

+ Код + Текст

✓ ОЗУ
Диск

8

▼ Подготовка

100

```
[1] 1 # imports
    2 import numpy as np
    3 import pandas as pd
    4 import matplotlib
    5 import plotly.express as px
    6 from sqlalchemy import create_engine
    7 from datetime import datetime, timedelta
    8 from plotly.subplots import make_subplots
```

```
[2] 1 # connection
 2 HOST = '37.139.42.145'
 3 DBNAME = 'game-analytics'
 4 USER = 'analytics'
 5 PASSWORD = 'BRtTaqYiJyr29WXN'
 6 TABLE_SCHEMA = 'data_viz_1068.project_dataset'
 7 engine = create_engine(f'postgresql://{{USER}}:{{PASSWORD}}@{{HOST}}/{{DBNAME}}')
```

```
[3] 1 # getting table from database
2 project_dataset = pd.read_sql(f"""
3     SELECT *
4     FROM {TABLE_SCHEMA}
5     """, con=engine)
6
7 # converting some types to datetime
8 project_dataset['event_date'] = pd.to_datetime(project_dataset['event_date'])
9 project_dataset['cohort_date'] = pd.to_datetime(project_dataset['cohort_date'])
10
11 # dropping full duplicates (if they exists)
12 project_dataset.drop_duplicates()
13
14 # printing
15 project_dataset.head(2)
```

	event_time	event_date	event_name	revenue_usd	region	country	device_type	platform	cohort_date	user_id	user_type	con
	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	Ige-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]   1 project_dataset = project_dataset.dropna(subset=['event_time'])
```

▼ Дополнение основного датасета

```
[5] 1 # copy of main df  
0 2 refined_df = project.dataset.copy()  
cek
```

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

[7] 1 # adding install datetime
2 install_times = refined_df[refined_df['event_name']=='FirstLaunchApp']
3 install_times = install_times.drop_duplicates(subset=['user_id'])
4 install_times = install_times[['user_id', 'event_time']]
5 install_times = install_times.rename(columns={'event_time': 'install_time'})
6 refined_df = refined_df.merge(install_times, on='user_id')
7 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	Ige-LM-X415K	android	2020-11-29	208976	organic	



[8] 1 # adding install month
2 refined_df['install_month'] = refined_df.install_time.dt.to_period('M')

[9] 1 # Removing events which happened before install times.
2 refined_df = refined_df[refined_df['event_time'] >= refined_df['install_time']].copy()
3 # adding living hours
4 refined_df['lifehour'] = ((refined_df['event_time'] - refined_df['install_time']).dt.total_seconds() / 3600).astype('int')
5 # adding living days
6 refined_df['lifeday'] = (refined_df['event_time'] - refined_df['install_time']).dt.days
7 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	Ige-LM-X415K	android	2020-11-29	208976	organic	



▼ Основные расчеты

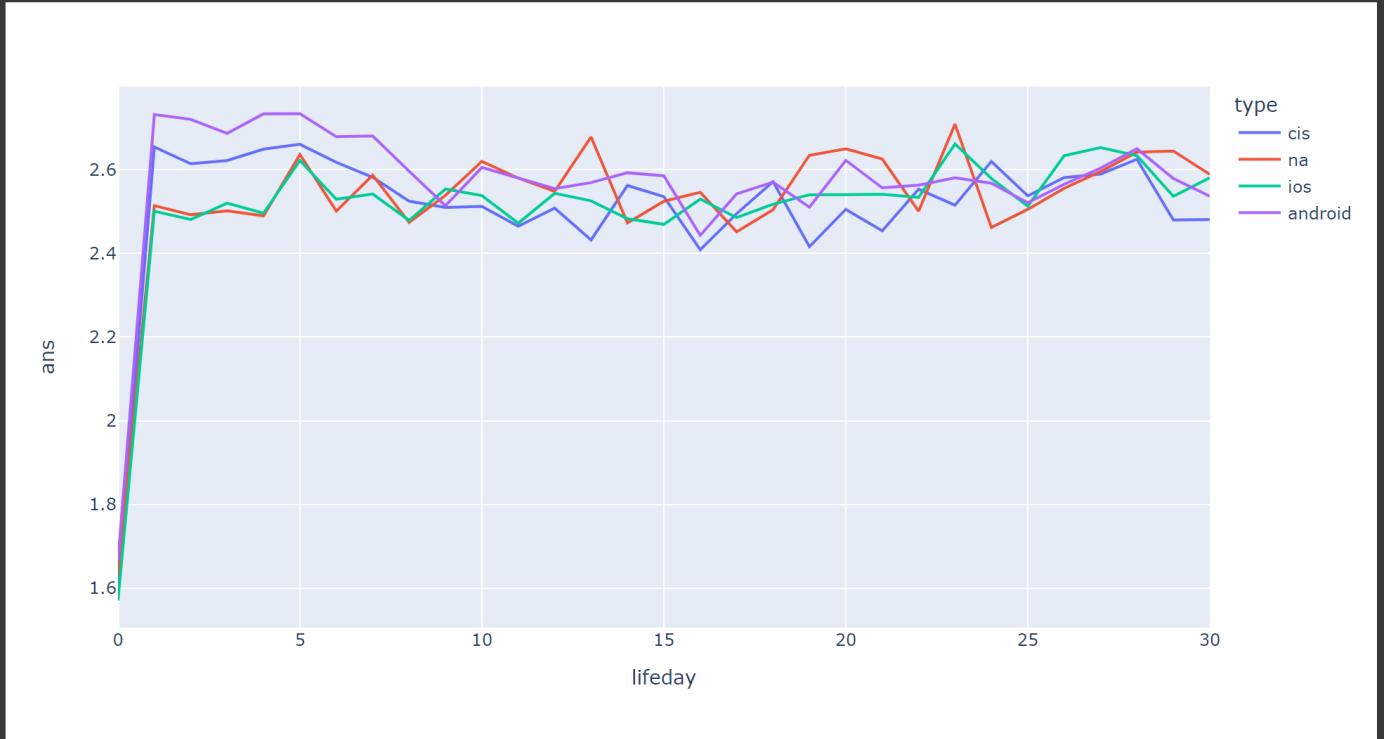
[10] 1 pre_retention = refined_df.copy()
2 pre_retention = pre_retention[pre_retention['lifeday'] <= 30]
3
4 cis = pre_retention[pre_retention['region']=='CIS']
5 na = pre_retention[pre_retention['region']=='NA']
6 ios = pre_retention[pre_retention['platform']=='ios']
7 android = pre_retention[pre_retention['platform']=='android']

[11] 1 def get_reteniton_and_ans(data_frame):
2 copy_df = data_frame.copy()
3 copy_df = copy_df.groupby('lifeday').apply(lambda g: pd.Series({
4 'ans' : g[g['event_name']=='LaunchApp']['event_name'].count() / g['user_id'].nunique(),
5 'unique_users' : g['user_id'].nunique()
6 }).reset_index())
7 copy_df['retention'] = round(copy_df['unique_users'] / copy_df['unique_users'][0], 4) * 100
8 copy_df = copy_df[['lifeday', 'ans', 'retention']]
9 return copy_df

[12] 1 rt = get_reteniton_and_ans(cis).assign(type='cis')
2 rt = rt.append(get_reteniton_and_ans(na).assign(type='na'))
3 rt = rt.append(get_reteniton_and_ans(ios).assign(type='ios'))
4 rt = rt.append(get_reteniton_and_ans(android).assign(type='android'))

▼ Ans разных когорт

[13] 1 px.line(rt, x='lifeday', y='ans', color='type').show()

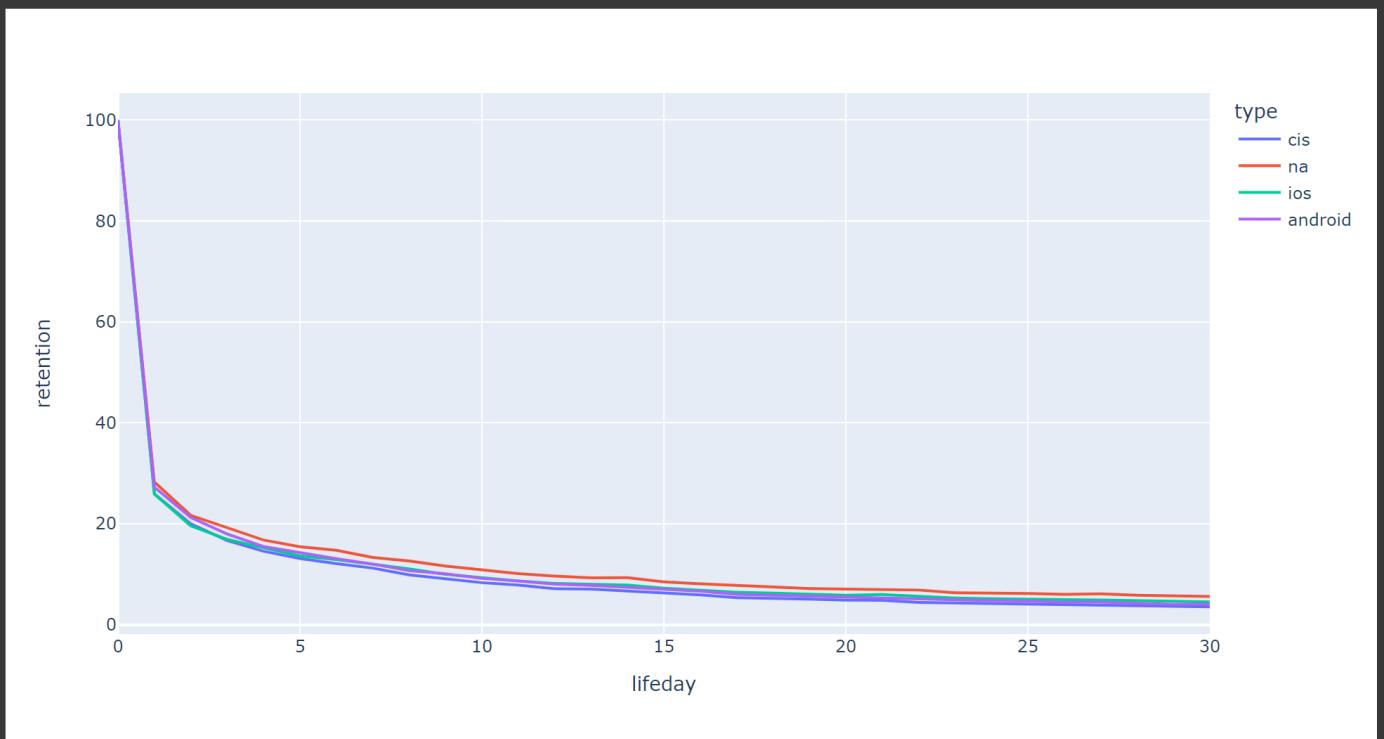


Ans нулевого дня почему то сильно ниже чем у всех остальных. В первый день наверно много народа отваливаются и не заходят второй раз.

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

▼ Retention

```
[14]: 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))

Разъве 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

```
[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=None)
```

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

✓
0
сек.

```
1 target_pre_correlation = pre_correlation[pre_correlation['type'] == 'cis']
2 correlation = target_pre_correlation.corr()
3 correlation.style.background_gradient(cmap='coolwarm', axis=None)
```

lifeday ans retention

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

✓
0
сек.

```
[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=None)
```

lifeday ans retention

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

✓
0
сек.

```
[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=None)
```

lifeday ans retention

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

✓
0
сек.

```
[23] 1 all_corelation = pre_correlation.corr()
2 all_corelation.style.background_gradient(cmap='coolwarm', axis=None)
```

lifeday ans retention

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

Корреляция между ans и retention отрицательная (-0.85)...

С увеличением ans retention получается падает..

Корреляция между lifeday и retention положительная (0.26).

С увеличением дня жизни среднее количество сессий растет.

Корреляция между lifeday и retention отрицательная (-0.54).

С увеличением дня жизни retention падает.

✓ 0 сек. выполнено в 22:31

