Исследование: "Поиск причин убытков развлекательного приложения Procrastinate Pro+"

ВВОДНЫЕ ДАННЫЕ: Несмотря на огромные вложения в рекламу развлекательного приложения Procrastinate Pro+, последние несколько месяцев компания терпит убытки.

Предоставлена информация о пользователях, привлечённых с 1 мая по 27 октября 2019 года:

- лог сервера с данными об их посещениях,
- выгрузка их покупок за этот период,
- рекламные расходы.

В бизнес-плане заложено, что пользователи должны окупаться не позднее чем через две недели после привлечения.

Текущая дата: 1 ноября 2019 г.

ЦЕЛЬ ПРОЕКТА: разобраться в причинах убытка и помочь компании выйти в плюс.

ПРЕДСТОИТ ИЗУЧИТЬ:

- 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.

ПЛАН РАБОТЫ:

1. Загрузка данных и подготовка их к анализу
    - изучение данных
    - предобработка данных: поиск пропусков и дубликатов, проверка на соответствие значений колонок типу данных(столбцы с датой и временем и другие).
2. Задать функции для расчёта и анализа LTV, ROI, удержания и конверсии.
3. Проведение исследовательского анализа данных:

- Составить профили пользователей. Определите минимальную и максимальную даты привлечения пользователей. Сформулировать выводы.
- Определить из каких стран пользователи приходят в приложение и на какую страну приходится больше всего платящих пользователей. Построить таблицу, отражающую количество пользователей и долю платящих из каждой страны. Сформулировать выводы.
- Определить какими устройствами пользуются клиенты и какие устройства предпочитают платящие пользователи. Построить таблицу, отражающую количество пользователей и долю платящих для каждого устройства. Сформулировать выводы.
- Изучить рекламные источники привлечения и определите каналы, из которых пришло больше всего платящих пользователей. Построить таблицу, отражающую количество пользователей и долю платящих для каждого канала привлечения. Сформулировать выводы.
- Вывод по разделу.

4. Маркетинг

- Расчет общей суммы расходов на маркетинг. Сформулировать выводы.
- Определить как траты распределены по рекламным источникам, то есть сколько денег потратили на каждый источник. Сформулировать выводы.
- Построить график с визуализацией динамики изменения расходов во времени по неделям по каждому источнику. Затем на другом графике визуализировать динамику изменения расходов во времени по месяцам по каждому источнику. Сформулировать выводы.
- Определить, сколько в среднем стоило привлечение одного пользователя (CAC) из каждого источника. Использовать профили пользователей.Сформулировать выводы.
- Вывод по разделу.

5. Оценка окупаемости рекламы.

- Провести анализ окупаемости рекламы, опираясь на графики LTV, ROI и CAC.
- Анализ окупаемости рекламы с помощью графиков LTV и ROI, а также графики динамики LTV, CAC и ROI.

- Проверить конверсию пользователей и динамику её изменения. Проверить удержание пользователей и динамику изменения удержания. Построить и изучить графики конверсии и удержания.
- Проанализировать окупаемость рекламы с разбивкой по устройствам. Построить графики LTV и ROI, а также графики динамики LTV, CAC и ROI.
- Проанализировать окупаемость рекламы с разбивкой по странам. Построить графики LTV и ROI, а также графики динамики LTV, CAC и ROI.
- Проанализировать окупаемость рекламы с разбивкой по рекламным каналам. Построить графики LTV и ROI, а также графики динамики LTV, CAC и ROI.
- Ответить на такие вопросы:
    - Окупается ли реклама, направленная на привлечение пользователей в целом?
    - Какие устройства, страны и рекламные каналы могут оказывать негативное влияние на окупаемость рекламы?
    - Чем могут быть вызваны проблемы окупаемости?
- итоги оценки: описать возможные причины обнаруженных проблем и промежуточные рекомендации для рекламного отдела.

6. Общий вывод

- Выделить причины неэффективности привлечения пользователей.
- Сформулировать рекомендации для отдела маркетинга.

1 Загрузка данных и подготовка их к анализу

1.1 Импортируем нужные библиотеки, загрузим данные и преобразуем значения в столбцах.

Пути к файлам

- визиты: `/datasets/visits_info_short.csv`. Скачать датасет;
- заказы: `/datasets/orders_info_short.csv`. Скачать датасет;
- расходы: `/datasets/costs_info_short.csv`. Скачать датасет.

datasets/visits_info_short.csv - хранит записи событий и сообщений, создаваемые программой или системой во время ее работы - лог сервера с информацией о посещениях сайта.
/datasets/orders_info_short.csv - хранит информацию о заказах.
/datasets/costs_info_short.csv - хранит информацию о расходах на рекламу.

1.1.1 Импортируем нужные библиотеки, загрузим данные

In [1]:

```
import pandas as pd
import numpy as np

from datetime import datetime, timedelta
from matplotlib import pyplot as plt
import matplotlib
%matplotlib inline

import plotly.express as px
from plotly.offline import iplot

import seaborn as sns

try:
   visits, orders, costs = (
       pd.read_csv('https: ссылка удалена .csv'),  # визиты
       pd.read_csv('https: ссылка удалена .csv'),  # заказы
       pd.read_csv('https: ссылка удалена .csv')  # рекламные расходы
   )
except:
   visits, orders, costs =(
```

```
        pd.read_csv('/datasets/visits_info_short.csv'),  # визиты
        pd.read_csv('/datasets/orders_info_short.csv'),  # заказы
        pd.read_csv('/datasets/costs_info_short.csv')  # рекламные расходы
    )
```

Посмотрим на несколько случайных строк в загруженных таблицах: visits, orders, costs.

In [2]:

```
 # n=2, random_state=2 заданном  значении выборка всегда будет содержать одни и те же строки.
display(visits.sample(n=2, random_state=2), orders.sample(n=2, random_state=2), costs.sample(n=2, random_state=2))
```

	User Id	Region	Device	Channel	Session Start	Session End
145597	83159630018	United States	PC	organic	2019-09-15 01:27:43	2019-09-15 02:27:50
4704	163725528596	United States	iPhone	organic	2019-05-06 22:55:36	2019-05-06 23:12:26

	User Id	Event Dt	Revenue
18906	161220173130	2019-09-20 09:25:06	4.99
26311	950770323895	2019-10-24 02:45:51	4.99

	dt	Channel	costs
1676	2019-06-26	lambdaMediaAds	4.8
1528	2019-07-28	WahooNetBanner	29.4

В таблицах visits, orders, costs названия столбцов даны с нарушением стиля:

- не соблюден змеиный регистр
- присутствуют прописные буквы, а не lowercase.
-

1.1.2 Преобразуем название столбцов датафреймов в соответствии со стилем написания.

Для приведения написания заголовков колонок в змеином регистре и строчными буквами сделаем следующее:

- методом str.lower() преобразуем в нижний регистр наименование столбцов;
- методом str.replace() заменяем пробелы на нижнее подчеркивание в наименовании столбцов.

После выведем список с названием столбцов в датафрейме и убедимся, что они теперь соответствует стилю написания.

In [3]:

```
visits.columns = visits.columns.str.replace(' ', '_').str.lower()
orders.columns = orders.columns.str.replace(' ', '_').str.lower()
costs.columns = costs.columns.str.lower()

print(
'Названия столбцов датафрейма visits после изменения:\n', list(visits.columns), '\n\
Названия столбцов датафрейма orders:', list(orders.columns), '\n\
Названия столбцов датафрейма costs:', list(costs.columns)
)
```

Названия столбцов датафрейма visits после изменения:
['user_id', 'region', 'device', 'channel', 'session_start', 'session_end']
Названия столбцов датафрейма orders: ['user_id', 'event_dt', 'revenue']
Названия столбцов датафрейма costs: ['dt', 'channel', 'costs']

Названия колонок изменены, теперь они в едином стиле. Можно выполнить предобработку данных.

1.2 Предобработка таблицы: visits - визиты

Выведем на экран датафрейм visits. Посмотрим сколько строк он содержит, есть ли в данных пропуски и явные дубликаты, убедимся, что типы данных во всех колонках соответствуют сохранённым в них значениям. Обратим внимание на столбцы с датой и временем.Выполним предобработку.

1.2.1 Изучим датафрейм visits.

In [4]:

```
name = 'visits(визиты)' # для вывода на экран
df = visits # для вывода на экран

display(df.sample(n=2, random_state=2))
print(f'Сводная информация о таблице {name}:\n')
print(df.info(), '\n')
print(f'Датафрейм {name} содержит строк: {len(df)};\n\
количество явных дубликатов в {name}: {df.duplicated().sum()} шт.;\n\
количество пропусков в {name}: {df.isnull().values.sum()}.')
```

	user_id	region	device	channel	session_start	session_end
145597	83159630018	United States	PC	organic	2019-09-15 01:27:43	2019-09-15 02:27:50
4704	163725528596	United States	iPhone	organic	2019-05-06 22:55:36	2019-05-06 23:12:26

Сводная информация о таблице visits(визиты):

<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

Датафрейм visits(визиты) содержит строк: 309901;
количество явных дубликатов в visits(визиты): 0 шт.;
количество пропусков в visits(визиты): 0.

1.2.2 Поиск неявных дубликатов в таблицах visits.

Проверим столбцы region, device и channel, на уникальность значений. Так как они содержат строчные данные с названиями стран, девайсов и каналов.

In [5]:

```
print(f"Уникальные значения в таблице visits:\nв столбце region:
{list(visits['region'].unique())},\n\
в столбце device: {list(visits['device'].unique())},\n\
в столбце channel: {list(visits['channel'].unique())}.")
```

Уникальные значения в таблице visits:
в столбце region: ['United States', 'UK', 'France', 'Germany'],
в столбце device: ['iPhone', 'Mac', 'Android', 'PC'],
в столбце channel: ['organic', 'TipTop', 'RocketSuperAds', 'YRabbit', 'FaceBoom', 'MediaTornado', 'AdNonSense', 'LeapBob', 'WahooNetBanner', 'OppleCreativeMedia', 'lambdaMediaAds'].

Проверка уникальные значений таблицы visits в столбцах region, device, channel не выявила противоречивых названий и признаков наличия неявных дубликатов.

1.2.3 Изменим тип данных столбцов в таблице visits

Приведем тип данных столбцов: session_start и session_end - в правильный формат, так как текущий тип данных object- не соответствует значением даты и времени.

Используя apply() для преобразования нескольких столбцов фрейма (session_start, session_end) функцией to_datetime() преобразуем тип данных в нужный формат.
После проверим, что все получилось - выведем тип данных на экран.

In [6]:

```
visits[['session_start', 'session_end']] = visits[['session_start', 'session_end']]\
                                           .apply(pd.to_datetime)

orders['event_dt'] = pd.to_datetime(orders['event_dt'])

name = 'session_start' #  для вывода на экран
name1 = 'session_end' #  для вывода на экран
df = visits['session_start']  # для вывода на экран
df1 = visits['session_end']  # для вывода на экран

print(f"После преобразования тип данных:\nв колонке {name} - {df.dtypes}\n\
в колонке {name1} -{df1.dtypes}.")
```

После преобразования тип данных:
в колонке session_start - datetime64[ns]
в колонке session_end -datetime64[ns].

1.2.4 Результат предобработки датафрейма visits.

Предобработка датафрейма visit завершена. Произведена замена типа данных и названий колонок в датафрейме.
Таблица visits содержит 309901 строк, явных и неявных дубликатов и пропусков в датафрейме нет.
Содержимое колонок приведено к соответствующему типу данных.
Названия столбцов датафрейма visits после предобработки с кратким описанием содержащихся в них данных:

- user_id — уникальный идентификатор пользователя,
- region — страна пользователя,
- device — тип устройства пользователя,
- channel — идентификатор источника перехода,
- session_start — дата и время начала сессии,
- session_end — дата и время окончания сессии.

Предварительно можно утверждать, что в таблице содержится достаточно данных о логах сервера с информацией о посещениях сайта для проведения исследования.

1.3 Предобработка orders - заказы

Выведем на экран датафрейм orders. Посмотрим сколько строк он содержит, есть ли в данных пропуски и явные дубликаты, убедимся, что типы данных во всех колонках соответствуют сохранённым в них значениям. Обратим внимание на столбцы с датой и временем. Выполним предобработку.

1.3.1 Изучим датафрейм orders

In [7]:

```
name = 'orders(заказы)'  # для вывода на экран
df = orders  # для вывода на экран

display(df.sample(n=2, random_state=2))
print(f'Сводная информация о таблице {name}:\n')
print(df.info(), '\n')
print(f'Датафрейм {name} содержит строк: {len(df)};\n\
количество явных дубликатов в {name}: {df.duplicated().sum()} шт.;\n\
количество пропусков в {name}: {df.isnull().values.sum()}.')
```

	user_id	event_dt	revenue
18906	161220173130	2019-09-20 09:25:06	4.99
26311	950770323895	2019-10-24 02:45:51	4.99

Сводная информация о таблице orders(заказы):

```
<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
```

Датафрейм orders(заказы) содержит строк: 40212;
количество явных дубликатов в orders(заказы): 0 шт.;
количество пропусков в orders(заказы): 0.

Тип данных столбца: event_dt - object, это не соответствует значению даты и времени, которому он содержит.

1.3.2 Изменим тип данных столбцов в таблице orders

Используем функцию to_datetime() и преобразуем тип данных в нужный формат.
После проверим, что все получилось - выведем тип данных на экран.

In [8]:

```
orders['event_dt'] = pd.to_datetime(orders['event_dt'])

name = 'event_dt'  # для вывода на экран
df = orders['event_dt']  # для вывода на экран
print(f"После преобразования тип данных:\nв колонке {name} - {df.dtypes}.")
```

После преобразования тип данных:
в колонке event_dt - datetime64[ns].

1.3.3 Результат предобработки датафрейма orders.

Предобработка датафрейма orders завершена. Произведена замена типа данных и названий колонок в датафрейме.
Таблица orders содержит 40212 строк, явных дубликатов и пропусков в датафрейме нет. Содержимое колонок приведено к соответствующему типу данных. Строковых данных таблица не содержит, поиск неявных дубликатов не нужен.
Названия столбцов датафрейма orders после предобработки с кратким описанием содержащихся в них данных:

- user_id — уникальный идентификатор пользователя,
- event_dt — дата и время покупки,
- revenue — сумма заказа.

Предварительно можно утверждать, что в таблице содержится достаточно данных о заказах для проведения исследования.

1.4 Предобработка costs - расходы на рекламу

Выведем на экран датафрейм costs. Посмотрим сколько строк он содержит, есть ли в данных пропуски и явные дубликаты, убедимся, что типы данных во всех колонках соответствуют сохранённым в них значениям. Обратим внимание на столбцы с датой и временем. Выполним предобработку.

1.4.1 Изучим датафрейм costs

In [9]:

```
name = 'costs(расходы на рекламу)'  # для вывода на экран
df = costs  # для вывода на экран
```

```
display(df.sample(n=2, random_state=2))
print(f'Сводная информация о таблице {name}:\n')
print(df.info(), '\n')
print(f'Датафрейм {name} содержит строк: {len(df)};\n\
количество явных дубликатов в {name}: {df.duplicated().sum()} шт.;\n\
количество пропусков в {name}: {df.isnull().values.sum()}.')
```

	dt	channel	costs
1676	2019-06-26	lambdaMediaAds	4.8
1528	2019-07-28	WahooNetBanner	29.4

Сводная информация о таблице costs(расходы на рекламу):

<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

Датафрейм costs(расходы на рекламу) содержит строк: 1800;
количество явных дубликатов в costs(расходы на рекламу): 0 шт.;
количество пропусков в costs(расходы на рекламу): 0.

1.4.2 Поиск неявных дубликатов в таблицах costs.

Проверим столбец со строчными данными на уникальность значений: channel

In [10]:

```
print(f"Уникальные значения в таблице costs:\nв столбце
channel:\n{list(costs['channel'].unique())}.")
```

Уникальные значения в таблице costs:
в столбце channel:
['FaceBoom', 'MediaTornado', 'RocketSuperAds', 'TipTop', 'YRabbit', 'AdNonSense', 'LeapBob',
'OppleCreativeMedia', 'WahooNetBanner', 'lambdaMediaAds'].

Проверка уникальных значений в столбце channel таблицы costs не выявила противоречивых названий и признаков наличия неявных дубликатов.

1.4.3 Изменим тип данных столбцов в таблице costs

Тип данных столбца: dt - object, это не соответствует значению даты и времени, которому он содержит.
Используем функцию to_datetime() и преобразуем тип данных в нужный формат.

In [11]:

```
costs['dt'] = pd.to_datetime(costs['dt']).dt.date
```

Добавим столбец с номером недели и столбец с номером месяца. Оставим в столбце dt дату без времени. Эти стобцы пригодятся для расчета метрик.

In [12]:

```
costs['dt'] = pd.to_datetime(costs['dt'])

costs['week'] = costs['dt'].dt.isocalendar().week
costs['month'] = pd.DatetimeIndex(costs['dt']).month
```

```
costs['dt'] = pd.to_datetime(costs['dt']).dt.date
costs.head(3)
```

Out[12]:

	dt	channel	costs	week	month
0	2019-05-01	FaceBoom	113.3	18	5
1	2019-05-02	FaceBoom	78.1	18	5
2	2019-05-03	FaceBoom	85.8	18	5

1.4.4 Результат предобработки датафрейма costs.

Предобработка датафрейма costs завершена. Произведена замена типа данных и названий колонок в датафрейме.

Таблица costs содержит 1800 строк, явных дубликатов и пропусков в датафрейме нет. Проверка уникальных значений в столбце channel таблицы costs не выявила противоречивых названий и признаков наличия неявных дубликатов. Содержимое колонок приведено к соответствующему типу данных.

Названия столбцов датафрейма costs после предобработки с кратким описанием содержащихся в них данных:

- dt — дата проведения рекламной кампании,
- channel — идентификатор рекламного источника,
- costs — расходы на эту кампанию.

Предварительно можно утверждать, что в таблице содержится достаточно данных о расходах на рекламу для проведения исследования.

1.5 Итоги раздела

Загружены данные в три датасета: visits, orders, costs. Проведена предобработка данных в ходе которой выявлено, что явных дубликатов и пропусков в таблицах нет. Неявных дубликатов строковых значений не обнаружено. Тип данных в столбцах, содержащих дату и время приведен в соответствие. Названия столбцов изменены к змеиному стилю написания и lowercase.

Перечень таблиц с кратким описанием содержимого и названием столбцов таблицы, что содержит столбец:

visits - хранит лог сервера с информацией о посещениях сайта (записи событий и сообщений, создаваемые программой или системой во время ее работы). Таблица visits содержит 309901 строк. Столбцы:

- user_id — уникальный идентификатор пользователя,
- region — страна пользователя,
- device — тип устройства пользователя,
- channel — идентификатор источника перехода,
- session_start — дата и время начала сессии,
- session_end — дата и время окончания сессии.

orders - хранит информацию о заказах. Датафрейм orders(заказы) содержит 40212 строк. Столбцы:

- user_id — уникальный идентификатор пользователя,
- event_dt — дата и время покупки,
- revenue — сумма заказа.

costs - хранит информацию о заказах. Датафрейм costs(расходы на рекламу) содержит 1800 строк. Столбцы:

- dt — дата проведения рекламной кампании,
- channel — идентификатор рекламного источника,
- costs — расходы на эту кампанию.

Предварительно можно утверждать, что таблицы содержат достаточно данных о логах сервера с информацией о посещениях сайта, о заказах и о расходах на рекламу для проведения дальнейшего исследования.

Теперь можно прописать функции, которые буду использованы для анализа данных.

2 Функции для расчёта и анализа LTV, ROI, удержания и конверсии.

Используем функции, которые помогут нам провести анализ:

2.1 Функции для вычисления значений метрик:

- `get_profiles()` — для создания профилей пользователей,

- `get_retention()` — для подсчёта Retention Rate,
- `get_conversion()` — для подсчёта конверсии,
- `get_ltv()` — для подсчёта LTV

2.1.1 Функция get_profiles() - для создания профилей пользователей

In [13]:

```
# функция для создания пользовательских профилей

def get_profiles(sessions, orders, events, ad_costs, event_names=[]):

    # 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'].values.astype('datetime64[M]')

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

    # добавляем флаги для всех событий из event_names
    for event in event_names:
        if event in events['event_name'].unique():
            profiles[event] = profiles['user_id'].isin(
                events.query('event_name == @event')['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
```

2.1.2 Функция get_retention() — для подсчёта Retention Rate (удержания)

In [14]:

```
# 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']
    ).dt.days

    # 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
```

2.1.3 Функция get_conversion() — для подсчёта конверсии

In [15]:

```
# 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
```

2.1.4 Функция get_ltv() — для подсчёта LTV

In [16]:

```
# 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
    )
```

2.2 Функции для построения графиков:

- `filter_data()` — для сглаживания данных,
- `plot_retention()` — для построения графика Retention Rate,
- `plot_conversion()` — для построения графика конверсии,
- `plot_ltv_roi` — для визуализации LTV и ROI.

2.2.1 Функция filter_data() для сглаживания фрейма

In [17]:

```
# 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
```

2.2.2 Функция `plot_retention()` — для построения графика Retention Rate

In [18]:

```
# 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()
```

2.2.3 Функция `plot_conversion()` — для построения графика конверсии

In [19]:

```
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()
```

2.2.4 Функция `plot_ltv_roi` — для визуализации LTV и ROI

In [20]:

```
# функция для визуализации LTV и 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
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)
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()
```

2.2.5 Итоги раздела.

Функции заданы. Приступим к исследовательскому анализу данных.

3 Исследовательский анализ данных

Составим профили пользователей, определим минимальную и максимальную даты привлечения пользователей. Выясним, из каких стран пользователи приходят в приложение и на какую страну приходится больше всего платящих пользователей, узнаем какими устройствами пользуются клиенты , изучим рекламные источники привлечения .

3.1 Составим профили пользователей.

3.1.1 Профили пользователей

Получим профили пользователей. Для этого вызовем функцию get_profiles(), передав ей данные о посещениях, покупках, событиях и тратах на рекламу. В переменной events укажем None, чтобы функция отработала как положено и не сломалась.

In [21]:

```
events = None #  других событий нет. добавляем, чтобы функция не сломалась.

profiles = get_profiles(visits, orders, events, costs, event_names=[]) # вызовем функцию get_profiles()

display(profiles.sample(3))
```

	user_id	first_ts	channel	device	region	dt	month	payer	acquisition_cost
24624	164309754215	2019-08-30 21:03:11	RocketSuperAds	Android	United States	2019-08-30	2019-08-01	True	0.278571
137414	916340991608	2019-09-09 00:11:33	FaceBoom	iPhone	United States	2019-09-09	2019-09-01	False	1.100000
15252	101465335556	2019-08-23 22:40:58	WahooNetBanner	PC	UK	2019-08-23	2019-08-01	False	0.621429

3.1.2 Получим минимальное и максимальную дату привлечения пользователей.

In [22]:

```
print(f"Доступный интервал привлечения пользователей с \
{profiles['dt'].min()} по {profiles['dt'].max()}.")
```

Доступный интервал привлечения пользователей с 2019-05-01 по 2019-10-27.

```
Минимальные дата для всех событий 01.05.2019, а максимальная дата 27.10.2019.
Представленные данные соответствуют тем, которые были заявлены для проверки.


Текущая дата 1 ноября 2019 года. В бизнес-плане заложено, что пользователи должны
окупаться не позднее чем через две недели после привлечения. Данные можно будет
использовать для анализа, учитывая где надо горизонт планирования в 14 дней (2
недели).
```

3.2 Из каких стран приходят пользователи.

Определим из каких стран пользователи приходят в приложение и на какую страну приходится больше всего платящих пользователей.

3.2.1 Страна пользователя.

Создадим сводную таблицу с группировкой пользователей по странам, выделим всех пользователей, а также отдельно платящих, добавим колонку % платящих пользователей.

In [23]:

```
contry_profiles = (profiles
    .pivot_table(index='region', values='payer', aggfunc=['count', 'sum', 'mean'])
    .rename(columns={'count': 'users', 'sum': 'payer', 'mean': 'ratio'})
    .droplevel(1, axis=1)
    .sort_values(by='users', ascending=False)
    .style.format({'ratio':'{:.2%}'}))
contry_profiles
```

Out[23]:

	users	payer	ratio
region			
United States	100002	6902	6.90%
UK	17575	700	3.98%
France	17450	663	3.80%
Germany	14981	616	4.11%

Визуализируем соотношение пользователей по странам на столбчатой диаграмме.

In [24]:

```
profiles.pivot_table(index='region', values='payer', aggfunc=['count', 'sum'])\
    .rename(columns={'count': 'Пользователи всего', 'sum': 'Пользователи платящие'})\
    .droplevel(1, axis=1)\
    .plot.barh(figsize=(12, 3),
       alpha=0.7,
       ec='black',
       linewidth=1,
       grid=True);
plt.title('Количество пользователей по странам')
plt.xlabel('Пользователи, количество человек')
plt.ylabel('Страна')
plt.show()
```

3.2.2 Выводы по странам пользователя.

Пользователи приходят из четырех регионов - США, Великобритания, Франция, Германия.

США более чем в 5 раз превосходит каждую из других стран по числу пользователей - их 100 тысяч, в то время как у других стран число пользователей не превышает 20 тысяч у каждой страны. Если быть точнее, то 17.5 тысяч у Великобритании и Франции, и 15 тысяч у Германии.

Пользователи из США еще и чаще совершают покупки 6,90%, чем в других странах, где процент совершающих покупки ниже и составляет- 3,8-4,11%.

Процент совершающих покупки 3,8%-6,9% в зависимости от страны. Кажется, что это очень мало. Нужно смотреть другие показатели. Но пока определим какими устройствами пользуются покупателями и на каких устройствах чаще совершаются покупки.

3.3 Какими устройствами пользуются клиенты

Определим какими устройствами пользуются клиенты и какие устройства предпочитают платящие пользователи.

3.3.1 Устройства пользователя

Построим сводную таблицу, отражающую количество пользователей и долю платящих для каждого устройства.

In [25]:

```
device_profiles = (profiles
    .pivot_table(index='device', values='payer', aggfunc=['count', 'sum', 'mean'])
    .rename(columns={'count': 'users', 'sum': 'payer', 'mean': 'ratio'})
    .droplevel(1, axis=1)
    .sort_values(by='users', ascending=False)
    .style.format({'ratio':'{:.2%}'}))
device_profiles
```

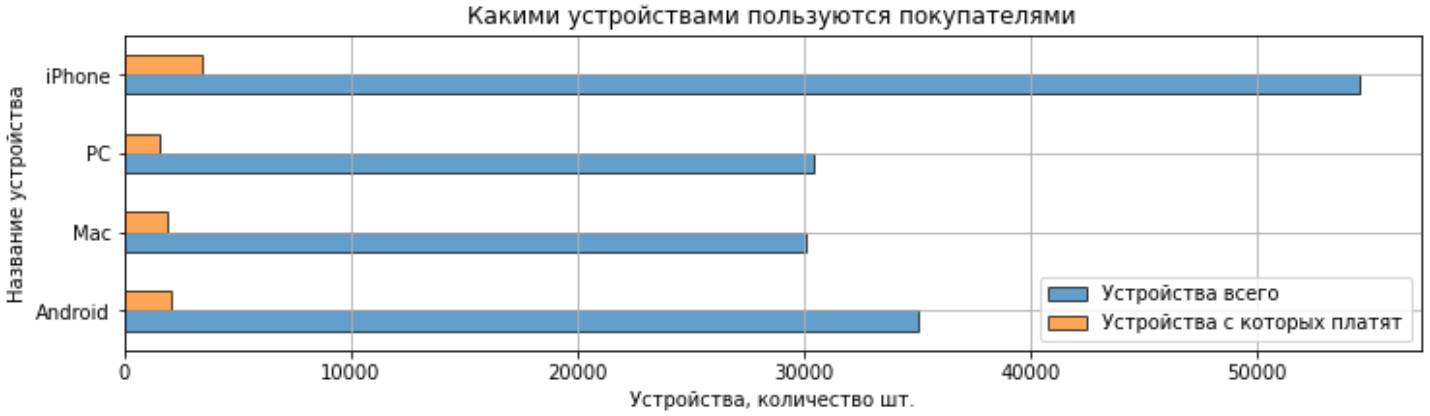
Out[25]:

	users	payer	ratio
device			
iPhone	54479	3382	6.21%
Android	35032	2050	5.85%
PC	30455	1537	5.05%
Mac	30042	1912	6.36%

3.3.2 Визуализируем соотношение устройств которыми пользуются пользователи на столбчатой диаграмме.

In [26]:

```
profiles.pivot_table(index='device', values='payer', aggfunc=['count', 'sum'])\
    .rename(columns={'count': 'Устройства всего', 'sum': 'Устройства с которых
платят'})\
    .droplevel(1, axis=1)\
    .plot.barh(figsize=(12, 3),
     alpha=0.7,
     ec='black',
     linewidth=1,
     grid=True);
plt.title('Какими устройствами пользуются покупателями')
plt.xlabel('Устройства, количество шт.')
plt.ylabel('Название устройства')
plt.show()
```



3.3.3 Выводы по устройствам пользователя.

iPhone стал лидером среди устройств, с которых чаще всего заходят в приложение пользователи стал - 54тыс. пользователей используют именно его.

Android выбрали 35 тыс. пользователей.

PC и Mac используют по 30 тыс. пользователей.

Совершают покупки чаще всего с Mac - 6.36% и iPhone - 6.21%, реже с Android -5.85% и PC - 5.05%.

3.4 Рекламные источники привлечения

Изучим рекламные источники привлечения и определим каналы, из которых пришло больше всего платящих пользователей.

3.4.1 Каналы привлечения

Построим сводную таблицу, отражающую количество пользователей и долю платящих для каждого канала привлечения.

In [27]:

```
channel_profiles = (profiles
    .pivot_table(index='channel', values='payer', aggfunc=['count', 'sum', 'mean'])
    .reset_index()
    .rename(columns={'count': 'users', 'sum': 'payer', 'mean': 'ratio'})
    .droplevel(1, axis=1)
    .sort_values(by='users', ascending=False)
    )
channel_profiles
```

Out[27]:

	channel	users	payer	ratio
10	organic	56439	1160	0.020553
1	FaceBoom	29144	3557	0.122049
6	TipTop	19561	1878	0.096007
4	OppleCreativeMedia	8605	233	0.027077
2	LeapBob	8553	262	0.030633
7	WahooNetBanner	8553	453	0.052964
5	RocketSuperAds	4448	352	0.079137
3	MediaTornado	4364	156	0.035747
8	YRabbit	4312	165	0.038265
0	AdNonSense	3880	440	0.113402
9	lambdaMediaAds	2149	225	0.104700

Видим, что есть ТОП-3 канала привлечения, из которых пришло больше всего пользователей без учета конверсии в платящих. Это

- organic - первое место 56439 человек,
- FaceBoom второе место 29144 человек,
- TipTop - третье место 19561 человек.

Каналы lambdaMediaAds, MediaTornado, MediaTornado, MediaTornado - самые мало эффективные с точки зрения прихода качественных клиентов. Конверсия в платящих меньше 4%.

Канал organic приносит естественных платящих пользователей, их всего 2%, но при этом за счет большого количества, а не % конверсии они занимают третье место. Это неплохо, ведь платы за привлечение таких пользователей нет. Само количество естественных пользователей очень велико, стоит всерьез подумать на предмет конверсии их в платящих.

Сгруппируем теперь таблицу и посмотрим каналы по конверсии в платящих пользователей, это столбец ratio.

In [28]:

```
channel_profiles_payer = (profiles
    .pivot_table(index='channel', values='payer', aggfunc=['count', 'sum', 'mean'])
    .reset_index()
    .rename(columns={'count': 'users1', 'sum': 'payer1', 'mean': 'ratio1'})
    .droplevel(1, axis=1)
    .sort_values(by='ratio1', ascending=False)
    )
display(channel_profiles_payer)
```

	channel	users1	payer1	ratio1
1	FaceBoom	29144	3557	0.122049
0	AdNonSense	3880	440	0.113402
9	lambdaMediaAds	2149	225	0.104700
6	TipTop	19561	1878	0.096007

5	RocketSuperAds	4448	352	0.079137
7	WahooNetBanner	8553	453	0.052964
8	YRabbit	4312	165	0.038265
3	MediaTornado	4364	156	0.035747
2	LeapBob	8553	262	0.030633
4	OppleCreativeMedia	8605	233	0.027077
10	organic	56439	1160	0.020553

Лидеры изменились по соотношению платящих пользователей от всех пользователей, привлеченных каналом.

- FaceBoom - первое место 12,2% платящих пользователей, при этом и самих пользователей пришло много, канал занял 2 место по их числу.
- AdNonSense - второе место 11,34% платящих пользователей, но при этом канал привлекает мало пользователей, он даже не в топ -5 по количеству привлеченных пользователей.
- lambdaMediaAds - третье место 10,47 % платящих пользователей, но привлеченных пользователей еще меньше, чем у AdNonSense
- TipTop - четвертое место 9,6% платящих пользователей, стоит отметить, что по количеству привлеченных пользователей этой канал также в лидерах - третье место, да и количество пользователей много 19561.
- RocketSuperAds - пятое место 7,9 % , но опять же, количество пользователей небольшое.

3.4.2 Топ каналов привлечения.

Соединим из таблиц с каналами пользователей и каналами платящих пользователей, выделив из каждой по топ - 5. Посмотрим точки пересечений и найдем наиболее успешные каналы привлечения.

In [29]:

```
channel_profiles_top = pd.merge(channel_profiles.head(5),\
                                channel_profiles_payer.head(5),\
                                left_index=True, right_index=True, how='outer')
channel_profiles_top
```

Out[29]:

	channel_x	users	payer	ratio	channel_y	users1	payer1	ratio1
0	NaN	NaN	NaN	NaN	AdNonSense	3880.0	440.0	0.113402
1	FaceBoom	29144.0	3557.0	0.122049	FaceBoom	29144.0	3557.0	0.122049
2	LeapBob	8553.0	262.0	0.030633	NaN	NaN	NaN	NaN
4	OppleCreativeMedia	8605.0	233.0	0.027077	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	RocketSuperAds	4448.0	352.0	0.079137
6	TipTop	19561.0	1878.0	0.096007	TipTop	19561.0	1878.0	0.096007
9	NaN	NaN	NaN	NaN	lambdaMediaAds	2149.0	225.0	0.104700
10	organic	56439.0	1160.0	0.020553	NaN	NaN	NaN	NaN

Точки пересечений в ТОП-5 есть только у FaceBoom и TipTop.Остальные каналы хороши только в одном - или конверсией в платящего пользователя, но малым количеством самих привлеченных пользователй или большим количеством привлеченных пользователей, но низкой конверсией в платящих.

3.4.3 Визуализируем каналы привлечения пользователей.

Построим три столбчатые диаграммы, отражающие каналы привлечения пользователей: первая диаграмма это топ 5 каналов, которые привлекли больше всего пользователей всего, вторую диаграмму топ 5 каналов по удельному весу платящих пользователей и третью топ 5 каналов по количеству платящих пользователей в количестве человек.количеству. Для начала сформируем три сводные талицы и затем визуализируем их на столбчатых диаграммах.

In [30]:

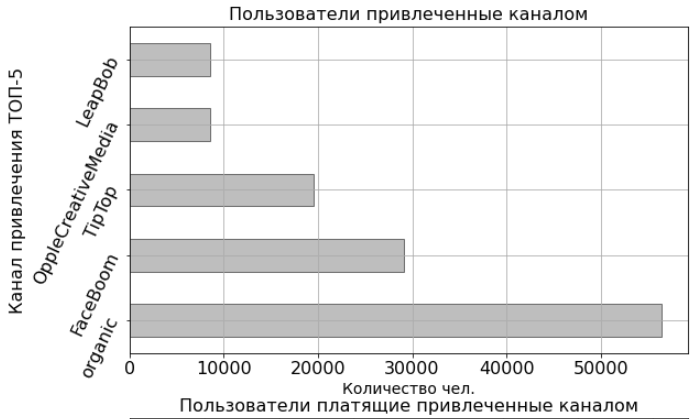
```
top_profiles = profiles.pivot_table(index='channel', values='payer',
aggfunc=['count', 'sum'])\
    .rename(columns={'count': 'Пользователи', 'sum': 'Платящие пользователи'})\
    .droplevel(1, axis=1)\
    .sort_values(by='Пользователи', ascending=False).head(5)\
```

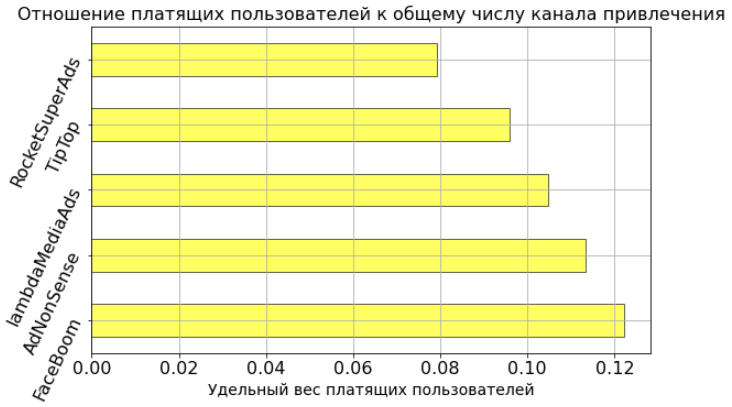
```
top_profiles_payer = profiles.pivot_table(index='channel', values='payer',
aggfunc=['count', 'sum'])\
    .rename(columns={'count': 'Пользователи', 'sum': 'Платящие пользователи'})\
    .droplevel(1, axis=1)\
    .sort_values(by='Платящие пользователи', ascending=False).head(5)

top_profiles_payer_ratio = profiles.pivot_table(index='channel', values='payer',
aggfunc=['count', 'sum', 'mean'])\
    .rename(columns={'count': 'Пользователи', 'sum': 'Платящие пользователи', 'mean':
'Платящие пользователи доля'})\
    .droplevel(1, axis=1)\
    .sort_values(by='Платящие пользователи доля', ascending=False).head(5)
```

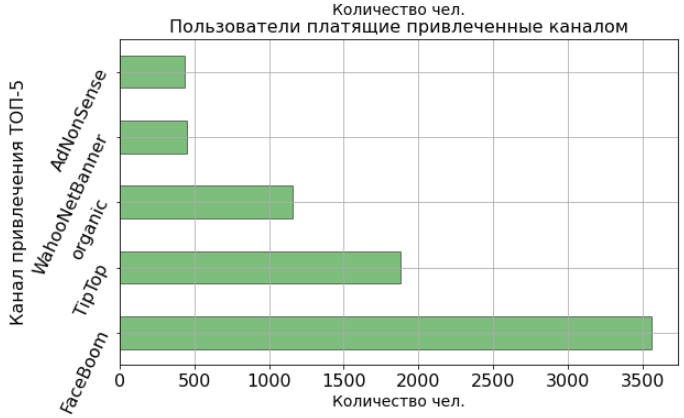
In [31]:

```
plt.figure(figsize=(20, 12))

top_profiles['Пользователи'].plot.barh(
       alpha=0.5,
       ec='black',
       linewidth=1,
       color = 'grey',
       grid=True,
       ax=plt.subplot(2,2,1));
plt.title('Пользователи привлеченные каналом', size=16)
plt.xlabel('Количество чел.', size=14)
plt.xticks(size=16)
plt.ylabel('Канал привлечения ТОП-5', size=16)
plt.yticks(rotation=65, fontsize = 16)
#ax.text(fontsize = 16)  # сделаем выравнивание по центру
   #  вторая диаграмма
top_profiles_payer_ratio['Платящие пользователи доля'].plot.barh(
       alpha=0.6,
       ec='black',
       linewidth=1,
       color = 'yellow',
       grid=True,
       ax=plt.subplot(2,2,2));
plt.title('Отношение платящих пользователей к общему числу канала привлечения',
size=16)
plt.xlabel('Удельный вес платящих пользователей', size=14)
plt.xticks(size=16)
plt.ylabel(' ', size=16)
plt.yticks(rotation=65, fontsize = 16);
   #  третья диаграмма
top_profiles_payer['Платящие пользователи'].plot.barh(
       alpha=0.5,
       ec='black',
       linewidth=1,
       color = 'green',
       grid=True,
       ax=plt.subplot(2,2,3));
plt.title('Пользователи платящие привлеченные каналом', size=16)
plt.xlabel('Количество чел.', size=14)
plt.xticks(size=16)
```

```
plt.ylabel('Канал привлечения ТОП-5', size=16)
plt.yticks(rotation=65, fontsize = 16);
```







3.4.4 Выводы по каналам привлечения.

Среди рекламных каналов привлечения с наибольшим количеством пользователей и хорошей конверсией в платящих пользователей выделяется FaceBoom, TipTop. Канал organic заслуживает внимания - он приносит самое большое число посетителей, но плохо конвертится в платящих пользователей. Это точка роста, нужно провести работу над увеличением платящих пользователей, ведь за стоимость привлечения платить не нужно.

AdNonSense и lambdaMediaAds приносят качественных пользователей, но охват маленький. Можно постараться увеличить охват пользователей и посмотреть на динамику их конверсии в платящих, если она сохранится, то это тоже точка роста. От органики так мало платящих, так как зачастую закупочные каналы оптимизируются с целью приведения максимально платящей аудитории, а органические пользователи случаются разные

3.5 Вывод по разделу

Составили профили пользователей. Минимальные дата для всех событий 01.05.2019, а максимальная дата 27.10.2019. Считаем, что самая поздняя дата это 27.10.2019, она же станет моментом анализа для анализа данных, которую будем использовать в дальнейшем.

Пользователи приходят в приложение из этих стран:

- США,
- Великобритания,
- Франция,
- Германия.

США более чем в 5 раз превосходит каждую из других стран по числу пользователей - их 100 тысяч, 17.5 тысяч пользователей у Великобритании и Франции, и 15 тысяч у Германии.

Пользователи из США еще и чаще совершают покупки 6,90%, чем в других странах, где процент совершающих покупки ниже и составляет- 3,8-4,11%.

Процент совершающих покупки 3,8%-6,9% в зависимости от страны. Кажется, что это очень мало. Нужно смотреть другие показатели. Определили какими устройствами пользуются покупателями и на каких устройствах чаще совершаются покупки:

- iPhone стал лидером среди устройств, с которых чаще всего заходят в приложение пользователи - 54тыс. пользователей используют именно его.
    - Android выбрали 35 тыс. пользователей.
    - PC и Mac используют по 30 тыс. пользователей.

Совершают покупки чаще всего

- с Mac - 6.36% и iPhone - 6.21%,
- реже с Android -5.85% и PC - 5.05%.

Среди рекламных каналов привлечения использованы 10 каналов и organic -естественный трафик пользователей, с которым каналов привлечения -11 штук.
Наибольшим количеством пользователей и хорошей конверсией в платящих пользователей выделяется FaceBoom, TipTop.
Канал organic заслуживает внимания - он приносит самое большое число посетителей, но плохо конвертится в платящих пользователей. Это не удивительно, ведь это естественный трафик, с которым приходит большое количество разных пользователей, которые не обязательно заинтересованы в нашем продукте. Все же это возможная точка роста, можно провести работу над увеличением платящих пользователей, ведь за стоимость привлечения платить не нужно.
AdNonSense и lambdaMediaAds приносят качественных пользователей, но охват маленький. Можно постараться увеличить охват пользователей и посмотреть на динамику их конверсии в платящих, если она сохранится, то это тоже точка роста.

4 Маркетинг

4.1 Общая сумма расходов на маркетинг.

Для удобства выведем несколько строк с таблицей profiles. Траты на маркетинг указаны в столбце acquisition_cost, просуммируем их. Валюта расходов не указана, поэтому будем считать в условных единицах, сокращенно у.е.

In [32]:

```
profiles.sample(n=2, random_state=2)
```

Out[32]:

	user_id	first_ts	channel	device	region	dt	month	payer	acquisition_cost
87967	585162981744	2019-09-18 09:03:11	LeapBob	Android	Germany	2019-09-18	2019-09-01	False	0.21
10548	70321669668	2019-09-29 16:23:45	TipTop	iPhone	United States	2019-09-29	2019-09-01	False	3.50

In [33]:

```
print(f"Общая сумма расходов на маркетинг:{round(profiles['acquisition_cost'].sum(), 2)} у.е.")
```

Общая сумма расходов на маркетинг:105497.3 у.е.

4.2 Распределение трат по рекламным источникам

4.2.1 Выясним сколько денег потратили на каждый источник.

Столбец acquisition_cost в таблице показывает расходы на каждого привлеченного пользователя. У пользователей пришедших по естественному каналу (organic) этот показатель равен 0, так как затрат на привлечение таких пользователей нет.
Постоим сводную таблицу? в которой названия строк будут даты привлечения пользователей - укрупним их по первому дню месяца привлечения, названия стобцов - каналы привлечения, а значения средний CAC.

In [34]:

```
profiles_total_costs = (profiles
                        .groupby('channel')
                        .agg({'acquisition_cost':'sum'})
                        .rename(columns={'acquisition_cost':'total_cost'})
                        .sort_values(by='total_cost', ascending=False)
                       )
profiles_total_costs
```

Out[34]:

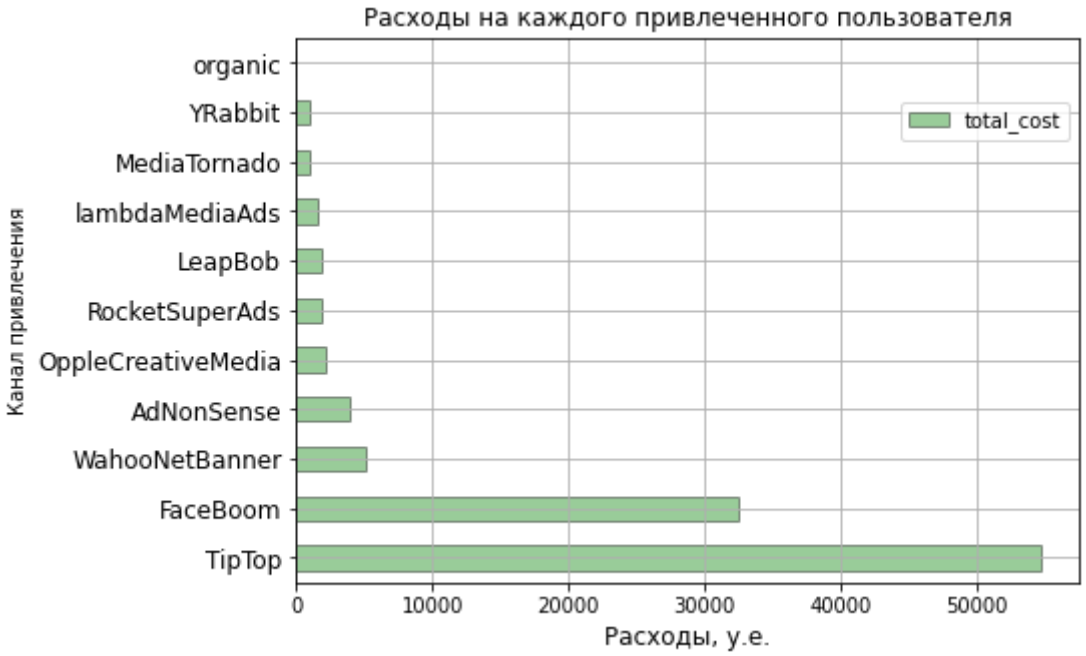
	total_cost
channel	
TipTop	54751.30
FaceBoom	32445.60
WahooNetBanner	5151.00

AdNonSense	3911.25
OppleCreativeMedia	2151.25
RocketSuperAds	1833.00
LeapBob	1797.60
lambdaMediaAds	1557.60
MediaTornado	954.48
YRabbit	944.22
organic	0.00

4.2.2 Визуализируем расходы на каждый канал привлечения пользователей столбчатой диаграммой.

In [35]:

```
profiles_total_costs.plot.barh(
   figsize=(7,5), alpha=0.4, color='green', ec='black', linewidth=1, grid=True);
plt.legend(bbox_to_anchor=(1, 0.9))
plt.title('Расходы на каждого привлеченного пользователя', size=12)
plt.xlabel('Расходы, у.е.', size=12) #, color='green')
plt.yticks(fontsize = 12)
plt.ylabel('Канал привлечения')
plt.show()
```



4.2.3 Посмотрим на каналы, рекламный бюджет которых больше всех, выведем топ-3 рекламных канала на графике.

In [36]:

```
profiles_total_costs.head(3).plot.bar(
   figsize=(6,4), alpha=0.4, color='yellow', ec='black', linewidth=2);
plt.legend(bbox_to_anchor=(1, 0.9))
plt.title('Топ-3 канала по рекламным расходам', size=12)
plt.ylabel('Расходы, у.е.', size=12) #, color='green')
plt.yticks(fontsize = 12)
plt.xticks(fontsize = 12, rotation=360)
plt.xlabel('Канал привлечения',  size=12)
plt.show()
```

4.2.4 Выводы подраздела

Ярко выделяются два лидера по растратам бюджета на привлечения пользователей. Это FaceBoom 54.7 тыс. у.е. и TipTop 32.4 тыс. у.е. Третье место с большим отрывом от двух лидеров, но с не таким сильным отрывом от всех остальных рекламных каналов занимает WahooNetBanner 5.2 тыс. у.е.

Безусловный лидер по отсутствию расходов - канал organic с естественным трафиком пользователей, стоимость привлечения для них логично лежит на отметке 0.

У MediaTornado 0,95 тыс у.е и у YRabbit 0,94 тыс у.е. рекламного бюджета - это каналы, которые меньше всего потратили на рекламу.

Смотреть на рекламный бюджет в отрыве от других показателей нельзя.

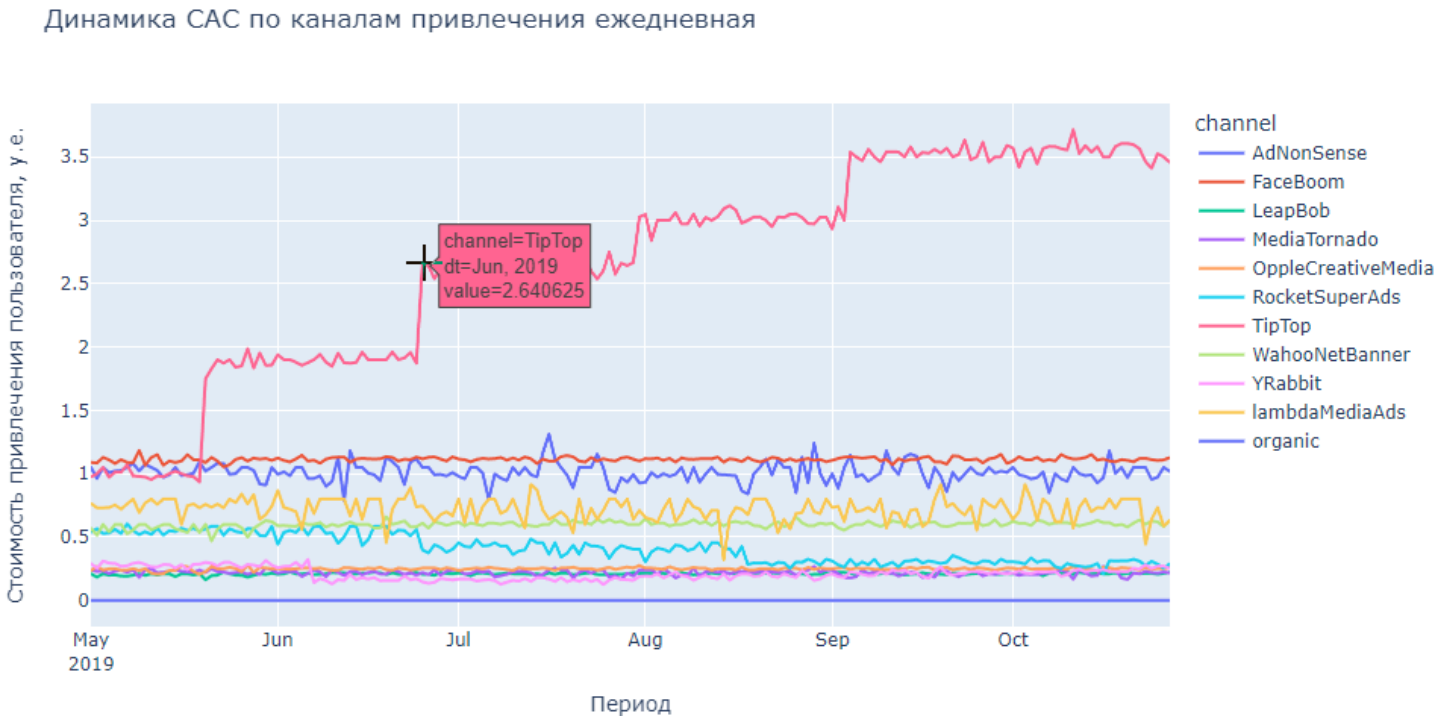
Для начала посмотрим на эти расходы на графиках в динамике.

4.3 Динамика изменения расходов

4.3.1 Построим график с визуализацией динамики изменения расходов.

In [37]:

```
 fig = px.line(profiles.pivot_table(
   index='dt', columns='channel', values='acquisition_cost', aggfunc='mean'
),
             title='Динамика CAC по каналам привлечения ежедневная',
             color="channel")
fig.update_xaxes(
   tickformat="%b\n%Y")
fig.update_layout(
   xaxis={
       'title':'Период',
       },
   yaxis={'title':'Стоимость привлечения пользователя, у.е.'})
fig.show()
```



Сгладим шумность графика и посмотрим на еженедельную и ежемесячную динамику расходов.

4.3.2 Создадим сводную таблицу с расходами за неделю.

На основе таблицы costs с расходами сформируем сводную таблицу с индексом неделя. По этой таблице будем формировать наш график

In [38]:

```
costs.pivot_table(
    index='week', columns='channel', values='costs', aggfunc='sum'
)
```

Out[38]:

channel	AdNonSense	FaceBoom	LeapBob	MediaTornado	OppleCreativeMedia	RocketSuperAds	TipTop	WahooNetBanner	YRabbit	lambdaMediaAds
week										
18	211.05	535.7	16.80	38.64	24.00	99.450	347.0	49.2	52.20	81.6
19	273.00	750.2	31.71	61.68	34.50	139.230	470.0	92.4	75.90	103.2
20	265.65	755.7	26.67	59.04	27.50	138.060	454.0	90.0	69.30	90.4
21	266.70	722.7	26.67	50.16	34.25	131.040	801.8	79.2	65.40	119.2
22	202.65	1208.9	65.52	36.96	81.75	97.695	1421.2	179.4	49.80	83.2
23	102.90	1081.3	66.99	27.60	79.50	76.050	1223.6	196.8	25.80	35.2
24	96.60	1042.8	52.29	29.76	71.50	63.180	1121.0	166.2	17.46	45.6
25	141.75	1280.4	68.46	32.16	79.75	80.730	1474.4	201.6	22.14	40.0
26	148.05	1647.8	99.75	40.56	107.00	78.260	2343.6	285.0	27.90	59.2
27	130.20	1536.7	87.36	43.44	102.50	75.075	2340.0	256.8	28.26	60.0
28	106.05	1124.2	62.58	28.80	81.00	55.965	1820.0	174.0	19.44	44.8
29	97.65	975.7	61.11	26.40	68.00	41.860	1552.2	159.6	15.66	41.6
30	118.65	1130.8	67.20	25.20	80.75	54.600	1713.4	182.4	22.86	49.6
31	141.75	1419.0	83.58	35.04	90.75	61.880	2493.6	204.6	27.33	52.0
32	116.55	1290.3	87.57	29.04	104.00	69.615	2448.0	231.6	28.56	41.6
33	117.60	1456.4	77.49	40.80	98.50	62.010	2538.0	238.2	28.56	46.4
34	142.80	1411.3	90.51	35.28	103.75	40.300	2514.0	244.8	30.03	57.6
35	133.35	1445.4	77.91	33.84	92.75	51.350	2583.0	235.8	24.36	54.4
36	100.80	1151.7	66.99	30.24	72.75	40.625	2563.0	174.6	27.93	52.0
37	100.80	1148.4	63.84	25.92	78.00	40.950	2506.0	192.6	29.76	41.6
38	139.65	1496.0	81.27	30.24	105.75	56.225	3241.0	226.2	39.60	57.6

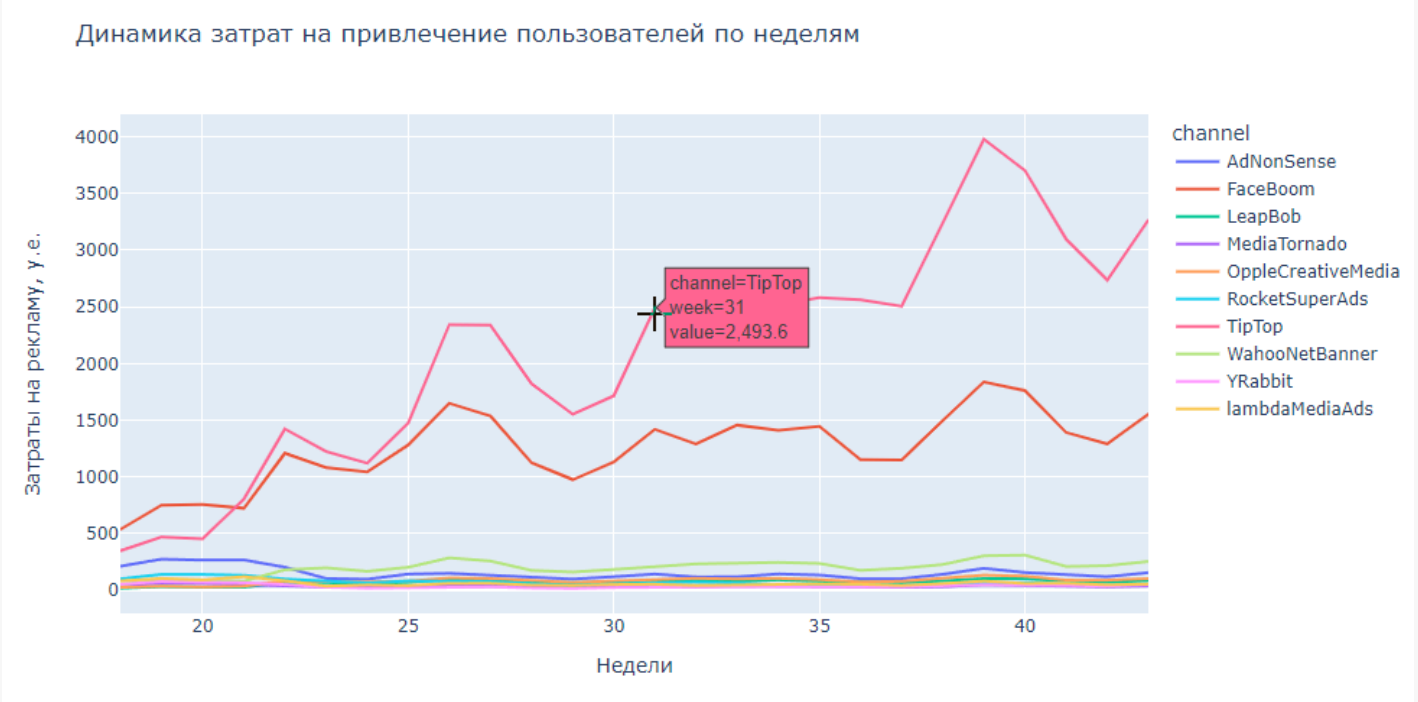
39	192.15	1837.0	101.22	48.24	132.25	63.700	3979.5	303.0	47.52	80.0
40	155.40	1761.1	101.01	42.00	123.00	66.300	3703.0	309.0	45.36	59.2
41	136.50	1392.6	78.96	36.96	88.25	51.675	3097.5	208.8	38.31	60.8
42	118.65	1291.4	69.72	29.04	88.00	46.475	2737.0	216.0	36.45	47.2
43	154.35	1552.1	84.42	37.44	101.50	50.700	3265.5	253.2	48.33	53.6

4.3.3 График динамики расходов по каналам привлечения, по неделям

На основе сводной таблицы построим график с визуализацией динамики изменения расходов во времени по неделям по каждому источнику.

In [39]:

```
fig = px.line(costs.pivot_table(
    index='week', columns='channel', values='costs', aggfunc='sum'
),
              title='Динамика затрат на привлечение пользователей по неделям',
              color="channel"
             )
fig.update_layout(
    xaxis={'title':'Недели'},
    yaxis={'title':'Затраты на рекламу, у.е.'}
    )
fig.show()
```



На недельном графике видно, что начальный уровень расходов по всем каналам, кроме двух сохранялся на низком уровне, с незначительными колебаниями. Диапазон расходов для каналов примерно от 20 до 300 у.е. за неделю, выше для для WahooNetBanner и AdNonSense и ниже для остальных каналов.

Канал FaceBoom в первые три недели расход не превышал 500 у.е. в неделю, а затем расходы начали расти скачкообразно - высокий рост, потом небольшое снижение, но с тенденцией общей на увеличение. Так повторяется на всем протяжении времени. Уровень расходов колеблется от 1000 у.е. до 1800 у.е. за неделю.

Похожая картина у канала с TipTop, с тем отличием, что начальный уровень расходов на этот рекламный канал стартанул с 500 у.е. и далее рос еще большими скачками до 2500 у.е. в неделю, а рост с 37 недели с 2000 у.е достиг в 39 неделю 4000 у.е., затем снизился до 2700. Видим большой рекламный бюджет на привлечение пользователей у канала TipTop.

4.3.4 График динамики расходов по каналам привлечения, по месяцам

На основе сводной таблицы costs создадим сводную таблицу с индексом месяц, на основе нее будем формировать график с визуализацией динамики изменения расходов во времени по месяцам по каждому источнику.

In [40]:

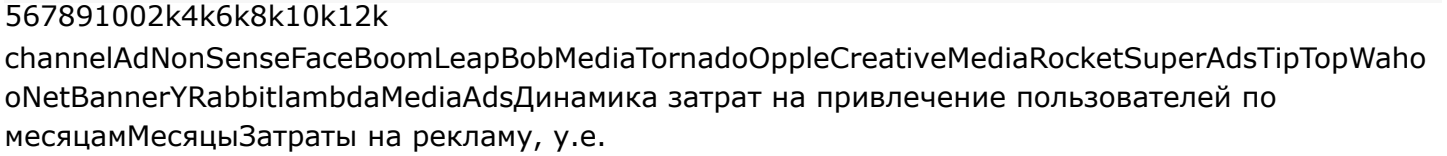
```
costs.pivot_table(
   index='month', columns='channel', values='costs', aggfunc='sum'
)
```

Out[40]:

channel	AdNon Sense	FaceBo om	LeapB ob	MediaTorn ado	OppleCreative Media	RocketSuper Ads	TipTop	WahooNet Banner	YRabbit	lambdaMe diaAds
month										
5	1169.70	3524.4	140.28	238.56	169.75	577.980	2981.0	418.8	299.70	458.4
6	538.65	5501.1	314.58	138.00	370.00	325.715	6675.6	921.0	106.20	199.2
7	504.00	5294.3	313.53	138.48	366.50	252.070	8410.2	851.4	97.38	219.2
8	579.60	6274.4	369.81	154.56	439.25	253.110	11202.0	1040.4	124.74	220.0
9	581.70	6114.9	343.98	144.72	427.75	218.400	13232.5	977.4	152.79	247.2
10	537.60	5736.5	315.42	140.16	378.00	205.725	12250.0	942.0	163.41	213.6

In [41]:

```
fig = px.line(costs.pivot_table(
   index='month', columns='channel', values='costs', aggfunc='sum'
),
              title='Динамика затрат на привлечение пользователей по месяцам',
              color="channel"
             )
fig.update_layout(
   xaxis={'title':'Месяцы'},
   yaxis={'title':'Затраты на рекламу, у.е.'}
   )
fig.show()
```



На графике за счет того,что расходы рассматриваем по месяцам, шумности меньше и больше видна тенденция на увеличение рекламных расходов у двух каналов FaceBoom и TipTop. На месячном графике видно, что начальный уровень расходов по всем каналам, кроме двух сохранялся на примерно одинаковом уровне, с незначительными колебаниями не превышал в среднем 1000 у.е. в месяц для канала WahooNetBanner и AdNonSense и ниже для остальных каналов 100 - 400 у.е. в месяц.

Канал FaceBoom стартанул с 3500 у.е. расходов в первый месяц май, а затем расходы в июне выросли до 5500 у.е, в августе до 6200 у.е и далее наблюдается небольшое снижение. В целом, можно сказать, что после повышения расходов в первый месяц они остаются на примерно одинаково высоком уровне весь период.

Канал TipTop стартанул с 3000 у.е расходов и уже за первый месяц более чем в два раза вырос рекламный бюджет - до 6700 у.е. в июне, затем рост продолжился наа 20-35 % в месяц и бюджет расходов достиг 13200 у.е. в сентябре, с небольшим уменьшением в октябре на 12200. Но учитывая, что расходы за октябрь даны не за полный месяц, можно предположить, что это снижение вызвано отсутствием данных да почти неделю. С этим учетом можно предположить, что снижения нет и рост продолжился.

Посчитаем, оправдан ли такой высокий бюджет расходов, найдем CAC.

4.3.5 Итоги подраздела анализа динамики расходов

Графики показывают, что устойчивый уровень CAC на протяжении всего периода у следующих каналов:

- 'FaceBoom',
- 'MediaTornado',
- 'YRabbit',
- 'LeapBob',

- 'OppleCreativeMedia',
- 'WahooNetBanner',
- 'organic' с нулевым CAC.

Краткосрочное колебание CAC то выше, то ниже на протяжении всего периода показывают эти каналы:

- 'lambdaMediaAds'
- 'AdNonSense'

Один канал показал колебания уровня CAC, затем снизил CAC с середины августа почти вдвое и приравнялся ко всем остальным каналам, которые изначально показывали устойчивый уровень CAC и также стал устойчивым. Это:

- 'RocketSuperAds'
- TipTop среди всех каналов привлечения явно выделяется. Стоимость привлечения на одного пользователя у TipTop увеличивается примерно раз в 4 недели до сентября и только в сентябре и октябре такого резкого роста нет и он находится на уровне примерно 3,5 у.е. за одного пользователя против 1 у.е. изначально.

Чем же вызвано такое увеличение стоимости? Политикой канала, сменой модели оплаты рекламы?

У всех каналов стоимость привлечения одного клиента осталась примерно на том же уровне, с которого стартовала, один TipTop удивил.

Посмотрим на стоимость привлечения одного пользователя, рассчитаем метрику CAC.

4.4 Метрика CAC

4.4.1 Момент анализа

Ранее мы находили самую позднюю дата запишем ее в переменной observation_date и будем использовать в дальнейшем как момент анализа.

In [42]:

```
observation_date = profiles['dt'].max()
```

4.4.2 Найдем сколько в среднем стоило привлечение одного пользователя (CAC) из каждого источника.

Отфильтруем пользовательские профили методом query(), возьмем лишь те, что участвуют в расчете LTV. Результат сохраним в переменной ltv_profiles.

In [43]:

```
#  отсекаем профили старше даты момента анализа
ltv_profiles = profiles.query('dt <= @observation_date')
```

4.4.3 Рассчитаем средний CAC по каждому каналу привлечения.

Для этого сгруппируем отфильтрованные профили и применим функцию mean() к значениям столбца acquisition_cost. Сохраним результат в переменную cac, изменив название столбца acquisition_cost на cac.

In [44]:

```
cac = (
   ltv_profiles.groupby('channel')
   .agg({'acquisition_cost':'mean'})
   .rename(columns={'acquisition_cost':'cac'})
   .sort_values(by='cac', ascending=False)
)
cac
```

Out[44]:

	cac
channel	
TipTop	2.799003
FaceBoom	1.113286
AdNonSense	1.008054
lambdaMediaAds	0.724802
WahooNetBanner	0.602245
RocketSuperAds	0.412095
OppleCreativeMedia	0.250000
YRabbit	0.218975

MediaTornado	0.218717
LeapBob	0.210172
organic	0.000000

4.4.4 Выводы подраздела

канал TipTop самый дорогой - стоимость привлечения одного пользователя 2.8 у.е., второй по стоимости FaceBoom - 1,11 у.е.
MediaTornado, LeapBob и YRabbit - самые дешевые каналы привлечения пользователей - 0,21-0,22 у.е.

4.5 Итоги раздела

Определили лидеров по величине рекламного бюджета. Это FaceBoom 54.7 тыс. у.е. и TipTop 32.4 тыс. у.е. Третье место с большим отрывом от двух лидеров, но с не таким сильным отрывом от всех остальных рекламных каналов занимает WahooNetBanner 5.2 тыс. у.е.

Графики подтверждают, что у TipTop рекламный бюджет не только вырос и достиг величины в 54,8 тыс. у.е, но и стоимость привлечения одного пользователя выросла в 3,5 раза с 1 у.е. до 3,5 у.е. Всего канал привлек 19,5тыс человек.

При этом лидер рекламного бюджета FaceBoom показал стабильный CAC на протяжении всего периода в районе 1.1 -1.2 у.е. за пользователя, а общий рекламный бюджет составил 32,4 тыс у.е., канал привлек 29 тыс. человек.

Остальные компании не удивили какими-то отклонениями, кроме RocketSuperAds, который показал снижение CAC после колебаний.

Нужно смотреть, окупается ли реклама.

5 Оценка окупаемости рекламы

Оценим окупаемость рекламы используя графики LTV, ROI и CAC. Текущая дата 1 ноября 2019 года. Бизнес-планом заложено, что пользователи должны окупаться не позднее чем через две недели после привлечения.

Исключим пользователей пришедших из естественного трафика из анализа, так как расходов на них нет.

5.1 Исключим канал 'organic' из анализа

Отфильтруем профили и удалим пользователей из канала 'organic', запишем в таблицу profiles_CAC.

In [45]:

```
profiles_CAC = profiles.query('channel != "organic"').copy()
print(f"Каналы рекламы после исключения канала
organic:{list(profiles_CAC['channel'].unique())}")
```

Каналы рекламы после исключения канала organic:['FaceBoom', 'AdNonSense', 'YRabbit', 'MediaTornado', 'RocketSuperAds', 'LeapBob', 'TipTop', 'WahooNetBanner', 'OppleCreativeMedia', 'lambdaMediaAds']

5.2 Горизонт анализа

Ранее мы установили момент анализа 27 октября 2019г. в переменной observation_date. Правилами компании принято считать, что окупаемость затрат на рекламу должна наступать не позднее, чем через 2 недели после привлечения пользователей, Исходя из этих данных установим горизонт анализа в переменной horizon_days 14 дней.

In [46]:

```
horizon_days = 14  # горизонт анализа
```

5.3 Конверсия пользователей и динамика её изменения.

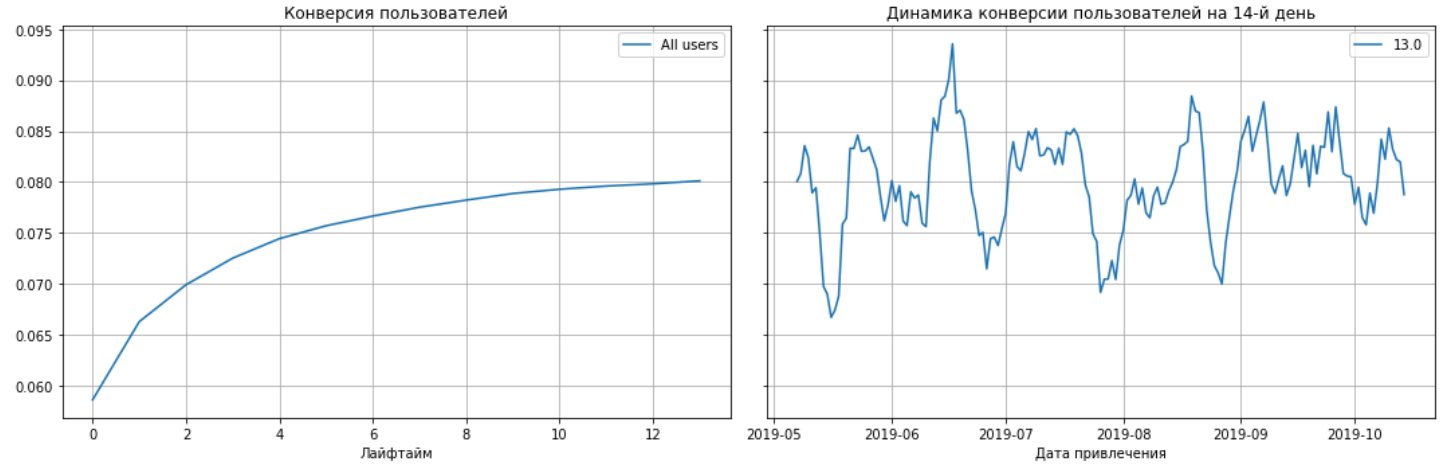
Для начала проверим конверсию пользователей и динамику её изменения. Построим и изучим график конверсии. Используем функцию get_conversion() — для подсчёта конверсии и вызовем функцию plot_conversion для графиков.

5.3.1 Конверсия пользователей без детализации

In [47]:

```
conversion_raw, convercion_grouped, conversion_history = get_conversion(
   profiles_CAC,
   orders,
   observation_date,
   horizon_days,
```

```
    dimensions=[],
    ignore_horizon=False)
plot_conversion(convercion_grouped, conversion_history, horizon_days)
```

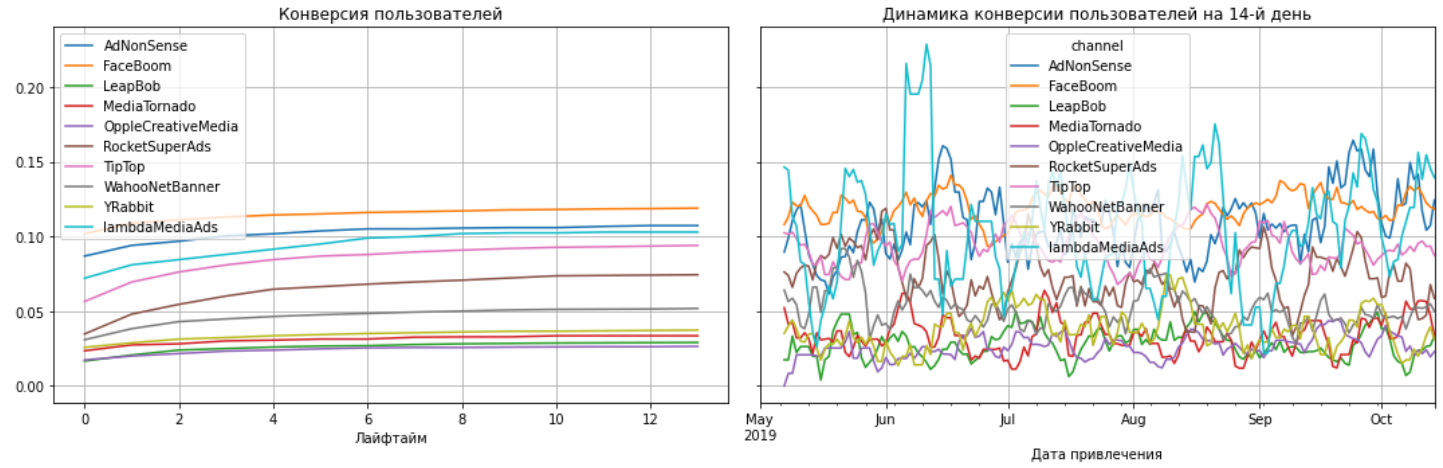


Конверсия в платящие пользователи постепенно растет и на на 14 день достигает 8%

5.3.2 Конверсия пользователей по каналам привлечения

In [48]:

```
conversion_raw, convercion_grouped, conversion_history = get_conversion(
    profiles_CAC,
    orders,
    observation_date,
    horizon_days,
    dimensions=['channel'],
    ignore_horizon=False)
plot_conversion(convercion_grouped, conversion_history, horizon_days)
```



На графике нет ни одного канала привлечения пользователей, который бы показал хороший рост конверсии в платящих пользователей.

Кривая конверсии должна приближаться к единице, а на нашем графике этого роста после первых 4х недель не наблюдается. Закон роста не напоминает эталонный.

Кривая стагнирована в основном.

Конверсия есть на уровне 9-12% у четырех каналов на 14-й день жизни, хоть и без роста, а именно: 'FaceBoom' 12% , второе место у 'AdNonSense' - 11%, 'lambdaMediaAds' -10 %, 'TipTop' 9%.

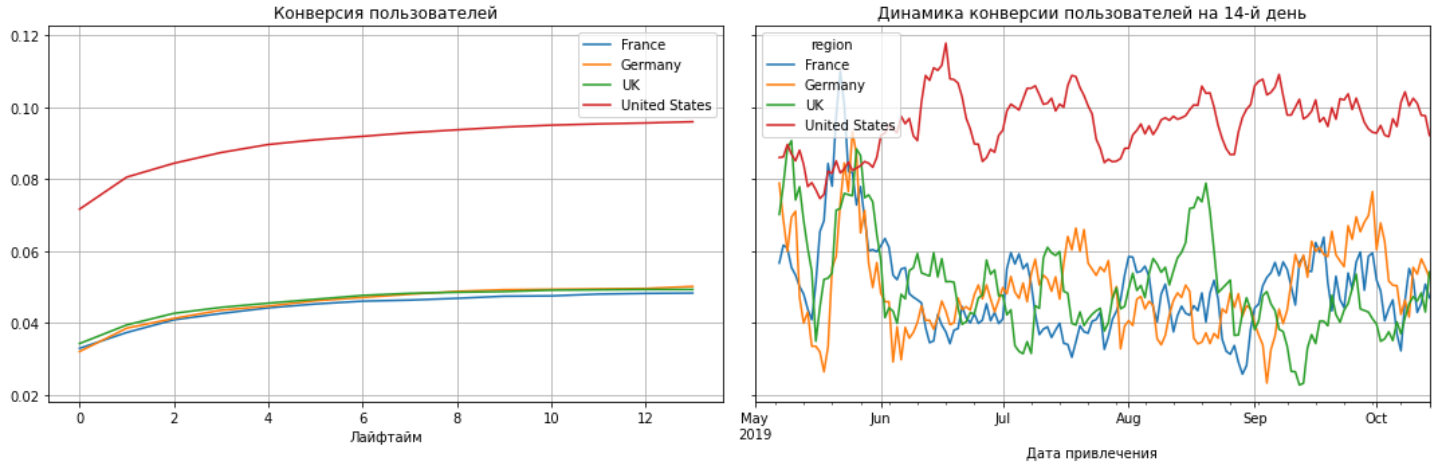
У остальных каналов ниже:'RocketSuperAds' - 7%, 'WahooNetBanner' - 5%, у 'YRabbit', 'MediaTornado', 'LeapBob', ' 'OppleCreativeMedia' примерно на уровне 3%. Это мало.

5.3.3 Конверсия пользователей по странам

In [49]:

```
conversion_raw, convercion_grouped, conversion_history = get_conversion(
    profiles_CAC,
```

```
    orders,
    observation_date,
    horizon_days,
    dimensions=['region'],
    ignore_horizon=False)
plot_conversion(convercion_grouped, conversion_history, horizon_days)
```



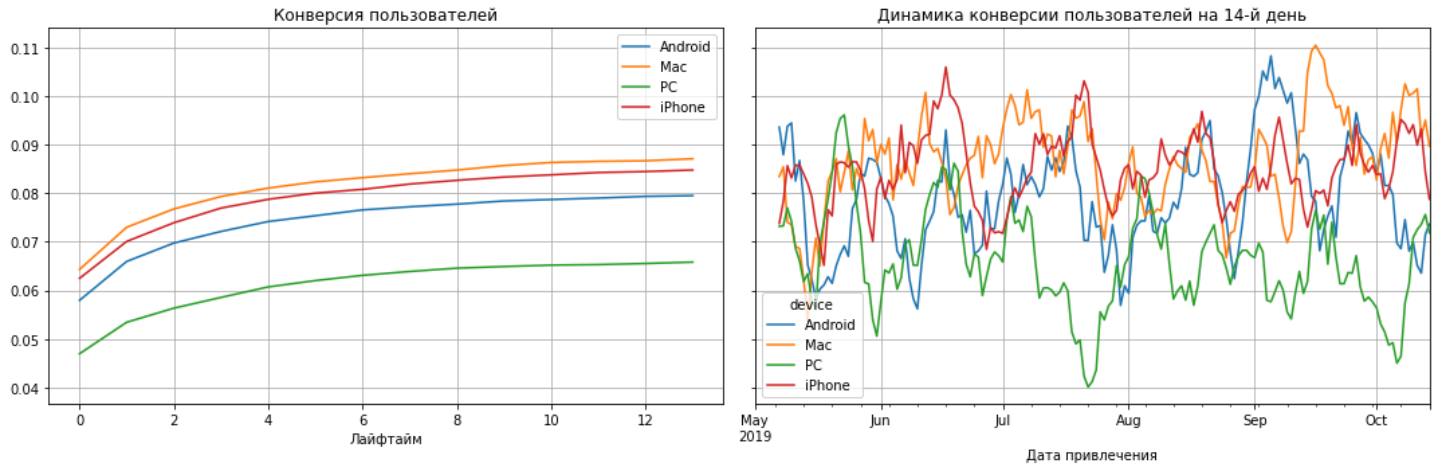
Пользователи из США хорошо конвертируются в платящих, на 14 день это 9 %, когда как у других стран конверсия в платящих пользователей только 5%. В начале мая и в конце мая остальные страны практически синхронно показали сопоставимый с США уровень конверсии резким ростом и резким падением. И после этого ни разу не смогли достичь такого же уровня, как у США.

Посмотрим зависит ли конверсия в платящего пользователя от устройст, которые используют для входа в приложение.

5.3.4 Конверсия пользователей по устройствам

In [50]:

```
conversion_raw, convercion_grouped, conversion_history = get_conversion(
    profiles_CAC,
    orders,
    observation_date,
    horizon_days,
    dimensions=['device'],
    ignore_horizon=False)
plot_conversion(convercion_grouped, conversion_history, horizon_days)
```



Видим, что лучше всего платят владельцы iPhone и Mac - уровень конверсии около 8,5 %, причем у iPhone немного выше. Android также показал 8 % конверсии в платящего пользователя на 14-й день. Хууже всех конвертится владелец PC.

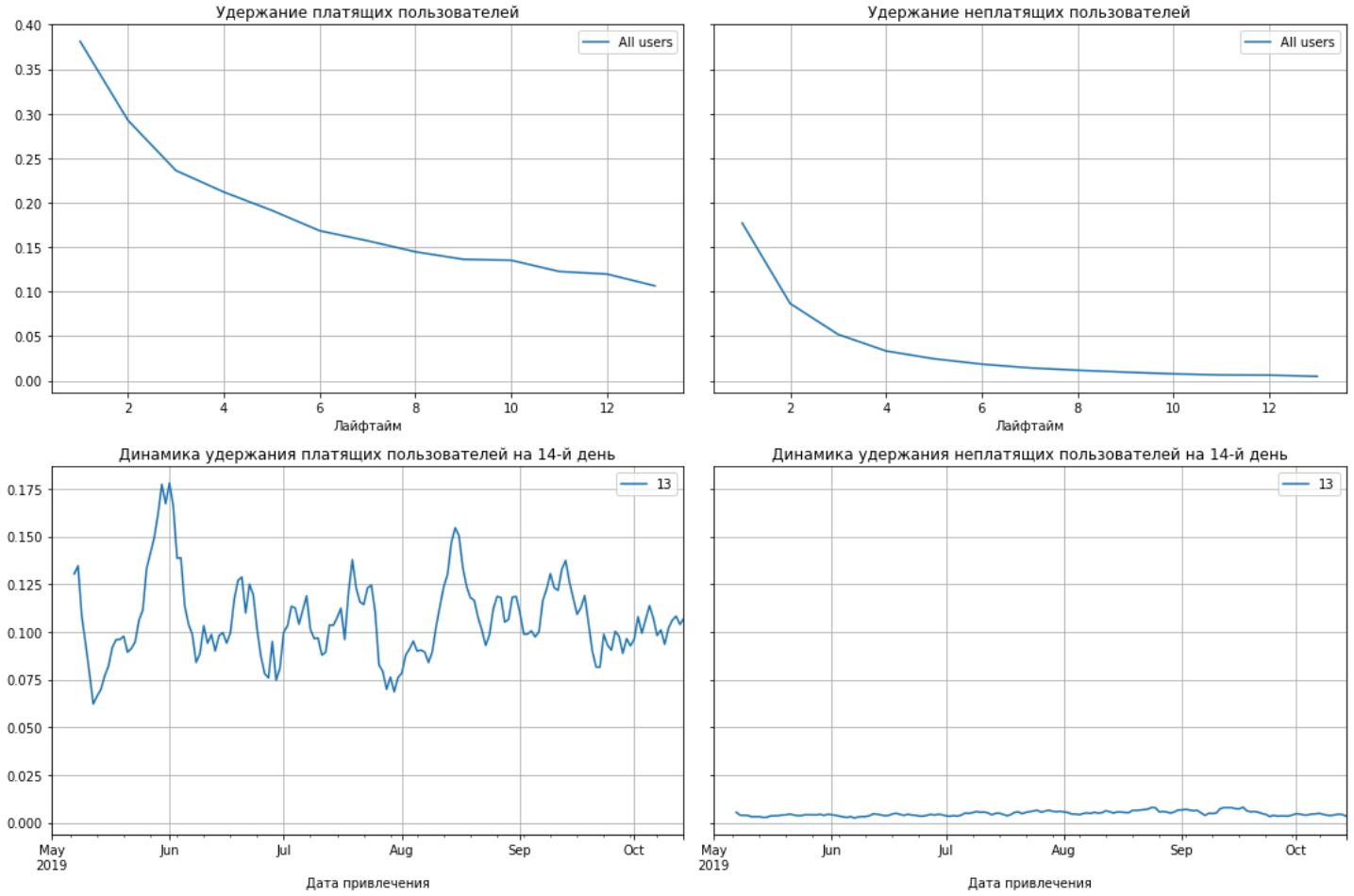
5.4 Удержание пользователей и динамика изменения.

Проверим удержание пользователей и динамику её изменения. Построим и изучим график удержания, используем функцию для расчета удержания

5.4.1 Удержание пользователей без детализации

In [51]:

```
retention_raw, retention_grouped, retention_history = get_retention(
   profiles_CAC,
   visits,
   observation_date,
   horizon_days,
   dimensions=[],
   ignore_horizon=False,
)
plot_retention(retention_grouped, retention_history, horizon_days)
```

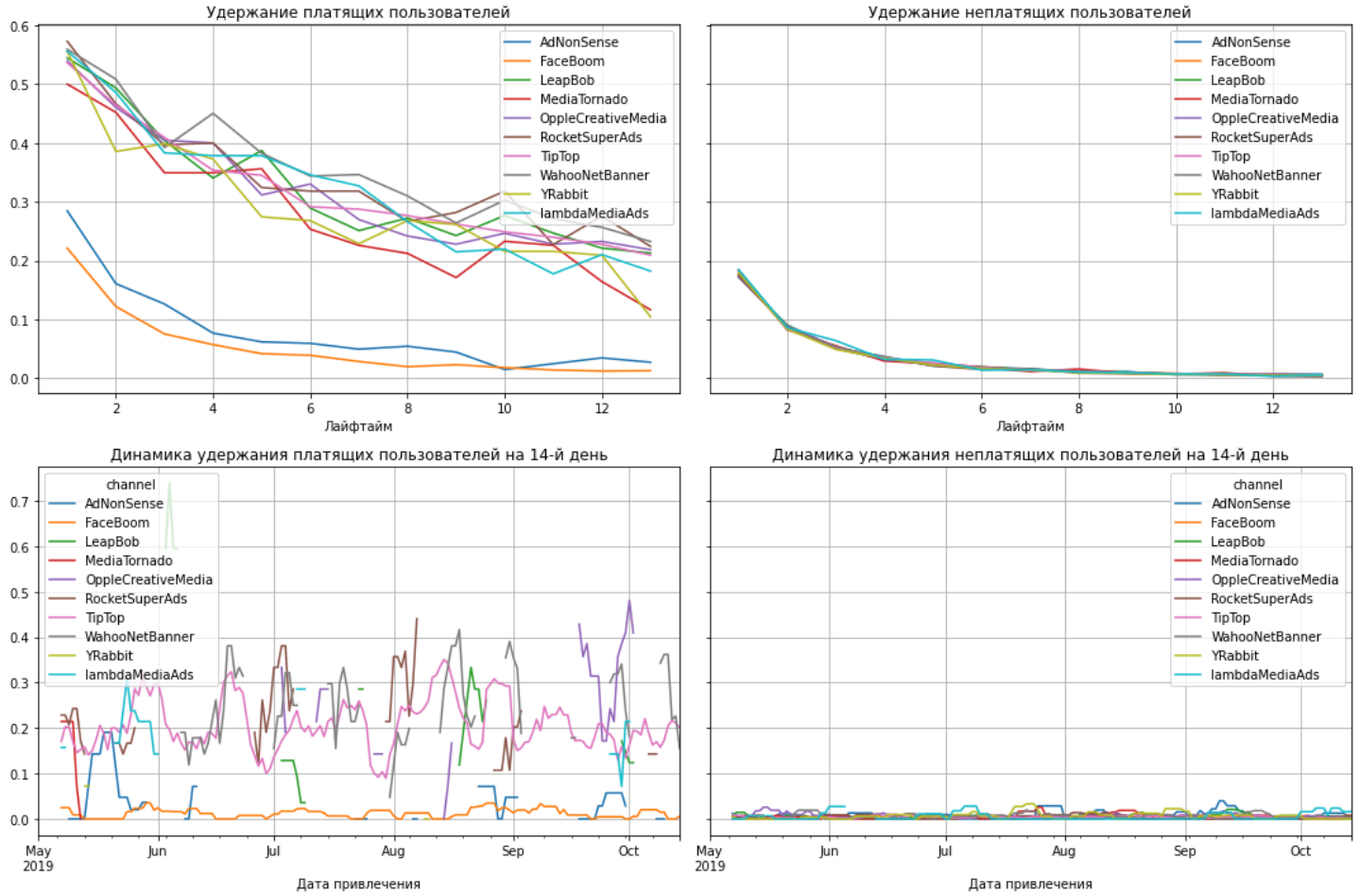


Удержание платящих пользователей на 14 день чуть более 10%. Кривая удержания платящих пользователей выше чем кривая неплатящих. Что собственно не вызвает никаких вопросов - платящие пользователи удерживаются лучше неплатящих. В конце июня и в середине августа были самые высокие показатели удержания платящих пользователей 17,5 %и 15,5 %. Неплатящие пользователи удерживаются стабильно плохо показатель ниже 1 %.

5.4.2 Удержание пользователей по каналам привлечения

In [52]:

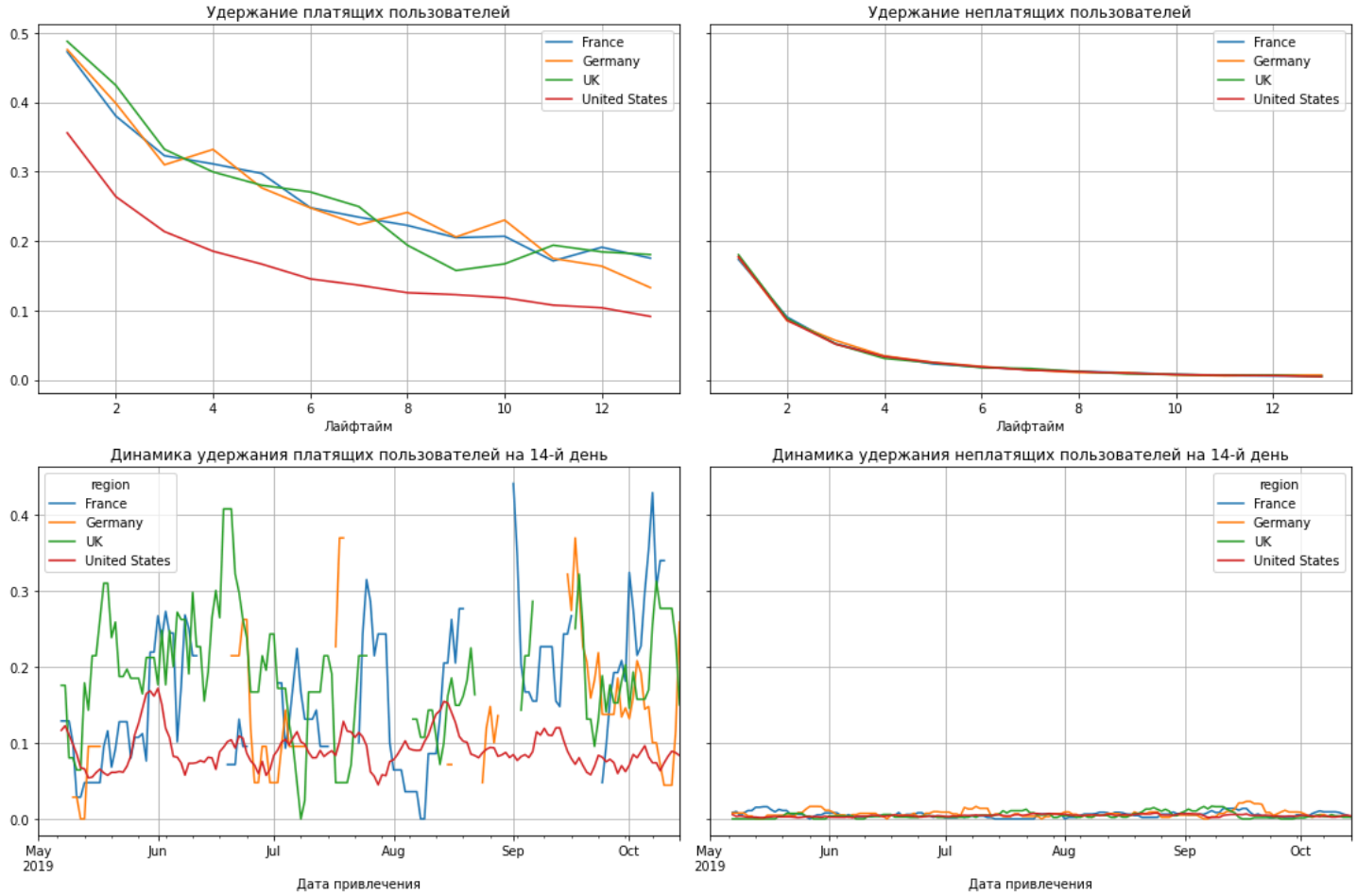
```
retention_raw, retention_grouped, retention_history = get_retention(
   profiles_CAC,
   visits,
   observation_date,
   horizon_days,
   dimensions=['channel'],
   ignore_horizon=False,
)
plot_retention(retention_grouped, retention_history, horizon_days)
```



5.4.3 Удержание пользователей по странам

In [53]:

```
retention_raw, retention_grouped, retention_history = get_retention(
    profiles_CAC,
    visits,
    observation_date,
    horizon_days,
    dimensions=['region'],
    ignore_horizon=False,
)
plot_retention(retention_grouped, retention_history, horizon_days)
```



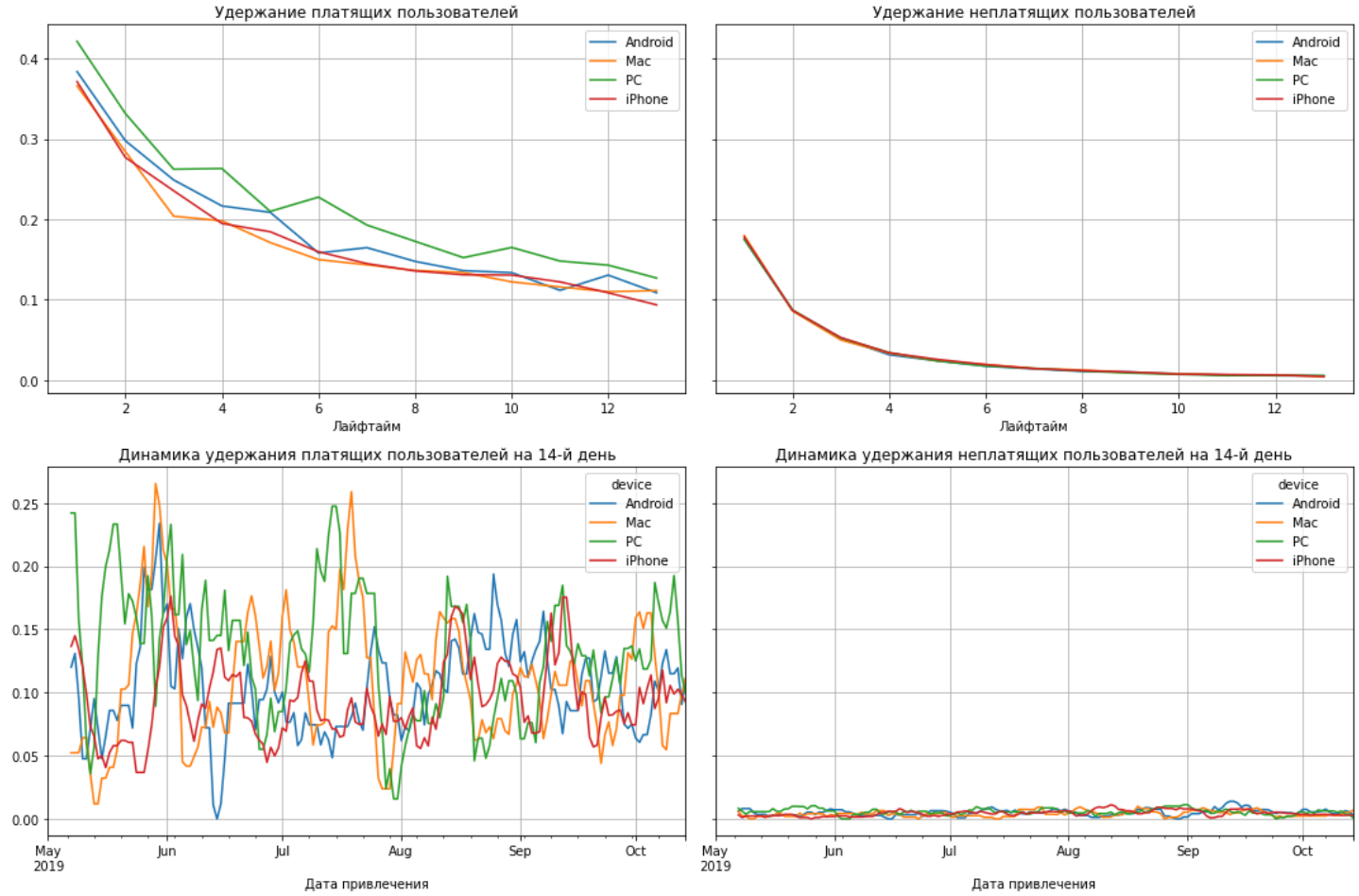
Пользователи из США удерживаются хуже всех, менее 10% на 14 день жизни, при этом конверсия в платящего пользователя в этой стране самая высокая! Остальные страны до 8 дня жизни показывали стабильное плавное сопоставимое между собой падение конверсии, пока Германия с 7 по 9 день включительно не показала отставание от Франции и Великобритании. На 9й день Германия улучшила показатели платящих пользователей и вместе с Францией на 14 день показала результат примерно 18 %, тогда как Германия показала с 9 дня жизни большое охлаждение пользователей к приложению с 18 % снизив до 13-13,5% показатель удержания платящих пользователей. Поведение неплатящие пользователей не показало существенной зависимости от страны, минимальные колебания видны на графике динамики удержания неплатящих пользователей у Франции и Германии.

5.4.4 Удержание пользователей по устройствам

Посмотрим есть ли зависимость между каналом привлечения и удержание платящих пользователей.

In [54]:

```
retention_raw, retention_grouped, retention_history = get_retention(
   profiles_CAC,
   visits,
   observation_date,
   horizon_days,
   dimensions=['device'],
   ignore_horizon=False,
)
plot_retention(retention_grouped, retention_history, horizon_days)
```



Лучшее удержание платящих пользователей у пользователей устройств PC с 42% до 12-13% на 14й день жизни. Может пользователям удобнее открывать приложением именно с этого устройства или производить оплату? Или дело в самих пользователях, может они консервативнее. Причем конверсия в платящих пользователей с устройств PC самая низкая, а удержание самое высокое. Интересный вывод получается.

Устройства Android также показывают удержание выше. чем у устройств iPhone и Mac.

Исходя из данных графика "Удержание платящих пользователей", видим, что 'FaceBoom' и 'AdNonSense' стабильно низкий показатель удержания платящих пользователей. Что примечательно, поведение кривой удержания у каналов 'FaceBoom' и 'AdNonSense очень напоминает ожидание поведения кривой удержания неплатящих пользователей.

у остальных:'YRabbit', 'MediaTornado', 'RocketSuperAds', 'LeapBob', 'TipTop', 'WahooNetBanner', 'OppleCreativeMedia', 'lambdaMediaAds' - этот показатель примерно на одном уровне. Динамика неплатящих пользователей также низкая но для всех каналов привлечения.

Также видно что стабильно платящие пользователи TipTop не имеют разрывов в графике динамики, в отличие от других каналов привлечения. Таких разрывов нет и у FaceBoom.

Примечательно, что удержание неплатящих пользователей из разных каналов, стран, платформ ровное, а удержание платящих очень отличается.

Как правило, пользовательские метрики платящих выше, чем показатели тех, кто не платит. Например, удержание (retention) платящих часто значительно превосходит удержание неплатящих (иногда в несколько раз), и в проекте они остаются намного дольше. Это и видно у нас на графике. И такое поведение логично: заплатив, пользователи проявляют свою лояльность к продукту, мотивация вернуться у них сильнее – нужно использовать то, за что было заплачено.

Однако почему у 'FaceBoom' и 'AdNonSense' очень отличается кривая платящих пользователей. Может быть повторную покупку пользователи этих каналов осуществить не могут из-за ошибки? **Стоит обратить внимание на этот момент**: аномалия в конверсии платящих пользователей у каналов: 'FaceBoom' и 'AdNonSense'.

Также стоит обратить внимание на то, что у США самая высокая конверсия платящих пользователей и самое низкое удержание платящих пользователей.

5.5 Окупаемость рекламы

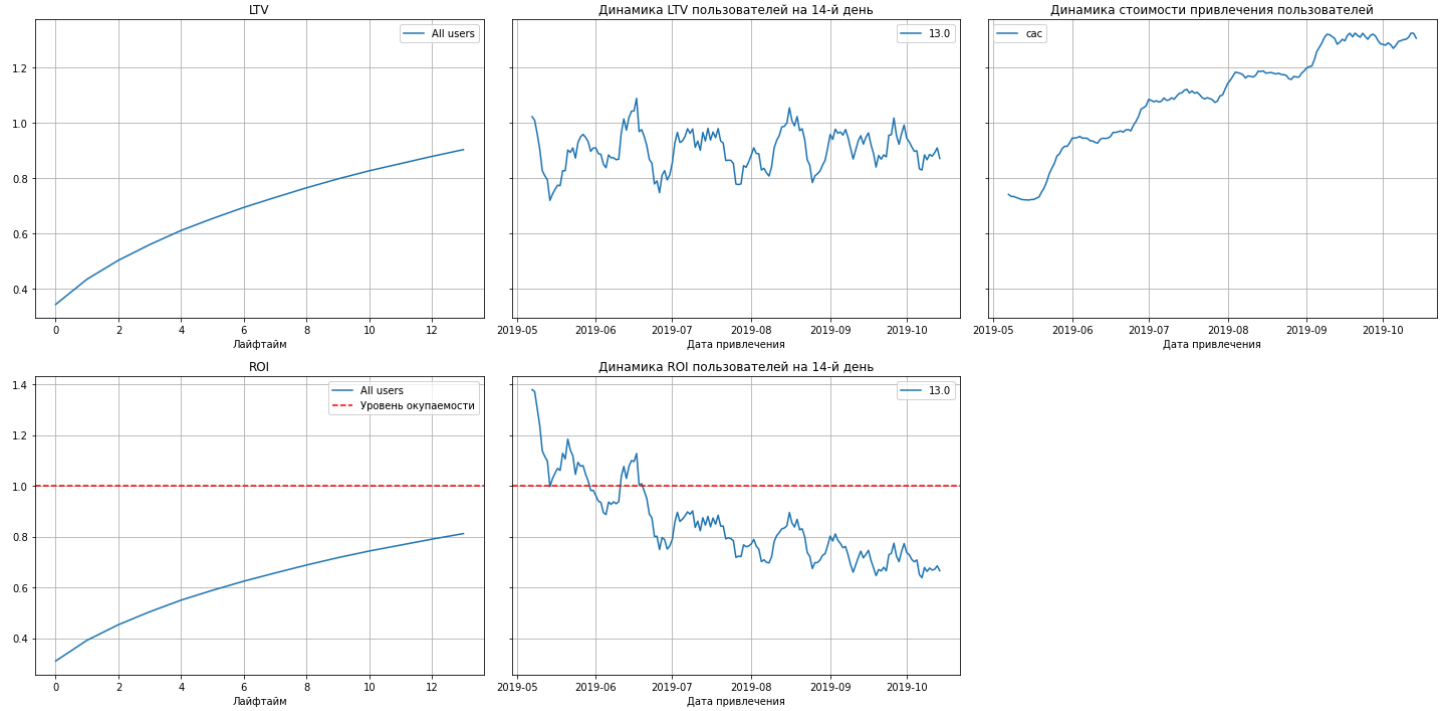
Декомпозируем весь трафик по странам, устройствам, регионам и поищем причины отсутствия прибыли.

5.5.1 Окупаемость рекламы без детализации

Построим графики LTV и ROI, графики динамики LTV, CAC и ROI и Проанализируем окупаемость рекламы в общем. Посчитаем LTV и ROI с помощью заранее подготовленной функции get_ltv, также выведем графики с помощью заранее подготовленной функции plot_ltv_roi.

In [55]:

```
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
  profiles_CAC, orders, observation_date, horizon_days)

plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days)
```



Графики показывают данные без разбивок и группировок, в целом.

Первый график показывает кривые LTV, второй график это график динамики LTV, третий график это динамика CAC, четвертый графиук показывает кривые ROI, а пятый динамика ROI.

Выводы можно сделать такие: реклама не окупается, стоимость привлечения пользователя растет, LTV не высок.

Поведение кривой LTV не вызывает вопросов - растет ожидаемым образом.

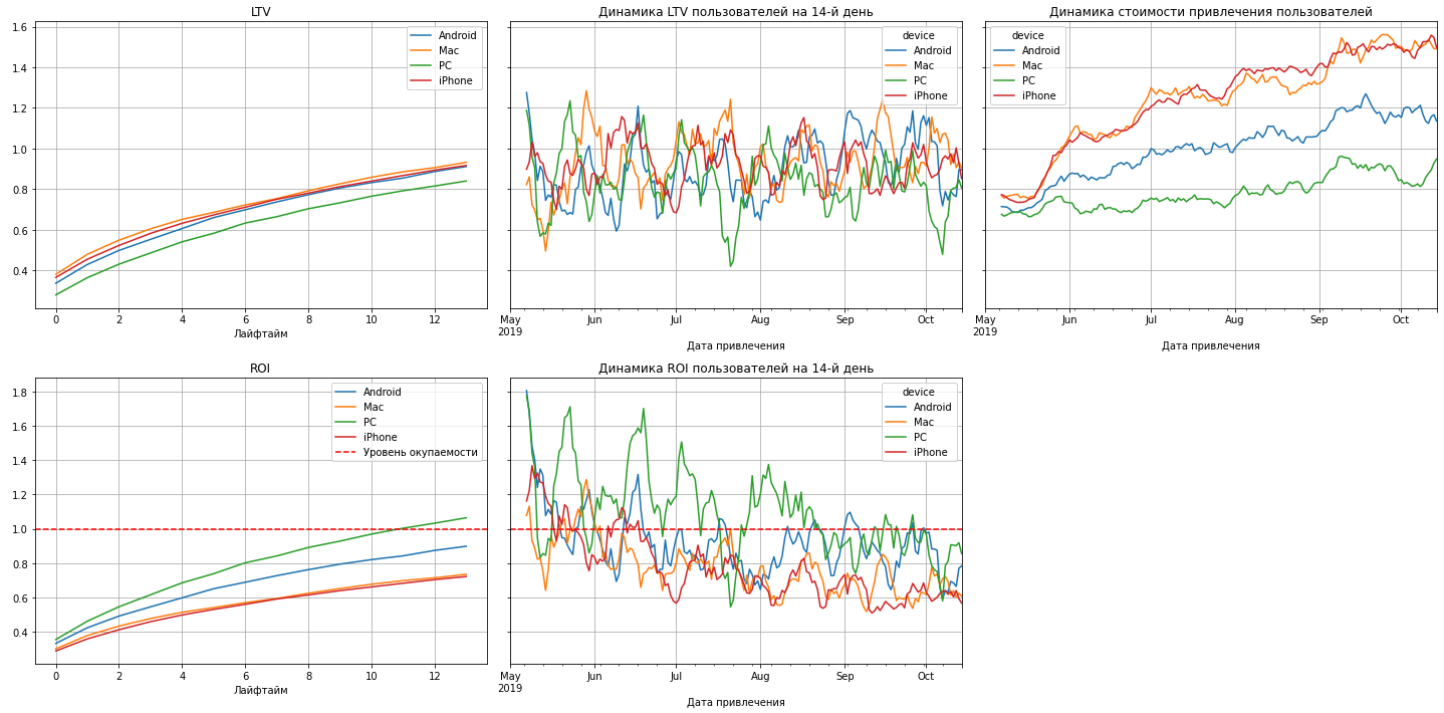
График ROI в динамике также показывает, что показатель падает. Окупаемость инвестиций должна быть выше 100%. Так как затраты на рекламу необходимо вернуть полностью, а всё что выше, уже прибыль компании. У нас же картина показывает отрицательный ROI

5.5.2 Окупаемость рекламы с разбивкой по устройствам.

Проанализируем окупаемость рекламы с разбивкой по устройствам. Построим графики LTV и ROI, а также графики динамики LTV, CAC и ROI но уже с учетом устройств. скоторых были привлечены пользователи.

In [56]:

```
dimensions = ['device']
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
  profiles_CAC, orders, observation_date, horizon_days, dimensions=dimensions)

plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days)
```



Лайфтаймвелью у всех четырех устройств стабильные, ниже всех у PC около 0.8 у.е., остальные примерно одного уровня и на 14й день не превышают 0.9 у.е. По PC также видно, что в динамике его LTV опускался ниже всех до 0,4 и 0,5 у.е, что в динимике CAC PC ниже всех, что значит, что расходы на стоимость привлечения пользователей PC ниже. у iPhone и Mac динамика стоиомсти привлечения пользователей очень близка друг к другу, а Android находится посередине между PC и iPhone и Mac.

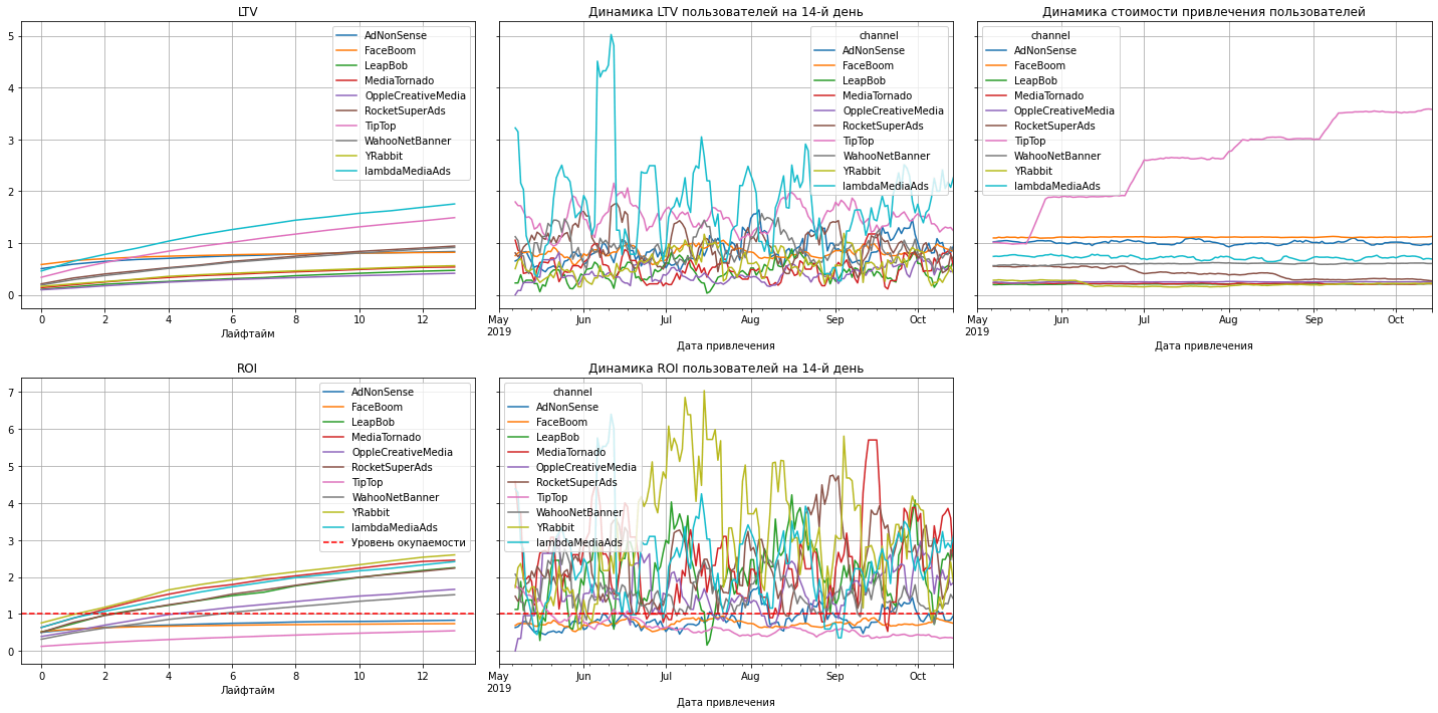
Что интересно, пользователи PC начинают окупаться уже на 11 день и на 12 приносить небольшую прибыль, а вот пользователи с остальных устройств нет, не удивительно, график динамики стоиомсти привлечения пользователей отражает картину на графике ROI - там iPhone и Mac также впереди всех но уже с негативным окрасом - с самым высоким уровнем убытков - они на уровне 0,7 на 14 день. Android также убыточен, его уровень 0,9 на 14 день.

5.5.3 Окупаемость рекламы с разбивкой каналам привлечения.

Проанализируем окупаемость рекламы с разбивкой по рекламным каналам. Построим графики LTV и ROI, а также графики динамики LTV, CAC и ROI.

In [57]:

```
dimensions = ['channel']
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
  profiles_CAC, orders, observation_date, horizon_days, dimensions=dimensions)

plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days)
```



Графики показывают то, что уже встречалось в предыдущих исследованиях. Опять TipTop растет в стоимости привлечения пользователей, остальные каналы стабильны.

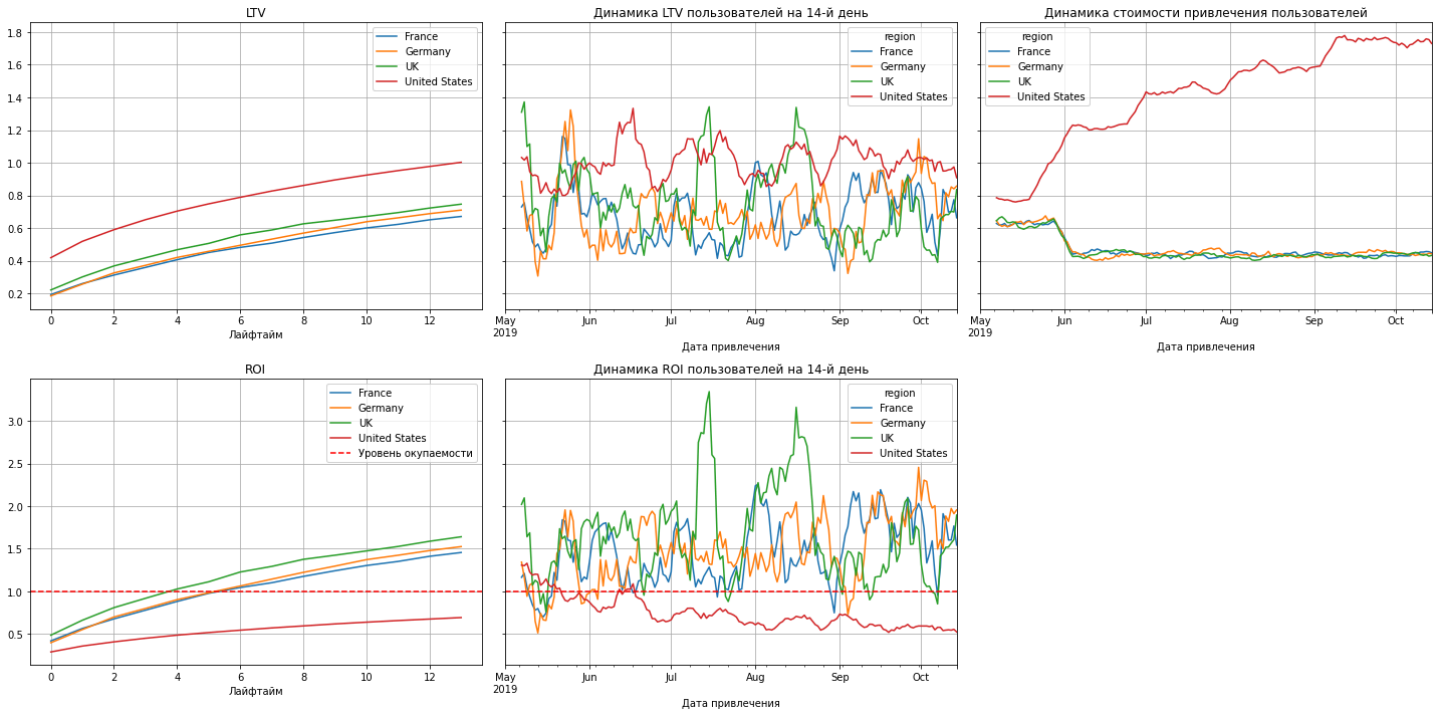
Три канала не окупаются 'FaceBoom', 'TipTop', 'AdNonSense'. остальные окупаются на 2-6дни:'YRabbit', 'MediaTornado', 'RocketSuperAds', 'LeapBob', 'WahooNetBanner', 'OppleCreativeMedia', 'lambdaMediaAds'.

5.5.4 Окупаемость рекламы с разбивкой по странам.

Проанализируем окупаемость рекламы с разбивкой по странам. Построим графики LTV и ROI, а также графики динамики LTV, CAC и ROI.

In [58]:

```
dimensions = ['region']
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
  profiles_CAC, orders, observation_date, horizon_days, dimensions=dimensions)

plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days)
```



LTV выше всего у пользователей из США, на 14 день это 1 у.е., а у остальных стран (Франция, Германия и Великобритания) около 0,7 у.е.

На графике динамики стоимости привлечения пользователей видно, что США выделяется и стоимость привлечения пользователей растет, у остальных стран расходы стабильны с июня, в конце мая было снижение и затем никаких изменений.
График ROI показывает, что все станы окупаются - Великобритания на 4-м дне, Германия и Франция на 5й день, а вот США даже на 14-й день находится ниже на 30-34% от уровня окупаемости, что подтверждает и график в динамике - с середины июня США ни разу не переваливал за черту окупаемости, когда остальные страны стабильно выше этого уровня.
Итоги раздела: Подведем итоги раздела и ответим на такие вопросы:

- Окупается ли реклама, направленная на привлечение пользователей в целом?

Нет реклама в целом, без разбивки на каналы привлечения не окупается это видно в пункте 5.3 на графике LTV - кривая находится на уровне 0.9 (90%) на 14-й день жизни. Это нарушает бизнес-план, по которому окупаемость уже должна быть на 14 день. для этого уровень кривой должен достичь 1 (100%). Прибыли не будет, но реклама окупится.

- Какие устройства, страны и рекламные каналы могут оказывать негативное влияние на окупаемость рекламы?

Регион США, устройства iPhone и Mac и TipTop оказывают негативное влияние на окупаемость. В это легко поверить, т.к. в США популярными устройствами являются iPhone и Mac.

- Чем могут быть вызваны проблемы окупаемости?

Вероятнее всего причина в какой-нибудь технической проблеме.

- итоги оценки: описать возможные причины обнаруженных проблем и промежуточные рекомендации для рекламного отдела.

Необходимо поискать технические ошибки для приложений iPhone и Mac, особенно через канал TipTop. Нет ли неверной настройки рекламы в США. Конверсия пользователей из США значительно выше чем у остальных, но удержание очень низкое.Это может быть инсайтом, зацепкой.

6 ВЫВОД

Огромный бюджет потрачен на привлечения клиентов в кампании FaceBoom, почти треть от всех рекламных затрат. При этом именно от этого источника идет самое низкое удержание платящих пользователей. Именно по нему кривая графика удержания показывает ожидаемое поведение как для неплатящих пользователей.

Пользователи из США, пользователи устройств Iphone и Mac имеют низкий уровень удержания, хотя при этом и показывают неплохую конверсию в платящих пользователей и удержание. При этом окупаемости нет, по устройствам Iphone и Mac, стране США это примерно 70%. в США для канала TipTop уровень CAC имеет аномально большое значение по сравнению с остальными- пиковое значение 3.5, когда остальные каналы примерно 1,1. В среднем один пользователь TipTop обходится в 3.5 раза дороже остальных.

Предлагается обратить внимание при формировании рекламного бюджета на пользователей из стран Великобритании, Германии и Франции, а также на пользователей PC и Android. Акцент с рекламы в FaceBoom, TipTop и AdNonSense сместить на другие площадки. Пользователи из США могут быть не целевыми пользователями. Еще раз просчитать целевую аудиторию, менталитет пользователя какой страны близок нашему продукту.

Придерживаясь этих правил наша реклама выйдет на окупаемость.