Дипломный проект: "Анализ результатов внедрения программы лояльности"

Материалы:

Ссылка на интерактивный график: https://public.tableau.com/app/profile/karin.vink/viz/Final_project_16721149442770/Dashboard1?publish=yes

Ссылка на презентацию: https://cloud.mail.ru/public/oPgC/oQA5eq8mc

Описание проекта

Заказчик: менеджер проекта, отвечающий за программу лояльности

Входные данные: датасет содержит данные о покупках в магазине строительных материалов «Строили, строили и наконец построили» за период декабря 2016 - февраль 2017гг.

Цель исследования: оценить результаты внедрения и понять насколько сработала программа лояльности (проверить, повышаются ли основные показатели у клиентов, использующих программу) для дальнейшего принятия менеджером решения об её использовании.

Загрузка данных и подготовка к исследованию

Импортируем необходимые библиотеки:

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from plotly import graph_objects as go #для построения круговой диаграммы
import scipy.stats as stats # для проверки стат. гипотез
```

Информация о датасете data:

- 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

Out[3]:

	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
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

Проверим корректность наименований колонок в датасете data и выполним переименования:

In [4]:
```
data = data.rename(columns={'purchaseid':'purchase_ID', 'purchasedate':'purchase_date','Quantity':'quantity', 'Cu
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105335 entries, 0 to 105334
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   purchase_ID      105335 non-null  object
 1   item_ID          105335 non-null  object
 2   quantity         105335 non-null  int64 
 3   purchase_date    105335 non-null  object
```

```
 4   customer_ID      69125 non-null   float64
 5   shop_ID          105335 non-null  object
 6   loyalty_program  105335 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 5.6+ MB
```

Информация о датасете df:

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

Out[7]:

	productID	price_per_one
0	85123A	2.55
1	71053	3.39
2	84406B	2.75

Проверим корректность наименований колонок в датасете df и выполним переименования:

In [8]:

```
df = df.rename(columns={'productID':'item_ID'})
```

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

Исследуем пропущенные значения:

Out[11]:

```
purchase_ID            0
item_ID                0
quantity               0
purchase_date          0
customer_ID        36210
shop_ID                0
loyalty_program        0
dtype: int64
```

Посмотрим долю пропусков в % от всех данных в столбце customer_ID:

Out[12]:

```
purchase_ID         0.0
item_ID             0.0
quantity            0.0
purchase_date       0.0
customer_ID        34.4
shop_ID             0.0
loyalty_program     0.0
dtype: float64
```

Т.к. пропуски в столбце customer_ID занимают значительную долю (34,4%) то их удаление негативно полвияет на точность результатов исследования, поэтому в данном случае обработаем пропуски заменив их на нулевые значения.

In [13]:

```
data['customer_ID'] = data['customer_ID'].fillna(value=0)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105335 entries, 0 to 105334
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   purchase_ID      105335 non-null  object
 1   item_ID          105335 non-null  object
 2   quantity         105335 non-null  int64
 3   purchase_date    105335 non-null  object
```

```
 4   customer_ID      105335 non-null  float64
 5   shop_ID          105335 non-null  object
 6   loyalty_program  105335 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 5.6+ MB
```

Также вызывает вопрос случаи, когда в поле 'quantity' значение = 0, т.е. в данном чеке кол-во данного товара = 0. Тогда почему этот товар присутсвует в чеке?

In [14]:
```
len(data[data['quantity']==0])
```

Out[14]:
```
33055
```

Проверим, есть ли отрицательные значения в поле 'quantity':

In [15]:
```
len(data[data['quantity'] < 0])
```

Out[15]:
```
2118
```

In [16]:
```
vozvrat = [data[data['quantity'] < 0]]
vozvrat
```

Out[16]:
```
[       purchase_ID item_ID  quantity        purchase_date  customer_ID  \
 64         C539944   22776        -2  2016-12-23 11:38:00      20239.0
 109        C542910   20726        -2  2017-02-01 15:38:00      23190.0
 112        C542426   22418       -25  2017-01-28 09:32:00      19825.0
 253        C539726   22791       -11  2016-12-21 14:24:00      22686.0
 344        C544034   21878        -2  2017-02-15 11:28:00      20380.0
 ...            ...     ...       ...                  ...          ...
 105160     C541650       M        -2  2017-01-20 11:44:00          0.0
 105172     C540246   79320        -2  2017-01-05 15:43:00      18760.0
 105211     C539467   22801        -2  2016-12-19 12:46:00      20723.0
 105250     C540847   22197        -3  2017-01-11 17:35:00      19137.0
 105300     C540164   21144       -13  2017-01-05 12:02:00      20590.0

        shop_ID  loyalty_program
 64      Shop 0              0.0
 109     Shop 0              1.0
 112     Shop 0              0.0
 253     Shop 0              1.0
 344     Shop 0              0.0
 ...        ...              ...
 105160  Shop 0              0.0
 105172  Shop 0              0.0
 105211  Shop 0              0.0
 105250  Shop 0              0.0
 105300  Shop 6              0.0

 [2118 rows x 7 columns]]
```

Отрицательные значения в поле 'quantity' могут быть связаны с возвратами товаров. Посмотрим, какое количество возвратов в процентном отношении было сделано клиентами программы лояльности, а какое обычными клиентами:

Out[18]:

	loyalty_program	amount_return_item	%
0	0.0	1673	79.0
1	1.0	445	21.0

Промежуточные выводы: по сделанным расчётам видим, что из 2118 наименований товаров, которые были возвращены 21% приходится на покупки по программе лояльности, остальные 79% на остальных покупателей.

Посмотрим, какой процент от всех записей занимают отрицательные значения:

In [19]:
```
print('% возвратов от всех записей в таблице:', round(len(data[data['quantity'] < 0])/len(data)*100,2))
```

```
% возвратов от всех записей в таблице: 2.01
```

Для дальнейшего анализа результатов программы исключим данные по возвратам:

In [20]:
```
data = data[data['quantity'] >= 0]
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103217 entries, 0 to 105334
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   purchase_ID      103217 non-null  object 
 1   item_ID          103217 non-null  object 
 2   quantity         103217 non-null  int64  
 3   purchase_date    103217 non-null  object 
 4   customer_ID      103217 non-null  float64
 5   shop_ID          103217 non-null  object 
 6   loyalty_program  103217 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.3+ MB
```

Посмотрим количество записей может быть по одному и тому же чеку и в результате видим, что идентификатор чека может повторяться в зависимости от того, сколько товаров в нём было, а значит ситуация с тем, что какой-то товар был обозначен в чеке как 0 (например, добавили в чек, а потом отменили) технически допустима.

In [21]:
```
data['purchase_ID'].value_counts()
```

Out[21]:
```
537434    675
538071    652
538349    620
537638    601
537237    597
         ...
541513      1
542783      1
543675      1
543988      1
544612      1
Name: purchase_ID, Length: 3875, dtype: int64
```

Проверим наличие пропусков в таблице df:

Out[22]:
```
item_ID          0
price_per_one    0
dtype: int64
```

Исследуем соответствие типов:

- purchasedate - необходимо изменить на тип datetime
- CustomerID - необходимо изменить на тип object
- loyalty_program - необходимо изменить на тип int

Выполним изменение типов:

In [23]:
```
data['purchase_date']= pd.to_datetime(data['purchase_date'], format ='%Y-%m-%dT%H:%M:%S')
```

In [24]:
```
data['customer_ID'] = data['customer_ID'].astype('object')
data['loyalty_program'] = data['loyalty_program'].astype('bool')
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103217 entries, 0 to 105334
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype         
---  ------           --------------   -----         
 0   purchase_ID      103217 non-null  object        
 1   item_ID          103217 non-null  object        
 2   quantity         103217 non-null  int64         
 3   purchase_date    103217 non-null  datetime64[ns]
 4   customer_ID      103217 non-null  object        
 5   shop_ID          103217 non-null  object        
```

```
 6   loyalty_program  103217 non-null  bool
dtypes: bool(1), datetime64[ns](1), int64(1), object(4)
memory usage: 5.6+ MB
```

In [26]:
```
data.head()
```

Out[26]:

	purchase_ID	item_ID	quantity	purchase_date	customer_ID	shop_ID	loyalty_program
0	538280	21873	11	2016-12-10 12:50:00	18427.0	Shop 0	False
1	538862	22195	0	2016-12-14 14:11:00	22389.0	Shop 0	True
2	538855	21239	7	2016-12-14 13:50:00	22182.0	Shop 0	True
3	543543	22271	0	2017-02-09 15:33:00	23522.0	Shop 0	True
4	543812	79321	0	2017-02-13 14:40:00	23151.0	Shop 0	True

В таблице df с типами всё в порядке, замены не требуется.

Исследуем наличие дубликатов:

Сначала посчитаем количество явных строк-дубликатов в таблице:

In [27]:
```
'Дубликатов в таблице:', data.duplicated().sum()
```

Out[27]:
```
('Дубликатов в таблице:', 991)
```

Удалим строки-дубликаты и заменим индексацию на новую:

In [28]:
```
data = data.drop_duplicates().reset_index(drop=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102226 entries, 0 to 102225
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   purchase_ID      102226 non-null  object
 1   item_ID          102226 non-null  object
 2   quantity         102226 non-null  int64
 3   purchase_date    102226 non-null  datetime64[ns]
 4   customer_ID      102226 non-null  object
 5   shop_ID          102226 non-null  object
 6   loyalty_program  102226 non-null  bool
dtypes: bool(1), datetime64[ns](1), int64(1), object(4)
memory usage: 4.8+ MB
```

Теперь проверим на наличие не явных строк-дубликатов:

дубликаты появляются в сочетании столбцов data[['purchaseid','item_ID','purchasedate', 'CustomerID']], но если мы добавляем к ним столбец Quantity, то дубликатов нет. Как мы выяснили выше, ситуация с разным количеством товаров является нормальной, проблемы в данных с неявными дубликатами нет.

Проверим наличие дубликатов в таблице df:

In [29]:
```
'Дубликатов в таблице df:', df.duplicated().sum()
```

Out[29]:
```
('Дубликатов в таблице df:', 0)
```

Проверим на неявные дубликаты:

In [30]:
```
povtor_1 = df.pivot_table(index=['item_ID'], values='price_per_one', aggfunc='count').reset_index()
povtor_1.sort_values(by='price_per_one', ascending = False)
```

Out[30]:

	item_ID	price_per_one

3150	DOT	174
3151	M	59
3153	S	29
3152	POST	15
3139	D	13
...	...	...
2593	85018C	1
2594	85018D	1
2596	85019B	1
2006	47420	1
3158	m	1

3159 rows × 2 columns

Видим, что в таблице со стоимостью товаров ряд товаров содержит несколько различных цен. Посмотрим, на двух продуктах, какие это значения, насколько в рамках одного товара они отличаются друг от друга:

In [31]:
```
df[df['item_ID']=='DOT'].sort_values(by ='price_per_one', ascending= False )
```

Out[31]:

	item_ID	price_per_one
3773	DOT	950.99
3541	DOT	940.87
4733	DOT	907.47
2655	DOT	887.52
4485	DOT	885.94
...	...	...
5724	DOT	50.64
5723	DOT	50.60
5726	DOT	29.53
8778	DOT	3.29
6132	DOT	2.51

174 rows × 2 columns

In [32]:
```
df[df['item_ID']=='M'].sort_values(by ='price_per_one', ascending= False )
```

Out[32]:

	item_ID	price_per_one
8729	M	1715.85
9476	M	1435.79
5777	M	1298.40
8167	M	1283.80
5043	M	1130.90
6243	M	1126.00
3882	M	924.59
9478	M	869.55
9378	M	856.48
9899	M	764.12
3900	M	631.31
8166	M	544.40
6217	M	458.29
9704	M	320.69
5042	M	316.30
3901	M	313.78
9413	M	208.16
9380	M	190.80

4624	M	133.08
9641	M	82.50
9363	M	71.46
9898	M	48.00
3634	M	35.00
9853	M	30.19
5639	M	22.00
6255	M	21.95
5070	M	20.00
1466	M	18.95
4538	M	15.00
6652	M	12.75
8303	M	10.00
6404	M	6.70
3778	M	5.95
5072	M	5.00
8739	M	4.65
9362	M	4.50
5068	M	4.25
8748	M	4.00
9494	M	3.75
5385	M	3.00
4251	M	2.95
2506	M	2.55
7366	M	2.50
9411	M	1.95
6126	M	1.65
5271	M	1.45
1462	M	1.25
3275	M	1.00
7585	M	0.87
2777	M	0.85
3195	M	0.65
4511	M	0.50
3071	M	0.42
9332	M	0.41
8087	M	0.38
5238	M	0.32
5049	M	0.21
6080	M	0.20
3620	M	0.19

Видим, что разброс цен в рамках одного товара может быть очень значительный. Чтобы исключить ситуацию, когда по одному товару в таблице df имеем несколько цен, посчитаем для каждого товара медианное значение цены.

Out[33]:

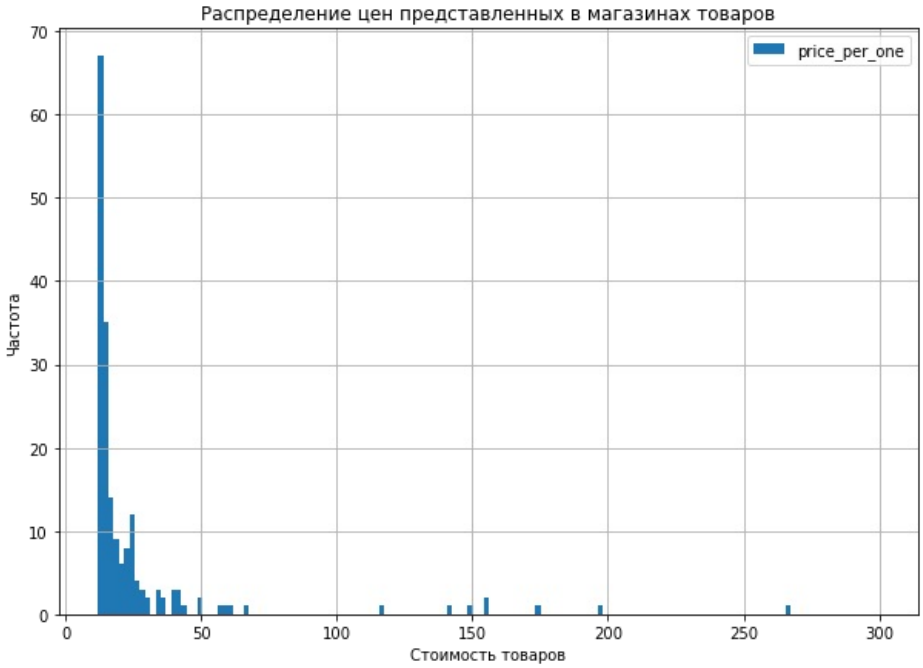
	item_ID	price_per_one
3136	AMAZONFEE	6706.71
1507	22655	265.50
3150	DOT	198.19
1671	22826	175.00
1672	22827	155.00

Посчитаем количество товаров, которые имеют нулевую стоимость:

In [37]:
```
print('Количество товаров с нулевой стоимостью в таблице df:',len(df[df['price_per_one']==0]))
```

```
Количество товаров с нулевой стоимостью в таблице df: 57
```

Посмотрим на гистограмме, как распределены цены:



Промежуточные выводы: на гистограмме видим и отсортированным данным видим, что в магазинах сети продаются товары в ценовом диапазоне 15 - 45 у.ед., товары по более высокой стоимости встречаются крайне редко.

- Есть товар item_ID=='AMAZONFEE', с крайне выходящей за пределы нормальной стомостью (6706.71). Т.к. такой товар был куплен единожды, можем предположить, что такой товар был куплен под заказ и исплючить этот выброс из дальнейших рассчётов, чтобы не искажать показатель среднего чека.
- Также уберем из расчётов товары и покупки с нулевой стоимостью.

In [39]:
```
df = ( df[
          (df['item_ID']!='AMAZONFEE')
          &(df['price_per_one']!=0)
         ]
     )
```

Добавим к таблице data данные из таблицы df:

Out[40]:

	purchase_ID	item_ID	quantity	purchase_date	customer_ID	shop_ID	loyalty_program	price_per_one
0	538280	21873	11	2016-12-10 12:50:00	18427.0	Shop 0	False	1.63
1	541104	21873	0	2017-01-13 14:29:00	0.0	Shop 0	False	1.63
2	540418	21873	1	2017-01-07 11:04:00	0.0	Shop 0	False	1.63
3	541516	21873	2	2017-01-18 17:34:00	0.0	Shop 0	False	1.63
4	541566	21873	35	2017-01-19 11:50:00	23401.0	Shop 0	True	1.63
...	...	...	...	...	...	...	...	...
102201	538200	15058A	0	2016-12-10 11:11:00	23591.0	Shop 0	True	7.95
102202	538852	90058B	35	2016-12-14 13:33:00	23051.0	Shop 0	True	0.38
102203	539988	46138B	1	2016-12-23 16:06:00	23795.0	Shop 0	True	1.95
102204	537025	90053	0	2016-12-03 16:21:00	0.0	Shop 0	False	2.55
102205	542731	17028J	5	2017-01-31 15:27:00	19279.0	Shop 0	False	0.42

102206 rows × 8 columns

```
Покупок с нулевой стоимостью в таблице data: 0
```

Проверим, что после обогащения данных, количество строк в исходной таблице не изменилось:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 102206 entries, 0 to 102205
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   purchase_ID      102206 non-null  object
 1   item_ID          102206 non-null  object
 2   quantity         102206 non-null  int64
 3   purchase_date    102206 non-null  datetime64[ns]
 4   customer_ID      102206 non-null  object
 5   shop_ID          102206 non-null  object
 6   loyalty_program  102206 non-null  bool
 7   price_per_one    102206 non-null  float64
dtypes: bool(1), datetime64[ns](1), float64(1), int64(1), object(4)
memory usage: 6.3+ MB
```

Исследовательский анализ данных

Посмотрим, за какой период получены данные:

```
Минимальная дата записи: 2016-12-01 08:26:00
Максимальная дата записи: 2017-02-28 17:01:00
```

Т.е. имеем данные за три месяца: декабрь 2016г, январь и февраль 2017г.

Добавим в объединенную таблицу столбец с днем недели, месяцем и годом покупки

Out[47]:

	purchase_ID	item_ID	quantity	purchase_date	customer_ID	shop_ID	loyalty_program	price_per_one	day_of_week_sale	day_of_sale	montl
0	538280	21873	11	2016-12-10 12:50:00	18427.0	Shop 0	False	1.63	5	2016-12-10	
1	541104	21873	0	2017-01-13 14:29:00	0.0	Shop 0	False	1.63	4	2017-01-13	
2	540418	21873	1	2017-01-07 11:04:00	0.0	Shop 0	False	1.63	5	2017-01-07	

Для дальнейших расчётов собёрем из таблицы data данные по каждому чеку:

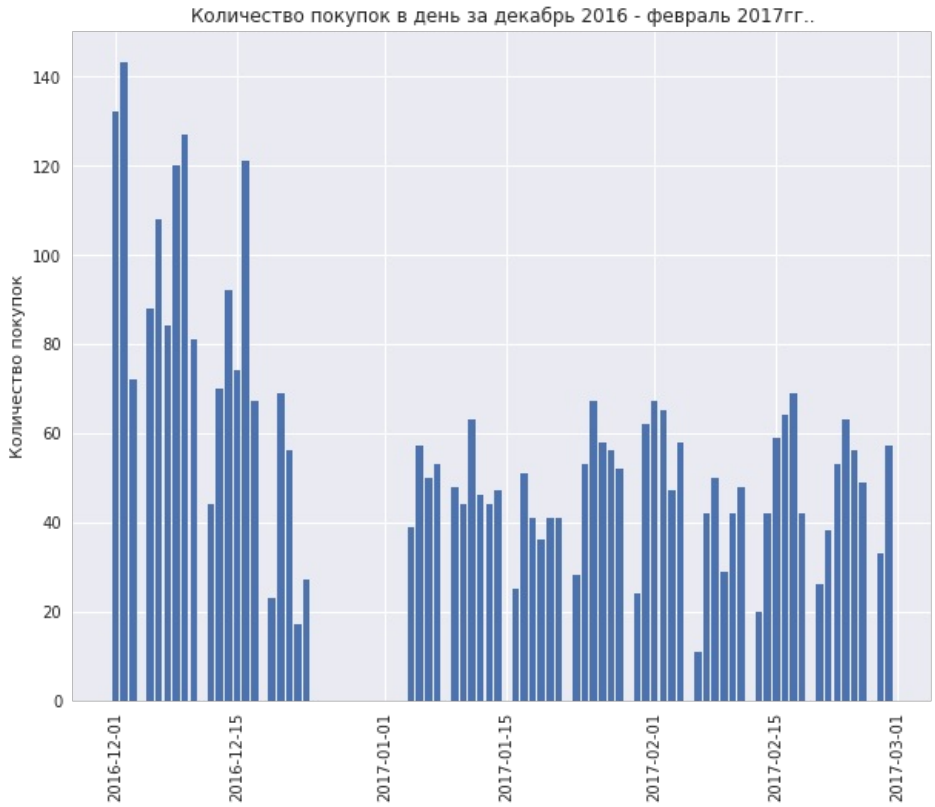
Out[48]:

	purchase_ID	customer_ID	shop_ID	loyalty_program	purchase_date	day_of_sale	day_of_week_sale	month_year_sale	purchase_price
0	536365	23529.0	Shop 0	True	2016-12-01 08:26:00	2016-12-01	3	2016-12-01	45.905
1	536366	23529.0	Shop 0	True	2016-12-01 08:28:00	2016-12-01	3	2016-12-01	3.950
2	536367	18726.0	Shop 0	False	2016-12-01 08:34:00	2016-12-01	3	2016-12-01	69.455
3	536368	18726.0	Shop 0	False	2016-12-01 08:34:00	2016-12-01	3	2016-12-01	38.640
4	536369	18726.0	Shop 0	False	2016-12-01 08:35:00	2016-12-01	3	2016-12-01	6.600

Рассчитаем количество покупок за каждый день и построим соответсующий график:

Out[49]:

	day_of_sale	cnt_sale
0	2016-12-01	132
1	2016-12-02	143
2	2016-12-03	72
3	2016-12-05	88
4	2016-12-06	108



Промежуточные выводы: на графике видим, что количество покупок в день в декабрь 2016 до середины месяца было выше, чем январе и феврале 2017г., что могло быть связано со стремлением людей завершить ремонтные и строительные работы в предверии праздников. В конце декабря и начале января видим пробел в продажах. Он может быть связан как с техническими проблемами в записи данных, так и с графиком работы магазинов. В первом случае - необходимо обратить на причины технических проблем для их избежания в дальнейшем. Во втором случае - обратить внимание на график работы в конце и начале года в праздничные дни, т.к. для сети магазинов это может ежегодно быть причиной недополучения значительной прибыли.

Средний чек

Посчитаем средний чек по всем данным за имеющийся период для покупателей, учавствующих и НЕ учавствующих в программе лояльности:

Out[51]:

	loyalty_program	mean_chek
0	False	133.491759
1	True	87.190739

Теперь расчитаем показатель среднего чека по каждому месяцу, с разделением на учавствующих и нет в программе лояльности:

(без учёта стоимости покупки карты для участия в программе лояльности)

Выделим из таблицы cheki, чеки покупателей учавствующих в программе лояльности и не учавствующих:

In [52]:
```
prog_da = cheki[cheki['loyalty_program']==1]
```

In [53]:
```
prog_net = cheki[cheki['loyalty_program']==0]
```

Расчитаем средние чеки по месяцам для обоих категорий клиентов:

Out[54]:

	month_year_sale	loyalty_program	mean_chek_prog_da
0	2016-12-01	True	82.398444
1	2017-01-01	True	98.819741
2	2017-02-01	True	84.433881

Out[55]:

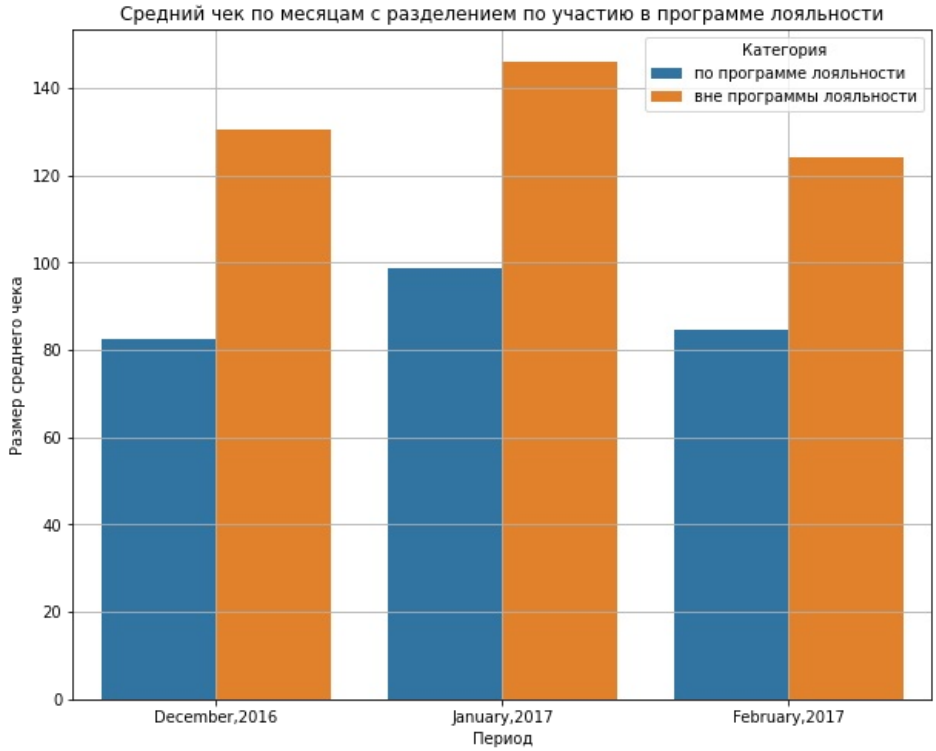
	month_year_sale	loyalty_program	mean_chek_prog_net
0	2016-12-01	False	130.553687
1	2017-01-01	False	146.122799
2	2017-02-01	False	124.289346

Объединим полученные таблицы в одну для построения графика:

Out[56]:

	month_year_sale	loyalty_program_x	mean_chek_prog_da	loyalty_program_y	mean_chek_prog_net
0	December,2016	True	82.398444	False	130.553687
1	January,2017	True	98.819741	False	146.122799
2	February,2017	True	84.433881	False	124.289346

Отобразим полученные расчётные значения на графике:



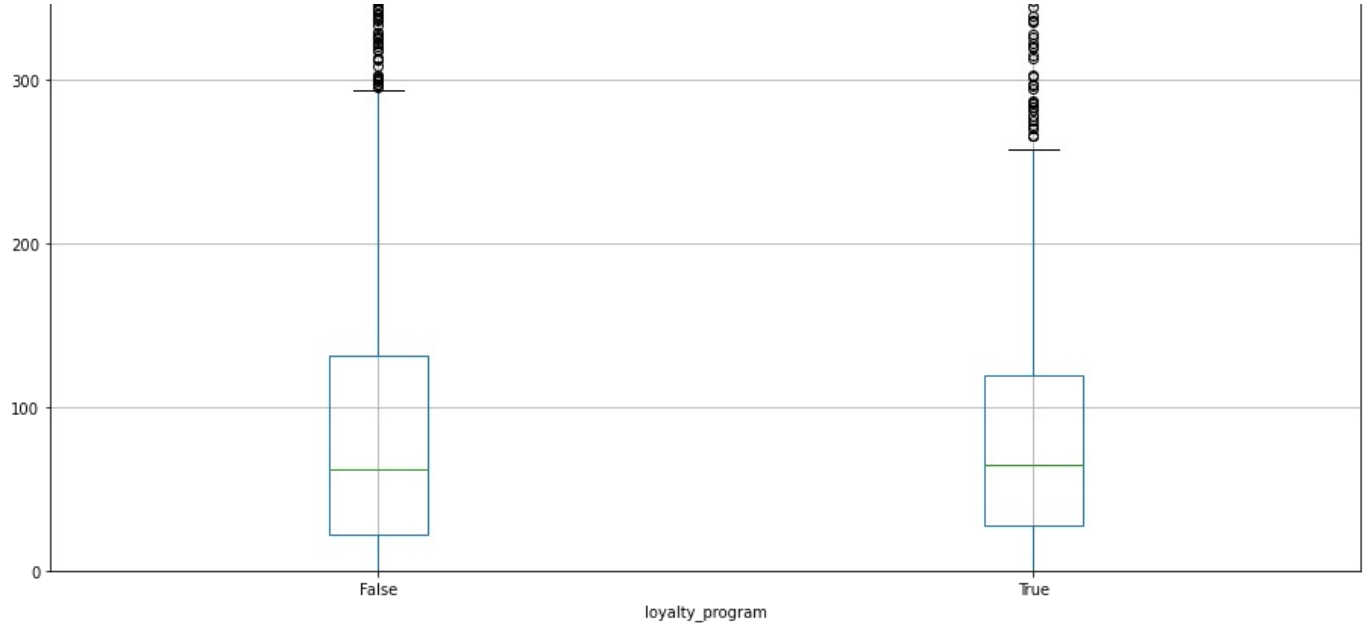
Промежуточные выводы: по сделанным расчетам и графику видим, что средний чек покупок, сделанных по программе лояльности за все месяца исследуемого периода существенно меньше, чем средний чек покупок вне программы лояльности.

Дополнительно посмотрим, как выглядит распределение размеров чеков по при использовании программы лояльности и без неё:

Построим диаграмму размаха ("Ящик с усами") по размерам чеков с использованием программы и без неё:

Out[58]:

```
(0.0, 500.0)
```



По диаграмме размаха ("Ящик с усами") видим, что нормальным распределением значений размера чеков для каждой категории (0 и 1) являются следующие значения:

- 0- покупки без участия в программе: стоимостью до 250 руб.
- 1- покупки с участием в программе: стоимостью до 240 руб.

Значения, расположенные после, являются выбросами.

Среднее количество покупок

Сначала определим соотношение покупок по количеству, сделанных по программе лояльности и вне её:

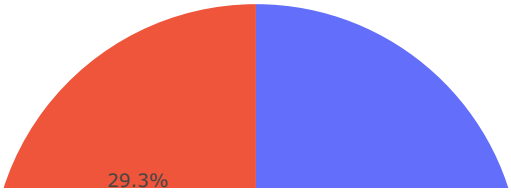
Рассчитаем и посмотрим на графике, количество и долю покупок сделанных по программе лояльности и вне её:

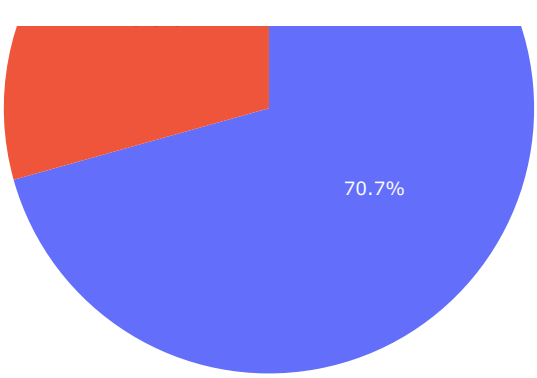
Out[59]:

	purchase_ID	customer_ID	shop_ID	loyalty_program	purchase_date	day_of_sale	day_of_week_sale	month_year_sale	purchase_price
0	536365	23529.0	Shop 0	True	2016-12-01 08:26:00	2016-12-01	3	2016-12-01	45.905
1	536366	23529.0	Shop 0	True	2016-12-01 08:28:00	2016-12-01	3	2016-12-01	3.950
2	536367	18726.0	Shop 0	False	2016-12-01 08:34:00	2016-12-01	3	2016-12-01	69.455
3	536368	18726.0	Shop 0	False	2016-12-01 08:34:00	2016-12-01	3	2016-12-01	38.640
4	536369	18726.0	Shop 0	False	2016-12-01 08:35:00	2016-12-01	3	2016-12-01	6.600

```
Количество покупок вне программы лояльности: 2735
```

```
Количество покупок по программе лояльности: 1136
```





Промежуточные вывыоды. На графике видим:

- что 70,8% покупок за исследуемый период сделаны вне программы лояльности,
- 29,2% покупок сделано по программе лояльности.

Посчитаем количество покупок по месяцам:

Out[63]:

	month_year_sale	loyalty_program	number_chek_prog_da
0	2016-12-01	True	511
1	2017-01-01	True	290
2	2017-02-01	True	335

Out[64]:

	month_year_sale	loyalty_program	number_chek_prog_net
0	2016-12-01	False	1104
1	2017-01-01	False	836
2	2017-02-01	False	795

Объединим полученные таблицы в одну для построения графика:

In [65]:
```
union_tab = cnt_month_chek_da.merge(cnt_month_chek_net, on = 'month_year_sale', how = 'left')
# сделаем замену значений в столбце 'month_year_sale', чтобы на графике период отображался в более удобном виде:
#union_tab ['month_year_sale']= union_tab ['month_year_sale'].replace(['2016-12-01','2017-01-01','2017-02-01'],['
union_tab['month_year_sale'] = union_tab['month_year_sale'].dt.strftime('%B,%Y')
```

In [66]:
```
union_tab
```

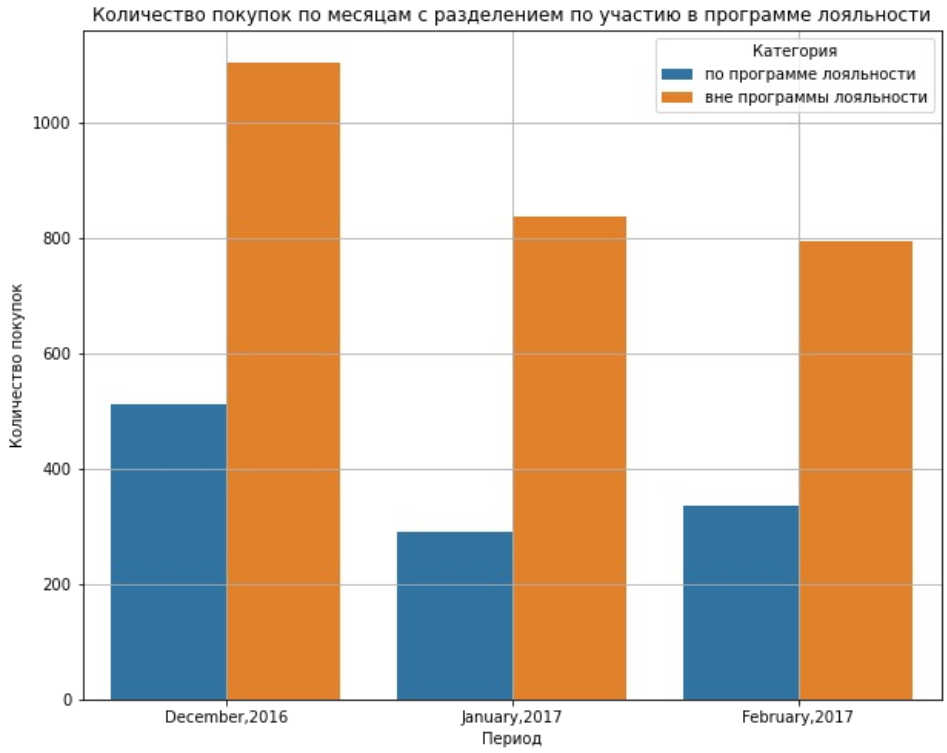
Out[66]:

	month_year_sale	loyalty_program_x	number_chek_prog_da	loyalty_program_y	number_chek_prog_net
0	December,2016	True	511	False	1104
1	January,2017	True	290	False	836
2	February,2017	True	335	False	795

Отобразим полученные расчётные значения на графике:

In [67]:
```
month = union_tab['month_year_sale']

# добавляем на график данные по месяцам о размере среднего чека клиентов, учавствующих 'в программе'
da = union_tab[['month_year_sale','number_chek_prog_da']].rename(columns={'number_chek_prog_da':'number_chek'})
da['Категория'] = 'по программе лояльности'

# добавляем на график данные по месяцам о размере среднего чека клиентов, 'вне программы'
net = union_tab[['month_year_sale','number_chek_prog_net']].rename(columns={'number_chek_prog_net':'number_chek'}
net['Категория'] = 'вне программы лояльности'
```

```
plt.figure(figsize=(10,8))

concat=pd.concat([da,net])
sns.barplot(x='month_year_sale', y='number_chek', hue='Категория', data=concat)

plt.grid()
plt.title('Количество покупок по месяцам с разделением по участию в программе лояльности')
plt.xlabel('Период')
plt.ylabel('Количество покупок')
plt.xticks #(rotation=90)
plt.show()
```



Промежуточные выводы: на графике видим, что количество покупок по программе лояльности по всем рассматриевым месяцам значительно меньше кол-ва покупок вне программы лояльности.

Теперь рассчитаем среднее количество покупок на одного покупателя по месяцам, также с учётом участия в программе лояльности.

Для расчёта используем таблицу ранее созданную cheki, разделённую на две части - данные о покупка по программе лояльности (таблица prog_da) и вне программы (таблица prog_net).

По программе лояльности посчитаем количество покупок у каждого покупателя в каждом расчётном месяце:

Out[68]:

	month_year_sale	customer_ID	number
0	2016-12-01	22006.0	1
1	2016-12-01	22032.0	4
2	2016-12-01	22044.0	2
3	2016-12-01	22046.0	1
4	2016-12-01	22064.0	1
...	...	...	...
807	2017-02-01	23909.0	1
808	2017-02-01	23910.0	1
809	2017-02-01	23929.0	1
810	2017-02-01	23936.0	2
811	2017-02-01	23962.0	1

812 rows × 3 columns

Здесь необходимо всмомнить, что в столбце 'customer_ID' у нас есть пропуски, которые мы заменили на нулевые значения. В данном случае их учёт может исказить результаты расчётов среднего количества покупок на одного покупателя, поэтому отфильруем таких "покупателей" на данном шаге:

In [69]:
```
number_sale_da = number_sale_da[number_sale_da['customer_ID']!=0]
```

Теперь посчитаем **среднее количество покупок у покупателей, учавствующих в программе лояльности**:

Out[70]:

	month_year_sale	mean_number_da
0	2016-12-01	1.59
1	2017-01-01	1.26
2	2017-02-01	1.29

Аналогично посчитаем количество покупок у каждого покупателя вне программе лояльности в каждом расчётном месяце:

Out[71]:

	month_year_sale	customer_ID	number
0	2016-12-01	0.0	214
1	2016-12-01	18026.0	1
2	2016-12-01	18027.0	1
3	2016-12-01	18049.0	2
4	2016-12-01	18056.0	1
...	...	...	...
1570	2017-02-01	21934.0	1
1571	2017-02-01	21944.0	2
1572	2017-02-01	21985.0	3
1573	2017-02-01	21996.0	1
1574	2017-02-01	22000.0	2

1575 rows × 3 columns

Здесь также отфильтруем из расчёта покупателей, у которых не был записан идентификатор:

In [72]:
```
number_sale_net = number_sale_net[number_sale_net['customer_ID']!=0]
```

Теперь посчитаем **среднее количество покупок у покупателей, НЕ учавствующих в программе лояльности**:

Out[73]:

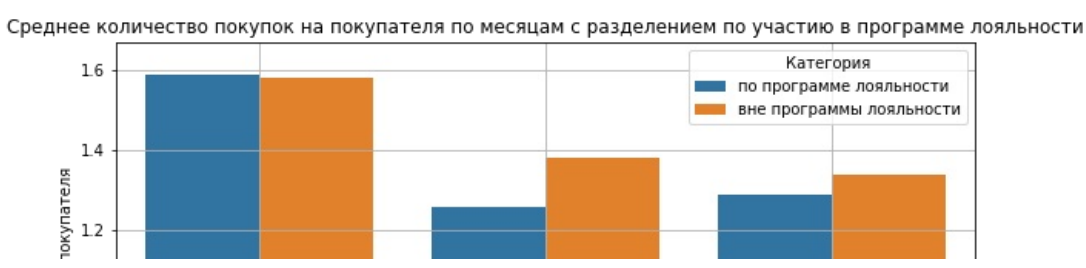
	month_year_sale	mean_number_net
0	2016-12-01	1.58
1	2017-01-01	1.38
2	2017-02-01	1.34

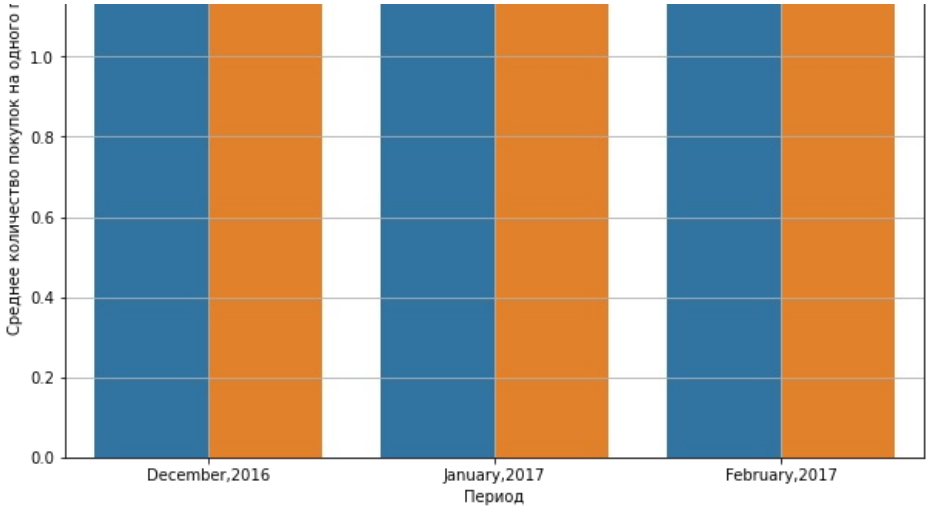
Объединим полученные таблицы в одну для построения графика:

Out[74]:

	month_year_sale	mean_number_da	mean_number_net
0	December,2016	1.59	1.58
1	January,2017	1.26	1.38
2	February,2017	1.29	1.34

Посмотрим наглядно полученный результат расчетов:





Промежуточные выводы: из расчетов и графика видим, что по показателю "Среднее количество покупок на одного покупателя" у покупателей вне программы лояльности этот показатель чуть выше, чем у покупателей учавствующих в программе.

Сделаем расчёт общей выручки от пользователя

In [76]:
```
data.head(3)
```

Out[76]:

	purchase_ID	item_ID	quantity	purchase_date	customer_ID	shop_ID	loyalty_program	price_per_one	day_of_week_sale	day_of_sale	montl
0	538280	21873	11	2016-12-10 12:50:00	18427.0	Shop 0	False	1.63	5	2016-12-10	
1	541104	21873	0	2017-01-13 14:29:00	0.0	Shop 0	False	1.63	4	2017-01-13	
2	540418	21873	1	2017-01-07 11:04:00	0.0	Shop 0	False	1.63	5	2017-01-07	

Составим профили пользователей:

- ID клиента,
- ID магазина,
- участие в программе лояльности,
- дата и время первой покупки,
- дата первой покупки,
- месяц первой покупки

Предварительно отфильтруем строки, где customer_ID=0

In [77]:
```
data_cl = data[data['customer_ID']!=0]
data_cl.head(1)
```

Out[77]:

	purchase_ID	item_ID	quantity	purchase_date	customer_ID	shop_ID	loyalty_program	price_per_one	day_of_week_sale	day_of_sale	montl
0	538280	21873	11	2016-12-10 12:50:00	18427.0	Shop 0	False	1.63	5	2016-12-10	

Для создания пользовательских профилей напишем функцию `get_profiles()`. В ней сгруппируем значения датафрейма по пользовательскому ID и применим функцию `first()`:

In [78]:
```
def get_profiles(data_cl):

    # сортируем данные по ID пользователя и дате покупок
    # группируем по ID и находим первые значения purchase_date и shop_ID
    # столбец с датой и временем первой покупки назовём first_sale
    profiles = (
        data_cl.sort_values(by=['customer_ID', 'purchase_date'])
        .groupby('customer_ID')
        .agg({'purchase_date': 'first', 'shop_ID': 'first', 'loyalty_program':'first'})
        .rename(columns={'purchase_date': 'first_sale'})
        .reset_index()  # возвращаем customer_ID из индекса
    )
```

```
    # добавляем признак участия клиента в программе лояльности
    #profiles['loyalty_program'] = profiles['customer_ID'].isin(data_cl['customer_ID'].unique())

    # определяем дату первой покупки
    profiles['dt'] = profiles['first_sale'].dt.date
    # и первый день месяца, в который эта покупка произошла
    profiles['month'] = profiles['first_sale'].astype('datetime64[M]')
    profiles['day_of_month'] = profiles['first_sale'].dt.day


    return profiles
```

Вызовем функцию get_profiles(), чтобы составить профили пользователей из датафрейма data_cl:

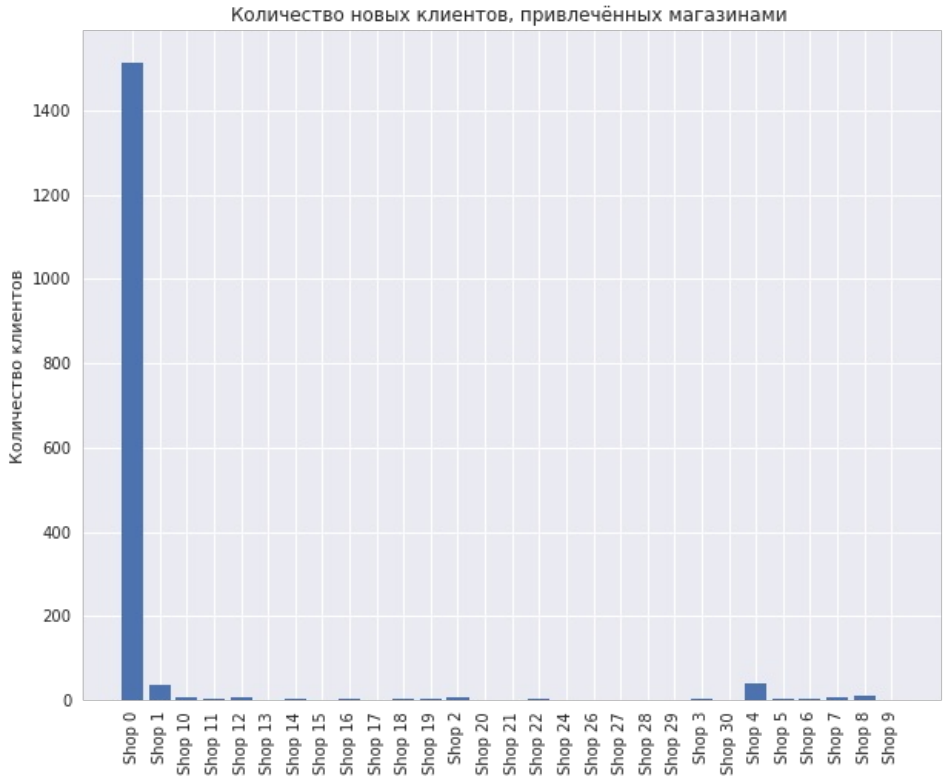
In [79]:
```
profiles = get_profiles(data_cl)
profiles.head(3)
```

Out[79]:

	customer_ID	first_sale	shop_ID	loyalty_program	dt	month	day_of_month
0	18025.0	2017-01-18 10:01:00	Shop 0	False	2017-01-18	2017-01-01	18
1	18026.0	2016-12-07 14:57:00	Shop 15	False	2016-12-07	2016-12-01	7
2	18027.0	2016-12-16 19:09:00	Shop 22	False	2016-12-16	2016-12-01	16

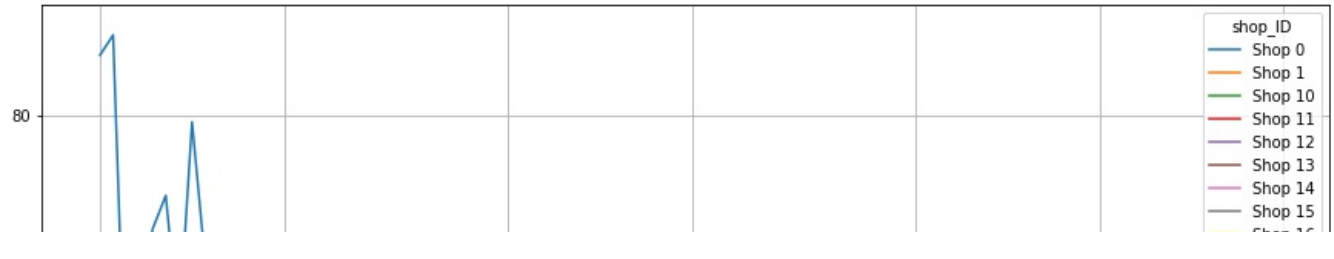
Имея готовые профили пользователей, легко узнать количество привлечённых каждым магазином за исследуемый период и результат посмотрим на графике:

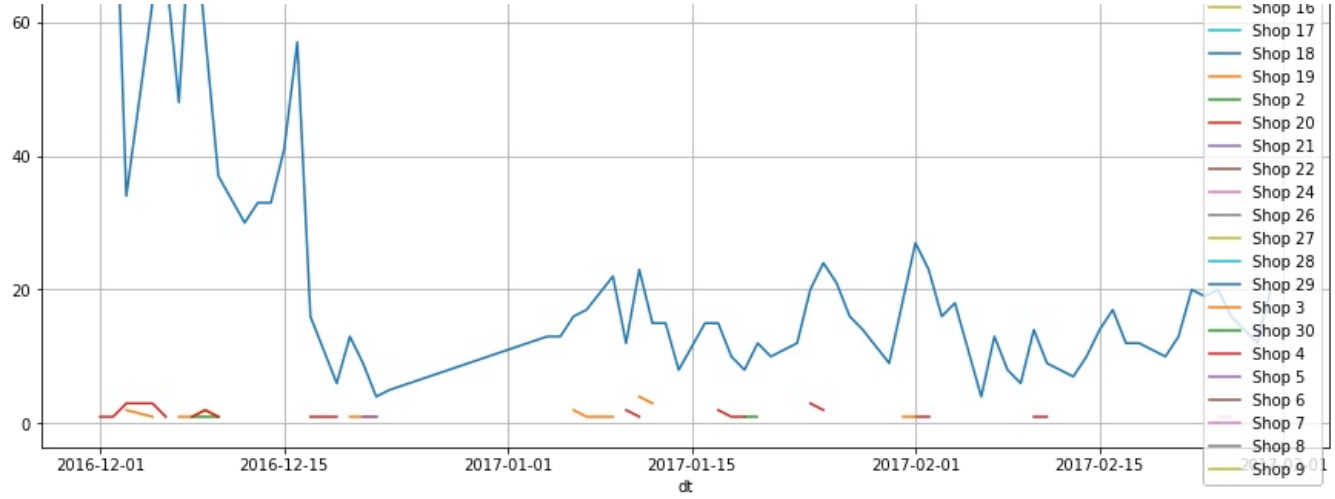
In [80]:
```
shop= profiles.groupby('shop_ID').agg({'customer_ID':'nunique'}).reset_index()
```



Промежуточные выводы: по графику видим, что абсолюным лидером по привлечению клиентов ялвяется магазин с идентификатором Shop 0.

А также можем посмотреть динамику привлечения покупателей за исследуемый период:





Промежуточные выводы: по графику видим, что у магазина с идентификатором Shop 0 на протяжении всего времени отличается регулярностью привлечения новых покупателей. Особенно успешным по параметру привлечения является для магазина первая половина декабря 2016г.

Разделим всех покупателей на три когорты по событию: совершившение первой покупку в декабре 2016, январе 2017 и феврале 2017. Дополнительный признак: участие в программе лояльности.

При этом участниками когорты будем считать тех клиентов, которые зарегистрировались с 1 по 7 число каждого месяца.

Таким образом каждый из клиентов после регистрации в начале месяца будет иметь период на покупки как минимум 21 день (т.к. в феврале 28 дней).

Out[83]:

	customer_ID	first_sale	shop_ID	loyalty_program	dt	month	day_of_month
1	18026.0	2016-12-07 14:57:00	Shop 15	False	2016-12-07	2016-12-01	7
3	18029.0	2017-02-02 16:01:00	Shop 5	False	2017-02-02	2017-02-01	2
12	18052.0	2017-02-01 13:10:00	Shop 20	False	2017-02-01	2017-02-01	1
18	18074.0	2016-12-03 16:35:00	Shop 12	False	2016-12-03	2016-12-01	3
21	18089.0	2017-02-04 10:38:00	Shop 7	False	2017-02-04	2017-02-01	4
...	...	...	...	...	...	...	...
1669	23905.0	2017-01-06 09:00:00	Shop 0	True	2017-01-06	2017-01-01	6
1670	23908.0	2016-12-01 16:25:00	Shop 0	True	2016-12-01	2016-12-01	1
1674	23918.0	2016-12-02 17:48:00	Shop 0	True	2016-12-02	2016-12-01	2
1676	23929.0	2017-02-02 13:16:00	Shop 0	True	2017-02-02	2017-02-01	2
1681	23962.0	2017-01-06 14:14:00	Shop 0	True	2017-01-06	2017-01-01	6

602 rows × 7 columns

Подготовим данные о покупках помесячно:

In [84]:
```
data_cl.head(1)
```

Out[84]:

	purchase_ID	item_ID	quantity	purchase_date	customer_ID	shop_ID	loyalty_program	price_per_one	day_of_week_sale	day_of_sale	month
0	538280	21873	11	2016-12-10 12:50:00	18427.0	Shop 0	False	1.63	5	2016-12-10	

Out[85]:

	customer_ID	month_year_sale	sum_month_sale
0	18025.0	2017-01-01	1.0
1	18026.0	2016-12-01	138.1
2	18026.0	2017-01-01	115.7
3	18027.0	2016-12-01	33.8
4	18027.0	2017-01-01	17.3

Выделим из таблицы sum_sale для каждого пользователя сумму покупок за первый месяц его обслуживания в магазине за исследуемый период:

Out[86]:

	customer_ID	month_year_sale	sum_first_month_sale
0	18025.0	2017-01-01	1.0
1	18026.0	2016-12-01	138.1
2	18027.0	2016-12-01	33.8
3	18029.0	2017-02-01	48.6
4	18031.0	2017-02-01	93.1

Добавим к таблице profiles_gorizont данные о покупках:

Out[87]:

	customer_ID	first_sale	shop_ID	loyalty_program	dt	month	day_of_month	month_year_sale	sum_first_month_sale
0	18026.0	2016-12-07 14:57:00	Shop 15	False	2016-12-07	2016-12-01	7	2016-12-01	138.1
1	18029.0	2017-02-02 16:01:00	Shop 5	False	2017-02-02	2017-02-01	2	2017-02-01	48.6
2	18052.0	2017-02-01 13:10:00	Shop 20	False	2017-02-01	2017-02-01	1	2017-02-01	57.8
3	18074.0	2016-12-03 16:35:00	Shop 12	False	2016-12-03	2016-12-01	3	2016-12-01	130.3
4	18089.0	2017-02-04 10:38:00	Shop 7	False	2017-02-04	2017-02-01	4	2017-02-01	178.9
...	...	...	...	...	...	...	...	...	...
597	23905.0	2017-01-06 09:00:00	Shop 0	True	2017-01-06	2017-01-01	6	2017-01-01	96.0
598	23908.0	2016-12-01 16:25:00	Shop 0	True	2016-12-01	2016-12-01	1	2016-12-01	103.3
599	23918.0	2016-12-02 17:48:00	Shop 0	True	2016-12-02	2016-12-01	2	2016-12-01	135.5
600	23929.0	2017-02-02 13:16:00	Shop 0	True	2017-02-02	2017-02-01	2	2017-02-01	108.3
601	23962.0	2017-01-06 14:14:00	Shop 0	True	2017-01-06	2017-01-01	6	2017-01-01	306.6

602 rows × 9 columns

Теперь рассчитаем:

- размер когорт (декабрьская, январская и февральская с разделением по признаку участия в программе лояльности)
- сумму покупок, сделанных в каждой кагорте

Out[88]:

	month	loyalty_program	cohor_size	sum_cohor_sale
0	2016-12-01	False	268	42479.0
1	2016-12-01	True	155	22993.3
2	2017-01-01	False	48	6475.9
3	2017-01-01	True	20	2605.4
4	2017-02-01	False	76	7854.0
5	2017-02-01	True	35	3388.5

Учтём тот факт, что для участия в программе лояльности клиентам необходимо оплатить 200 руб/мес:

In [89]:

```
def alert_program_category(loyalty_program):
    if loyalty_program == True:
        return 200
    return 0
#print(alert_program_category(True)) #выполним проверку работы функции
```

Теперь создадим доп. столбец, отражающий стоимость участия в программе и добавим итоговый столбец с суммарной суммой покупок с учётом дохода от продаж карт участия в программе лояльности:

Out[90]:

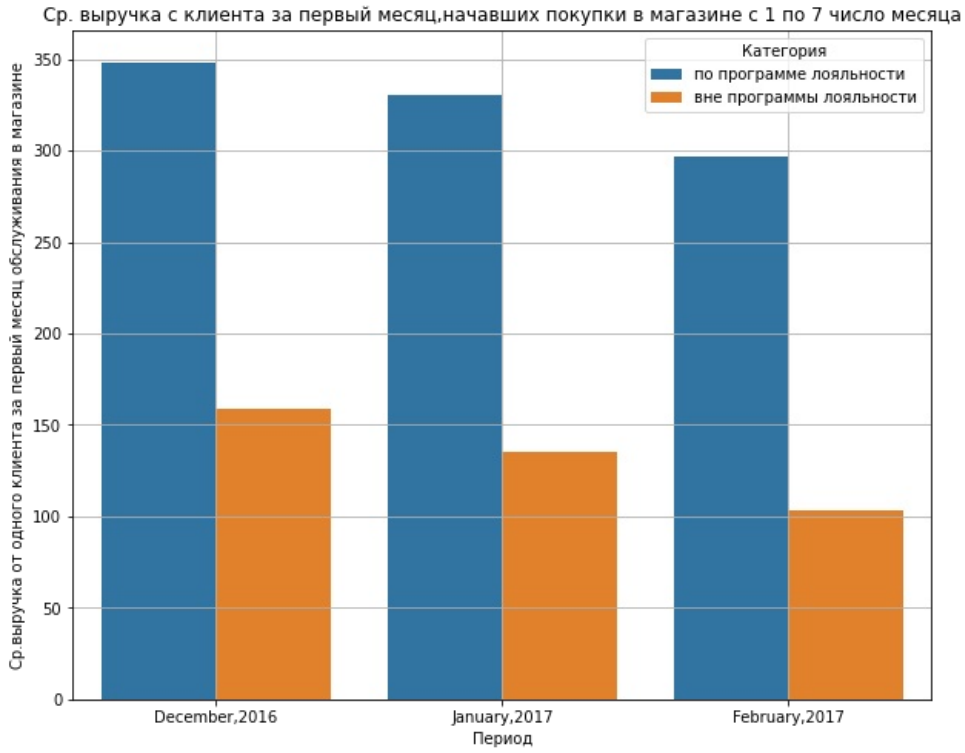
	month	loyalty_program	cohor_size	sum_cohor_sale	pay_program	total_cohor_sale
0	2016-12-01	False	268	42479.0	0	42479.0
1	2016-12-01	True	155	22993.3	31000	53993.3
2	2017-01-01	False	48	6475.9	0	6475.9
3	2017-01-01	True	20	2605.4	4000	6605.4
4	2017-02-01	False	76	7854.0	0	7854.0
5	2017-02-01	True	35	3388.5	7000	10388.5

Сделаем итоговый расчёт средней выручки от пользователя, начавших обслуживание в магазине с 1 по 7 число каждого месяца включительно у учётом участия в программе лояльности:

Out[91]:

	month	loyalty_program	cohor_size	sum_cohor_sale	pay_program	total_cohor_sale	ltv
0	December,2016	False	268	42479.0	0	42479.0	158.5
1	December,2016	True	155	22993.3	31000	53993.3	348.3
2	January,2017	False	48	6475.9	0	6475.9	134.9
3	January,2017	True	20	2605.4	4000	6605.4	330.3
4	February,2017	False	76	7854.0	0	7854.0	103.3
5	February,2017	True	35	3388.5	7000	10388.5	296.8

Отобразим результаты расчётов на графике:



Промежуточные выводы: по показателю ср. выручки на клиента видим, что участники программы лояльности за каждый месяц работы программы принесли магазину доход более чем в 2 раза выше, чем обычные клиенты.

Статистический анализ с формулировкой гипотез

Сформулируем и проверим первую гипотезу:

H0: НЕТ РАЗЛИЧИЙ в размере среднего чека между клиентами участвующими в программе лояльности и не участвующими в ней (т.е. размер среднего чека участников программы лояльности РАВЕН размеру среднего чека у покупателей, не участвующих в программе лояльности).

H1: ЕСТЬ РАЗЛИЧИЯ в размере среднего чека у участников программы лояльности и у остальных покупателей.

Для расчёта статистической значимости различий в среднем чеке между , передадим критерию mannwhitneyu() данные о стоимости покупок, а также найдём относительные различия в среднем чеке между группами клиентов, участвующих в программе лояльности и остальных покупателей:

In [94]:
```
print('p-value=','{0:.5f}'.format(stats.mannwhitneyu(cheki[cheki['loyalty_program']== 1]['purchase_price'], cheki

print('{0:.3f}'.format(cheki[cheki['loyalty_program']== 0]['purchase_price'].mean()/cheki[cheki['loyalty_program'
```

```
p-value= 0.98627
0.531
```

Выводы к расчётам:

p-value = 0.00019 меньше 0.05, значит нулевую гипотезу о том статистически значимых различий в среднем чеке покупок между группами нет - отвергаем.

Второе число 0,659 - говорит о том, средний чек у клиентов, не участвующих в программе лояльности выше среднего чека клиентов, участвующих в программе на 65,9%.

Сформулируем и проверим вторую гипотезу:

H0: НЕТ РАЗЛИЧИЙ среднем количестве покупок на одного покупателя между клиентами участвующими в программе лояльности и не участвующими в ней (т.е. среднее количество покупок на одного покупателя у участников программы лояльности РАВНО количеству покупок на одного покупателя у клиентов, не участвующих в программе лояльности).

H1: ЕСТЬ РАЗЛИЧИЯ в среднем количестве покупок на одного покупателя у участников программы лояльности и у остальных покупателей.

Для проверки гипотезы используем результаты ранее сделанных расчётов:

Среднее количество покупок у покупателей, учавствующих в программе лояльности:

Out[95]:

	month_year_sale	mean_number_da
0	2016-12-01	1.59
1	2017-01-01	1.26
2	2017-02-01	1.29

Среднее количество покупок у покупателей, НЕ учавствующих в программе лояльности:

Out[96]:

	month_year_sale	mean_number_net
0	2016-12-01	1.58
1	2017-01-01	1.38
2	2017-02-01	1.34

In [97]:
```
print('p-value=','{0:.3f}'.format(stats.mannwhitneyu(mean_number_sale_da['mean_number_da'], mean_number_sale_net[

print('{0:.3f}'.format(mean_number_sale_net['mean_number_net'].mean()/mean_number_sale_da['mean_number_da'].mean(
```

```
p-value= 0.700
0.039
```

Выводы к расчётам:

p-value = 0,400 больше 0.05, значит НЕ ОТВЕРГАЕМ нулевую гипотезу о том, что нет статистически значимых различий среднем количестве покупок на одного покупателя между клиентами участвующими в программе лояльности и не участвующими в ней.

Второе число 0,071 - говорит о том, что среднее количество покупок у клиентов, не участвующих в программе лояльности на 7,1% выше, чем у клиентов участвующих в программе лояльности, но это превышение в данном случае не является статистически значимым.

Дашборд

Дашборд

Сделаем выгрузку данных для дашборда:

In [98]:
```
cheki.to_csv('Vigruzka.csv')
```

Ссылка на дашборд: https://public.tableau.com/app/profile/karin.vink/viz/Final_project_16721149442770/Dashboard1?publish=yes

Выводы и рекомендации

По сделанным расчетам и графикам графикам видим:

1) По показателю «Средний чек» клиенты программы лояльности уступают обычным покупателям в 1,5 раза.

2) «Количество покупок всего» показывает нам, что уже почти 30% покупок сделано с использованием программы, при том, что программа находится в тестовом режиме.

3) Показатель количества покупок на одного клиента примерно одинаков для обоих групп клиентов, что положительно характеризует практику использования программы и возможности её дальнейшего использования, с учётом того, что в декабре 2016 у клиентов-участников этот показатель был выше.

4) Значимым аргументом в пользу дальнейшего применения программы является тот факт, что по показателю средней выручки с одного нового клиента за первый месяц покупок в магазине, клиенты – участники за каждый месяц работы программы принесли магазину доход более чем в 2 раза выше, чем обычные клиенты.

По графику дашборда видим:

Количество покупок в день в декабрь 2016 до середины месяца было выше, чем январе и феврале 2017г., что могло быть связано со стремлением людей завершить ремонтные и строительные работы в преддверии праздников. С 29 декабря 2016 г. по 3 января 2017г видим пробел в продажах. Он может быть связан как с техническими проблемами в записи данных, так и с графиком работы магазинов.

В первом случае - необходимо обратить на причины технических проблем для их избежания в дальнейшем.

Во втором случае - обратить внимание на график работы в конце и начале года в праздничные дни, т.к. для сети магазинов это может ежегодно быть причиной недополучения значительной прибыли.

Рекомендации по использованию программы:

По сделанным расчётам есть все основания полагать, что у программы лояльности положительная динамика использования и есть возможности для её расширения и увеличения основных показателей (среднего чека, количества покупок и средней выручки на одного клиента).

1) Для возможной модернизации и развития программы желательно провести дополнительный опрос, что можно улучшить в программе у клиентов её использующих и что останавливает от участия тех клиентов , кто программой ещё не пользовался.

2) Особое внимание необходимо обратить на показатель среднего чека с целью его увеличения, например, включить в программу более дорогостоящие товарные позиции, а также бонусы за покупку нескольких товаров за раз.

Рекомендации для коммерческого отдела:

1) Необходимо масштабировать опыт работы магазина “Shop 0”, как лидера по привлечению новых клиентов, для использования его успешной практики во всей сети.

2) Необходимо обратить внимание на режим работы магазинов последние дни перед новым годом и первые дни нового года, что поможет увеличить доход сети магазинов в аналогичный период следующего года.

Презентация

В презентации представлены результаты исследования, описание функционала дашборда, а также выводы ко всему исследованию и рекомендации для менеджера проекта программы лояльности и коллег из коммерческого департамента:

Ссылка на презентацию: https://cloud.mail.ru/public/oPgC/oQA5eq8mc