Metrics_AntonNew_HomeWork_7.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменен

Комментировать Поделиться

+ Код + Текст

ОЗУ Диск

Редактирование

Подготовка

[1]
```
# imports
import numpy as np
import pandas as pd
import matplotlib
import plotly.express as px
from sqlalchemy import create_engine
from datetime import datetime, timedelta
from plotly.subplots import make_subplots
```

[2]
```
# connection
HOST = '37.139.42.145'
DBNAME = 'game-analytics'
USER = 'analytics'
PASSWORD = 'BRtTaqYiJyr29WXN'
TABLE_SCHEMA = 'data_viz_1068.project_dataset'
engine = create_engine(f'postgresql://{USER}:{PASSWORD}@{HOST}/{DBNAME}')
```

```
/usr/local/lib/python3.7/dist-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2 wheel package will be renamed from r
  """)
```

[3]
```
# getting table from database
project_dataset = pd.read_sql(f"""
    SELECT *
    FROM {TABLE_SCHEMA}
""", con=engine)

# converting some types to datetime
project_dataset['event_date'] = pd.to_datetime(project_dataset['event_date'])
project_dataset['cohort_date'] = pd.to_datetime(project_dataset['cohort_date'])

# dropping full duplicates (if they exists)
project_dataset.drop_duplicates()

# printing
project_dataset.head(2)
```

	event_time	event_date	event_name	revenue_usd	region	country	device_type	platform	cohort_date	user_id	user_type	con
0	2020-12-15 13:41:28	2020-12-15	LaunchApp	0.0	AS	UZ	lge-LM-X415K	android	2020-11-29	208976	organic	
1	2020-12-15 13:41:48	2020-12-15	LaunchApp	0.0	EU	RU	HONOR-DUA-L22	android	2020-12-10	211774	organic	

[4]
```
project_dataset = project_dataset.dropna(subset=['event_time'])
```

Дополнение основного датасета

[5]
```
# copy of main df
refined_df = project_dataset.copy()
```

[6]
```
# adding CIS region
refined_df['region'] = (np.where(
                        refined_df['country'].isin(['RU', 'UA', 'BE', 'AZ', 'AM', 'KZ', 'KG', 'MD', 'TJ', 'TM', 'UZ']
                        'CIS',
```

```
                                    refined_df['region']))
```

```
[7]  # adding install datetime
     install_times = refined_df[refined_df['event_name']=='FirstLaunchApp']
     install_times = install_times.drop_duplicates(subset=('user_id'))
     install_times = install_times[['user_id', 'event_time']]
     install_times = install_times.rename(columns={'event_time' : 'install_time'})
     refined_df = refined_df.merge(install_times, on='user_id')
     refined_df.head(1)
```

	event_time	event_date	event_name	revenue_usd	region	country	device_type	platform	cohort_date	user_id	user_type	con
0	2020-12-15 13:41:28	2020-12-15	LaunchApp	0.0	CIS	UZ	lge-LM-X415K	android	2020-11-29	208976	organic	

```
[8]  # adding install month
     refined_df['install_month'] = refined_df.install_time.dt.to_period('M')
```

```
[9]  # Removing events which happend before install times.
     refined_df = refined_df[refined_df['event_time'] >= refined_df['install_time']].copy()
     # adding living hours
     refined_df['lifehour'] = ((refined_df['event_time'] - refined_df['install_time']).dt.total_seconds() / 3600).astype('int'
     # adding living days
     refined_df['lifeday'] = (refined_df['event_time'] - refined_df['install_time']).dt.days
     refined_df.head(1)
```

	event_time	event_date	event_name	revenue_usd	region	country	device_type	platform	cohort_date	user_id	user_type	con
3	2021-01-01 16:57:30	2021-01-01	LaunchApp	0.0	CIS	UZ	lge-LM-X415K	android	2020-11-29	208976	organic	

Основные расчеты

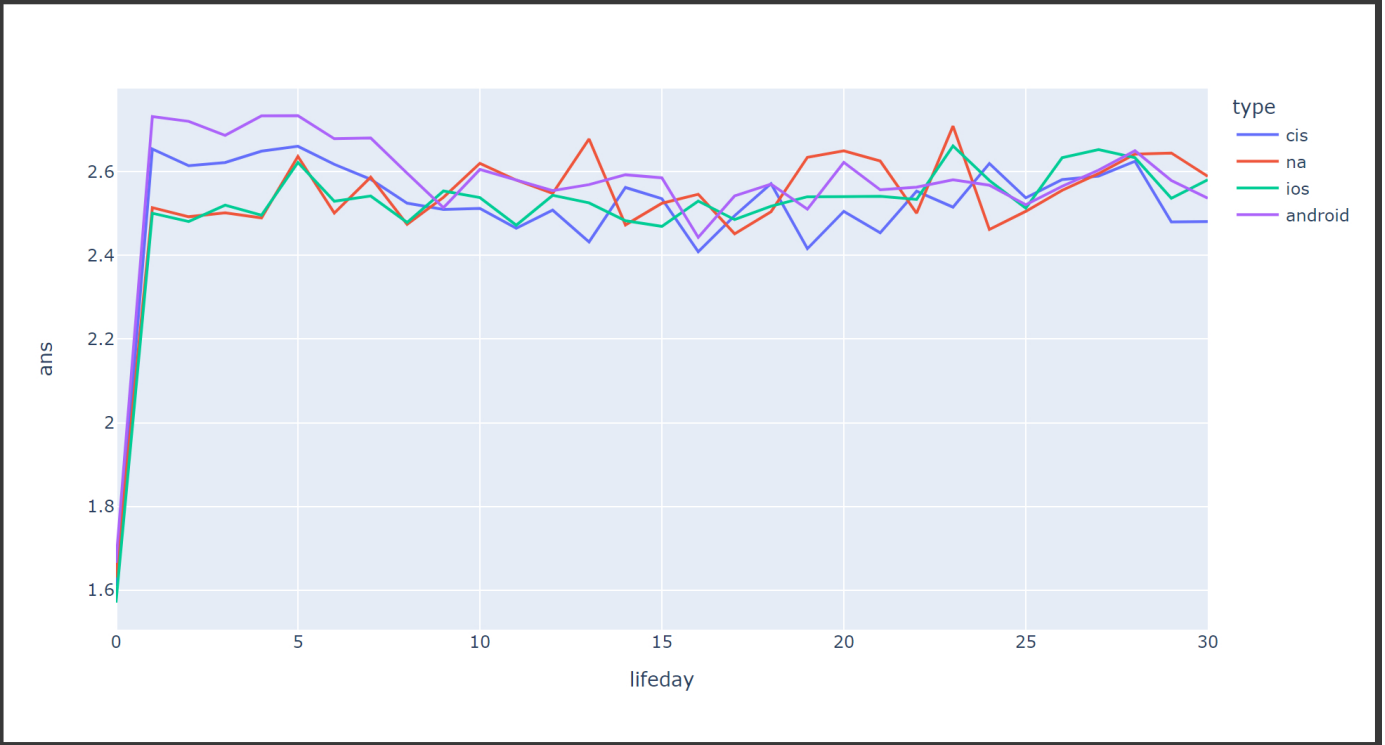
```
     pre_retention = refined_df.copy()
     pre_retention = pre_retention[pre_retention['lifeday'] <= 30]

     cis = pre_retention[pre_retention['region']=='CIS']
     na = pre_retention[pre_retention['region']=='NA']
     ios = pre_retention[pre_retention['platform']=='ios']
     android = pre_retention[pre_retention['platform']=='android']
```

```
[11] def get_reteniton_and_ans(data_frame):
         copy_df = data_frame.copy()
         copy_df = copy_df.groupby('lifeday').apply(lambda g: pd.Series({
                         'ans' : g[g['event_name']=='LaunchApp']['event_name'].count() / g['user_id'].nunique(),
                         'unique_users' : g['user_id'].nunique()
                     })).reset_index()
         copy_df['retention'] = round(copy_df['unique_users'] / copy_df['unique_users'][0], 4) * 100
         copy_df = copy_df[['lifeday', 'ans', 'retention']]
         return copy_df
```

```
[12] rt = get_reteniton_and_ans(cis).assign(type='cis')
     rt = rt.append(get_reteniton_and_ans(na).assign(type='na'))
     rt = rt.append(get_reteniton_and_ans(ios).assign(type='ios'))
     rt = rt.append(get_reteniton_and_ans(android).assign(type='android'))
```

Ans разных когорт

```
[13] px.line(rt, x='lifeday', y='ans', color='type').show()
```

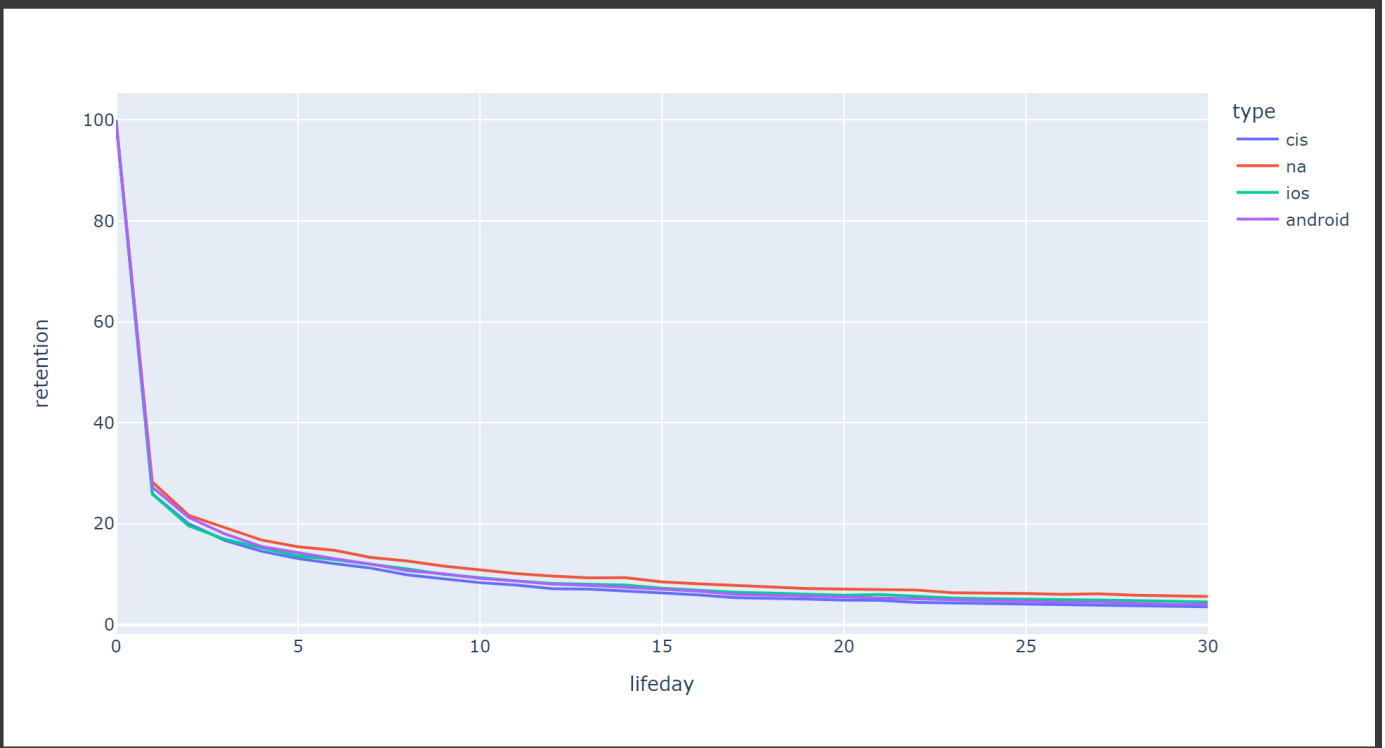


Ans нулевого дня почему то сильно ниже чем у всех остальных. В первый день наверно много народу отваливаются и не заходят второй раз.

В первые деветь дней Ans у Android выше чем у всех остальных, у CIS чуть меньше. После этого дня показания у когорт прыгают.

Retention

```
[14]  1  px.line(rt, x='lifeday', y='retention', color='type').show()
```



Кривая Retention у всех когорт сильно похожа.)

Лидером является NA. (так же помню что NA является и лидером по платящим, наглядное доказательства повышенного ретеншена у более платящей аудитории))

Ans/retention

```
[15]  1  dt = rt.copy()
      2  dt['ans_divided_by_retention'] = dt['ans'] / dt['retention']
      3  px.line(dt, x='lifeday', y='ans_divided_by_retention', color='type').show()
```

Не уверен что могу что то сказать по этому графику))
Разьве что показания расходятся сильнее к последнему дню.

Correlation

```
[16]  1  pre_correlation = rt.copy()
```

```
[17]  1  pre_correlation.head()
```

	lifeday	ans	retention	type
0	0	1.609944	100.00	cis
1	1	2.653954	25.90	cis
2	2	2.614035	20.00	cis
3	3	2.621498	16.69	cis
4	4	2.648851	14.56	cis

Общая

```
[18]  1  all_corelation = pre_correlation.corr()
      2  all_corelation.style.background_gradient(cmap='coolwarm', axis=1)
```

	lifeday	ans	retention
lifeday	1.000000	0.259820	-0.544317
ans	0.259820	1.000000	-0.852926

retention	-0.544317	-0.852926	1.000000

```
[19]  1  target_pre_correlation = pre_correlation[pre_correlation['type'] == 'na']
      2  correlation = target_pre_correlation.corr()
      3  correlation.style.background_gradient(cmap='coolwarm', axis=1)
```

	lifeday	ans	retention
lifeday	1.000000	0.402758	-0.550832
ans	0.402758	1.000000	-0.914313
retention	-0.550832	-0.914313	1.000000

```
[20]  1  target_pre_correlation = pre_correlation[pre_correlation['type'] == 'ios']
      2  correlation = target_pre_correlation.corr()
      3  correlation.style.background_gradient(cmap='coolwarm', axis=1)
```

	lifeday	ans	retention
lifeday	1.000000	0.433340	-0.534086
ans	0.433340	1.000000	-0.946444
retention	-0.534086	-0.946444	1.000000

```
[21]  1  target_pre_correlation = pre_correlation[pre_correlation['type'] == 'android']
      2  correlation = target_pre_correlation.corr()
      3  correlation.style.background_gradient(cmap='coolwarm', axis=1)
```

	lifeday	ans	retention
lifeday	1.000000	0.058574	-0.552723
ans	0.058574	1.000000	-0.777204
retention	-0.552723	-0.777204	1.000000

Для почти всех когорт:

кореляция между ans и retention отрицательная...

С увеличением ans retention получается падает..

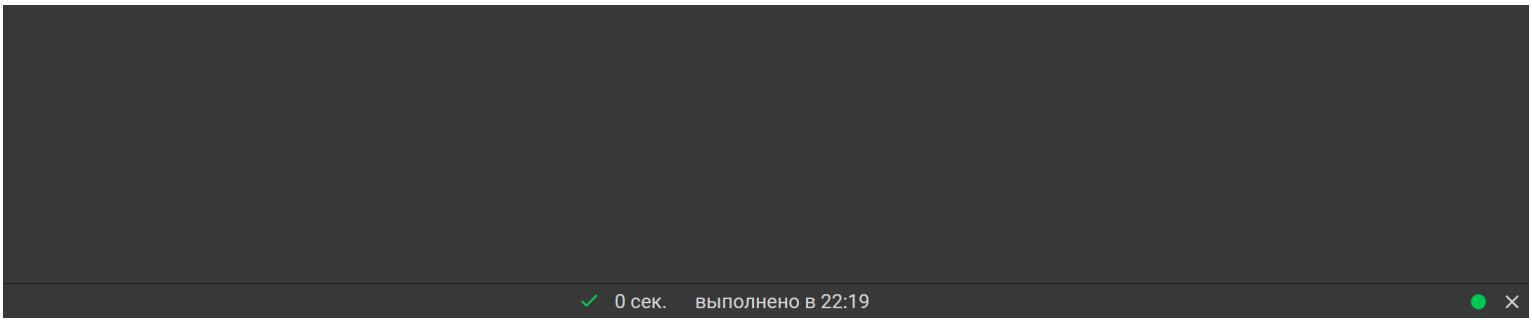
С увеличением дня жизни среднее количество сессий растет.

CIS

```
[22]  1  target_pre_correlation = pre_correlation[pre_correlation['type'] == 'cis']
      2  correlation = target_pre_correlation.corr()
      3  correlation.style.background_gradient(cmap='coolwarm', axis=1)
```

	lifeday	ans	retention
lifeday	1.000000	0.154821	-0.542130
ans	0.154821	1.000000	-0.813032
retention	-0.542130	-0.813032	1.000000

А вот для CIS: ситуация обратная.


0 сек. выполнено в 22:19