Оглавление

ЦЕЛЬ ИССЛЕДОВАНИЯ: Определить возможные причины снижения покупательской активности, несмотря на активную рекламную деятельность. Определить с каких устройств чаще приходят пользователи, с какого региона.Сколько платит каждый пользователь и скольско тратится на его привлечение. Дать рекомнендации по оптимизации расходов на рекламу

Откроем данные , изучим информацию

In [1]:
```
# импортируем библиотеки
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
from datetime import datetime, timedelta
import warnings
warnings.filterwarnings(action = 'ignore')
pd.options.display.max_columns = 40
```

In [2]:
```
# загружаем данные
session = pd.read_csv('C:/Users/User/Downloads/visits_info_short.csv')
orders = pd.read_csv('C:/Users/User/Downloads/orders_info_short.csv')
costs = pd.read_csv('C:/Users/User/Downloads/costs_info_short.csv')
```

In [3]:
```
# датасет содержит информацию о пользовательских сессиях
session.head(5)
```

Out[3]:

	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

In [4]:
```
# датасет содержит данные о тратах пользователей
orders.head(5)
```

Out[4]:

	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

In [5]:
```
# датасет содержит данные о тратах на рекламу
costs.head(5)
```

Out[5]:

	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

In [6]:
```
# откроем информацию о датасетах
session.info()

orders.info()

costs.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  object
 5   Session End    309901 non-null  object
dtypes: int64(1), object(5)
memory usage: 14.2+ MB
<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
<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
```

Данные не содержат пропусков, стоит обратить внимание на тип колонок содержащих дату.Для удобства произведем переименование столбцов

In [7]:
```
# переименуем столбцы датасетов
session.columns = ('user_id', 'region','device', 'channel', 'session_start','se
```

```
orders.columns = ('user_id', 'event_dt', 'revenue')
costs.columns = ('dt', 'channel', 'costs')
```

Вывод:

Получены три датасета, содержащие данные о пользовательских сессиях, покупках и расходах на рекламу. Данные не содержат пропусков, произведена замена наименований столбцов

Предобработка данных

In [8]:
```
# приведем некоторые столбцы к нужному формату
session['session_start'] = pd.to_datetime(session['session_start'])
session['session_end'] = pd.to_datetime(session['session_end'])
orders['event_dt'] = pd.to_datetime(orders['event_dt'])
costs['dt'] = pd.to_datetime(costs['dt']).dt.date
```

In [9]:
```
#посчитаем дубликаты
print('Количество явных дубликатов в пользовательских сессиях: ',session.duplic
print('Количество явных дубликатов в данных о покупках: ',orders.duplicated().s
print('Количество явных дубликатов в данных о рекламных затратах: ',costs.dupli
```

```
Количество явных дубликатов в пользовательских сессиях:  0
Количество явных дубликатов в данных о покупках:  0
Количество явных дубликатов в данных о рекламных затратах:  0
```

In [10]:
```
# проверим уникальные значения столбцов region, device,channel для возможного в
print('Уникальные значения местоположения пользователей :')
[print(i) for i in session['region'].unique()]
print()
print('Уникальные значения устройств пользователей :')
[print(i) for i in session['device'].unique()]
print()
print('Уникальные значения  каналов привлечения пользователей :'),
[print(i) for i in sorted(session['channel'].unique())]
print()
print('Уникальные значения  каналов привлечения пользователей для данных о рекл
[print(i) for i in sorted(costs['channel'].unique())]
print()
```

```
Уникальные значения местоположения пользователей :
United States
UK
France
Germany

Уникальные значения устройств пользователей :
iPhone
Mac
Android
PC

Уникальные значения  каналов привлечения пользователей :
AdNonSense
FaceBoom
LeapBob
MediaTornado
OppleCreativeMedia
RocketSuperAds
TipTop
WahooNetBanner
YRabbit
lambdaMediaAds
organic

Уникальные значения  каналов привлечения пользователей для данных о рекламных
затратах:
AdNonSense
FaceBoom
LeapBob
MediaTornado
OppleCreativeMedia
RocketSuperAds
TipTop
WahooNetBanner
YRabbit
lambdaMediaAds
```

Возможных ошибок в написании не выявленно

Вывод:

Столбцы, которые должны содержать дату или дату и время переведены в нужный тип. Проверка на явные дубликаты не выявила оных. Произведен поиск возможных ошибочных написаний в столбцах с регионом проживания, устройствами, а также каналами привлечения пользователей. Данные собраны корректно, неявных дубликатов нет.

Исследовательский анализ данных

Создание пользовательских профилей

Зададим функцию для создания пользовательских профилей, которые будут содержать информацию о персональном индентификаторе пользователя, регионе проживания, устройстве, дате и времени первой сессии, канале привлечения, дате первого посещения и первое число месяца, когда это посещение произошло, призак плательщика и стоимость привлечения пользователя.

In [11]:
```
# 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'].astype('datetime64[M]')

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

Передадим в функцию датасеты с информацией о пользовательских сессиях, покупках пользователей и тратах на рекламу.

In [12]:
```
#предаем датасеты в функцию создания пользовательских профилей, сохраним резуль
profiles = get_profiles(session, orders, costs)
# выведем первые 5 строчек
display(profiles.head())
```

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

In [13]:
```
# выведем информацию о данных
profiles.info()
```

```
<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
```

Пользовательские профили созданы, всего 150008 уникальных пользователей, в данных пропусков нет, столбцы содержат корректные типы данных.

Максимальная и минимальная даты привлечения

Определим минимальную и максимальную даты привлечения

In [14]:
```
print('Минимальная дата привлечения: ',profiles['dt'].min())
print('Максимальная дата привлечения: ', profiles['dt'].max())
```

```
Минимальная дата привлечения:  2019-05-01
Максимальная дата привлечения:  2019-10-27
```

Распределение пользователей по регионам проживания

Построим таблицу, где посчитаем количество пользователей согласно их регионам проживания, количество платных пользователей, долю платных пользователей от общего числа пользователей конкретного региона.

In [15]:
```
# строим сводную таблицу, сгруппируем данные профилей по региону, посчитаем об
#  пользователей согласно признаку payer
country = profiles.groupby('region').agg({'user_id':'nunique', 'payer':['sum','
.droplevel(level=0, axis=1).reset_index().rename(columns = {'region':'страна',
                                  'sum':'количество платных пользователей',\
                                  'mean':'доля платных пользователей'})\
.sort_values(by = 'доля платных пользователей', ascending = False).round({'доля
```

In [16]:
```
country
```

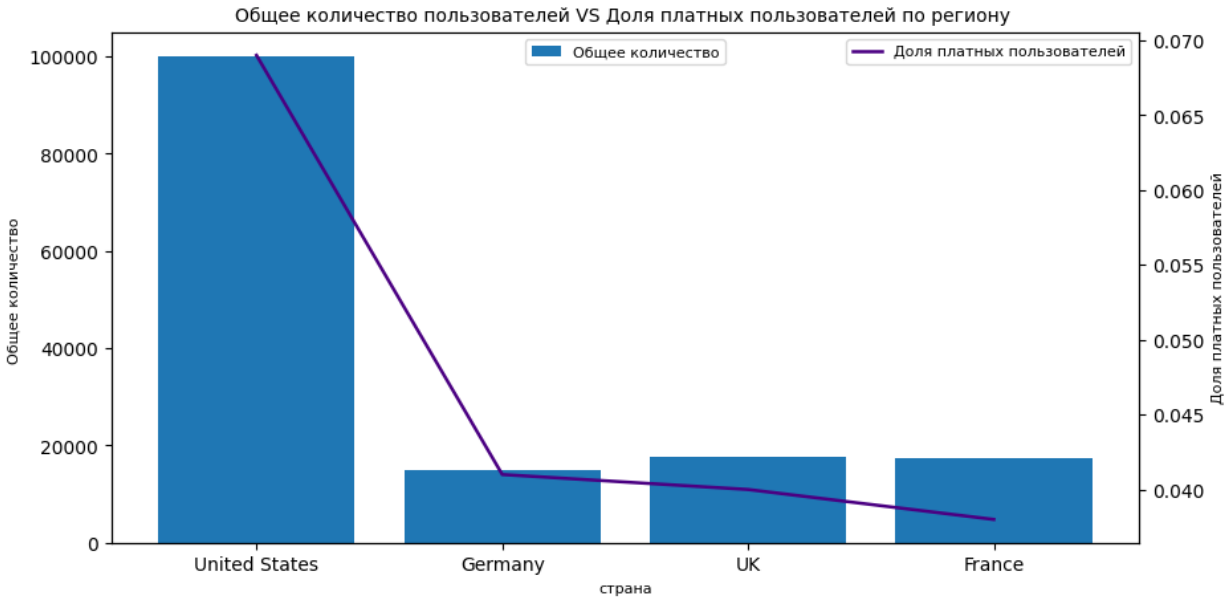
Out[16]:

	страна	общее количество пользователей	количество платных пользователей	доля платных пользователей
3	United States	100002	6902	0.069
1	Germany	14981	616	0.041
2	UK	17575	700	0.040
0	France	17450	663	0.038

In [17]:
```
# для наглядности построим визуализацию
fig = plt.figure(figsize = (15,10))
fig,ax1 = plt.subplots(figsize=(10,5))

# построим столбчатый график зависимости страны и  количетсва пользователей
ax1.bar(country['страна'],country['общее количество пользователей'])
ax1.set_xlabel('страна',size = 8 )
ax1.set_ylabel('Общее количество ',fontsize = 8)
ax1.set_title('Общее количество пользователей VS Доля платных пользователей по
ax1.grid(False)
ax1.legend(['Общее количество'],fontsize = 8,loc = 'upper center')

# добавим вторую ось
ax2 = ax1.twinx()

# построим линейный график зависимости региона и доли платных пользователй
ax2.plot(country.loc[:,['страна','доля платных пользователей']]\
         .set_index('страна'),linewidth=1.8, color = 'indigo',linestyle = '-',
ax2.set_ylabel('Доля платных пользователей',fontsize = 8)
```

```
ax2.legend(['Доля платных пользователей'],fontsize = 8,loc = 'upper right')
plt.show();
```

```
<Figure size 1500x1000 with 0 Axes>
```



Наибольшее количество пользователей из США, этой же стране принадлежит максимальная доля платных пользователей. Германия находится на последнем месте по численности, хотя доля платных пользователей выше, чем в Англии и Франции

Виды пользовательских устройств

Рассмотрим какие типы устройств чаще используются, какую долю составляют платные пользователи

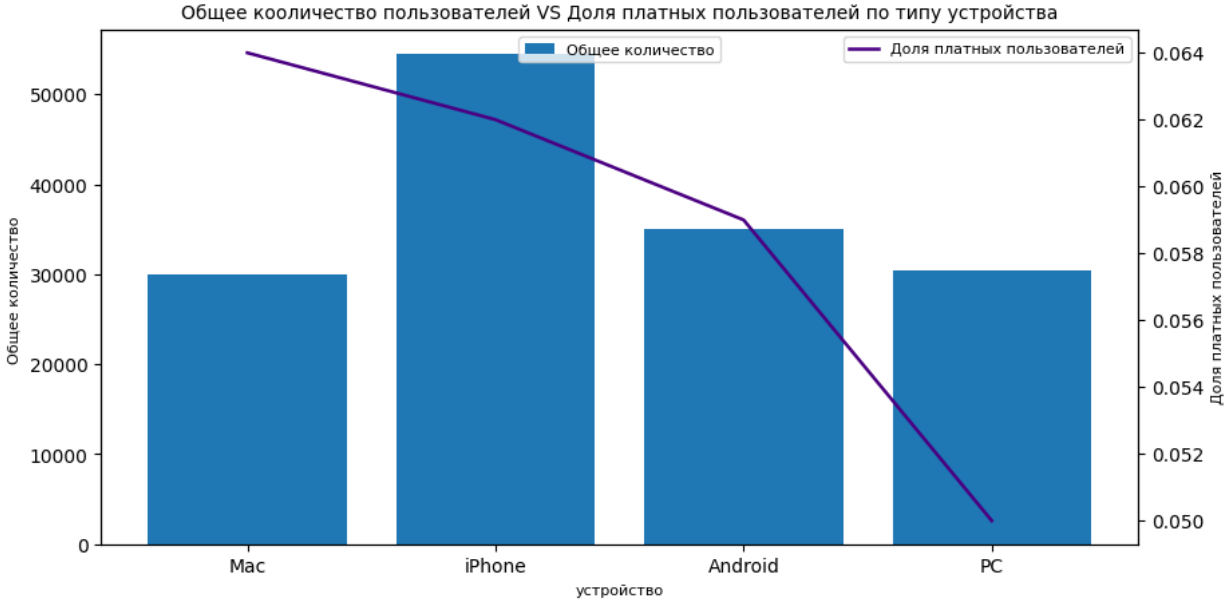
In [18]:
```
# построим сводную таблицу, посчимтаем количество пользователей, которые испол
# пользователей, их количество. Сгруппируем профили пользователей по столбцу de
device = profiles.groupby('device').agg({'user_id':'nunique', 'payer':['sum','m
.droplevel(level=0, axis=1).reset_index().rename(columns = {'device':'устройств
                                    'sum':'количество платных пользователей',\
                                    'mean':'доля платных пользователей'})\
.sort_values(by = 'доля платных пользователей', ascending = False).round({'доля
display(device)
```

	устройство	общее количество пользователей	количество платных пользователей	доля платных пользователей
1	Mac	30042	1912	0.064
3	iPhone	54479	3382	0.062
0	Android	35032	2050	0.059
2	PC	30455	1537	0.050

In [19]:
```
# для наглядности используем визуализацию
fig = plt.figure(figsize = (15,10))
fig,ax1 = plt.subplots(figsize=(10,5))
```

```
# построим столбчатый график зависимости устройства и  количетсва пользователей
ax1.bar(device['устройство'],device['общее количество пользователей'])
ax1.set_xlabel('устройство',size = 8 )
ax1.set_ylabel('Общее количество ',fontsize = 8)
ax1.set_title('Общее кооличество пользователей VS Доля платных пользователей по
ax1.grid(False)
ax1.legend(['Общее количество'],fontsize = 8,loc = 'upper center')
# добавим вторую ось
ax2 = ax1.twinx()

# построим линейный график зависимости устройства и доли платных пользователй
ax2.plot(device.loc[:,['устройство','доля платных пользователей']]\
         .set_index('устройство'),linewidth=1.8, color = 'indigo',linestyle = '
ax2.set_ylabel('Доля платных пользователей',fontsize = 8)
ax2.legend(['Доля платных пользователей'],fontsize = 8,loc = 'upper right')
plt.show();
```

```
<Figure size 1500x1000 with 0 Axes>
```



Наибольшее количество пользователей предпочитают в качестве девайса IPhone, вместе с тем доля платных пользователей самая высокая у Mac в сочетании с одним из самых низких показателей по общему количеству. Самая низкая доля платных пользователей у PC.

Распределение пользователей по каналу привлечения

Построим сводную таблицу, сгруппировав профили пользователей по каналу привлечения. Посчитаем общее количество пользователей, долю платных клиентов, их количество

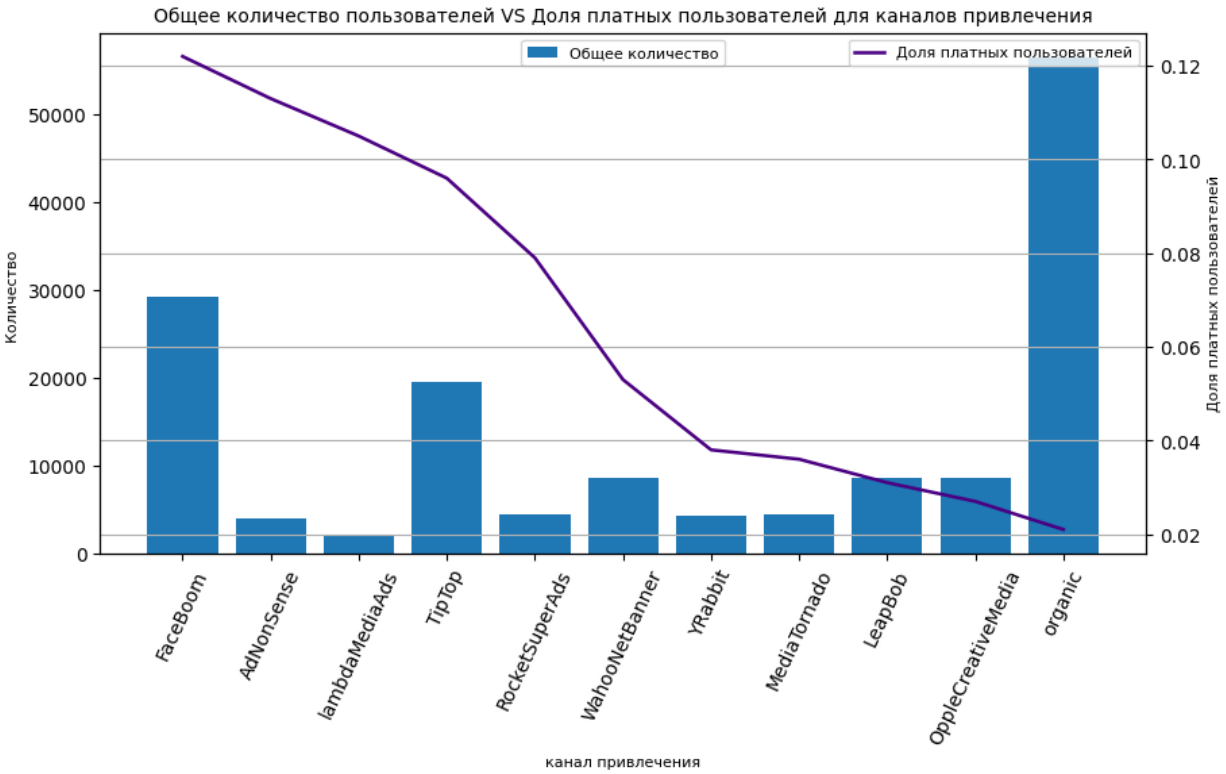
In [20]:
```
#строим сводную таблицу
channel = profiles.groupby('channel').agg({'user_id':'nunique', 'payer':['sum',
.droplevel(level=0, axis=1).reset_index().rename(columns = {'channel':'канал пр
                                                            'nunique':'общее ко
                                    'sum':'количество платных пользователей',\
                                    'mean':'доля платных пользователей'})\
```

```
.sort_values(by = 'доля платных пользователей', ascending = False).round({'доля
display(channel)
```

	канал привлечения	общее количество пользователей	количество платных пользователей	доля платных пользователей
1	FaceBoom	29144	3557	0.122
0	AdNonSense	3880	440	0.113
9	lambdaMediaAds	2149	225	0.105
6	TipTop	19561	1878	0.096
5	RocketSuperAds	4448	352	0.079
7	WahooNetBanner	8553	453	0.053
8	YRabbit	4312	165	0.038
3	MediaTornado	4364	156	0.036
2	LeapBob	8553	262	0.031
4	OppleCreativeMedia	8605	233	0.027
10	organic	56439	1160	0.021

In [21]:
```
# для наглядности построим визуализацию
fig = plt.figure(figsize = (15,10))
fig,ax1 = plt.subplots(figsize=(10,5))

# построим столбчатый график зависимости канала привлечения и  количетсва польз
ax1.bar(channel['канал привлечения'],channel['общее количество пользователей'])

ax1.set_xlabel('канал привлечения',size = 8 )
ax1.set_ylabel('Количество ',fontsize = 8)
ax1.set_title('Общее количество пользователей VS Доля платных пользователей для
ax1.grid(False)
ax1.tick_params(axis = 'x',rotation = 65)
ax1.legend(['Общее количество'],fontsize = 8,loc = 'upper center')
# добавим вторую ось
ax2 = ax1.twinx()

# построим линейный график зависимости канала привлечения и доли платных польз
ax2.plot(channel.loc[:,['канал привлечения','доля платных пользователей']]\
         .set_index('канал привлечения'),linewidth=1.8, color = 'indigo',linest
ax2.set_ylabel('Доля платных пользователей',fontsize = 8)
ax2.legend(['Доля платных пользователей'],fontsize = 8,loc = 'upper right')
plt.grid()
plt.show();
```

```
<Figure size 1500x1000 with 0 Axes>
```



Наибольшее количество пользователей пришло из канала Organic(вне платной рекламы) , в тоже время эти пользователи показывают самый низкий процент платных пользователей. Самая высокая доля платных пользователей пришла с канала привлечения FaceBoom. AdNonSense и lambdaMediads показывают хорошую долю платных пользователей, но показатели количества привлеченных пользователей одни из самых низких.

Вывод:

После создания профилей пользователей, определен период наблюдения - с 01 мая 2019 по 27 октября 2019. В исследовании участвуют пользователи из 4 стран : США, Англия, Франция, и Германия. Большее количество пользователей из США примерно 100 тыс из 150 008, процент платных пользователей 7%. Самый низкий показатель процента платных пользователей у Франции - около 4%. Виды пользовательских устройств представлены Mac, iPhone, Android, PC. Наибольшее количество пользуются iPhone, но процент платных пользователей выше у Mac, на уровне 6,4%. Самый низкий процент платных пользователей у PC - 5%. Пользователи распределены по 11 каналам привлечения, включая органических пользователей, их в исследовании больше всего (56439), доля платных пользователей самая низкая, на уровне 0,02, что достаточно неплохо при отсутствии затрат на них. Из рекламных каналов самое большое количество пользователей из FaceBoom, у этого же канала самый высокий процент платных пользователей 12%. Самое маленькое количество пользователей привлечено из канала lambdaMediaAds, но показатель платных пользователей высокий - 10,5%. Самый маленький процент платных пользователей среди рекламных каналов приносит OppleCreativeMedia - 2,7%

Маркетинговый анализ

Общая сумма расходов

In [22]:
```
#посчитаем общую сумму расходов
print("Общие маркетинговые расходы: ",round(costs['costs'].sum(), 1))
```

```
Общие маркетинговые расходы:  105497.3
```

Затраты по каналам привлечения

Посчитаем затраты по каждому каналу привлечения.Для этого сгруппируем данные в таблице `costs` по каналу привлечения, посчитаем суммарные затраты

In [23]:
```
# строим сводную таблицу
cost_channel = costs.groupby('channel').agg({'costs':'sum'}).reset_index()\
.rename(columns = {'channel':'канал привлечения','costs':'общая сумма затрат'})
.sort_values(by = 'общая сумма затрат', ascending = False).round({'общая сумма
display(cost_channel)
```

	канал привлечения	общая сумма затрат
6	TipTop	54751.30
1	FaceBoom	32445.60
7	WahooNetBanner	5151.00
0	AdNonSense	3911.25
4	OppleCreativeMedia	2151.25
5	RocketSuperAds	1833.00
2	LeapBob	1797.60
9	lambdaMediaAds	1557.60
3	MediaTornado	954.48
8	YRabbit	944.22

In [24]:
```
# строим визулизацию
fig = px.bar(cost_channel, x = 'канал привлечения', y = 'общая сумма затрат',\
                  title = 'Общие суммы затрат для каналов привлечения', text =
fig.show()
```

Очевидно, что около 80% затрат на рекламу приходядся на каналы TipTop и FaceBoom.Остальные канала привлечения сильно уступают по денежному объему. Визуализируем расходы на рекламу по каналам привлечения во времени

In [25]:
```
# строим сводную таблицу, где отобразим расходы на рекламу по каналам согласно
costs_time = costs.pivot_table(index = 'channel', columns = 'dt', values = 'cos
# построим график визулизации, где каждая линия - это канал привлечения
costs_time.plot(title ='Изменение расходов на рекламу во времени',figsize = (10
```

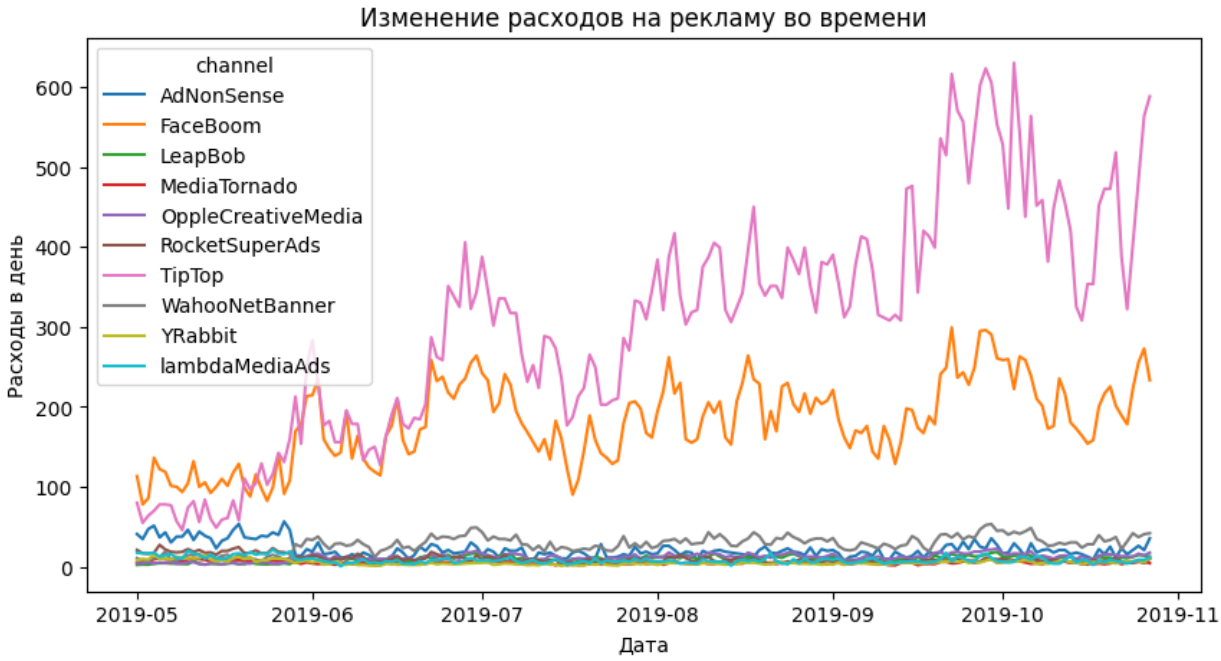


График изменения во времени показывает очевидный рост расходов на каналы TipTop и FaceBoom. Причем в начале исследуемого периода можно выделить тройку лидеров (канал AdNone Sense), но уже к июню месяцу остается 2 лидирующих канала. В июле можно выделить отрыв канала TipTop от конкурентов

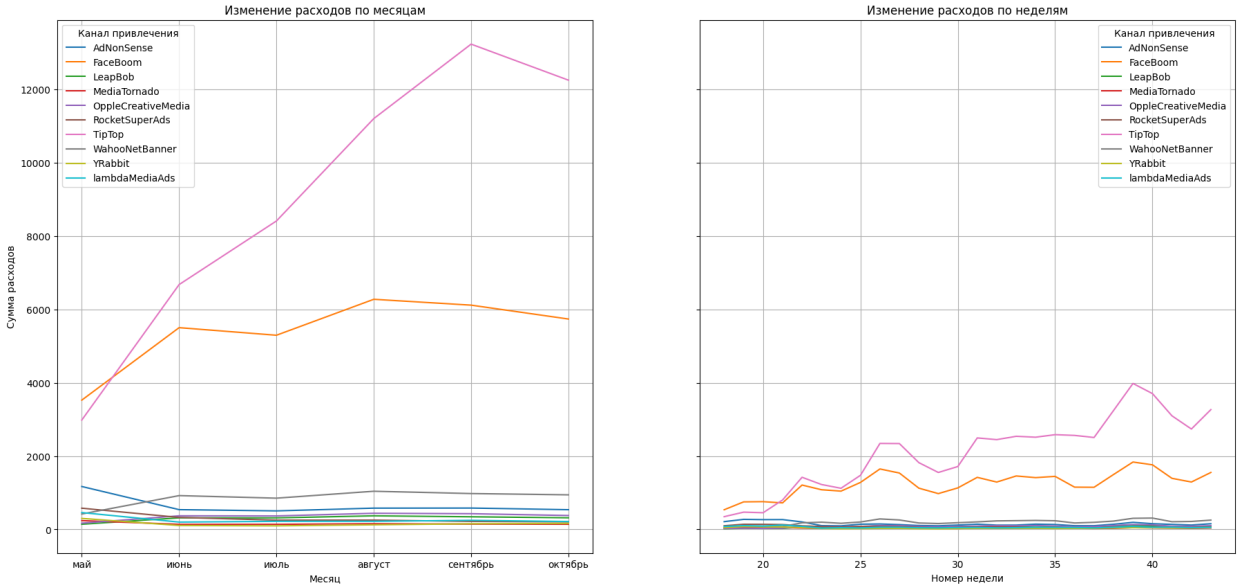
Посчитаем сумму расходов в разрезе каналов привлечения в месячный и недельный период

In [26]:
```
# создадим столбцы с номером месяца и недели
costs['month'] = pd.to_datetime(costs['dt']).dt.month
costs['week'] = pd.to_datetime(costs['dt']).dt.isocalendar().week
```

Построим визуализации для изменения расходов по месяцам и неделям

In [27]:
```
plt.figure(figsize=(22, 10))
#задаем первый график
ax1 = plt.subplot(1, 2, 1)
#создаем датасет, группируя по кналау, месяцу привлечения,считаем сумму расходо
month_cost = costs.pivot_table(index = 'channel', columns = 'month', values = '
# строим график зависимости расходов по месяцам
month_cost.plot(ax=ax1)
plt.xticks(month_cost.index,['май', 'июнь', 'июль', 'август', 'сентябрь', 'октя
plt.legend(title = 'Канал привлечения')
plt.grid()
plt.xlabel('Месяц')
plt.ylabel('Сумма расходов')
plt.title('Изменение расходов по месяцам')

#задаем местоположение 2 графика
ax2 = plt.subplot(1, 2, 2 , sharey=ax1)
#строим сводную тиаблицу, группируем по номеру недели, считаем сумму расходов
costs.pivot_table(index = ['week'], columns = 'channel', values = 'costs', aggf
plt.legend()
```

```
plt.xlabel('Номер недели')
plt.title('Изменение расходов по неделям')
plt.legend(title = 'Канал привлечения')

plt.show()
```



Исходя из визуализации можно говорить о том, что рост расходов канала TipTop стремительно рос с мая месяца, с пиком в сентябре на 39 неделе, после наблюдаем спад. Некоторые снижения наблюдались и на 24 и 29, но последующий рост сглаживает падение.Первоначально с небольшим отрывом по расходом лидировал FaceBoom.В мае после 21 недели ситуация меняется и большая часть расходов у канала TipTop. Остальные канала имеют примерно похожие суммы затрат на уровне до 1000.

Средняя стоимость привлечения одного клиента для каналов

Посчитаем средний CAC по всему проекту

In [28]:
```
print('Средний общий CAC равен: ', round(profiles.query('channel != "organic"')
```

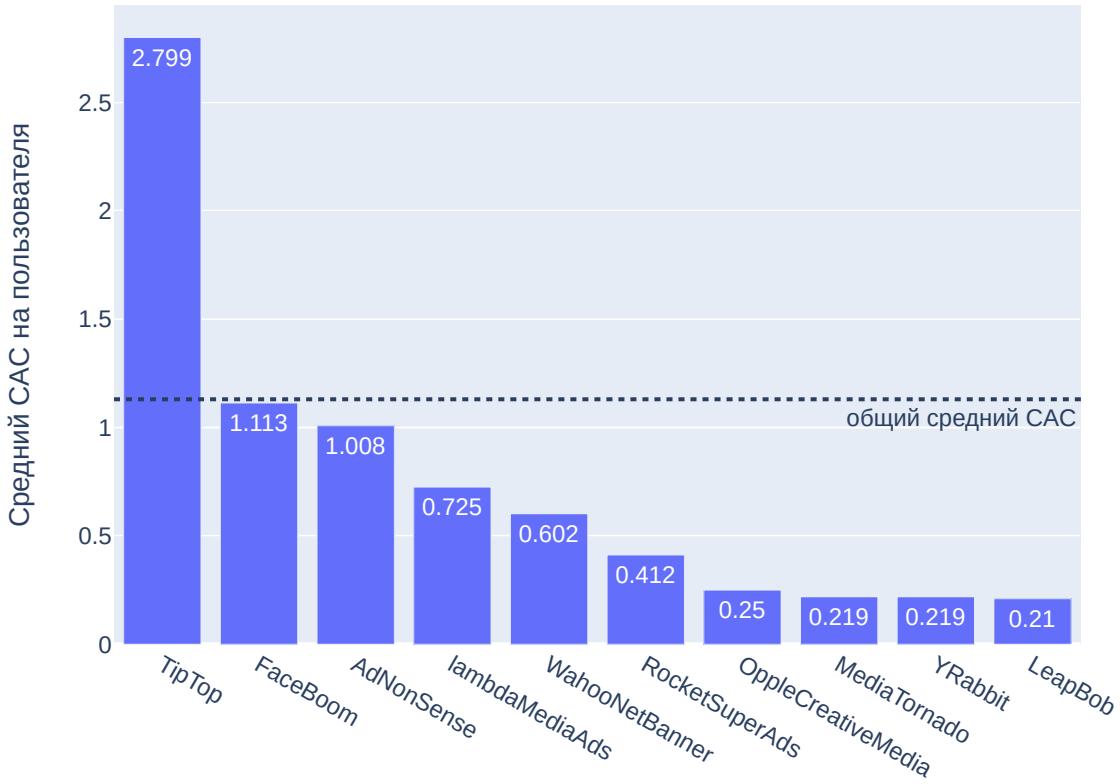
```
Средний общий CAC равен:  1.13
```

Посчитаем сколько в среднем приходилось затрат на привлечение 1 клиента для разных каналов привлечения, для этого сгруппируем данные профилей пользователей по каналу привлечения, посчитаем среднее по столбцу `acquisition_cost`

In [29]:
```
#создаем сводную таблицу, группируем по каналу привлечения, считаем среднюю сто
mean_cac = profiles.query('channel != "organic"').groupby('channel').agg({'acqu
.round({'acquisition_cost': 3})\
.sort_values(by = 'acquisition_cost', ascending = False).reset_index()\
.rename(columns = {'channel':'канал привлечения', 'acquisition_cost': 'Средний
# выводим таблицу
display(mean_cac)
# для удобства визуализируем данные
fig = px.bar(mean_cac, x ='канал привлечения', y = 'Средний CAC на пользовател
             title = 'Средний CAC на пользователя по каналам привлечения')
(y=round(profiles.query('channel != "organic"')['acquisition_cost'
```

```
                    mean(),2),line_dash="dot",annotation_text="общий средний
                annotation_position="bottom right")
fig.show()
```

	канал привлечения	Средний CAC на пользователя
0	TipTop	2.799
1	FaceBoom	1.113
2	AdNonSense	1.008
3	lambdaMediaAds	0.725
4	WahooNetBanner	0.602
5	RocketSuperAds	0.412
6	OppleCreativeMedia	0.250
7	MediaTornado	0.219
8	YRabbit	0.219
9	LeapBob	0.210



Как и ожидалось самая высокая стоимость привлечения у канала TipTop - 2,8, которая значительно превышаент среднюю по проекту. FaceBoom и AdNonSense имеют примерно одинаковую стоимость привлечения 1.008-1.113, самая низкая у канала LeapBob -

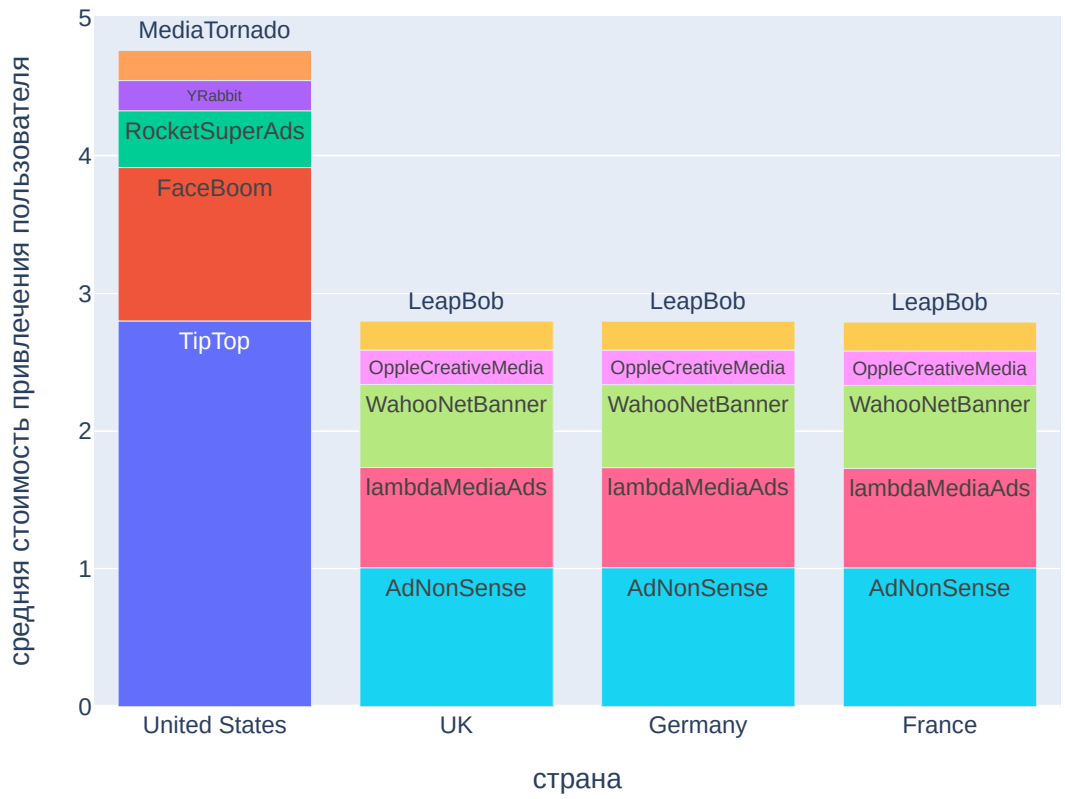
0.21.Построим сводную таблицу, где посчитаем средние расходы на рекламу по стране и каналу привлечения

In [30]:
```
# группируем пользователей по стране и каналу привлечения, считаем средний CAC
region_channel = profiles.query('channel != "organic"').groupby(['region','char
.sort_values(by = ['region','acquisition_cost'], ascending = False).rename_axis
rename(columns = {'acquisition_cost':'средняя стоимость привлечения'})
region_channel
```

Out[30]:

		средняя стоимость привлечения
регион	**канал привлечения**	
United States	**TipTop**	2.799003
	FaceBoom	1.113286
	RocketSuperAds	0.412095
	YRabbit	0.218975
	MediaTornado	0.218717
UK	**AdNonSense**	1.008224
	lambdaMediaAds	0.727142
	WahooNetBanner	0.602361
	OppleCreativeMedia	0.250059
	LeapBob	0.209983
Germany	**AdNonSense**	1.008435
	lambdaMediaAds	0.726176
	WahooNetBanner	0.602161
	OppleCreativeMedia	0.250091
	LeapBob	0.210380
France	**AdNonSense**	1.007553
	lambdaMediaAds	0.721211
	WahooNetBanner	0.602200
	OppleCreativeMedia	0.249862
	LeapBob	0.210189

In [31]:
```
# визуализируем для удобства анализа
fig = px.bar(region_channel.reset_index(), x = 'регион', y = 'средняя стоимость
             color = 'канал привлечения', text = 'канал привлечения',\
            title = 'Средняя стоимость привлечения пользователя для каналов в р
fig.update_xaxes(title_text = 'страна')
fig.update_yaxes(title_text = 'средняя стоимость привлечения пользователя')
fig.update_layout(showlegend=False)
fig.show()
```



Очевидно, что каналы привлечения делятся по территориальному признаку. Для европеской части канал AdNonSense показывает наибольшую сумму расходов. Среди аменриканских каналов привлечения самый высокий показатель - TipTop со средней стотимостью 2,8, а также FaceBoom с показателем на уровне 1,1. Суммарная средняя стоимость привлечения для каждой европейской страны примерно равна стоимости привлеченмя одного лишь канала TipTop в США

Вывод:

Общая сумма расходов на рекламу 105497.3. Из них на канал TipTop расходуется 54751.3, на канал FaceBoom 32445.6. Самая маленькая сумма у YRabbit - 944,22. Расходы на каналы TipTop и FaceBoom начали стремительно расти с середины мая 2019. При этом у канала TipTop крайне высокая средняя стоимость расходов на привлечения 1 пользователя - 2.8. На втором месте FaceBoom с показателем 1,1. Самая низкая стоимость привлечения у LeapBob - 0,2. Распределение каналов разделено на 2 рынка: американский и европейский. Очевидно 2 американских канала TipTop и FaceBoom имеют самую высокую стоимость на 1 пользователя, среди европейских каналов по стоимости привлечения 1 пользователя лидирует AdNonSense.

Оценка окупаемости рекламы

Анализ окупаемости рекламы, графики LTV и ROI, а также графики динамики LTV, CAC и ROI.

Так как в содержащихся данных наибольшее количество пользователей пришли органическим путем, для выявления имеено проблем окупаемости пользователей пришедших с платных каналов, не будем включать органических пользователей в анализ окупаемости

In [32]:
```
profiles = profiles.query('channel !="organic"')
```

Зададим функцию для расета LTV и ROI

In [33]:
```
# функция для расчёта LTV и ROI

def get_ltv(
    profiles,
    purchases,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
):

    # 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']
    ).dt.days
    # 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 CAC,
        # добавляя параметры из dimensions
        cac = df[['user_id', 'acquisition_cost'] + dims].drop_duplicates()

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

        # считаем ROI: делим LTV на CAC
        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 CAC в таблицу ROI
        roi['cac'] = cac['cac']

        # в финальной таблице оставляем размеры когорт, CAC
        # 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 + ['dt'], horizon_days
```

```
    )

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

In [34]:
```
# зададим дату начала анализа и горизонт
observation_date = datetime(2019, 11, 1).date()
# реклама должна окупаться через 14 дней, зададим горизонт анализа равный 14
horizon_days = 14
```

In [35]:
```
# передадим функции get_ltv профили пользователей, данные о покупках, дату анал
ltv_row, ltv, ltv_history, roi, roi_history = get_ltv(profiles, orders, observa
```

Получим таблицы с сырыми данными, таблицу LTV, таблицу динамики LTV, таблицу ROI, таблицу динамики ROI

In [36]:
```
max_date = datetime(2019, 11, 1).date() - timedelta(days = 13)
profiles.query('dt <=@max_date')['user_id'].count()
```

Out[36]: 88644

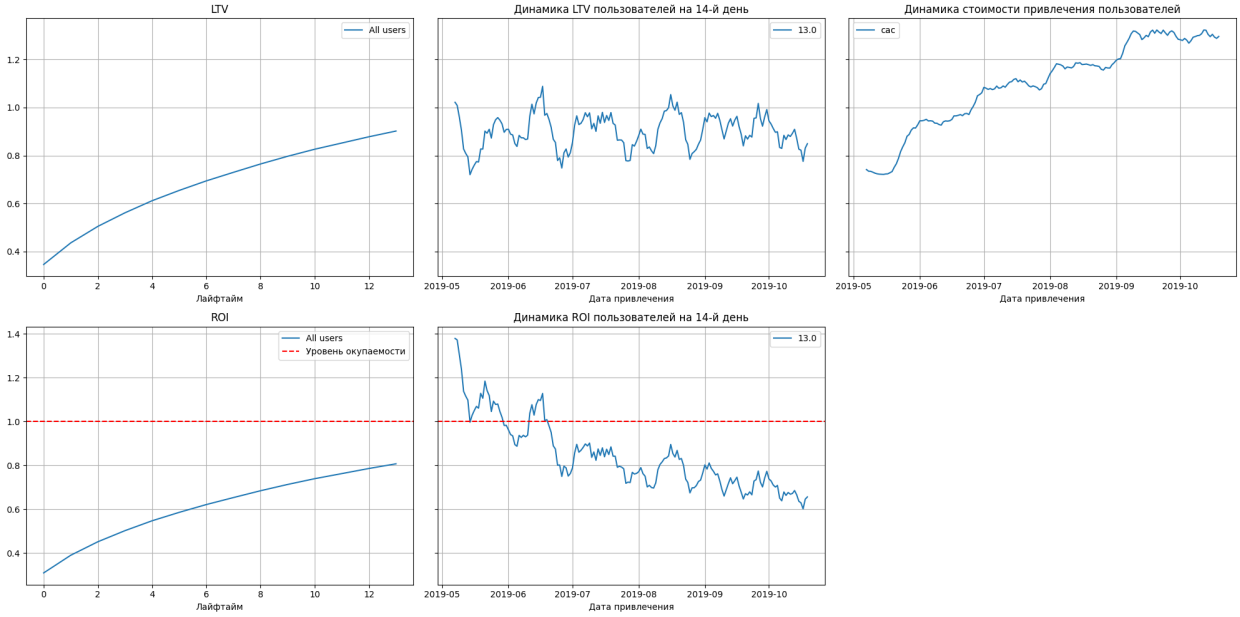
Зададим функции для отрисовки данных, для большей наглядности сгладим графики с помощью фуункции сглаживания.

In [37]:
```
# 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
```

In [38]:
```
#зададим функцию для визуализации ltv, динамики ltv, roi и динамики roi
def plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon, window=7):

    # 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
    ory = 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)
    plt.legend()
    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 ['dt']]
    filtered_data = ltv_history.pivot_table(
        index='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax2)
    plt.xlabel('Дата привлечения')
    plt.title('Динамика LTV пользователей на {}-й день'.format(horizon))

    # 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 ['dt']]
    filtered_data = cac_history.pivot_table(
        index='dt', columns=columns, values='cac', aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax3)
    plt.xlabel('Дата привлечения')
    plt.title('Динамика стоимости привлечения пользователей')

    # четвёртый график — кривые roi
    ax4 = plt.subplot(2, 3, 4)
    roi.T.plot(grid=True, ax=ax4)
    plt.axhline(y=1, color='red', linestyle='--', label='Уровень окупаемости')
    plt.legend()
    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 ['dt']]
    filtered_data = roi_history.pivot_table(
        index='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax5)
    plt.axhline(y=1, color='red', linestyle='--', label='Уровень окупаемости')
    plt.xlabel('Дата привлечения')
    plt.title('Динамика ROI пользователей на {}-й день'.format(horizon))

    plt.tight_layout()
    plt.show()
```

In [39]:
```
# передадим функции plot_ltv_roi полученные таблица, окно сглаживания зададим р
ltv, ltv_history, roi, roi_history, horizon_days, window=7)
```



Вывод:

Кривая LTV плавно растет, динамика изменения LTV пользователей на 14 день подвержена колебаниям, минимальгный LTV был у пользователей пришедших в середине мая. Динамика CAC же неуклонно растет, что означает увеличение стоимости привлечения при одниковом уровне "качества" пользователя. По динамике изменения ROI видно, что окупаемость имеет четкую тенденцию к снижению. Кривая ROI на 14 день так и не достигла уровня окупаемости, а значит реклама убыточна.

Конверсию пользователей и динамика её изменения.

Зададим функцию для расчета конверсии,и функцию для визуализации конверсии и ее динамики во времени, органических пользователей исключим из анализа, так как такие пользователи не расходуют рекламные средства.

In [40]:
```
# функция для расчёта конверсии

def get_conversion(
    profiles,
    purchases,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
):

    # 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')
```

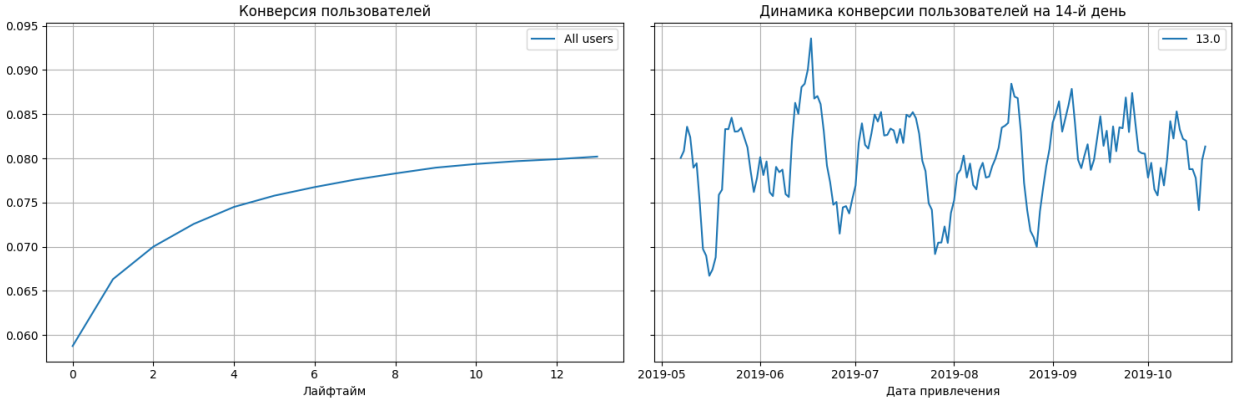
```
    # определяем дату и время первой покупки для каждого пользователя
    first_purchases = (
        purchases.sort_values(by=['user_id', 'event_dt'])
        .groupby('user_id')
        .agg({'event_dt': 'first'})
        .reset_index()
    )

    # 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']
    ).dt.days

    # 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 + ['dt'], horizon_days
    )

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

Получим таблицы сырых данных, таблицу конверсии и таблицу динамики конверсии

In [41]:
```
# передадим  в функцию данные профилей пользователей,покупки, момент и горизонт
conversion_row, conversion, conversion_history = get_conversion(profiles\
                                                                ,orders,observa
```

In [42]:
```
# функция для визуализации конверсии

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

    # 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)
    plt.legend()
    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 ['dt']
    ]
    filtered_data = conversion_history.pivot_table(
        index='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax2)
    plt.xlabel('Дата привлечения')
    plt.title('Динамика конверсии пользователей на {}-й день'.format(horizon))

    plt.tight_layout()
    plt.show()
```

In [43]:
```
# предадим в функцию визуализации таблицы конверсии, и ее динамики изменения во
plot_conversion(conversion, conversion_history, horizon_days,window=7)
```



Вывод:

Кривая конверсии имеет типичный вид и растет с 6% до 8%, динамика конверсии меняется во времени. Для пользователей привлеченных в середине мая, конверсия самая низкая, самая высокая у пользователей привлеченных в середине июня, динамика конверсии подвержена сезонным изменениям, но достаточно стабильна

Удержанием пользователей.

Для расчета удержания пользователей зададим функцию, удержание считаем с разделением по признаку платящих/неплатящих пользователей.

In [44]:
```
# функция для расчёта удержания

def get_retention(
    profiles,
    sessions,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
):

    # 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']
```

```
    # 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 + ['dt'], horizon_days
    )

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

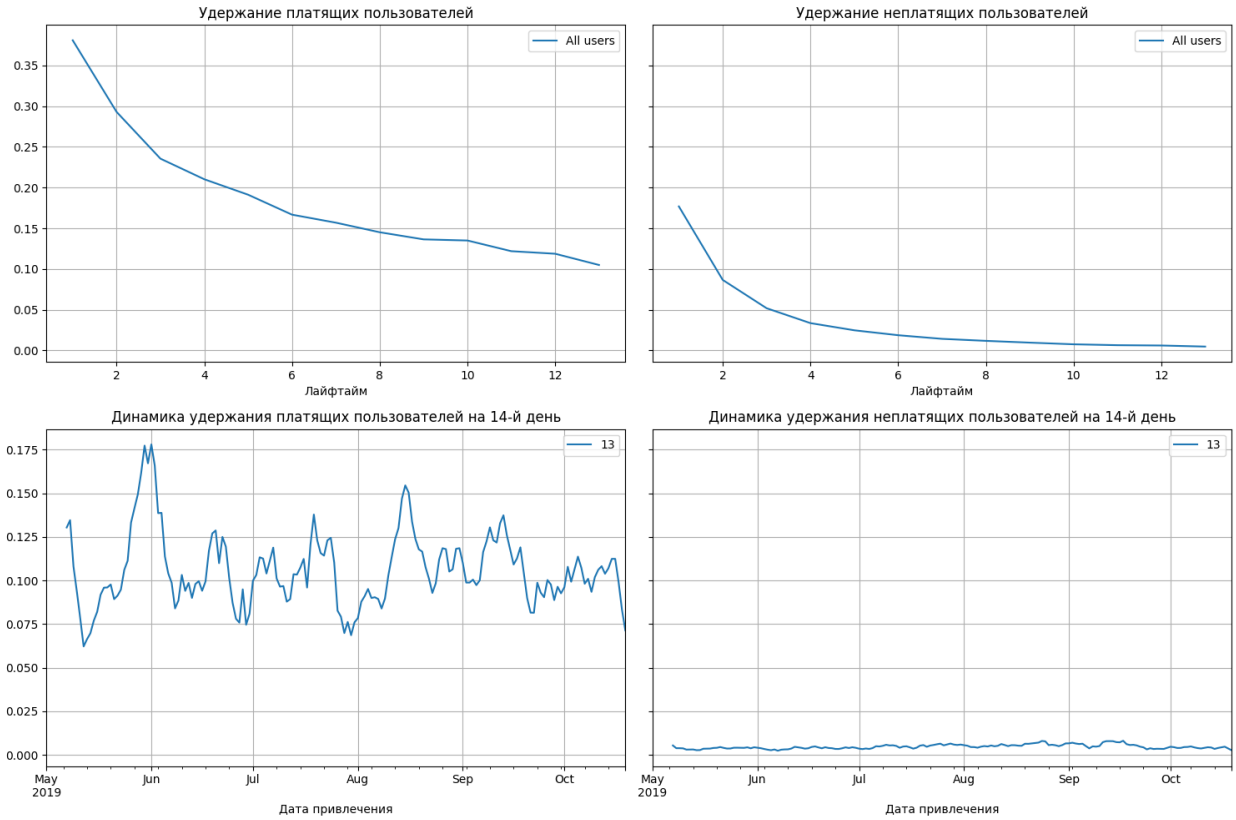
In [45]:
```
# передадим в функцию данные о профилях пользователей, сессиях, дату момента и
# получим таблицу сырых данных, таблицу удержания и динамику удержания
retention_raw, retention, retention_history = get_retention(profiles,session,ob
```

Определим функцию для визуализации удержания

In [46]:
```
# функция для визуализации удержания

def plot_retention(retention, retention_history, horizon, window=7):

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

    # исключаем размеры когорт и удержание первого дня
    retention = retention.drop(columns=['cohort_size', 0])
    # в таблице динамики оставляем только нужный лайфтайм
    retention_history = retention_history.drop(columns=['cohort_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
    )
    plt.legend()
    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
    )
    plt.legend()
    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 ['dt', 'payer']
    ]
    # фильтруем данные и строим график
    filtered_data = retention_history.query('payer == True').pivot_table(
        index='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax3)
    plt.xlabel('Дата привлечения')
    plt.title(
        'Динамика удержания платящих пользователей на {}-й день'.format(
            horizon
        )
    )

    # 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='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax4)
    plt.xlabel('Дата привлечения')
    plt.title(
        'Динамика удержания неплатящих пользователей на {}-й день'.format(
            horizon
        )
    )

    plt.tight_layout()
    plt.show()
```

In [47]:
```
# передадим в функцию визуализации удержания таблицу удержания, динамику удержа
n(retention, retention_history, horizon_days,window=7 )
```



Вывод:

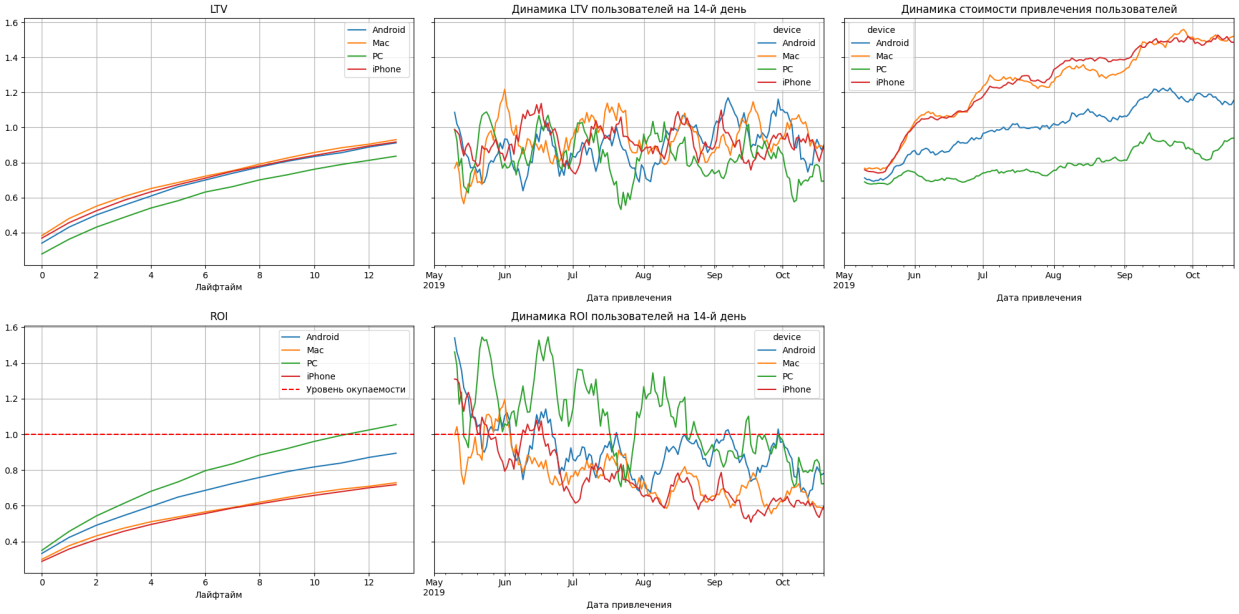
Удержание платящих пользователей ожидаемо выше неплатящих, но обе линии плавно снижаются, что соответсвует нормальной ситуации. Динамика неплатящих пользователей слабо меняется и достаточно стабильна. Динамика удержания платящих пользователей подвержена сезонному влиянию, минимальный показатель -около 6% у пользователей привлеченных в середине мая. Самый высокий показатель у пользователей привлеченных в конце мая - 17.5%

Окупаемость рекламы с разбивкой по устройствам.

Рассмотрим ценность пользователей, стоимость привлечения и окупаемость рекламы в разрезе типов устройств

In [48]:
```
# передадим функции get_ltv профили пользователей, данные о покупках, дату анал
ltv_row, ltv, ltv_history, roi, roi_history = get_ltv(profiles, orders, observa
```

In [49]:
```
# передадим функции plot_ltv_roi полученные таблица, окно сглаживания зададим р
plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon_days, window=10)
```



Вывод:

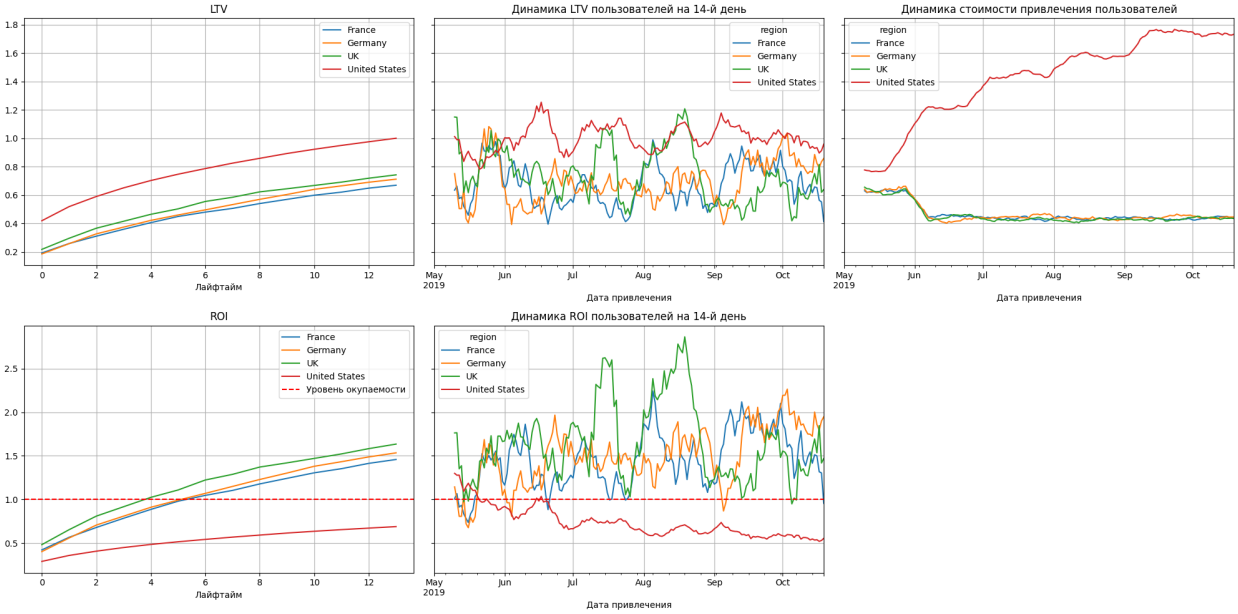
LTV пользователей на PC немного ниже, чем на других устройствах.Динамика изменения LTV пользователей на 14 день для всех устройств изменяется во времени,для PC LTV не всегда ниже, например для пользователей привлеченных в середине мая LTV самый высокий . Динамика CAC иммет явную тенденцию роста для всех устройств, особенно выделяются обладатели MAC и IPhone.Скачек произошел в середине мая. MAC и IPhone, Android не окупаются в отличие от PC, что видно по графику ROI.Динамика ROI имеет тенденцию к снижению для всех устройств, для пользователей Android,MAC и IPhone привлеченных после середины июня показатель ROI почти всегда ниже линии окупаемости.Для PC динамика ROI выглядит лучше всего, но для пользователей привлеченных после сентября ROI также опускается ниже уровня окупаемости за редким исключением. Можно предположить, что снижение ROI происходит из-за резкого скачка трат на рекламу, произошедшего в середине мая, что косвенно подтверждается динамикой изменения ROI для пользователей пришедших после середины мая(на графике видим резкое падение показателя). Так как общая тенденция для всех типов устройств одинакова - это рост динамики стоимости привлечения и снижение динамики окупаемости, можно предположить, что тип устройства не оказывает существенного влияния на окупаемость.

Окупаемость рекламы с разбивкой по странам.

Так как большую часть пользователей представляют жители США, проверим окупаемость рекламы в разных странах

In [50]:
```
# передадим функции get_ltv профили пользователей, данные о покупках, дату анал
ltv_row, ltv, ltv_history, roi, roi_history = get_ltv(profiles, orders, observa
```

In [51]:
```
# передадим функции plot_ltv_roi полученные таблица, окно сглаживания зададим р
plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon_days, window=10)
```



Вывод:

LTV пользователей из США стабильно выше, чем из стран европейского региона. По динамике LTV видно, что невсегда LTV пользователей из США выше , чем в других странах. Например для пользователей привлеченных в середине мая LTV выше у жителей Германии, как и в середине августа для пользователей UK. Динамика LTV подвержена сезонным изменениям по всем странам, хотя определенно LTV пользователей из США чаще выше.График динамики стоимости привлечения показывает примерно одинаковый уровень в европейских странах, видим даже некоторое снижение для пользователей пришедших в конце мая. Показатель пользователей США напротив имеет резкий подьем в середине мая,CAC только растет и сильно отличается от тенденции изменения CAC других стран. Анализируя график кривых ROI можно увидеть, что пользователи европейских стран стабильно и хорошо окупаются, в то время как показатель ROI пользователей из США так и не приблизился к линии окупаемости. Изучив график динамики ROI становится очевидно, что для пользователей привлеченных за весь наблюдаемый период в европейских странах затраты на рекламу чаще оправданы, в отличие от пользователей из США. Динамика ROI для них имеет тенденцию к снижению, и уже для пользователей привлеченных в июне месяце показатель ROI опускается ниже уровня окупаемости.Можно предположить, что убыточность компании связанна с возросшими расходами на рекламу у пользователей из США.

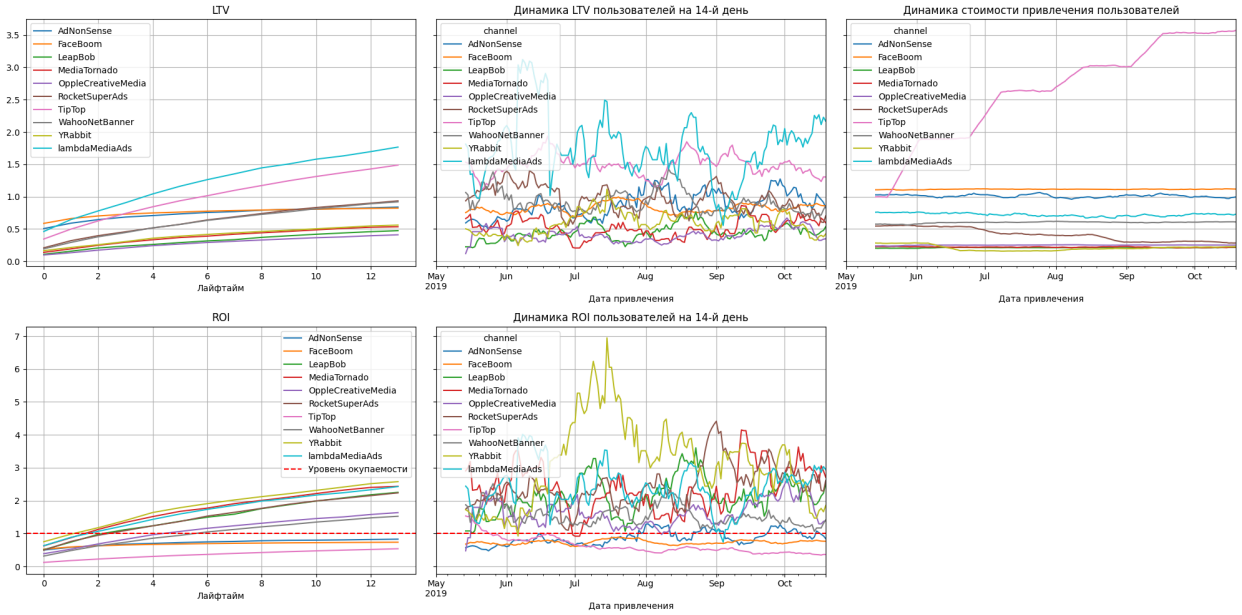
Окупаемость рекламы с разбивкой по каналам привлечения.

Известно, что рекламные источники делятся по территориальному признаку. Для США это TipTop, FaceBoom, RocketSuperAds, YRabbit, MediaTornado. Рекламные источники для европейских стран это каналы AdNonSense, lambdaMediaAds, WahooNetBanner, OppleCreativeMedia, LeapBob. Исследую окупаемость по странам и устройствам можно сделать предварительный вывод о завышенной стоимости привлечения пользователей из

США при относительно похожем качестве пользователя в сравнении с другими странами, уделим особое внимание на каналы привлечения из США.

In [52]:
```
# передадим функции get_ltv профили пользователей, данные о покупках, дату анал
#привлечения
ltv_row, ltv, ltv_history, roi, roi_history = get_ltv(profiles, orders, observa
```

In [53]:
```
# передадим функции plot_ltv_roi полученные таблица, окно сглаживания зададим р
plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon_days, window=14)
```



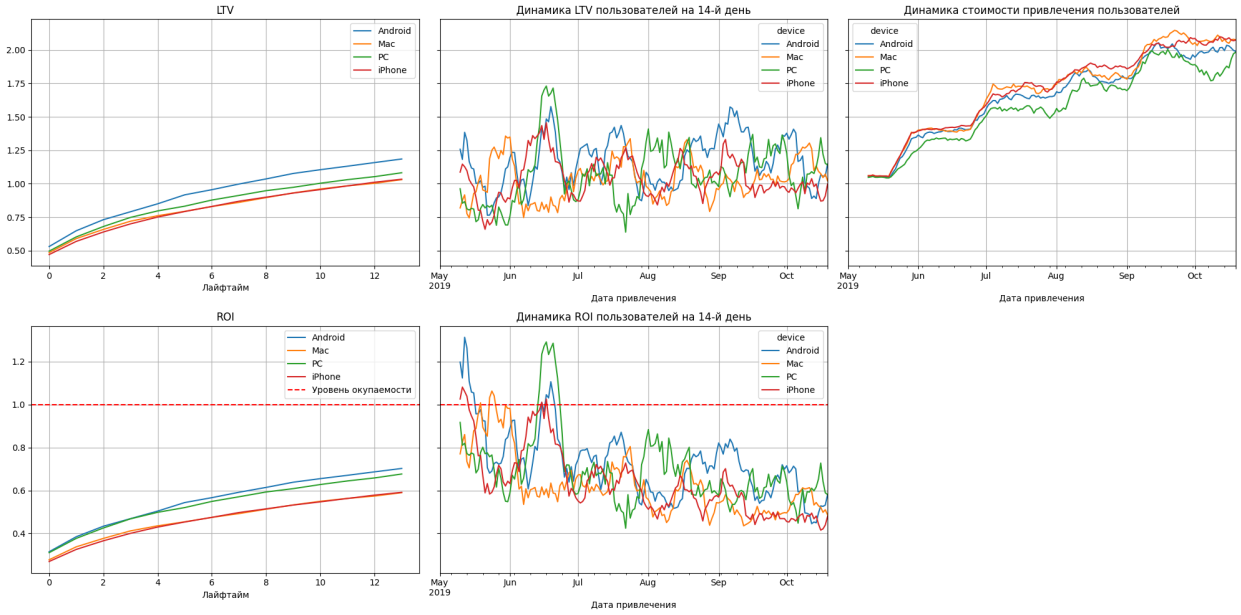
Вывод:

Согласно графику кривых LTV европейский lambdaMediaAds и каналы из США RocketSuperAds,TipTop имеют наиболее высокий уровень LTV к концу горизонта анализа. Динамика LTV пользователей подверженна сезонным изменениям по всем каналам, но особенно шумный график у канала lambdaMediaAds, который вцелом имеет более выскокий LTV, ощутимое падение было только у пользователей пришедших в начале сентября. Канал TipTop из США согласно динамике изменения LTV как правило находится на 2 месте по уровню показателя. Динамика изменения стоимости привлеченных пользователей относительно стабильна для всех каналов привлечения, кроме американского TipTop, чья стоимость привлечения неуклонно растет с начала изучаемого периода и к концу уже превышает показатель в 3.5 (для сравнения второй канал по уровню CAC с показателем чуть больше единицы и это тоже американский канал - FaceBoom). Подтвердим предположение, что убыточность рекламы связанна со стоимостью привлечения пользователя для американских каналов. На графике ROI можно заметить, что ни американские TipTop и FaceBoom, а также европейский AdNonSense не приблизились к уровню окупаемости, в отличие от всех остальных каналов. На графике динамики ROI можно заметить, что все пользователи пришедшие за изучаемый период с канала FaceBoom не перешли уровень окупаемости, пользователи пришедшие с канала TipTop в мае частично окупались, после все пришедшие меют низкий ROI, а начиная с пользователей пришедших с июля и вовсе

имеют самый низкий ROI из всех каналов.Европейский AdNonSense тоже имеет стабильно плохой результат, лишь изредка пересекая черту окупаемости для нескольких групп пользователей пришедших после августа.

Чтобы однозначно исключить влияние пользовательских устройств на показатели построим графики ценности, CAC и ROI только для пользователей, привлеченных с убыточных каналов

In [54]:
```
# передаем в функцию отфильтрованные по убыточным каналам данные
ltv_row, ltv, ltv_history, roi, roi_history = get_ltv(profiles.\
query('channel == "AdNonSense" or channel == "TipTop" or channel == "FaceBoom"'
                                                      orders, observation_date,
plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon_days, window=10)
```

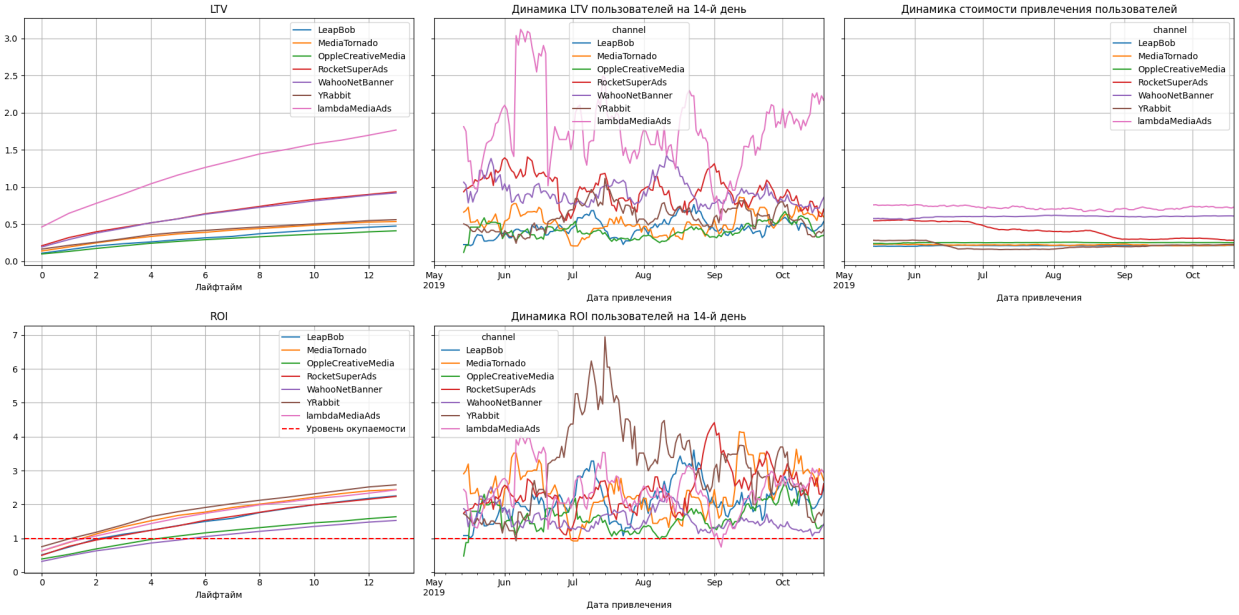


Можно утверждать, что тип устройства не влияет на показатели окупаемости, так как качество пользователя стабильно, CAC сильно растет, окупаемость сильно ниже желаемого уровня для всех типов устройств.

Рекомендации

Суммируя вышеизложенное можно говорить о завышенной стоимости привлечения пользователей для каналов TipTop,FaceBoom и европейского AdNonSense.Все три канала имеют одни из самых высоких долей платящих пользователей. Поэтому рекомендуется по возможности не отказываться от сотрудничества, но в срочном порядке пересмотреть систему оплаты привлечения. Рассмотрим показатели для удачных каналов привлечения

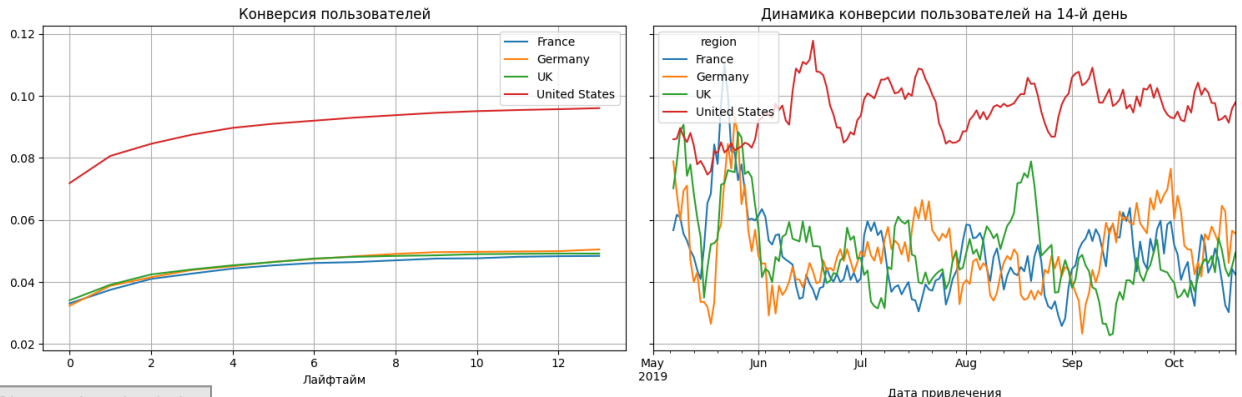
In [55]:
```
# посчитаем показатели для отфильтрованных по удачным каналам данных
ltv_row, ltv, ltv_history, roi, roi_history = get_ltv(profiles.\
query('channel != "AdNonSense" and channel != "TipTop" and channel != "FaceBoo
                                                      orders, observation_date,
plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon_days, window=14)
```



Можно заметить, что удачные каналы хорошо окупаются, ROI от 1,5 до 2,5. Стоит обратить внимание на канал lambdaMediaAds, который имеет высокий уровень LTV и даже при самом высоком показателе CAC имеет хороший уровень окупаемости. Американский RocketSuperAds имеет неплохой потенциал, так как расходы на пользователей снижаются, а "качество" приходящих на высоком уровне.Канал YRabbit, хотя и имеет небольшое количество привлеченных, для пользователей пришедших в середине июля показал высокий LTV при сниженнной стоимости привлечения, что дает высокий показатель ROI(больше 5). Возможно привлечение таких пользователей это не случайность, стоит лучше исследовать этот период, возможно найдутся некие точки роста. Также нужно помнить о большом количестве органических пользователей (пришедших не из платных рекламных каналов). Возможно персональные предложения для таких пользователей (например скидки, промокоды) помогут конвертировать больший процент платных пользователей для этого канала.

Рассчитаем конверсию и удержание в разрезе стран.

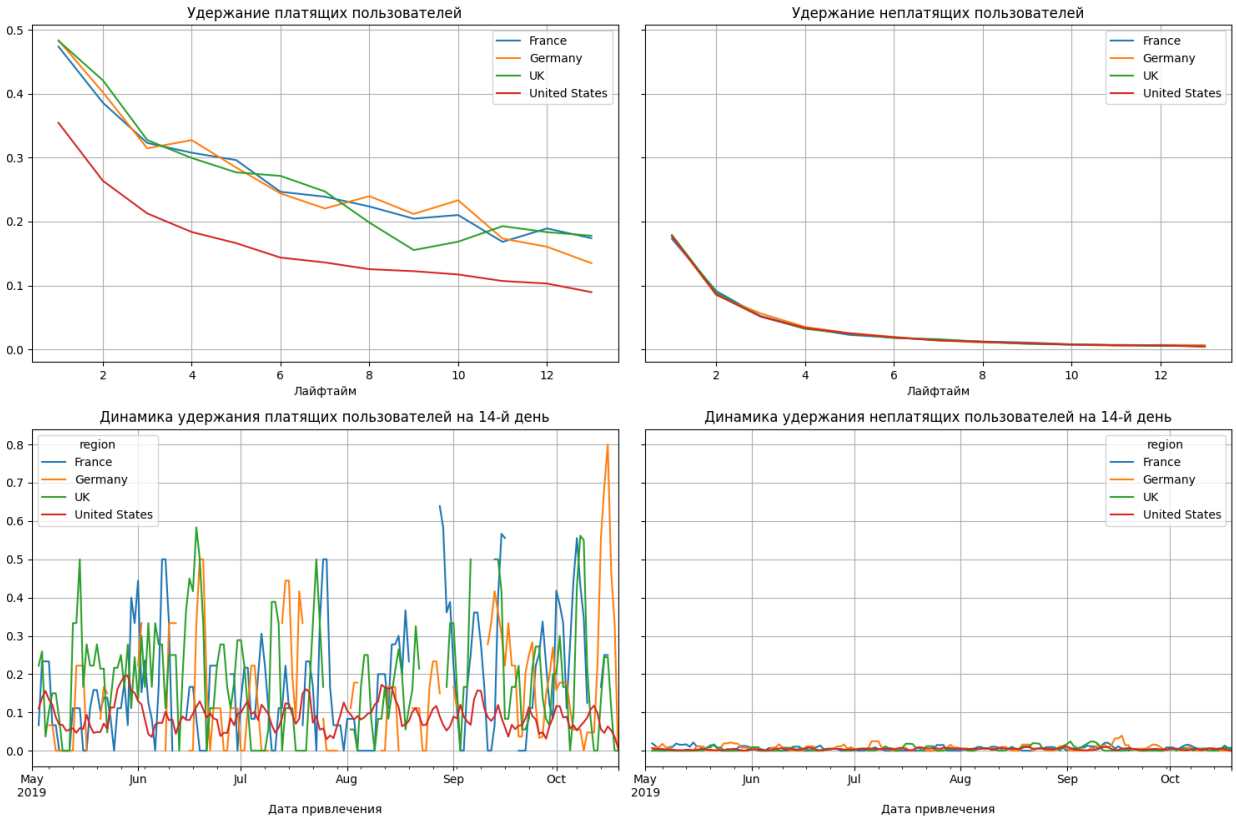
In [56]:
```
# передадим в функцию расчета конверсии признак разделения по регионам
conversion_row, conversion, conversion_history = get_conversion(profiles#.query
                                                                ,orders,observa
# визуализируем конверсию и ее динамику для стран
plot_conversion(conversion, conversion_history, horizon_days,window=7)
```



По кривым конверсии заметно, что пользователи из США конвертируются почти в 2 раза лучше, чем пользователи европейских стран. Для пользователей из США, привлеченных после июня месяца, конверсия стабильно выше. Динамика конверсии пользователей европейских стран имеет небольшую тенденцию к снижению. Можно предположить, что продукт пользуется большей популярностью в США, возможно находится в стадии набора скорости популярности. Это в свою очередь может означать, что и европейские пользователи могут как минимум достигнуть уровня конверсии пользователей из США при удачном выборе маркетинговой стратегии и каналов привлечения.

Посмотрим на удержание пользователей по странам

In [57]:
```
# передадим в функцию признак разделения по странам
retention_raw, retention, retention_history = get_retention(profiles,\
                        session,observation_date,horizon_days,dimensions=['regi
# визуализируем кривые удержание и динамику удержания, окно сглаживания оставим
plot_retention(retention, retention_history, horizon_days,window=3 )
```



Удержание и динимика удержания для неплатящих пользователей всех стран практически не отличается.Платящие пользователи из США реже возвращаются, чем пользователи из Европы. По графику динамики видно,что метрика платящих пользователей из США хоть и ниже и подвержена сезонному изменению, но достаточно стабильна. Пользователи из Европы обладают куда более хаотичным поведением. Удержание скачет для разных групп привлечения платящих пользователей. Усредненно кривая для европейских стран выше, но это может быть типичным только для наблюдаемого периода, так как поведение европейских пользователей сложнее предсказать. Нельзя однозначно утверждать, что платящие пользователи из США проблематичны с точки зрения удержания, скорее нужно

обратить внимание на возвращаемость пользователей из Европы, так как такому специфическому поведению могут найтись логичные причины, которые можно использовать для увеличения показателя удержания для этих стран.

ВЫВОД:

Для исследования есть 3 датасета, с сессиями пользователей, их тратами и расходами на рекламу. Данные не имеют пропусков.Произведена некоторая замена названий колонок датасетов, данные с датой и временем переведены в соответствующие типы.Явные и неявные дубликаты не выявлены.

На основе имеющихся данных собраны профили пользователей, включающие в себя описание первого посещения пользователя, источних привлечения, признак платящего пользователя, тип устройства и местоположение, стоимость привлечения, месяц и дату первого посещения.

Временной интревал пользовательских профилей включает даты с 1.05.2019 по 27.10.2019

Наибольшее количество пользователей находится в США, около 100000, в это же стране самый высокий процент пользователей, приносящих доход(платных пользоватялей) - 6,8%. Другие страны,а это Германия, Англия и Франция имеют значительно меньшее количество пользователей около 14тыс - 17тыс, и имеют похожий процент платных пользователей, в пределах 3,8 - 4,1%.

Чуть больше 54тыс человек пользуется IPHONE. Android предпочитают около 35тыс, у PC и Mac по 30тыс пользователей. Тем не менее наиболее высокий процент платных пользователей у Mac - 6,4%, наименьший у PC - 5%

Пользователи распределенны по 10 каналам привлечения + канал organic (самоприходящие пользователи), которые лидируют по количеству,чуть больше 56тыс пользователей и в то же время имеют самый низкий процент платных клиентов - 2,1%. Самаую большую долю платных клиентов принес канал FaceBoom с 12,2 % платных пользователей, AdNonSense и lambdaMediaAds имеют также хороший процент платящих - в среденем 11%, но значительно уступают в общем количестве пользователей 3880 и 2149 неользователей соответственно, последний имеет наименьшее общее количество в абсолютных показателях. Худший показатель платных клиентов это канал OppleCreativeMedia, оттуда пришли лишь 2,7% платных пользователей.

Общая сумма маркетинговых затрат составила 105497.3

Распределение расходов неравномерно, лидирующее звено это канал TipTop - около 50% от всей суммы расходов. Второе место у канала FaceBoom, расходы составляют чуть более 32тыс. Диапазон расходов остальных каналов привлечения около 1000-5000. До сентября TipTop показывал уверенный рост расходов, после небольшое снижение. Второй лидер FaceBoom ощутимо рос до июня месяца, позже показывал небольшой прирост

расходов. Расходы других каналов привлечения редко в среденем превышают 1000 в месяц.

Средняя стоимость привлечения одного пользователя у TipTop - 2.8, это самый высокий показатель, у второго по списку FaceBoom показатель на уровне 1,1, самая низкая средняя стоимость у канала LeapBob - около 0,2. YRabbit,MediaTornado и OppleCreativeMedia также имеют довольно низкую среднюю стоимость в пределах от 0,22 до 0,25.

Каналы привлечения также распределены по территориальному признаку: на каналы привлечения для США (TipTop, FaceBoom, RocketSuperAds, YRabbit, MediaTornado) и европейских стран (AdNonSense, lambdaMediaAds, WahooNetBanner, OppleCreativeMedia, LeapBob). При визуальной оценке можно сказать, что средняя стоимость по каналу TipTop соизмерима с общей стоимостью привлечения по каждой европейской стране, также высокой средней стоимостью отличается американский канал FaceBoom. Вообще расходы на рекламу по США значительно отличаются в большую сторону от европейских стран. По средней стоимости среди европейских каналов можно выделить канал AdNonSense.

Для анализа окупаемости выбран горизонт анализа - 14 дней, именно за этот срок затраты на рекламу должны окупиться. Кривая общего показателя LTV (без разделения на признаки) имеет нормальный вид и плавно растет от 0,1 до 0,7, динамика изменения LTV для пользователей хотя и имеет сезонные колебания, но показатель достаточно стабилен. Минимальный LTV показывают пользователи, присоединившиеся в середине мая. Примерно в этом периоде происходит резкий скачок стоимости привлечения для пользователей с 0,3 до 0,6, рост продолжается и дальше. Кривая ROI ниже линии окупаемости, значит затраты превышают прибыль. По динамике измениния ROI видно, что показатель снижается и уже для пользователей подключившихся с конца июня и далее затраты в инвестиции (рекламу) неоправданы. Реклама не окупается.

Расчет конверсии показывает, что пользователи хорошо конвертируются, динамика конверсии подвержена сезонному влиянию, но вполне стабильна. Удержание платных и неплатящих пользователей ожидаемо различается, динамика удержания платных пользователей изменяется во времени, но вполне стабильна,конверсия неплатящих изменяется вовсе незначительно.

Окупаемость рекламы с разбивкой пользователей по типу устройства показывает окупаемость только для устройств PC, но динамика изменения LTV у всех устройств имеет тенденцию только к сезонному колебанию, динамика стоимости привлечения напротив растет для все устройств, но кривая PC находится ниже других устройств, что может косвенно влиять на высокий уровень ROI(так как расходы в принципе меньше, "качество" пользователя примерно одинаковое у все устройств). На графике динамики изменения ROI можно отметить, что кривые для всех типов устройств в большей или меньшей степени имеют тенденцию к снижению. Можно сделать промежуточный вывод о низком влиянии типа устройств пользователя на окупаемость рекламы.

Рассмотрим окупаемость с точки зрения географического расположения. По кривым LTV можно определить лидера - это США, европейские страны по LTV схожи между собой, более низкие показатели у Франции. По динамике LTV заметно, что пользователи из США как правило имеют LTV выше для большинства дат присоединения, среди европейских стран наибольшему сезонному колебанию подвержен LTV пользователей из UK.Максимальный LTV - 1.2 у пользователей из США, привлеченных в середине июня, минимальный показатель - 0,4 у пользователей из Франции и Германии, привлеченных в разные даты. Наибольший инетерес представляет динамика CAC, которая показывает несоизмеримо большой скачок значения у пользователей из США начиная с середины мая, рост продолжается вплоть до конца изучаемого периода, максимальный CAC - 1.8, в тоже время стоимость привлечения для пользователей европейских стран наоборот имеет падение в середине мая со стабилизацией в июне на отметке в 0,4. Кривые ROI говорят о хорошей окупаемости у европейских стран, и об обратной ситуации с пользователями из США. На графике динамики изменения ROI можно отследить, что как правило ROI клиентов из Германии, Англии и Франции в любой день привлечения находится над чертой окупаемости. В тоже время возврат инвестиций у пользователей из США имеет тенденцию к снижению , окупились только пользователи привлеченные до середины мая и некоторые группы середины июня. Можно сделать вывод о чрезмерных затратах на привлечение пользователей из США.

Рассмотрев кривые LTV по каналам привлечения выделим лидеров - это американский TipTop и европейский lambdaMediaAds. Динамика изменения показателя подвержена сезонному влиянию. На графике изменения стоимости привлечения видим достаточно стабильную картину расходов для всех каналов, кроме американского TipTop, CAC которого планомерно растет до показателя 3,5, что значительно отличается от общей тенденции. Оценка ROI говорит о 3 каналах,которые не выходят на окупаемость, это уже знакомый TipTop, также американский FaceBoom и европейский AdNonSense. Эти каналы как раз показывали самый высокий показатель средней стоимости. Хотя каналы FaceBoom и AdNonSense не имеют ярковыраженной ненормальности, но соотношение ценности пользователя (недостаточно высокое) и расходов на рекламу(видимо больше чем нужно) не позволяет этим канал выйти на окупаемость. Проверка бизнес-показателй для TipTop,FaceBoom, AdNonSense в разрезе устройств позволяет однозначно отвергнуть влияние типа устройства на окупаемость, так как у данных каналов ни один тип устройств не вышел на окупаемость.

Рекомендованно пересмотреть систему оплаты привлечения для каналов TipTop,FaceBoom, AdNonSense, по возможности достичь договоренность о продолжении сотрудничества. Можно ориентироваться на максимальную стоимость привлечения 1 пользователя у успешного канала привлечения - это средняя стоимость равная 0,7. Для возможных точек роста предложенны для рассмотрения каналы lambdaMediaAds, RocketSuperAds и YRabbit. В связи с большим количеством органических пользователей рекомендованно рассмотреть возможность перераспределения рекламного бюджета для формирования персональных предложений, которые помогут конвертировать неплатящих органических пользователй в платных.