Ссылки на дашборд и презентацию.

- Дашборд https://public.tableau.com/authoring/-_16515687271690/Dashboard2#1
- Презе https://www.canva.com/design/DAE_oXKUBp4/cdinrOQDVTQCEVrCueFixw/edit?utm_content=DAE_oXKUBp4&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

---

Мобильные приложения — Анализ поведения пользователей в мобильном приложении

В [1]:
```
#загружаем необходимые библиотеки
 импортировать  pandas  как  pd
 импортировать  matplotlib.pyplot  как  plt
 из  plotly  импортировать  graph_objects  как  go
 импортировать  plotly.express  как  px
 импорт  plotly.io  как  pio
 pio.templates.default = "plotly_white"
 импортировать  numpy  как  np
 импортировать  seaborn  как  sns
 импортировать   itertools
 из  sklearn.metrics  импортировать  silhouette_score
 из  sklearn.кластерный  импорт  K означает
 из  sklearn.импорт метрик   accuracy_score, precision_score, recall_score, f1_score,mean_absol
 из  sklearn.предварительная обработка  импортируйте  стандартный масштаб
 из  sklearn.model_selection  импортируйте  train_test_split
 из  sklearn.linear_model  импортируйте  Лассо,  гребень,LogisticRegression
 из  sklearn.tree  импортируйте  DecisionTreeRegressor
 из  sklearn.ensemble  импортируйте  RandomForestRegressor,  GradientBoostingRegressor, RandomF
 импортируйте  matplotlib.pyplot  как  plt
 из  scipy.cluster.hierarchy  импортируйте  дендрограмму,  привязку
 sns.set(style="whitegrid")
цвета  = ["#ef476f","#ffd166 ","#06d6a0","#118ab2","#073b4c"]
sns.set_palette(sns.color_palette(цвета))
импортировать  re
 из  scipy  импортировать  статистику  как  st
 импортировать  математику  как  mth
 импортировать  предупреждения
 warnings.filterwarnings('игнорировать')
```

Загружаем данные

Первый датафрейм

В [2]:
```
#Загружаем данные и выведем 5 строк датафрейма :
df = pd.read_csv('/datasets/mobile_dataset.csv')
df.head()
```

Выход [...

	событие.время	event.name	user.id
0	2019-10-07 00:00:00.431357	advert_open	020292ab-89bc-4156-9acf-68bc2783f894
1	2019-10-07 00:00:01.236320	tips_show	020292ab-89bc-4156-9acf-68bc2783f894
2	2019-10-07 00:00:02.245341	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c
3	2019-10-07 00:00:07.039334	tips_show	020292ab-89bc-4156-9acf-68bc2783f894
4	2019-10-07 00:00:56.319813	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74197 entries, 0 to 74196
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   event.time  74197 non-null  object
 1   event.name  74197 non-null  object
 2   user.id     74197 non-null  object
dtypes: object(3)
memory usage: 1.7+ MB
```

In [4]:
```
#Выведем размер
print(df.shape)
```

```
(74197, 3)
```

В фрейме 74197 строки, 3 столбца.Тип данных - object.

Имеем столбцы:

- event.time - время совершения
- event.name - действие пользователя
- user.id - идентификатор пользователя

Второй датафрейм:

In [5]:
```
#Загружаем данные и выведем 5 строк датафрейма :
df1 = pd.read_csv('/datasets/mobile_soures.csv')
df1.head()
```

Out[5]:

	userId	source
0	020292ab-89bc-4156-9acf-68bc2783f894	other
1	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex
2	8c356c42-3ba9-4cb6-80b8-3f868d0192c3	yandex
3	d9b06b47-0f36-419b-bbb0-3533e582a6cb	other
4	f32e1e2a-3027-4693-b793-b7b3ff274439	google

In [6]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4293 entries, 0 to 4292
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   userId  4293 non-null   object
 1   source  4293 non-null   object
dtypes: object(2)
memory usage: 67.2+ KB
```

In [7]:
```
#Выведем размер
print(df1.shape)
```

```
(4293, 2)
```

В фрейме 4293 строки, 2 столбца.Тип данных - object.

Имеем столбцы:

- userId - идентификатор пользователя
- source - источник, с которого пользователь установил приложение

Предобработка и исследовательский анализ данных

Предобработка

Первый датасет:

In [8]:
```
# Названия столбцов:
df.columns
```

Out[8]:
```
Index(['event.time', 'event.name', 'user.id'], dtype='object')
```

In [9]:
```
# Переименуем  столбцы :
df = df.rename(columns={'event.time':'even_time'})
df = df.rename(columns={'event.name':'event_name'})
df = df.rename(columns={'user.id':'user_id'})
```

In [10]:
```
df.head()
```

Out[10]:

	even_time	event_name	user_id
0	2019-10-07 00:00:00.431357	advert_open	020292ab-89bc-4156-9acf-68bc2783f894
1	2019-10-07 00:00:01.236320	tips_show	020292ab-89bc-4156-9acf-68bc2783f894
2	2019-10-07 00:00:02.245341	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c
3	2019-10-07 00:00:07.039334	tips_show	020292ab-89bc-4156-9acf-68bc2783f894
4	2019-10-07 00:00:56.319813	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c

In [11]:
```
# проверим на дубликаты
df.duplicated(subset=['even_time','event_name','user_id']).sum()
```

Out[11]:
```
0
```

Дубликатов не обнаружено

In [12]:
```
# проверим на пропуски
df.isna().sum()
```

Out[12]:
```
even_time     0
event_name    0
user_id       0
dtype: int64
```

Пропусков не обнаружено

Заменим типы данных

In [13]:
```
# Заменим типы данных
df['even_time'] = df['even_time'].astype('datetime64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74197 entries, 0 to 74196
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   even_time   74197 non-null  datetime64[ns]
 1   event_name  74197 non-null  object        
 2   user_id     74197 non-null  object        
dtypes: datetime64[ns](1), object(2)
memory usage: 1.7+ MB
```

In [14]:
```
df.sample(3)
```

Out[14]:

	even_time	event_name	user_id
1696	2019-10-07 19:55:31.525460	tips_show	6015244c-a368-4a92-adc8-4088589ea08a
72229	2019-11-03 14:09:35.632610	photos_show	a107daca-36e9-458e-93f2-ab615bdbb394
9629	2019-10-11 10:56:22.893908	search_1	fbcc37df-bb78-4adc-858f-5fe0047ea247

Второй датасет:

In [15]:
```
# Названия столбцов:
df1.columns
```

Out[15]:
```
Index(['userId', 'source'], dtype='object')
```

In [16]:
```
# Переименуем  столбец :
df1 = df1.rename(columns={'userId':'user_id'})
df1.head()
```

Out[16]:

	user_id	source
0	020292ab-89bc-4156-9acf-68bc2783f894	other
1	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex
2	8c356c42-3ba9-4cb6-80b8-3f868d0192c3	yandex
3	d9b06b47-0f36-419b-bbb0-3533e582a6cb	other
4	f32e1e2a-3027-4693-b793-b7b3ff274439	google

In [17]:
```
# проверим на дубликаты
df1.duplicated(subset=['source','user_id']).sum()
```

Out[17]:
```
0
```

Дубликатов не обнаружено

In [18]:
```
# проверим на пропуски
df1.isna().sum()
```

Out[18]:
```
user_id    0
source     0
dtype: int64
```

Пропусков не обнаружено

Исследование данных (EDA)

Первый датасет

In [19]:
```
df.sample(3)
```

Out[19]:

	even_time	event_name	user_id
47980	2019-10-25 18:57:07.583657	advert_open	28af1ea4-9be3-4e9e-be07-1838cfe0e1cb
45234	2019-10-24 18:22:37.089546	photos_show	1022b793-7ae7-4ac0-8c52-54ffed1c8182

	even_time	event_name	user_id
19871	2019-10-15 14:30:16.464849	tips_show	78c69c24-9280-4c80-be68-1539d69e4a9a

In [20]:
```
# столбец even_time
df['even_time'].describe()
```

Out[20]:
```
count                           74197
unique                          74197
top       2019-10-19 18:36:07.773802
freq                                1
first     2019-10-07 00:00:00.431357
last      2019-11-03 23:58:12.532487
Name: even_time, dtype: object
```

In [21]:
```
df1.sample(3)
```

Out[21]:

	user_id	source
823	389ba51a-f86f-47ba-8e11-a70e8e92e31e	google
610	4a2a1260-dc8c-4c9a-b79f-06f74a9c4a2e	yandex
2043	86eba2c8-e5ad-49cf-baa7-fc6b7a4e15a1	other

In [22]:
```
print('Мы располагаем периодом =', (df['even_time'].max() - df['even_time'].min()))
```

```
Мы располагаем периодом = 27 days 23:58:12.101130
```

Мы располагаем данными с 7октября по 3ноября 2019 года. Период времени 28 дней

In [23]:
```
# столбец event_name
df['event_name'].value_counts()
```

Out[23]:
```
tips_show        40055
photos_show      10012
advert_open       6164
contacts_show     4450
map               3881
search_1          3506
favorites_add     1417
search_5          1049
tips_click         814
search_4           701
contacts_call      541
search_3           522
search_6           460
search_2           324
search_7           222
show_contacts       79
Name: event_name, dtype: int64
```

Имеем схожие столбцы contacts_show и show_contacts, а так же search_1-7. Объединим их.

In [24]:
```
df.loc[(df.query('event_name == "show_contacts"').index),'event_name'] = "contacts_show"
df.loc[(df.query('event_name == "search_1"').index),'event_name'] = "search"
df.loc[(df.query('event_name == "search_2"').index),'event_name'] = "search"
df.loc[(df.query('event_name == "search_3"').index),'event_name'] = "search"
df.loc[(df.query('event_name == "search_4"').index),'event_name'] = "search"
df.loc[(df.query('event_name == "search_5"').index),'event_name'] = "search"
df.loc[(df.query('event_name == "search_6"').index),'event_name'] = "search"
df.loc[(df.query('event_name == "search_7"').index),'event_name'] = "search"
df['event_name'].value_counts()
```

Out[24]:
```
tips_show        40055
```

```
photos_show      10012
search            6784
advert_open       6164
contacts_show     4529
map               3881
favorites_add     1417
tips_click         814
contacts_call      541
Name: event_name, dtype: int64
```

- Имеем виды действий:
- tips_show - увидел рекомендованные объявления
- photos_show - просмотрел фотографий в объявлении
- search - разные действия, связанные с поиском по сайту
- advert_open - открыл карточки объявления
- contacts_show - посмотрел номер телефона
- map - открыл карту объявлений
- favorites_add - добавил объявление в избранное
- tips_click - кликнул по рекомендованному объявлению
- contacts_call - позвонил по номеру из объявления

In [25]:
```
print('Кол-во уникальных пользователей =',df['user_id'].nunique())
```

```
Кол-во уникальных пользователей = 4293
```

Имеем 4293 уникальных пользователя,

Второй датасет

In [26]:
```
df1['source'].value_counts()
```

Out[26]:
```
yandex    1934
other     1230
google    1129
Name: source, dtype: int64
```

In [27]:
```
print(f'Всего строк - {len(df1)} \nУникальных идентификаторов - {df1["user_id"].nunique()}')
```

```
Всего строк - 4293
Уникальных идентификаторов - 4293
```

Имеем три имени источника.Все пользователи уникальны. 4293 пользователя, как и в таблице df.

Проверим совпадают ли идентификаторы в обоих датасетах

In [28]:
```
if set(df['user_id']) == set(df1['user_id']):
    print('Множества идентификаторов в обоих датасетах совпадают +++')
else:
    print('Множества не совпадают ---')
```

```
Множества идентификаторов в обоих датасетах совпадают +++
```

Идентификаторы и их кол-во совпадают . Объединим таблицы.

In [29]:
```
df = df.merge(df1, on='user_id', how='left')
df.head()
```

Out[29]:

	even_time	event_name	user_id	source
0	2019-10-07 00:00:00.431357	advert_open	020292ab-89bc-4156-9acf-68bc2783f894	other
1	2019-10-07 00:00:01.236320	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other
2	2019-10-07 00:00:02.245341	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex

	even_time	event_name	user_id	source
3	2019-10-07 00:00:07.039334	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other
4	2019-10-07 00:00:56.319813	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex

Вывод:

```
- В процессе предобработки и исследования данных :
    - пропусков и дубликатов не обнаружено
    - заменены названия некоторых столбцов
    - располагаем периодом = 27 days 23:58:12.101130
    - объединены схожие виды действий
    - заменен тип данных на datetime
    - оба датасета объединены в один
    - кол-во уникальных пользователей = 4293
```

Исследовательская часть

Рассчет retention rate

Retention Rate — это коэффициент удержания клиентов. Он показывает, насколько долго остаются постоянные клиенты с компанией.Стабильность продаж и рост выручки — постоянные клиенты увеличивают прибыль.

In [30]:
```
df['even_time'] = pd.to_datetime(df['even_time'], format='%Y-%m-%d %H:%M:%S')
df['date_time'] = df['even_time'].dt.strftime("%Y-%m-%d %H:%M:%S")
df['even_time'] = df['even_time'].dt.strftime("%Y-%m-%d")
df['even_time'] = pd.to_datetime(df['even_time'], format='%Y-%m-%d')
df['date_time'] = pd.to_datetime(df['date_time'], format='%Y-%m-%d %H:%M:%S')
df.head()
```

Out[30]:

	even_time	event_name	user_id	source	date_time
0	2019-10-07	advert_open	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:00
1	2019-10-07	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:01
2	2019-10-07	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex	2019-10-07 00:00:02
3	2019-10-07	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:07
4	2019-10-07	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex	2019-10-07 00:00:56

Определим событие и период — на их основе сформируем когорту. Возьмём дату, когда пользователь впервые проявил активность в мобильном приложении

In [31]:
```
first_activity_date = df.groupby(['user_id'])['even_time'].min()
first_activity_date.name = 'first_activity_date'
user_activity = df.join(first_activity_date,on='user_id')
user_activity.head()
```

Out[31]:

	even_time	event_name	user_id	source	date_time	first_activity_date
0	2019-10-07	advert_open	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:00	2019-10-07
1	2019-10-07	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:01	2019-10-07
2	2019-10-07	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex	2019-10-07 00:00:02	2019-10-07

	even_time	event_name	user_id	source	date_time	first_activity_date
3	2019-10-07	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:07	2019-10-07
4	2019-10-07	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex	2019-10-07 00:00:56	2019-10-07

Получим день начала недели, за которую произошло событие. Он станет идентификатором недели. Параметр unit метода pd.to_timedelta задаёт единицу измерения — в нашем случае дня: unit='d'. Вычтем из даты порядковый номер дня:

In [32]:
```
user_activity['activity_week'] = pd.to_datetime(user_activity['even_time'],
                                                unit='d') - pd.to_timedelta(user_activity['ever
user_activity['first_activity_week'] = pd.to_datetime(user_activity['first_activity_date'],
                                                      unit='d') - pd.to_timedelta(user_activity
user_activity['first_activity_week'].head()
```

Out[32]:
```
0   2019-10-07
1   2019-10-07
2   2019-10-07
3   2019-10-07
4   2019-10-07
Name: first_activity_week, dtype: datetime64[ns]
```

Теперь для каждой строки датафрейма можно рассчитать lifetime(время нахождения) пользователя в рамках когорты

In [33]:
```
user_activity['cohort_lifetime'] = user_activity['activity_week'] - user_activity['first_activi
user_activity['cohort_lifetime'] = user_activity['cohort_lifetime'] / np.timedelta64(1,'W')
user_activity['cohort_lifetime'] = user_activity['cohort_lifetime'].astype(int)
user_activity['cohort_lifetime']
```

Out[33]:
```
0        0
1        0
2        0
3        0
4        0
        ..
74192    2
74193    2
74194    1
74195    2
74196    2
Name: cohort_lifetime, Length: 74197, dtype: int64
```

Сгруппируем данные по когорте и lifetime. Посчитаем для каждой когорты количество активных пользователей на определённую «неделю жизни

In [34]:
```
cohorts = user_activity.groupby(['first_activity_week','cohort_lifetime']).agg({'user_id':'nuni
cohorts.head()
```

Out[34]:

	first_activity_week	cohort_lifetime	user_id
0	2019-10-07	0	1130
1	2019-10-07	1	272
2	2019-10-07	2	170
3	2019-10-07	3	119
4	2019-10-14	0	1166

Чтобы найти Retention Rate, нужно сперва получить число пользователей, изначально бывших в

когорте, и на него разделить число пользователей в каждую следующую неделю. Найдём исходное количество пользователей в когорте. Возьмём их число на нулевую неделю:

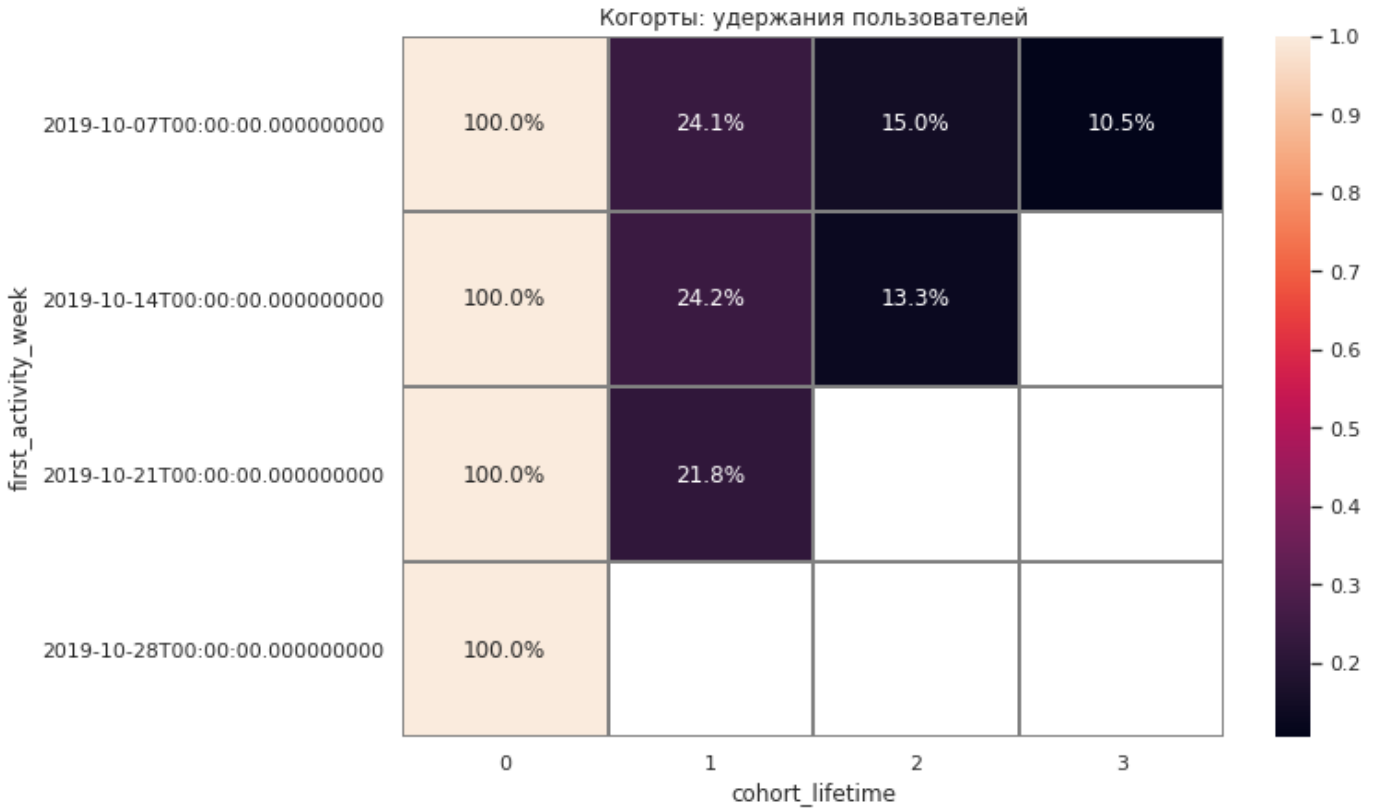
In [35]:
```
initial_users_count = cohorts[cohorts['cohort_lifetime'] == 0][['first_activity_week','user_id'
initial_users_count = initial_users_count.rename(columns={'user_id':'cohort_users'})
cohorts = cohorts.merge(initial_users_count,on='first_activity_week')
cohorts.head()
```

Out[35]:

	first_activity_week	cohort_lifetime	user_id	cohort_users
0	2019-10-07	0	1130	1130
1	2019-10-07	1	272	1130
2	2019-10-07	2	170	1130
3	2019-10-07	3	119	1130
4	2019-10-14	0	1166	1166

Наконец, рассчитаем Retention Rate. Разделим количество активных пользователей в каждую из недель на исходное число пользователей в когорте:

In [36]:
```
cohorts['retention'] = cohorts['user_id']/cohorts['cohort_users']
retention_pivot = cohorts.pivot_table(index='first_activity_week',columns='cohort_lifetime',val
sns.set(style='white')
plt.figure(figsize=(10, 7))
plt.title('Когорты: удержания пользователей')
sns.heatmap(retention_pivot, annot=True, fmt='.1%', linewidths=1, linecolor='gray')
plt.show()
```



Вывод:

- Retention Rate в первую неделю убывает по когортам с течением времени. Если для когорты пользователей, пришедших с 7 октября по 13 октября Retention Rate в первую неделю составляет 24,1%, то для пользователей, пришедших с 21 по 27 октября — уже 21,8%.
- Для малого и среднего бизнеса хорошим Retention Rate считается более 60%.
- В данном случае имеем намного ниже. Коэффициент Оттока более 70%

- Так же чем больше когорта проводит времени в приложении, тем меньше пользователей осталось
- Следует обратить внимание на то, почему уходят клиенты, что именно их не устраивает или вызывает негатив

Анализ событий

Оценим по частоте события, совершаемые пользователями, просматривающих контакты

In [37]:
```
df['event_name'].value_counts()
```

Out[37]:
```
tips_show        40055
photos_show      10012
search            6784
advert_open       6164
contacts_show     4529
map               3881
favorites_add     1417
tips_click         814
contacts_call      541
Name: event_name, dtype: int64
```

Самое частые действия - tips_show (40055) и photos_show (10012), самые редкие - contacts_call(541) и tips_click (814)

По условию, ЦЕЛЕВОЕ СОБЫТИЕ (далее ЦС) - это "contacts_show"

In [38]:
```
# Посчитаем пользователей и события
print("Кол-во пользователей =", df['user_id'].nunique())
print("Кол-во событий =", df['event_name'].count())
```

```
Кол-во пользователей = 4293
Кол-во событий = 74197
```

In [39]:
```
df.groupby('user_id')['event_name'].count().describe()
```

Out[39]:
```
count    4293.000000
mean       17.283252
std        29.130677
min         1.000000
25%         5.000000
50%         9.000000
75%        17.000000
max       478.000000
Name: event_name, dtype: float64
```

In [40]:
```
# Вычислим среднее кол-во событий пользователя за день
print('Кол-во событий которое совершает один пользователь за день -', round(df.groupby('user_id
```

```
Кол-во событий которое совершает один пользователь за день - 9.0
```

Посчитаем сколько пользователей совершали "contacts_show" и сколько раз

In [41]:
```
contact_show_users = df.query('event_name == "contacts_show"').groupby('user_id')['event_name']
print('Кол-во пользователей совершивших ЦЕЛЕВОЕ действие =', len(contact_show_users))
contact_show_users.head()
```

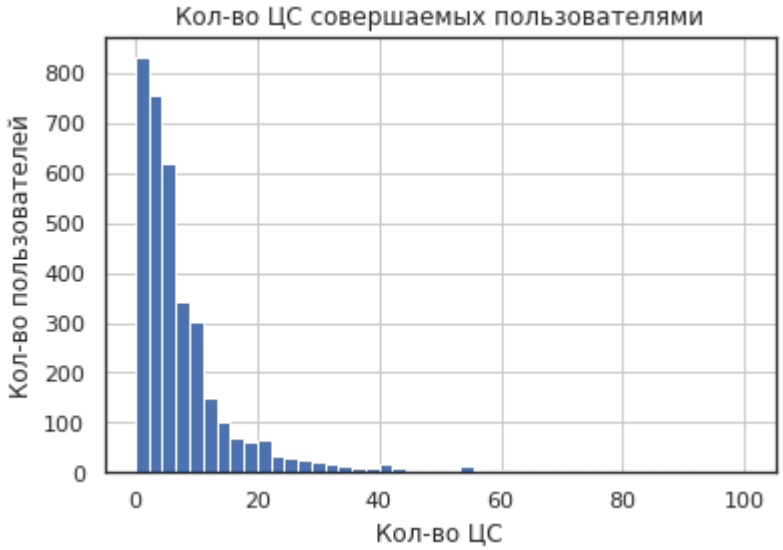
```
Кол-во пользователей совершивших ЦЕЛЕВОЕ действие = 981
```

Out[41]:
```
user_id
e38cb669-7335-4d56-9de5-c8d5d2f13fd3    137
320cab3c-e823-4dff-8c01-c4253764640a    100
cb36854f-570a-41f4-baa8-36680b396370     86
be1449f6-ca45-4f94-93a7-ea4b079b8f0f     83
9b835c74-8ede-4586-9f59-e5473aa48de2     74
Name: event_name, dtype: int64
```

Подсчитаем сколько ЦС совершает пользователь

При расчете нужно удалить из датафрейма событие tips_show, так как это автоматическое событие, котороые не отражает взаимодействие с приложением

In [42]:
```
event_by_user = df.query('event_name !="tips_show"').groupby(['user_id']).agg({'event_name':'co
event_by_user['event_name'].hist(bins=45, range=(0,100)), event_by_user.describe(), event_by_us
plt.title('Кол-во ЦС совершаемых пользователями')
plt.xlabel('Кол-во ЦС')
plt.ylabel('Кол-во пользователей');
```



In [43]:
```
event_by_user.describe()
```

Out[43]:

	event_name
count	3586.000000
mean	9.520915
std	16.318064
min	1.000000
25%	3.000000
50%	5.000000
75%	10.000000
max	336.000000

- В среднем, пользователь совершает от 3 до 10 просмотров контактов,медианное -5. Max - 336. Данные за представленный период 28 дней.

Подсчитаем сколько и каких событий совершает один пользователь

In [44]:
```
df['week'] = df['even_time'].dt.week
df['day'] = df['even_time'].dt.strftime('%Y-%m-%d')
df.head()
```

Out[44]:

	even_time	event_name	user_id	source	date_time	week	day
0	2019-10-07	advert_open	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:00	41	2019-10-07
1	2019-10-07	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:01	41	2019-10-07

	even_time	event_name	user_id	source	date_time	week	day
2	2019-10-07	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex	2019-10-07 00:00:02	41	2019-10-07
3	2019-10-07	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	other	2019-10-07 00:00:07	41	2019-10-07
4	2019-10-07	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex	2019-10-07 00:00:56	41	2019-10-07

Для визуализации кол-ва событий за день построим графики:

- с учетом tips_show'
- без учета tips_show

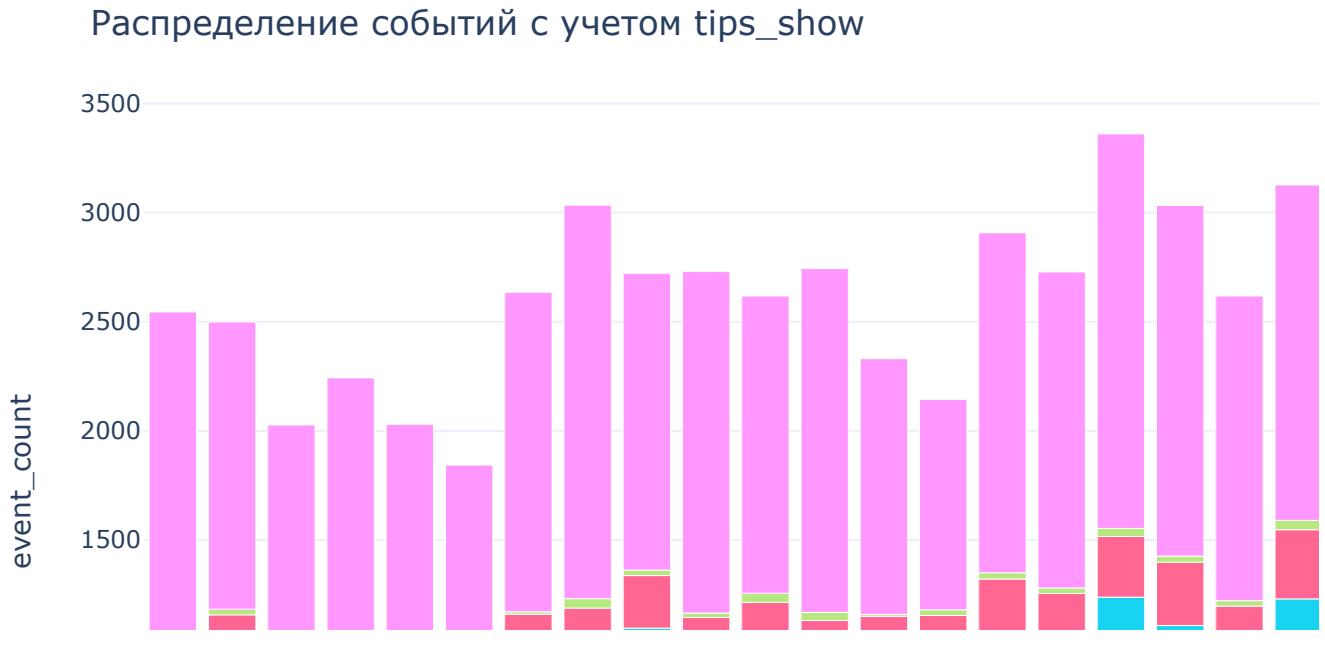
In [45]:
```
distribution_event = df.groupby(['even_time','event_name']).agg({'date_time':'count'}).reset_ir
distribution_event.columns = ['even_time', 'event_name', 'event_count']
count_sum = distribution_event.groupby('even_time').agg({'event_count':'sum'})['event_count'].r
count_sum.columns = ['even_time', 'event_sum']
distribution_event = distribution_event.merge(count_sum, on='even_time', how='left')
distribution_event.head()
```

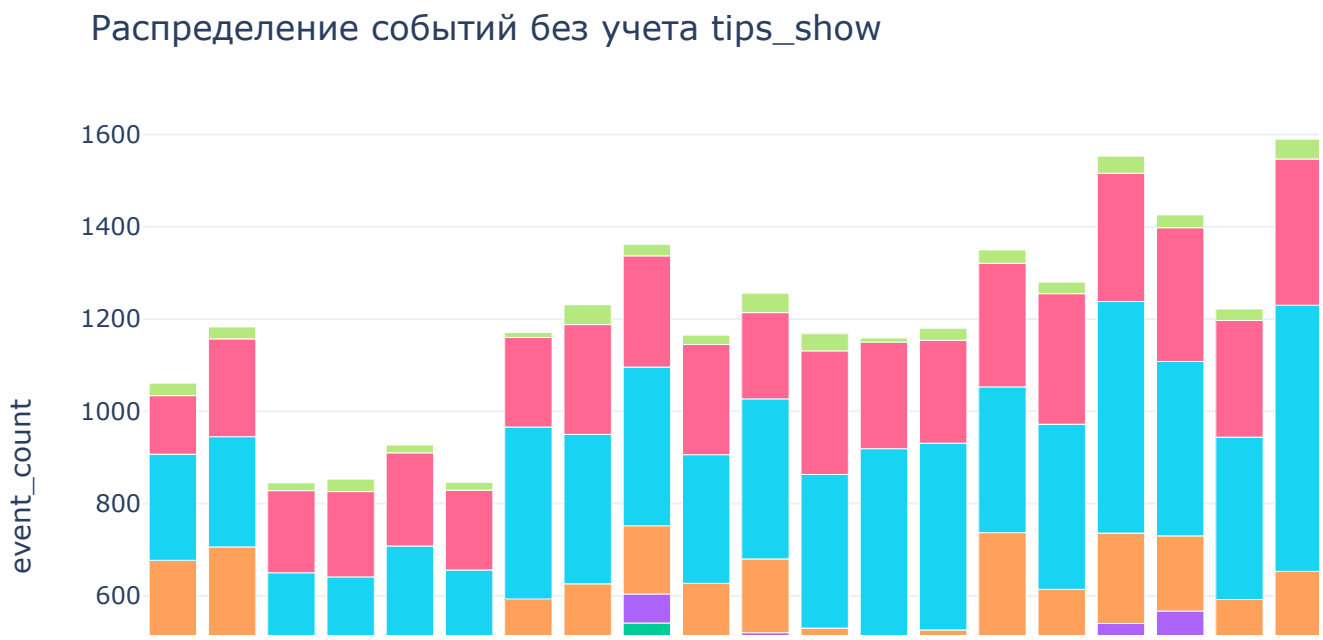
Out[45]:

	even_time	event_name	event_count	event_sum
0	2019-10-07	advert_open	401	2545
1	2019-10-07	contacts_call	7	2545
2	2019-10-07	contacts_show	61	2545
3	2019-10-07	favorites_add	40	2545
4	2019-10-07	map	168	2545

In [46]:
```
fig_distribution_event_1 = px.bar(distribution_event, x='even_time', y='event_count', color='ev
fig_distribution_event_1.update_layout(title_text='Распределение событий с учетом tips_show')
fig_distribution_event_1.for_each_trace(lambda t: t.update(name=t.name.split("=")[0]))
fig_distribution_event_1.show()

fig_distribution_event = px.bar(distribution_event.query('event_name !="tips_show"'), x='even_t
fig_distribution_event.update_layout(title_text='Распределение событий без учета tips_show')
fig_distribution_event.for_each_trace(lambda t: t.update(name=t.name.split("=")[0]))
fig_distribution_event.show()
```





Для лучшей видимости, во втором графике был удален столбец с событиями 'tips_show'

Вывод:

- Активность пользователей меняется по дням недели, на выходных спад
- Наибольшее количество у просмотров рекомендованных объявлений, почти в 2 раза больше, чем у остальных вместе взятых
- На втором месте просмотры фото, что объяснимо(зачастую вещи б/у)
- На третьем поиск на сайте. Многие пользователи предпочитают искать самостоятельно
- Очень мало кликнувших по рекомендованному объявлению, конверсия низкая
- Так же очень мало добавивших в избранное и позвонивших
- Гораздо больше открывших карту объявлений, особенно в начале. Пользователи отслеживают свои объявления

Проанализируем, часто ли эти события бывают самостоятельными событиями пользователя или образуют какую- то воронку

Выделим событие "Посмотрел номер телефона" и посмотрим, какие дейсвия пользователи совершают чаще до и после него.

In [47]:
```
df['event_name'].value_counts()
```

Out[47]:
```
tips_show        40055
photos_show      10012
search            6784
advert_open       6164
contacts_show     4529
map               3881
favorites_add     1417
tips_click         814
contacts_call      541
Name: event_name, dtype: int64
```

Анализ действий пользователей показывает, что имеем два варианта:

- поиск на сайте - открыл карточку объявлений - посмотрел фото - посмотрел номер телефона - позвонил
- увидел рекомендованное объявление -кликнул - открыл карточку объявлений-- посмотрел фото - посмотрел номер телефона - позвонил
- Общей воронки нет, есть несколько локальных

Построим две из них

Сравним конверсию воронки взаимодействия, с основными карточками, с помощью поиска по сайту search

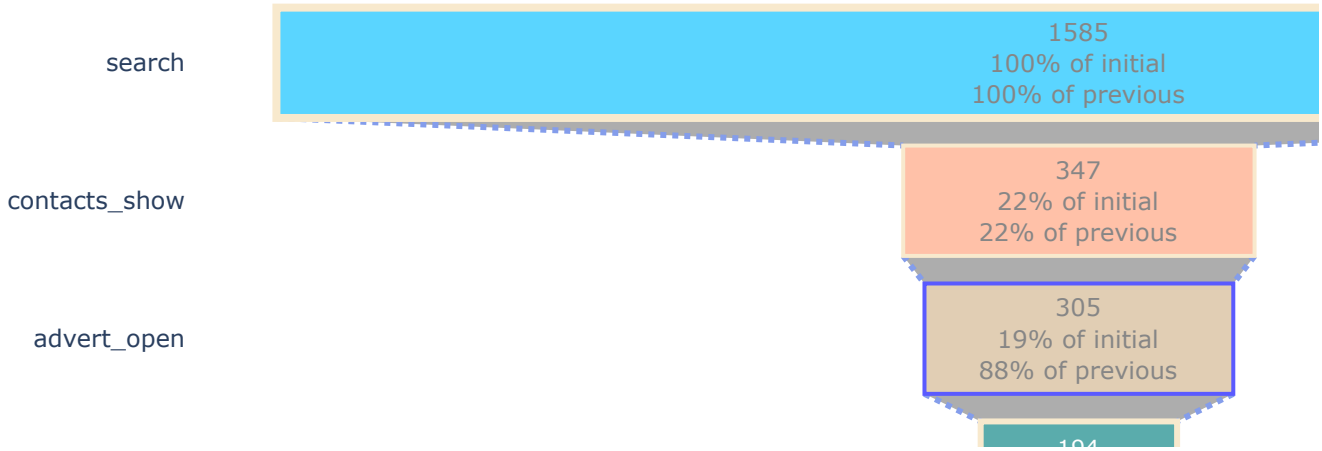
- Сформируем датафрейм :
  - Уберем пользователей, которые совершали взаимодейсвтией с карточкой tips_click
  - Оставим в датафрейме карточки: search, advert_open, contacts_show, favorites_add, contacts_call

In [48]:
```
tips_click = df.query('event_name == "tips_click"')
tips_click_users_list = tips_click['user_id'].unique().tolist()
search = df.query('event_name == "search"')
search_users_list = search['user_id'].unique().tolist()
search_funnel = df.query('user_id not in @tips_click_users_list')
search_funnel = search_funnel.query('user_id in @search_users_list')
search_funnel =  search_funnel.query('event_name == "search" or event_name == "advert_open" or
search_funnel = search_funnel.groupby('event_name').agg({'user_id': 'nunique'}).reset_index()
search_funnel.sort_values(by='user_id')
```

Out[48]:

	event_name	user_id
1	contacts_call	112
3	favorites_add	194
0	advert_open	305
2	contacts_show	347
4	search	1585

In [49]:
```
data = dict(values=[1585,347,305,194,112],
            labels=['search', 'contacts_show','advert_open','favorites_add',
                    'contacts_call'])
fig = go.Figure(go.Funnel(y=['search', 'contacts_show','advert_open','favorites_add',
                    'contacts_call'], x=[1585,347,305,194,112],textposition = "inside",
    textinfo = "value+percent initial+percent previous",
    opacity = 0.65, marker = {"color": ["deepskyblue","lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}}))
fig.update_layout(title='Воронка взаимодействий пользователей, кто воспользовался только поиско
fig.show()
```



Вывод:

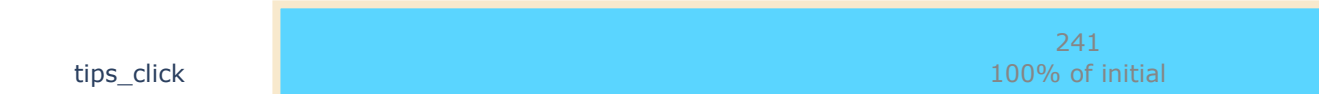
- После поиска по сайту 22% пользователей открывают карточку объявлений
- Из открывших 88% смотрит номер телефона,это 21% от общего числа пользователей
- 64% добавляют контакт в избранное,хороший результат.Пользователи находят нужное
- 58% из них звонят
- Но от общего числа пользователей это всего 7%

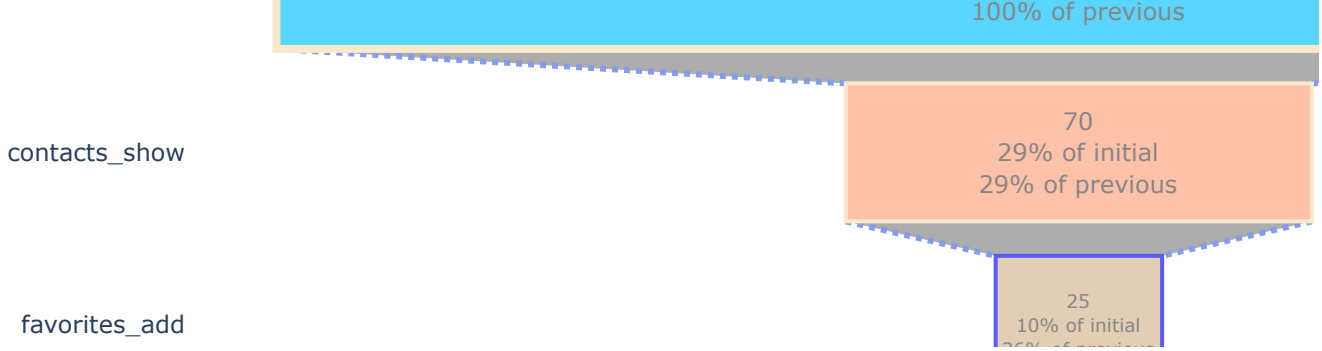
Сравним конверсию воронки взаимодействия, с основными карточками, с помощью tips_click

Сформируем датафрейм, в котором отследим конверсию в целевое contacts_show, а так же contacts_call,tips_click,advert_open,photos_show и favorites_add

In [50]:
```
data = dict(values=[241,70,25,1],
            labels=['tips_click','contacts_show','favorites_add',
                    'photos_show'])

fig = go.Figure(go.Funnel(y=['tips_click','contacts_show','favorites_add',
                     'photos_show'], x=[241,70,25,1],
                 textposition = "inside",
    textinfo = "value+percent initial+percent previous",
    opacity = 0.65, marker = {"color": ["deepskyblue", "lightsalmon", "tan", "teal", "silver"],
    "line": {"width": [4, 2, 2, 3, 1, 1], "color": ["wheat", "wheat", "blue", "wheat", "wheat"]
    connector = {"line": {"color": "royalblue", "dash": "dot", "width": 3}})
    )
fig.update_layout(title='Воронка взаимодействий пользователей, кто воспользовался только поиско
fig.show()
```



Вывод:

```
  - Только 29% смотрят контакты из кликнувших по рекомендованному объявлению
  - Затем 36% добавляют контакты в избранное
  - Практически никто - 4% не смотрит фото(непонятно)
  - У пользователей кто взаимодействовал с карточкой tips_click нет
взаимодейсвтия с карточкой contacts_call
  - Очевидно, общаются на сайте
```

Общий вывод по воронкам:

- Пользователи предпочитают самостоятельный поиск на сайте
- Они редко кликают на рекомендованные объявления, кликнув, в 29% смотрят контакты.
- 36% от просмотренных контактов добавляют в избранное
- У действий с помощью поиска на сайте лучше конверсия

Посмотрим,хорошо ли влияют события advert_open, photos_show и tips_click на целевое событие

In [51]:
```
mean_events_per_day = df.groupby(['day', 'event_name'])['date_time'].nunique().reset_index()
mean_events_per_day.columns = ['day', 'event_name', 'n_events']
mean_events_per_day = mean_events_per_day.groupby('event_name')['n_events'].agg(['median', 'mea
mean_events_per_day = mean_events_per_day.sort_values(by='mean', ascending=False).reset_index()

print('Медианное и Среднее кол-во событий за день и кол-во всех событий за весь период\n')

n_events = df['event_name'].value_counts().reset_index() # Общее кол-во каждого события
n_events.columns = ['event_name', 'n_events']

print(mean_events_per_day.merge(n_events, on='event_name'))
```

```
Медианное и Среднее кол-во событий за день и кол-во всех событий за весь период

      event_name  median    mean  n_events
0      tips_show  1423.0  1406.4     40055
1    photos_show   319.5   332.6     10012
2         search   236.0   238.3      6784
3    advert_open   225.0   219.1      6164
4  contacts_show   168.0   156.2      4529
5            map   133.0   133.8      3881
6  favorites_add    44.0    50.5      1417
```

```
7     tips_click    27.0    29.0       814
8  contacts_call    18.5    19.1       541
```

Среднее мало отличается от медианы. А также пропорционально кол-вам всем событий за весь период. Самые частые события - "tips_show" и "photos_show". Самые редкие - tips_click и contacts_call

Посчитаем общую конверсию всех событий в Целевое - "contacts_show"

In [52]:
```
no_contacts_show = df[df['event_name'] !='contacts_show']['user_id']. nunique()
contacts_show = df[df['event_name'] =='contacts_show']['user_id']. nunique()
print(f"Кол-во всех событий, кроме contacts_show =", no_contacts_show)
print(f"Общая конверсия всех событий в Целевое - 'contacts_show' = {round((contacts_show/no_cor
```

```
Кол-во всех событий, кроме contacts_show = 4260
Общая конверсия всех событий в Целевое - 'contacts_show' = 23.03%
```

Конверсия advert_open,photos_show и tips_click в ЦС

Общее количество пользователей, совершивших просмотр фото

In [53]:
```
photos_show = df.query('event_name =="photos_show"')['user_id'].unique().tolist()
print('Кол-во пользователей все "photos_show" =', len(photos_show))
```

```
Кол-во пользователей все "photos_show" = 1095
```

Количество пользователей, совершивших просмотр фото и пеоешедших в просмотр контактов

In [54]:
```
df_photos_show = df.query('user_id in @photos_show')
df_photos_show[df_photos_show['event_name'] == 'contacts_show']['user_id'].nunique()
df_photos_show_b = df_photos_show[df_photos_show['event_name'] == 'contacts_show']['user_id'].r
df_photos_show_b
```

Out[54]:
```
339
```

Конверсия photos_show в ЦС

In [55]:
```
print(f'Конверсия photos_show в ЦС  = {round(df_photos_show_b/len(photos_show)*100)}%')
```

```
Конверсия photos_show в ЦС  = 31%
```

Общее количество пользователей,открывших карточку объявления

In [56]:
```
advert_open = df.query('event_name =="advert_open"')['user_id'].unique().tolist()
print('Кол-во пользователей все "advert_open" =', len(advert_open))
```

```
Кол-во пользователей все "advert_open" = 751
```

Количество пользователей, открывших карточку объявлений и перешедших в просмотр контактов

In [57]:
```
df_advert_open = df.query('user_id in @advert_open')
df_advert_open[df_advert_open['event_name'] == 'contacts_show']['user_id'].nunique()
df_advert_open_b = df_advert_open[df_advert_open['event_name'] == 'contacts_show']['user_id'].r
df_advert_open_b
```

Out[57]:
```
138
```

Конверсия advert_open в ЦС

In [58]:
```
print(f'Конверсия advert_open в ЦС  = {round(df_advert_open_b/len(advert_open)*100)}%')
```

```
Конверсия advert_open в ЦС  = 18%
```

Общее количество пользователей,кликнувших по рекомендованному объявлению

In [59]:
```
tips_click = df.query('event_name =="tips_click"')['user_id'].unique().tolist()
print('Кол-во пользователей все "tips_click" =', len(tips_click))
```

```
Кол-во пользователей все "tips_click" = 322
```

Количество пользователей, кликнувших по объявлению и перешедших в просмотр контактов

In [60]:
```
df_tips_click = df.query('user_id in @tips_click')
df_tips_click[df_tips_click['event_name'] == 'contacts_show']['user_id'].nunique()
df_tips_click_b = df_tips_click[df_tips_click['event_name'] == 'contacts_show']['user_id'].nuni
df_tips_click_b
```

Out[60]:
```
100
```

Конверсия tips_click в ЦС

In [61]:
```
print(f'Конверсия tips_click в ЦС  = {round(df_tips_click_b/len(tips_click)*100)}%')
```

```
Конверсия tips_click в ЦС  = 31%
```

Вывод:

```
- Общая конверсия всех событий 'contacts_show' - 23%
- 18% пользователей доходят от открытия карточки до просмотра контактов
- 31% пользователей доходят от просмотра фото до просмотра контактов
- 31% пользователей доходят от клика по рекомендованному объявлению до
просмотра контактов
```

Рассчет среднего времени от вспомогательного события до Целевого

Создадим таблицу пользователей, где присутствуют дата первого события и дата первого Целевого (первого, так как возможно каждый пользователь мог совершать несколько целевых событий)

In [62]:
```
# найдем первое целевое событие
user_first_target = df.query('event_name == "contacts_show"')
df_target = user_first_target.groupby(['user_id', 'even_time']).agg({'date_time':'min'})
df_target.head()
```

Out[62]:

		date_time
user_id	**even_time**	
00157779-810c-4498-9e05-a1e9e3cedf93	**2019-10-20**	2019-10-20 19:17:18
	2019-10-29	2019-10-29 21:26:40
	2019-10-30	2019-10-30 08:01:05
	2019-11-03	2019-11-03 17:12:09
00551e79-152e-4441-9cf7-565d7eb04090	**2019-10-25**	2019-10-25 16:44:41

In [63]:
```
# первое совершенное событие кроме целевого
source_time_before = df.query('event_name not in "contacts_show"')
df_before = source_time_before.groupby(['user_id', 'even_time']).agg({'date_time':'min'})
df_before.head()
```

Out[63]:

		date_time
user_id	**even_time**	
0001b1d5-b74a-4cbf-aeb0-7df5947bf349	**2019-10-07**	2019-10-07 13:39:45
	2019-10-09	2019-10-09 18:33:55
	2019-10-21	2019-10-21 19:52:30
	2019-10-22	2019-10-22 11:18:14

		date_time
user_id	**even_time**	
00157779-810c-4498-9e05-a1e9e3cedf93	**2019-10-19**	2019-10-19 21:34:33

In [64]:
```
# соединим 2 фрейма и найдем разницу в среднем по всем дням
diff = df_target.merge(df_before, on=['user_id', 'even_time'], how='left')
diff.head()
```

Out[64]:

		date_time_x	date_time_y
user_id	**even_time**		
00157779-810c-4498-9e05-a1e9e3cedf93	**2019-10-20**	2019-10-20 19:17:18	2019-10-20 18:49:24
	2019-10-29	2019-10-29 21:26:40	2019-10-29 21:18:24
	2019-10-30	2019-10-30 08:01:05	2019-10-30 07:50:45
	2019-11-03	2019-11-03 17:12:09	NaT
00551e79-152e-4441-9cf7-565d7eb04090	**2019-10-25**	2019-10-25 16:44:41	2019-10-25 16:44:44

In [65]:
```
diff['diff'] = diff['date_time_x'] - diff['date_time_y']
print(diff['diff'].mean())
print(diff['diff'].median())
```

```
0 days 00:46:04.890813253
0 days 00:04:42.500000
```

In [66]:
```
display(diff)
diff['diff'].describe()
```

		date_time_x	date_time_y	diff
user_id	**even_time**			
00157779-810c-4498-9e05-a1e9e3cedf93	**2019-10-20**	2019-10-20 19:17:18	2019-10-20 18:49:24	0 days 00:27:54
	2019-10-29	2019-10-29 21:26:40	2019-10-29 21:18:24	0 days 00:08:16
	2019-10-30	2019-10-30 08:01:05	2019-10-30 07:50:45	0 days 00:10:20
	2019-11-03	2019-11-03 17:12:09	NaT	NaT
00551e79-152e-4441-9cf7-565d7eb04090	**2019-10-25**	2019-10-25 16:44:41	2019-10-25 16:44:44	-1 days +23:59:57
...	**...**	...	...	...
fffb9e79-b927-4dbb-9b48-7fd09b23a62b	**2019-10-28**	2019-10-28 15:00:42	2019-10-28 11:49:38	0 days 03:11:04
	2019-10-29	2019-10-29 14:00:14	2019-10-29 13:58:47	0 days 00:01:27
	2019-10-30	2019-10-30 00:15:43	2019-10-30 00:15:43	0 days 00:00:00
	2019-11-02	2019-11-02 18:17:41	2019-11-02 01:16:48	0 days 17:00:53
	2019-11-03	2019-11-03 14:33:47	2019-11-03 14:32:55	0 days 00:00:52

1444 rows × 3 columns

Out[66]:
```
count                         1328
mean     0 days 00:46:04.890813253
std      0 days 02:27:50.808445097
min              -1 days +05:19:49
25%         0 days 00:00:32.750000
50%         0 days 00:04:42.500000
75%         0 days 00:23:48.500000
max               0 days 23:06:13
Name: diff, dtype: object
```

In [67]:
```
diff.isna().sum()
```

Out[67]:
```
date_time_x      0
date_time_y    116
diff           116
dtype: int64
```

У нас есть пропуски, 9%. Для чистоты расчета от них следует избавиться. Заменим на медианное.

In [68]:
```
diff = diff.fillna(diff.median())
```

Построим гистограмму

Переведем время в секунды и разделим на 60 (будет время в минутах)

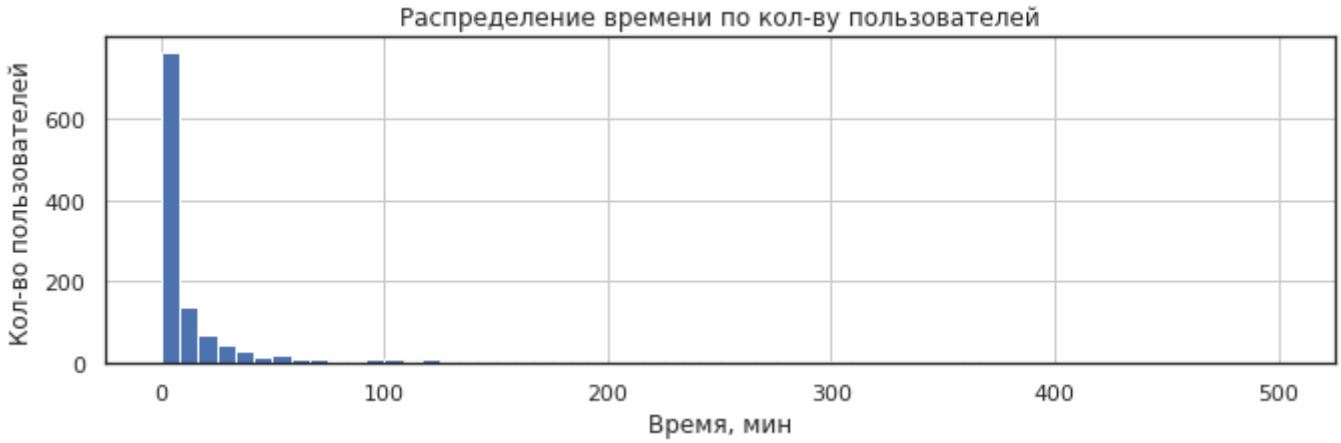
In [69]:
```
delta_minutes = round(diff['diff'].dt.total_seconds()/60)
delta_minutes
```

Out[69]:
```
user_id                               even_time
00157779-810c-4498-9e05-a1e9e3cedf93  2019-10-20      28.0
                                      2019-10-29       8.0
                                      2019-10-30      10.0
                                      2019-11-03       5.0
00551e79-152e-4441-9cf7-565d7eb04090  2019-10-25      -0.0
                                                      ...
fffb9e79-b927-4dbb-9b48-7fd09b23a62b  2019-10-28     191.0
                                      2019-10-29       1.0
                                      2019-10-30       0.0
                                      2019-11-02    1021.0
                                      2019-11-03       1.0
Name: diff, Length: 1444, dtype: float64
```

In [70]:
```
delta_minutes.describe()
```

Out[70]:
```
count    1444.000000
mean       42.781163
std       142.221005
min     -1120.000000
25%         1.000000
50%         5.000000
75%        20.000000
max      1386.000000
Name: diff, dtype: float64
```

In [71]:
```
delta_minutes.hist(figsize=(11,3),bins=60, range=(0,500));
plt.title('Распределение времени по кол-ву пользователей')
plt.ylabel('Кол-во пользователей')
plt.xlabel('Время, мин');
plt.show()
```



Вывод:

- Медианное время 5 минут
- среднее 42 минуты
- у большинства пользователей среднее время от вспомогательного события до Целевого составило от 1 минуты до 20 мин
- мах - почти сутки.

Вывод

- между ЦС и предыдущим проходит от 42 сек до 12 минут
- самое быстрое - открыие карточки
- самое медленное - клик по объявлению

Рассчет динамики количества взаимодействий в приложении по дням и неделям (DAU\WAU)

In [72]:
```
count_users_by_day = user_activity.groupby('even_time').agg({'user_id':'count'}).reset_index()
count_users_by_week = user_activity.groupby('activity_week').agg({'user_id':'count'}).reset_inc


fig = go.Figure(data=[
    go.Bar(x=count_users_by_day['even_time'].to_list(),
           y=count_users_by_day['user_id'].to_list(),
           text=count_users_by_day['user_id'].to_list(),
           textposition='auto')])
fig.update_layout(title='График количества взаимодействий пользователей по дням')
fig.show()
```

Вывод:

- По графику количества взаимодействий пользователей по дням видно, что динамика меняется по дням недели - в выходные спад
  - В целом, динамика имеет ровный характер
  - Максимальное число пользователей 23.10 - 3361, мин 12.10 - 1843
  - Воронка по неделям:
    - Во вторую неделю пришло на 16%% больше пользователей чем в первую
    - В третью неделю пришло на 13% больше пользователей чем во вторую и на 30% больше чем в первую
    - В четвертую неделю пришло на 6% меньше пользователей чем в третью и на 23% меньше чем в первую

Общий вывод по анализу событий

- Вместе с целевым событием (contacts_show \show_contacts) пользователи часто совершают:
  - просмотр рекомендованного объяления(tips_show)- 40055
  - просмотр фото(photos_show) - 10012
  - поиск на сайте(search) - 6784
- Одной воронки нет, есть несколько локальных:
  - поиск на сайте - открыл карточку объявлений - посмотрел фото - посмотрел номер телефона - позвонил
  - увидел рекомендованное объявление -кликнул - открыл карточку объявлений-- посмотрел фото - посмотрел номер телефона - позвонил
- между ЦС и предыдущим проходит от 42 сек до 12 минут
  - самое быстрое - открыие карточки
  - самое медленное - клик по объявлению
- в день пользователь совершаетсобытий - 9.0
- у большинства пользователей среднее время от вспомогательного события до Целевого составило от 1 минуты до 20 мин

Проверка статистических гипотез

Различается ли конверсия в просмотры контактов tips_show+tips_click и tips_show

Одни пользователи совершают действия tips_show и tips_click, другие — только tips_show. Проверим гипотезу: конверсия в просмотры контактов различается у этих двух групп.

Выделим всех пользователей, кто совершил "tips_show", а затем разделим их на 2 группы - кто совершал "tips_click" и кто нет.

In [73]:
```
tips_show_users = df.query('event_name =="tips_show"')['user_id'].unique().tolist()
print('Кол-во пользователей все "tips_show" =', len(tips_show_users))

tips_show_click_users = df.query('event_name =="tips_click" and user_id==@tips_show_users')['us
print('Кол-во пользователей "tips_show"+"tips_click" =', len(tips_show_click_users))
```

```
only_tips_show_users = list(set(tips_show_users) - set(tips_show_click_users))
print('Кол-во пользователей только "tips_show" =', len(only_tips_show_users))
```

```
Кол-во пользователей все "tips_show" = 2801
Кол-во пользователей "tips_show"+"tips_click" = 297
Кол-во пользователей только "tips_show" = 2504
```

Подсчитаем конверсию в "contacts_show" всех действий каждой группы

Количество пользователей из группы "только tips_show", совершивших конверсию в ЦС

In [74]:
```
df_tips_show = df.query('user_id in @only_tips_show_users')
df_tips_show[df_tips_show['event_name'] == 'contacts_show']['user_id'].nunique()
df_tips_b = df_tips_show[df_tips_show['event_name'] == 'contacts_show']['user_id'].nunique()
df_tips_b
```

Out[74]: 425

Количество пользователей из группы "tips_show"+"tips_click", совершивших конверсию в ЦС

In [75]:
```
df_tips_show_1 = df.query('user_id in @tips_show_click_users')
df_tips_show_1[df_tips_show_1['event_name'] == 'contacts_show']['user_id'].nunique()
df_tips_a = df_tips_show_1[df_tips_show_1['event_name'] == 'contacts_show']['user_id'].nunique(
df_tips_a
```

Out[75]: 91

- Проверка гипотезы
- Для проверки гипотезы нам подходит метод - проверка гипотезы о равенстве долей
- Соберем данные для теста:
  - Кол-во всего пользователей у tips_show+click пользователей = len(only_tips_show_users))
  - Кол-во ЦС у tips_show+click пользователей = df_tips_a
  - Кол-во всего событий у только tips_show пользователей = len(tips_show_click_users)
  - Кол-во ЦС у только tips_show пользователей = df_tips_b
  - Теперь подставим в тест и сравним доли клиентов, совершивших ЦС (Нулевая гипотеза - между долями значимая разница отсутствует. Альтернативная - разница есть; критический уровень статистической значимости возьмем стандартный равный 5%)
- Напишем функцию stat_test(successes, trials, alpha), которая проводит тестирование на равенство долей, где:
  - successes - кол-во успешных попыток [группа1, группа2]
  - trials - всего попыток [группа1, группа2]
  - alpha - критический уровень статистической значимости

In [76]:
```
def stat_test(successes, trials, alpha):
    alpha = alpha
    successes = successes
    trials = trials

    # пропорция успехов в первой группе:
    p1 = successes[0]/trials[0]

    # пропорция успехов во второй группе:
    p2 = successes[1]/trials[1]

    # пропорция успехов в комбинированном датасете:
    p_combined = (successes[0] + successes[1]) / (trials[0] + trials[1])

    # разница пропорций в датасетах
```

```
    difference = p1 - p2


    # считаем статистику в ст.отклонениях стандартного нормального распределения
    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1/trials[0] + 1/trials[1]))

    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)


    # считаем статистику в ст.отклонениях стандартного нормального распределения
    z_value = difference / mth.sqrt(
        p_combined * (1 - p_combined) * (1 / trials[0] + 1 / trials[1])
    )

    # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
    distr = st.norm(0, 1)

    p_value = (1 - distr.cdf(abs(z_value))) * 2

    print('p-значение: ', p_value)

    if p_value < alpha:
        print('Отвергаем нулевую гипотезу: между долями есть значимая разница')
    else:
        print(
            'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'
        )


alpha = .05

successes = [df_tips_b,df_tips_a]

trials = [len(only_tips_show_users),len(tips_show_click_users)]

stat_test(successes, trials, alpha)
print(trials)
print(successes)
```

```
p-значение:  9.218316554537864e-09
Отвергаем нулевую гипотезу: между долями есть значимая разница
[2504, 297]
[425, 91]
```

Вывод

- Конверсия в просмотры контактов у двух групп пользователей (в первой совершают действия tips_show и tips_click, во второй — только tips_show) - различается

Своя гипотеза

Гипотеза - Пользователи совершают действия "search" и "photos_show". Проверим гипотезу: Чистая Конверсия в просмотры контактов различается у этих двух событий.

In [77]:
```
search_users = df.query('event_name =="search"')['user_id'].unique().tolist()
print('Кол-во пользователей все "search" =', len(search_users))

photos_show = df.query('event_name =="photos_show"')['user_id'].unique().tolist()
print('Кол-во пользователей все "photos_show" =', len(photos_show))
```

```
Кол-во пользователей все "search" = 1666
Кол-во пользователей все "photos_show" = 1095
```

Количество пользователей из группы "search", совершивших конверсию в ЦС

In [78]:
```
df_search = df.query('user_id in @search_users')
df_search[df_search['event_name'] == 'contacts_show']['user_id'].nunique()
```

```
df_search_b = df_search[df_search['event_name'] == 'contacts_show']['user_id'].nunique()
df_search_b
```

Out[78]: 377

Количество пользователей из группы "photos_show", совершивших конверсию в ЦС

In [79]:
```
df_photos_show = df.query('user_id in @photos_show')
df_photos_show[df_photos_show['event_name'] == 'contacts_show']['user_id'].nunique()
df_photos_show_b = df_photos_show[df_photos_show['event_name'] == 'contacts_show']['user_id'].r
df_photos_show_b
```

Out[79]: 339

- Для проверки гипотезы нам подходит метод - проверка гипотезы о равенстве долей
- Соберем данные для теста:
  - Кол-во всего пользователей,совершивших "photos_show" = len(photos_show)
  - Кол-во ЦС у "photos_show" пользователей = df_photos_show_b
  - Кол-во всего пользователей,совершивших "search" = len(search_users)
  - Кол-во пользователей ЦС "search" = df_search_b
  - Теперь подставим в тест и сравним доли клиентов, совершивших ЦС (Нулевая гипотеза - между долями значимая разница отсутствует. Альтернативная - разница есть; критический уровень статистической значимости возьмем стандартный равный 5%)

In [80]:
```
alpha = .05
successes = [df_photos_show_b,df_search_b ]
trials = [len(photos_show),len(search_users)]

stat_test(successes, trials, alpha)
```

```
p-значение:  1.0314969967062382e-06
Отвергаем нулевую гипотезу: между долями есть значимая разница
```

Вывод

- Конверсия в ЦС у пользователей (тех кто совершают действия "photos_show" , и тех кто — "search") - имеет статистические различия

Выводы:

- Мы располагаем данными с 7 октября (00:00) по 3 ноября (23:58) 2019 года. Период времени почти 28 дней
- Кол-во пользователей = 4293
- Кол-во событий = 74197
- Кол-во событий которое совершает один пользователь за день - 9
- В среднем, пользователь совершает от 1 до 10 просмотров контактов,медианное -5. Max - 336.
- Самые частые события - tips_show (более 40000) и photos_show(более 10000).
- Самые редкие - contacts_call(541) и tips_click (814)
- Retention Rate низкий (24%),в первую неделю убывает по когортам с течением времени
- Пользователи предпочитают самостоятельный поиск на сайте
- Они редко кликают на рекомендованные объявления, хотя кликнув, в 83% смотрят контакты. Это говорит о хорошей системе рекомендованных объявлений
- 23% от просмотренных контактов добавляют в избранное

- У действий с помощью поиска на сайте лучше конверсия
- Очень мало кликнувших по рекомендованному объявлению, конверсия низкая
- Так же очень мало добавивших в избранное и позвонивших
- У пользователей кто взаимодействовал с карточкой tips_click нет взаимодейсвтия с карточкой contacts_call
- между ЦС и предыдущим проходит от 42 сек до 12 минут
- самое быстрое - открыие карточки
- самое медленное - клик по объявлению
- В целом, динамика взаимодействий пользователей по дням имеет ровный характер
- Конверсия в просмотры контактов у двух групп пользователей (в первой совершают действия tips_show и tips_click, во второй — только tips_show) - различается
- Конверсия в ЦС у пользователей (тех кто совершают действия "photos_show" , и тех кто — "search") - имеет статистические различия

Рекомендации

У данного сайта низкий коэффициент удержания. Как один из вариантов его поднять - Email рассылки.
Email рассылки

mail рассылки — один из популярных способов напомнить о своем бренде и привлечь покупателя скидками и акциями, а также увеличить его лояльность за счет регулярных полезных рассылок, писем из бонусных программ и поздравительных сообщений.

Email рассылки решают задачи:

Информирования клиента о продукции, скидках, бонусах и программах лояльности. Налаживания обратной связи с потребителями. Поддержания вовлеченности пользователей. Сбора данных о потребителей, чтобы создать персонализированные рассылки и конвертировать лиды в продажи.