

None

# Когортный анализ пользователей развлекательного приложения Procrastinate Pro+

## Основная цель:

- Разобраться в причинах убытков и помочь компании выйти в плюс

## Сопутствующие вопросы:

- откуда приходят пользователи и какими устройствами они пользуются
- сколько стоит привлечение пользователей из различных рекламных каналов
- сколько денег приносит каждый клиент
- когда расходы на привлечение клиента окупаются
- какие факторы мешают привлечению клиентов

```
In [1]: # импорт библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from datetime import datetime, timedelta
```

## Предобработка

Разберем таблицу с информацией о расходах на рекламу "costs"

```
In [3]: # отобразим основную информацию
display(costs.head());
print('*****')
display(costs.info());
print("*****")
# посчитаем количество полных дубликотов
print('Количество полных дубликатов равно:', costs.duplicated().sum())
```

	dt	Channel	costs
0	2019-05-01	FaceBoom	113.3
1	2019-05-02	FaceBoom	78.1
2	2019-05-03	FaceBoom	85.8
3	2019-05-04	FaceBoom	136.4
4	2019-05-05	FaceBoom	122.1

```
*****
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1800 entries, 0 to 1799
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype  
 ---  --       --           --      
 0   dt        1800 non-null    object 
 1   Channel   1800 non-null    object 
 2   costs     1800 non-null    float64
dtypes: float64(1), object(2)
memory usage: 42.3+ KB
None
*****
```

Количество полных дубликатов равно: 0

- Пропусков и дубликатов нет, исправим названия полей и приведем дату к нужному формату

In [4]: `# напишем функцию для корректировки названий полей`

```
def corrected_columns(list_col):
    try:
        columns = []
        for i in list_col:
            columns.append(i.lower().strip().replace(' ', '_'))
        return(columns)
    except: print('err')
```

In [5]: `# исправим`

```
costs.columns = corrected_columns(costs.columns)

# приведем формат
costs['dt'] = pd.to_datetime(costs['dt']).dt.date

# проверим
display(costs.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1800 entries, 0 to 1799
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype  
 ---  --       --           --      
 0   dt        1800 non-null    object 
 1   channel   1800 non-null    object 
 2   costs     1800 non-null    float64
dtypes: float64(1), object(2)
memory usage: 42.3+ KB
None
```

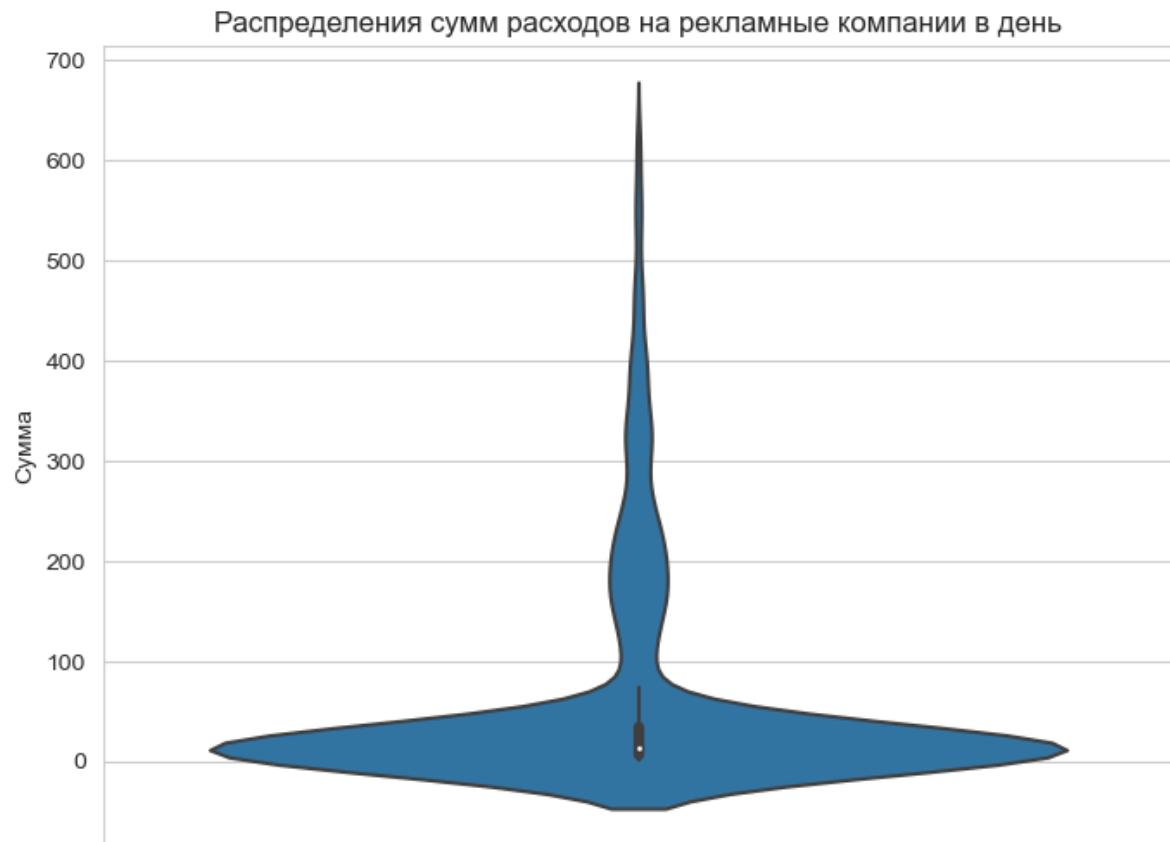
In [6]: `# проверим неявные дубликаты`

```
display(costs['channel'].unique())
```

```
array(['FaceBoom', 'MediaTornado', 'RocketSuperAds', 'TipTop', 'YRabbit',
       'AdNonSense', 'LeapBob', 'OppleCreativeMedia', 'WahooNetBanner',
       'lambdaMediaAds'], dtype=object)
```

In [7]:

```
# посмотрим на распределение сумм расходов на рекламные компании
sns.set_style('whitegrid')
plt.figure(figsize=(8,6))
sns.violinplot(y='costs', data=costs)
plt.title('Распределения сумм расходов на рекламные компании в день')
plt.xlabel('')
plt.ylabel('Сумма')
plt.show();
print("*****")
display(costs['costs'].describe())
```



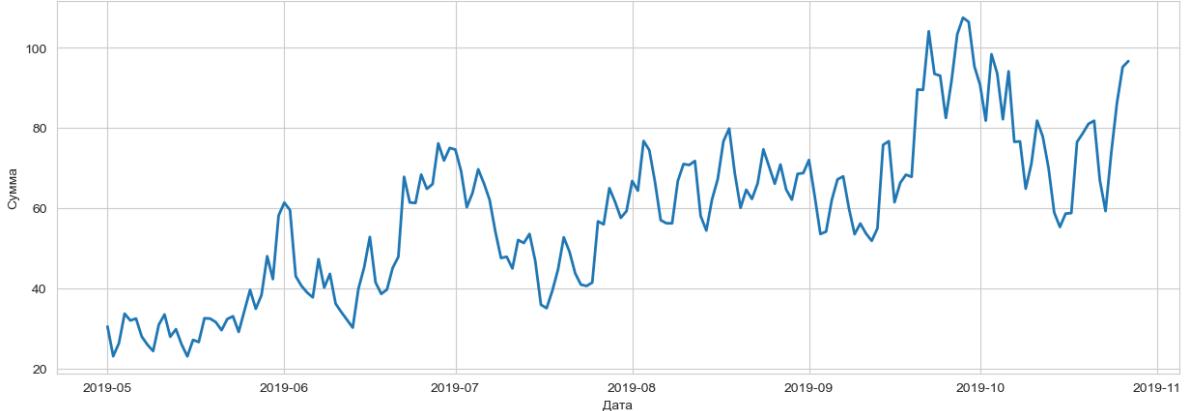
```
*****
count    1800.000000
mean      58.609611
std       107.740223
min       0.800000
25%      6.495000
50%     12.285000
75%     33.600000
max      630.000000
Name: costs, dtype: float64
```

- аномалий нет, средняя сумма расходов в день порядка 12\$

In [8]:

```
# проверим динамику суммы расходов и диапозон дат
costs.groupby(by=costs['dt'])['costs'].mean().plot(figsize=(15,5), linewidth=2)
plt.title('Динамика средней суммы рекламной компании')
plt.xlabel('Дата')
plt.ylabel('Сумма')
plt.show()
```

Динамика средней суммы рекламной компании



- проблем с датами нет диапазон между 2019-05 и 2019-11, расходы на рекламу циклически растут

## Разберем таблицу с информацией о заказах "orders"

In [9]:

```
# отобразим основную информацию
display(orders.head());
print('*****')
display(orders.info());
print("*****")
# посчитаем количество полных дубликотов
print('Количество полных дубликатов равно:', orders.duplicated().sum())
```

	User Id	Event Dt	Revenue
0	188246423999	2019-05-01 23:09:52	4.99
1	174361394180	2019-05-01 12:24:04	4.99
2	529610067795	2019-05-01 11:34:04	4.99
3	319939546352	2019-05-01 15:34:40	4.99
4	366000285810	2019-05-01 13:59:51	4.99

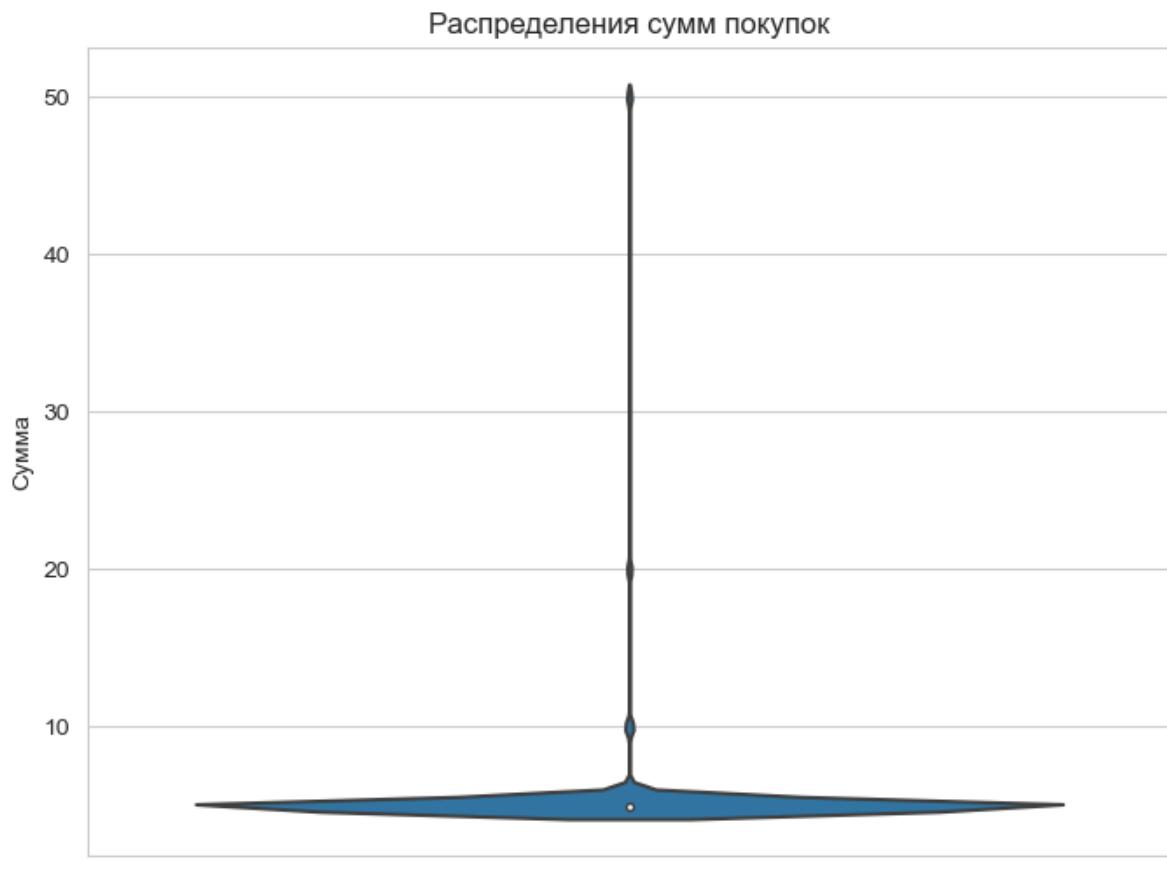
```
*****
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40212 entries, 0 to 40211
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --    
 0   User Id    40212 non-null   int64  
 1   Event Dt   40212 non-null   object 
 2   Revenue    40212 non-null   float64 
dtypes: float64(1), int64(1), object(1)
memory usage: 942.6+ KB
None
*****
Количество полных дубликатов равно: 0
```

- Пропусков и дубликатов нет, исправим названия полей и приведем дату к нужному формату

```
In [10]: # исправим  
orders.columns = corrected_columns(orders.columns)  
  
# приведем формат  
orders['event_dt'] = pd.to_datetime(orders['event_dt'])  
  
# проверим  
display(orders.info())
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 40212 entries, 0 to 40211  
Data columns (total 3 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          --          --  
 0   user_id     40212 non-null   int64    
 1   event_dt    40212 non-null   datetime64[ns]  
 2   revenue     40212 non-null   float64  
dtypes: datetime64[ns](1), float64(1), int64(1)  
memory usage: 942.6 KB  
None
```

```
In [11]: # посмотрим на распределение сумм покупок  
sns.set_style('whitegrid')  
plt.figure(figsize=(8,6))  
sns.violinplot(y='revenue', data=orders)  
plt.title('Распределения сумм покупок')  
plt.xlabel('')  
plt.ylabel('Сумма')  
plt.show();  
print("*****")  
display(orders['revenue'].describe())
```



```
count      40212.000000
mean       5.370608
std        3.454208
min        4.990000
25%       4.990000
50%       4.990000
75%       4.990000
max       49.990000
Name: revenue, dtype: float64
```

- основная часть заказов стоит около 5, также мы видим незначительное количество заказов в виде выбросов выше 10, возможно это долгосрочные подписки.
- Максимальна сумма заказа около 50

```
In [12]: orders.drop_duplicates(subset='user_id', keep=False).groupby(by=orders['event_dt']).size()
```

```
Out[12]: event_dt
2019-05-01    10
2019-05-02     5
2019-05-03     6
2019-05-04    17
2019-05-05     7
...
2019-10-27    45
2019-10-28    15
2019-10-29    11
2019-10-30     4
2019-10-31     7
Name: user_id, Length: 184, dtype: int64
```

```
In [13]: # проверим динамику и диапазон дат
orders.drop_duplicates(subset='user_id', keep=False).groupby(by=orders['event_dt']).size()
plot(figsize=(15,5), legend=False, linewidth=2)
plt.title('Динамика количества уникальных покупателей')
plt.xlabel('Дата')
plt.ylabel('Количество')
plt.show();
print('*****')
print('Количество уникальных покупателей:', len(orders.user_id.unique()))
print("Диапозон дат от: ", orders['event_dt'].dt.date.min(), ', до:', orders['event_dt'].dt.date.max())
```



```
*****
Количество уникальных покупателей: 8881
Диапозон дат от: 2019-05-01 , до: 2019-10-31
```

- количество уникальных покупателей имеет незначительный рост, диапазон дат с 2019-05-01 по 2019-10-31. Аномалий нет

## Проверим таблицу, которая хранит лог сервера с информацией о посещениях сайта "visits"

In [14]:

```
# отобразим основную информацию
display(visits.head());
print('*****')
display(visits.info());
print("*****")
# посчитаем количество полных дубликатов
print('Количество полных дубликатов равно:', visits.duplicated().sum())
```

	User Id	Region	Device	Channel	Session Start	Session End
0	981449118918	United States	iPhone	organic	2019-05-01 02:36:01	2019-05-01 02:45:01
1	278965908054	United States	iPhone	organic	2019-05-01 04:46:31	2019-05-01 04:47:35
2	590706206550	United States	Mac	organic	2019-05-01 14:09:25	2019-05-01 15:32:08
3	326433527971	United States	Android	TipTop	2019-05-01 00:29:59	2019-05-01 00:54:25
4	349773784594	United States	Mac	organic	2019-05-01 03:33:35	2019-05-01 03:57:40

\*\*\*\*\*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309901 entries, 0 to 309900
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   User Id          309901 non-null   int64  
 1   Region           309901 non-null   object  
 2   Device            309901 non-null   object  
 3   Channel           309901 non-null   object  
 4   Session Start     309901 non-null   object  
 5   Session End       309901 non-null   object  
dtypes: int64(1), object(5)
memory usage: 14.2+ MB
None
*****
```

Количество полных дубликатов равно: 0

- Пропусков и дубликатов нет, исправим названия полей и приведем дату к нужному формату

In [15]:

```
# исправим
visits.columns = corrected_columns(visits.columns)

# приведем формат
visits['session_start'] = pd.to_datetime(visits['session_start'])
visits['session_end'] = pd.to_datetime(visits['session_end'])

# проверим
display(visits.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309901 entries, 0 to 309900
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   user_id          309901 non-null   int64  
 1   region           309901 non-null   object  
 2   device            309901 non-null   object  
 3   channel           309901 non-null   object  
 4   session_start     309901 non-null   datetime64[ns]
 5   session_end       309901 non-null   datetime64[ns]
dtypes: datetime64[ns](2), int64(1), object(3)
memory usage: 14.2+ MB
None
```

In [16]: # проверим поле region на неявные дубликаты и ошибки

```
print(visits.region.unique())
['United States' 'UK' 'France' 'Germany']
```

In [17]: # проверим поле device на неявные дубликаты и ошибки

```
print(visits.device.unique())
['iPhone' 'Mac' 'Android' 'PC']
```

In [18]: # проверим поле region на неявные дубликаты и ошибки

```
print(visits.channel.unique())
['organic' 'TipTop' 'RocketSuperAds' 'YRabbit' 'FaceBoom' 'MediaTornado'
 'AdNonSense' 'LeapBob' 'WahooNetBanner' 'OppleCreativeMedia'
 'lambdaMediaAds']
```

- дубликатов и ошибок нет

In [19]: # проверим динамику и диапазон дат

```
print("Диапазон дат начала сессий от: ",visits['session_start'].dt.date.astype('date'))
print("Диапазон дат начала сессий от: ",visits['session_end'].dt.date.astype('date'))
```

Диапазон дат начала сессий от: 2019-05-01 00:00:00 , до: 2019-10-31 00:00:00  
Диапазон дат начала сессий от: 2019-05-01 00:00:00 , до: 2019-11-01 00:00:00

- аномальных значений нет

In [20]: # проверим динамику и количества уникальных посетителей

```
visits.drop_duplicates(subset=['user_id'], keep=False).groupby(by=visits['session_start'])
plt.title('Динамика количества уникальных посетителей')
plt.xlabel('Дата')
plt.ylabel('Количество')
plt.show();
print('*****')
print('Количество уникальных посетителей:',len(visits.user_id.unique()))
```



- количество уникальных посетителей имеет незначительный циклический рост, что может быть связано с проводимыми рекламными компаниями

**После предобработки данных явных проблем не выявлено, диапазоны дат совпадают, дубликатов нет. Форматы дат приведены к верному значению, названия полей стандартизированы**

**Подготовим функции для расчёта и анализа LTV, ROI, удержания и конверсии**

**get\_profiles() — для создания профилей пользователей**

```
In [21]: # функция для создания пользовательских профилей

def get_profiles(sessions, orders, ad_costs):

    # находим параметры первых посещений
    profiles = (
        sessions.sort_values(by=['user_id', 'session_start'])
        .groupby('user_id')
        .agg(
            {
                'session_start': 'first',
                'channel': 'first',
                'device': 'first',
                'region': 'first',
            }
        )
        .rename(columns={'session_start': 'first_ts'})
        .reset_index()
    )

    # для когортного анализа определяем дату первого посещения
```

```

# и первый день месяца, в который это посещение произошло
profiles['dt'] = profiles['first_ts'].dt.date
profiles['month'] = profiles['first_ts'].dt.date.astype('datetime64[M]')

# добавляем признак платящих пользователей
profiles['payer'] = profiles['user_id'].isin(orders['user_id'].unique())

# считаем количество уникальных пользователей
# с одинаковыми источником и датой привлечения
new_users = (
    profiles.groupby(['dt', 'channel'])
    .agg({'user_id': 'nunique'})
    .rename(columns={'user_id': 'unique_users'})
    .reset_index()
)

# объединяем траты на рекламу и число привлечённых пользователей
ad_costs = ad_costs.merge(new_users, on=['dt', 'channel'], how='left')

# делим рекламные расходы на число привлечённых пользователей
ad_costs['acquisition_cost'] = ad_costs['costs'] / ad_costs['unique_users']

# добавляем стоимость привлечения в профили
profiles = profiles.merge(
    ad_costs[['dt', 'channel', 'acquisition_cost']],
    on=['dt', 'channel'],
    how='left',
)
# стоимость привлечения органических пользователей равна нулю
profiles['acquisition_cost'] = profiles['acquisition_cost'].fillna(0)
return profiles

```

## get\_retention() — для подсчёта Retention Rate

In [22]: # функция для расчёта удержания

```

def get_retention_new(
    profiles,
    sessions,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
    cogort_size = 'D',
    locker = ['', '']
):
    #Locked block
    try:
        failed = locker[0]
        value = locker[1]
        if not value + failed == '':
            profiles = profiles.loc[profiles[failed]==value].copy()
    except:print('inter faild name and locked value')

    #create cogort data size filed
    profiles[cogort_size] = profiles.first_ts.astype(object).astype(F"datetime64[{cogort_size}]")
    # добавляем столбец payer в передаваемый dimensions список
    dimensions = ['payer'] + dimensions

    # исключаем пользователей, не «доживших» до горизонта анализа
    last_suitable_acquisition_date = observation_date

```

```

if not ignore_horizon:
    last_suitable_acquisition_date = observation_date - timedelta(
        days=horizon_days - 1
    )
result_raw = profiles.query('dt <= @last_suitable_acquisition_date')

# собираем «сырые» данные для расчёта удержания
result_raw = result_raw.merge(
    sessions[['user_id', 'session_start']], on='user_id', how='left'
)
result_raw['lifetime'] = result_raw['session_start'] - result_raw['first_ts']
result_raw['lifetime'] = result_raw['lifetime']//np.timedelta64(1,cogort_size)

# функция для группировки таблицы по желаемым признакам
def group_by_dimensions(df, dims, horizon_days):
    result = df.pivot_table(
        index=dims, columns='lifetime', values='user_id', aggfunc='nunique'
    )
    cohort_sizes = (
        df.groupby(dims)
        .agg({'user_id': 'nunique'})
        .rename(columns={'user_id': 'cohort_size'})
    )
    result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)
    result = result.div(result['cohort_size'], axis=0)
    result = result[['cohort_size']] + list(range(horizon_days))
    result['cohort_size'] = cohort_sizes
    return result

# получаем таблицу удержания
result_grouped = group_by_dimensions(result_raw, dimensions, horizon_days)

# получаем таблицу динамики удержания
result_in_time = group_by_dimensions(
    result_raw, dimensions + [cogort_size], horizon_days
)

# возвращаем обе таблицы и сырье данные
return result_raw, result_grouped, result_in_time

```

## get\_conversion() — для подсчёта конверсии

In [23]: # функция для расчёта конверсии

```

def get_conversion_new(
    profiles,
    purchases,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
    cogort_size = 'D',
    locker = ['', '']
):

    #Locked block
    try:
        failed = locker[0]
        value = locker[1]
        if not value + failed == '':
            profiles = profiles.loc[profiles[failed]==value].copy()
    except:print('inter faild name and locked value')

```

```

# create cogort data size filed
profiles[cogort_size] = profiles.first_ts.astype(object).astype("datetime64[")
# исключаем пользователей, не «доживших» до горизонта анализа
last_suitable_acquisition_date = observation_date
if not ignore_horizon:
    last_suitable_acquisition_date = observation_date - timedelta(
        days=horizon_days - 1
    )
result_raw = profiles.query('dt <= @last_suitable_acquisition_date')

# определяем дату и время первой покупки для каждого пользователя
first_purchases = (
    purchases.sort_values(by=['user_id', 'event_dt'])
    .groupby('user_id')
    .agg({'event_dt': 'first'})
    .reset_index()
)

# добавляем данные о покупках в профили
result_raw = result_raw.merge(
    first_purchases[['user_id', 'event_dt']], on='user_id', how='left'
)

# рассчитываем лайфтайм для каждой покупки
result_raw['lifetime'] = result_raw['event_dt'] - result_raw['first_ts']
result_raw['lifetime'] = result_raw['lifetime'] // np.timedelta64(1, cogort_size)

# группируем по cohort, если в dimensions ничего нет
if len(dimensions) == 0:
    result_raw['cohort'] = 'All users'
    dimensions = dimensions + ['cohort']

# функция для группировки таблицы по желаемым признакам
def group_by_dimensions(df, dims, horizon_days):
    result = df.pivot_table(
        index=dims, columns='lifetime', values='user_id', aggfunc='nunique'
    )
    result = result.fillna(0).cumsum(axis = 1)
    cohort_sizes = (
        df.groupby(dims)
        .agg({'user_id': 'nunique'})
        .rename(columns={'user_id': 'cohort_size'})
    )
    result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)
    # делим каждую «ячейку» в строке на размер когорты
    # и получаем conversion rate
    result = result.div(result['cohort_size'], axis=0)
    result = result[['cohort_size'] + list(range(horizon_days))]
    result['cohort_size'] = cohort_sizes
    return result

# получаем таблицу конверсии
result_grouped = group_by_dimensions(result_raw, dimensions, horizon_days)

# для таблицы динамики конверсии убираем 'cohort' из dimensions
if 'cohort' in dimensions:
    dimensions = []

# получаем таблицу динамики конверсии
result_in_time = group_by_dimensions(
    result_raw, dimensions + [cogort_size], horizon_days
)

```

```
# возвращаем обе таблицы и сырье данные
return result_raw, result_grouped, result_in_time
```

## get\_ltv() — для подсчёта LTV

In [24]: # функция для расчёта LTV и ROI

```
def get_ltv_new(
    profiles,
    purchases,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
    cogort_size = 'D',
    locker = ['', '']
):

    #Locked block
    try:
        failed = locker[0]
        value = locker[1]
        if not value + failed == '':
            profiles = profiles.loc[profiles[failed]==value].copy()
    except:print('inter faild name and locked value')

    #create cogort data size filed
    profiles[cogort_size] = profiles.first_ts.astype(object).astype("datetime64[{}")
    # исключаем пользователей, не «доживших» до горизонта анализа
    last_suitable_acquisition_date = observation_date
    if not ignore_horizon:
        last_suitable_acquisition_date = observation_date - timedelta(
            days=horizon_days - 1
        )
    result_raw = profiles.query('dt <= @last_suitable_acquisition_date')
    # добавляем данные о покупках в профили
    result_raw = result_raw.merge(
        purchases[['user_id', 'event_dt', 'revenue']], on='user_id', how='left'
    )
    # рассчитываем лайфтайм пользователя для каждой покупки
    result_raw['lifetime'] = result_raw['event_dt'] - result_raw['first_ts']
    result_raw['lifetime'] = result_raw['lifetime']//np.timedelta64(1,cogort_size)
    # группируем по cohort, если в dimensions ничего нет
    if len(dimensions) == 0:
        result_raw['cohort'] = 'All users'
        dimensions = dimensions + ['cohort']

    # функция группировки по желаемым признакам
    def group_by_dimensions(df, dims, horizon_days):
        # строим «треугольную» таблицу выручки
        result = df.pivot_table(
            index=dims, columns='lifetime', values='revenue', aggfunc='sum'
        )
        # находим сумму выручки с накоплением
        result = result.fillna(0).cumsum(axis=1)
        # вычисляем размеры когорт
        cohort_sizes = (
            df.groupby(dims)
            .agg({'user_id': 'nunique'})
            .rename(columns={'user_id': 'cohort_size'})
        )
        # объединяем размеры когорт и таблицу выручки
```

```

result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)
# считаем LTV: делим каждую «ячейку» в строке на размер когорты
result = result.div(result['cohort_size'], axis=0)
# исключаем все лайфтаймы, превышающие горизонт анализа
result = result[['cohort_size'] + list(range(horizon_days))]
# восстанавливаем размеры когорт
result['cohort_size'] = cohort_sizes

# собираем датафрейм с данными пользователей и значениями САС,
# добавляя параметры из dimensions
cac = df[['user_id', 'acquisition_cost'] + dims].drop_duplicates()

# считаем средний САС по параметрам из dimensions
cac = (
    cac.groupby(dims)
    .agg({'acquisition_cost': 'mean'})
    .rename(columns={'acquisition_cost': 'cac'})
)

# считаем ROI: делим LTV на САС
roi = result.div(cac['cac'], axis=0)

# удаляем строки с бесконечным ROI
roi = roi[~roi['cohort_size'].isin([np.inf])]

# восстанавливаем размеры когорт в таблице ROI
roi['cohort_size'] = cohort_sizes

# добавляем САС в таблицу ROI
roi['cac'] = cac['cac']

# в финальной таблице оставляем размеры когорт, САС
# и ROI в лайфтаймы, не превышающие горизонт анализа
roi = roi[['cohort_size', 'cac'] + list(range(horizon_days))]

# возвращаем таблицы LTV и ROI
return result, roi

# получаем таблицы LTV и ROI
result_grouped, roi_grouped = group_by_dimensions(
    result_raw, dimensions, horizon_days
)

# для таблиц динамики убираем 'cohort' из dimensions
if 'cohort' in dimensions:
    dimensions = []

# получаем таблицы динамики LTV и ROI
result_in_time, roi_in_time = group_by_dimensions(
    result_raw, dimensions + [cogort_size], horizon_days
)

return (
    result_raw, # сырье данные
    result_grouped, # таблица LTV
    result_in_time, # таблица динамики LTV
    roi_grouped, # таблица ROI
    roi_in_time, # таблица динамики ROI
)

```

## filter\_data() — для сглаживания данных

In [25]: # функция для сглаживания фрейма

```
def filter_data(df, window):
    # для каждого столбца применяем скользящее среднее
    for column in df.columns.values:
        df[column] = df[column].rolling(window).mean()
    return df
```

## для построения графика Retention Rate

In [26]: # функция для визуализации удержания

```
def plot_retention(retention, retention_history, horizon, window=7, cogort_size = 'All users'):

    # задаём размер сетки для графиков
    plt.figure(figsize=(15, 10))

    # исключаем размеры когорт и удержание первого дня
    retention = retention.drop(columns=['cogort_size', 0])
    # в таблице динамики оставляем только нужный лайфтайм
    retention_history = retention_history.drop(columns=['cogort_size'])[
        [horizon - 1]
    ]

    # если в индексах таблицы удержания только payer,
    # добавляем второй признак – cohort
    if retention.index.nlevels == 1:
        retention['cohort'] = 'All users'
        retention = retention.reset_index().set_index(['cohort', 'payer'])

    # в таблице графиков – два столбца и две строки, четыре ячейки
    # в первой строим кривые удержания платящих пользователей
    ax1 = plt.subplot(2, 2, 1)
    retention.query('payer == True').droplevel('payer').T.plot(
        grid=True, ax=ax1, cmap='tab20', linewidth=2
    )
    plt.legend()
    plt.legend(frameon=True, ncol=3)
    plt.xlabel('Лайфтайм')
    plt.title('Удержание платящих пользователей')

    # во второй ячейке строим кривые удержания неплатящих
    # вертикальная ось – от графика из первой ячейки
    ax2 = plt.subplot(2, 2, 2, sharey=ax1)
    retention.query('payer == False').droplevel('payer').T.plot(
        grid=True, ax=ax2, cmap='tab20', linewidth=2
    )
    plt.legend()
    plt.legend(frameon=True, ncol=3)
    plt.xlabel('Лайфтайм')
    plt.title('Удержание неплатящих пользователей')

    # в третьей ячейке – динамика удержания платящих
    ax3 = plt.subplot(2, 2, 3)
    # получаем названия столбцов для сводной таблицы
    columns = [
        name
        for name in retention_history.index.names
        if name not in [cogort_size, 'payer']
    ]
    # фильтруем данные и строим график
```

```

filtered_data = retention_history.query('payer == True').pivot_table(
    index=cogort_size, columns=columns, values=horizon - 1, aggfunc='mean'
)
filter_data(filtered_data, window).plot(grid=True, ax=ax3, cmap='tab20', linewidth=2)
plt.legend(frameon=True, ncol=3)
plt.xlabel('Дата привлечения')
plt.title(
    'Динамика удержания платящих пользователей на {}-й {}'.format(
        horizon, cogort_size
    )
)

# В чётвертой ячейке – динамика удержания неплатящих
ax4 = plt.subplot(2, 2, 4, sharey=ax3)
# фильтруем данные и строим график
filtered_data = retention_history.query('payer == False').pivot_table(
    index=cogort_size, columns=columns, values=horizon - 1, aggfunc='mean'
)
filter_data(filtered_data, window).plot(grid=True, ax=ax4, cmap='tab20', linewidth=2)
plt.legend(frameon=True, ncol=3)
plt.xlabel('Дата привлечения')
plt.title(
    'Динамика удержания неплатящих пользователей на {}-й {}'.format(
        horizon, cogort_size
    )
)

plt.tight_layout()
plt.show()

```

## plot\_conversion() — для построения графика конверсии

In [27]: # функция для визуализации конверсии

```

def plot_conversion(conversion, conversion_history, horizon, window=7, cogort_size : int):

    # задаём размер сетки для графиков
    plt.figure(figsize=(15, 5))

    # исключаем размеры когорт
    conversion = conversion.drop(columns=['cohort_size'])
    # в таблице динамики оставляем только нужный лайфтайм
    conversion_history = conversion_history.drop(columns=['cohort_size'])[
        [horizon - 1]
    ]

    # первый график – кривые конверсии
    ax1 = plt.subplot(1, 2, 1)
    conversion.T.plot(grid=True, ax=ax1, cmap='tab20', linewidth=2)
    plt.legend()
    plt.legend(frameon=True, ncol=3)
    plt.xlabel('Лайфтайм')
    plt.title('Конверсия пользователей')

    # второй график – динамика конверсии
    ax2 = plt.subplot(1, 2, 2, sharey=ax1)
    columns = [
        # столбцами сводной таблицы станут все столбцы индекса, кроме даты
        name for name in conversion_history.index.names if name not in [cogort_size]
    ]

```

```

    filtered_data = conversion_history.pivot_table(
        index=cogort_size, columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax2, cmap='tab20', linewidth=2)
    plt.xlabel('Дата привлечения')
    plt.title('Динамика конверсии пользователей на {}-й {}'.format(horizon, cogort_size))

    plt.tight_layout()
    plt.legend(frameon=True, ncol=3)
    plt.show()

```

## plot\_ltv\_roi — для визуализации LTV и ROI

```

In [28]: # функция для визуализации LTV и ROI

def plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon, window=7, cogort_size):

    # задаём сетку отрисовки графиков
    plt.figure(figsize=(20, 10))

    # из таблицы ltv исключаем размеры когорт
    ltv = ltv.drop(columns=['cohort_size'])
    # в таблице динамики ltv оставляем только нужный лайфтайм
    ltv_history = ltv_history.drop(columns=['cohort_size'])[[horizon - 1]]

    # стоимость привлечения запишем в отдельный фрейм
    cac_history = roi_history[['cac']]

    # из таблицы roi исключаем размеры когорт и cac
    roi = roi.drop(columns=['cohort_size', 'cac'])
    # в таблице динамики roi оставляем только нужный лайфтайм
    roi_history = roi_history.drop(columns=['cohort_size', 'cac'])[
        [horizon - 1]
    ]

    # первый график – кривые ltv
    ax1 = plt.subplot(2, 3, 1)
    ltv.T.plot(grid=True, ax=ax1, linewidth=2, cmap='tab20')
    plt.legend()
    plt.legend(frameon=True, ncol=3)
    plt.xlabel('Лайфтайм')
    plt.title('LTV')

    # второй график – динамика ltv
    ax2 = plt.subplot(2, 3, 2, sharey=ax1)
    # столбцами сводной таблицы станут все столбцы индекса, кроме даты
    columns = [name for name in ltv_history.index.names if name not in [cogort_size]]
    filtered_data = ltv_history.pivot_table(
        index=cogort_size, columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax2, linewidth=2, cmap='tab20')
    plt.xlabel('Дата привлечения')
    plt.legend(frameon=True, ncol=3)
    plt.title('Динамика LTV пользователей на {}-й {}'.format(horizon, cogort_size))

    # третий график – динамика cac
    ax3 = plt.subplot(2, 3, 3, sharey=ax1)
    # столбцами сводной таблицы станут все столбцы индекса, кроме даты
    columns = [name for name in cac_history.index.names if name not in [cogort_size]]
    filtered_data = cac_history.pivot_table(
        index=cogort_size, columns=columns, values='cac', aggfunc='mean'
    )

```

```

filter_data(filtered_data, window).plot(grid=True, ax=ax3, linewidth=2, cmap='tab10')
plt.xlabel('Дата привлечения')
plt.legend(frameon=True, ncol=3)
plt.title('Динамика стоимости привлечения пользователей')

# четвёртый график – кривые roi
ax4 = plt.subplot(2, 3, 4)
roi.T.plot(grid=True, ax=ax4, linewidth=2, cmap='tab20')
plt.axhline(y=1, color='red', linestyle='--', label='Уровень окупаемости', linewidth=2)
plt.legend()
plt.legend(frameon=True, ncol=3)
plt.xlabel('Лайфтайм')
plt.title('ROI')

# пятый график – динамика roi
ax5 = plt.subplot(2, 3, 5, sharey=ax4)
# столбцами сводной таблицы станут все столбцы индекса, кроме даты
columns = [name for name in roi_history.index.names if name not in [cogort_size]]
filtered_data = roi_history.pivot_table(
    index=cogort_size, columns=columns, values=horizon - 1, aggfunc='mean'
)
filter_data(filtered_data, window).plot(grid=True, ax=ax5, linewidth=2, cmap='tab10')
plt.axhline(y=1, color='red', linestyle='--', label='Уровень окупаемости', linewidth=2)
plt.xlabel('Дата привлечения')
plt.title('Динамика ROI пользователей на {}-й {}'.format(horizon, cogort_size))
plt.legend(frameon=True, ncol=3)
plt.tight_layout()
plt.show()

```

## add\_value\_labels - добавляет значение в bar и barh plot на столбики

In [29]:

```

# добавляет аннотации к барам
def add_value_labels(ax, bar_or_barh, space=4, fmt=" {:.0f}"):
    """Add labels to the end of each bar in a bar chart.

    Arguments:
        ax (matplotlib.axes.Axes): The matplotlib object containing the axes
            of the plot to annotate.
        spacing (int): The distance between the labels and the bars.
        frm: format annotations
        bar_or_barh: chose bar or barh pyplot
    """
    if bar_or_barh == 'bar':
        # For each bar: Place a Label
        heights = [rect.get_height() for rect in ax.patches]
        for rect in ax.patches:
            # Get X and Y placement of Label from rect.
            y_value = rect.get_height()
            x_value = rect.get_x() + rect.get_width() / 2

            if y_value > 0:
                va = 'bottom'
                label = fmt.format(y_value)
                # Create annotation
                if y_value > max(heights)/2:
                    ax.annotate(
                        label,
                        (x_value, y_value/2),
                        xytext=(0, space),
                        # Use `label` as Label
                        # Place label at end of the bar
                        # Vertically shift Label by `space`

```

```

textcoords="offset points", # Interpret `xytext` as offset
ha='center',               # Horizontally center Label
va=va,
rotation=90,color='black',bbox = dict(boxstyle="round", fc="white", ec="black", lw=1)
else:
    ax.annotate(
        label,                      # Use `label` as label
        (x_value, rect.get_y()+rect.get_height()),      # Place
        xytext=(0, space),           # Vertically shift Label by `space`
        textcoords="offset points", # Interpret `xytext` as offset
        ha='center',                # Horizontally center Label
        va=va,
        rotation=90,size = 10)
elif y_value == 0:
    va = 'top'
    label = fmt.format(y_value)
    ax.annotate(
        label,                      # Use `label` as label
        (x_value, rect.get_y()+rect.get_height()),      # Place
        xytext=(0, space),           # Vertically shift Label by `space`
        textcoords="offset points", # Interpret `xytext` as offset
        ha='center',                # Horizontally center Label
        va='center',
        rotation=90,size = 10)
else:
    # Invert space to place Label below
    # Vertically align Label at top
    va = 'top'
    # Use Y value as label and format number with one decimal place
    label = fmt.format(y_value)
    # Create annotation
    if y_value < min(heights)/2:
        ax.annotate(
            label,                      # Use `label` as label
            (x_value, y_value/2),       # Place Label at end of the bar
            xytext=(0, 0),             # Vertically shift Label by `space`
            textcoords="offset points", # Interpret `xytext` as offset
            ha='center',                # Horizontally center Label
            va=va,
            rotation=90,color='black',bbox = dict(boxstyle="round", fc="white", ec="black", lw=1)
            # positive and negative values
    else:
        ax.annotate(
            label,                      # Use `label` as label
            (x_value, rect.get_y()+rect.get_height()),      # Place
            xytext=(0, 0),           # Vertically shift Label by `space`
            textcoords="offset points", # Interpret `xytext` as offset
            ha='center',                # Horizontally center Label
            va=va,
            rotation=90,size = 10)      # Vertically align Label
            # positive and negative values.
    elif bar_or_barh == 'barh':
        # Получаем ширины столбиков
        widths = [rect.get_width() for rect in ax.patches]
        heights = [rect.get_height() for rect in ax.patches]
        # Добавляем аннотации на каждый столбик
        for i, rect in enumerate(ax.patches):
            # Перемещаем аннотацию внутрь столбика
            if widths[i] < 0:
                if widths[i] < min(widths)/2:
                    space *= -1

```

```

    ax.annotate(fmt.format(widths[i]), xy=(widths[i]/2, rect.get_y()
                                xytext=(0, 0), textcoords="offset points", ha='center')
    else:
        space *= -1
        ax.annotate(fmt.format(widths[i]), xy=(rect.get_x() + rect.get_x(),
                                xytext=(space-3, 0), textcoords="offset points", ha='right')

    elif widths[i] == 0:
        ax.annotate(fmt.format(widths[i]), xy=(widths[i], rect.get_y() + rect.get_y(),
                                xytext=(0, 0), textcoords="offset points", ha='center')

    elif widths[i] > 0:
        if widths[i] > max(widths)/2:
            ax.annotate(fmt.format(widths[i]), xy=(widths[i]/2, rect.get_y()
                                xytext=(0, 0), textcoords="offset points", ha='center')
        else:
            ax.annotate(fmt.format(widths[i]), xy=(rect.get_x() + rect.get_x(),
                                xytext=(space, 0), textcoords="offset points", ha='left')
    else:
        return(print("inter name 'bar or bash'"))

```

## Исследовательский анализ данных

### Составим профили пользователей

```

In [30]: # соберем профили пользователей
profiles = get_profiles(visits, orders, ad_costs = costs)

# проверим
display(profiles.info(), ****)
print("*****")
print('Количество новых пользователей:', len(profiles))
print(f'Границы дат привлечения новых пользователей от: {profiles.month.min().date()}

<class 'pandas.core.frame.DataFrame'>
Int64Index: 150008 entries, 0 to 150007
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   user_id          150008 non-null   int64  
 1   first_ts         150008 non-null   datetime64[ns]
 2   channel          150008 non-null   object  
 3   device            150008 non-null   object  
 4   region            150008 non-null   object  
 5   dt                150008 non-null   object  
 6   month             150008 non-null   datetime64[ns]
 7   payer              150008 non-null   bool    
 8   acquisition_cost  150008 non-null   float64 
dtypes: bool(1), datetime64[ns](2), float64(1), int64(1), object(4)
memory usage: 10.4+ MB
None
'*****'

```

	<b>user_id</b>	<b>first_ts</b>	<b>channel</b>	<b>device</b>	<b>region</b>	<b>dt</b>	<b>month</b>	<b>payer</b>	<b>acquisition_cost</b>
<b>0</b>	599326	2019-05-07 20:58:57	FaceBoom	Mac	United States	2019-05-07	2019-05-01	True	1.088172
<b>1</b>	4919697	2019-07-09 12:46:07	FaceBoom	iPhone	United States	2019-07-09	2019-07-01	False	1.107237
<b>2</b>	6085896	2019-10-01 09:58:33	organic	iPhone	France	2019-10-01	2019-10-01	False	0.000000
<b>3</b>	22593348	2019-08-22 21:35:48	AdNonSense	PC	Germany	2019-08-22	2019-08-01	False	0.988235
<b>4</b>	31989216	2019-10-02 00:07:44	YRabbit	iPhone	United States	2019-10-02	2019-10-01	False	0.230769

\*\*\*\*\*

Количество новых пользователей: 150008

Границы дат привлечения новых пользователей от: 2019-05-01 до: 2019-10-01

**Проверим из каких стран и источников приходят новые пользователи и где выше доля покупателей**

## Срез по странам

```
In [31]: # построим график изменения количества новых пользователей
ax1 = plt.subplot(2,2,1)
sns.heatmap(profiles.pivot_table(index=profiles.month.dt.date,columns='region',values='user_id',aggfunc='count').T,annot=True,fmt = 'd',cbar_kws={'label': 'Количество новых пользователей'},xticklabels=True, yticklabels=True,cmap='magma')
plt.tight_layout(h_pad=-1)
plt.title('Динамика изменения количества новых пользователей')
plt.xlabel('Дата')
plt.yticks(fontsize=10)
plt.ylabel('Регион')
plt.xticks(rotation=0)

# построим график изменения доли покупателей
ax2 = plt.subplot(2,2,2,sharey=ax1)
sns.heatmap(profiles.pivot_table(index=profiles.month.dt.date,columns='region',values='user_id',aggfunc='count').T,annot=True,fmt = '.2%',cbar_kws={'label': 'Количество новых пользователей'},xticklabels=True, yticklabels=True,ax=ax2,cmap='magma')
plt.title('Динамика доли покупателей')
plt.xlabel('Дата')
#plt.yticks(fontsize=10)
#plt.xticks(fontsize=10)
plt.ylabel('Регион')
plt.xticks(rotation=0)

# построим график общего количества привлеченных пользователей
ax3 = plt.subplot(2,2,3)
profiles.pivot_table(columns='region',values='user_id',aggfunc='count').T.sort_values(by='user_id',ascending=False).plot.bar(ax=ax3,figsize=(20,9),cmap='magma',edgecolor='black')
add_value_labels(ax3,'bar',3)
```

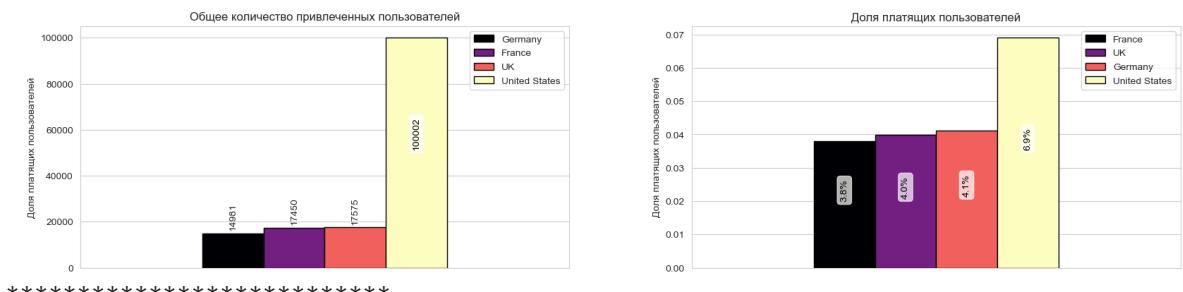
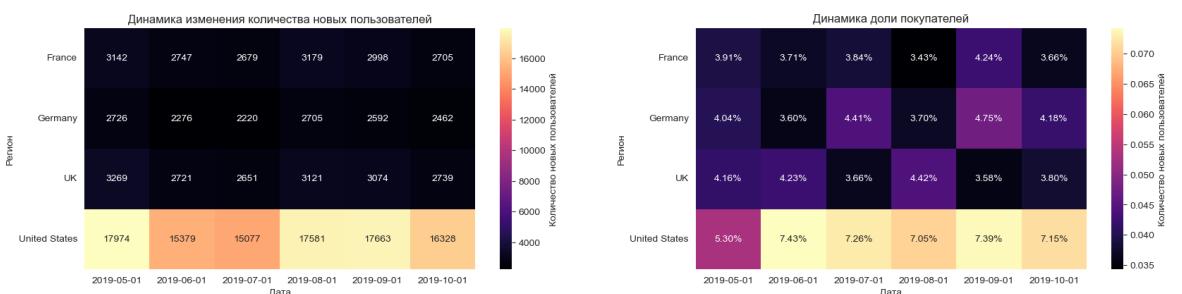
```

plt.title('Общее количество привлеченных пользователей')
plt.xticks([])
plt.ylabel('Доля платящих пользователей')
plt.legend()

# построим график общей доли покупателей
ax4 = plt.subplot(2,2,4)
profiles.pivot_table(columns='region',values='payer',aggfunc='mean').T.sort_values
plot(kind='bar', cmap='magma', ax=ax4, edgecolor='black')
plt.title('Доля платящих пользователей')
add_value_labels(ax4,'bar',3,"{:.1%}")
plt.xticks([])
plt.xticks(rotation=0)
plt.ylabel('Доля платящих пользователей')
plt.legend()
plt.show()

# отобразим соднею таблицу
print('*****')
pivot_region = profiles.pivot_table(index='region',values=[ 'payer'],aggfunc=[ 'count'])
pivot_region.columns = [ 'Регион','Привлеченные пользователи','Доля покупателей']
display(pivot_region.sort_values(by=[ 'Привлеченные пользователи','Доля покупателей'])

```



Регион	Привлеченные пользователи	Доля покупателей
United States	100002	0.069019
UK	17575	0.039829
France	17450	0.037994
Germany	14981	0.041119

- Основная часть новых пользователей - 100000 пришла из Соединённых Штатов. Из Германии, Англии и Франции их поток кратно ниже и равен от 15 до 17,5 тысяч. Так же доля платящих пользователей самая высокая в Соединённых Штатах, она равна порядка 7%, тогда как у пользователей из остальных стран - около 4%. Явной динамики по регионам не выявлено, однако стоит отметить рост в 2% количества платящих пользователей после первого месяца в Соединённых Штатах

## Срез по каналам привлечения

In [32]:

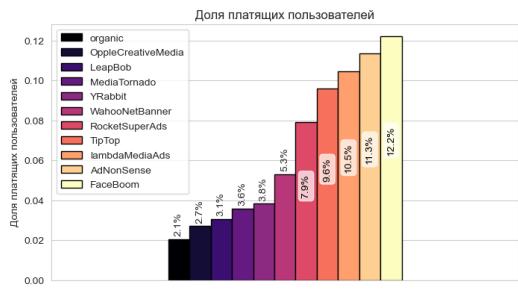
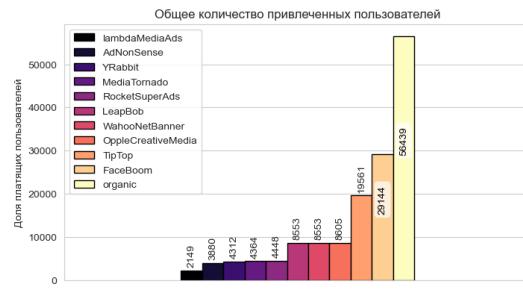
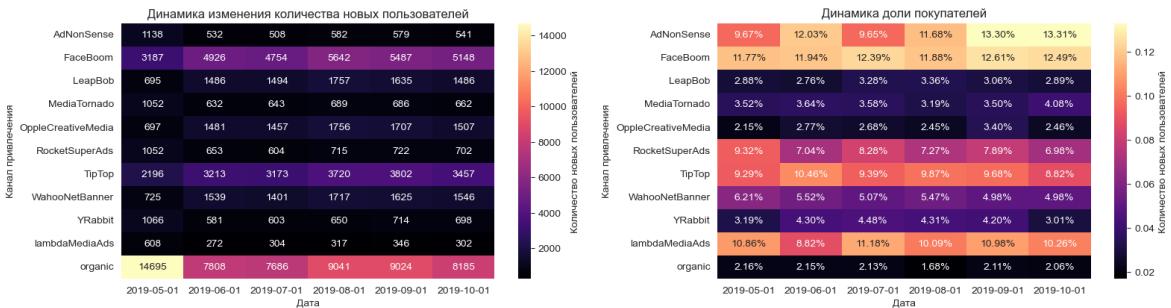
```
# построим график изменения количества новых пользователей
ax1 = plt.subplot(2,2,1)
sns.heatmap(profiles.pivot_table(index=profiles.month.dt.date,columns='channel',values='user_id',aggfunc='count').T.sort_values(ascending=False),annot=True,fmt = 'd',cbar_kws={'label': 'Количество новых пользователей'},xticklabels=True, yticklabels=True,cmap='magma')
plt.tight_layout(h_pad=-1)
plt.title('Динамика изменения количества новых пользователей')
plt.xlabel('Дата')
plt.yticks(fontsize=10)
plt.ylabel('Канал привлечения')
plt.xticks(rotation=0)

# построим график изменения доли покупателей
ax2 = plt.subplot(2,2,2,sharey=ax1)
sns.heatmap(profiles.pivot_table(index=profiles.month.dt.date,columns='channel',values='payer',aggfunc='mean').T.sort_values(ascending=False),annot=True,fmt = '.2%',cbar_kws={'label': 'Количество новых пользователей'},xticklabels=True, yticklabels=True,ax=ax2,cmap='magma')
plt.title('Динамика доли покупателей')
plt.xlabel('Дата')
plt.yticks(fontsize=10)
plt.xticks(fontsize=10)
plt.ylabel('Канал привлечения')
plt.xticks(rotation=0)

# построим график общего количества привлеченных пользователей
ax3 = plt.subplot(2,2,3)
profiles.pivot_table(columns='channel',values='user_id',aggfunc='count').T.sort_values(ascending=False).plot.bar(ax=ax3,figsize=(20,9),cmap='magma',edgecolor='black')
plt.title('Общее количество привлеченных пользователей')
plt.xticks([])
add_value_labels(ax3,'bar',3,"{:.0f}")
plt.ylabel('Доля платящих пользователей')
plt.legend()

# построим график общей доли покупателей
ax4 = plt.subplot(2,2,4)
profiles.pivot_table(columns='channel',values='payer',aggfunc='mean').T.sort_values(ascending=False).plot(kind='bar',cmap='magma',ax=ax4,edgecolor='black')
plt.title('Доля платящих пользователей')
plt.xticks([])
add_value_labels(ax4,'bar',3,"{:.1%}")
plt.xticks(rotation=0)
plt.ylabel('Доля платящих пользователей')
plt.legend()
plt.show()

# отобразим соднею таблицу
print('*****')
pivot_region = profiles.pivot_table(index='channel',values=[ 'payer'],aggfunc=[ 'count','mean'])
pivot_region.columns = [ 'Канал','Привлеченные пользователи','Доля покупателей']
display(pivot_region.sort_values(by=[ 'Привлеченные пользователи','Доля покупателей'],ascending=False))
```



\*\*\*\*\*

Канал	Привлеченные пользователи	Доля покупателей
10 organic	56439	0.020553
1 FaceBoom	29144	0.122049
6 TipTop	19561	0.096007
4 OppleCreativeMedia	8605	0.027077
7 WahooNetBanner	8553	0.052964
2 LeapBob	8553	0.030633
5 RocketSuperAds	4448	0.079137
3 MediaTornado	4364	0.035747
8 YRabbit	4312	0.038265
0 AdNonSense	3880	0.113402
9 lambdaMediaAds	2149	0.104700

- Основная часть пользователей - органические их число равно 56439, однако доля покупателей тут самая низкая и равна 2.1%, далее идет FaceBoom и TipTop с 29144 и 19561 привлеченными пользователями соответственно. Самый низкий приток новых пользователей от AdNonSense и lambdaMediaAds. Самую высокую долю платящих пользователей мы видим в FaceBoom, AdNonSense и lambdaMediaAds со значениями 12.2%, 11.5% и 10.5% соответственно, тогда как OppleCreativeMedia и LeapBob имеют. По динамике мы видим падение в 2 раза числа органических пользователей после первого месяца, что довольно странно. Также можно заметить явный тренд на увеличений количества пользователей из FaceBoom и менее выраженный из TipTop. По динамике доли покупателей мы можем заметить тренды на снижение их числа у TipTop и RocketSuperAds, тогда как FaceBoom и AdNonSense имеют в целом положительную динамику. Можно

**предположить что каналы lambdaMediaAds, FaceBoom и AdNonSense могут быть неплохими точками роста новых клиентов**

## Срез по устройствам

In [33]:

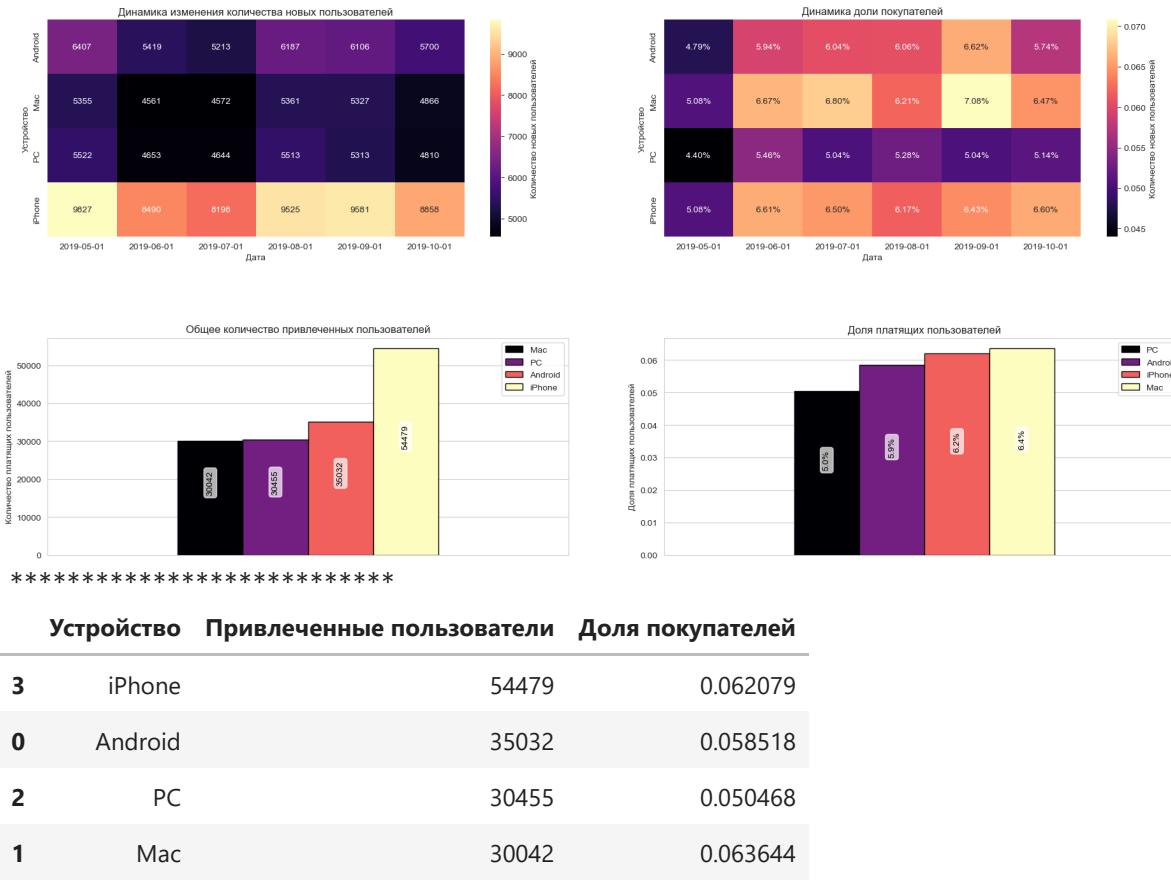
```
# построим график изменения количества новых пользователей
ax1 = plt.subplot(2,2,1)
sns.heatmap(profiles.pivot_table(index=profiles.month.dt.date,columns='device',values='user_id',aggfunc='count').T.sort_values(ascending=False),annot=True,fmt = 'd',cbar_kws={'label': 'Количество новых пользователей'},xticklabels=True, yticklabels=True,cmap='magma')
plt.tight_layout(h_pad=-1)
plt.title('Динамика изменения количества новых пользователей')
plt.xlabel('Дата')
plt.yticks(fontsize=10)
plt.ylabel('Устройство')
plt.xticks(rotation=0)

# построим график изменения доли покупателей
ax2 = plt.subplot(2,2,2,sharey=ax1)
sns.heatmap(profiles.pivot_table(index=profiles.month.dt.date,columns='device',values='payer',aggfunc='mean').T.sort_values(ascending=False),annot=True,fmt = '.2%',cbar_kws={'label': 'Количество новых пользователей'},xticklabels=True, yticklabels=True,ax=ax2,cmap='magma')
plt.title('Динамика доли покупателей')
plt.xlabel('Дата')
plt.yticks(fontsize=10)
plt.xticks(fontsize=10)
plt.ylabel('Устройство')
plt.xticks(rotation=0)

# построим график общего количества привлеченных пользователей
ax3 = plt.subplot(2,2,3)
profiles.pivot_table(columns='device',values='user_id',aggfunc='count').T.sort_values(ascending=False).plot.bar(ax=ax3,figsize=(20,9),cmap='magma',edgecolor='black')
plt.title('Общее количество привлеченных пользователей')
plt.xticks([])
add_value_labels(ax3,'bar',3)
plt.ylabel('Количество привлеченных пользователей')
plt.legend()

# построим график общей доли покупателей
ax4 = plt.subplot(2,2,4)
profiles.pivot_table(columns='device',values='payer',aggfunc='mean').T.sort_values(ascending=False).plot(kind='bar',cmap='magma',ax=ax4,edgecolor='black')
plt.title('Доля платящих пользователей')
plt.xticks([])
add_value_labels(ax4,'bar',3,"{:.1%}")
plt.xticks(rotation=0)
plt.ylabel('Доля платящих пользователей')
plt.legend()
plt.show()

# отобразим соднею таблицу
print('*****')
pivot_region = profiles.pivot_table(index='device',values=[ 'payer'],aggfunc=[ 'count'])
pivot_region.columns = [ 'Устройство','Привлеченные пользователи','Доля покупателей']
display(pivot_region.sort_values(by=[ 'Привлеченные пользователи','Доля покупателей']))
```



- На первом месте по количеству пользователи с iPhone, тогда как MAC их на порядок меньше. Доля покупателей у пользователей продукции Apple самая высокая, пользователи MAC выглядят неплохой точкой роста. Можно заметить пониженную долю покупателей в первом месяце, доля покупателей на PC стабильно ниже на всем периоде анализа

## Вывод по исследовательскому анализу

- Количество новых пользователей за период наблюдения: 150008
- Основная часть новых пользователей - 100000 пришла из Соединённых Штатов. Из Германии, Англии и Франции их поток кратно ниже и равен от 15 до 17,5 тысяч. Так же доля платящих пользователей самая высокая в Соединённых Штатах, она равна порядка 7%, тогда как у пользователей из остальных стран - около 4%. Явной динамики по регионам не выявлено, однако стоит отметить рост в 2% количества платящих пользователей после первого месяца в Соединённых Штатах
- Основная часть пользователей - органические их число равно 56439, однако доля покупателей тут самая низкая и равна 2.1%, далее идет FaceBoom и TipTop с 29144 и 19561 привлеченными пользователями соответственно. Самый низкий приток новых пользователей от AdNonSense и LambdaMediaAds. Самую высокую долю платящих пользователей мы видим в FaceBoom, AdNonSense и LambdaMediaAds со значениями 12.2%, 11.5% и 10.5% соответственно, тогда как OppleCreativeMedia и LeapBob имеют. По динамике мы видим падение в 2 раза числа органических пользователей

после первого месяца, что довольно странно. Также можно заметить явный тренд на увеличения количества пользователей из FaceBoom менее выраженный из TipTop. По динамике доли покупателей мы можем заметить тренды на снижение их числа у TipTop и RocketSuperAds, тогда как FaceBoom и AdNonSense имеют в целом положительную динамику. Можно предположить что каналы LambdaMediaAds, FaceBoom и AdNonSense могут быть неплохими точками роста новых клиентов

- На первом месте по количеству пользователи с Iphone, тогда как MAC их на порядок меньше. Доля покупателей у пользователей продукции Apple самая высокая, пользователи MAC выглядят неплохой точкой роста. Можно заметить пониженную долю покупателей в первом месяце, доля покупателей на РС стабильно ниже на всем периоде анализа

## Маркетинг

### Проверим динамику количества новых пользователей с разбивкой по признаку "payer"

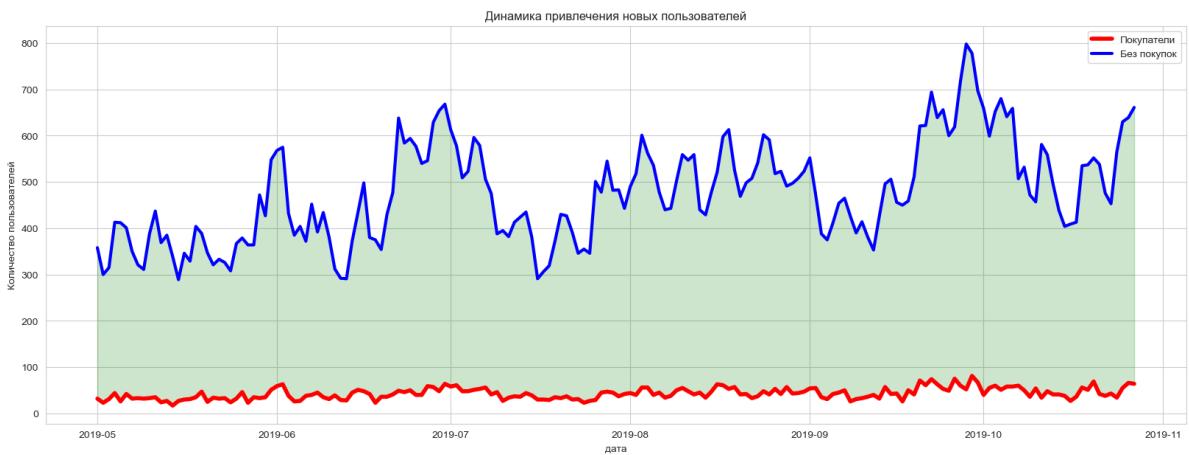
```
In [81]: # создадим график динамики количества уникальных пользователей

ax = profiles.loc[profiles['payer'] == True].groupby(by='dt')['user_id'].count().\
plot(c='r', linewidth=4, title='Динамика привлечения новых пользователей', label='Покупатели')

profiles.loc[profiles['payer'] == False].groupby(by='dt')['user_id'].count().\
plot(ax=ax, figsize=(20, 7), grid=True, color='blue', linewidth=3, label='Без покупок')

plt.xlabel('дата')
plt.ylabel('Количество пользователей')
plt.fill_between(profiles.loc[profiles['payer'] == True].groupby(by='dt')['user_id'],
                profiles.loc[profiles['payer'] == True].groupby(by='dt')['user_id'],
                profiles.loc[profiles['payer'] == False].groupby(by='dt')['user_id'])

plt.show()
print('*****')
print('Общее количество уникальных покупателей за весь период наблюдения:', len(profiles))
print('Общее количество уникальных пользователей без покупок за весь период наблюдения:', len(profiles[profiles['payer'] == False]))
```



```
*****
*****
```

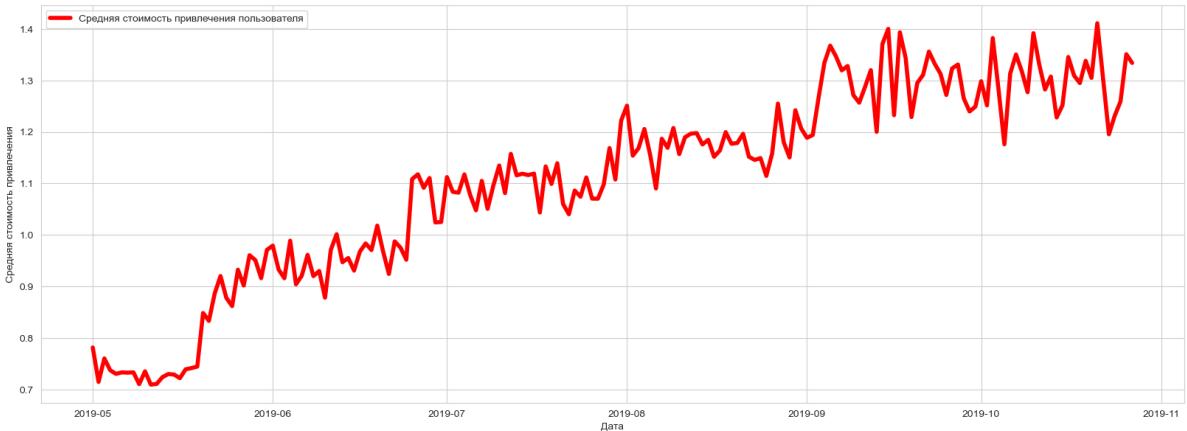
Общее количество унакальных покупателей за весь период наблюдения: 7721  
 Общее количество унакальных пользователей без покупок за весь период наблюдения: 85848

- Явной динамики на увеличение пользователей нет. Количество новых покупателей держится на постоянном, относительно низком уровне, тогда как количество новых пользователей без покупок имеет циклические волны, что может быть реакцией на рекламные компании или сезонность.**
- Возможна проблема с качеством привлекаемых клиентов, доля покупателей среди которых не повторяет тенденции на увеличение в паре с клиентами без покупок в циклах роста.**

## Проверим динамику стоимости привлечения нового пользователя

In [35]:

```
# построим график
profiles.loc[profiles['channel']!='organic'].groupby(by='dt')[['acquisition_cost']].plot(figsize=(20,7),grid=True,color='red',linewidth=4,label='Средняя стоимость привлечения')
plt.xlabel('Дата')
plt.ylabel('Средняя стоимость привлечения')
plt.show()
print('*****')
print("Средняя стоимость превличения нового пользователя равна",round(profiles['acquisition_cost'].mean(),3))
print("Общая сумма рекламных трат равна",round(profiles['acquisition_cost'].sum(),3))
# checking
print(round(costs.costs.sum(),3))
```



```
*****
Средняя стоимость превличения нового пользователя равна 0.703
Общая сумма рекламных трат равна 105497.3
105497.3
```

- Стоимость привлечения одного нового пользователя растет с порядка 0.3 в мае 2019 года до 8.5 в октябре. В совокупности с отсутствием роста новых покупателей, можно предположить о плохой эффективности рекламных компаний. Средняя стоимость привлечения клиента за весь период равна 0.703, общая сумма потраченных на маркетинг средств равна 105497.3.**

# Анализ распределения средств по источникам привлечения

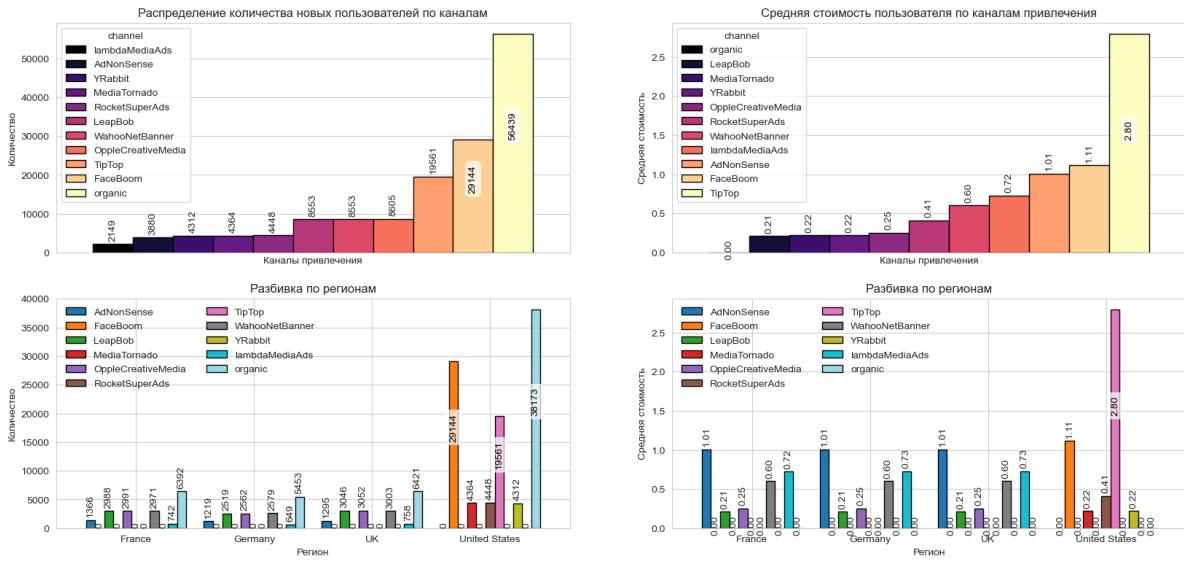
In [36]:

```
# построим график Разбивка по регионам количество
ax1 = plt.subplot(2, 2, 3)
profiles.pivot_table(index='region',columns='channel',values='acquisition_cost',aggfunc='sum').T.\
plot.bar(width = 0.85,stacked=False,legend=True,edgecolor='black',cmap='tab20',ax=ax1)
plt.legend(frameon=False, loc='upper left', ncol=2)
plt.title('Разбивка по регионам')
plt.xlabel('Регион')
plt.ylabel('Количество')
add_value_labels(ax1,'bar',3,"{:.0f}")
plt.xticks(rotation=0)

# построим график Разбивка по регионам стоимость привлечения
ax2 = plt.subplot(2, 2, 4)
profiles.pivot_table(index='region',columns='channel',values='acquisition_cost',aggfunc='mean').T.\
plot.bar(width = 0.85,stacked=False,legend=True,edgecolor='black',cmap='tab20',ax=ax2)
plt.legend(frameon=False, loc='upper left', ncol=2)
plt.title('Разбивка по регионам')
plt.ylabel('Средняя стоимость')
plt.xlabel('Регион')
add_value_labels(ax2,'bar',3,"{:.2f}")
plt.xticks(rotation=0)

# построим график стоимость привлечения по каналам
ax3 = plt.subplot(2, 2, 2)
profiles.pivot_table(columns='channel',values='acquisition_cost',aggfunc='mean').T.\
plot.bar(cmap='magma',edgecolor='black',width = 3,ax=ax3,figsize=(20,9))
plt.title('Средняя стоимость пользователя по каналам привлечения')
plt.xticks([])
plt.yticks(rotation=0)
add_value_labels(ax3,'bar',3,"{:.2f}")
plt.ylabel('Средняя стоимость')
plt.xlabel('Каналы привлечения')
plt.fontsize = 1

# построим график количество новых пользователей по каналам
ax4 = plt.subplot(2, 2, 1)
profiles.pivot_table(columns='channel',values='acquisition_cost',aggfunc='count').T.\
plot.bar(cmap='magma',edgecolor='black',width = 3,ax=ax4,figsize=(20,9))
plt.title('Распределение количества новых пользователей по каналам')
plt.xticks([])
plt.yticks(rotation=0)
plt.ylabel('Количество')
plt.xlabel('Каналы привлечения')
add_value_labels(ax4,'bar',3,"{:.0f}")
plt.fontsize = 1
plt.show()
```



- Основная часть пользователей пришла естественным путем, причем по всем регионам. Далее можно заметить что в разных регионах разные каналы привлечения, так во всех странах Европы это одни тогда как в Штатах другие. Лидерами по привлечению в Европе является WahooNetBanner, OppleCreativeMedia и LeapBob, а в США - FaceBoom и TipTop. Самых дорогих пользователей в США мы получаем у TipTop, самых дешевых у MediaTornado причем разница в стоимости привлечения порядка 10 раз. В Европе самые дорогие пользователи из AdNonSense, lambdaMediaAds и WahooNetBanner тогда как LeapBob самые дешевые пользователи, разница в цене между крайними показателями не столь существенна как в США. Во всех странах европы мы видим одинаковые каналы привлечения, которые имеют сопоставимую стоимость одного пользователя. Исходя из вышеизложенного можно предположить о наличии проблем в распределении бюджетов рекламных трат по каналам привлечения в США.

## Анализ динамики изменения расходов по каналам привлечения

```
In [37]: # Динамика изменения суммы затрат
plt.figure(figsize = (22,4))
sns.heatmap(profiles.pivot_table(index=profiles.dt.astype('datetime64[W]'),columns=drop(columns='organic')).T,\n            annot=True,fmt = '.0f',cbar_kws={'label': 'Сумма затрат', 'orientation':\n            'vertical', 'labelPosition': 'bottom'},\n            xticklabels=False, yticklabels=True,cmap='magma')
plt.title('Динамика изменения средней суммы затрат')
plt.yticks(fontsize=10)
plt.xticks([])
plt.xlabel('Недели')
plt.ylabel('Канал привлечения')

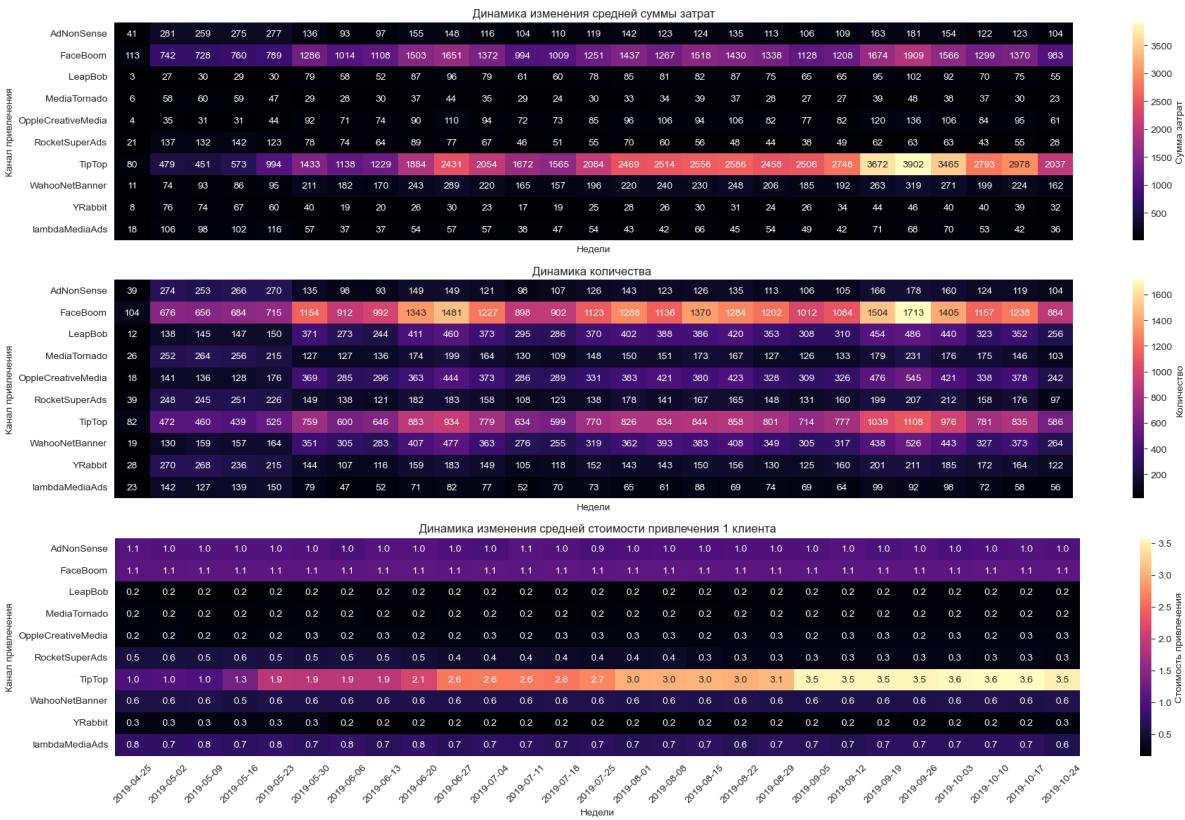
# динамика изменения количества привлеченных пользователей
plt.figure(figsize = (22,4))
sns.heatmap(profiles.pivot_table(index=profiles.dt.astype('datetime64[W]'),columns=drop(columns='organic')).T,\n            annot=True,fmt = '.0f',cbar_kws={'label': 'Количество', 'orientation':\n            'vertical', 'labelPosition': 'bottom'},\n            xticklabels=False, yticklabels=True,cmap='magma')
```

```

xticklabels=False, yticklabels=True, cmap='magma')
plt.title('Динамика количества')
plt.xlabel('Недели')
plt.yticks(fontsize=10)
plt.ylabel('Канал привлечения')

# Динамика изменения средней стоимости привлечения 1 клиента
plt.figure(figsize = (22,4))
sns.heatmap(profiles.pivot_table(index=profiles.dt.astype('datetime64[W]').dt.date,
drop(columns='organic').T,\n    annot=True, fmt = '.1f', cbar_kws={'label': 'Стоимость привлечения', 'orientation': 'vertical'},\n    xticklabels=True, yticklabels=True, cmap='magma')
plt.title('Динамика изменения средней стоимости привлечения 1 клиента')
plt.yticks(fontsize=10)
#plt.xticks([])
plt.tick_params(axis="x", which="both", direction="out", rotation=45)
plt.xlabel('Недели')
plt.ylabel('Канал привлечения')
plt.show()

```



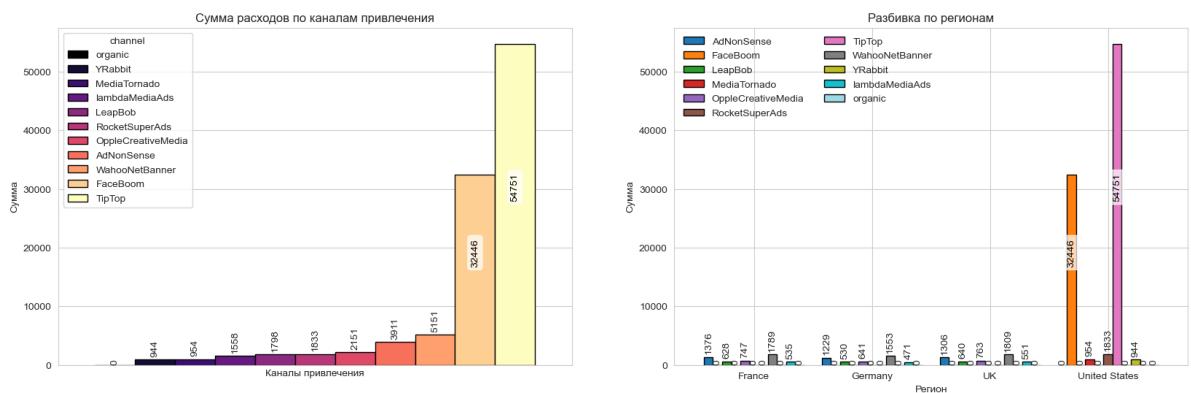
- На графике видно как растет сумма расходов по каналам FaceBoom и TipTop, причем у TipTop рост более выражен и имеет скачок вверх каждый месяц. Однако по графику изменения количества новых пользователей мы видим что в TipTop несмотря на большие расходы рост количества привлеченных пользователей не столь значителен - расходы выросли практически в 10 раз, а количество привлекаемых пользователей всего в около 2 раз. В FaceBoom мы видим пропорциональную динамику по этим показателям - то есть увеличение расходов на рекламу откликается увеличением количества пользователей. Исходя из этого на графике средней стоимости привлечения можно увидеть линейный рост стоимости привлечения по каналу TipTop в 3.5 раза, тогда как стоимость по каналу FaceBoom остается стабильной. Как

**итог можно предположить что потенциал привлечения канала TipTop был исчерпан и увеличение расходов по этому каналу не оправдано и может привести к убыткам.**

## Анализ распределения расходов по каналам

```
In [38]: # построим график суммы расходов по каналам
ax1 = plt.subplot(1, 2, 1)
profiles.pivot_table(columns='channel', values='acquisition_cost', aggfunc='sum').T.\_
plot.bar(cmap='magma', edgecolor='black', width = 3, ax=ax1, figsize=(20,6))
plt.title('Сумма расходов по каналам привлечения')
plt.xticks([])
plt.yticks(rotation=0)
add_value_labels(ax1, 'bar', 3, "{:.0f}")
plt.ylabel('Сумма')
plt.xlabel('Каналы привлечения')
plt.fontsize = 1

# построим график Разбивка по регионам суммы расходов
ax2 = plt.subplot(1, 2, 2)
profiles.pivot_table(index='region', columns='channel', values='acquisition_cost', aggfunc='sum').T.\_
plot.bar(width = 0.85, stacked=False, legend=True, edgecolor='black', cmap='tab20', ax=ax2)
plt.title('Разбивка по регионам')
plt.ylabel('Сумма')
plt.xlabel('Регион')
plt.legend(frameon=False, loc='upper left', ncol=2)
add_value_labels(ax2, 'bar', 3, "{:.0f}")
plt.xticks(rotation=0)
plt.show()
```



- Два самых дорогих канала это Tiptop и FaceBoom расположились в США. Остальные кратно дешевле и относительно равномерно распределяются по регионам

## Вывод по маркетингу

- Явной динамики на увеличение пользователей нет. Количество новых покупателей держится на постоянном, относительно низком уровне, тогда как количество новых пользователей без покупок имеет циклические волны, что может быть реакцией на рекламные компании или сезонность.

**Возможна проблема с качеством привлекаемых клиентов, доля покупателей среди которых не повторяет тенденции на увеличение в паре с клиентами без покупок в циклах роста.**

- Стоимость привлечения одного нового пользователя растет с порядка 0.3 в мае 2019 года до 8.5 в октябре. В совокупности с отсутствием роста новых покупателей, можно предположить о плохой эффективности рекламных компаний. Средняя стоимость привлечения клиента за весь период равна 0.703, общая сумма потраченных на маркетинг средств равна 105497.3. Резкий всплеск на границе первого месяца может быть связан с резким падением количества органических пользователей за которых не надо было платить
- Основная часть пользователей пришла естественным путем, причем по всем регионам. Далее можно заметить что в разных регионах разные каналы привлечения, так во всех странах Европы это одни тогда как в Штатах другие. Лидерами по привлечению в Европе является WahooNetBanner, OppleCreativeMedia и LeapBob, а в США - FaceBoom и TipTop. Самых дорогих пользователей в США мы получаем у TipTop, самых дешевых у MediaTornado причем разница в стоимости привлечения порядка 10 раз. В Европе самые дорогие пользователи из AdNonSense, LambdaMediaAds и WahooNetBanner тогда как LeapBob самые дешевые пользователи, разница в цене между крайними показателями не столь существенна как в США. Во всех странах европы мы видим одинаковые каналы привлечения, которые имеют сопоставимую стоимость одного пользователя. Исходя из вышеизложенного можно предположить о наличии проблем в распределению бюджетов рекламных трат по каналам привлечения в США.
- На графике видно как растет сумма расходов по каналам FaceBoom и TipTop, причем у TipTop рост более выражен. Однако по графику изменения количества новых пользователей мы видим что в TipTop несмотря на большие расходы рост количества привлеченных пользователей не столь значителен - расходы выросли практически в 10 раз, а количество привлекаемых пользователей всего в около 2 раз. В FaceBoom мы видим пропорциональную динамику по этим показателям — то-есть увеличение расходов на рекламу откликается увеличением количества пользователей. Исходя из этого на графике средней стоимости привлечения можно увидеть линейный рост стоимости привлечения по каналу TipTop в 3.5 раза, тогда как стоимость по каналу FaceBoom остается стабильной. Исходя из этого можно предположить что потенциал привлечения канала TipTop был исчерпан и увеличение расходов по этому каналу не оправдано и может привести к убыткам.
- Два самых дорогих канала это Tiptop и FaceBoom расположились в США. Остальные кратно дешевле и относительно равномерно распределяются по регионам

## Когортный анализ

## Общие показатели окупаемости - LTV, ROI, CAC

In [39]:

```
# зададим момент анализа
observation_date = datetime.strptime('2019-11-01', '%Y-%m-%d')
observation_date = observation_date.date()

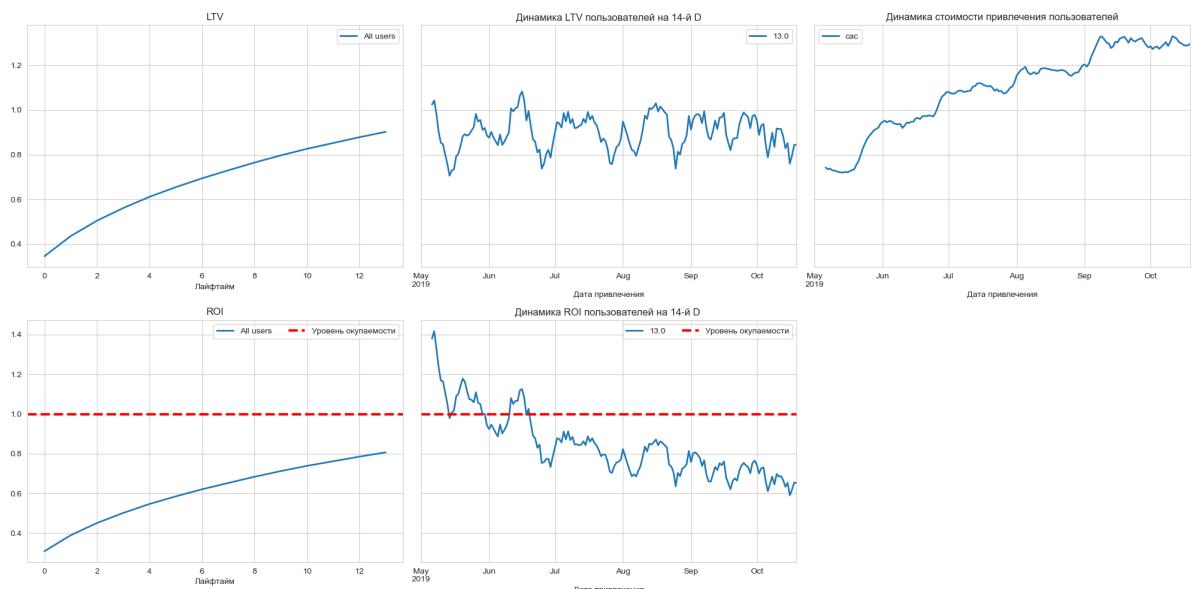
# уберем органиков
profiles = profiles.loc[profiles['channel'] != 'organic']
```

In [40]:

```
# посчитаем бизнес показатели при помощи нашей формулы, горизонт зададим в 3 недели
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv_new(
    profiles,
    orders,
    observation_date,
    horizon_days=14,
    dimensions=[],
    ignore_horizon=True,
    cogort_size='1D')
```

In [41]:

```
# отобразим результат
plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, 14, window=6, cogort_size='1D')
```



- В целом в срок 2 недели реклама не окупается. Прибыль на пользователя в течении горизонта растет, в динамике периода наблюдения у пользователей 14 дня мы видим циклические колебания LTV внутри месяца. Пользователи 14 дня жизни становятся убыточными со второго месяца, возможно из-за постоянного увеличения стоимости их привлечения. Стоимость привлечения растет на всем периоде наблюдения, что может быть причиной проблем окупаемости.**

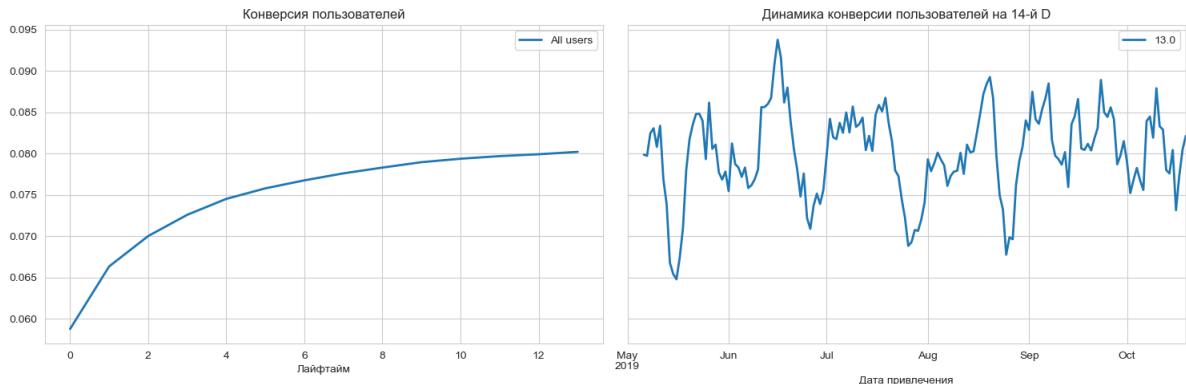
## Общие показатели конверсии

In [42]:

```
# Посчитаем конверсию для всех пользователей
c_result_raw, conversion_grouped, conversion_in_time = get_conversion_new(
    profiles,
    orders,
    observation_date,
```

```
14,
dimensions=[],
ignore_horizon=False,
cogort_size = 'D'
)
```

In [43]: # отобразим результат  
`plot_conversion(conversion_grouped, conversion_in_time, 14, window=6, cogort_size =`

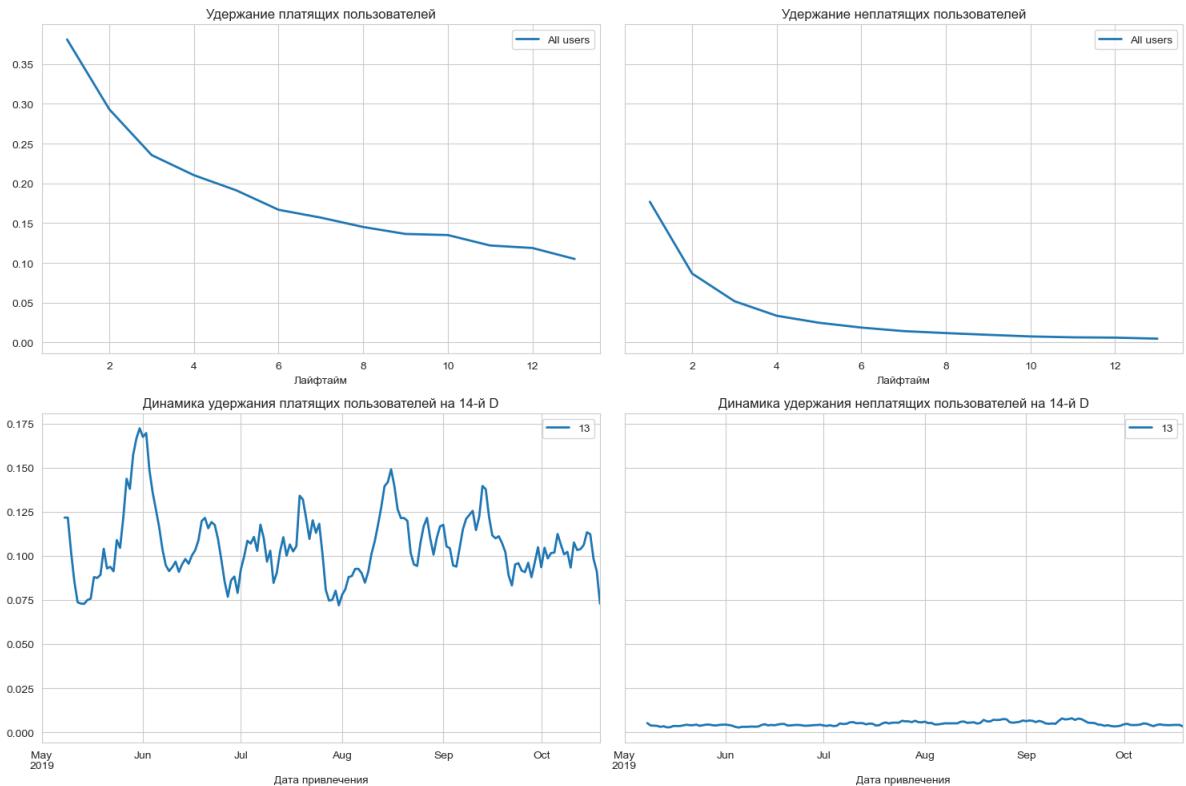


- В целом мы видим неплохую конверсию с высоким ростом в течении первых 2 недель. В динамике видим цикличность внутри месяца, но показатель держится на довольно высоком уровне. Заметно пониженное значение конверсии в первом месяце наблюдения, необходимо выяснить причину.

## Удержание общей выборки

In [44]: # посчитаем удержание общей выборки  
`result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,`  
`observation_date,`  
`14,`  
`dimensions=[],`  
`ignore_horizon=False,`  
`cogort_size = 'D')`

In [45]: #посмотрим на результат  
`plot_retention(result_grouped, result_in_time, 14, window=8, cogort_size = 'D')`



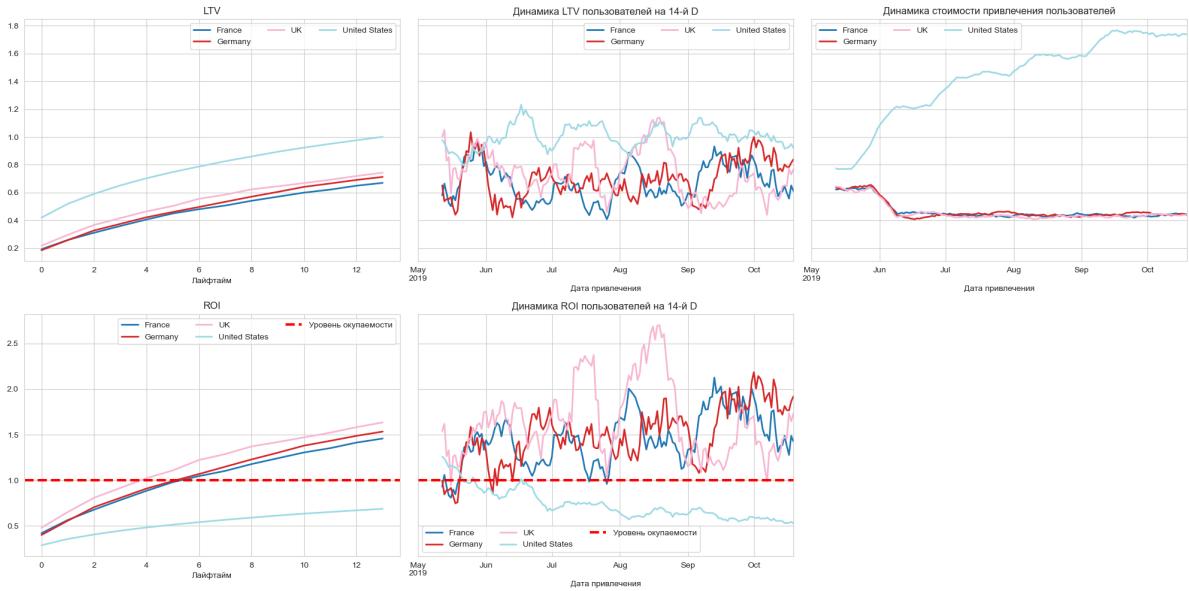
- Аномалий в удержании по общей выборке не заметно, кривые покупателей и не платящих пользователей имеют плавное снижение внутри срока жизни, причем покупатели удерживаются значительно лучше. Динамика удержания пользователей 14 дня жизни держится в пределах 15 процентов с циклическими колебаниями внутри месяца, не платящие пользователи в динамике находятся на около 0 значении.**

## Показатели окупаемости с разбивкой по странам

```
In [46]: # Получим показатели окупаемости с разбивкой по странам
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv_new(
    profiles,
    orders,
    observation_d
    horizon_days=14,
    dimensions=['country'],
    ignore_horizon=True,
    cogort_size='1d')
```

```
In [47]: # отобразим результат
plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, 14, window=12, cogort_size='1d')
```

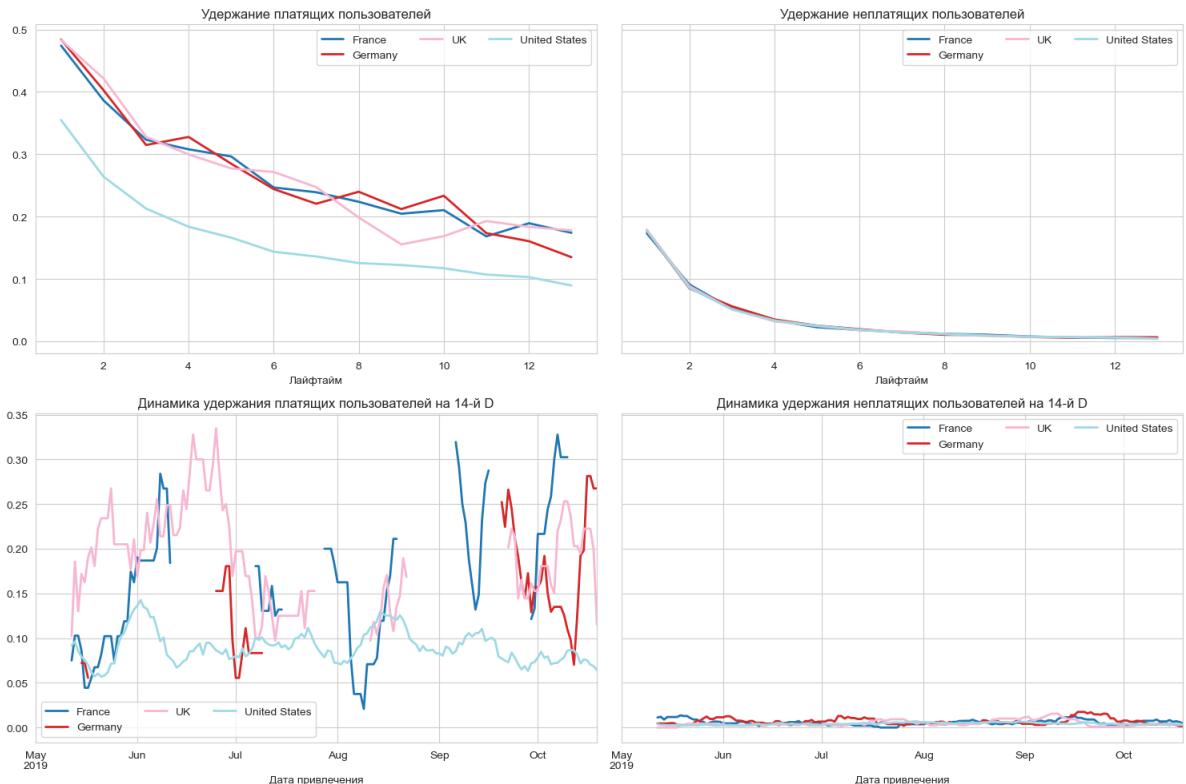


- На графике видно, что несмотря на самый высокий уровень LTV пользователи из США имеют самую плохую окупаемость, тогда как Европа окупается около 4 дней штаты остаются в убытке весь горизонт. По динамике LTV видно что пользователи из США стабильно больше приносят денег, несмотря на это динамика ROI показывает что они уходят в убыток со 2 месяца наблюдения. На графике динамики САС видна возможная причина проблем в значительном превышении стоимости клиентов из США над Европой, которая в добавок имеет тенденцию на увеличение. Европейские страны не имеют существенных негативных моментов и окупятся в пределах установленного срока, лучше всех себя проявляют пользователи из Англии.

## Удержание на уровне регионов

```
In [48]: # посчитаем удержание на уровне регионов
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               14,
                                                               dimensions=['region'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'D')
```

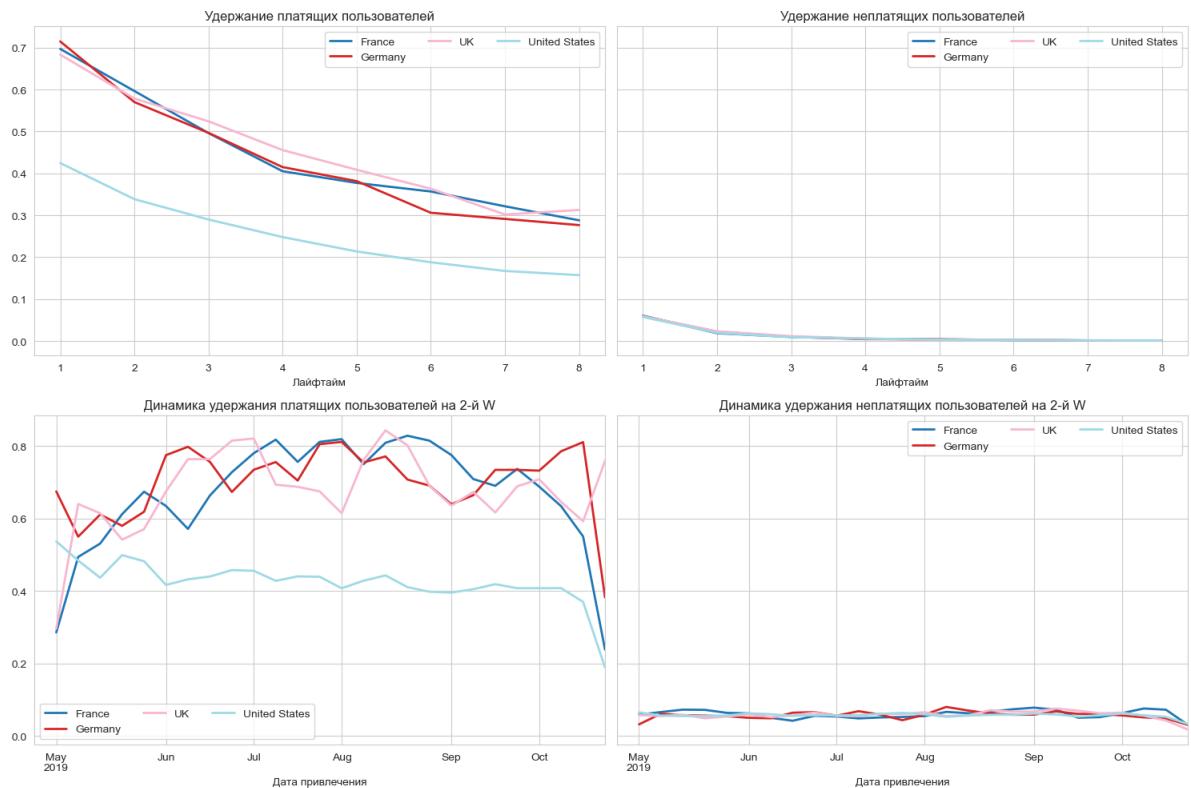
```
In [49]: #посмотрим на результат
plot_retention(result_grouped, result_in_time, 14, window=12, cogort_size = 'D')
```



- На графике можно заметить что покупатели из США заметно хуже удерживаются чем Европа, хотя не платящие пользователи такой проблемы не имеют. График динамики удержания уже трудно читать из-за маленькой выборки, можно предположить о ее однородности и сформировать когорты по неделям

```
In [50]: # посчитаем удержание на уровне регионов для когорт сформированных по неделям
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               9,
                                                               dimensions=['region'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'W')
```

```
In [51]: #посмотрим на результат
plot_retention(result_grouped, result_in_time, 2, window=2, cogort_size = 'W')
```

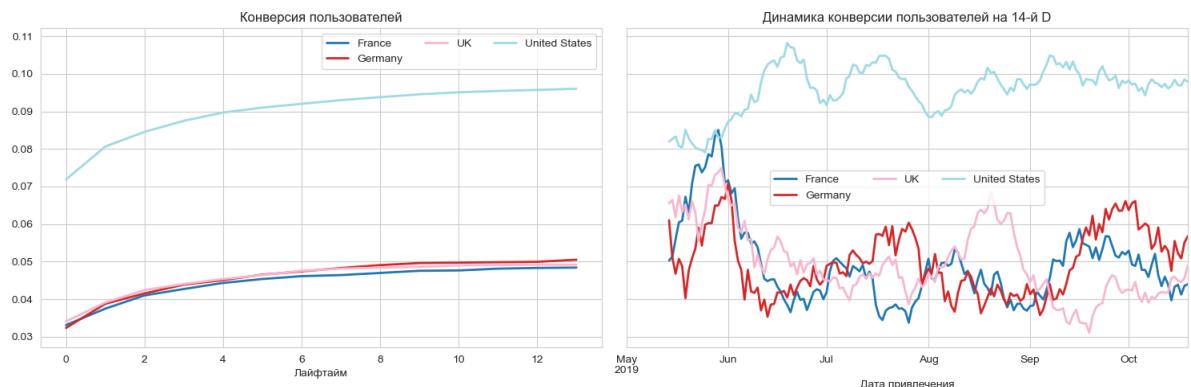


- Тенденции сохранились, поэтому можно доверять результату. График динамики хорошо читается и показывает что пользователи США стабильно хужедерживаются.

## Конверсия на уровне регионов

```
In [52]: # Посчитаем конверсию для на уровне регионов
c_result_raw, conversion_grouped, conversion_in_time = get_conversion_new(
    profiles,
    orders,
    observation_date,
    14,
    dimensions=['region'],
    ignore_horizon=False,
    cogort_size = 'D'
)
```

```
In [53]: # отобразим результат
plot_conversion(conversion_grouped, conversion_in_time, 14, window=14,cogort_size :
```



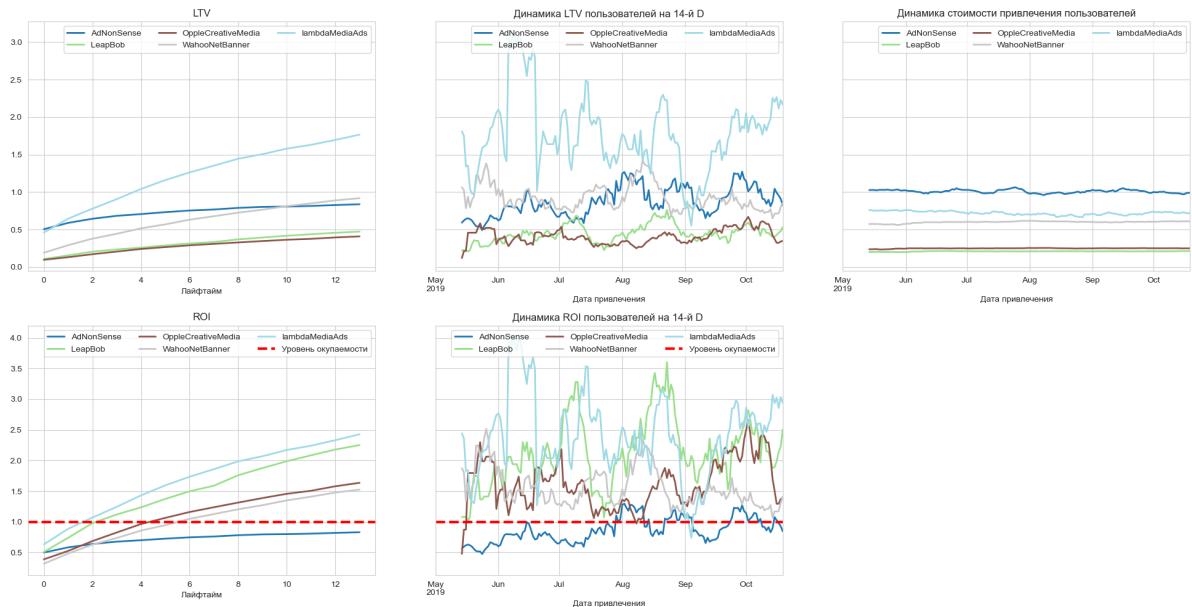
- Конверсия пользователей из США растет в пределах их жизни и стабильно выше чем у Европы, по динамике также видно, что конверсия США стабильно выше на всем периоде наблюдения. Так же можно отметить резкий рост конверсии в США в первый месяц наблюдения

## Окупаемость на уровне каналов привлечения - Европа

```
In [54]: # Добавим в профили новый маркер по признаку continent для работы с фильтром нашей
profiles['continent'] = np.where(profiles['region']=='United States', 'USA', 'euro
```

```
In [55]: # Получим показатели окупаемости с разбивкой по каналам Европы
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv_new(
    profiles,
    orders,
    observation_d
    horizon_days=14,
    dimensions=['c
    ignore_horizon
    cogort_size=1
```

```
In [56]: # отобразим результат
plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, 14, window=14, cogor
```

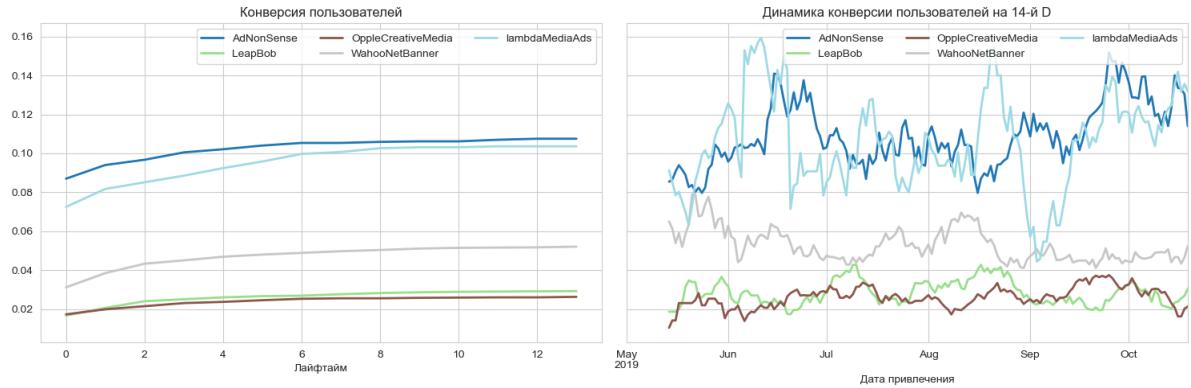


- По Европе мы видим 5 каналов привлечения, LambdaMediaAds имеет отличные показатели окупаемости: самый высокий показатель LTV который круто растет в период жизни и имеет циклически высокий уровень в период наблюдения, стоимость привлечения остается на стабильно низком уровне, на основании этого он может быть хорошей точкой роста. Напротив AdNonSense имеет худшие показатели окупаемости: даже после 3 недель жизни пользователи тут не окупаются, LTV не самый низкий, но практически не меняется за период жизни, стоимость привлечения стабильно на высоком уровне, по динамике ROI пользователей 14 дня мы видим весь период наблюдения обточные значения

## Конверсия на уровне каналов Европы

```
In [57]: # Посчитаем конверсию для на уровне каналов Европы
c_result_raw, conversion_grouped, conversion_in_time = get_conversion_new(
    profiles,
    orders,
    observation_date,
    14,
    dimensions=['channel'],
    ignore_horizon=False,
    cogort_size = 'D', locke
)
```

```
In [58]: # отобразим результат
plot_conversion(conversion_grouped, conversion_in_time, 14, window=14,cogort_size :
```

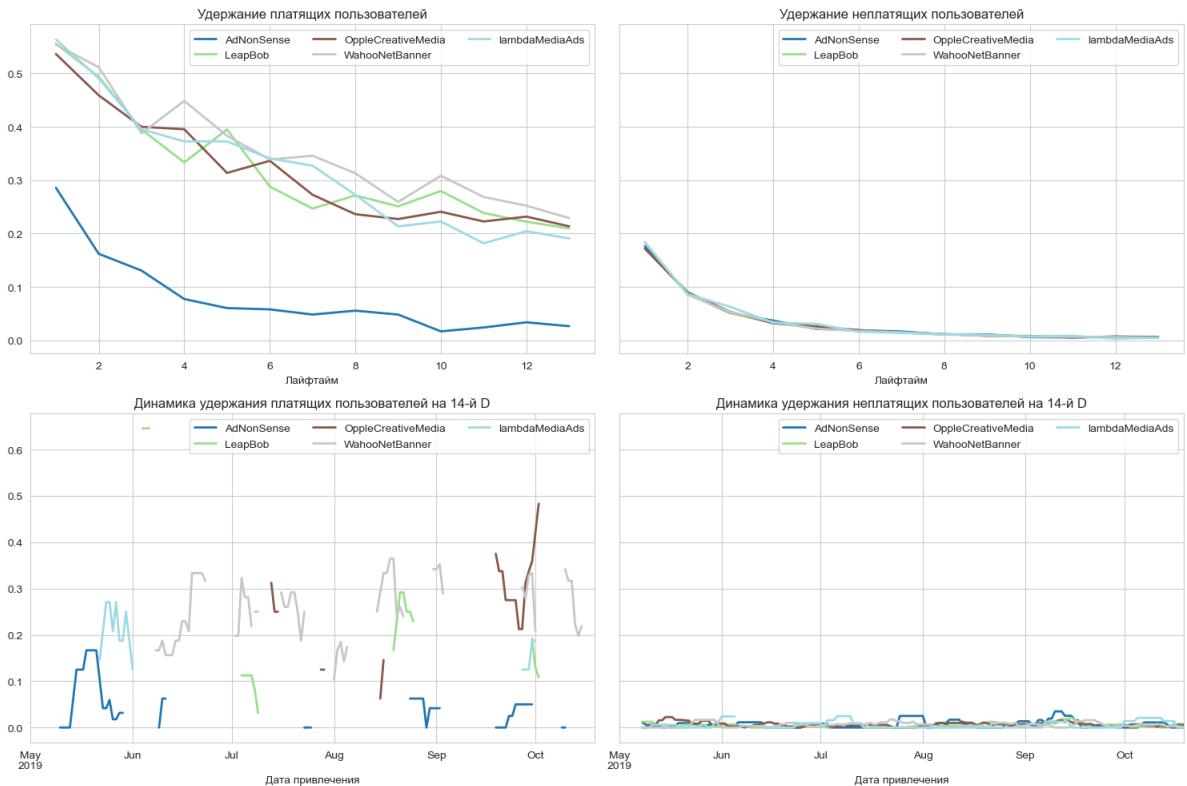


- Конверсия пользователей LambdaMediaAds имеет рост в первую неделю жизни, далее стабилизируется на высоком уровне вместе с AdNonSense, динамика конверсии пользователей 14 дня также показывает высокий уровень на протяжении всего периода наблюдения. Канал LambdaMediaAds предпочтителен как точка роста.

## Удержание на уровне каналов Европы

```
In [59]: # посчитаем удержание на уровне регионов для когорт сформированных по неделям
result_raw, result_grouped,result_in_time = get_retention_new(profiles,visits,
    observation_date,
    14,
    dimensions=['channel'],
    ignore_horizon=False,
    cogort_size = 'D', locke
```

```
In [60]: plot_retention(result_grouped, result_in_time, 14,window=8, cogort_size = 'D')
```



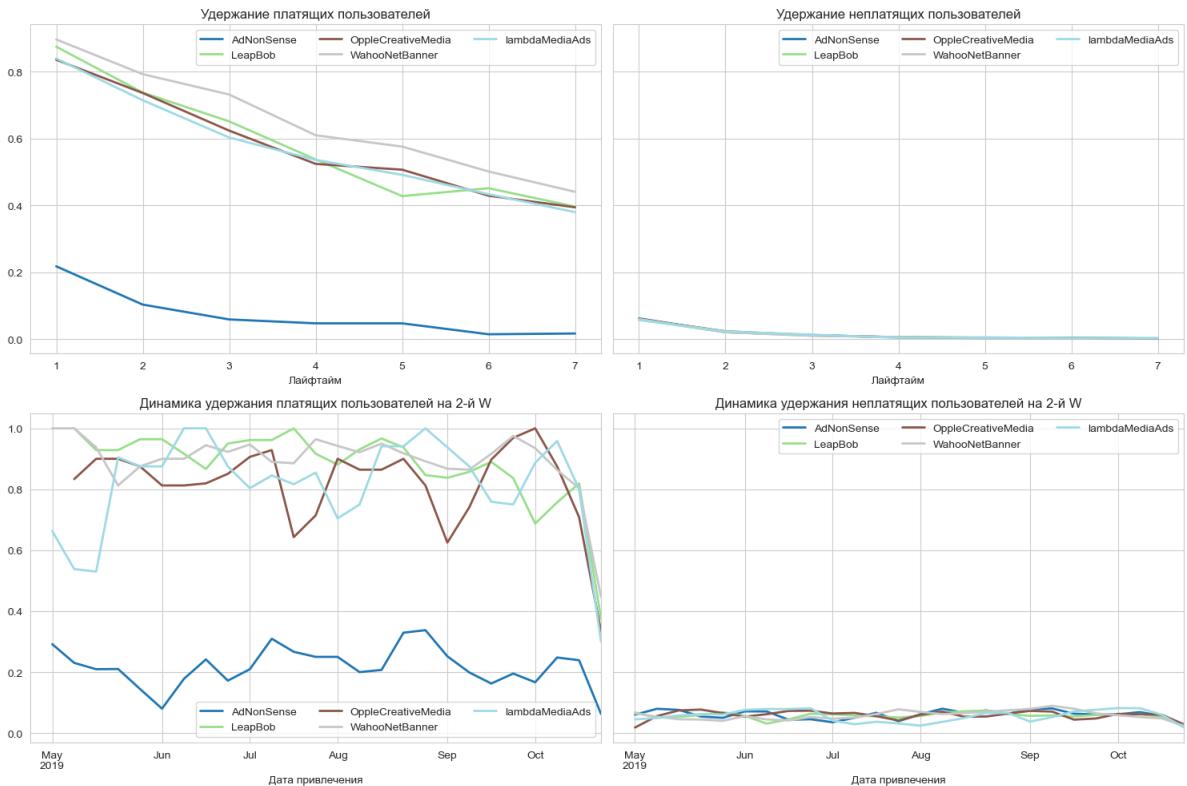
- На графике мы видим аномально низкое удержание платящих пользователей из канала AdNonSense что может быть причиной плохих показателей окупаемости этого канала, необходимо выяснить причины данной проблемы. Показатели остальных каналов в норме. График динамики удержания сломался из-за маленькой выборки, повторим наш опыт с укрупнением когорт**

In [61]:

```
# посчитаем удержание на уровне регионов для когорт сформированных по неделям
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               8,
                                                               dimensions=['channel'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'W', locke
```

In [62]:

```
plot_retention(result_grouped, result_in_time, 2, window=2, cogort_size = 'W')
```

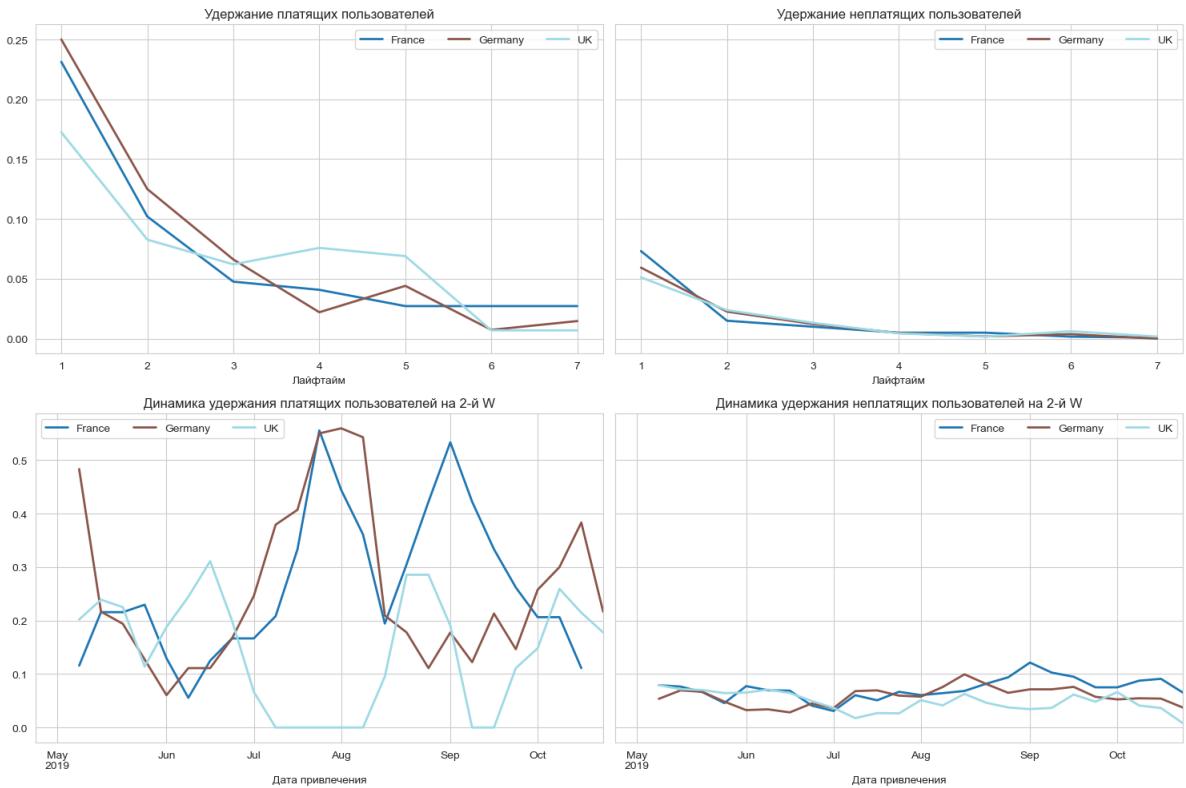


- Тенденции сохранились, график динамики читается, на нем видно стабильно низкое удержание на всем периоде наблюдения у AdNonSense

## Посчитаем удержание на уровне регионов и устройств для AdNonSense

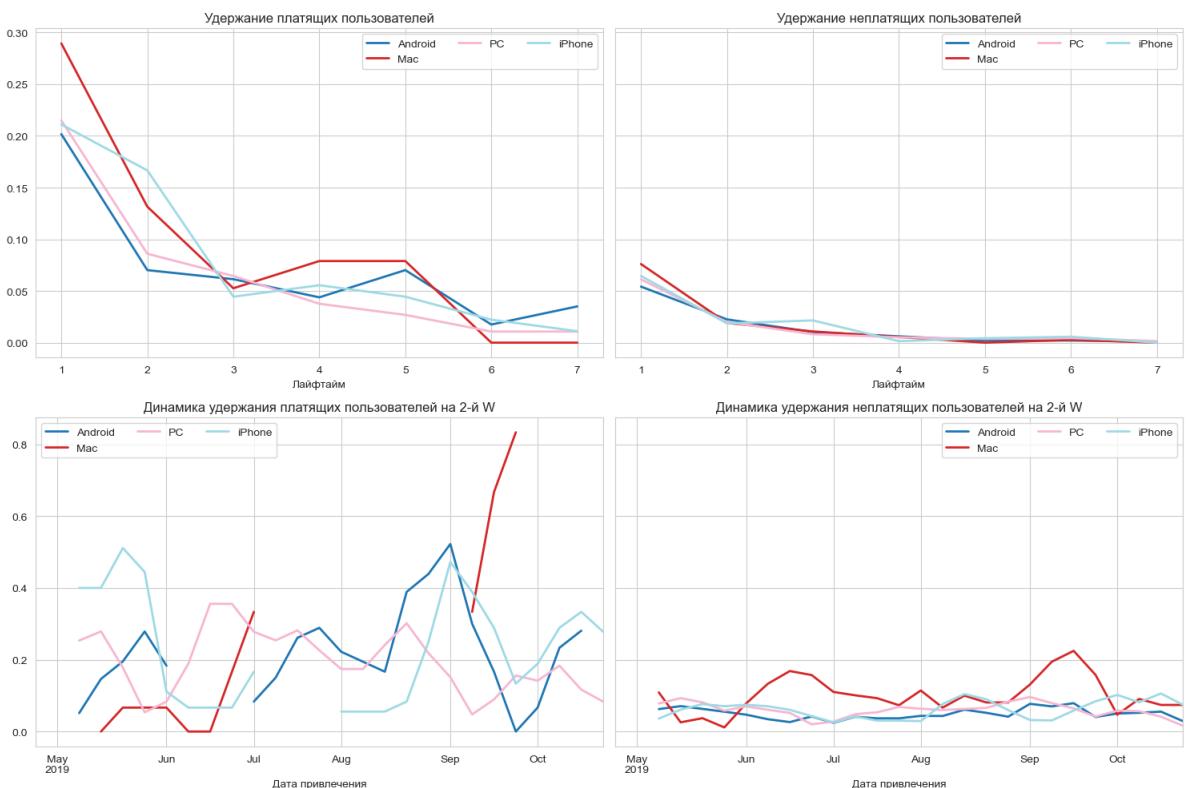
```
In [63]: # посчитаем удержание на уровне регионов для AdNonSense
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               8,
                                                               dimensions=['region'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'W', locke
```

```
In [64]: # посмотрим на результат
plot_retention(result_grouped, result_in_time, 2, window=3, cogort_size = 'W')
```



```
In [65]: # посчитаем удержание на уровне регионов для AdNonSense
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               8,
                                                               dimensions=['device'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'W', locked=True)
```

```
In [66]: # посмотрим на результаты
plot_retention(result_grouped, result_in_time, 2, window=3, cogort_size = 'W')
```



- Внутри канала на уровне регионов и устройств одинаковый уровень удержания, значит проблема именно в канале AdNonSense

## Окупаемость на уровне каналов привлечения - США

```
In [67]: # Получим показатели окупаемости с разбивкой по каналам США
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv_new(
    profiles,
    orders,
    observation_d
    horizon_days=14,
    dimensions=['channel'],
    ignore_horizon=True,
    cogort_size=14)
```

```
In [68]: # отобразим результат
plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, 14, window=14, cogort_size=14)
```



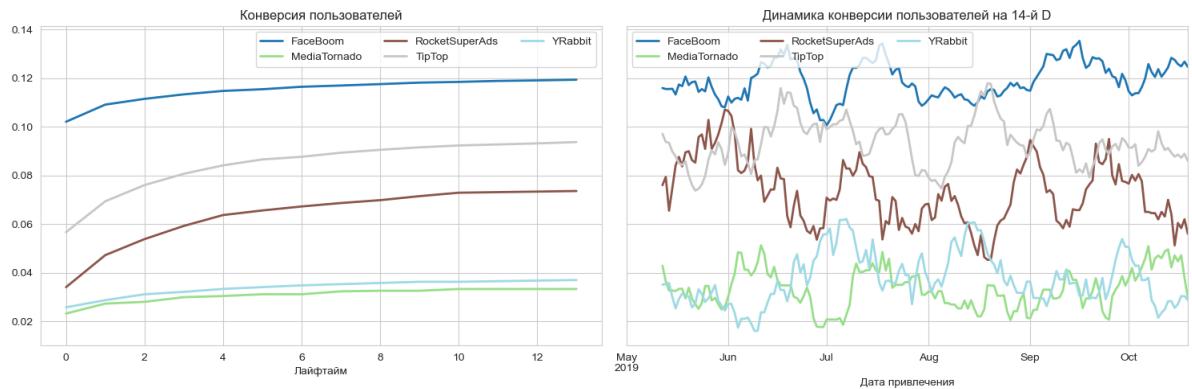
- На графике видно, что пользователи из 2 каналов не окупаются в течение 2 недель жизни, это TipTop и FaceBoom, несмотря на хорошую положительную динамику LTV в период жизни и высокое его общее значение у пользователей 14 дня в динамике периода наблюдения у первого. На графике динамики стоимости привлечения четко виден крутой рост по каналу TipTop и стабильно высокий уровень у FaceBoom что может быть причиной плохой окупаемости. Динамика ROI пользователей 14 дня жизни показывает стабильно низкие значения весь период наблюдения, которые расположены за зоной окупаемости. По остальным каналам существенных аномалий не видно

## Конверсия на уровне каналов США

```
In [69]: # Посчитаем конверсию для на уровне каналов США
c_result_raw, conversion_grouped, conversion_in_time = get_conversion_new(
```

```
profiles,
orders,
observation_date,
14,
dimensions=['channel'],
ignore_horizon=False,
cogort_size = 'D',locked
)
```

In [70]: `plot_conversion(conversion_grouped, conversion_in_time, 14, window=12, cogort_size = 'D')`



- Мы видим довольно большой разброс по конверсии между каналами привлечения США, который сохраняется и в динамике пользователей 14 дня, необходимо выяснить причину. FaceBoom имеют самый высокий уровень конверсии.**

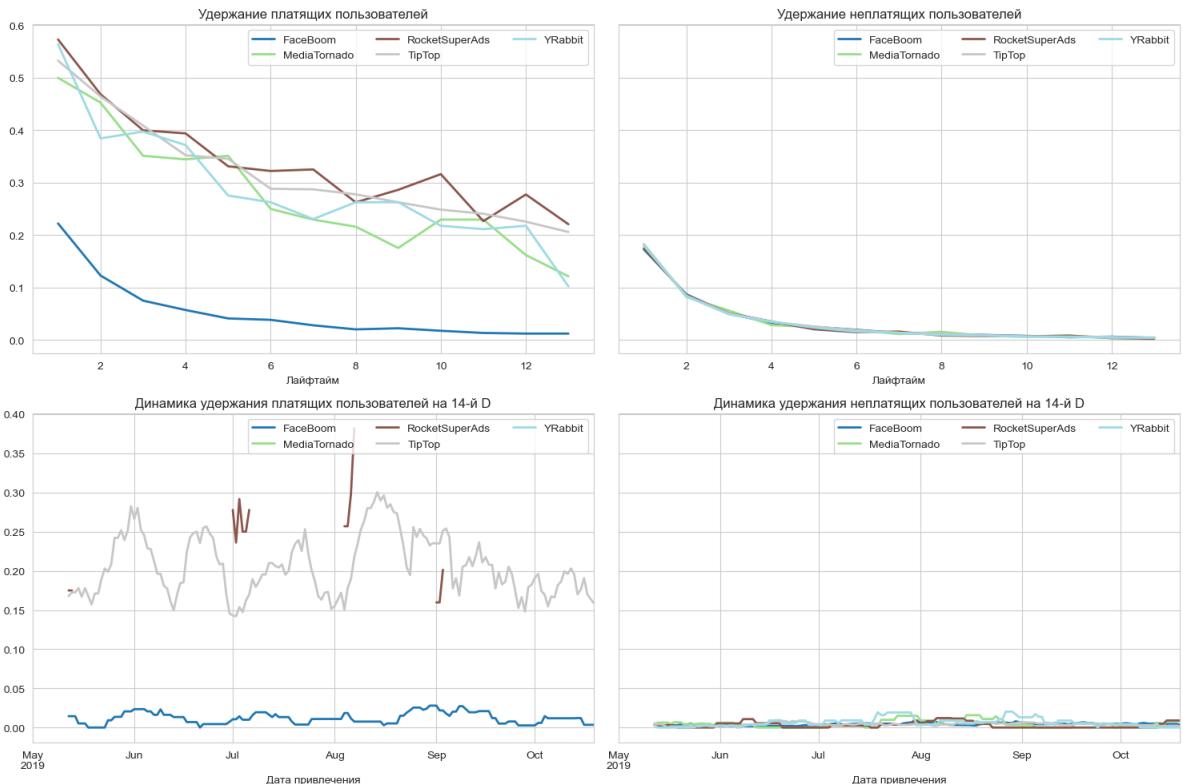
## Удержание на уровне каналов привлечения - США

In [71]: `# посчитаем удержание на уровне каналов в США`

```
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               14,
                                                               dimensions=['channel'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'D',locked
)
```

In [72]: `# посмотрим на результат`

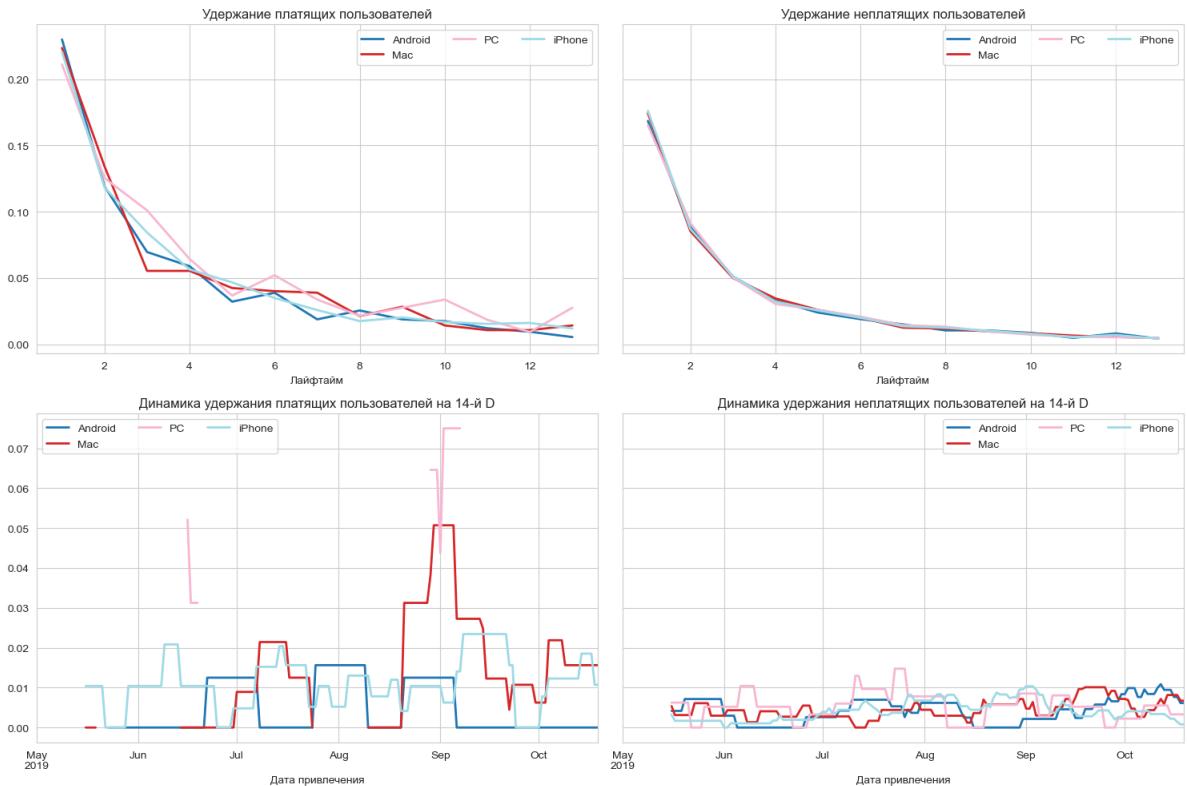
```
plot_retention(result_grouped, result_in_time, 14, window=12, cogort_size = 'D')
```



- Видим стабильно низкий уровень удержания в канале FaceBoom у покупателей, как внутри когорты так и в динамике у покупателей 14 дней, что может быть причиной убыточности данного канала.

```
In [73]: # посчитаем удержание на уровне устройств по каналу FaceBoom
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               14,
                                                               dimensions=['device'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'D', locke
```

```
In [74]: # посмотрим на результат
plot_retention(result_grouped, result_in_time, 14, window=16, cogort_size = 'D')
```

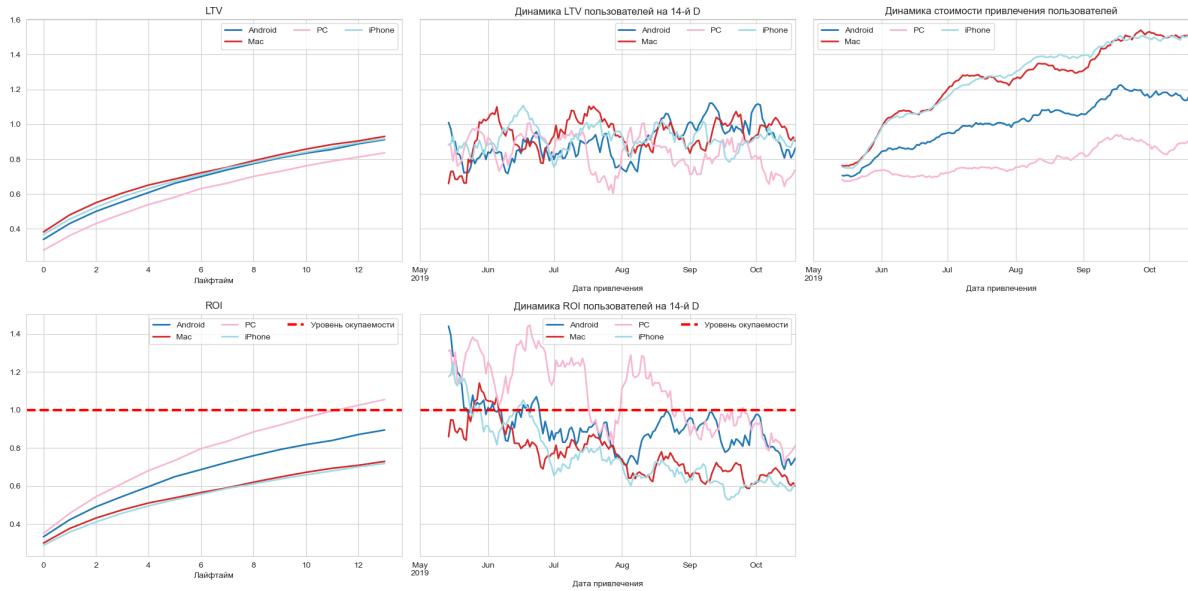


- Динамика по устройствам схожа, значит проблема именно в удержание канала FaceBoom

## Окупаемость с разбивкой по устройствам

```
In [75]: # Получим показатели окупаемости с разбивкой по каналам США
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv_new(
    profiles,
    orders,
    observation_da
horizon_days=14,
dimensions=['
ignore_horizon
cogort_size='1
)
```

```
In [76]: # отобразим результат
plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, 14, window=14, cogor
```

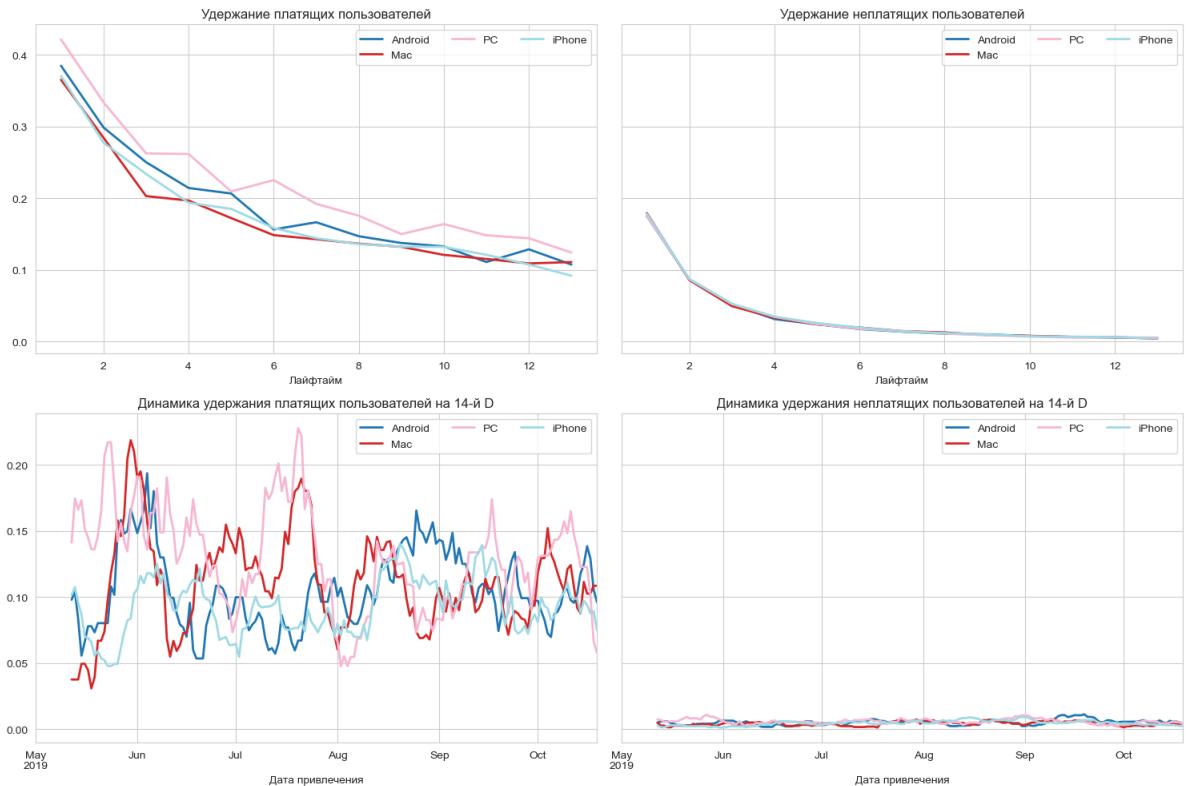


- Пользователи РС окупаются в 2 недели жизни, тогда как Android, Iphone и MAC нет. Это может быть связано с самой высокой стоимостью привлечения, которая имеет динамику увеличения на всем периоде наблюдения. LTV стабильно растет на всем горизонте, причем значение РС немного ниже остальных. Тенденция LTV пользователей 14 дня жизни на довольно высоком уровне и циклична внутри месяца. САС во всех группах растет на протяжении всего периода наблюдения, пользователи РС имеют самый низкий ее уровень. Пользователи MAC, Iphone и Android 14 дня жизни уже со 2 месяца уходят в убыток. Возможной причиной является неверное бюджетирование каналов, которое мы определили выше.

## Удержание на уровне устройств

```
In [77]: # посчитаем удержание на уровне каналов в США
result_raw, result_grouped, result_in_time = get_retention_new(profiles, visits,
                                                               observation_date,
                                                               14,
                                                               dimensions=['device'],
                                                               ignore_horizon=False,
                                                               cogort_size = 'D')

In [78]: # посмотрим на результат
plot_retention(result_grouped, result_in_time, 14, window=12, cogort_size = 'D')
```

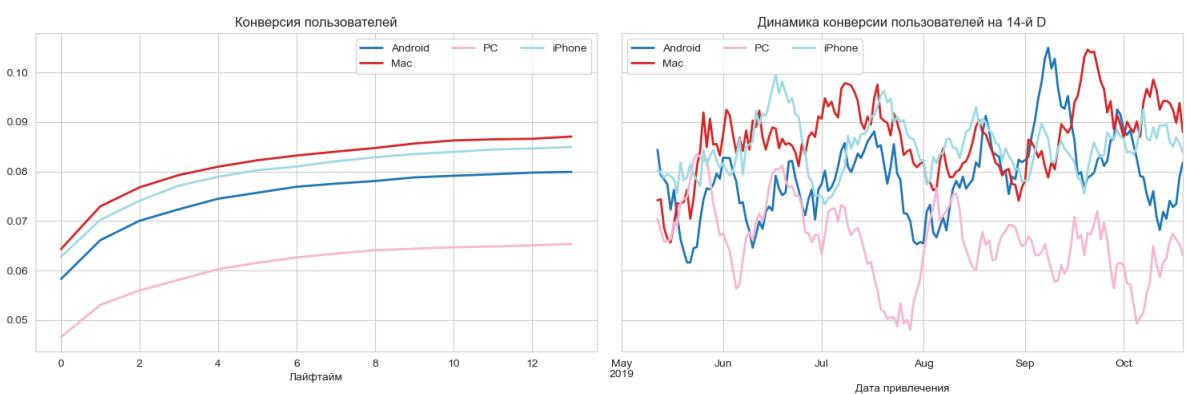


- Явных проблем с удержанием нет, покупатели РС имеют незначительно выше показатели

## Конверсия на уровне устройств

```
In [79]: # Посчитаем конверсию для на уровне каналов США
c_result_raw, conversion_grouped, conversion_in_time = get_conversion_new(
    profiles,
    orders,
    observation_date,
    14,
    dimensions=['device'],
    ignore_horizon=False,
    cogort_size = 'D'
)
```

```
In [80]: plot_conversion(conversion_grouped, conversion_in_time, 14, window=12,cogort_size :
```



- Пользователи РС стабильно хуже конвертируются как в период горизонта внутри когорт так и в динамике пользователей 14 дней, в свете хорошей

**окупаемости и высокого удержания, выяснив причину плохой конверсии можно еще увеличить прибыльность этих пользователей.**

## Вывод по когортному анализу

- В целом в срок 2 недели реклама не окупается. Прибыль на пользователя в течении горизонта растет, в динамике периода наблюдения у пользователей 14 дня мы видим циклические колебания LTV внутри месяца. Пользователи 14 дня жизни становятся убыточными со второго месяца, возможно из-за постоянного увеличения стоимости их привлечения. Стоимость привлечения растет на всем периоде наблюдения, что может быть причиной проблем окупаемости.
- В целом мы видим неплохую конверсию с высоким ростом в течении первых 2 недель. В динамике видим цикличность внутри месяца, но показатель держится на довольно высоком уровне. Заметно пониженное значение конверсии в первом месяце наблюдения, необходимо выяснить причину.
- Аномалий в удержании по общей выборке не заметно, кривые покупателей и не платящих пользователей имеют плавное снижение внутри срока жизни, причем покупатели удерживаются значительно лучше. Динамика удержания пользователей 14 дня жизни держится в пределах 15 процентов с циклическими колебаниями внутри месяца, не платящие пользователи в динамике находятся на около 0 значении.
- Несмотря на самый высокий уровень LTV пользователи из США имеют самую плохую окупаемость, тогда как Европа окупается около 4 дней штаты остаются в убытке весь горизонт. По динамике LTV видно что пользователи из США стабильно больше приносят денег, несмотря на это динамика ROI показывает что они уходят в убыток со 2 месяца наблюдения. На графике динамики SAC видна возможная причина проблем в значительном превышении стоимости клиентов из США над Европой, которая в добавок имеет тенденцию на увеличение. Европейские страны не имеют существенных негативных моментов и окупаются в пределах установленного срока, лучше всех себя проявляют пользователи из Англии.
- Покупатели из США заметно хуже удерживаются чем из Европы, хотя не платящие пользователи такой проблемы не имеют.
- Конверсия пользователей из США растет в пределах их жизни и стабильно выше чем у Европы, по динамике также видно, что конверсия США стабильно выше на всем периоде наблюдения. Так же можно отметить резкий рост конверсии в США в первый месяц наблюдения
- По Европе мы видим 5 каналов привлечения, LambdameadiaAds имеет отличные показатели окупаемости: самый высокий показатель LTV который круто растет в период жизни и имеет циклически высокий уровень в период наблюдения, стоимость привлечения остается на стабильно низком уровне, на основании этого он может быть хорошей точкой роста. Напротив AdNonSense имеет худшие показатели окупаемости: даже после 3 недель

**жизни пользователи тут не окупаются, LTV не самый низкий, но практически не меняется за период жизни, стоимость привлечения стабильно на высоком уровне, по динамике ROI пользователей 14 дня мы видим весь период наблюдения убыточные значения**

- Конверсия пользователей LambdameadiaAds имеет рост в первую неделю жизни, далее стабилизируется на высоком уровне вместе с AdNonSense, динамика конверсии пользователей 14 дня также показывает высокий уровень на протяжении всего периода наблюдения. Канал LambdameadiaAds предпочтителен как точка роста.
- Аномально низкое удержание платящих пользователей из канала AdNonSense что может быть причиной плохих показателей окупаемости этого канала, необходимо выяснить причины данной проблемы. Показатели остальных каналов в норме. Стабильно низкое удержание на всем периоде наблюдения у пользователей AdNonSense
- Пользователи из 2 каналов США не окупаются в течение 2 недель жизни, это TipTop и FaceBoom, несмотря на хорошую положительную динамику LTV в период жизни и высокое его общее значение у пользователей 14 дня в динамике периода наблюдения у первого. На графике динамики стоимости привлечения четко виден крутой рост по каналу TipTop и стабильно высокий уровень у FaceBoom что может быть причиной плохой окупаемости. Динамика ROI пользователей 14 дня жизни показывает стабильно низкие значения весь период наблюдения, которые расположены за зоной окупаемости. По остальным каналам существенных аномалий не видно
- Большой разброс по конверсии между каналами привлечения США, который сохраняется и в динамике пользователей 14 дня, необходимо выяснить причину. FaceBoom имеют самый высокий уровень конверсии.
- Видим стабильно низкий уровень удержания в канале FaceBoom, как внутри когорты так и в динамике у пользователей 14 дня, что может быть причиной убыточности данного канала.
- Пользователи РС окупаются в 2 недели жизни, тогда как Android, Iphone и MAC нет. Это может быть связано с самой высокой стоимостью привлечения, которая имеет динамику увеличения на всем периоде наблюдения. LTV стабильно растет на всем горизонте, причем значение РС немного ниже остальных. Тенденция LTV пользователей 14 дня жизни на довольно высоком уровне и циклична внутри месяца. САС во всех группах растет на протяжении всего периода наблюдения, пользователи РС имеют самый низкий ее уровень. Пользователи MAC, Iphone и Android 14 дня жизни уже со 2 месяца уходят в убыток. Возможной причиной является неверное бюджетирование каналов, которое мы определили выше.
- Явных проблем с удержанием нет, пользователи РС имеют незначительно выше показатели
- Пользователи РС стабильно хуже конвертируются как в период горизонта внутри когорты так и в динамике пользователей 14 дня, в свете хорошей окупаемости и высокого удержания, выяснив причину плохой конверсии можно еще увеличить прибыльность этих пользователей.

# Основные причины неэффективности привлечения пользователей

- Неверное распределение маркетинговых усилий по каналам привлечения:  
избыток средств направлен в канал TipTop - США и AdNonSense - Европа
- Аномально низкое удержание покупателей канала AdNonSense для европы и FaceBoom для США
- Низкая конвертация пользователей ПК

## Рекомендации

- Снизить финансирование канала TipTop - США
- Выяснить причину резкого снижения количества органических пользователей после первого месяца наблюдения
- Выяснить и устранить причины низкого удержания каналов покупателей AdNonSense для европы и FaceBoom для США
- Выяснить причину низкой конвертации пользователей ПК
- Высвободившиеся средства перенаправить в более перспективные каналы роста, такие как LambdameadiaAds для Европы и RoketSuperAds для США