```
Задание:
              В игру введена новуая функциональность. Необходимо проанализировать ее и определить полезна ли она игре,
              сравнить поведение игры после введения новой функциональности. Определить есть ли рост основных игровых
              метрик (retention, ARPU), и посмотреть удерживает игроков новый функционал лучше.
 In [1]: %matplotlib inline
              import pandas as pd
              import numpy as np
              Шаг 1. Получение, предобработка данных об активных и зарегистрированных
              пользователях игры
 In [2]: registered_users_count=pd.read_csv("registered_users.csv")
              registered_users_count.head()
 Out[2]:
                  Unnamed: 0 registration_date registered_users_count
               0
                              0
                                        2019-06-01
                                                                          4833
               1
                             1
                                        2019-06-02
                                                                          5255
                                        2019-06-03
                                                                          4193
               3
                              3
                                        2019-06-04
                                                                          4194
                                        2019-06-05
                                                                          3998
 In [3]: #исключим первый лишний столбец
              registered_users_count=registered_users_count.iloc[:, 1:4]
 In [4]: registered_users_count.head()
 Out[4]:
                   registration_date registered_users_count
                                                           4833
               0
                         2019-06-01
               1
                         2019-06-02
                                                           5255
                         2019-06-03
               2
                                                           4193
               3
                         2019-06-04
                                                           4194
               4
                         2019-06-05
                                                           3998
 In [5]: active_users_count_with_cohorts=pd.read_csv("active_users.csv")
              active_users_count_with_cohorts.head()
 Out[5]:
                  Unnamed: 0 activity_date registration_date active_users_count
               0
                                   2019-06-01
                                                       2019-06-01
                                                                                    1651
                                   2019-06-02
                                                       2019-06-01
                                                                                    1429
                                   2019-06-02
                                                       2019-06-02
                                                                                    1933
                                   2019-06-03
                                                       2019-06-01
                                                                                    1021
                                   2019-06-03
                                                       2019-06-02
                                                                                     1550
 In [6]: | active_users_count_with_cohorts=active_users_count_with_cohorts.iloc[:, 1:4]
              active_users_count_with_cohorts.head()
 Out[6]:
                  activity_date registration_date active_users_count
                    2019-06-01
                                         2019-06-01
                                                                      1651
                    2019-06-02
                                                                      1429
                                         2019-06-01
                    2019-06-02
                                         2019-06-02
                                                                      1933
                    2019-06-03
                                         2019-06-01
                                                                      1021
                     2019-06-03
                                         2019-06-02
                                                                      1550
              Посмотрим на информацию о датафрейме registered_users:
 In [7]: registered_users_count.info()
              <class 'pandas.core.frame.DataFrame'>
              RangeIndex: 30 entries, 0 to 29
              Data columns (total 2 columns):
               # Column
                                                          Non-Null Count Dtype
               0 registration_date
                                                          30 non-null
                                                                                   object
                    registered_users_count 30 non-null
                                                                                   int64
              dtypes: int64(1), object(1)
              memory usage: 608.0+ bytes
 In [8]: registered_users_count.head(4)
 Out[8]:
                   registration_date registered_users_count
                         2019-06-01
                                                           4833
               1
                         2019-06-02
                                                           5255
               2
                         2019-06-03
                                                           4193
               3
                                                           4194
                         2019-06-04
              Поменяем тип колонки с датой - object , на datetime :
 In [9]: registered_users_count['registration_date'] = pd.to_datetime(registered_users_count['registr
              ation_date'])
              Посмотрим на информацию о датафрейме active_users_count_with_cohorts:
In [10]: active_users_count_with_cohorts.info()
              <class 'pandas.core.frame.DataFrame'>
              RangeIndex: 930 entries, 0 to 929
              Data columns (total 3 columns):
                                                    Non-Null Count Dtype
                     Column
              - - -
               0
                     activity_date
                                                    930 non-null
                                                                             object
                     registration_date 930 non-null
                                                                             object
                     active_users_count 930 non-null
                                                                             int64
              dtypes: int64(1), object(2)
              memory usage: 21.9+ KB
              Типы колонок с датой - object, поменяем их на datetime:
In [11]: | for col in ['activity_date', 'registration_date']:
                    active_users_count_with_cohorts[col] = pd.to_datetime(active_users_count_with_cohorts[col])
              1])
                 1. Расчет Retention Rate Объединим таблицы два датафрейма: active_users_count_with_cohorts и
                    registered_users_count по колонке registration_date и запишем объединенные данные в датафрейм
                    retention_table.
              Посчитаем retention_rate - отношение числа активных пользователей active_users_count на конкретную
              неделю к общему числу пользователей в когорте registered_users_count .
              retention_table = active_users_count_with_cohorts.merge(registered_users_count,on=['registra
               tion_date'], how='left')
               retention_table['retention_rate'] = retention_table['active_users_count'] / retention_table[
               'registered_users_count']
              retention_table.head()
Out[12]:
                  activity date registration date active users count registered users count retention rate
               0 2019-06-01
                                         2019-06-01
                                                                      1651
                                                                                                 4833
                                                                                                              0.341610
                    2019-06-02
                                         2019-06-01
                                                                      1429
                                                                                                 4833
                                                                                                              0.295676
               2 2019-06-02
                                         2019-06-02
                                                                      1933
                                                                                                 5255
                                                                                                              0.367840
                    2019-06-03
                                         2019-06-01
                                                                      1021
                                                                                                 4833
                                                                                                              0.211256
                    2019-06-03
                                         2019-06-02
                                                                      1550
                                                                                                 5255
                                                                                                              0.294957
              Посчитаем показатель lifetime -разницу между датой активности activity_date и датой регистрации
               registration_date:
In [13]: retention_table['lifetime'] = retention_table['activity_date'] - retention_table['registrati
              retention_table['lifetime'] = retention_table['lifetime']/np.timedelta64(1,'D')
              retention_table['lifetime'] = retention_table['lifetime'].round().astype(int)
              retention_pivot = retention_table.pivot_table(index='registration_date',columns='lifetime',v
              alues='retention_rate', aggfunc='sum')
              Посчитаем усредненный Retention Rate по различным lifetime:
In [14]: retention_mean_by_lifetime = retention_pivot.mean()
              print (retention_mean_by_lifetime)
              lifetime
              0
                       0.350607
                       0.305707
              1
              2
                       0.223304
                       0.186753
              4
                       0.161584
              5
                       0.144832
              6
                       0.134225
              7
                       0.125151
              8
                       0.115505
              9
                       0.106549
                       0.100147
              10
                       0.094230
              11
              12
                       0.090701
                       0.087670
              13
              14
                       0.085370
              15
                       0.080326
              16
                       0.076764
              17
                       0.073749
              18
                       0.070988
              19
                       0.068546
              20
                       0.067471
              21
                       0.065852
              22
                       0.063643
              23
                       0.060973
              24
                       0.057966
                       0.056677
                       0.054686
              27
                       0.053935
              28
                       0.052781
              29
                       0.040779
                       0.027130
              30
              dtype: float64
              Шаг 2. Анализируем Retention Rate
              Построим график изменения Retention Rate в зависимости от lifetime:
In [15]: from matplotlib import pyplot as plt
              plt.figure(figsize=(13, 9))
              plt.title('Retention Rate в зависимости от lifetime для июньских пользователей')
              retention_mean_by_lifetime.plot.bar()
Out[15]: <AxesSubplot:title={'center':'Retention Rate в зависимости от lifetime для июньских пользоват
              елей'}, xlabel='lifetime'>
                                              Retention Rate в зависимости от lifetime для июньских пользователей
               0.35
               0.30
               0.25
               0.20
               0.15
               0.10
               0.05
                                                                Как видно, с течением времени Retention Rate убывает, при этом наблюдается существенное процентное падение на
              2, 29 и 30 день. Это будет особенно заметно, если построить график процентного изменения Retention Rate
              относительно предыдущего дня:
In [16]: plt.figure(figsize=(13, 9))
              plt.title('Процентное изменение Retention Rate по сравнению с предыдущим днем')
              retention_mean_by_lifetime.pct_change().plot.bar()
Out[16]: <AxesSubplot:title={'center':'Процентное изменение Retention Rate по сравнению с предыдущим д
              нем'}, xlabel='lifetime'>
                                             Процентное изменение Retention Rate по сравнению с предыдущим днем
                -0.05
               -0.10
                -0.15
                -0.20
                -0.25
                -0.30
              Построим тепловую карту изменения Retention Rate:
In [17]: import seaborn as sns
              from matplotlib import pyplot as plt
              plt.figure(figsize=(25, 15))
              plt.title('Retention Rate')
              sns.heatmap(retention_pivot, annot=True, fmt='.1%', linewidths=1, linecolor='gray')
Out[17]: <AxesSubplot:title={'center':'Retention Rate'}, xlabel='lifetime', ylabel='registration_dat</pre>
                                          10.8% 17.3% 15.1% 12.3% 11.8% 11.5% 11.1% 9.9% 9.4% 8.6% 8.5% 7.8% 8.1% 7.5% 7.5% 7.5% 6.7% 6.7% 6.7% 6.5% 6.5% 6.5% 6.5% 5.5% 5.5% 5.5% 5.3% 5.2% 4.2% 2.9%
                                           18 0% 14 8% 13 3% 13 6% 12 2% 10 7% 9 8% 9 0% 8 8% 8 0% 8 5% 8 0% 7 4% 6 8% 6 5% 6 5% 6 5% 6 5 6 6 4% 6 0 % 5 8 % 5 4 % 5 2 % 5 2 % 5 1 % 4 8 % 3 6 % 2 4 % 5 6 % 6 8 % 6 5 % 6 4 % 6 0 % 5 8 % 5 4 % 5 2 % 5 2 % 5 1 % 4 8 % 3 6 % 2 4 % 5 6 % 6 8 % 6 5 % 6 4 % 6 0 % 5 8 % 5 2 % 5 2 % 5 1 % 4 8 % 3 6 % 2 4 % 5 6 % 6 8 % 6 5 % 6 4 % 6 0 % 5 8 % 5 2 % 5 2 % 5 1 % 4 8 % 3 6 % 2 4 % 5 6 6 % 6 6 % 6 0 % 5 8 % 5 2 % 5 2 % 5 2 % 5 1 % 4 8 % 3 6 % 2 4 % 5 6 % 6 8 % 6 5 % 6 4 % 6 0 % 5 8 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 % 5 2 
                                            17.2% 14.7% 14.1% 13.2% 11.8% 10.8% 9.8% 9.4% 8.9% 8.8% 8.6% 8.4% 7.5% 7.5% 7.0% 6.9% 7.0% 7.0% 6.6% 6.4% 6.0% 5.6% 5.5% 5.5% 5.5% 5.4% 5.1% 3.8% 2.8%
                                            k 17.9% 16.9% 14.2% 13.1% 11.8% 11.0% 9.9% 9.6% 9.2% 9.0% 9.0% 8.6% 7.9% 7.3% 7.2% 7.4% 7.0% 6.6% 6.6% 6.6% 6.4% 5.7% 5.7% 5.5% 5.5% 5.6% 5.5% 4.6% 3.0%
                                           8% 19.7% 17.6% 16.1% 14.5% 14.5% 14.5% 13.7% 12.3% 12.2% 10.9% 10.3% 10.0% 9.8% 9.3% 8.9% 8.1% 7.9% 7.8% 7.4% 7.2% 7.0% 7.0% 6.7% 6.6% 6.1% 5.9% 6.1% 4.4% 3.1%
                                            6 19 0% 15 7% 14 5% 13 .3% 12 4% 11 .5% 11 .3% 10 .1% 9 .3% 8 .5% 7 .9% 7 .4% 7 .5% 6 .5% 6 .5% 6 .5% 6 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5% 5 .5

        4
        19
        14
        7%
        13
        5%
        12
        16
        11
        4%
        10
        18
        8
        8
        8
        7
        9%
        7
        6
        7
        6
        6
        6
        9%
        6
        18
        6
        18
        6
        18
        6
        18
        6
        18
        6
        18
        6
        18
        6
        18
        18
        18
        8
        8
        8
        7
        9%
        7
        6
        8
        6
        8
        6
        8
        6
        8
        6
        8
        6
        9
        6
        9
        18
        7
        9
        7
        8
        7
        9
        7
        7
        8
        9
        8
        8
        8
        8
        8
        9
        8
        9
        8
        9
        8
        9
        9
        8
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9
        9</
                                            18 9W 17.5% 16.6% 14.9% 12.9% 12.3% 10.8% 9.2% 9.4% 9.5% 8.5% 8.1% 8.0% 7.3% 6.9% 6.4% 6.7% 6.8% 6.4% 6.3% 5.8% 5.6% 5.3% 5.4% 5.3% 5.1% 3.7% 2.6%
                                           18, 17.98 15.38 13.0% 12.6% 11.6% 10.6% 9.7% 8.9% 8.5% 8.5% 8.3% 8.0% 7.6% 7.2% 6.6% 6.5% 6.4% 6.0% 6.0% 5.8% 5.5% 5.1% 5.1% 4.5% 4.8% 4.9% 3.5% 2.2%
                                           12% 16.6% 14.2% 12.1% 11.2% 10.2% 9.2% 8.9% 8.5% 8.0% 7.7% 7.2% 6.8% 6.2% 5.7% 5.9% 5.7% 5.9% 5.7% 5.2% 4.8% 4.9% 4.5% 4.3% 4.1% 4.2% 3.9% 3.9% 3.0% 2.1%
                                           1% 19.7% 16.8% 15.2% 13.7% 12.8% 12.2% 11.6% 10.2% 10.0% 9.5% 9.3% 9.0% 9.1% 8.5% 8.2% 7.4% 7.5% 7.4% 7.1% 7.4% 6.8% 6.3% 6.0% 5.7% 5.6% 5.8% 4.3% 3.0%
                                                17.2% 15.7% 15.2% 14.9% 13.5% 12.8% 12.2% 11.0% 10.0% 9.4% 9.1% 9.1% 9.1% 7.8% 81% 7.2% 7.1% 7.0% 6.8% 6.5% 6.4% 6.5% 6.4% 5.8% 5.6% 6.0% 4.4% 2.8%
              Шаг 3. Получение,предобработка данных о выручке
In [18]: revenue=pd.read_csv("revenue_users.csv")
              revenue=revenue.iloc[:,1:5]
              revenue.head()
Out[18]:
                         date registration_date users_count_with_revenue revenue
               0 2019-06-01
                                       2019-06-01
                                                                           1082 1611.860
               1 2019-06-02
                                       2019-06-01
                                                                             928
                                                                                  317.738
               2 2019-06-02
                                       2019-06-02
                                                                           1257 1305.220
                                       2019-06-01
               3 2019-06-03
                                                                             623
                                                                                    192.273
               4 2019-06-03
                                       2019-06-02
                                                                           1008
                                                                                    554.859
              Посмотрим на информацию о датафрейме revenue :
In [19]: revenue.info()
              <class 'pandas.core.frame.DataFrame'>
              RangeIndex: 833 entries, 0 to 832
              Data columns (total 4 columns):
                     Column
                                                             Non-Null Count Dtype
               0
                     date
                                                             833 non-null
                                                                                      object
               1
                     registration_date
                                                             833 non-null
                                                                                      object
                     users_count_with_revenue 833 non-null
                                                                                      int64
                                                             833 non-null
                                                                                     float64
               3
                     revenue
              dtypes: float64(1), int64(1), object(2)
              memory usage: 26.2+ KB
              Типы колонок с датой - object, поменяем их на datetime:
In [20]: for col in ['date', 'registration_date']:
                    revenue[col] = pd.to_datetime(revenue[col])
              Шаг 4. Подсчет выручки по когортам
              Объединим данные датафрейма revenue с данными датафрейма retention_table . Но прежде нам нужно
              измненить название колонки date на activity_date в датафрейме revenue. Это нужно для того, фчтобы в
              дальнейшем объединять датафреймы по одинаковому названию колонок.
In [21]: revenue = revenue.rename(columns={'date':'activity_date'})
              После переименования, можно объединить датафреймы в новый датафрейм retention_table_with_revenue :
In [22]: retention_table_with_revenue = retention_table.merge(revenue, on=['registration_date', 'activi
               ty_date'],how='left')
              Посмотрим на получившийся объединенный датафрейм:
In [23]: retention_table_with_revenue.info()
              <class 'pandas.core.frame.DataFrame'>
              Int64Index: 930 entries, 0 to 929
              Data columns (total 8 columns):
                     Column
                                                             Non-Null Count Dtype
                                                             930 non-null
                     activity_date
                                                                                      datetime64[ns]
                     registration_date
                                                             930 non-null
                                                                                      datetime64[ns]
                     active_users_count
                                                             930 non-null
                                                                                      int64
                    retention_rate
                                                             930 non-null
                                                                                      float64
                    lifetime
                                                             930 non-null
                                                                                      int32
                    users_count_with_revenue 833 non-null
                                                                                      float64
                     revenue
                                                             833 non-null
                                                                                      float64
              dtypes: datetime64[ns](2), float64(3), int32(1), int64(2)
              memory usage: 61.8 KB
              Видим, что после объединения у нас получились пропущенные значения. Это говорит о том, что не во все дни была
              выручка с пользователей.
              Произведем замену пропущенных значений на 0 с помощью функции fillna():
In [24]: for col in ['revenue', 'users_count_with_revenue']:
                    retention_table_with_revenue[col] = retention_table_with_revenue[col].fillna(0)
              Преобразуем тип в колонке users_count_with_revenue в целочисленный:
In [25]: retention_table_with_revenue['users_count_with_revenue'] = retention_table_with_revenue['use
               rs_count_with_revenue'].astype(int)
              Рассчитаем ARPU в колонку arpu:
              retention_table_with_revenue['arpu'] = retention_table_with_revenue['revenue'] / retention_t
              able_with_revenue['active_users_count']
              Шаг 5. Анализируем ARPU
              Создадим сводную таблицу arpu_pivot , где в строках будут даты регистрации пользователей, в колонках lifetime, а
              в показателе arpu:
In [27]: | arpu_pivot = retention_table_with_revenue.pivot_table(index='registration_date', columns='lif
              etime', values='arpu', aggfunc='sum')
              Построим тепловую карту по таблице arpu_pivot:
In [28]:
             plt.figure(figsize=(25, 15))
              plt.title('ARPU')
              sns.heatmap(arpu_pivot, annot=True, fmt='.2f', linewidths=1, linecolor='gray')
Out[28]: <AxesSubplot:title={'center':'ARPU'}, xlabel='lifetime', ylabel='registration_date'>
                                    024 004 013 009 044 007 006 006 007 002 027 025 012 001 030 006 006 007 002 027 025 012 001 030 006 056 002 000 006 018 005 032 002 004 005 000 000 003 000
                                    7 028 016 025 021 031 020 011 020 054 020 024 012 006 013 020 011 020 054 020 024 012 006 013 012 023 019 004 011 034 004 022 000 004 002 005 000 036 000 005
                                     005 052 016 020 000 007 013 012 047 006 019 011 012 047 006 019 001 012 028 007 012 019 023 017 007 008 004 007 028 004 031 013 036 019 02
                                   019 021 014 044 010 025 036 008 010 027 006 003 037 021 022 005 001 000 004 006 006 000 003 007 002 004 018 000 002 000
                                     035 016 010 0.08 0.25 0.14 0.15 0.06 0.11 0.16 0.17 0.06 0.12 0.13 0.04 0.02 0.04 0.18 0.02 0.03 0.04 0.02 0.08 0.00 0.12 0.06 0.04 0.00 0.03 0.21
                                   41 0.43 0.69 0.43 0.08 0.11 0.32 0.05 0.34 0.18 0.07 0.21 0.06 0.10 0.15 0.04 0.44 0.20 0.11 0.06 0.11 0.12 0.09 0.07 0.09 0.08 0.06 0.05 0.04 0.02 0.04
                                    055 013 073 042 050 001 086 006 001 044 023 001 000 010 010 010 010 034 005 010 004 015 006 007 044 013 000 000 016 016 008 00
                                    021 026 024 002 048 005 009 013 107 003 011 021 008 005 028 000 011 038 002 006 080 019 007 014 000 025 013 009 014
                                    51 021 021 011 022 027 004 038 015 031 022 027 004 038 015 031 029 0.55 0.63 021 0.02 002 002 002 000 000 037 003 019 014 014 000 0.20 0.00 0.25 0.20 0.00
                                   158 012 007 022 023 025 001 001 004 024 024 024 024 024 024 024 026 004 078 007 002 014 006 028 000 006 009 003 000 013 026 050 003 020 051 092 058 013 016 001 000 023 054 004 071 046 007 062 012 026 086 044 000 009 009 004 011 007 007 000 000 004 009 003 012 020 034
                                     0.08 0.08 0.07 0.15 0.13 0.08 0.08 0.02 0.00 0.38 0.17 0.17 0.18 0.01 0.07 0.16 0.61 0.12 0.16 0.59 0.00 0.00
              plt.figure(figsize=(13, 9))
              plt.title('Изменение ARPU в нулевой день')
              arpu_pivot[0].plot.bar()
Out[29]: <AxesSubplot:title={'center':'Изменение ARPU в нулевой день'}, xlabel='registration_date'>
                                                                   Изменение ARPU в нулевой день
               1.4
               1.2
               1.0
               0.2
                                 .00:00:00
                                          2019-06-06 00:00:00
                                              2019-06-07 00:00:00
                                                           2019-06-10 00:00:00
                                                                         2019-06-13 00:00:00
                                                                                               2019-06-18 00:00:00
                                                                                                                     2019-06-23 00:00:00
                                                                                                                         2019-06-24 00:00:00
                                                                                                                                                2019-06-29 00:00:00
                    2019-06-01 00:00:00
                                      2019-06-05 00:00:00
                                                   2019-06-08 00:00:00
                                                       2019-06-09 00:00:00
                                                                              2019-06-14 00:00:00
                                                                                  2019-06-15 00:00:00
                                                                                          2019-06-17 00:00:00
                                                                                                   2019-06-19 00:00:00
                                                                                                        2019-06-20 00:00:00
                                                                                                            2019-06-21 00:00:00
                                                                                                                 2019-06-22 00:00:00
                                                                                                                              2019-06-25 00:00:00
                                                                                                                                  2019-06-26 00:00:00
                                                                                                                                           2019-06-28 00:00:00
                                                                                                                                                    2019-06-30 00:00:00
                                                                2019-06-11 00:00:00
                                                                     2019-06-12 00:00:00
                                                                                      2019-06-16 00:00:00
                                                                                                                                       2019-06-27 00:00:00
              Как видно, сильнее всего выделяется ARPU в нулевой день в когорте 2019-06-26.
              Посчитаем средний ARPU в зависимости от lifetime:
In [30]: | arpu_mean_by_lifetime = arpu_pivot.mean()
              print(arpu_mean_by_lifetime)
              lifetime
                       0.748997
              1
                       0.247612
              2
                       0.231219
              3
                       0.226372
                       0.161454
              5
                       0.308641
              6
                       0.175224
                       0.153147
              8
                       0.140943
                       0.234994
                       0.237923
              10
              11
                       0.216741
              12
                       0.208700
              13
                       0.158504
              14
                       0.148857
              15
                       0.148024
              16
                       0.187117
              17
                       0.167570
              18
                       0.134404
                       0.100599
              19
                       0.153664
              20
              21
                       0.094581
              22
                       0.099685
              23
                       0.144491
              24
                       0.091224
              25
                       0.153679
              26
                       0.108511
              27
                       0.161853
              28
                       0.124374
              29
                       0.150843
                       0.162709
              30
              dtype: float64
In [31]: plt.figure(figsize=(13, 9))
              plt.title('Изменение среднего ARPU в зависимости от lifetime')
              arpu_mean_by_lifetime.plot.bar()
Out[31]: <AxesSubplot:title={'center':'Изменение среднего ARPU в зависимости от lifetime'}, xlabel='li
              fetime'>
                                                       Изменение среднего ARPU в зависимости от lifetime
               0.5
               0.3
                                                              Из графика можно сделать вывод, что сильнее всего ARPU подниматся на 5 день lifetime, также начиная с 9 дня
              наблюдается рост.
              Точнее о росте APRU -посмотрим на гистограмму изменения среднего ARPU относительно предыдущего дня:
In [32]: plt.figure(figsize=(13, 9))
              plt.title('Изменение среднего ARPU по сравнению с предыдущим днем')
              arpu_mean_by_lifetime.pct_change().plot.bar()
Out[32]: <AxesSubplot:title={'center':'Изменение среднего ARPU по сравнению с предыдущим днем'}, xlabe
              l='lifetime'>
                                                   Изменение среднего ARPU по сравнению с предыдущим днем
                 0.8
                 0.6
                 0.4
                 0.2
                 0.0
                -0.2
                -0.4
                -0.6
                                                                     Как видно, после 16 дня наблюдается картина, где ARPU чередуется днями роста и спада
```

АНАЛИЗ НОВОЙ ФУНКЦИОНАЛЬНОСТИ ИГРЫ.СРАВНЕНЕЕ ПОКАЗАТЕЛЕЙ ИГРЫ ДО И ПОСЛЕ ВВЕДЕНИЯ НОВОЙ

ФУНКЦИОНАЛЬНОСТИ.