"АНАЛИЗ БИЗНЕС ПОКАЗАТЕЛЕЙ"

Заказчик: Яндекс.Афиша

Цель:

На основании данных с июня 2017 по конец мая 2018 года выявить невыгодные источники трафика и перспективные когорты клиентов для компании для снижения оптимизации маркетинговых расходов и перераспределения бюджета. Для анализа предоставлены данные с датой посещения клиентов и источников перехода на сервис, данные с временем совершения покупки и суммой заказа и данные о времени прохождения рекламных кампаний с ситочнкиками перехода и затраты на них.

Ввод [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

Первоначальное изучение данных:

Изучение информации о визитах:

Ввод [2]:
```
visits = pd.read_csv('D:/АНАЛИТИКА/_5 Анализ бизнес показателей/visits_log.csv')
print('Данные таблицы с визитами:')
display(visits.head())
print()
visits.info()
```

```
Данные таблицы с визитами:
```

	Device	End Ts	Source Id	Start Ts	Uid
0	touch	2017-12-20 17:38:00	4	2017-12-20 17:20:00	16879256277535980062
1	desktop	2018-02-19 17:21:00	2	2018-02-19 16:53:00	104060357244891740
2	touch	2017-07-01 01:54:00	5	2017-07-01 01:54:00	7459035603376831527
3	desktop	2018-05-20 11:23:00	9	2018-05-20 10:59:00	16174680259334210214
4	desktop	2017-12-27 14:06:00	3	2017-12-27 14:06:00	9969694820036681168

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 359400 entries, 0 to 359399
Data columns (total 5 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   Device     359400 non-null  object
 1   End Ts     359400 non-null  object
 2   Source Id  359400 non-null  int64 
 3   Start Ts   359400 non-null  object
 4   Uid        359400 non-null  uint64
dtypes: int64(1), object(3), uint64(1)
memory usage: 13.7+ MB
```

Выводы по подготовке данных по визитам:

- **пропусков в данных нет,**
- **изменить тип данных на корректный в столбцах с датой и временем начала и окончания сессии,**
- **переименоавть названия столбцов датафрейма.**

Изучение информации о заказах:

Ввод [3]:
```
orders = pd.read_csv('D:/АНАЛИТИКА/_5 Анализ бизнес показателей/orders_log.csv')
print('Данные таблицы с заказами:')
display(orders.head())
print()
orders.info()
```

```
Данные таблицы с заказами:
```

	Buy Ts	Revenue	Uid
0	2017-06-01 00:10:00	17.00	10329302124590727494
1	2017-06-01 00:25:00	0.55	11627257723692907447
2	2017-06-01 00:27:00	0.37	17903680561304213844
3	2017-06-01 00:29:00	0.55	16109239769442553005
4	2017-06-01 07:58:00	0.37	14200605875248379450

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50415 entries, 0 to 50414
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Buy Ts   50415 non-null  object 
 1   Revenue  50415 non-null  float64
 2   Uid      50415 non-null  uint64 
dtypes: float64(1), object(1), uint64(1)
memory usage: 1.2+ MB
```

Выводы по подготовке данных о заказах:

- **пропусков в данных нет,**
- **изменить тип данных на корректный в столбце с датой и временем заказа,**
- **переименоавть названия столбцов датафрейма.**

Изучение информации о рекламных расходах:

Ввод [4]:
```
costs = pd.read_csv('D:/АНАЛИТИКА/_5 Анализ бизнес показателей/costs.csv')
print('Данные таблицы с расходами:')
display(costs.head())
print()
costs.info()
```

```
Данные таблицы с расходами:
```

	source_id	dt	costs
0	1	2017-06-01	75.20
1	1	2017-06-02	62.25
2	1	2017-06-03	36.53
3	1	2017-06-04	55.00
4	1	2017-06-05	57.08

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2542 entries, 0 to 2541
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   source_id  2542 non-null   int64  
 1   dt         2542 non-null   object 
 2   costs      2542 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 59.7+ KB
```

Выводы по подготовке данных о рекламных расходах:

- **пропусков в данных нет,**
- **изменить тип данных на корректный в столбце с датой проведения рекламной кампании.**

Обработка данных:

Визиты - переименование столбцов и типов данных:

Ввод [5]:
```
#переименование столбцов:
visits.set_axis(['device', 'end_ts', 'source_id', 'start_ts', 'uid'], axis = 'columns', inplace = Tru

# приведение данных с датой и временем к корр. типу:
visits['end_ts']=pd.to_datetime(visits['end_ts'], format='%Y-%m-%d %H:%M:%S')
visits['start_ts']=pd.to_datetime(visits['start_ts'], format='%Y-%m-%d %H:%M:%S')


print('Проверка изменения названия столбцов и типов данных:')
print()
visits.info()
```

```
Проверка изменения названия столбцов и типов данных:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 359400 entries, 0 to 359399
Data columns (total 5 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   device     359400 non-null  object
 1   end_ts     359400 non-null  datetime64[ns]
 2   source_id  359400 non-null  int64
 3   start_ts   359400 non-null  datetime64[ns]
 4   uid        359400 non-null  uint64
dtypes: datetime64[ns](2), int64(1), object(1), uint64(1)
memory usage: 13.7+ MB
```

Заказы - переименование столбцов и изменение типов данных:

Ввод [6]:
```
#переименование столбцов:
orders.set_axis(['buy_ts', 'revenue', 'uid'], axis = 'columns', inplace = True)

# приведение данных с датой и временем к корр. типу:
orders['buy_ts']=pd.to_datetime(orders['buy_ts'], format='%Y-%m-%d %H:%M:%S')

print('Проверка изменения названия столбцов и типов данных:')
print()
orders.info()
```

```
Проверка изменения названия столбцов и типов данных:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50415 entries, 0 to 50414
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   buy_ts   50415 non-null  datetime64[ns]
 1   revenue  50415 non-null  float64
 2   uid      50415 non-null  uint64
dtypes: datetime64[ns](1), float64(1), uint64(1)
memory usage: 1.2 MB
```

Расходы на рекламу - изменение типов данных:

Ввод [7]:
```
# приведение данных с датой и временем к корр. типу:
costs['dt']=pd.to_datetime(costs['dt'], format='%Y-%m-%d')

print('Проверка изменения типов данных:')
print()
costs.info()
```

```
Проверка изменения типов данных:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2542 entries, 0 to 2541
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   source_id  2542 non-null   int64
 1   dt         2542 non-null   datetime64[ns]
 2   costs      2542 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 59.7 KB
```

Вывод по предобработке данных:

- во всех таблицах с данными в столбцы с датами были приведены к корректному типу данных,
- в таблицах с визитами и заказами, названия столбцов были переименованы.

Продуктовые метрики:

DAU, WAU, MAU:

Оценка пользовательской активности. Получение данных с уникальными пользователями в день, неделю, месяц.

Работа с данными таблицы посещений.

Ввод [8]:
```
display(visits.head())
```

	device	end_ts	source_id	start_ts	uid
0	touch	2017-12-20 17:38:00	4	2017-12-20 17:20:00	16879256277535980062
1	desktop	2018-02-19 17:21:00	2	2018-02-19 16:53:00	104060357244891740
2	touch	2017-07-01 01:54:00	5	2017-07-01 01:54:00	7459035603376831527
3	desktop	2018-05-20 11:23:00	9	2018-05-20 10:59:00	16174680259334210214
4	desktop	2017-12-27 14:06:00	3	2017-12-27 14:06:00	9969694820036681168

Выделение столбцов с годом, месяцем, неделей и полной датой.

Ввод [9]:
```
visits['session_year']=visits['start_ts'].dt.year
visits['session_month']=visits['start_ts'].dt.month
visits['session_week']=visits['start_ts'].dt.isocalendar().week
visits['session_date']=visits['start_ts'].dt.date
display(visits.head())
```

	device	end_ts	source_id	start_ts	uid	session_year	session_month	session_week	session_date
0	touch	2017-12-20 17:38:00	4	2017-12-20 17:20:00	16879256277535980062	2017	12	51	2017-12-20
1	desktop	2018-02-19 17:21:00	2	2018-02-19 16:53:00	104060357244891740	2018	2	8	2018-02-19
2	touch	2017-07-01 01:54:00	5	2017-07-01 01:54:00	7459035603376831527	2017	7	26	2017-07-01
3	desktop	2018-05-20 11:23:00	9	2018-05-20 10:59:00	16174680259334210214	2018	5	20	2018-05-20
4	desktop	2017-12-27 14:06:00	3	2017-12-27 14:06:00	9969694820036681168	2017	12	52	2017-12-27

Получение значений DAU, WAU, MAU

Ввод [10]:
```
# определение количества уникальных пользоваителей в день
dau_total = visits.groupby('session_date').agg({'uid':'nunique'}).mean().astype(int)
print('Количество уникальных пользователей в день:')
print(dau_total)
print()

# определение количества уникальных пользоваителей в неделю
wau_total = (
    visits.groupby(['session_year', 'session_week'])
    .agg({'uid': 'nunique'})
    .mean()
).astype(int)
print('Количество уникальных пользователей в неделю:')
print(wau_total)
print()

# определение количества уникальных пользоваителей в месяц
mau_total = (
    visits.groupby(['session_year', 'session_month'])
    .agg({'uid': 'nunique'})
    .mean()
).astype(int)
print('Количество уникальных пользователей в месяц:')
print(mau_total)
print()

# определение регулярности использования Яндекс.Афиша
# в неделю
sticky_wau = dau_total / wau_total*100
print('Значение еженедельного использования Яндекс.Афиша пользователями:')
print(sticky_wau)
print()

# определение регулярности пользования Яндекс.Афиша
# в месяц
sticky_mau = dau_total / mau_total*100
print('Значение ежемесячного использования Яндекс.Афиша пользователями:')
print(sticky_mau)
print()
```

```
Количество уникальных пользователей в день:
uid    907
dtype: int32

Количество уникальных пользователей в неделю:
uid    5716
dtype: int32

Количество уникальных пользователей в месяц:
uid    23228
dtype: int32

Значение еженедельного использования Яндекс.Афиша пользователями:
uid    15.86774
dtype: float64

Значение ежемесячного использования Яндекс.Афиша пользователями:
uid    3.90477
dtype: float64
```

График изменения DAU во времени

Ввод [11]:
```
dau_group = visits.groupby('session_date').agg({'uid':'nunique'})
dau_group.plot(grid=True, figsize = (15,5))
plt.title('График изменения количества уникальных пользователей в день:')
plt.xlabel('Время сессии')
plt.ylabel('Количество пользователей')

plt.show()
```

Резкий всплеск в количестве уникальных клиентов пришёлся наконец ноября 2017 года, предположительно, это реакция на рекламную кампанию по привлечению новых клиентов, обусловленная предостоящими новогодними каникулами. Также на графике заметен резкий спад в появлении уникальных пользователейна границе марта и апреля 2017. Учитывая, что падение остановилось на нулевой отметке, предположительно произошел сбой в работе сайта.

График изменения WAU во времени

Ввод [12]:
```
wau_group = visits.groupby(['session_year', 'session_week']).agg({'uid': 'nunique'})
wau_group.plot(grid=True, figsize = (15,5))
plt.title('График изменения количества уникальных пользователей в неделю:')
plt.xlabel('Дата сессии')
plt.ylabel('Количество пользователей')

plt.show()
```

График показывает стабильный рост количества уникальных клиентов в неделю, с августа 2017 года до

конца ноября, затем спад появления уникальных пользователей и значения держатся в течение первых трех месяцев 2018 года на стабильном уровне, после мартовского всплеска, происходит спад с дальнейшим снижением числа уникальных клиентов сервиса. Появление новых клиентов всегда связано с кампанией по их привлечению, рост появления новых клиентов однозначно указывает на эффективность проводимых кампаний, спад возможен либо с прекращением кампании, либо с перераспределением денежных средств, либо с корректировкой выбора источника доставки информации до целевой группы.

График изменения MAU во времени

Ввод [13]:
```
mau_group = visits.groupby(['session_year', 'session_month']).agg({'uid': 'nunique'})
mau_group.plot(grid=True, figsize = (15,5))
plt.title('График изменения количества уникальных пользователей в месяц:')
plt.xlabel('Дата сессии')
plt.ylabel('Количество пользователей')

plt.show()
```

На графике измиенений появления ежемесячного количества уникальных клиентов, также заметен рост с августа 2017 года до ноября, после чего происходит снижение к январю 2018 года и резкий спад в марте 2018. Рекламная кампания не окупается, или на неёё просто не хватило средств?

Вывод по значениям DAU, WAU, MAU: данные по постоянному притоку новых активных пользователей говорят о том, что проводится кампания по привлечению новых посетителей, и предпринимаемые маркетинговые ходы приносят результат.

Пользовательские сессии:

Определение среднего количества посещений сайта пользователями в день:

Таблица с количеством ежедневных посещений Яндекс.Афиши пользователями.

Ввод [14]:
```
session_per_user_day = visits.groupby('session_date').agg({'uid':['count','nunique']})
session_per_user_day.columns = ['n_sess', 'n_user']
session_per_user_day['sess_per_user_day'] = session_per_user_day['n_sess'] / session_per_user_day['n_
display(session_per_user_day.head())
```

session_date	n_sess	n_user	sess_per_user_day
2017-06-01	664	605	1.097521
2017-06-02	658	608	1.082237
2017-06-03	477	445	1.071910
2017-06-04	510	476	1.071429
2017-06-05	893	820	1.089024

График изменений ежедневных посещений Яндекс.Афиши:

Ввод [15]:
```
session_per_user_day['sess_per_user_day'].plot(grid=True, figsize = (15,5))
plt.title('График изменений ежедневных посещений:')
plt.xlabel('Дата посещения')
plt.ylabel('Количество посещений')

plt.show()
```

В среднем пользователь бывает на сайтеодин раз в день в конце ноября 2017 наблюдался небольшое увеличение пользовательской активности в посещении, но незначительное от 1.05 в среднем к 1.2

Средняя продолжительность пользовательской сессии:

Построение распределения продолжительности пользовательской сессии.

Ввод [16]:
```
visits['session_duration_sec'] = (visits['end_ts'] - visits['start_ts']).dt.seconds
visits['session_duration_sec'].hist(bins=100, range=(0, 10000))
plt.title('График распределения пользовательской сессии:')
plt.show()
```

Определение средней продолжительности сессии. Тремя показателями:

Ввод [17]:
```
asl_mode = visits['session_duration_sec'].mode()
asl_mean = visits['session_duration_sec'].mean()
asl_median = visits['session_duration_sec'].median()
print('Модальная продолжительность типичной пользовательской сессии, сек:')
print(asl_mode)
print('Средняя продолжительность типичной пользовательской сессии, сек:')
print(asl_mean)
print('Медианная продолжительность типичной пользовательской сессии, сек:')
print(asl_median)
```

```
Модальная продолжительность типичной пользовательской сессии, сек:
0    60
dtype: int64
Средняя продолжительность типичной пользовательской сессии, сек:
643.506488592098
Медианная продолжительность типичной пользовательской сессии, сек:
300.0
```

Изучение распределиния количества сессий продолжительностью 60 секунд по типам устройств:

Ввод [18]:
```
device_group_touch = visits[(visits['device'] == 'touch') & (visits['session_duration_sec'] == 60)]
device_group_desktop = visits[(visits['device'] == 'desktop') & (visits['session_duration_sec'] == 60
print('Количество сессий продолжительносью 60 секунд для устройства touch')
print(device_group_touch['session_duration_sec'].count())
print()
print('Количество сессий продолжительносью 60 секунд для устройства desktop')
print(device_group_desktop['session_duration_sec'].count())
```

```
Количество сессий продолжительносью 60 секунд для устройства touch
17618

Количество сессий продолжительносью 60 секунд для устройства desktop
35411
```

Среднее значение скошено вправо относительно медианы и почти вдвое превышает медианное значение. Что обусловлено наличием околонулевых значений, которые хорошо видны на гистограмме, а также выбросами по длительности в диапазоне от 2000 до 5000 секунд. Возможно именно именно эта продолжительность является моментом покупки. В любом случае продолжительность пользовательской сессии в пять или в 10 минут в среднем говорит о том, что пользователь успевает ознакомиться с продуктом, по рекламе которого он перешёл на сервис Яндекс.Афиша, а наиболее часто встречающееся значение в 60 секунд, подтверждает, что реклама по привлечению клиентов работает, возможно стоит обратить внимание на продуктовую линейку, за 60 секунд пользователь понимает, что ему данное предложение всё-таки неинтересно. Или Данный пользователь не является представителем целевой аудитории, например, его финансовые возможности недостаточны для покупки.Также необходимо проверить быстроту и корректность работы сайта, при анализе сессий в 60 секунд относительно устройств можно отрицать, что у пользователей устройств стоит ограничение по продолжительности интернет-сессии. У настольных ПК количество 60-секундных сессий - 35411, хотя данные устройства как правило имеют бездимитный интернет.

Определение коэффициента удержания - Retention Rate:

Создание таблицы со столбцом с данными о первом посещении Яндекс.Афиши.

Ввод [19]:
```
#определение первого посещения:
first_visit_date=visits.groupby('uid')['start_ts'].min()
first_visit_date.name = 'first_visit_date'
# добавление столбца с первым посещением в таблицу посещений:
visits = visits.join(first_visit_date, on = 'uid')
visits['start_ts_month']=visits['start_ts'].astype('datetime64[M]')
visits['first_visit_month']=visits['first_visit_date'].astype('datetime64[M]')
display(visits.head())
```

	device	end_ts	source_id	start_ts	uid	session_year	session_month	session_week	session_date
0	touch	2017-12-20 17:38:00	4	2017-12-20 17:20:00	16879256277535980062	2017	12	51	2017-12-20
1	desktop	2018-02-19 17:21:00	2	2018-02-19 16:53:00	104060357244891740	2018	2	8	2018-02-19
2	touch	2017-07-01 01:54:00	5	2017-07-01 01:54:00	7459035603376831527	2017	7	26	2017-07-01
3	desktop	2018-05-20 11:23:00	9	2018-05-20 10:59:00	16174680259334210214	2018	5	20	2018-05-20
4	desktop	2017-12-27 14:06:00	3	2017-12-27 14:06:00	9969694820036681168	2017	12	52	2017-12-27

Выделение когорты и определение коэффициента удержания

В качестве события определим посещения пользователями Яндекс.Афиши ежемесячно.

Ввод [20]:
```
# определение времени жизни когорты
visits['cohort_lifetime']= visits['start_ts_month'] - visits['first_visit_month']
visits['cohort_lifetime'] = visits['cohort_lifetime'] / np.timedelta64(1,'M')
visits['cohort_lifetime'] = visits['cohort_lifetime'].round().astype(int)

# создание когорты
cohorts = visits.groupby(['first_visit_month','cohort_lifetime']).agg({'uid': 'nunique'}).reset_index
initial_visits_count = cohorts[cohorts['cohort_lifetime'] == 0][['first_visit_month', 'uid']]
initial_visits_count = initial_visits_count.rename(columns={'uid':'cohort_users'})
cohorts = cohorts.merge(initial_visits_count, on='first_visit_month')

print(cohorts.head())
```

```
  first_visit_month  cohort_lifetime    uid  cohort_users
0        2017-06-01                0  13259         13259
1        2017-06-01                1   1043         13259
2        2017-06-01                2    713         13259
3        2017-06-01                3    814         13259
4        2017-06-01                4    909         13259
```

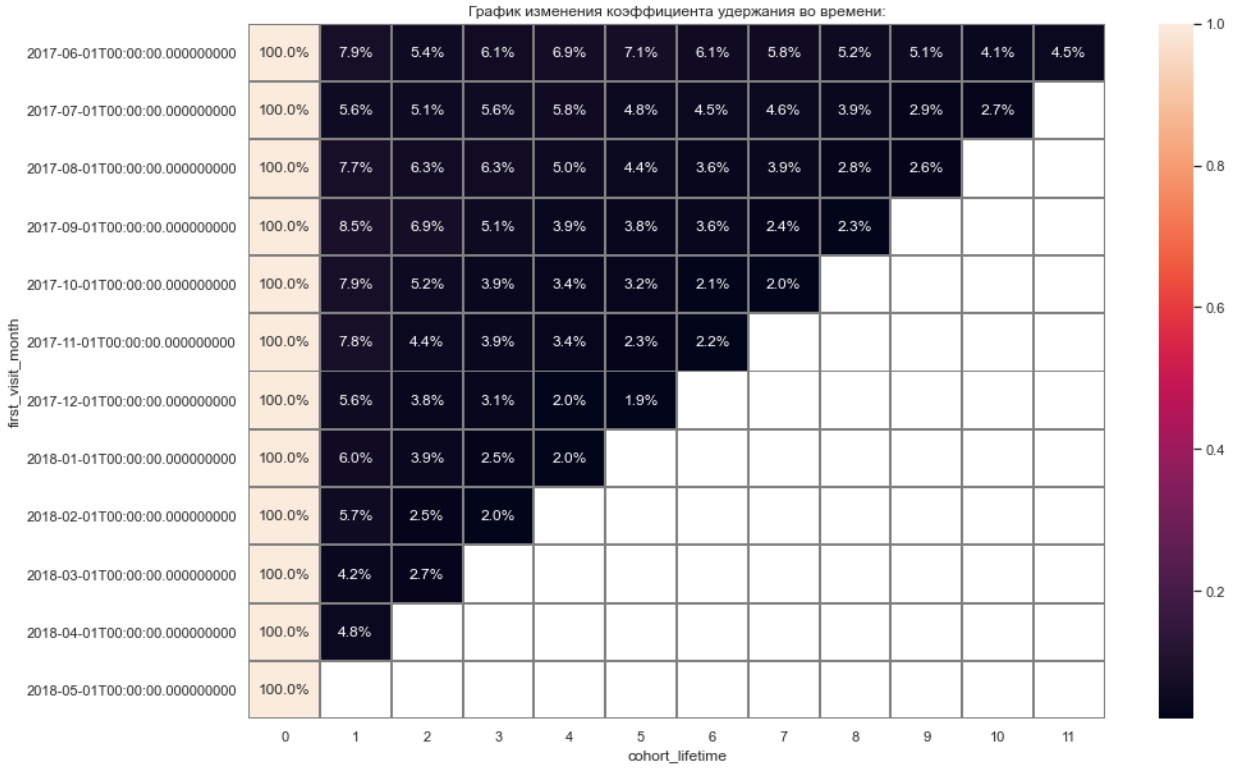
Определение коэффициента удержания и сводная таблица изменения коэффициента удержания:

Ввод [21]:
```
# определение коэффициента удержания:
cohorts['retention'] = cohorts['uid']/cohorts['cohort_users']

# сводная таблица:
retention_pivot = cohorts.pivot_table(
    index='first_visit_month',
    columns='cohort_lifetime',
    values='retention',
    aggfunc='sum')
display(retention_pivot.fillna(''))
```

cohort_lifetime	0	1	2	3	4	5	6	7	8	9	10
first_visit_month											
2017-06-01	1.0	0.078664	0.053775	0.061392	0.068557	0.071423	0.061015	0.057772	0.052342	0.050833	0.040652
2017-07-01	1.0	0.056088	0.051294	0.056164	0.058219	0.048174	0.045358	0.045738	0.038813	0.028615	0.027473
2017-08-01	1.0	0.076908	0.062862	0.062764	0.050093	0.044004	0.036342	0.039485	0.027895	0.026029	
2017-09-01	1.0	0.085489	0.069205	0.050706	0.039392	0.037835	0.03586	0.024186	0.022809		
2017-10-01	1.0	0.078608	0.052239	0.038958	0.034261	0.032221	0.021365	0.020364			
2017-11-01	1.0	0.078281	0.044113	0.038682	0.033727	0.023415	0.0218				
2017-12-01	1.0	0.055802	0.037993	0.031107	0.020263	0.019036					
2018-01-01	1.0	0.059715	0.039339	0.024973	0.020244						
2018-02-01	1.0	0.05708	0.025454	0.020093							
2018-03-01	1.0	0.041818	0.027053								
2018-04-01	1.0	0.04838									
2018-05-01	1.0										

Тепловая карта изменения коэффициента удержания во времени:

Ввод [22]:
```
sns.set(style='white')
plt.figure(figsize=(15,10))
plt.title('График изменения коэффициента удержания во времени:')
sns.heatmap(retention_pivot,annot=True, fmt='.1%', linewidths=1, linecolor='gray')
plt.show()
```



Расчёт среднего значения Retention Rate на второй месяц существования когорты:

Ввод [23]:
```
# срез данных для второго месяца "жизни" когорт:
initial_count_2 = cohorts[cohorts['cohort_lifetime'] == 2][['first_visit_month', 'uid']]
initial_count_2 = initial_count_2.rename(columns={'uid':'cohort_users_2'})
cohorts_2 = cohorts.merge(initial_count_2, on='first_visit_month')
cohorts_2['retention'] = cohorts_2['uid']/cohorts_2['cohort_users_2']

# среднее значение коэффициента удержания на второй месяц жизни коггорт:
cohorts_2_mean = cohorts_2['retention'].mean()
print('Среднее значение Retention Rate на второй месяц lifetime:')
print(cohorts_2_mean)
```

```
Среднее значение Retention Rate на второй месяц lifetime:
3.9528350530343603
```

Среднее значение коффициента удержания всего 4% для второго месяца жизни когорты Клиенты не спешат переходить в разряд постоянных.

В целом тенденция изменения коэффициента удержания - снижение с течением времени жизни когорт

Метрики электронной коммерции:

Средняя продолжительность принятия решения о покупке:

Ввод [24]:
```
# создание датафрейма с временем визита и временем покупки
visits_orders = visits.merge(orders, on='uid')

# столбец с данными промежутка времени между первым посещением и покупкой
visits_orders['session_order_sec'] = (visits_orders['buy_ts'] - visits_orders['start_ts']).dt.seconds

# определение средного времени между первым посещением и покупкой
visits_orders_mode = visits_orders['session_order_sec'].mode()
visits_orders_mean = visits_orders['session_order_sec'].mean()
visits_orders_median = visits_orders['session_order_sec'].median()
print('Модальное значение между первым посещением сайта и совершением покупки в секундах:')
print(visits_orders_mode)
print('Среднее значение между первым посещением сайта и совершением покупки в секундах:')
print(visits_orders_mean)
print('Медианное значение между первым посещением сайта и совершением покупки в секундах:')
print(visits_orders_median)

# определение количества посетителей покупающих за 60 секЖ
print('Количество посетителей, принимающих решение о покупке в течение 60 секунд:')
print(len(visits_orders[visits_orders['session_order_sec'] == 60]))

#определение количества посетителей, со средней сессией 60сек:
print('Количество посетителей, со средней сессией 60 секунд:')
print(len(visits[visits['session_duration_sec'] == 60]))

display(visits_orders['session_order_sec'].describe())
```

```
Модальное значение между первым посещением сайта и совершением покупки в секундах:
0    60
dtype: int64
Среднее значение между первым посещением сайта и совершением покупки в секундах:
40268.81782393703
Медианное значение между первым посещением сайта и совершением покупки в секундах:
31620.0
Количество посетителей, принимающих решение о покупке в течение 60 секунд:
5675
Количество посетителей, со средней сессией 60 секунд:
53029

count    761807.000000
mean      40268.817824
std       31619.473060
min           0.000000
25%        9120.000000
50%       31620.000000
75%       73380.000000
max       86340.000000
Name: session_order_sec, dtype: float64
```

Из средних значений периода времени, который необходим посетителям для принятия решения о покупке, можно сделать следующий вывод: из 762 тыс. случаев совершения покупки всего 5,7 тысяч покупок состоялись в течение 60 секунд, при этом количество сессий продолжительностью 60 секунд составляет 53тыс., около 10% принимают решение купить, остальные 90% покидают сайт. Возможен промах с целевой рекламой. В среднем же посетителю на принятие решения требуется почти 9 часов с момента первого посещения. Учитывая, что средняя сессия 5 минут, которых достаточно, чтобы разобраться с предложением, что-то не так с предложением или присутствуют технические проблемы с оформлением покупки на сайте.

Среднее количество покупок на одного покупателя:

Определение среднего количества покупок за период

Ввод [25]:
```
# расчёт количества покупателей
orders['buy_ts_year'] = orders['buy_ts'].dt.year
orders_2017= orders[orders['buy_ts_year'] == 2017]
orders_buyer_mean = orders_2017.groupby('uid')['buy_ts'].count().mean()

print(orders_buyer_mean)
```

```
1.326844113810927
```

В качестве периода для нахождения среднего количества покупок был выбран 2017 год. В 2017 году каждый покупатель совершал чуть меньше полутора покупок.

Определение среднего количества покупок за период с применением когортного анализа.

Ввод [26]:
```
display(orders_2017.head())
```

	buy_ts	revenue	uid	buy_ts_year
0	2017-06-01 00:10:00	17.00	10329302124590727494	2017
1	2017-06-01 00:25:00	0.55	11627257723692907447	2017
2	2017-06-01 00:27:00	0.37	17903680561304213844	2017
3	2017-06-01 00:29:00	0.55	16109239769442553005	2017
4	2017-06-01 07:58:00	0.37	14200605875248379450	2017

Создание таблицы с месяцем первой покупки для последующего создания когорты:

Ввод [27]:
```
# определение даты первой покупки
first_order_date_2017=orders_2017.groupby('uid').agg({'buy_ts': 'min'}).reset_index()
first_order_date_2017.columns=['uid','first_order_date_2017']

# объединение датафреймов
orders_2017=orders_2017.merge(first_order_date_2017, on='uid')

# получение столбцов с данными по месяцу первой покупки и месяцев последующих покупок
orders_2017['first_order_2017_month'] = orders_2017['first_order_date_2017'].astype('datetime64[M]')
orders_2017['buy_ts_month'] = orders_2017['buy_ts'].astype('datetime64[M]')
display(orders_2017.head())
```

	buy_ts	revenue	uid	buy_ts_year	first_order_date_2017	first_order_2017_month	buy_ts_month
0	2017-06-01 00:10:00	17.00	10329302124590727494	2017	2017-06-01 00:10:00	2017-06-01	2017-06-01
1	2017-06-01 00:25:00	0.55	11627257723692907447	2017	2017-06-01 00:25:00	2017-06-01	2017-06-01
2	2017-06-01 00:27:00	0.37	17903680561304213844	2017	2017-06-01 00:27:00	2017-06-01	2017-06-01
3	2017-06-01 00:29:00	0.55	16109239769442553005	2017	2017-06-01 00:29:00	2017-06-01	2017-06-01
4	2017-06-01 07:58:00	0.37	14200605875248379450	2017	2017-06-01 07:58:00	2017-06-01	2017-06-01

Создание когорты:

Ввод [28]:
```
# создание когорты
cohorts_2017 = orders_2017.groupby(
    ['first_order_2017_month', 'buy_ts_month']).agg({'buy_ts':'count','uid': 'nunique'}).reset_index(
cohorts_2017 = cohorts_2017.rename(columns={'buy_ts':'count_purshase'})

# разделим количество покупок на количество уникальных покупателей:
cohorts_2017['count_per_buyer'] = cohorts_2017['count_purshase']/cohorts_2017['uid']

# добавление времени жизни когорты
cohorts_2017['cohort_lifetime']= cohorts_2017['buy_ts_month'] - cohorts_2017['first_order_2017_month'
cohorts_2017['cohort_lifetime'] = cohorts_2017['cohort_lifetime'] / np.timedelta64(1,'M')
cohorts_2017['cohort_lifetime'] = cohorts_2017['cohort_lifetime'].round().astype(int)

# сводная таблица:
count_order_pivot = cohorts_2017.pivot_table(
    index='first_order_2017_month',
    columns='cohort_lifetime',
    values='count_per_buyer',
    aggfunc='mean')
display(count_order_pivot.fillna(''))
print(count_order_pivot.mean().mean())
```

cohort_lifetime	0	1	2	3	4	5	6
first_order_2017_month							
2017-06-01	1.163618	2.901639	3.48	4.185185	3.318182	3.253731	4.451613
2017-07-01	1.136765	1.923077	2.105263	1.625	1.469388	2.157895	
2017-08-01	1.118978	1.862069	1.886792	1.840909	2.125		
2017-09-01	1.136381	1.684615	1.61	2.216216			
2017-10-01	1.143779	1.524272	1.317073				
2017-11-01	1.179368	1.788288					
2017-12-01	1.152635						

```
2.4432795265151728
```

В таблице с когортным анализом среднего числа покупок на покупателя со второго месяца жизни когорты происходит рост этого показателя для всех когорт. Затем показатели меняются с каждым месяцем, рост сменяется снижением, наиболее успешной выглядит первая когорта

Средний чек покупки:

Построение графика изменений среднего чека во времени

Ввод [29]:
```
display(orders)
```

	buy_ts	revenue	uid	buy_ts_year
0	2017-06-01 00:10:00	17.00	10329302124590727494	2017
1	2017-06-01 00:25:00	0.55	11627257723692907447	2017
2	2017-06-01 00:27:00	0.37	17903680561304213844	2017
3	2017-06-01 00:29:00	0.55	16109239769442553005	2017
4	2017-06-01 07:58:00	0.37	14200605875248379450	2017
...	...	...	...	...
50410	2018-05-31 23:50:00	4.64	12296626599487328624	2018
50411	2018-05-31 23:50:00	5.80	11369640365507475976	2018
50412	2018-05-31 23:54:00	0.30	1786462140797698849	2018
50413	2018-05-31 23:56:00	3.67	3993697860786194247	2018
50414	2018-06-01 00:02:00	3.42	83872787173869366	2018

50415 rows × 4 columns

Ввод [30]:
```
orders['order_month'] = orders['buy_ts'].astype('datetime64[M]')

# расчёт среднего чека покупки
revenue_mean = orders.pivot_table(index='order_month',
                    values='revenue',
                    aggfunc='mean')

revenue_mean.plot(figsize=(15,5)).set(title = 'Годовая динамика среднего чека',
                                      xlabel = 'Месяц',
                                      ylabel = 'y.e.')
plt.show()

print('Средний чек покупки за весь период:')
print(revenue_mean.mean().round(), 'y.e.')
```


```
Средний чек покупки за весь период:
revenue    5.0
dtype: float64 y.e.
```

На графике наблюдается ряд резких всплесков объёма среднего чека покупки. Если всплеск в начале года объясним новогодними каникулами, а в конце апреля наступающими новогодними, то всплеск приходящийся на конец марта, середину июля, пока объяснения не имеет

Расчёт ценности клиента - LTV:

Создание когорт покупателей для расчёта LTV

Ввод [31]:

```
#определение первого месяца заказа
first_orders = orders.groupby('uid').agg({'order_month':'min'}).reset_index()
first_orders.columns=['uid','first_order_month']

#определение количества людей купивших в первый раз
cohort_size=(
    first_orders.groupby('first_order_month')
    .agg({'uid': 'nunique'})
    .reset_index()
)
cohort_size.columns=['first_order_month', 'n_buyers']

# добавление месяца первого заказа к покупкам
orders_first_month = pd.merge(orders, first_orders, on='uid')

# группировка заказов в когорты
cohorts_order = (
    orders_first_month.groupby(['first_order_month','order_month'])
    .agg({'revenue':'sum'})
    .reset_index()
)

#объединение когорт и когорт первых покупок
report=pd.merge(cohort_size, cohorts_order, on='first_order_month')

display(report.head())
```

	first_order_month	n_buyers	order_month	revenue
0	2017-06-01	2023	2017-06-01	9557.49
1	2017-06-01	2023	2017-07-01	981.82
2	2017-06-01	2023	2017-08-01	885.34
3	2017-06-01	2023	2017-09-01	1931.30
4	2017-06-01	2023	2017-10-01	2068.58

Расчёт LTV

Ввод [32]:

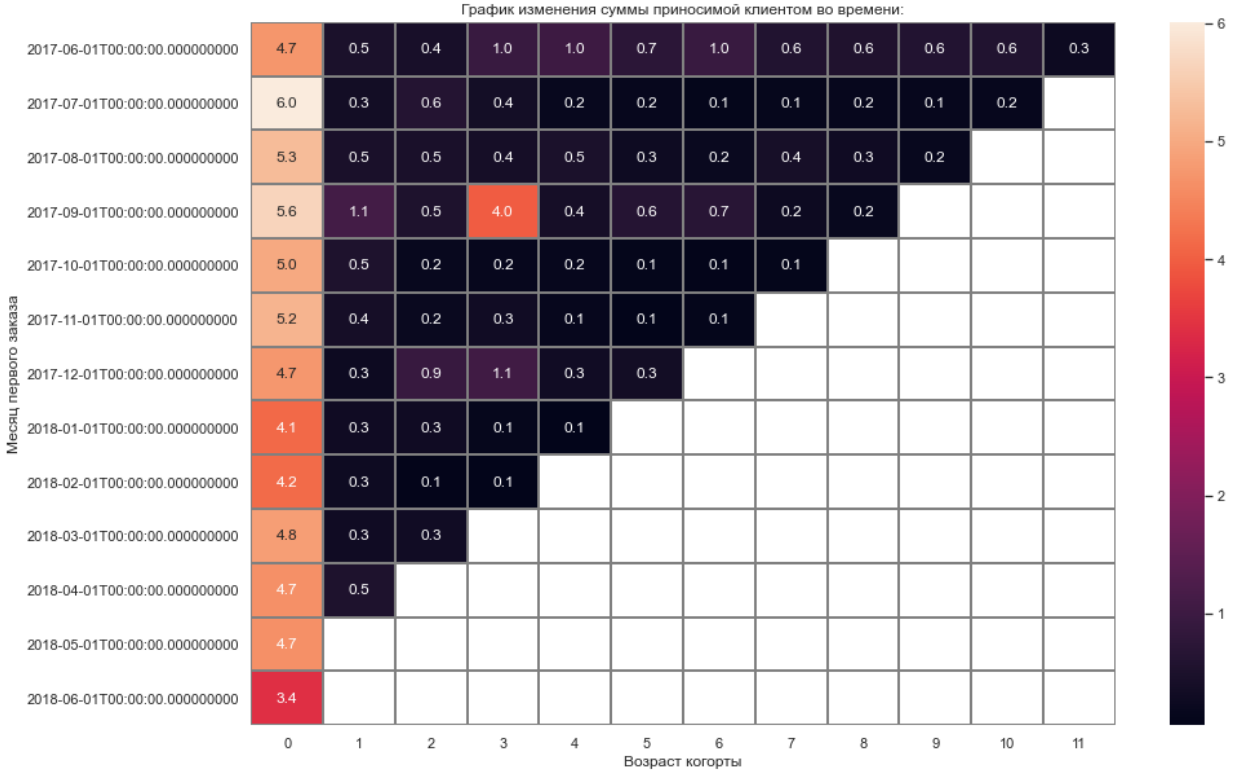
```
# маржинальность 100%
margin_rate = 1

# добавление столбца с данными валовой прибыли
report['gp']=report['revenue']*margin_rate

#добавление столбца с данными возраста когорты
report['age']=(
    report['order_month'] - report['first_order_month']
)/np.timedelta64(1,'M')
report['age']=report['age'].round().astype('int')

#расчёт жизненной ценности клиента
report['ltv']=report['gp']/report['n_buyers']

result = report.pivot_table(
    index='first_order_month', columns='age', values='ltv', aggfunc='mean'
).round()

# расчёт LTV для когорт с возрастом 6 месяцев
month6_cum_ltv=result.cumsum(axis=1).mean(axis=0)[5]
print('Средний LTV по когортам за 6 месяцев:')
print(month6_cum_ltv, 'y.e.')
```

```
Средний LTV по когортам за 6 месяцев:
7.285714285714286 y.e.
```

Чтобы вложения в маркетинг окупились привлечение одного покупателя не должно превышать 7,3 у.е.

График изменения LTV во времени:

Ввод [33]:
```
result_1 = report.pivot_table(
    index='first_order_month', columns='age', values='ltv', aggfunc='mean'
)
sns.set(style='white')
plt.figure(figsize=(15,10))
plt.title('График изменения суммы приносимой клиентом во времени:')
sns.heatmap(result_1,annot=True, fmt='.1f', linewidths=1, linecolor='gray')
plt.xlabel('Возраст когорты')
plt.ylabel('Месяц первого заказа')
plt.show()
```



Вывод по жизненной ценности клиента: метрика показывает низкую жизненную ценность, обращает на себя внимание у когорты сентября 2017 года на третий месяц жизни (декабрь 2017) произошёл резкий подъём суммы покупок на одного человека это связано с тем, что третий месяц пришёлся на декабрь, в конце года покупательская активность заметно повышается.

Маркетинговые метрики:

Ввод [34]:
```
costs['month'] = costs['dt'].astype('datetime64[M]')
display(costs.head())
costs.info()
```

	source_id	dt	costs	month
0	1	2017-06-01	75.20	2017-06-01
1	1	2017-06-02	62.25	2017-06-01
2	1	2017-06-03	36.53	2017-06-01
3	1	2017-06-04	55.00	2017-06-01
4	1	2017-06-05	57.08	2017-06-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2542 entries, 0 to 2541
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   source_id  2542 non-null   int64
 1   dt         2542 non-null   datetime64[ns]
 2   costs      2542 non-null   float64
 3   month      2542 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(1), int64(1)
memory usage: 79.6 KB
```

Общая сумма расходов на маркетинг и распределение по источникам:

Общая сумма расходов на маркетинг:

Ввод [35]:
```
costs_total = costs['costs'].sum()
print('Общая сумма расходов на маркетинг, у.е.')
print(costs_total)
```

```
Общая сумма расходов на маркетинг, у.е.
329131.62
```

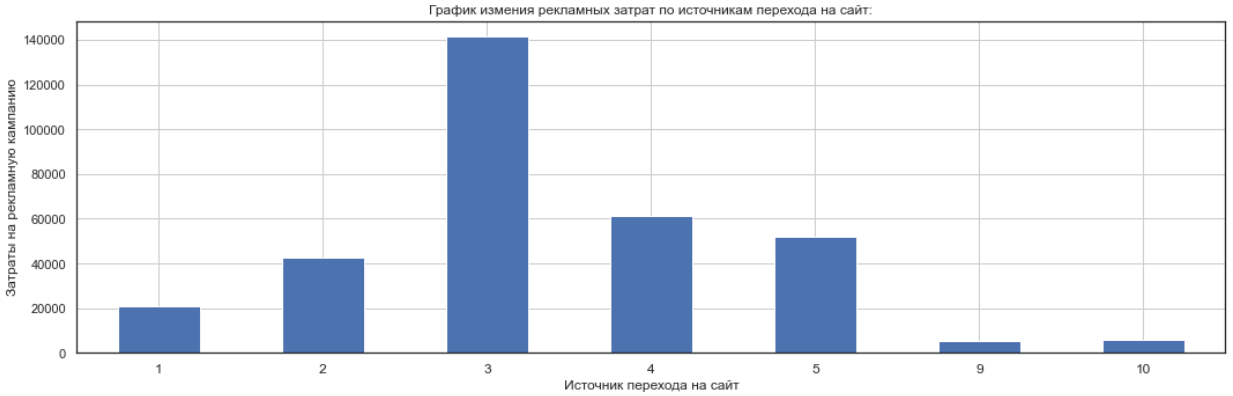
Сумма расходов на маркетинг по источникам перехода на сайт:

Ввод [36]:
```
costs_source = costs.groupby('source_id')['costs'].sum()
print(costs_source)
```

```
source_id
1      20833.27
2      42806.04
3     141321.63
4      61073.60
5      51757.10
9       5517.49
10      5822.49
Name: costs, dtype: float64
```

График расходов на маркетинг по источникам перехода на сайт:

Ввод [37]:
```
costs_source.plot(kind='bar', figsize=(15,5))
plt.xticks(rotation=0, size=12)
plt.title('График измения рекламных затрат по источникам перехода на сайт:')
plt.xlabel('Источник перехода на сайт')
plt.ylabel('Затраты на рекламную кампанию')
plt.grid()
plt.tight_layout()
```



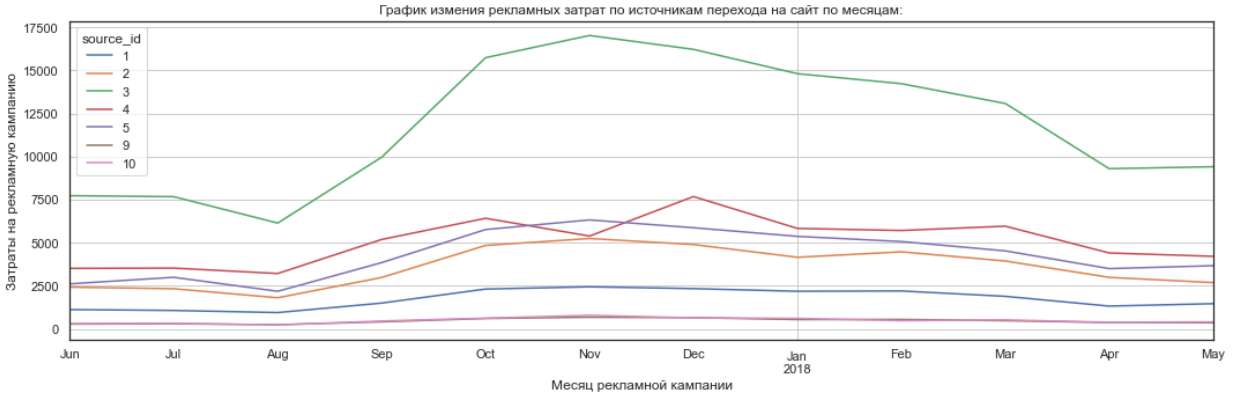
Расчёт распределения расходов на маркетинг по источникам перехода на сайт во времени:

Ввод [38]:
```
costs_source_time = costs.pivot_table(
    index='month',
    columns='source_id',
    values='costs',
    aggfunc='sum')
display(costs_source_time)
```

source_id	1	2	3	4	5	9	10
month							
2017-06-01	1125.61	2427.38	7731.65	3514.80	2616.12	285.22	314.22
2017-07-01	1072.88	2333.11	7674.37	3529.73	2998.14	302.54	329.82
2017-08-01	951.81	1811.05	6143.54	3217.36	2185.28	248.93	232.57
2017-09-01	1502.01	2985.66	9963.55	5192.26	3849.14	415.62	460.67
2017-10-01	2315.75	4845.00	15737.24	6420.84	5767.40	609.41	627.24
2017-11-01	2445.16	5247.68	17025.34	5388.82	6325.34	683.18	792.36
2017-12-01	2341.20	4897.80	16219.52	7680.47	5872.52	657.98	645.86
2018-01-01	2186.18	4157.74	14808.78	5832.79	5371.52	547.16	614.35
2018-02-01	2204.48	4474.34	14228.56	5711.96	5071.31	551.50	480.88
2018-03-01	1893.09	3943.14	13080.85	5961.87	4529.62	480.29	526.41
2018-04-01	1327.49	2993.70	9296.81	4408.49	3501.15	373.49	388.25
2018-05-01	1467.61	2689.44	9411.42	4214.21	3669.56	362.17	409.86

Графики распределения расходов на маркетинг по источникам перехода на сайт во времени:

Ввод [39]:
```
costs_source_time.plot(figsize=(15,5))
plt.title('График измения рекламных затрат по источникам перехода на сайт по месяцам:')
plt.xlabel('Месяц рекламной кампании')
plt.ylabel('Затраты на рекламную кампанию')
plt.grid()
plt.tight_layout()
```



Источник №3 является лидером по вложениям в него средств на рекламу

Источник №10 не пользуется популярностью у маркетологов, затраты на него держатся в районе нуля

Источники №№ 5, 2, 4 идут практически одной группой затраты на них колеблются от 2500 до 7500 в месяц

В целом рост затрат на рекламную кампанию совпадает с ростом появления уникальных клиентов. Распределение средств на продвижение сайта выглядит понятным и стандартным, касательно любого источника увеличение вложений средств на маркетинг начинается с окончанием летнего периода и появлением бизнес активности на рынке расходы идут по нарастающей до появления новогоднего спроса, затем после небольшого снижения расходов рынку не дают остыть вплоть до летнего периода, когда спрос начинает расти на другие виды услуг и товаров.

Расчёт среднего CAC:

Расчёт среднего CAC для всего проекта:

Ввод [40]:
```
display(report.head())
```

	first_order_month	n_buyers	order_month	revenue	gp	age	ltv
0	2017-06-01	2023	2017-06-01	9557.49	9557.49	0	4.724414
1	2017-06-01	2023	2017-07-01	981.82	981.82	1	0.485329
2	2017-06-01	2023	2017-08-01	885.34	885.34	2	0.437637
3	2017-06-01	2023	2017-09-01	1931.30	1931.30	3	0.954671
4	2017-06-01	2023	2017-10-01	2068.58	2068.58	4	1.022531

Расчёт помесячных расходов и добавление данных в когортный отчёт

Ввод [41]:
```
#расчёт помесячных расходов
monthly_costs = costs.groupby('month')['costs'].sum()
display(monthly_costs)
```

```
month
2017-06-01    18015.00
2017-07-01    18240.59
2017-08-01    14790.54
2017-09-01    24368.91
2017-10-01    36322.88
2017-11-01    37907.88
2017-12-01    38315.35
2018-01-01    33518.52
2018-02-01    32723.03
2018-03-01    30415.27
2018-04-01    22289.38
2018-05-01    22224.27
Name: costs, dtype: float64
```

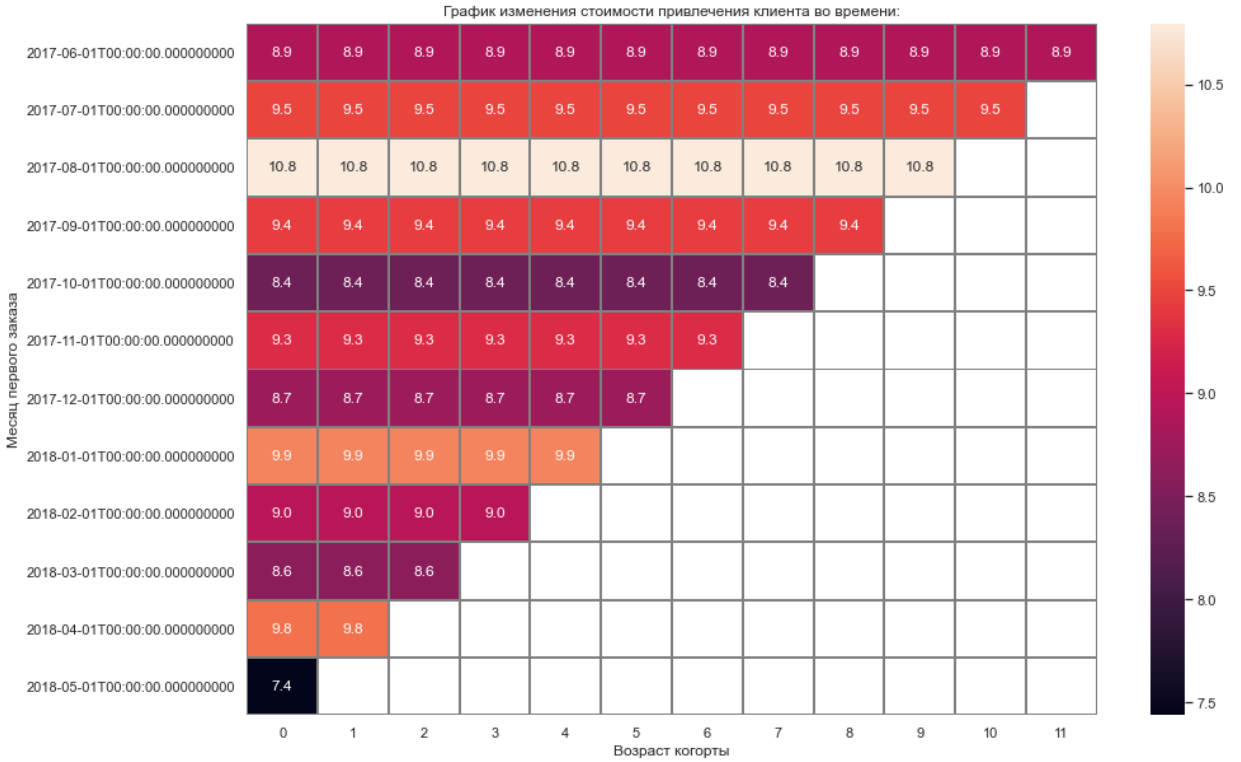
Ввод [42]:
```
#добавление данных в когортный отчет
report_new = pd.merge(report, monthly_costs, left_on='first_order_month', right_on='month')

# расчёт CAC и добавление в таблицу
report_new['cac']=report_new['costs']/report_new['n_buyers']

# расчёт CAC за весь период
result_cac_month = report_new.pivot_table(
    index='first_order_month',
    columns='age',
    values='cac',
    aggfunc='mean')

print('Средний CAC на одного пользователя за весь период у.е.')
print(result_cac_month.mean().mean())
display(report_new.head())
```

```
Средний CAC на одного пользователя за весь период у.е.
9.329712935836932
```

	first_order_month	n_buyers	order_month	revenue	gp	age	ltv	costs	cac
0	2017-06-01	2023	2017-06-01	9557.49	9557.49	0	4.724414	18015.0	8.905091
1	2017-06-01	2023	2017-07-01	981.82	981.82	1	0.485329	18015.0	8.905091
2	2017-06-01	2023	2017-08-01	885.34	885.34	2	0.437637	18015.0	8.905091
3	2017-06-01	2023	2017-09-01	1931.30	1931.30	3	0.954671	18015.0	8.905091
4	2017-06-01	2023	2017-10-01	2068.58	2068.58	4	1.022531	18015.0	8.905091

Тепловая карта среднего CAC для всего проекта:

Ввод [43]:
```
sns.set(style='white')
plt.figure(figsize=(15,10))
plt.title('График изменения стоимости привлечения клиента во времени:')
sns.heatmap(result_cac_month,annot=True, fmt='.1f', linewidths=1, linecolor='gray')
plt.xlabel('Возраст когорты')
plt.ylabel('Месяц первого заказа')
plt.show()
```



Вывод по среднему CAC для всего проекта: для всего периода денег на привлечение тратится больше, чем приносит в среднем один клиент. По этим данным деньги на маркетинг не окупаются.

Расчёт среднего CAC и ROMI для каждого источника:

Создание датафрейма для расчётов CAC по каждому источнику

Ввод [44]:
```
# создадим свежую таблицу для расчётов
visits_s = visits[['device','end_ts','source_id','start_ts','uid']]
visits_s_orders = visits_s.merge(orders, on='uid')
display(visits_s_orders.head())
```

	device	end_ts	source_id	start_ts	uid	buy_ts	revenue	buy_ts_year	order_month
0	desktop	2018-05-20 11:23:00	9	2018-05-20 10:59:00	16174680259334210214	2018-03-09 20:25:00	2.33	2018	2018-03-01
1	desktop	2018-03-09 20:33:00	4	2018-03-09 20:05:00	16174680259334210214	2018-03-09 20:25:00	2.33	2018	2018-03-01
2	desktop	2017-09-03 21:36:00	5	2017-09-03 21:35:00	16007536194108375387	2017-09-04 12:46:00	2.44	2017	2017-09-01
3	desktop	2017-09-03 21:36:00	5	2017-09-03 21:35:00	16007536194108375387	2017-10-28 00:01:00	1.53	2017	2017-10-01
4	desktop	2017-09-03 21:36:00	5	2017-09-03 21:35:00	16007536194108375387	2017-10-28 19:16:00	1.53	2017	2017-10-01

Датафрейм с расходами по месяцам и источникам

Ввод [45]:
```
monthly_costs_source = costs.groupby(['month','source_id'])['costs'].sum()
display(monthly_costs_source.head())
```

```
month       source_id
2017-06-01  1            1125.61
            2            2427.38
            3            7731.65
            4            3514.80
            5            2616.12
Name: costs, dtype: float64
```

Создание функции для построения тепловой карты CAC и ROMI для каждого источника

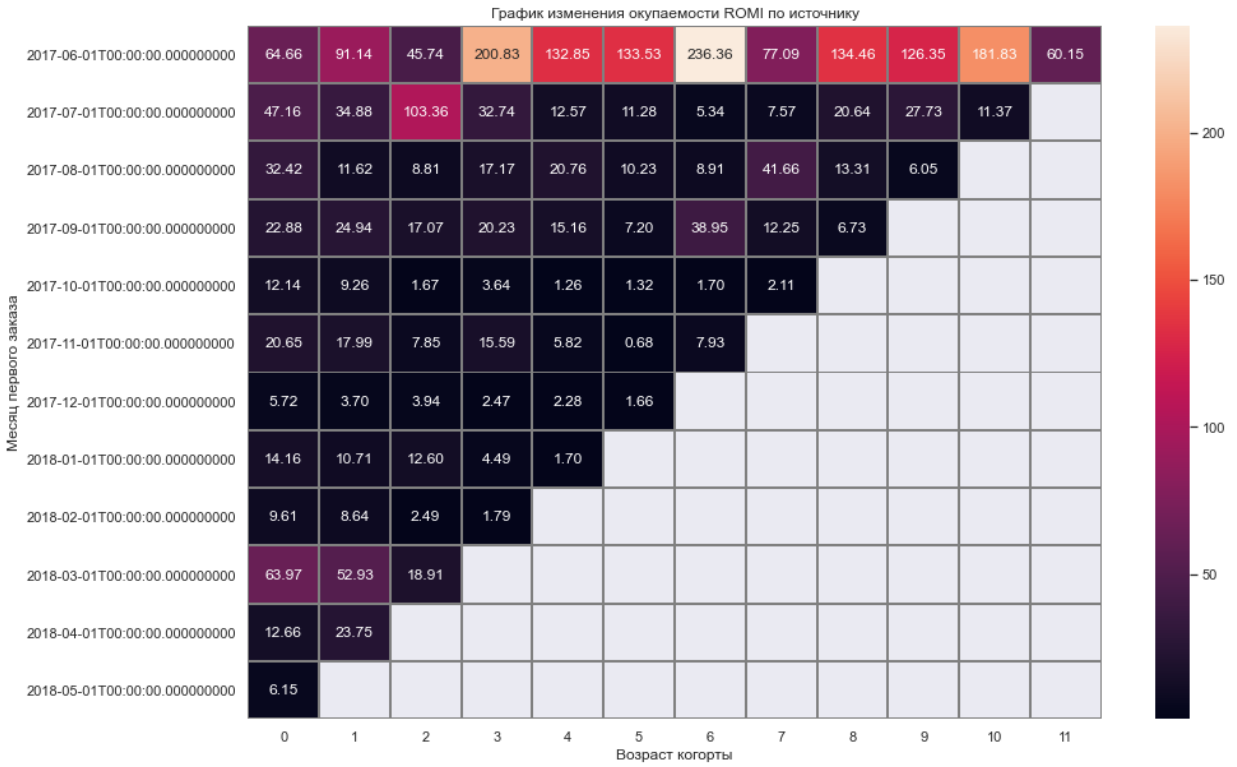
Ввод [46]:
```
# создание функции, которая в качестве аргумента будет брать срез по номеру источника, а возвращать C
def print_cac_source(visits_s_orders):
    first_orders_s=visits_s_orders.groupby('uid').agg({'order_month':'min'}).reset_index()
    first_orders_s.columns=['uid','first_order_month']# первый месяц заказа

    # вычисление уникальных покупателей:
    cohort_size_s=(first_orders_s.groupby('first_order_month').agg({'uid': 'nunique'}).reset_index())
    cohort_size_s.columns=['first_order_month', 'n_buyers']

    # объединение основной таблицы с первм месяцем заказа :
    visits_first_month_s = pd.merge(visits_s_orders, first_orders_s, on='uid')

    # создание когортыполучение выручки по группировки месяца заказа и первого месяца заказаЖ
    cohorts_order_s = (visits_first_month_s.groupby(['first_order_month','order_month']).agg({'revenu

    # датафрейм из уникальных покупателей и выручки
    report_s=pd.merge(cohort_size_s, cohorts_order_s, on='first_order_month')

    # добавление данных по расходам с группировкой по месяцу и источнику
    report_s=pd.merge(report_s, monthly_costs_source, left_on='first_order_month', right_on='month')

    # расчёт валовой прибыли, ltv, cac, romi, довабление возраста когорты
    report_s['gp']=report_s['revenue']*1
    report_s['ltv']=report_s['gp']/report_s['n_buyers']
    report_s['cac']=report_s['costs']/report_s['n_buyers']
    report_s['age']=(report_s['order_month'] - report_s['first_order_month'])/np.timedelta64(1,'M')
    report_s['age']=report_s['age'].round().astype('int')

    # CAC тепловая карта
    grouped_cac_source = report_s.pivot_table(index='first_order_month',columns='age',values='cac',ag
    print('Средний CAC по источнику:',grouped_cac_source.mean().mean())
    sns.set(style='white')
    plt.figure(figsize=(15,10))
    plt.title('График изменения стоимости привлечения клиента CAC из источника')
    sns.heatmap(grouped_cac_source,annot=True, fmt='.2f', linewidths=1, linecolor='gray')
    plt.xlabel('Возраст когорты')
    plt.ylabel('Месяц первого заказа')

    # распределение ltv по когортам
    ltv = report_s.pivot_table(index='first_order_month',columns='age',values='ltv',aggfunc='mean')
    print('Пожизненная ценность клиента по источнику LTV:')
    display(ltv.fillna(''))

    # ROMI тепловая карта
    report_s['romi']=report_s['ltv']/report_s['cac']
    grouped_romi_source = report_s.pivot_table(index='first_order_month',columns='age',values='romi',
    grouped_romi_source.cumsum(axis=1).round(2)
    sns.set(style='dark')
    plt.figure(figsize=(15,10))
    plt.title('График изменения окупаемости ROMI по источнику')
    sns.heatmap(grouped_romi_source,annot=True, fmt='.2f', linewidths=1, linecolor='gray')
    plt.xlabel('Возраст когорты')
    plt.ylabel('Месяц первого заказа')
```

CAC и ROMI для источника №1:

Ввод [47]:
```
print_cac_source(visits_s_orders[visits_s_orders['source_id'] == 1])
```

```
Средний CAC по источнику: 6.594861875654856
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	
first_order_month									
2017-06-01	128.491147	181.106883	90.895586	399.052594	263.985187	265.332294	469.668529	153.191696	267.1866
2017-07-01	97.716925	72.275575	214.182925	67.8434	26.03745	23.37955	11.073875	15.689625	42.7711
2017-08-01	72.617535	26.022222	19.740799	38.452326	46.493715	22.908924	19.960382	93.322535	29.8110
2017-09-01	49.658650	54.139068	37.048232	43.910038	32.918935	15.634278	84.554297	26.593707	14.6115
2017-10-01	23.905846	18.235126	3.27976	7.170324	2.485798	2.602593	3.352437	4.159676	
2017-11-01	41.068951	35.783581	15.62482	31.019407	11.574047	1.348665	15.778549		
2017-12-01	10.926301	7.066733	7.514086	4.706257	4.358007	3.159391			
2018-01-01	33.944107	25.676897	30.197712	10.767978	4.080486				
2018-02-01	20.766981	18.675402	5.38243	3.865294					
2018-03-01	126.697914	104.818743	37.450044						
2018-04-01	32.562020	61.098772							
2018-05-01	12.743923								



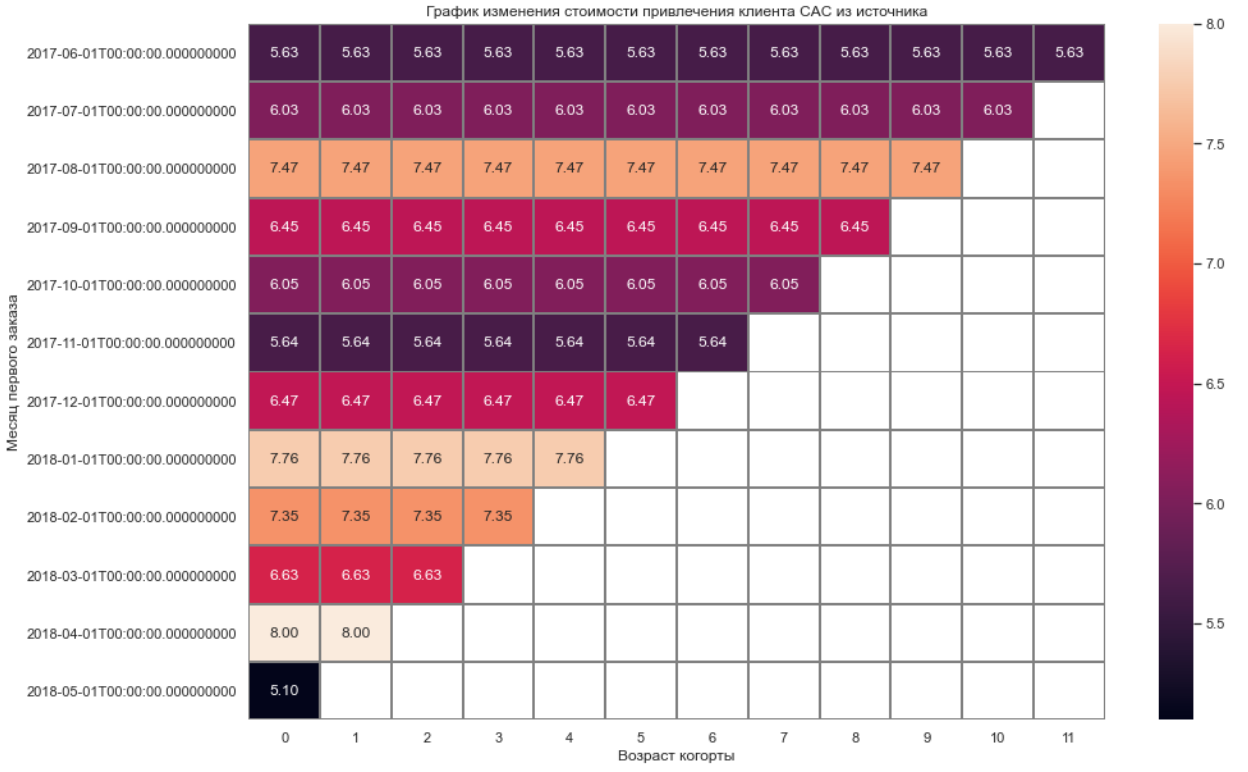
По источнику №1 можно сделать следующий вывод: в целом рекламные вложения окупились, для некоторых когорт в определённое время жизни в 200 раз, саммыми удачными для данного источника являются июньсквая когорта и мартовская. Неважно покупки шли у когорты зарегистрированной в декабре. Когорта зарегистрированная в ноябре не окупила вложенные в неё средства в апреле, так и не сделав нужное количество покупок.

CAC и ROMI для источника №2

Ввод [48]:
```
print_cac_source(visits_s_orders[visits_s_orders['source_id'] == 2])
```

```
Средний CAC по источнику: 6.314069334996788
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	
first_order_month									
2017-06-01	108.590832	89.336871	85.376018	276.357418	279.109497	132.41442	193.414748	208.345383	178.52822
2017-07-01	52.100185	16.745278	49.225556	22.524792	9.52037	12.026968	6.655486	7.55463	7.5024(
2017-08-01	30.323039	32.006502	25.303322	24.120035	54.007562	19.221873	14.849611	29.80053	17.01646
2017-09-01	153.888389	202.679796	52.410741	962.169574	97.636574	146.249056	193.262241	62.910556	30.8390
2017-10-01	33.095571	14.889685	5.737075	4.500909	5.788322	4.972016	4.149848	5.891294	
2017-11-01	25.007458	25.107646	20.851813	25.940542	6.86024	1.832823	2.846781		
2017-12-01	34.789362	28.913026	61.348262	54.3074	17.969113	16.93104			
2018-01-01	20.783663	17.298703	30.189465	6.964554	5.736807				
2018-02-01	11.326651	8.319119	5.231808	3.898286					
2018-03-01	90.770336	82.374183	46.763069						
2018-04-01	21.496683	31.299146							
2018-05-01	13.890482								



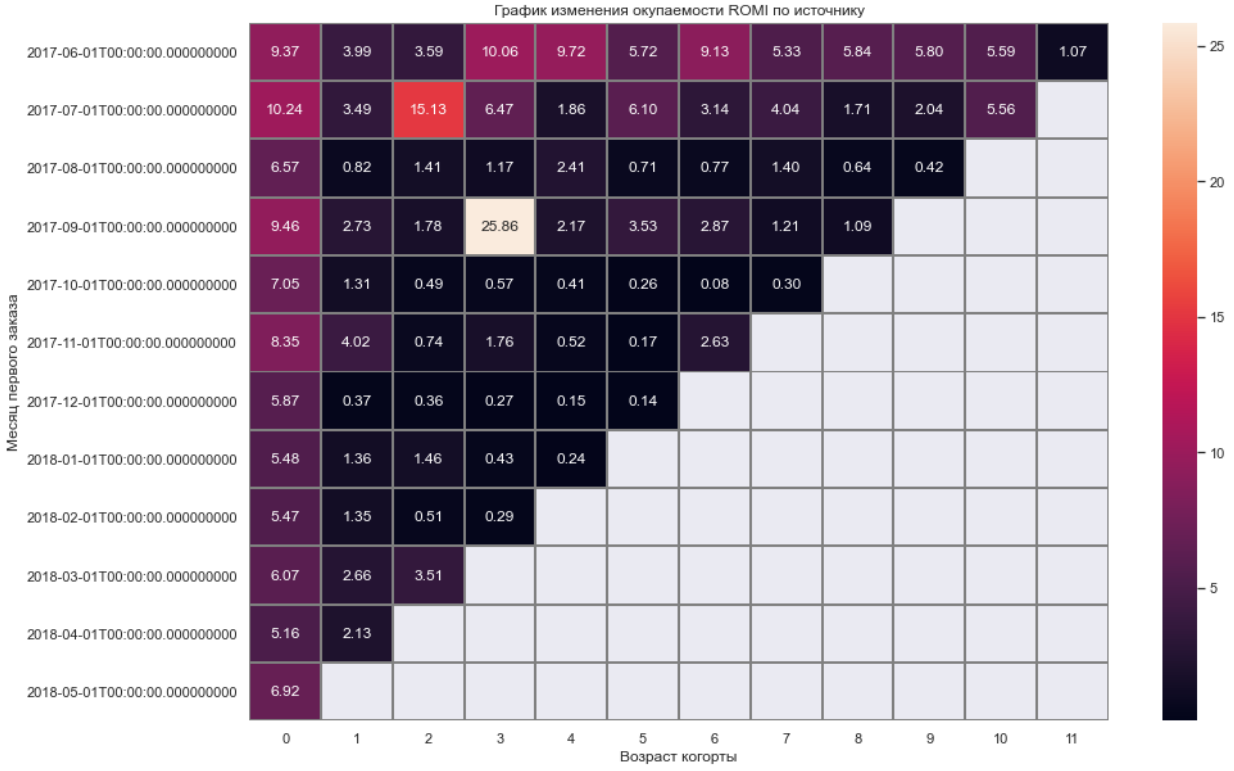
По источнику №2 самая удачная когорта июньская при самой низкой стоимости привлечения получилинаилучший результат на протяженииив сей жизни когорты, сентябрьская и мартовские когорты также здорово окупились особенно отличилась сентябрьская в декабре в 400 раз выше точкиокупаемости. Ноябрьская когорта как и в случае с источником №1 неокупилась в апреле.

CAC и ROMI для источника №3

Ввод [49]:
```
print_cac_source(visits_s_orders[visits_s_orders['source_id'] == 3])
```

```
Средний CAC по источнику: 3.424502340544232
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	9	
first_order_month											
2017-06-01	8.544416	3.633421	3.270847	9.173432	8.857002	5.211556	8.319199	4.856453	5.327975	5.290172	5
2017-07-01	11.873608	4.043678	17.542168	7.504406	2.155343	7.069636	3.645734	4.679189	1.982196	2.362112	6
2017-08-01	8.629959	1.080102	1.854521	1.541202	3.164684	0.935886	1.006599	1.844929	0.844134	0.554175	
2017-09-01	10.473802	3.019981	1.974054	28.640776	2.404345	3.906644	3.17807	1.335713	1.207789		
2017-10-01	7.132731	1.321418	0.496141	0.574544	0.413755	0.266412	0.078329	0.305136			
2017-11-01	9.353459	4.502385	0.828527	1.96523	0.585289	0.186094	2.946941				
2017-12-01	5.833855	0.366734	0.353491	0.270894	0.14633	0.135355					
2018-01-01	6.456399	1.606453	1.72552	0.511712	0.286608						
2018-02-01	5.593375	1.375798	0.52563	0.301288							
2018-03-01	6.388656	2.802327	3.69919								
2018-04-01	6.804055	2.81458									
2018-05-01	7.006412										



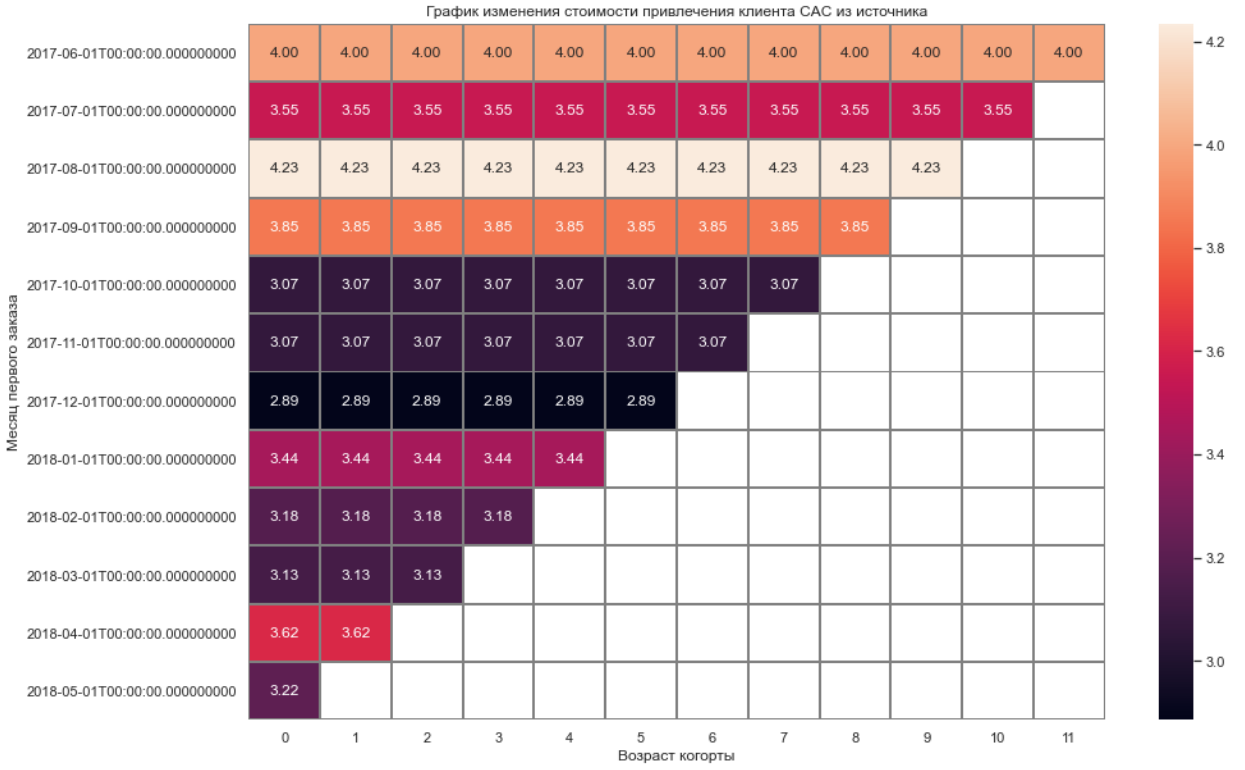
По источнику №3, на него приходится наибольшее количество денежных затрат чем на продвижение остальных источников. Это не самый дорогой источник, на привлечение одного клиента тратится в два раза меньше средств чем на источники №1 и №2. В этом источнике также луччше всего выглядит июньская когорта. Очень большое количество когорт не окупились в течение жизни. Источник очень слабый, средства на него потрачены зря.

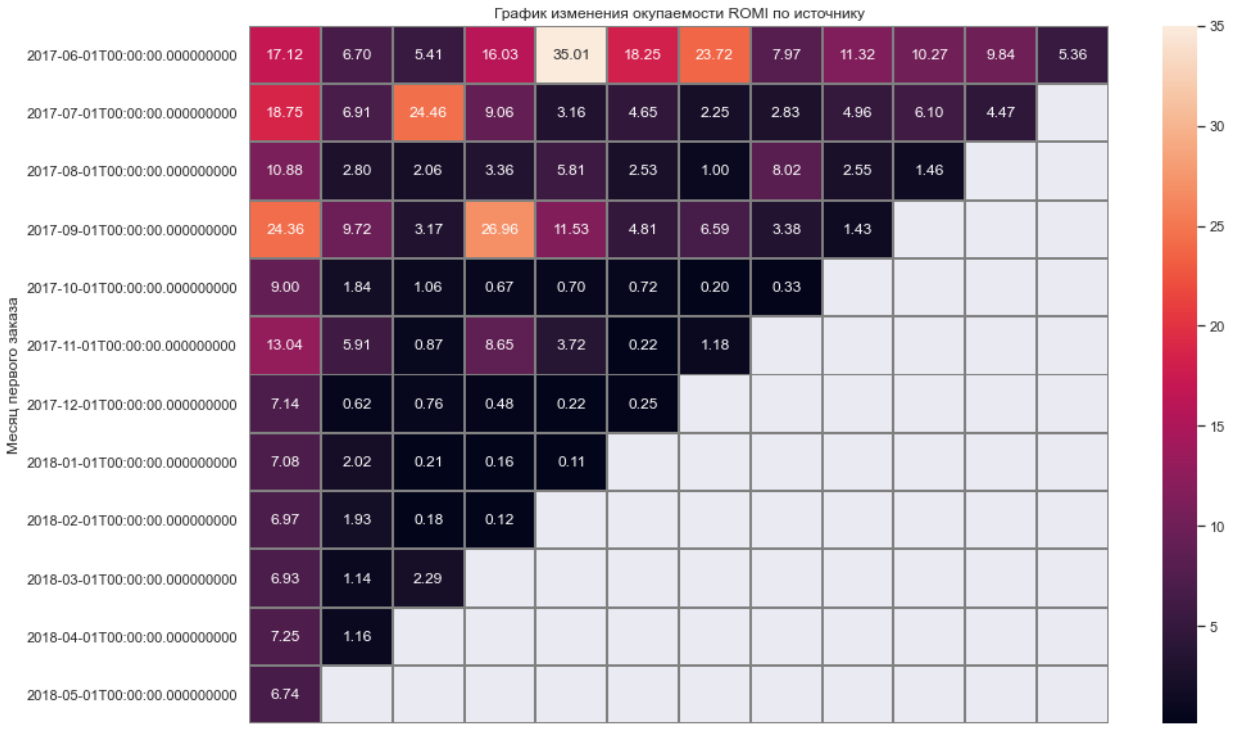
CAC и ROMI для источника №4

Ввод [50]:
```
print_cac_source(visits_s_orders[visits_s_orders['source_id'] == 4])
```

```
Средний CAC по источнику: 3.650280422241669
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	
first_order_month										
2017-06-01	21.186599	8.285668	6.689348	19.827795	43.312873	22.586506	29.351584	9.863059	14.005342	12.703
2017-07-01	21.142095	7.796612	27.582354	10.216898	3.568041	5.245143	2.532599	3.188	5.588163	6.876
2017-08-01	14.068537	3.625792	2.664729	4.344048	7.517074	3.270782	1.291443	10.367315	3.291984	1.882
2017-09-01	30.764469	12.277434	4.001825	34.057876	14.565819	6.078009	8.318407	4.268053	1.800575	
2017-10-01	8.732069	1.784232	1.024817	0.648664	0.678717	0.699616	0.194326	0.321117		
2017-11-01	13.863482	6.281285	0.928624	9.198307	3.957616	0.235147	1.254717			
2017-12-01	6.487422	0.564776	0.694481	0.436964	0.201708	0.228297				
2018-01-01	7.781402	2.225758	0.230115	0.170625	0.123681					
2018-02-01	6.621498	1.834799	0.173227	0.111184						
2018-03-01	6.682901	1.097091	2.202801							
2018-04-01	8.300205	1.327281								
2018-05-01	6.954498									





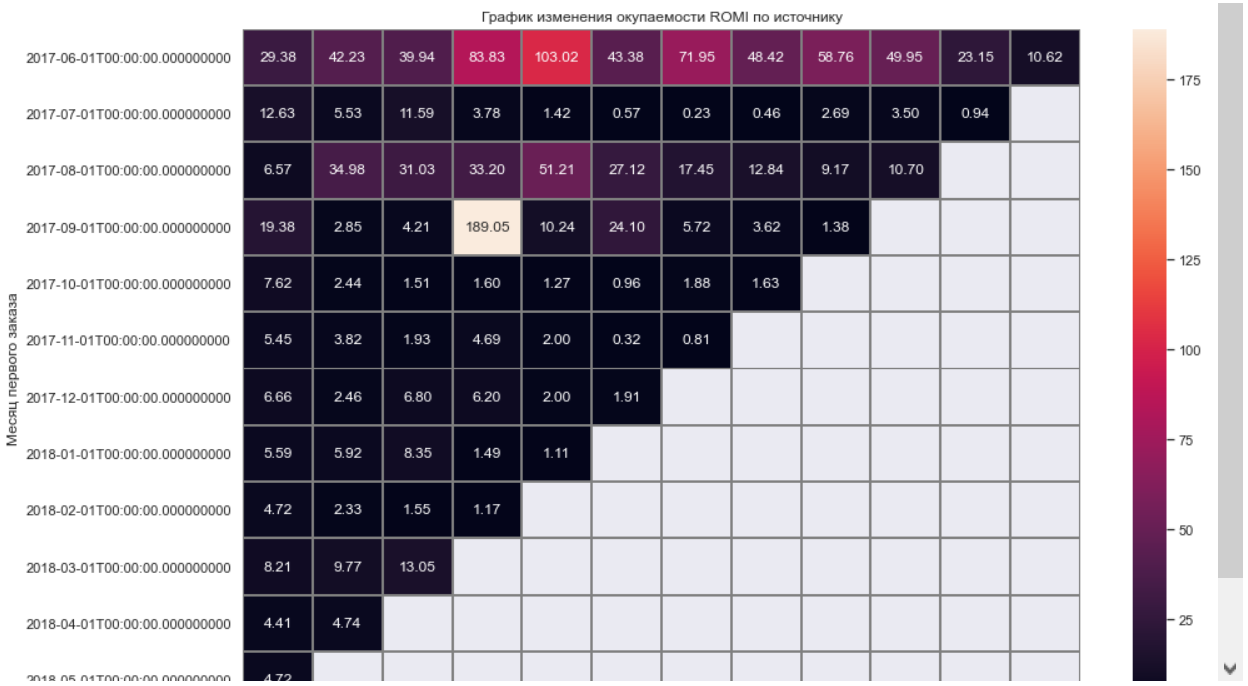
По источнику №4, для продвижения этого источника средств вкладывается соразмерно источнику №2, но привлечение одного человека стоит почти в два раза дешевле чем в том источнике. У когорт с июня по сентябрь двольно высокая окупаемость, очень плохая окупаемость, а точнее отсутствие окпуаемости начиная с декабрь ской когорты и позже, причём затраты на привлечение одного человека стабильны.

CAC и ROMI для источника №5

Ввод [51]:
```
print_cac_source(visits_s_orders[visits_s_orders['source_id'] == 5])
```

```
Средний CAC по источнику: 4.489084536928937
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	
first_order_month										
2017-06-01	40.994028	58.927145	55.734553	116.979632	143.76056	60.531436	100.404869	67.564606	81.997198	69
2017-07-01	17.481803	7.6502	16.033272	5.224925	1.968297	0.791903	0.324858	0.632387	3.724908	4.8
2017-08-01	9.153348	48.734298	43.231577	46.260022	71.350972	37.789611	24.312073	17.884341	12.772613	14.9
2017-09-01	25.676148	3.773492	5.577889	250.43319	13.567181	31.919432	7.571462	4.797193	1.830174	
2017-10-01	9.327664	2.990336	1.845052	1.964149	1.552985	1.181082	2.30556	1.996209		
2017-11-01	9.261910	6.498271	3.275801	7.970941	3.392136	0.549919	1.370136			
2017-12-01	10.304174	3.812316	10.526553	9.586086	3.092047	2.948303				
2018-01-01	9.201130	9.740269	13.74662	2.45775	1.8262					
2018-02-01	6.726357	3.313898	2.205133	1.673561						
2018-03-01	12.760674	15.186957	20.292009							
2018-04-01	7.058108	7.575183								
2018-05-01	6.846060									



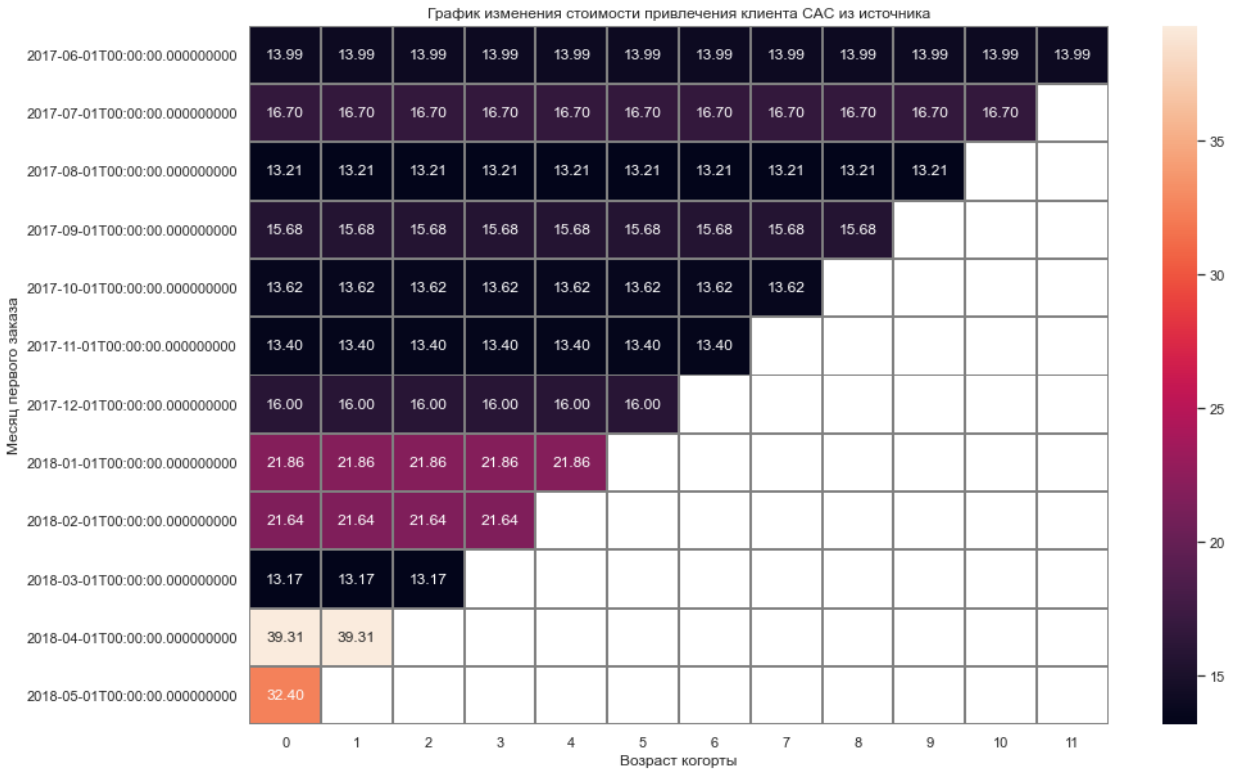
По источнику №5, находится в тройке средней популярности по вложениям средств в продвижение, но источник стабильный и все когорты окупаются.

CAC и ROMI для источника №9

Ввод [52]:
```
print_cac_source(visits_s_orders[visits_s_orders['source_id'] == 9])
```

```
Средний CAC по источнику: 15.634531813029431
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	9	
first_order_month											
2017-06-01	8.264457	0.480272	1.798261	2.613804	2.083696	1.404348	2.025	0.324402	1.866359	1.76625	3.0
2017-07-01	10.829487	1.716603	5.199231	2.134872	1.383141	0.642821	0.459487	0.539872	1.606026	0.746026	0.4
2017-08-01	9.628875	1.309	2.370813	1.646688	1.64375	0.945688	1.49075	2.35325	3.696	1.147625	
2017-09-01	7.418919	2.694595	1.037072	0.598694	0.531396	0.351036	0.44509	0.601937	0.726757		
2017-10-01	7.309449	1.427087	0.676929	0.296877	0.28147	0.509948	0.129318	0.185827			
2017-11-01	8.937030	3.313911	0.339282	0.580619	0.19948	0.214183	2.309257				
2017-12-01	6.151199	0.852632	0.522281	0.54617	0.402719	0.436754					
2018-01-01	5.120548	0.390685	0.744064	0.153242	0.042648						
2018-02-01	6.689074	0.548472	0.126343	0.018889							
2018-03-01	4.697939	0.195697	0.159879								
2018-04-01	12.862963	1.151728									
2018-05-01	5.721224										



По источнику №9, источник абсолютно бесполезный. Средств на его продвижение тратится немного, но и окупаемости от него нет, даже самая успешная июньская когорта, окупилась только в первый месЯц своего существования.

CAC и ROMI для источника №10

Ввод [53]:
```
print_cac_source(visits_s_orders[visits_s_orders['source_id'] == 10])
```

```
Средний CAC по источнику: 29.684255363285462
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	9	
first_order_month											
2017-06-01	7.750750	1.42825	0.924083	0.861167	2.640417	1.234083	0.886417	0.062583	1.700583	3.096583	2.42
2017-07-01	4.698169	1.485915	1.515211	0.451831		0.052535	0.116056	0.161549	0.038028	0.027887	0.37
2017-08-01	7.301786	0.679107	0.915714	0.133393	0.332857	0.007857	0.146071	0.165179		0.070893	
2017-09-01	3.802182	0.319455	0.418909	0.295909	0.022182	0.016636	0.026545	0.069364	0.377182		
2017-10-01	3.572780	0.147683	0.241583	0.196564	0.31834	0.082432	0.016062	0.069344			
2017-11-01	7.554973	0.333005	0.112459	3.057923	0.234754	0.178579	0.09541				
2017-12-01	5.018130	0.117236	0.071951	0.076341	0.009919	0.094959					
2018-01-01	3.575246	0.177787	0.178279	0.013525	0.022541						
2018-02-01	4.540678	0.244181	0.256271	0.034463							
2018-03-01	7.166434	0.084344	0.080656								
2018-04-01	3.517778	0.20875									
2018-05-01	8.481646										

По источнику №10, данный источник можно выводить из интересов маркетологов, окупаемости нет, а в двух когортах: в июльской в ноябре и в августовской в апрле, похоже вообще не было уникальных клиентов

Вывод по расходам по привлечению в когортах по источникам: во всех источниках кроме 10-го когорты окупились в первый месяц своего существования. По объемй затраченных средств можно говорить, что маркетологои ошиблись с использованием источников 9, 10 и переоценили источник №3. Источники:2,4,5 находятся в одном диапазоне по вливанию в них денежных средств и клиенты переходящие из этих источников обеспечивают окупаемость.

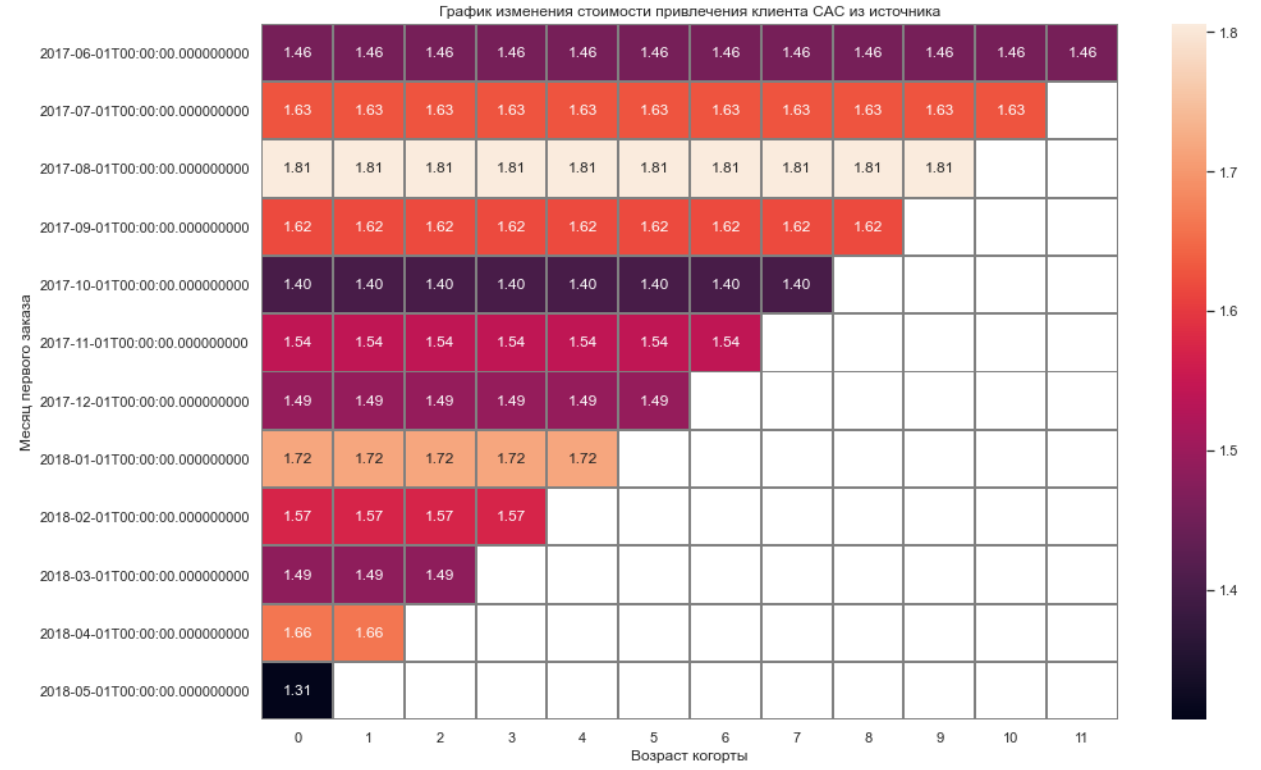
CAC, LTV, ROMI для девайсов:

CAC, LTV, ROMI для desktop:

Ввод [54]:
```
print_cac_source(visits_s_orders[visits_s_orders['device'] == 'desktop'])
```

```
Средний CAC по источнику: 1.5712015837409232
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8
first_order_month									
2017-06-01	77.432476	86.756924	64.282289	211.842601	196.029904	123.982748	203.822181	116.325824	140.205428
2017-07-01	53.920843	30.288869	91.660493	32.218944	10.125928	14.497783	7.15772	9.325509	16.433304
2017-08-01	35.812120	35.22288	30.012239	36.079667	57.147538	26.837957	19.165624	42.44135	17.970094
2017-09-01	67.988401	50.377356	17.735395	374.778002	41.017542	56.946097	71.945539	25.536324	12.914322
2017-10-01	20.528018	9.123824	3.367953	3.830041	2.872012	2.43922	2.645741	3.163675	
2017-11-01	24.873790	22.294966	11.495188	19.946822	6.877751	1.235137	7.724932		
2017-12-01	17.892674	9.923989	19.677162	16.910586	6.316546	5.771693			
2018-01-01	19.558590	14.540951	19.063454	5.023885	2.964351				
2018-02-01	13.011154	8.09998	3.326341	2.400781					
2018-03-01	28.498667	26.782619	25.121056						
2018-04-01	17.912062	22.977824							
2018-05-01	12.299193								



CAC, LTV, ROMI для touch:

Ввод [55]:
```
print_cac_source(visits_s_orders[visits_s_orders['device'] == 'touch'])
```

```
Средний CAC по источнику: 5.509399699516462
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	9
first_order_month										
2017-06-01	25.226806	6.043128	3.173106	1.276586	12.864229	4.741498	2.125595	1.258965	5.089515	10.009802
2017-07-01	22.673885	2.363865	6.785442	3.183577	7.451019	1.58625	1.130538	1.485769	3.940038	7.077769
2017-08-01	15.511815	1.515446	1.289138	0.925446	1.334554	1.1604	0.520338	1.229015	2.887662	1.456431
2017-09-01	39.809272	73.700712	31.51695	19.352663	14.289567	2.73483	17.70144	3.897368	1.858421	
2017-10-01	14.773721	3.911413	0.51194	0.475075	0.489731	0.811622	0.15393	0.603572		
2017-11-01	31.412528	6.938552	1.226575	13.53105	5.300316	0.151957	0.122477			
2017-12-01	9.357980	0.63727	0.322202	0.308681	0.170864	0.108399				
2018-01-01	11.326585	1.041065	0.315214	0.465116	0.160037					
2018-02-01	8.389097	1.308214	0.131499	0.070565						
2018-03-01	106.300967	73.63248	8.846268							
2018-04-01	10.786706	1.033135								
2018-05-01	8.070240									

Вывод по девайсам:

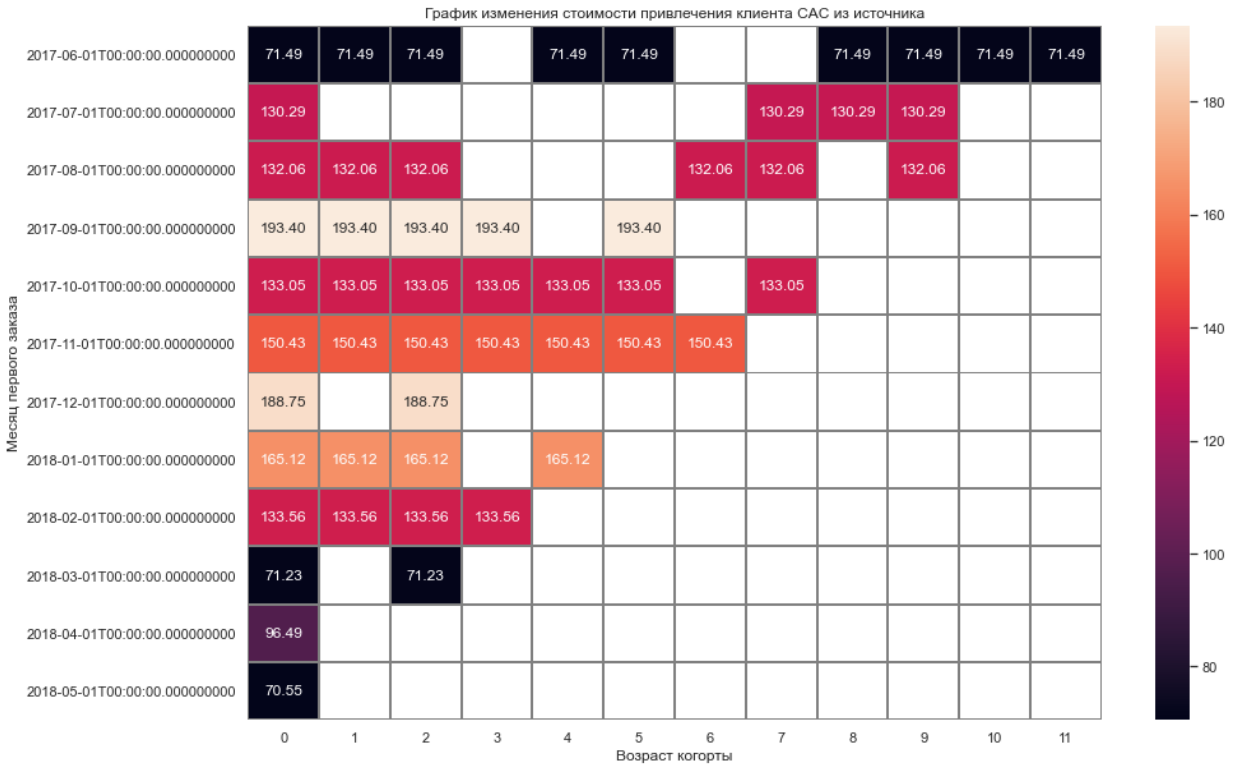
Сервис Яндекс.Афиша судя по всему больше подходит пользователям настольных ПК нежели смартфонов, на привлечение юзеров ПК тратится меньше денег и эффект от потраченных средств значительно превышает пользователей переносных гаджетов. Возможно из-за большего экрана, возможно из-за отутствия хорошо работающего приложения.

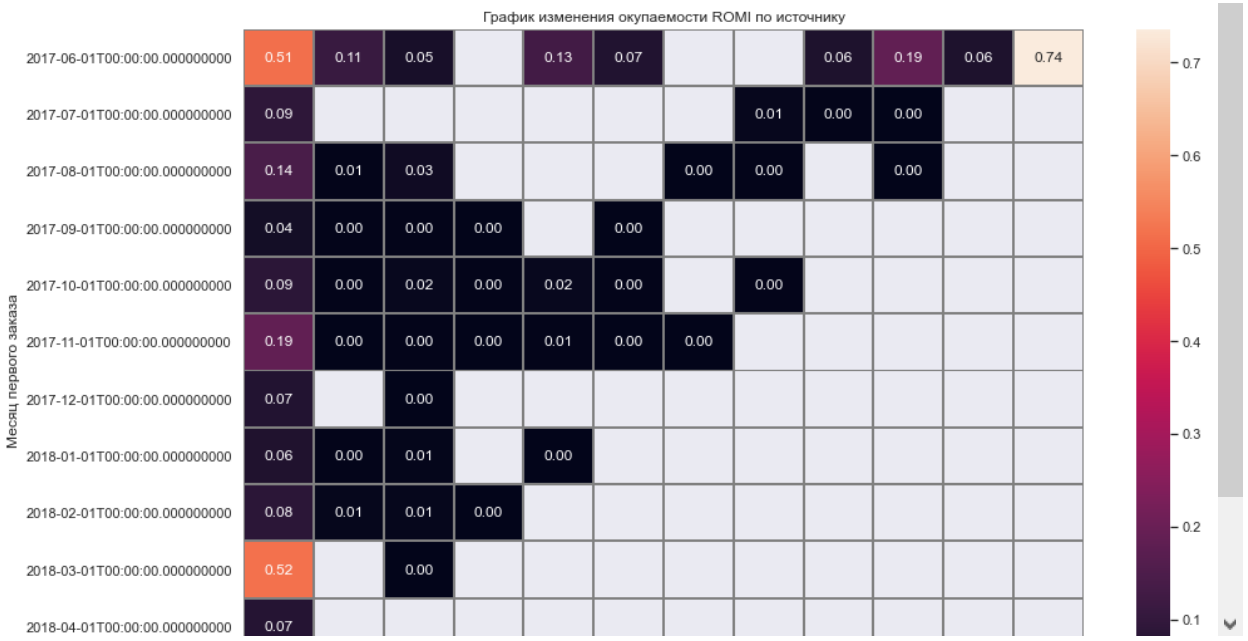
CAC, LTV, ROMI для touch и источника №10:

Ввод [56]:
```
print_cac_source(visits_s_orders[(visits_s_orders['device'] == 'touch')&(visits_s_orders['source_id']
```

```
Средний CAC по источнику: 121.12463761019505
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	9	
first_order_month											
2017-06-01	11.367778	2.479722	1.069167		2.851667	1.466667			1.2825	4.099167	1.2933
2017-07-01	3.800500							0.5735	0.135	0.099	
2017-08-01	5.733125	0.22125	1.2675				0.095625	0.13375		0.038125	
2017-09-01	2.407778	0.223889	0.283333	0.081111		0.101667					
2017-10-01	3.617692	0.141026	0.708205	0.144103	0.689231	0.156923		0.181795			
2017-11-01	9.692222	0.100833	0.178333	0.128889	0.276111	0.081667	0.025556				
2017-12-01	3.988276		0.084138								
2018-01-01	3.273793	0.221034	0.36		0.094828						
2018-02-01	3.390286	0.444857	0.374	0.087143							
2018-03-01	11.329016		0.004918								
2018-04-01	2.260303										
2018-05-01	5.933333										





CAC, LTV, ROMI для desktop и источника №10:

Ввод [57]:
```
print_cac_source(visits_s_orders[(visits_s_orders['device'] == 'desktop')&(visits_s_orders['source_id
```

```
Средний CAC по источнику: 37.54427793517663
Пожизненная ценность клиента по источнику LTV:
```

age	0	1	2	3	4	5	6	7	8	9	
first_order_month											
2017-06-01	5.918750	0.933182	0.822727	1.174318	2.433977	1.082841	1.20875	0.085341	1.794318	2.545682	2.78
2017-07-01	5.050196	2.068627	2.109412	0.62902		0.073137	0.161569				0.52
2017-08-01	7.735854	0.84122	0.756098	0.182195	0.454634	0.010732	0.162195	0.173415		0.081951	
2017-09-01	3.946316	0.327474	0.431368	0.327263	0.025684		0.030737	0.080316	0.436737		
2017-10-01	3.485600	0.145556	0.155333	0.201289	0.246978	0.067689	0.018489	0.048311			
2017-11-01	6.890933	0.382067	0.0944	3.699733	0.220133	0.198267	0.110267				
2017-12-01	5.170825	0.14866	0.066082	0.096804	0.012577	0.120412					
2018-01-01	3.630213	0.162553	0.120319	0.017553							
2018-02-01	4.597584	0.18557	0.216577	0.02047							
2018-03-01	5.595450	0.108889	0.10254								
2018-04-01	3.856875	0.268393									
2018-05-01	9.213033										

Вывод по девайсам для источника №10:

Среди пользователей обоих девайсов периодически не было посещений в источнике №10, но именно в июльской когорте в ноябре и августовской в апреле не было посещений с обоих устройств, на графике 3.2.2 отражена нулевая посещаемость в апреле. Это говорит о явных техническихпроблемах. А совпадение в отсутствии посещений с обоих устройств в источнике привело к тому, что в данных LTV, CAC, ROMI по источнику №10 в июльской когорте в ноябре и августовской в апрлеле повились NAN/

Общий вывод:

- В общей картине среднее значение привлечения клиента CAC 9,3 у.е. превышает среднее значение его жизненной ценности LTV 7,3 у.е, среднее количество покупок 1,32 шт со средним чеком 5 у.е говорит о том, что рекламные расходы не окупаются и требуют их оптимизации.
- Прирост уникальных клиентов например DAU равный 907 говорит о том, что новые клиенты привлекаются, успех у рекламы есть, но коэфициент удержания 4% свидетельстыует, что нет программ для того, чтобы реализовать правило - 20% клиентов приносят 80% дохода. Отсюда большие рекламные расходы.
- Рекомендации:
  - Не все источники перехода нужны в использовании, можно смело отказываться от источников 9 и 10. Сократить финансирование источника №3 вдвое, высвободившиеся средства перераспределить между источниками 1, 2 и 5 как самыми действенными.
  - Создать качественное приложение или адаптацию сайта для переносных устройств, 17 618 60-ти секундных сессий у пользователей touch и абсолютная неокупаемость рекламы. Им неинтересен сервис. возможно поэтому неокупаются источники 9 и 10, так как могут быть расчитаны именно на этот сегмент пользователей.
  - Проанализировать продуктовую линейку и выяснить причину, почему пользователи е переходят в разряд постоянных покупателей.

Ввод [ ]: