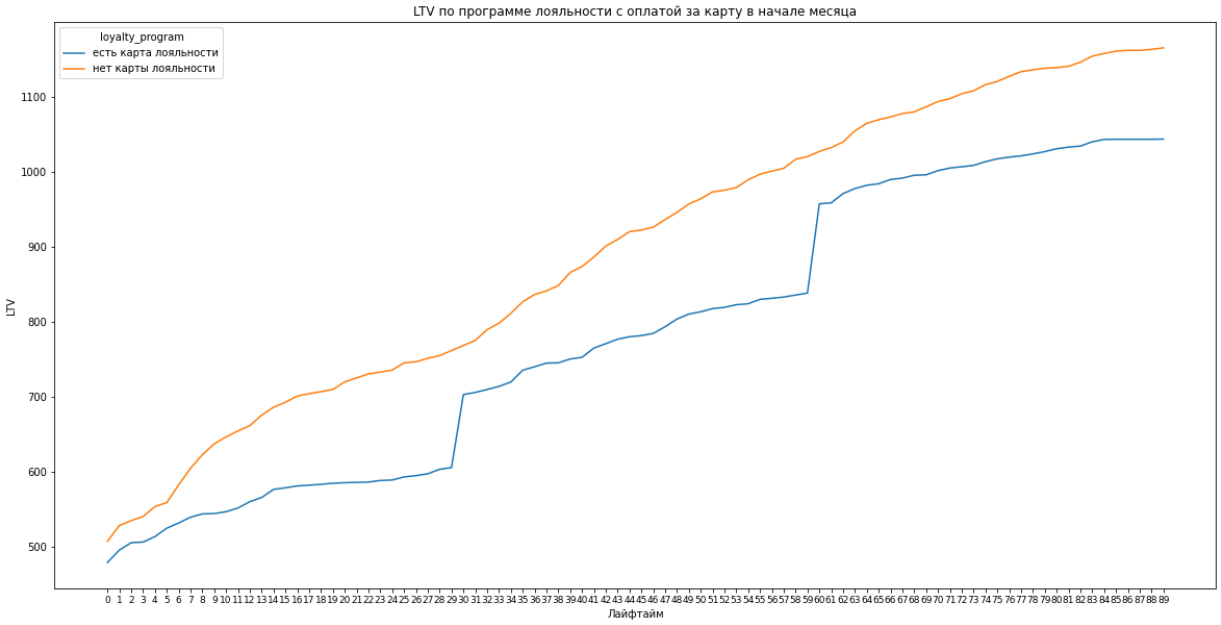
```
result_card.loc['есть карта лояльности',0] = \
result_card.loc['есть карта лояльности',0] + customer_loyalty.loc[1,'customer_revenue'

result_card.loc['есть карта лояльности',30] = \
result_card.loc['есть карта лояльности',30] + customer_loyalty.loc[2,'customer_revenue

result_card.loc['есть карта лояльности',60] = \
result_card.loc['есть карта лояльности',60] + customer_loyalty.loc[12,'customer_revenu
```

In [77]:

```
result_card = result_card.fillna(0).cumsum(axis=1)

cohort_sizes_card = (retail.groupby(dims).agg({'customer_id': 'nunique'}).rename(colum

result_card = cohort_sizes_card.merge(result_card, on=dims, how='left').fillna(0)

result_card = result_card.div(result_card['cohort_size'], axis=0)
```

In [78]:

```
report_card = result_card.drop(columns=['cohort_size'])
report_card.T.plot(grid=False, figsize=(20, 10), xticks=list(report_card.columns.value
plt.tick_params(axis='x', which='both', labelsize=9)
plt.title('LTV по программе лояльности с оплатой за карту в начале месяца')
plt.ylabel('LTV')
plt.xlabel('Лайфтайм')
#plt.legend(['нет карты лояльности','есть карта лояльности'], loc=2)
plt.show()
```



В рамках когортного анализа расчёта LTV выручка покупателей с картой лояльности меньше, даже с учетом стоимости покупки карты.

Вывод

Большинство покупателей предпочитают не участвовать в программе лояльности магазина.

Покупатели без карты лояльности чаще совершают покупки.

Среднее количество товаров на пользователя в месяц без карты лояльности выше, чем с ней.

Среднее количество товаров в одном чеке в месяц с картой лояльности выше, чем с без нее.
Программа лояльности стимулирует покупать больше товаров.

Средний чек без карты лояльности на графике по месяцам выше, чем с ней на протяжении всех месяцев наблюдений.

В рамках когортного анализа расчёта LTV выручка покупателей с картой лояльности меньше, чем у покупателей без карты лояльности.

Программа лояльности не работает, не повышает средний чек, но стимулирует брать большее количество товаров.
В рамках когортного анализа расчёта LTV выручка покупателей с картой лояльности меньше, даже с учетом стоимости покупки карты.

Шаг 4. Проверка гипотез

Среднее количество покупаймых товаров с картой и без нее

Нулевая гипотеза: среднее количество покупаймых товаров с картой лояльности и без неё не отличается.
Альтернативная гипотеза: среднее количество покупаймых товаров с картой лояльности и без неё будет разным.

In [79]:
```
retail= retail[retail['quantity']>0]
#делаем срезы, считаем количество товаров в чеке
sample_1 = (retail
                    .query('loyalty_program == "есть карта лояльности"')
                    .pivot_table(index = 'purchase_id', values = 'quantity', aggfunc =
sample_2 = (retail
                    .query('loyalty_program == "нет карты лояльности"')
                    .pivot_table(index = 'purchase_id', values = 'quantity', aggfunc =
```

Так как у нас две назависимые выборки, а в данных есть выбросы, то для проверки гипотезы, будем пользоваться критерием Манна-Уитни.

In [80]:
```
alpha = 0.05

results = st.mannwhitneyu(
    sample_1['quantity'],
    sample_2['quantity'],
    alternative='less')

print('p-value:', '{0:.6f}'.format(results.pvalue))

if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

```
p-value: 0.002559
Отвергаем нулевую гипотезу
```

Необходимо отвергнуть нулевую гипотезу. Количество товаров в чеке отличается в зависимости от факта наличия карты лояльности.

Средний чек с картой и без нее

Нулевая гипотеза: средний чек с картой программы лояльности и без неё не отличается.

Альтернативная гипотеза: средний чек с картой лояльности отличается от среднего чека без неё

In [81]:
```
# делаем срезы, считаем средний чек
loyalty_program_1 = (retail.query('loyalty_program == "есть карта лояльности"')
                           .pivot_table(index = 'purchase_id', values = 'revenue', ag
loyalty_program_0 = (retail
                           .query('loyalty_program == "нет карты лояльности"')
                           .pivot_table(index = 'purchase_id', values = 'revenue', ag
```

Так как у нас две назависимые выборки, а в данных есть выбросы, то для проверки гипотезы, будем пользоваться критерием Манна-Уитни.

In [82]:
```
alpha = 0.05

results = st.mannwhitneyu(
    loyalty_program_1['revenue'],
    loyalty_program_0['revenue'],
    alternative='less')

print('p-value:', '{0:.6f}'.format(results.pvalue))

if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

```
p-value: 0.000356
Отвергаем нулевую гипотезу
```

Придется отвергнуть нулевую гипотезу и признать, что средний чек с картой лояльности и без значительно отличается.

Вывод

По результатам статистического анализа установили, что в средних расходах и среднем количестве покупок у покупателей, участвующих в программе лояльности и нет, существуют статистически значимые различия, то есть клиенты не участвующие в программе лояльности тратят больше и покупают часто.

Программа лояльности не работает.

Выводы и рекомендации

Вывод

Было изучено и проведена предобработка данных:

Заменили название столбцов на удобные.
Дубликатов в датасете `retail_dataset` оказалось 1033, что состовляет 1%.
Удалили все полные дубликаты.
Заменили на медийное значение товары с несколькими вариантами цен.

Заменили пропуски на нули.
Изменили типы данных у столбца `purchase_date` на "datetime64", а у столбцов `customer_id и loyalty_program` на "int64".
Добавили столбцы с годом, месяцем, неделей, датой и общей суммой покупки.
Объединил две таблицы в одну с названием `retail` .

Аналих Данных:

Были удалены выбросы, общее количество записей стало 104289.
Shop 0 явно самый крупный магазин сети. Это либо интернет магазин, либо оптовый склад.
В тройку лидеров входят магазины номер 4, 1 и 6. Есть магазинов, где продали менее 100 позиций.
Наблюдения распределены по временной шкале примерно равномерно, делать срез данных за какой-то определенный адекватный период не требуется.
В датасете отсутсвуют данные за период с 24 декабря 2016 по 3 января 2017.

Аналих программы лояльности:

Большинство покупателей предпочитают не участвовать в программе лояльности магазина.
Покупатели без карты лояльности чаще совершают покупки.
Среднее количество товаров на пользователя в месяц без карты лояльности выше, чем с ней.
Среднее количество товаров в одном чеке в месяц с картой лояльности выше, чем с без нее.
Программа лояльности стимулирует покупать больше товаров.
Средний чек без карты лояльности на графике по месяцам выше, чем с ней на протяжении всех месяцев наблюдений.
В рамках когортного анализа расчёта LTV выручка покупателей с картой лояльности меньше, чем у покупателей без карты лояльности.
В рамках когортного анализа расчёта LTV выручка покупателей с картой лояльности меньше, даже с учетом стоимости покупки карты.

Программа лояльности не работает, не повышает средний чек, но стимулирует брать большее количество товаров.

Проверка гипотиз:

По результатам статистического анализа установили, что в средних расходах и среднем количестве покупок у покупателей, участвующих в программе лояльности и нет, существуют статистически значимые различия, то есть клиенты не участвующие в программе лояльности тратят больше и покупают часто.

Программа лояльности не работает.

Рекомендации

Нужно поменять программу лояльности или совсем отказаться от нее.
Необходимо посмотреть данные за более продолжительный период, например за год, так как возможно покупатели с картой лояльности ждут начала сезона.
Стоит изучить большие или оптовые продажи.

В чеках содержится довольно много бесплатных товаров, если это всё подарки, то нужно выяснить насколько это окупается.