Contents

In [1]:
```
import numpy as np
import pandas as pd

from scipy import polyval, stats
import math
from statsmodels.graphics.mosaicplot import mosaic
import statsmodels.formula.api as sf
import statsmodels.api as sm

from scipy.stats import normaltest  #для проверки распределения на нормальность

from scipy.stats import spearmanr  #корреляция Спирмена

import seaborn as sns

sns.set()
import matplotlib.pyplot as plt

from collections import Counter

from datetime import timedelta
from sqlalchemy import create_engine
```

executed in 832ms, finished 23:29:49 2021-06-07

In [2]:
```
# настройки отображения графиков
%matplotlib inline
plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = (15, 5)
plt.rcParams['font.family'] = 'sans-serif'
```

executed in 9ms, finished 23:29:49 2021-06-07

1 Посчитайте различные KPI метрики игры: DAU, ARPU, ARPPU, Conversion.

```
DAU (Daily Active Users) - число уникальных пользователей в день

ARPU (Average Revenue Per User)- средний доход с пользователя
       определяется путем соотношения брутто-дохода от пользователей к среднему показателю посещаемости
       в день/неделю/месяц.
       ARPU = Revenue / Users (чистый доход «поделить» на количество всех пользователей)

       +Paying Share = доля платящих пользователей
       ARPU = ARPPU × Paying Share

ARPPU (Average Revenue Per Paying User)- доход с одного платящего пользователя
       брутто-доход соотносим с количеством платящих пользователей или PU
       (число клиентов, оплативших дополнительный контент в конкретный период).
       ARPPU всегда будет больше APRU.


Conversion -Конверсия (какая доля посетителей сайта совершила покупку)
    Conversion rate = общее количество покупок / общее количество уникальных посетителей * 100
```

Contents

In [3]:
```
engine = create_engine('postgresql://postgres:@192.168.0.165/Playgendary')
# engine = create_engine('postgresql://tuser:faexoh9A@htstmysql35.plg.dev/test')

conn = engine.connect()
```

executed in 53ms, finished 23:29:49 2021-06-07

In [4]:
```
# количество уникальных игроков за месяц
total_users = pd.read_sql(
    """select count(distinct user_id) uniq_user_id from test.events_data""",
    conn)
total_users
```

executed in 1m 10.2s, finished 23:30:59 2021-06-07

Out[4]:

	uniq_user_id
0	505034

In [5]:
```
# DAU (Daily Active Users) - число уникальных пользователей в день

uniq_users_everyday = pd.read_sql(
    """select date(data_event), count(distinct user_id) DAU from
    (select *, to_timestamp(floor(event_timestamp/1000000)) data_event , max (event_id) meid
    from test.events_data group by 1,2,3,4) uniqid
    group by 1""", conn)
```

executed in 1m 29.0s, finished 23:32:28 2021-06-07

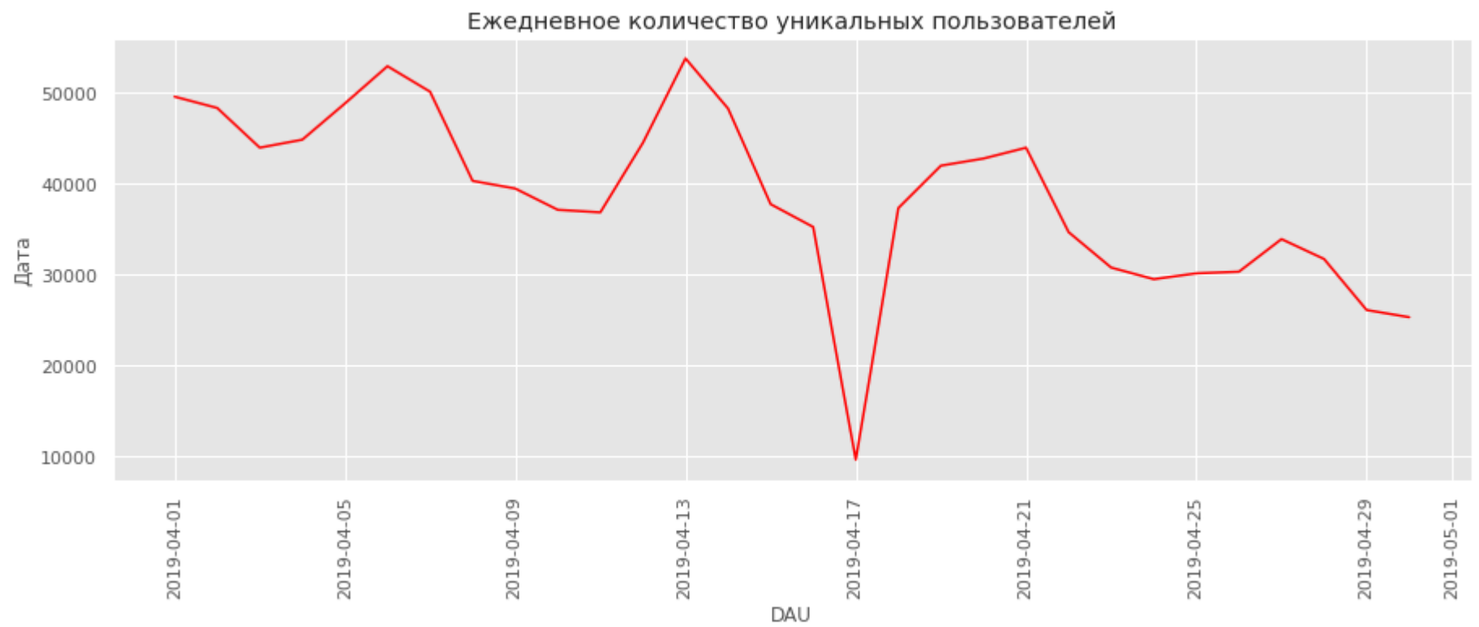
In [6]:
```
uniq_users_everyday.shape
```

executed in 2ms, finished 23:32:28 2021-06-07

Out[6]: (30, 2)

In [7]:
```
# DAU (Daily Active Users) - число уникальных пользователей в день
plt.xlabel('DAU')
plt.ylabel('Дата')
plt.title('Ежедневное количество уникальных пользователей')

x = uniq_users_everyday.date
y = uniq_users_everyday.dau
sns.lineplot(x=x, y=y, data=uniq_users_everyday, color='red')

plt.xticks(rotation=90);
```

executed in 326ms, finished 23:32:28 2021-06-07



Contents

In [8]:

```
#  В таблице parameters_data сведены в param_value все значения из 4х столбцов
# (param_value_int, param_value_float, param_value_double, param_value_string),
# unixtimestamp приведен к виду дата
# из parameters_data выбраны event_id по product_id
# для категорий покупок (purchase_first, purchase_second, purchase_third) назначено количество покупок = 1
# в currency_type тип валюты coins не был найден в product_id, поэтому параметр не включен

metrics_per_day = pd.read_sql(
    """with collection as (
        select event_id,
               param_key,
               param_value
        from
        (select event_id, param_key,
        coalesce (param_value_string, param_value_int, param_value_float, param_value_double) param_value
                          from (select event_id, param_key,
                                                  NULLIF (param_value_string,'') param_value_string,
                                                  NULLIF (param_value_int,'') param_value_int,
                                                  NULLIF (param_value_float,'') param_value_float,
                                                  NULLIF (param_value_double,'') param_value_double
                                from  test.parameters_data) n
        where param_key is not null) m)
select s.data_event,
        count (s.user_id) count_purchases,
        count (distinct s.user_id) count_uniq_pay_user,
        sum (s.revenue) total_revenue,
        total_count.DAU,
        sum(s.revenue)/total_count.DAU ARPU,
        sum(s.revenue)/count(distinct s.user_id) ARPPU,
        cast(count(distinct s.user_id)as float)/cast(total_count.DAU as float) paying_share,
        (cast(count(s.user_id) as float) / cast (total_count.DAU as float) *100) conversion_rate
from (
        select distinct c1.event_id,
               c2.param_value as product_id,
               coalesce (cast(c3.param_value as integer), 1 ) as quantity,
               c4.param_value as currency,
               price_game.price,
               events.user_id,
               events.data_event,
               events.event_name,
               cast((price_game.price * coalesce (cast(c3.param_value as integer), 1 ))as double precision) revenue
from collection c1
                    left join collection c2 on c1.event_id = c2.event_id and c2.param_key = 'product_id'
                    left join collection c3 on c1.event_id = c3.event_id and c3.param_key = 'quantity'
                    left join collection c4 on c1.event_id = c4.event_id and c4.param_key = 'currency'
                    left join test.prices_data price_game on c2.param_value = price_game.product_id
                    left join (select max_event_id, date(data_event) data_event, user_id, event_name from
                              (select *, to_timestamp(floor(event_timestamp/1000000)) data_event ,
                              max (event_id) max_event_id from test.events_data group by 1,2,3,4) m ) events
                    on events.max_event_id = c1.event_id
                where c1.param_key = 'product_id') s
left join (select date(data_event) data_event,
                  count(distinct user_id) DAU
           from (select *, to_timestamp(floor(event_timestamp/1000000)) data_event ,
                           max (event_id) max_event_id
           from test.events_data group by 1,2,3,4) m group by 1) total_count
        on total_count.data_event = s.data_event
group by s.data_event, total_count.dau
    """, conn)
```

executed in 3m 6s, finished 23:35:34 2021-06-07

In [9]:
```
metrics_per_day.head()
```

executed in 8ms, finished 23:35:34 2021-06-07

Out[9]:

	data_event	count_purchases	count_uniq_pay_user	total_revenue	dau	arpu	arppu	paying_share	conversion_rate
0	2019-04-01	481	163	4208.73	49563	0.084917	25.820429	0.003289	0.970482
1	2019-04-02	434	130	3880.71	48320	0.080313	29.851615	0.002690	0.898179
2	2019-04-03	248	90	2233.06	43954	0.050804	24.811778	0.002048	0.564226
3	2019-04-04	288	94	2990.65	44832	0.066708	31.815426	0.002097	0.642398
4	2019-04-05	310	101	2446.43	48813	0.050118	24.222079	0.002069	0.635077

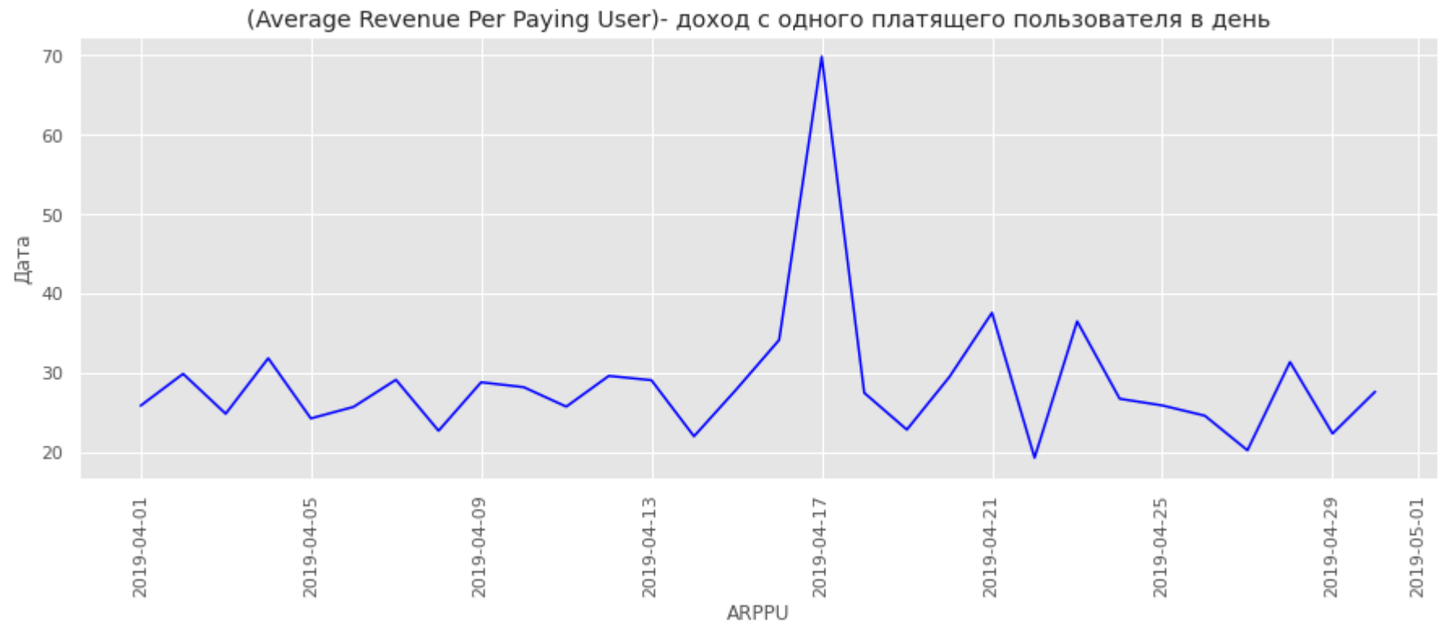
In [10]:
```
# ARPU (Average Revenue Per User)- средний доход с пользователя
plt.xlabel('ARPU')
plt.ylabel('Дата')
plt.title('(Average Revenue Per User)- средний доход с пользователя в день')

x = metrics_per_day.data_event
y = metrics_per_day.arpu
sns.lineplot(x=x, y=y, data=metrics_per_day, color='black')

plt.xticks(rotation=90);
```

executed in 241ms, finished 23:35:35 2021-06-07

Contents

In [11]:
```
# ARPPU (Average Revenue Per Paying User)- доход с одного платящего пользователя
plt.xlabel('ARPPU')
plt.ylabel('Дата')
plt.title(
    '(Average Revenue Per Paying User)- доход с одного платящего пользователя в день'
)

x = metrics_per_day.data_event
y = metrics_per_day.arppu
sns.lineplot(x=x, y=y, data=metrics_per_day, color='blue')

plt.xticks(rotation=90);
```

executed in 227ms, finished 23:35:35 2021-06-07



In [12]:
```
# Conversion -Конверсия (какая доля посетителей сайта совершила покупку)
plt.xlabel('ARPPU')
plt.ylabel('Дата')
plt.title(
    'Conversion - Конверсия (какая доля посетителей сайта совершила покупку)')

x = metrics_per_day.data_event
y = metrics_per_day.conversion_rate
sns.lineplot(x=x, y=y, data=metrics_per_day, color='green')

plt.xticks(rotation=90);
```

executed in 225ms, finished 23:35:35 2021-06-07

2 Рейтинг оружия

```
В нашей игре есть большое разнообразие оружия.
Составьте рейтинг популярности оружия среди игроков.
Аргументируйте критерии, по которым вы оцениваете популярность.
Чем эти данные могут помочь гейм-дизайнеру для развития игры.
```

Contents

In [13]:
```
# Количество активностей за месяц с разным типом оружия

pop_gun = pd.read_sql(
    """with guns as (select event_id,
                        param_value_string,
                        count (param_value_string) from test.parameters_data
            where  param_key = 'content_type' group by 1,2)
select
       event_name,
       param_value_string gun,
       count (user_id) total_use,
       count (distinct user_id) count_user
from
    (select guns.event_id,
            guns.param_value_string,
            date(total_count.data_event) data_event,
            total_count.event_name,
            total_count.user_id
from guns left join (select *,
            to_timestamp(floor(event_timestamp/1000000)) data_event, max (event_id) max_event_id from test.events_dat
        on guns.event_id = total_count.max_event_id)x group by 1,2
            """,conn)
```

executed in 1m 34.5s, finished 23:37:10 2021-06-07

In [14]:
```
pop_gun.head()
```

executed in 10ms, finished 23:37:10 2021-06-07

Out[14]:

	event_name	gun	total_use	count_user
0	match_finish	Airhorn	282422	72500
1	match_finish	AK47	435333	92448
2	match_finish	AlienBlaster	4262	688
3	match_finish	AWP	1180	363
4	match_finish	Bazooka	41045	19724

In [15]:
```
# Перечень типов активностей пользователей с использованием игрового оружия
pop_gun.event_name.unique()
```

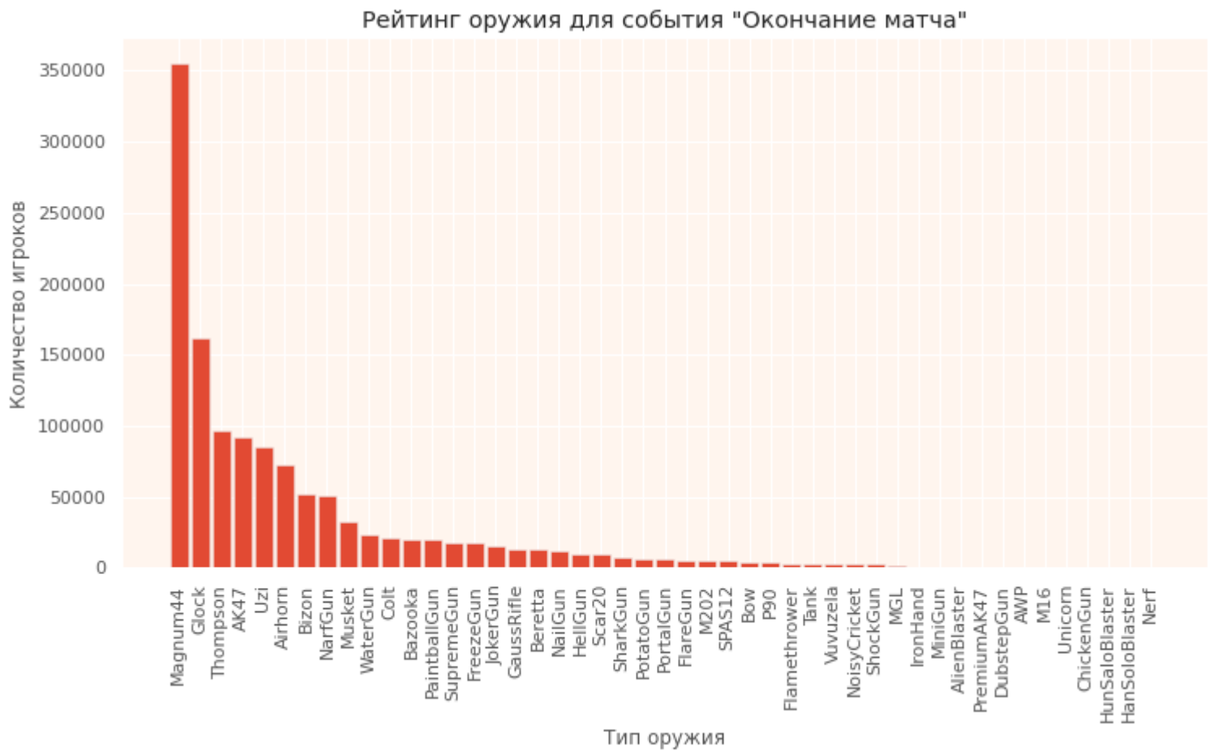
executed in 9ms, finished 23:37:10 2021-06-07

Out[15]:
```
array(['match_finish', 'match_start', 'spend_ingame_currency',
       'unlock_content', 'unlock_with_ad', 'unlock_with_bucks',
       'unlock_with_coins', 'unlock_with_daily_gift',
       'unlock_with_diamond_membership', 'unlock_with_instagram',
       'unlock_with_invites'], dtype=object)
```

Contents

In [16]:
```
# Рейтинг оружия для события "match_finish" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "match_finish"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "Окончание матча"')
plt.xticks(rotation=90)
plt.show()
```

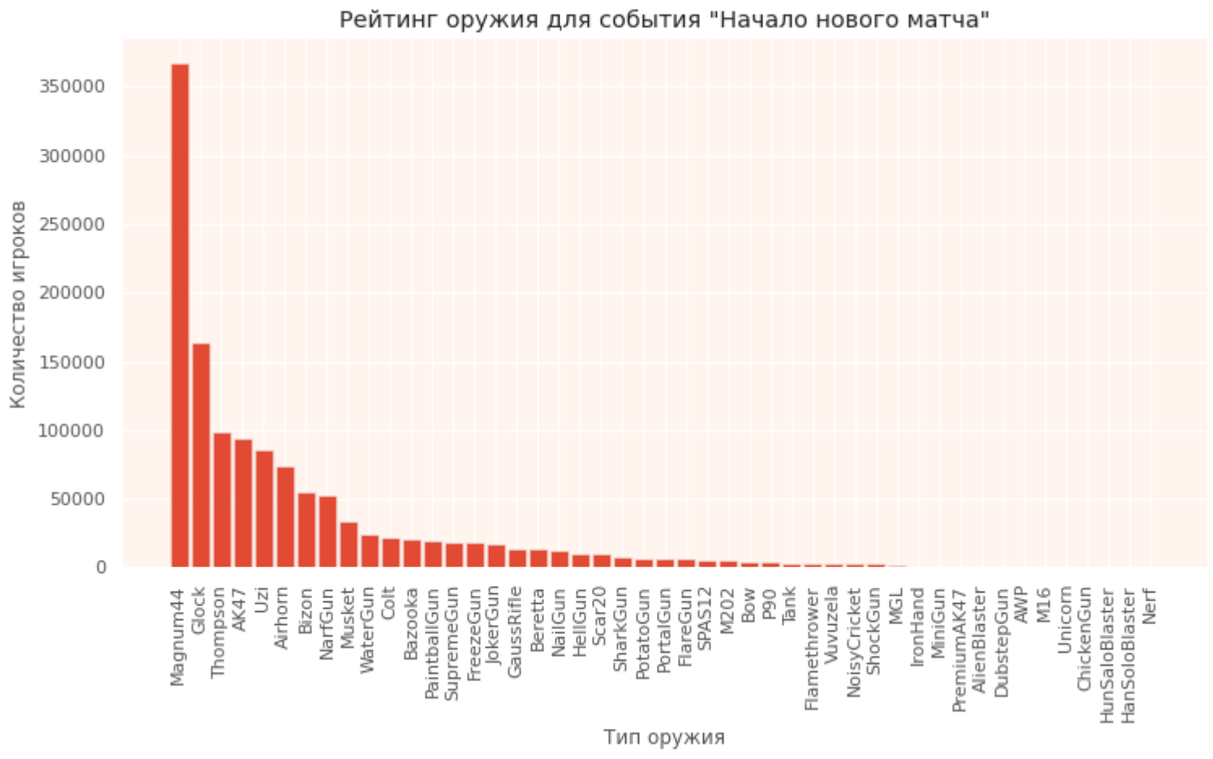
executed in 419ms, finished 23:37:10 2021-06-07



Contents

In [17]:
```
# Рейтинг оружия для события "match_start" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "match_start"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "Начало нового матча"')
plt.xticks(rotation=90)
plt.show()
```

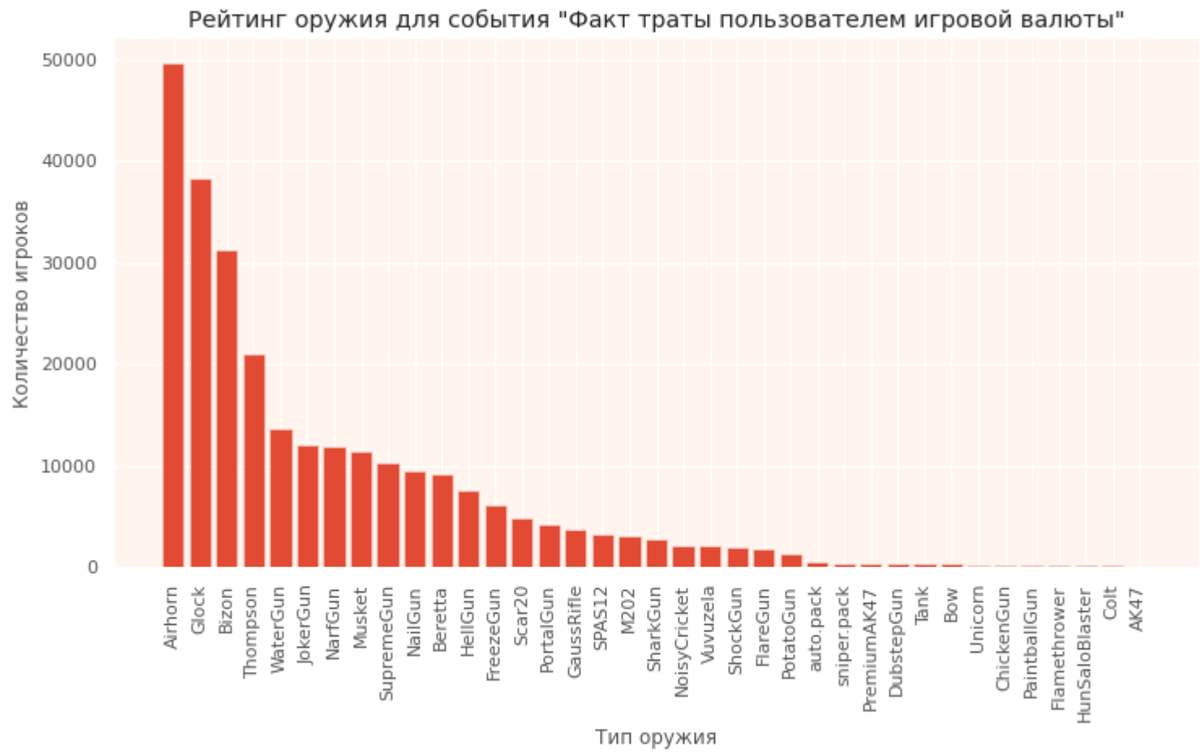
executed in 409ms, finished 23:37:11 2021-06-07



Contents

In [18]:
```
# Рейтинг оружия для события "spend_ingame_currency" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "spend_ingame_currency"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "Факт траты пользователем игровой валюты"')
plt.xticks(rotation=90)
plt.show()
```

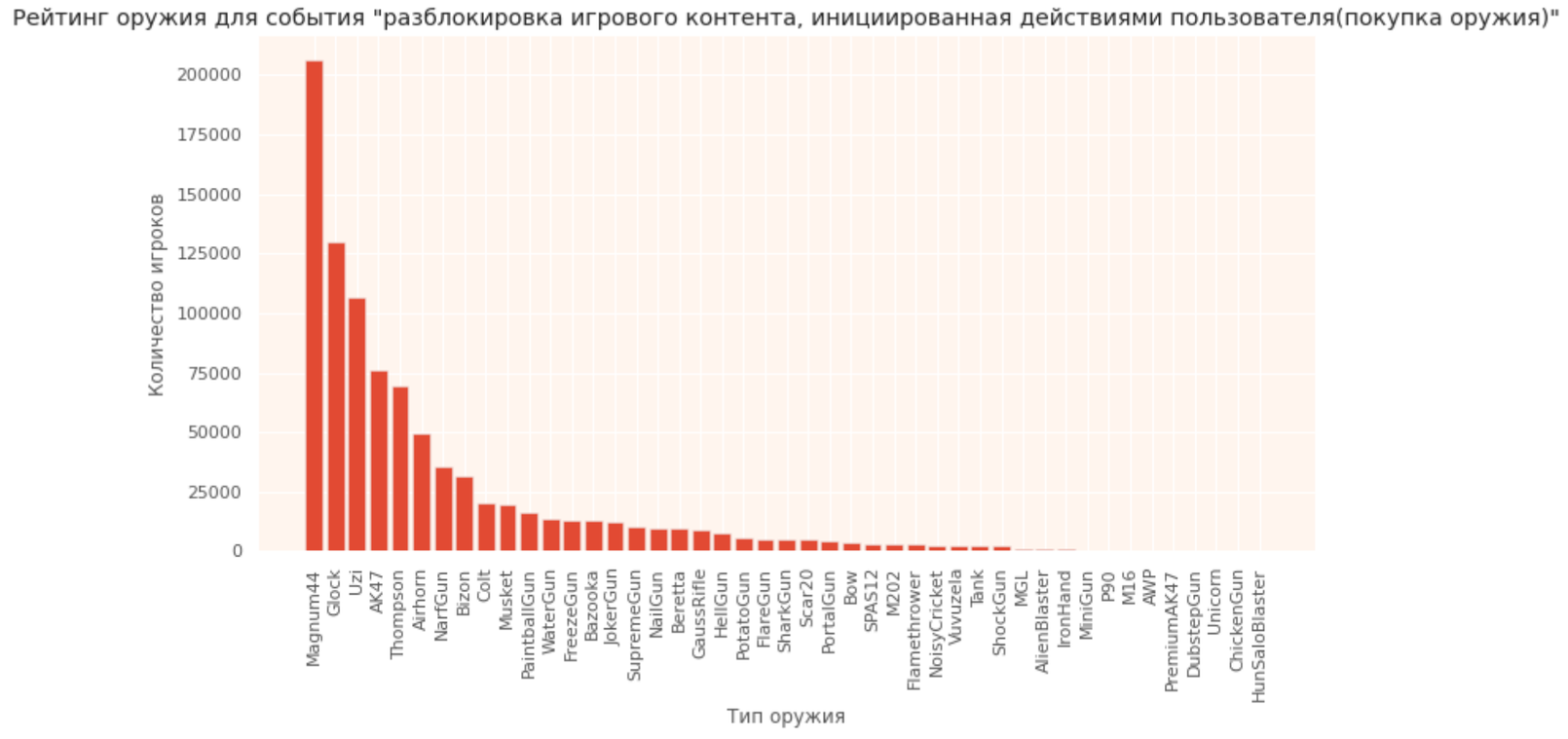
executed in 281ms, finished 23:37:11 2021-06-07



Contents

In [19]:
```
# Рейтинг оружия для события "unlock_content" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_content"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "разблокировка игрового контента, инициированная действиями пользователя(покупк
plt.xticks(rotation=90)
plt.show()
```

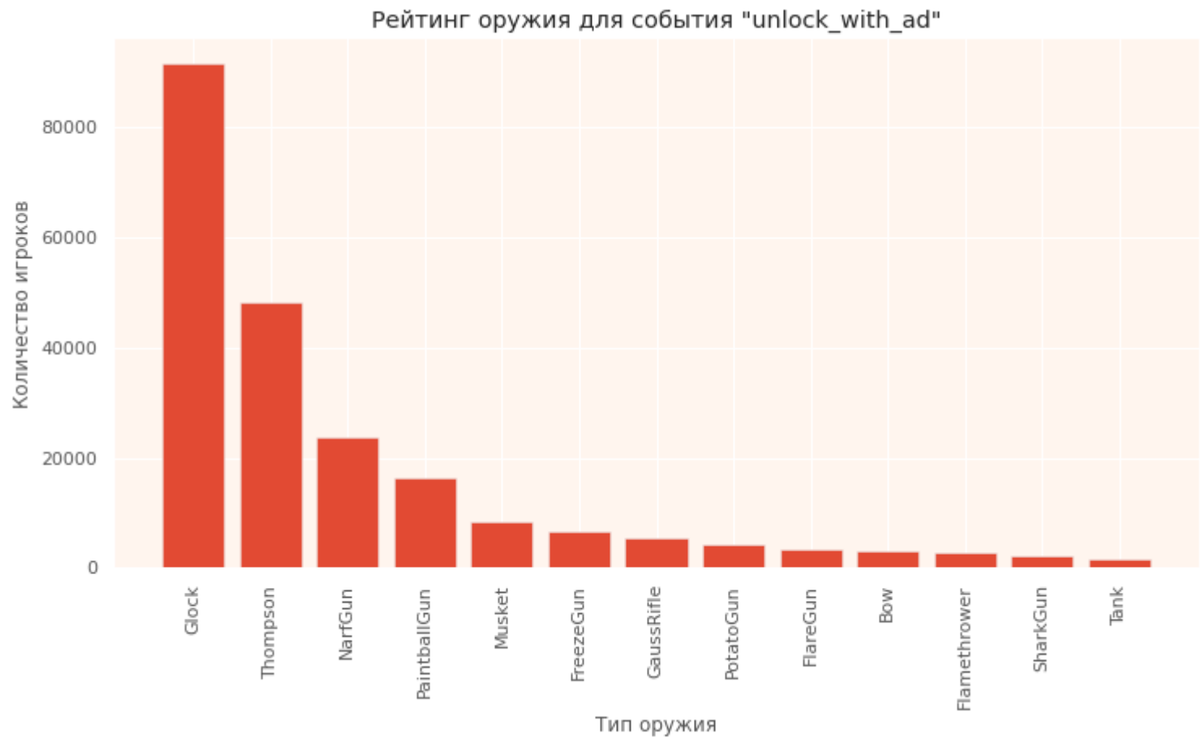
executed in 456ms, finished 23:37:11 2021-06-07



Contents

In [20]:
```
# Рейтинг оружия для события "unlock_with_ad" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_with_ad"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "unlock_with_ad"')
plt.xticks(rotation=90)
plt.show()
```

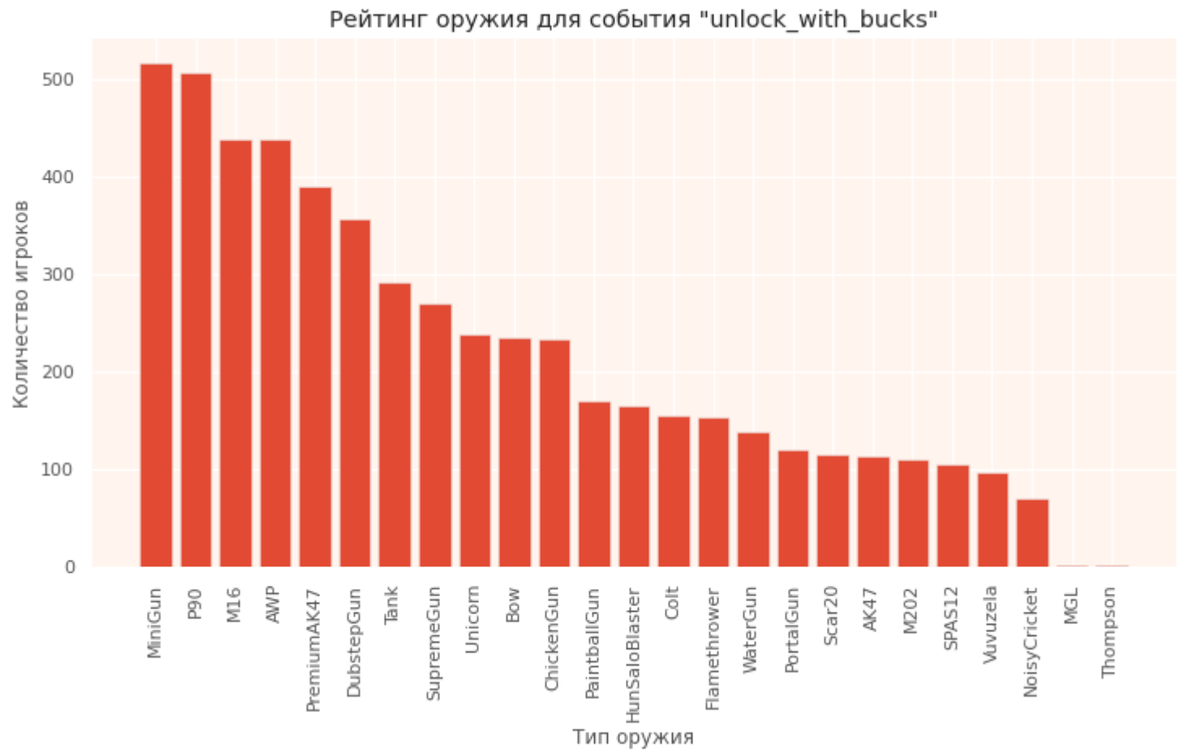
executed in 150ms, finished 23:37:11 2021-06-07



Contents

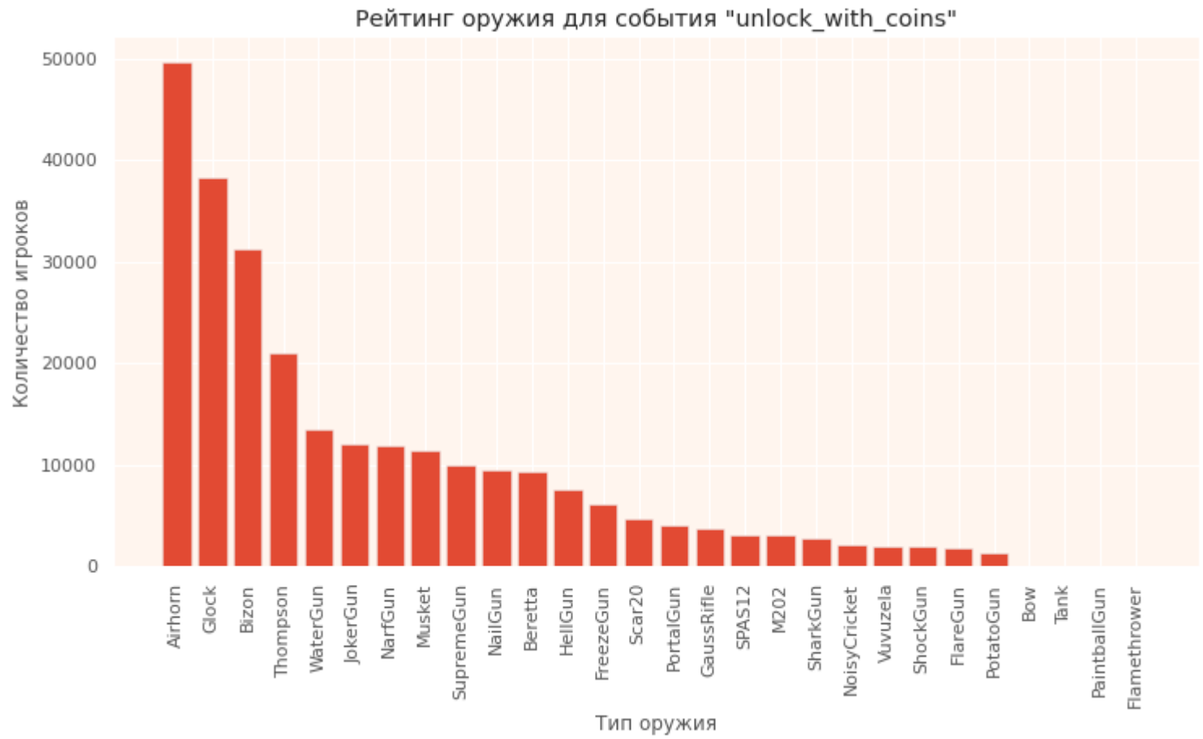
In [21]:
```
# Рейтинг оружия для события "unlock_with_bucks" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_with_bucks"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "unlock_with_bucks"')
plt.xticks(rotation=90)
plt.show()
```

executed in 215ms, finished 23:37:12 2021-06-07



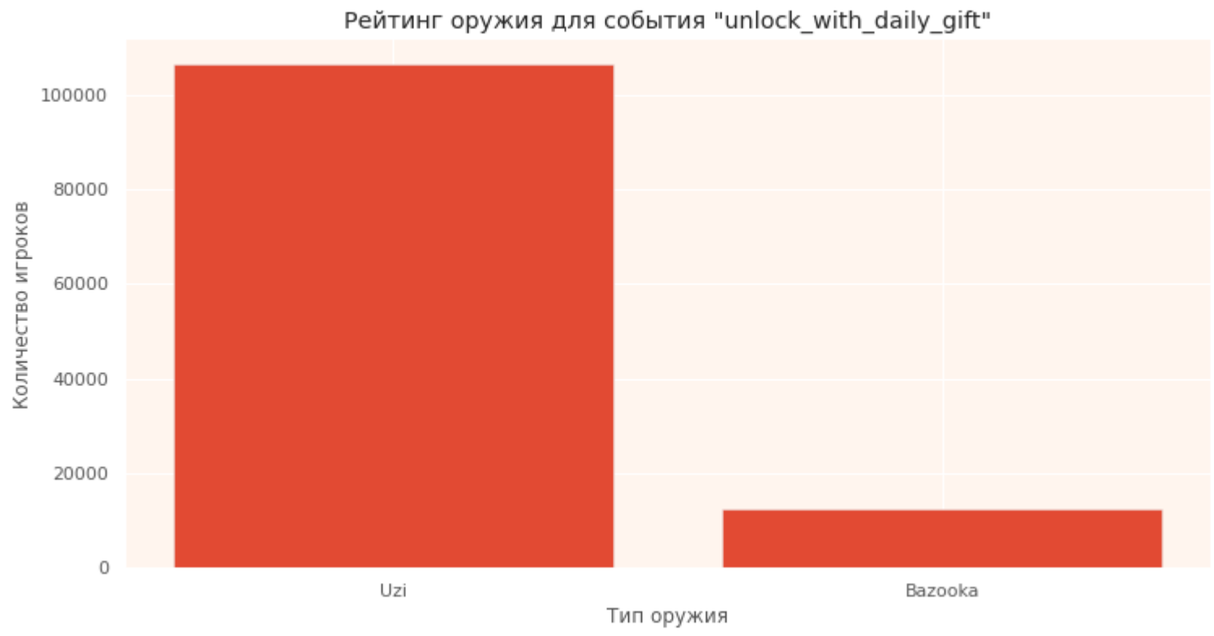
In [22]:
```
# Рейтинг оружия для события "unlock_with_coins" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_with_coins"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "unlock_with_coins"')
plt.xticks(rotation=90);
plt.show()
```

executed in 230ms, finished 23:37:12 2021-06-07



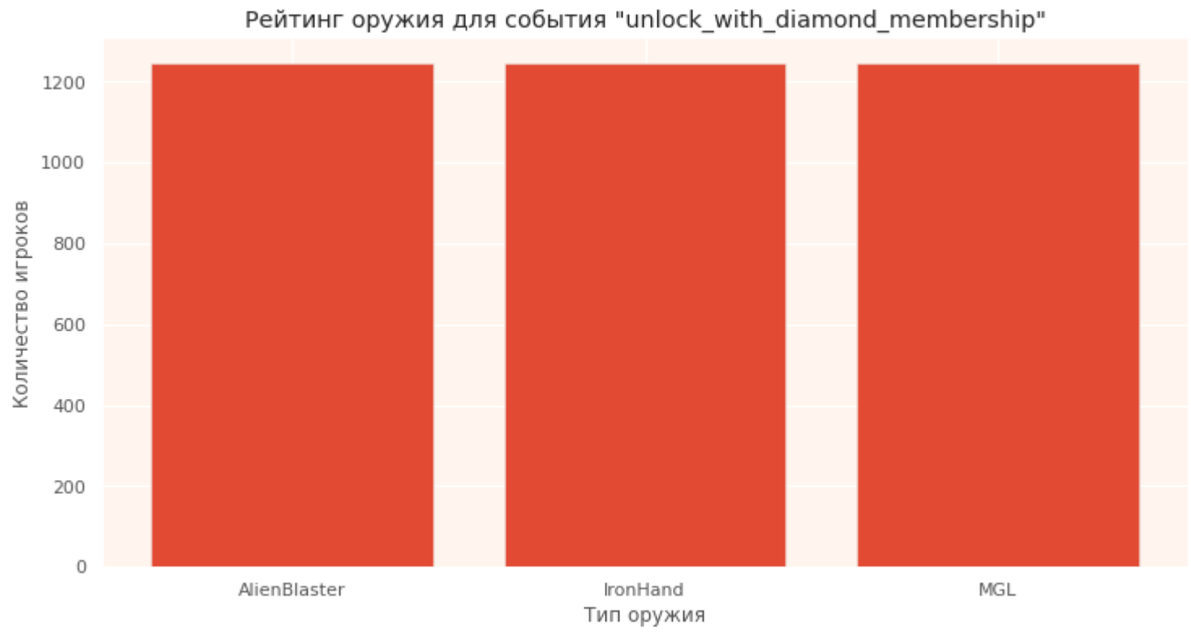
In [23]:
```
# Рейтинг оружия для события "unlock_with_daily_gift" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_with_daily_gift"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "unlock_with_daily_gift"')
plt.show()
```

executed in 89ms, finished 23:37:12 2021-06-07



In [24]:
```
# Рейтинг оружия для события "unlock_with_diamond_membership" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_with_diamond_membership"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "unlock_with_diamond_membership"')
plt.show()
```

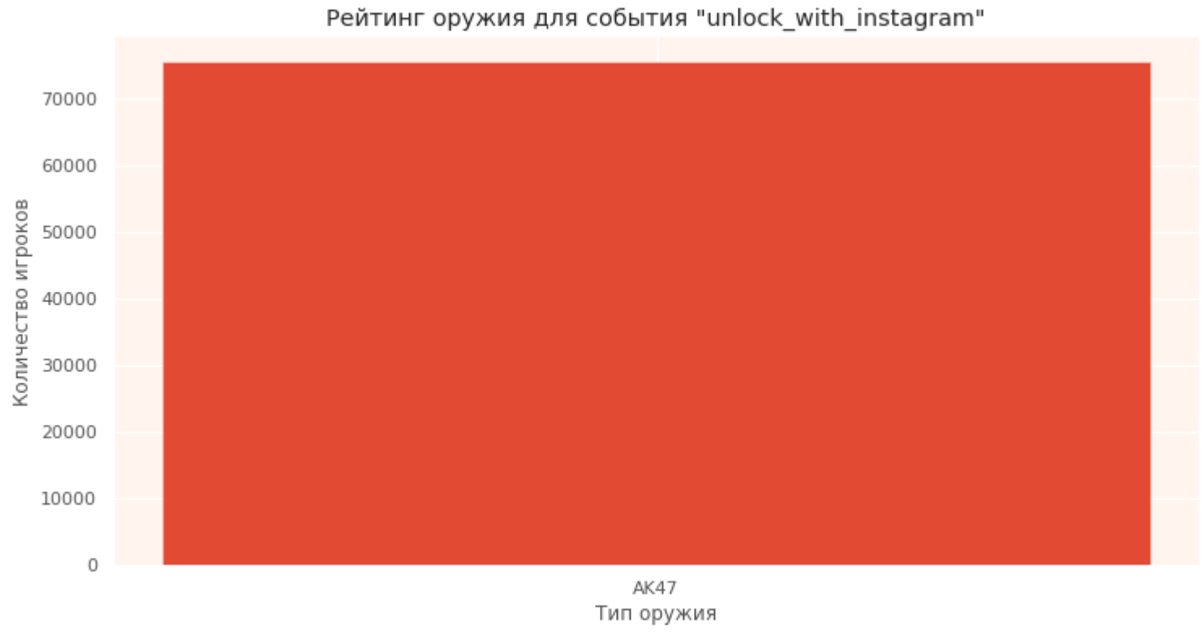
executed in 94ms, finished 23:37:12 2021-06-07



Contents

In [25]:
```
# Рейтинг оружия для события "unlock_with_instagram" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_with_instagram"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "unlock_with_instagram"')
plt.show()
```

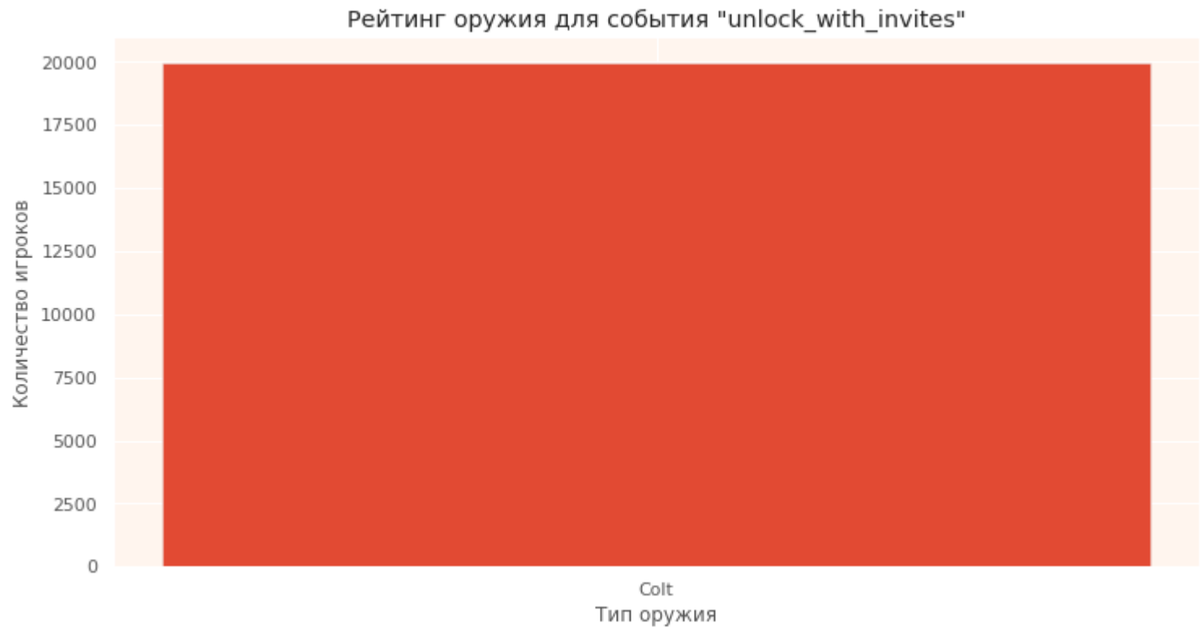
executed in 87ms, finished 23:37:12 2021-06-07



Contents

In [26]:
```
# Рейтинг оружия для события "unlock_with_invites" за месяц
unrelevant_info = pop_gun[pop_gun['event_name'] == "unlock_with_invites"]

unrelevant_info = unrelevant_info.sort_index().sort_values('count_user', ascending=False)

x = unrelevant_info['gun']
y1 = unrelevant_info['count_user']

fig, ax = plt.subplots()

ax.bar(x, y1)

ax.set_facecolor('seashell')
fig.set_figwidth(12)  #  ширина Figure
fig.set_figheight(6)  #  высота Figure
plt.xlabel('Тип оружия')
plt.ylabel('Количество игроков')
plt.title('Рейтинг оружия для события "unlock_with_invites"')
plt.show()
```

executed in 92ms, finished 23:37:12 2021-06-07




```
Наибольшее количество игроков используют в начале нового матча и в окончании матча Magnum44,
на втором месте Glock, на третьем Thompson.
```

Contents


```
Действие "unlock_with_bucks" привлекает значительно меньше пользователей, чем другие активности,
но наиболее востребованное оружие в этой категории MiniGun, P90, M16

Действие unlock_with_instagram привлекло более 70 000 пользователей за месяц к оружию AK47

Наименее используемое оружие - Nerff

Помочь гейм-дизайнеру для развития игры может информация о наименее популярных видах оружия, которому может быть поле
а так же полезна информация о наиболее востребованном оружии, особенно платном, характеристики которого наиболее прод
```

3 Критерии лояльности

Хорошая практика разделять пользователей игры по группам лояльности.

Предложите свои критерии лояльности, разделите игроков по группам.

```
Для удержания пользователей может быть эффективно разделить игроков по группам лояльности исходя из длительности пери
Например, за 30 дней ежедневного посещения игрок получает achieve - бронзовую защиту, за 60 дней - серебряную, за 90
Для игрока, совершившего покупку "unlock_with_bucks", может быть засчитан один день пребывания за два, либо накопленн
```

```
Псевдокод
С помощью рекурсивного запроса написать расчет количества непрерывного порядка чисел месяца для каждого user_id
(day_to_bonus)
При наличии покупок умножать полученное число дней на 2 в степени N (N - количество покупок) (day_to_bonus_correct)

Case (when day_to_bonus = 1 then day_to_bonus, when day_to_bonus >1 then day_to_bonus_correct из предыдущего дня)
```

4 AB-тест

```
В нашей игре проводился АБ-тест. Каждый игрок был распределен в одну из групп (контрольная или тестовая) и получил
идентификатор 0 или 1 соответственно. В тесте мы хотели проверить гипотезу о том, что изменения в тестовой группе
положительно повлияют на денежные показатели: ARPU и конверсию. Проведите анализ АБ-теста: подтвердилась ли наша
гипотеза, какая группа победила в тесте?
```

```
Рассчитать метрики ARPU и Conversia для каждой группы.
Учитывать оплаты только с даты включения игрока в группу (where data_event>=joined_dt) (не повлияло на результат)

ab_group – номер группы АБ-теста, в которую попал игрок (0– контрольная, 1 – тестовая)

Из-за перегруза памяти сравнение групп проведено в 3 итерации:
1. временная таблица с перечнем покупок и ценами
2. добавление в test.events_data - покупок, стоимости, количества покупок, разделение по группам 1 и 0
3. сравнение метрик в группах
```

In [27]:
```
# группировка параметров для контрольной и тестовой группы

groups_ab = pd.read_sql(
    """CREATE TABLE test.collection_pay AS     --временная таблица с перечнем покупок и ценами
  with
collection as (
        select event_id, param_key, param_value
        from
        (select event_id, param_key,
        coalesce (param_value_string, param_value_int, param_value_float, param_value_double) param_value
                          from (select event_id, param_key,
                                                  NULLIF (param_value_string,'') param_value_string,
                                                  NULLIF (param_value_int,'') param_value_int,
                                                  NULLIF (param_value_float,'') param_value_float,
                                                  NULLIF (param_value_double,'') param_value_double
                                from  test.parameters_data) n
        where param_key is not null) m)
    select
        distinct c1.event_id,
        c1.param_key,
        c2.param_value as product_id,
        price_game.price
            from collection c1
                    left join collection c2 on c1.event_id = c2.event_id and c2.param_key = 'product_id'
                    left join test.prices_data price_game on c2.param_value = price_game.product_id
where c1.param_key = 'product_id';

select * from test.collection_pay
""",
    conn)
```

executed in 23.1s, finished 23:37:35 2021-06-07

0— контрольная (control_group)

1 — тестовая (test_group)

Contents

In [28]:
```
# --добавление в test.events_data - покупок, стоимости, количества покупок, разделение по группам 1 и 0

test_group = pd.read_sql(
    """CREATE TABLE test.grooping AS
with
ab_test as (select user_id, ab_group, date(to_timestamp(floor(joined/1000000))) joined_dt
from test.ab_data order by joined),
users as (select m.max_event_id, date(m.data_event) data_event, m.user_id, m.event_name, ab_test.ab_group from
        (select *, date(to_timestamp(floor(event_timestamp/1000000))) data_event , max (event_id) max_event_id
        from test.events_data group by 1,2,3,4) m
        left join ab_test on m.user_id = ab_test.user_id and m.data_event>= ab_test.joined_dt)
    select
         users.max_event_id,
         users.data_event,
         users.user_id,
         users.event_name,
         users.ab_group,
         collection_pay.product_id,
         collection_pay.price,
         count(collection_pay.product_id) over (partition by users.data_event, users.user_id) count_purchse
         from users
            left join test.collection_pay on collection_pay.event_id = users.max_event_id;

select
        data_event,
        count(distinct user_id) all_users_in_group,
        count(product_id) count_pay_users_in_group,
        sum(price) revenue,
        sum(count_purchse) count_purchse,
        sum(price)/count(product_id) ARPPU,
        (cast(sum(count_purchse) as float) / cast (count(distinct user_id) as float) *100) conversion_rate
from test.grooping where ab_group = 1 group by 1;


""",
    conn)
```

executed in 3m 33s, finished 23:41:08 2021-06-07

```
test_group= pd.read_sql(
    """select
        data_event,
        count(distinct user_id) all_users_in_group,
        count(product_id) count_pay_users_in_group,
        sum(price) revenue,
        sum(count_purchse) count_purchse,
        sum(price)/count(product_id) ARPPU,
        (cast(sum(count_purchse) as float) / cast (count(distinct user_id) as float) *100) conversion_rate
from test.grooping where ab_group = 1 group by 1""",
    conn)
```

Contents

In [29]:
```
test_group.head()
```
executed in 10ms, finished 23:41:08 2021-06-07

Out[29]:

	data_event	all_users_in_group	count_pay_users_in_group	revenue	count_purchse	arppu	conversion_rate
0	2019-04-01	24538	185	1423.65	15427.0	7.695405	62.869835
1	2019-04-02	24076	221	2028.80	17990.0	9.180090	74.721715
2	2019-04-03	21850	133	997.20	11760.0	7.497744	53.821510
3	2019-04-04	22265	106	1171.44	12331.0	11.051321	55.382888
4	2019-04-05	24304	104	1047.46	13114.0	10.071731	53.958196

In [30]:
```
control_group= pd.read_sql(
    """select
        data_event,
        count(distinct user_id) all_users_in_group,
        count(product_id) count_pay_users_in_group,
        sum(price) revenue,
        sum(count_purchse) count_purchse,
        sum(price)/count(product_id) ARPPU,
        (cast(sum(count_purchse) as float) / cast (count(distinct user_id) as float) *100) conversion_rate
from test.grooping where ab_group = 0 group by 1""",
    conn)
```
executed in 14.6s, finished 23:41:23 2021-06-07

In [31]:
```
control_group.head()
```
executed in 8ms, finished 23:41:23 2021-06-07

Out[31]:

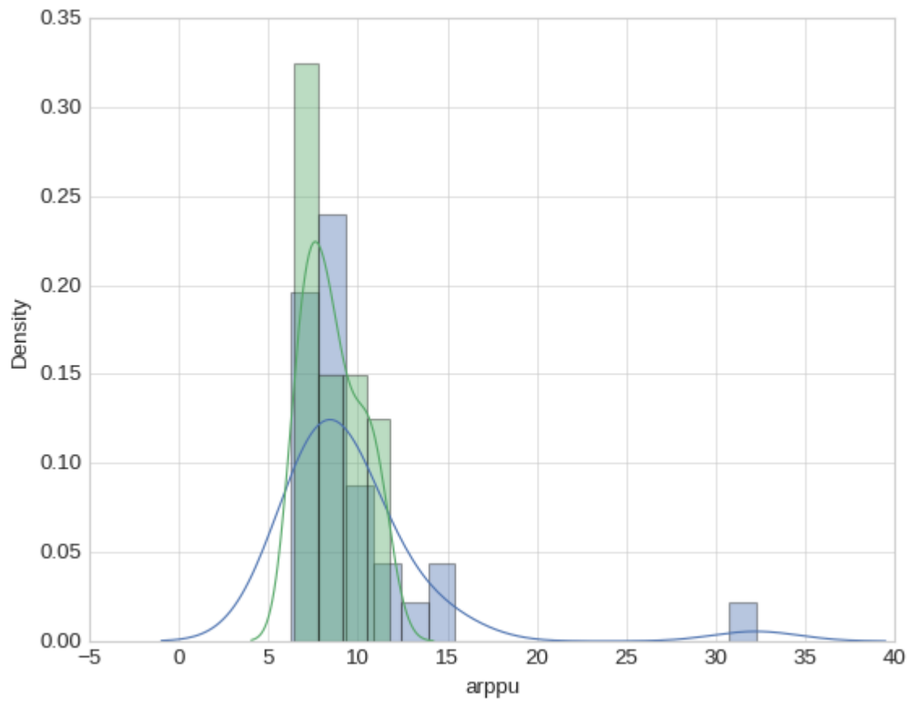
	data_event	all_users_in_group	count_pay_users_in_group	revenue	count_purchse	arppu	conversion_rate
0	2019-04-01	25025	296	2785.08	26419.0	9.409054	105.570430
1	2019-04-02	24244	213	1851.91	43605.0	8.694413	179.858934
2	2019-04-03	22104	115	1235.86	9273.0	10.746609	41.951683
3	2019-04-04	22567	182	1819.21	13949.0	9.995659	61.811495
4	2019-04-05	24509	206	1398.97	32470.0	6.791117	132.481945

Contents

In [32]:
```
plt.style.use('classic')
plt.style.use('seaborn-whitegrid')

data = pd.DataFrame(control_group, columns=['arppu'])
data = pd.DataFrame(test_group, columns=['arppu'])
sns.distplot(control_group['arppu'])
sns.distplot(test_group['arppu']);

plt.show()
```

executed in 161ms, finished 23:41:23 2021-06-07

```
/home/kate/.local/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecate
rsion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot
  warnings.warn(msg, FutureWarning)
/home/kate/.local/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecate
rsion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot
  warnings.warn(msg, FutureWarning)
```

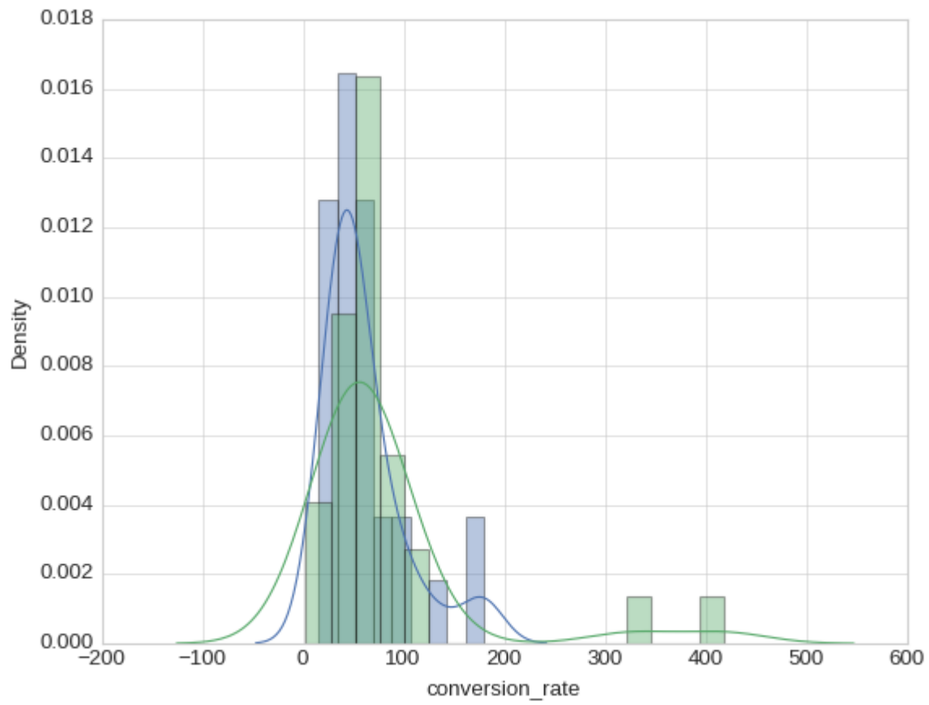


Contents

In [33]:
```
plt.style.use('classic')
plt.style.use('seaborn-whitegrid')

data = pd.DataFrame(control_group, columns=['conversion_rate'])
data = pd.DataFrame(test_group, columns=['conversion_rate'])
sns.distplot(control_group['conversion_rate'])
sns.distplot(test_group['conversion_rate']);

plt.show()
```

executed in 152ms, finished 23:41:23 2021-06-07

```
/home/kate/.local/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecate
rsion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot
  warnings.warn(msg, FutureWarning)
/home/kate/.local/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecate
rsion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot
  warnings.warn(msg, FutureWarning)
```



4.1 Как изменения в тестовой группе влияют на ARPU

Contents

In [34]:
```
# Проверка распределения на нормальность test_group

data = test_group.arppu
stat, p = normaltest(data)
alpha = 0.05
if p > alpha:
    print('Нормальное распределение')
else:
    print('Распределение не является нормальным')
```

executed in 3ms, finished 23:41:23 2021-06-07

```
Нормальное распределение
```

In [35]:
```
# Проверка распределения на нормальность control_group

data = control_group.arppu
stat, p = normaltest(data)
alpha = 0.05
if p > alpha:
    print('Нормальное распределение')
else:
    print('Распределение не является нормальным')
```

executed in 6ms, finished 23:41:23 2021-06-07

```
Распределение не является нормальным
```

```
Для оценки статистической значимости будет использован непараметрический метод. Нулевая гипотеза - Группы не имеют ст
гипотеза - Группы имеют значимые различия
```

In [36]:
```
from scipy.stats import mannwhitneyu

# generate two independent samples
data1 = test_group.arppu
data2 = control_group.arppu
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
     print('Не можем отклонить нулевую гипотезу ')
else:
    print('Отклоняем нулевую гипотезу')
```

executed in 6ms, finished 23:41:23 2021-06-07

```
Statistics=401.000, p=0.237
Не можем отклонить нулевую гипотезу
```

4.2 Как изменения в тестовой группе влияют на конверсию

Contents

In [37]:
```
# Проверка распределения на нормальность test_group

data = test_group.conversion_rate
stat, p = normaltest(data)
alpha = 0.05
if p > alpha:
    print('Нормальное распределение')
else:
    print('Распределение не является нормальным')
```

executed in 8ms, finished 23:41:23 2021-06-07

```
Распределение не является нормальным
```

In [38]:
```
# Проверка распределения на нормальность control_group

data = control_group.conversion_rate
stat, p = normaltest(data)
alpha = 0.05
if p > alpha:
    print('Нормальное распределение')
else:
    print('Распределение не является нормальным')
```

executed in 5ms, finished 23:41:23 2021-06-07

```
Распределение не является нормальным
```

```
Для оценки статистической значимости будет использован непараметрический метод. Нулевая гипотеза - Группы не имеют ст
гипотеза - Группы имеют значимые различия
```

In [39]:
```
from scipy.stats import mannwhitneyu

# generate two independent samples
data1 = test_group.conversion_rate
data2 = control_group.conversion_rate
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
     print('Не можем отклонить нулевую гипотезу ')
else:
    print('Отклоняем нулевую гипотезу')
```

executed in 4ms, finished 23:41:23 2021-06-07

```
Statistics=372.000, p=0.126
Не можем отклонить нулевую гипотезу
```

4.3 Вывод:

In [40]:
```
# -- сравнение метрик в группах

comp_metrics = pd.read_sql(
    """select ab_group,
            count(distinct user_id) all_users_in_group,
            count(product_id) count_pay_users_in_group,
            sum(price) revenue,
            sum(count_purchse) count_purchse,
            sum(price)/count(product_id) ARPPU,
            (cast(sum(count_purchse) as float) / cast (count(distinct user_id) as float) *100) conversion_rate
from test.grooping group by 1;
""",
    conn)
```

executed in 39.1s, finished 23:42:02 2021-06-07

In [41]:
```
comp_metrics.head()
```

executed in 8ms, finished 23:42:02 2021-06-07

Out[41]:

	ab_group	all_users_in_group	count_pay_users_in_group	revenue	count_purchse	arppu	conversion_rate
0	0	252242	3909	34262.41	384546.0	8.765006	152.451217
1	1	252792	3935	33887.16	468151.0	8.611731	185.192174

```
(0– контрольная, 1 – тестовая)
       В тесте мы хотели проверить гипотезу о том, что изменения в тестовой группе
       положительно повлияют на денежные показатели: ARPU и конверсию.

Группы не имеют статистически значимых различий.
       Проведите анализ АБ-теста: подтвердилась ли наша гипотеза, какая группа победила в тесте?

Гипотеза не подтвердилась, учитывая отсутствие статистически значимых различий, нельзя утверждать, что изменения в те
показатели: ARPU и конверсию.
```