



## ▼ Подготовка

{x}

[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 import datetime  
8 from datetime import timedelta  
9 from plotly.subplots import make\_subplots

[2] 1 # connection  
2 HOST = '37.139.42.145'  
3 DBNAME = 'game-analytics'  
4 USER = 'analytics'  
5 PASSWORD = 'BRTaqYiJyr29WXN'  
6 TABLE\_SCHEMA = 'data\_viz\_1068.project\_dataset'  
7 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  
""")



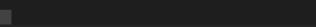
[3] 1 # getting table from database  
2 project\_dataset = pd.read\_sql(f"""  
3 SELECT \*  
4 FROM {TABLE\_SCHEMA}  
5 """, con=engine)  
6  
7 # converting some types to datetime  
8 project\_dataset['event\_date'] = pd.to\_datetime(project\_dataset['event\_date'])  
9 project\_dataset['cohort\_date'] = 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_ty
0	2020-12-11 16:20:44	2020-12-11	LaunchApp	0.0	EU	RU	samsung-SM-A510F	android	2020-10-31	182930	orga
1	2020-12-11 16:20:53	2020-12-11	LaunchApp	0.0	EU	RU	Redmi-Redmi Note 8 Pro	android	2020-11-11	199323	non_orga

[4] 1 project\_dataset = project\_dataset.dropna(subset=['event\_time'])

## ▼ Дополнение основного датасета

[5] 1 project\_dataset['region'] = (np.where(project\_dataset['country'].isin(['RU', 'UA', 'BE', 'AZ', 'AM', 'KZ', 'KG',  
2 'CIS',  
3 project\_dataset['region']]))



```
[6] 1 # adding install datetime
2 install_times = project_dataset[project_dataset['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 table_24 = project_dataset.merge(install_times, on='user_id')
7 table_24.head(1)
```

	event_time	event_date	event_name	revenue_usd	region	country	device_type	platform	cohort_date	user_id	user_type
0	2020-12-11 16:20:44	2020-12-11	LaunchApp	0.0	CIS	RU	samsung-SM-A510F	android	2020-10-31	182930	organic

```
[7] 1 # adding living times
2 table_24 = table_24[table_24['event_time'] >= table_24['install_time']]
3 table_24['lifehour'] = ((table_24['event_time'] - table_24['install_time']).dt.total_seconds() / 3600).astype('int')
4 table_24['lifeday'] = (table_24['event_time'] - table_24['install_time']).dt.days
5 table_24.head(1)
6 #table_24 = table_24[table_24['event_time'] <= table_24['install_time'] + timedelta(hours=24)]
```

	event_time	event_date	event_name	revenue_usd	region	country	device_type	platform	cohort_date	user_id	user_type
0	2020-12-11 16:20:44	2020-12-11	LaunchApp	0.0	CIS	RU	samsung-SM-A510F	android	2020-10-31	182930	organic

```
[8] 1 refined_df = table_24.copy()
2
3 refined_df['install_month'] = refined_df.install_time.dt.to_period('M').astype('str')
4 refined_df.install_month.unique()
```

```
array(['2020-12', '2020-11', '2021-03', '2021-02', '2021-05', '2020-10',
       '2021-04', '2021-01'], dtype=object)
```

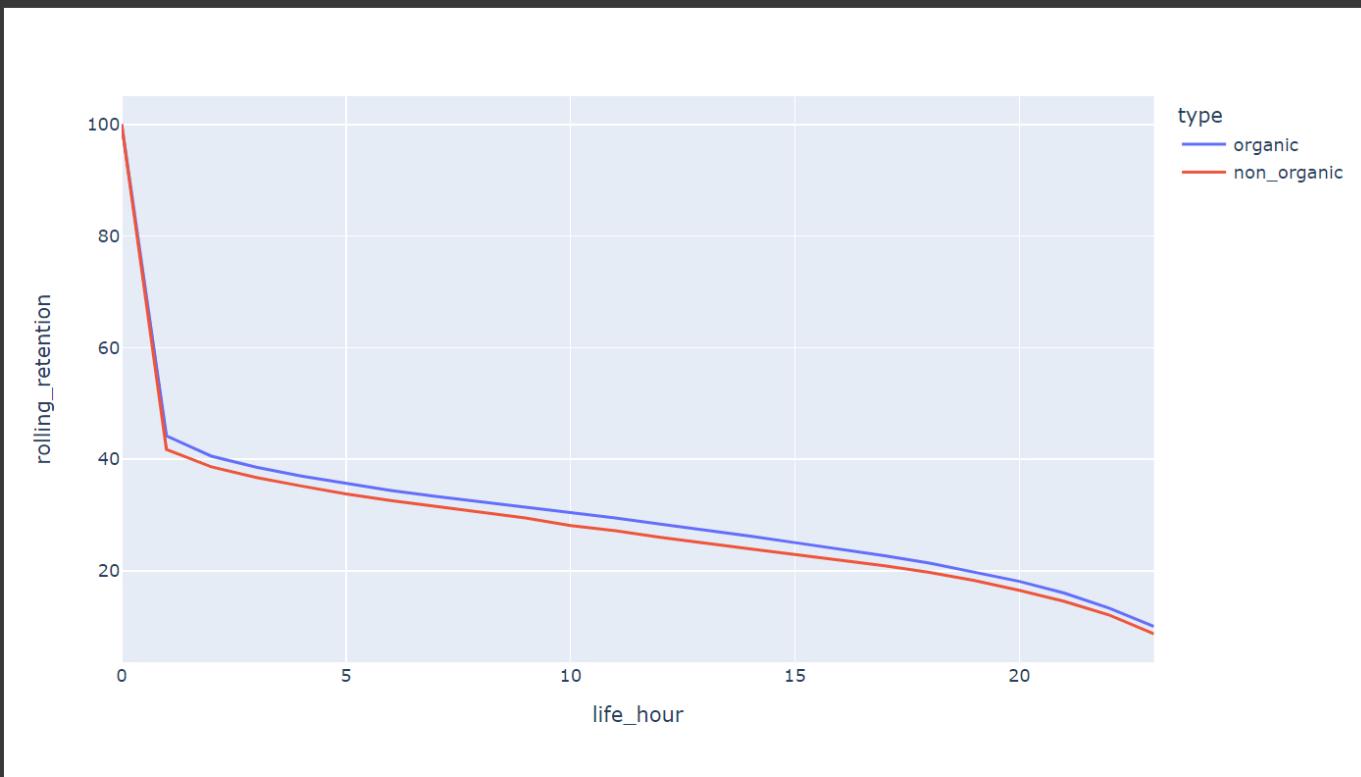
## ▼ 1. Rolling Retention В Первые 24 часа

```
[9] 1 pre_rolling = refined_df.copy()
2 pre_rolling = pre_rolling[pre_rolling['lifehour'] <= 24]
3 pre_rolling = pre_rolling[pre_rolling['region'].isin(['CIS', 'EU', 'NA'])]
4
5 organics = pre_rolling[pre_rolling['user_type']=='organic']
6 non_organics = pre_rolling[pre_rolling['user_type']=='non_organic']
```

```
[32] 1 def get_rolling_retention(data_frame):
2     copy_df = data_frame
3     copy_df = copy_df.groupby('user_id', as_index = False).agg(max_lifehour=('lifehour', 'max'))
4
5     df_1_to_24 = pd.DataFrame({'life_hour' : range(0,24)})
6     df_1_to_24['for_merge'] = 1
7     copy_df['for_merge'] = 1
8
9     df_rolling = pd.merge(df_1_to_24, copy_df, on = 'for_merge')
10    df_rolling = df_rolling[df_rolling['max_lifehour'] >= df_rolling['life_hour']]
11    df_rolling = (df_rolling.groupby('life_hour', as_index = False).agg(unique_users=('user_id', 'nunique'))
12                .sort_values(by='life_hour'))
13    df_rolling['rolling_retention'] = round(df_rolling['unique_users'] / df_rolling['unique_users'][0], 4) * 100
14
15    return df_rolling
16
17 def get_rolling_retention_per_region(data_frame):
18     result = pd.DataFrame([])
19     for name, region_group in data_frame.groupby('region', as_index=False):
20         result = result.append(get_rolling_retention(region_group).assign(region=name))
21
22     return result
23
```

## ▼ Rolling Retention для в разрезе органики / не органики

```
[11] 1 user_type_retention = (get_rolling_retention(organics).assign(type='organic')  
2 | | | | | .append(get_rolling_retention(non_organics).assign(type='non_organic')))  
3 px.line(user_type_retention, x='life_hour', y='rolling_retention', color='type').show()
```



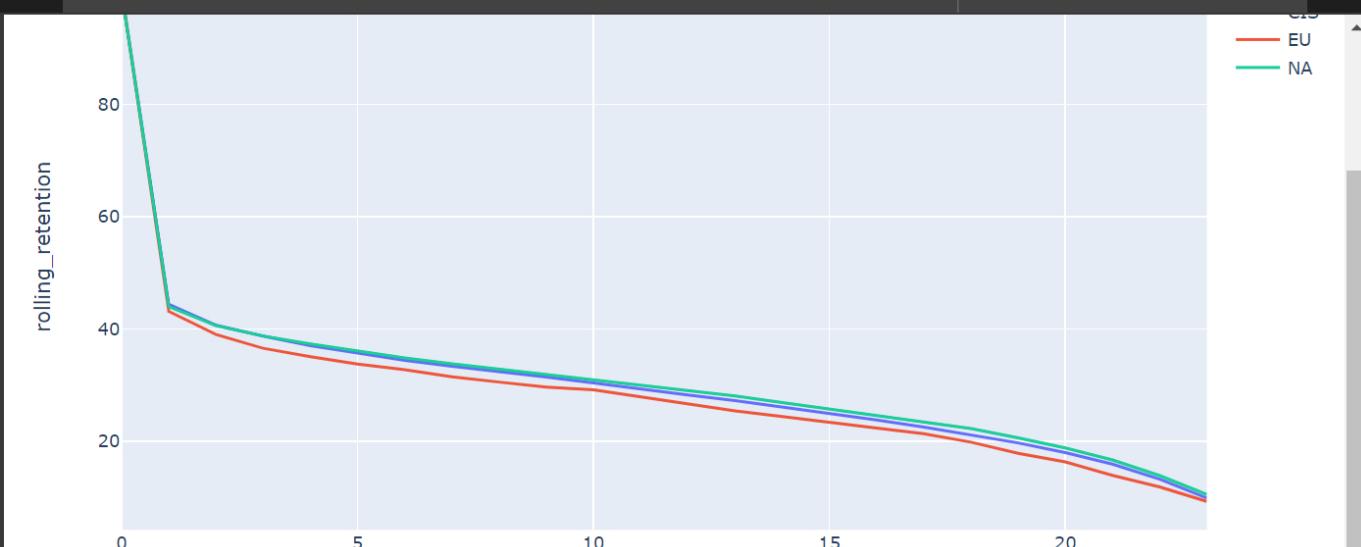
Rolling Retention органических пользователей в целом лучше чем у неорганических.

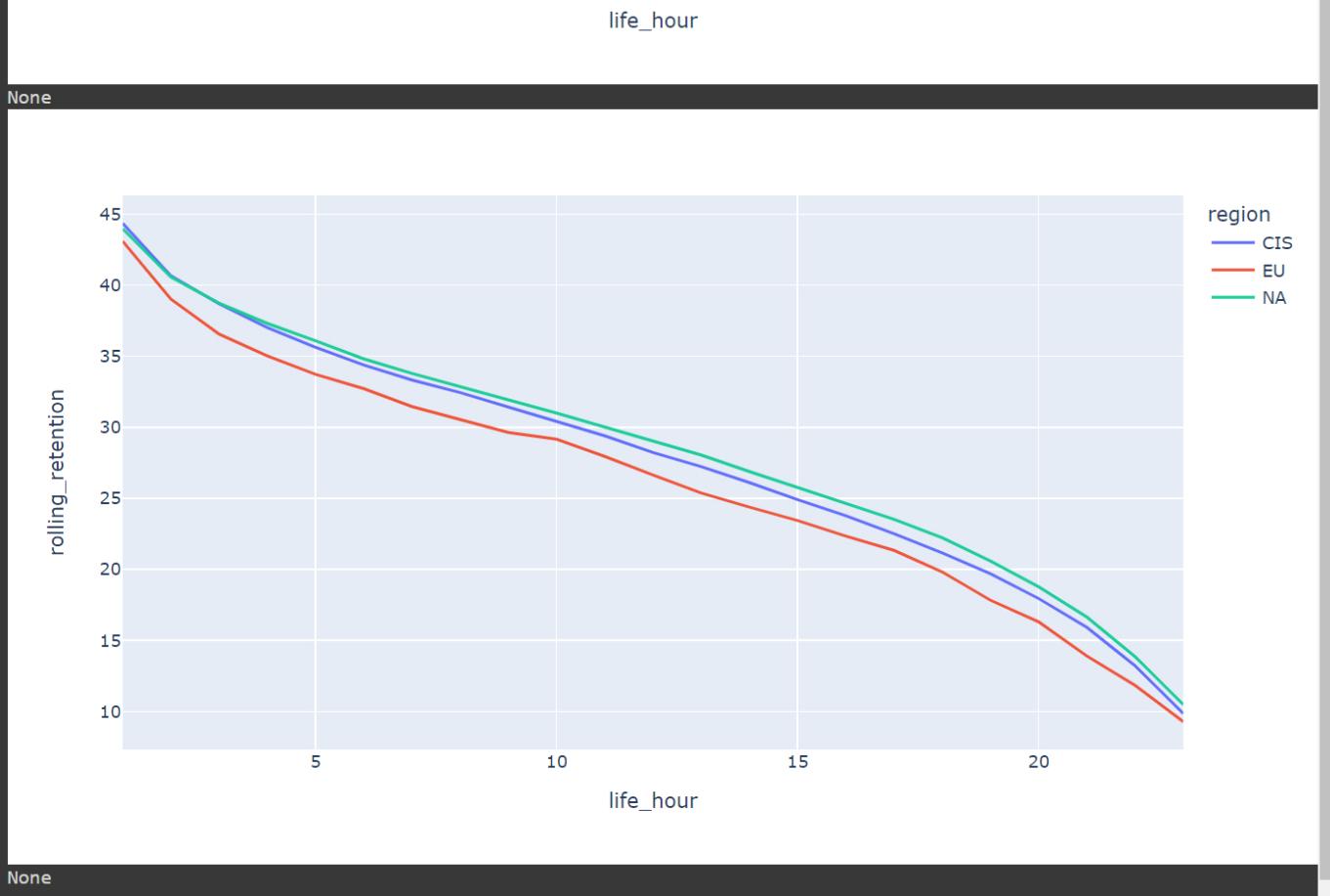
Кажется маркетинговая компания оказалась не очень удачная.

(Она приводила пользователей которые увлекаются игрой хуже чем пользователи зашедшие сами).

## ▼ Органика

```
[12] 1 rolling_retention_organic_regions = get_rolling_retention_per_region(organics)  
2 display(px.line(rolling_retention_organic_regions, x='life_hour', y='rolling_retention', color='region').show())  
3 without_0 = rolling_retention_organic_regions[rolling_retention_organic_regions['life_hour'] != 0]  
4 display(px.line(without_0, x='life_hour', y='rolling_retention', color='region').show())
```





Для органики аудитория из NA имеет лучшее удержание.

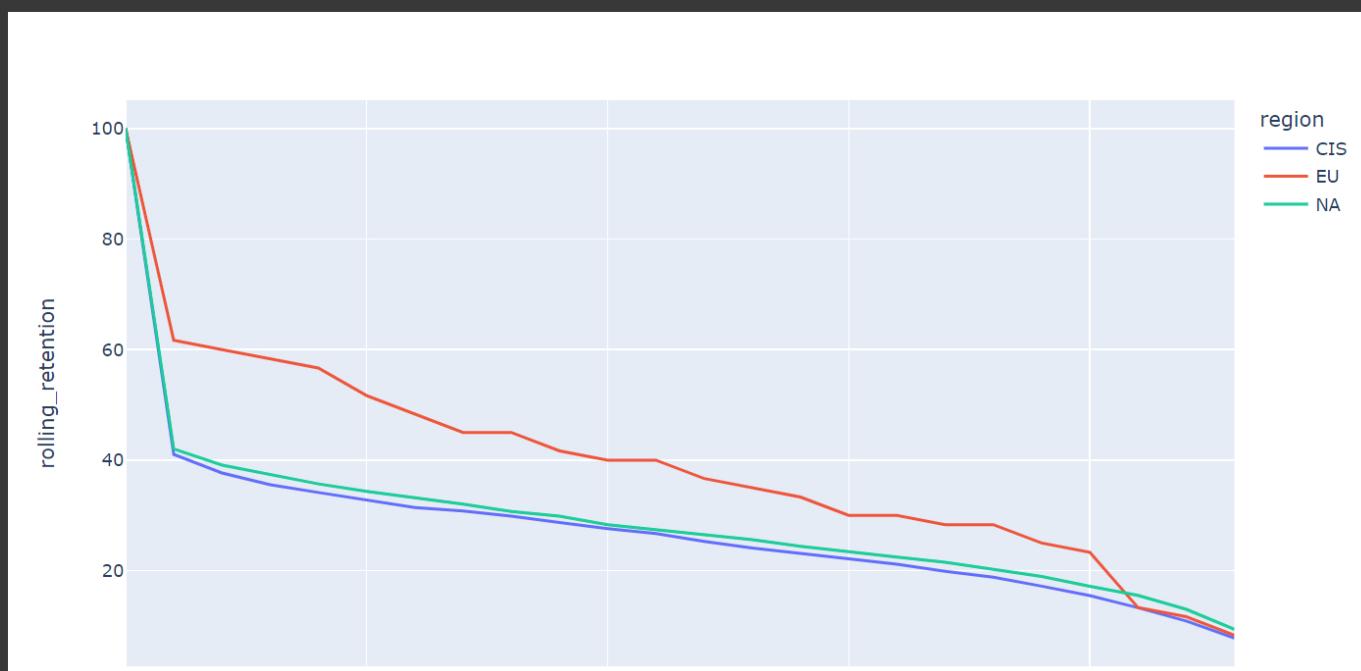
CIS отстает где то на 1%.

EU отстает еще на 2%.

NA выглядит более интересной аудиторией с точки зрения удержания в первые 24 часа.

#### ▼ Не органика

```
[13] 1  rolling_retention_non_organic_regions = get_rolling_retention_per_region(non_organics)
    2  px.line(rolling_retention_non_organic_regions, x='life_hour', y='rolling_retention', color='region').show()
```



Странно но EU, которая показала себя хуже всего для органики, теперь показывает результат сильно выше чем у других регионах.

```
[14] 1 print(non_organics.region.value_counts())
      2 print(non_organics.region.value_counts(normalize=True) * 100)

      NA    12258
      CIS    9900
      EU     252
      Name: region, dtype: int64
      NA    54.698795
      CIS   44.176707
      EU    1.124498
      Name: region, dtype: float64
```

Аудитория EU для неорганики - это всего 252 человека - 1% от общего числа из 3 исследуемых регионов. Кажется что данных по ней слишком мало и они противоречат данным по органики. Думаю что это какая то аномалия и ее не следует рассматривать.

(ну либо как вариант маркетинг очень хорошо попал в аудиторию именно в европе, гипотетически, но врятли)

Для неорганики NA попрежнему лучше всего удерживается. Далее идет CIS.

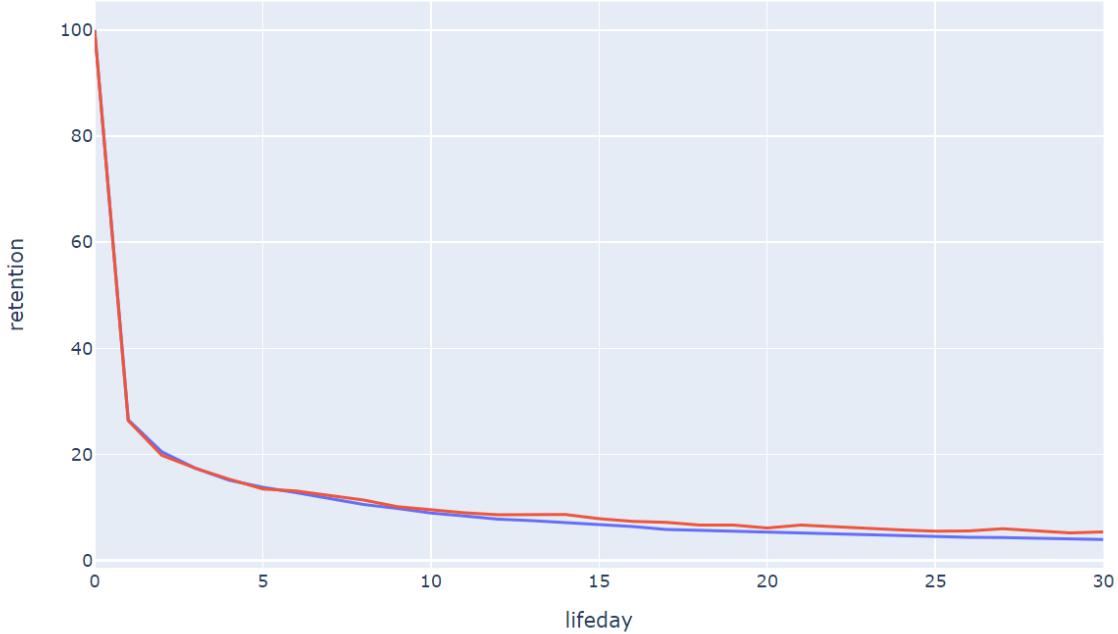
## ▼ 2. Retention rate 30 дней

```
[15] 1 pre_retention = refined_df.copy()
      2 pre_retention = pre_retention[pre_retention['lifeday'] <= 30]
      3 pre_retention = pre_retention[pre_retention['region'].isin(['CIS', 'EU', 'NA'])]
      4
      5 organics = pre_retention[pre_retention['user_type']=='organic']
      6 non_organics = pre_retention[pre_retention['user_type']=='non_organic']
      7
      8 ios = pre_retention[pre_retention['platform']=='ios']
      9 android = pre_retention[pre_retention['platform']=='android']
```

```
[16] 1 def get_retention_rate(data_frame):
      2     copy_df = data_frame
      3
      4     copy_df = copy_df.groupby('lifeday', as_index = False).agg(unique_users=('user_id', 'nunique')).sort_values(by=
      5     copy_df['retention'] = round(copy_df['unique_users'] / copy_df['unique_users'][0], 4) * 100
      6
      7     return copy_df
      8
      9 def get_retention_rate_per_region(data_frame):
     10     result = pd.DataFrame([])
     11     for name, group in data_frame.groupby(['region']):
     12         result = result.append(get_retention_rate(group).assign(region=name))
     13
     14     return result
```

## ▼ Retention Rate у органики и не органики

```
[17] 1 retention_rate_30 = get_retention_rate(organics).assign(type='organic')
      2 retention_rate_30 = retention_rate_30.append(get_retention_rate(non_organics).assign(type='non_organic'))
      3 display(px.line(retention_rate_30, x='lifeday', y='retention', color='type').show())
      4 without_0 = retention_rate_30[retention_rate_30['lifeday'] != 0]
      5 display(px.line(without_0, x='lifeday', y='retention', color='type').show())
```



None

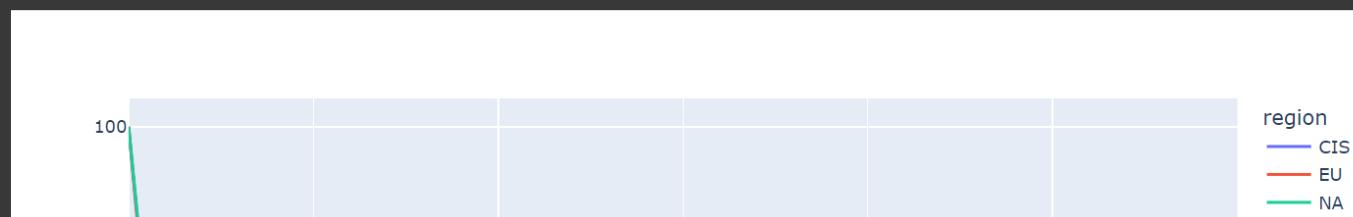


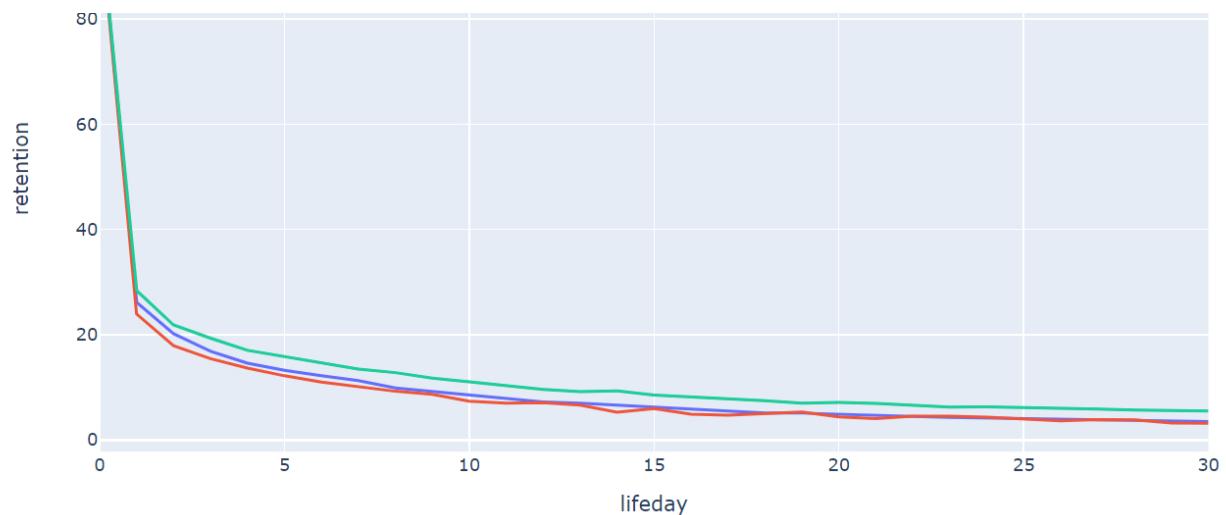
[18] 1 pre\_retention.region.value\_counts()

```
CIS    261673
NA     142150
EU     18743
Name: region, dtype: int64
```

#### ▼ Retention Rate organics

[19] 1 retention\_rate\_30\_organic = get\_retention\_rate\_per\_region(organics)
2 display(px.line(retention\_rate\_30\_organic, x='lifeday', y='retention', color='region').show())





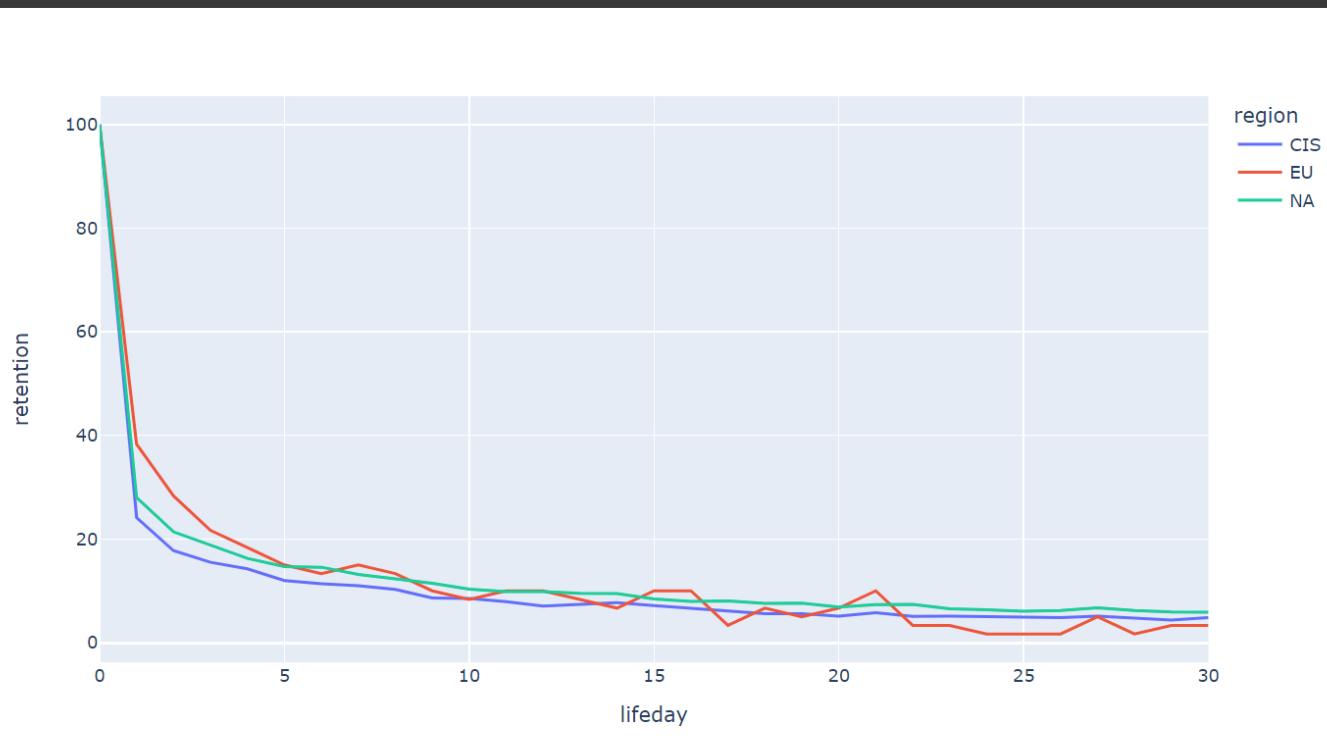
None

Для органики Retention Rate лучше всех у NA. До 15 дня EU чуть меньше чем у CIS. Далее CIS и EU практически сравниваются.

Rolling Retention посчитанный в первые 24 часа показывает примерно такую же картину

#### ▼ Retantion Rate non organics

```
[20] 1  retention_rate_30_non_organic = get_retention_rate_per_region(non_organics)
2  px.line(retention_rate_30_non_organic, x='lifeday', y='retention', color='region').show()
```



```
[21] 1  display(non_organics[non_organics['lifeday']==1].region.value_counts())
2  display(non_organics[non_organics['lifeday']==1].region.value_counts(normalize=True)*100)
```

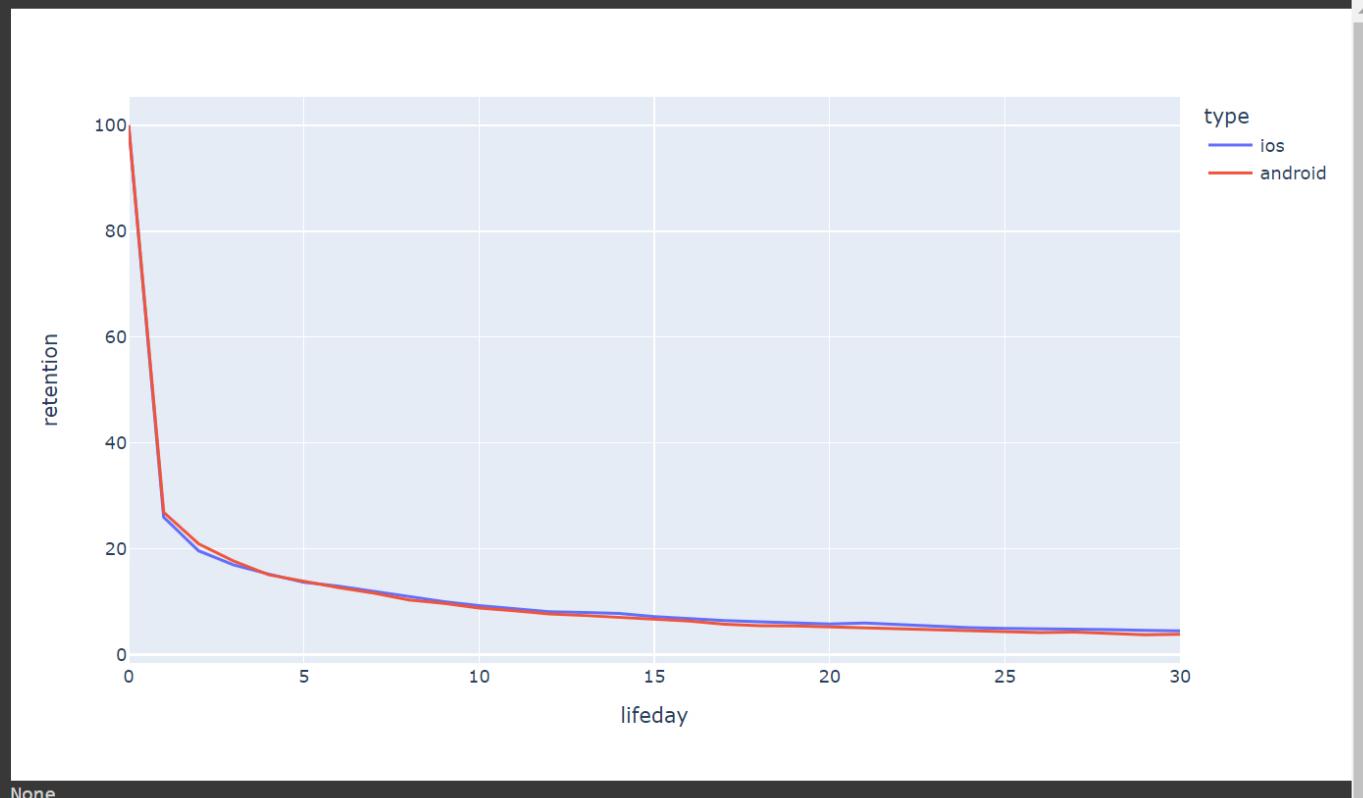
NA	2854
CIS	2071
EU	66
Name: region, dtype: int64	
NA	57.182929
CIS	41.494690
EU	1.322380

```
Name: region, dtype: float64
```

Хоть по европе данных и очень мало в процентах но все таки кажется что те пользователи что покупались (чего не скажешь про органических EU пользователей) в EU очень хорошо удерживаются относительно других, покрайней мере в первые 8 дней. Возможно была какая то специальная таргетированная компания именно в EU. Далее как всегда лидирует NA. Следом CIS.

#### ▼ Retention Rate y IOS / Android

```
[22] 1 retention_rate_30 = get_retention_rate(ios).assign(type='ios')
      2 retention_rate_30 = retention_rate_30.append(get_retention_rate(android).assign(type='android'))
      3 display(px.line(retention_rate_30, x='lifeday', y='retention', color='type').show())
      4 without_0 = retention_rate_30[retention_rate_30['lifeday'] != 0]
      5 display(px.line(without_0, x='lifeday', y='retention', color='type').show())
```



None



```
[23] 1 pre_retention.platform.value_counts()  
0  
сек.  
+ android 230797  
ios 191769  
Name: platform, dtype: int64
```

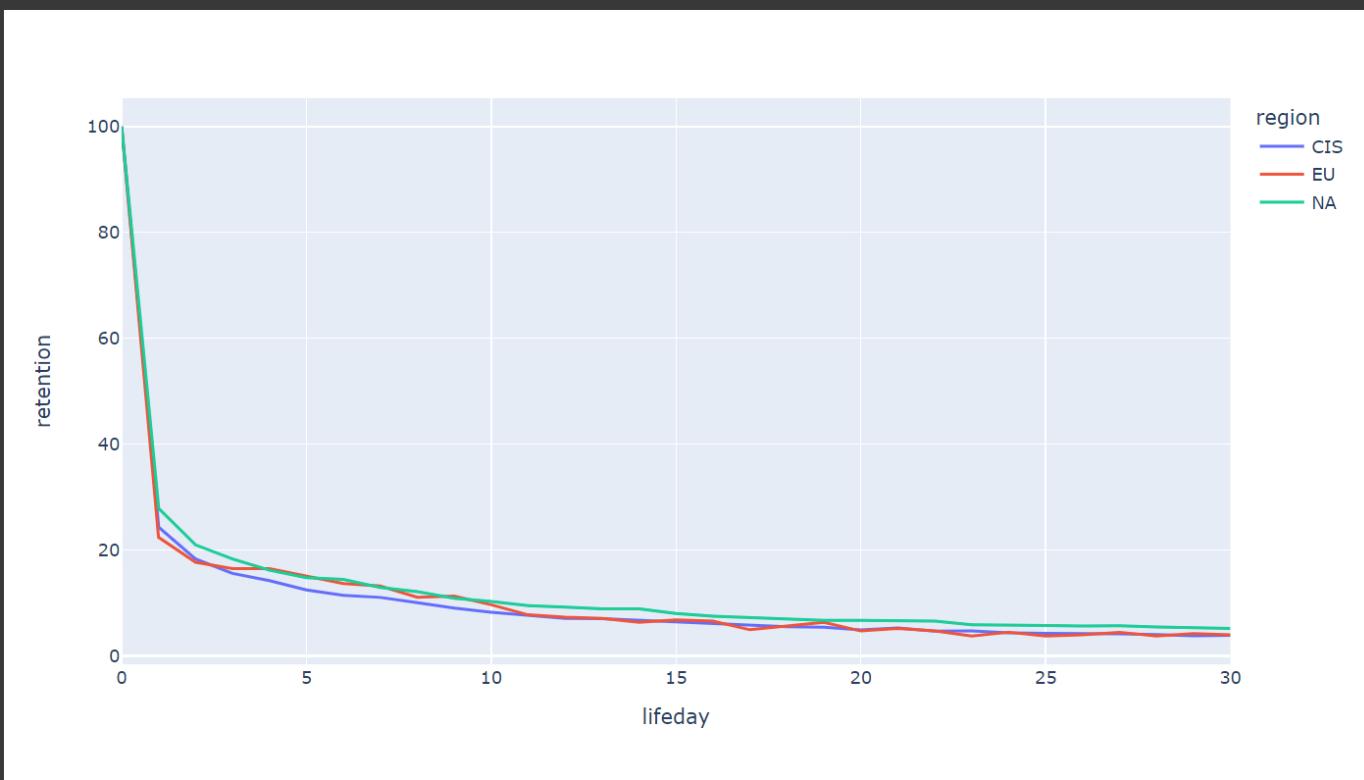
Retention Rate у Android в первые 3 дня заметно лучше чем у iOS.

На 4-6 день ретеншн у платформ примерно одинаковый.

Далее уже iOS показывает себя лучше чем Android.

Retention Rate для iOS

```
[24] 1 retention_rate_30_ios = get_retention_rate_per_region(ios)  
2 px.line(retention_rate_30_ios, x='lifeday', y='retention', color='region').show()
```



```
[25] 1 ios.region.value_counts()  
0  
сек.  
+ NA 101172  
CIS 86185  
EU 4412  
Name: region, dtype: int64
```

```
[26] 1 ios[ios.user_type=='non_organic'].region.value_counts()  
0  
сек.  
+ NA 31422  
CIS 22117  
EU 230  
Name: region, dtype: int64
```

```
[27] 1 android[android.user_type=='non_organic'].region.value_counts()  
0  
сек.  
+ NA 9166  
CIS 8108  
EU 435  
Name: region, dtype: int64
```

NA - лидер.

EU и CIS примерно равны

ЕС и ФРГ примерно равны.

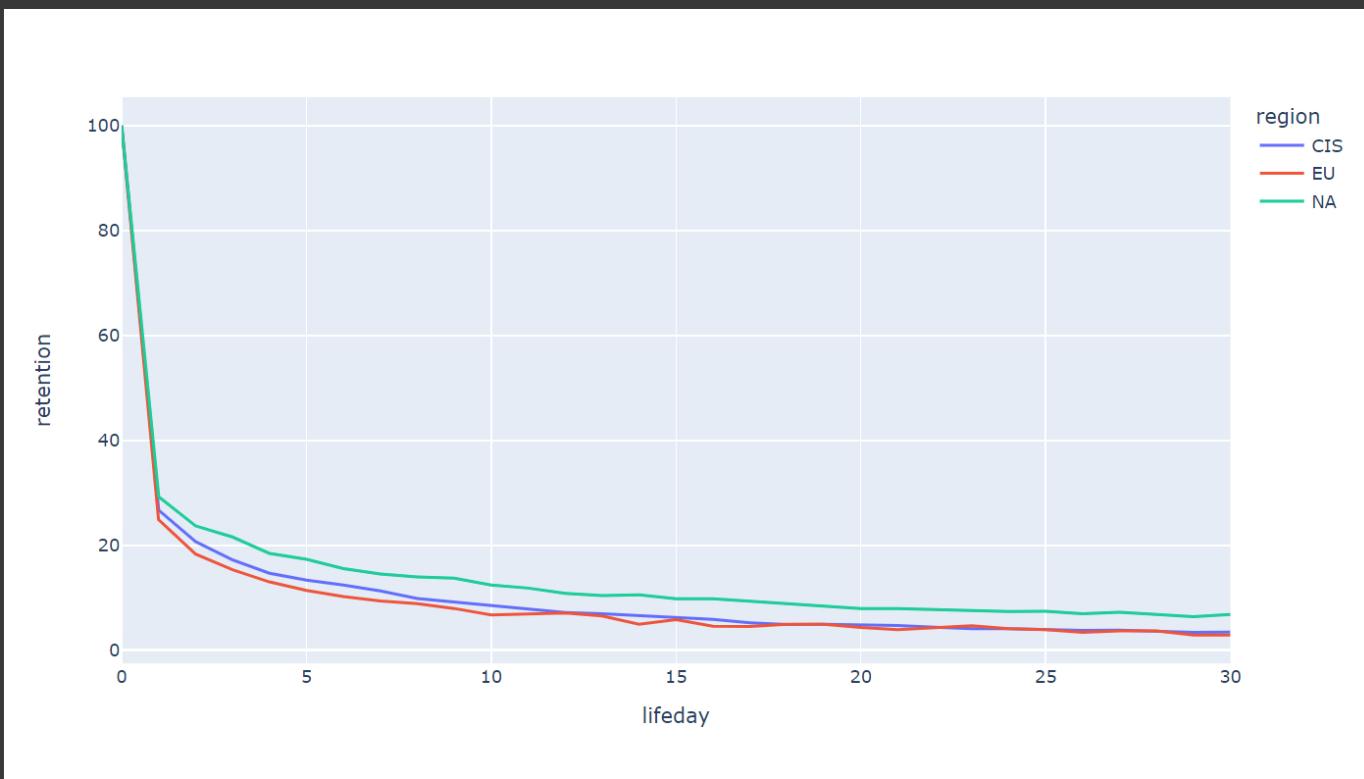
Но с 4 по 11 день EU чуть ли не вырывается вперед. Что как то странно.

В EU купленные 230 пользователей на IOS ведут себя слишком активно чем все остальные и искажают статистику европы.

Кто они такие?

#### ▼ Retention Rate Android

```
[28] 1 retention_rate_30_android = get_retention_rate_per_region(android)
2 px.line(retention_rate_30_android, x='lifeday', y='retention', color='region').show()
```



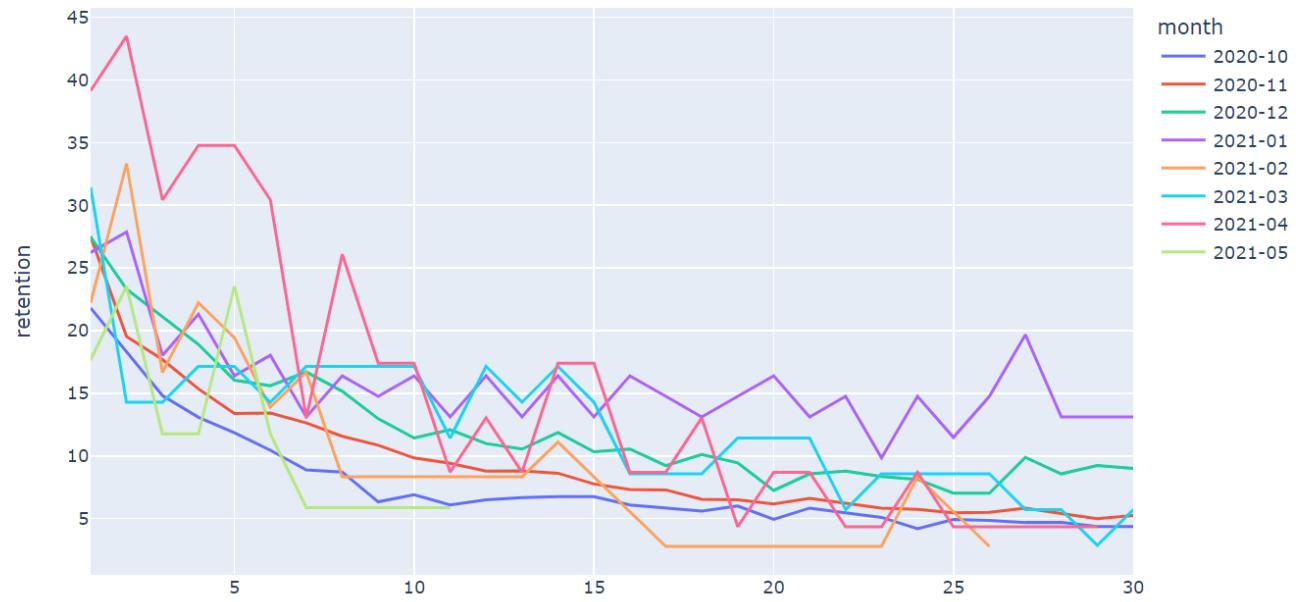
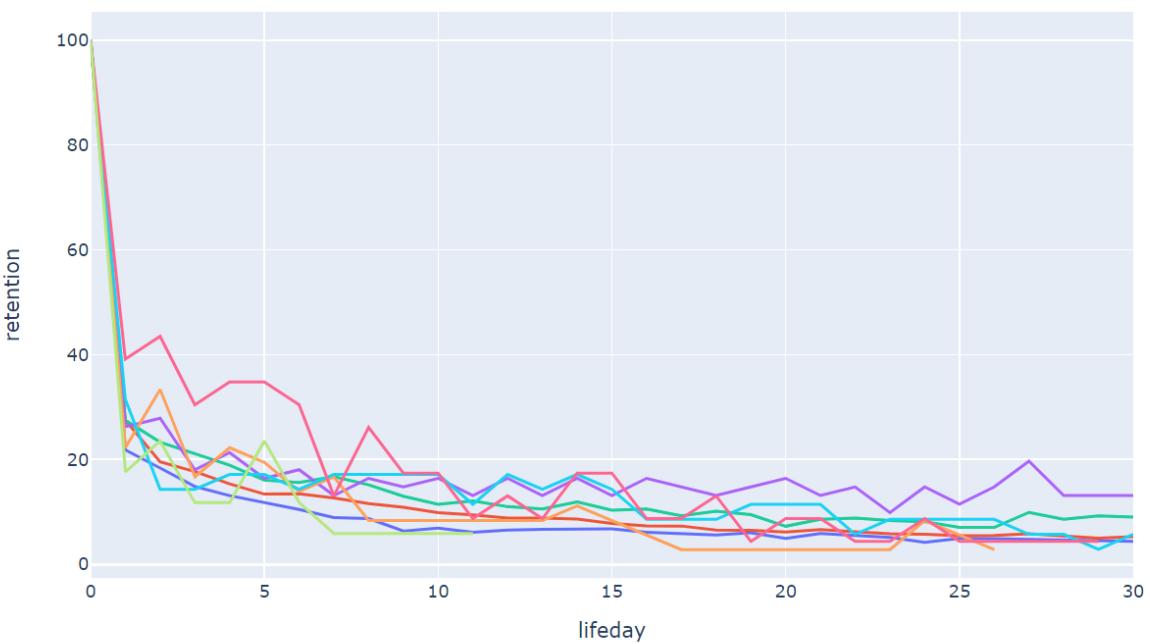
Как и везде НА лидер

CIS следом.

EU чуть хуже.

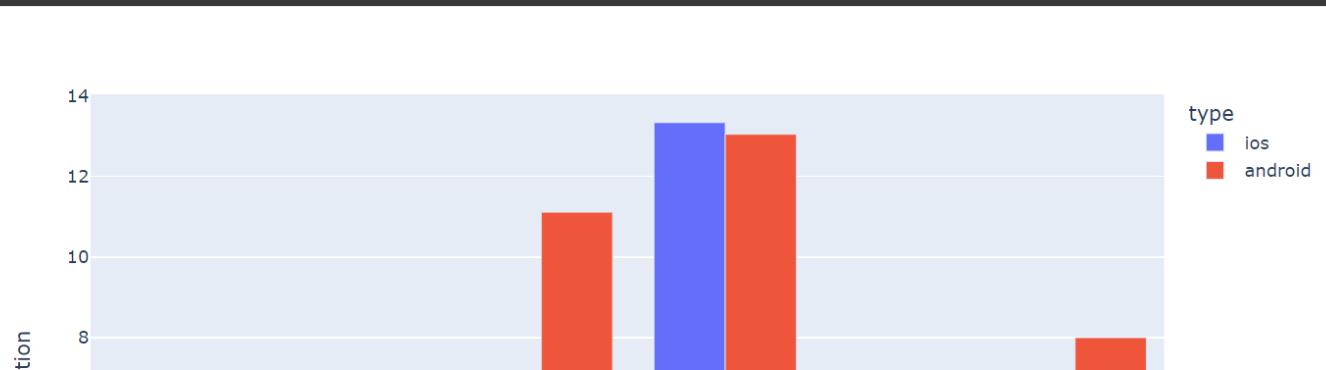
#### ▼ 3. Retention Rate помесячных когорт.

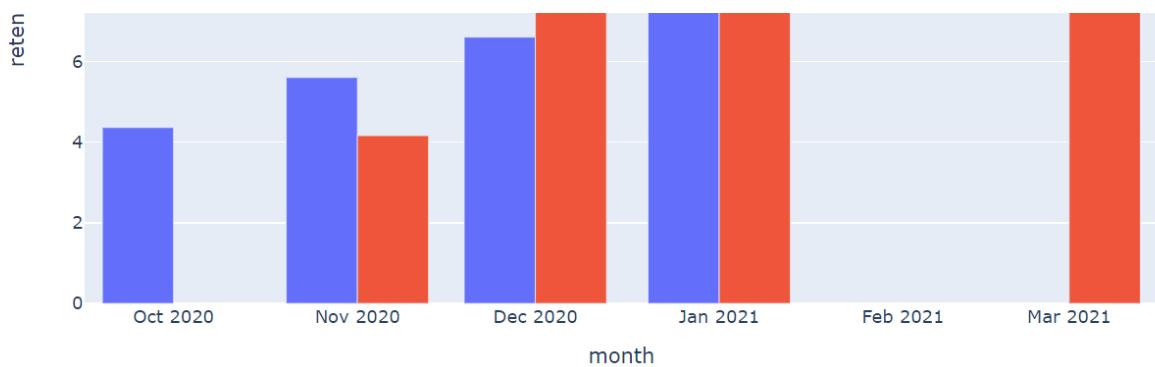
```
[29] 1 prepared = pre_retention[pre_retention.user_type=='non_organic']
2
3 ios = prepared[prepared.platform=='ios']
4 android = prepared[prepared.platform=='android']
5
6 def get_monthly_retention(data_frame):
7     result = pd.DataFrame([])
8     for name, group in data_frame.groupby(['install_month']):
9         result = result.append(get_retention_rate(group).assign(month=name))
10    return result
11
12 monthly_retention_rate_30 = get_monthly_retention(prepared)
13
14 display(px.line(monthly_retention_rate_30, x='lifeday', y='retention', color='month'))
15 without_0 = monthly_retention_rate_30[monthly_retention_rate_30['lifeday'] != 0]
16 display(px.line(without_0, x='lifeday', y='retention', color='month'))
```



#### ▼ Montply day30 by platform

```
[30]: 1 platform_retention = get_monthly_retention(prepared[prepared.platform=='ios']).assign(type='ios')
2 platform_retention = platform_retention.append(get_monthly_retention(prepared[prepared.platform=='android']).assign(
3 platform_retention = platform_retention[platform_retention.lifeday==30]
4 px.bar(platform_retention, x='month', y='retention', color='type', barmode='group')
```





В Oct 2020 никто из Android пользователей не появился на 30 день.

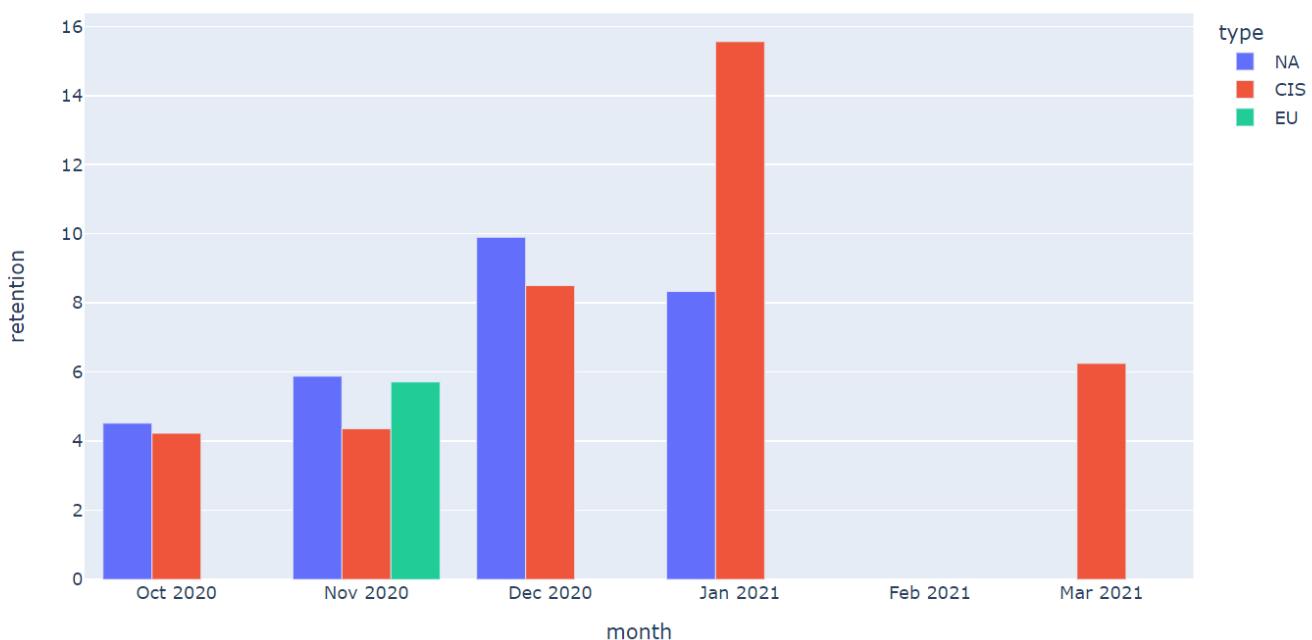
В Feb 2021 никто не доиграл до 30 дня.(Было мало пользователей?, пользователи были не вовлечены?)

В Mar 2021 никто из IOS не появился на 30 день.

В отдельные месяцы ведет IOS в отдельные Android. На глаз - Android удерживается чуть лучше.

#### ▼ Monthly day 30 Retention by region

```
[31] 1  region_retention = get_monthly_retention(prepared[prepared.region=='NA']).assign(type='NA')
      2  region_retention = region_retention.append(get_monthly_retention(prepared[prepared.region=='CIS']).assign(type='CIS'))
      3  region_retention = region_retention.append(get_monthly_retention(prepared[prepared.region=='EU']).assign(type='EU'))
      4  region_retention = region_retention[region_retention.lifeday==30]
      5  px.bar(region_retention, x='month', y='retention', color='type', barmode='group')
```



Пользователи из европы почти не заходят на 30 день.

Показатели за разные месяцы сильно разнятся.

С виду показатели у разных регионов не так сильно отличаются друг от друга.

Но вот в Jan 2021 показатели в CIS регионе сильно подскочили. (Возможно повлияли новогодние каникулы как вариант)

● ×