Тестовое задание

Татьяна Королева

Задание 1

```
In [1]: import pandas as pd
        spent = pd.read_sql_table('spent', 'sqlite:///db.db')
        payments = pd.read_sql_table('payments', 'sqlite:///db.db')
```

Найдем самые актуальные данные (с максимальным тайм-стемпом) для когорт, определенных по дате, операционной системе и странам. Сохраним результат в новую таблицу в нашу базу

In [4]:
```
mydata = pd.read_sql_query('SELECT p1.* FROM payments AS p1 INNER JOIN (SELECT DISTINCT MAX(ts) AS maxts, dat
e, country, os FROM payments GROUP BY date, country, os) AS p2 ON p1.date = p2.date AND p1.country = p2.count
ry AND p1.os = p2.os AND p1.ts = p2.maxts', 'sqlite:///db.db')
mydata
mydata.to_sql('payments2','sqlite:///db.db')
```

Out[4]:

	date	ts	os	country	purchases	unique_purchases	app_revenue
0	2020-07-25	2020-08-15 04:02:21.559178	android	PL	0	0	0.00000
1	2020-07-20	2020-08-15 04:02:21.559178	android	PT	1	1	3.51162
2	2020-06-13	2020-07-11 04:04:01.674296	android	GF	0	0	0.00000
3	2020-08-04	2020-08-15 04:02:21.559178	android	PT	0	0	0.00000
4	2020-08-05	2020-08-15 04:02:21.559178	android	KZ	5	1	9.56804
...	...	...	...	...	...	...	...
5431	2020-06-05	2020-07-03 04:04:41.964411	android	GR	0	0	0.00000
5432	2020-07-21	2020-08-15 04:02:21.559178	ios	CA	0	0	0.00000
5433	2020-07-09	2020-08-06 04:04:06.104637	android	KZ	0	0	0.00000
5434	2020-06-05	2020-07-03 04:04:41.964411	ios	IL	0	0	0.00000
5435	2020-06-07	2020-07-05 04:04:21.826532	android	BY	0	0	0.00000

5436 rows × 7 columns

Уберем пользователей, которые не совершали покупок, из нашей выборки, чтобы потом посчитать среднее. Также сохраним результат в новую таблицу

In [5]:
```
mydata2 = pd.read_sql_query('SELECT * FROM payments2 WHERE app_revenue > 0', 'sqlite:///db.db')
mydata2
mydata2.to_sql('payments_payingusers','sqlite:///db.db')
```

Out[5]:

	index	date	ts	os	country	purchases	unique_purchases	app_revenue
0	1	2020-07-20	2020-08-15 04:02:21.559178	android	PT	1	1	3.51162
1	4	2020-08-05	2020-08-15 04:02:21.559178	android	KZ	5	1	9.56804
2	7	2020-06-17	2020-07-15 04:03:57.658857	android	TR	3	1	4.22534
3	9	2020-06-07	2020-07-05 04:04:21.826532	android	ID	3	3	3.01564
4	14	2020-07-31	2020-08-15 04:02:21.559178	android	HU	1	1	1.19426
...	...	...	...	...	...	...	...	...
1115	5419	2020-08-12	2020-08-15 04:02:21.559178	ios	ES	3	1	5.26816
1116	5426	2020-06-04	2020-07-02 04:04:47.437432	android	HU	8	3	6.67350
1117	5427	2020-06-12	2020-07-10 04:04:04.953384	ios	AU	1	1	5.49057
1118	5429	2020-06-10	2020-07-08 04:04:14.834493	android	DE	11	3	35.84910
1119	5430	2020-06-02	2020-06-30 04:04:49.366921	android	HN	1	1	3.37000

1120 rows × 8 columns

И посчитаем средние доходы по android и ios, а затем запишем результат в текстовый документ.

In [6]:
```
avg_for_android = pd.read_sql_query('SELECT AVG(app_revenue) AS Average_revenue_android FROM payments_payingu
sers WHERE os = "android"', 'sqlite:///db.db')

avg_for_android
```

Out[6]:

	Average_revenue_android
0	46.545955

In [7]:
```
avg_for_ios = pd.read_sql_query('SELECT AVG(app_revenue) AS Average_revenue_ios FROM payments_payingusers WHE
RE os = "ios"', 'sqlite:///db.db')
avg_for_ios
```

Out[7]:

	Average_revenue_ios
0	77.787432

In [8]:
```
output = open('res1.txt', 'w')
output.write(str(avg_for_android.loc[0, 'Average_revenue_android']) + ' ' + str(avg_for_ios.loc[0, 'Average_r
evenue_ios']))
output.close()
```

Задача 2

Создадим таблицу с пользователями, которые используют приложение 28 дней (ts-date=28), и которые делают покупки в приложении

In [9]:
```
paymentsinfo = pd.read_sql_query('SELECT *, ROUND(julianday(ts)-julianday(date)) AS Number_of_diffdays FROM p
ayments WHERE ROUND (julianday(ts)-julianday(date)) = 28  AND app_revenue > 0', 'sqlite:///db.db')
paymentsinfo
paymentsinfo.to_sql('paymentsinfo','sqlite:///db.db')
```

Out[9]:

	date	ts	os	country	purchases	unique_purchases	app_revenue	Number_of_diffdays
0	2020-06-17	2020-07-15 04:03:57.658857	android	TR	3	1	4.22534	28.0
1	2020-06-07	2020-07-05 04:04:21.826532	android	ID	3	3	3.01564	28.0
2	2020-06-03	2020-07-01 04:05:00.527794	android	IT	8	2	16.76930	28.0
3	2020-06-09	2020-07-07 04:04:19.329535	android	PH	2	1	2.36278	28.0
4	2020-06-01	2020-06-29 04:05:14.890169	ios	RU	19	7	32.47890	28.0
...	...	...	...	...	...	...	...	...
824	2020-06-08	2020-07-06 04:04:24.707034	ios	SG	1	1	1.06476	28.0
825	2020-06-04	2020-07-02 04:04:47.437432	android	HU	8	3	6.67350	28.0
826	2020-06-12	2020-07-10 04:04:04.953384	ios	AU	1	1	5.49057	28.0
827	2020-06-10	2020-07-08 04:04:14.834493	android	DE	11	3	35.84910	28.0
828	2020-06-02	2020-06-30 04:04:49.366921	android	HN	1	1	3.37000	28.0

829 rows × 8 columns

Создадим дополнительную колонку, в которую запишем месяц из переменной date, чтобы в будущем помесячно посчитать коэффициент.

In [10]:
```
paymentsinfo['month'] = pd.DatetimeIndex(paymentsinfo['date']).month
paymentsinfo['month'].value_counts()
```

Out[10]:
```
6    732
7     97
Name: month, dtype: int64
```

In [11]:
```
paymentsinfo.to_sql('paymentsinfo1','sqlite:///db.db')
```

И посчитаем значения для каждой платформы отдельно. Затем, чтобы получить коэффициенты роста, вычтем из 6 месяца данные за 7 месяц.Получившиеся коффициенты для каждой платформы запишем в текстовый файл.

In [12]:
```
andr = pd.read_sql_query('SELECT p1.month, SUM (p1.app_revenue) / SUM (p2.spend) AS res FROM paymentsinfo1 AS p1 INNER JOIN spent AS p2 ON p1.date = p2.date AND p1.country = p2.country AND p1.os = p2.os WHERE p1.os = "android" GROUP BY p1.month', 'sqlite:///db.db')
andr
```

Out[12]:

	month	res
0	6	1.102008
1	7	0.710958

In [13]:
```
andr1 = andr.loc[1, 'res'] - andr.loc[0, 'res']
andr1
```

Out[13]:
```
-0.39104992557507645
```

In [14]:
```
ios = pd.read_sql_query('SELECT p1.month, SUM(p1.app_revenue) / SUM(p2.spend) AS res FROM paymentsinfo1 AS p1 INNER JOIN spent AS p2 ON p1.date = p2.date AND p1.country = p2.country AND p1.os = p2.os WHERE p1.os = "ios" GROUP BY p1.month ', 'sqlite:///db.db')
ios
```

Out[14]:

	month	res
0	6	1.028467
1	7	1.311157

In [15]:
```
ios1 = ios.loc[1, 'res'] - ios.loc[0, 'res']
ios1
```

Out[15]:
```
0.2826905951264189
```

In [16]:
```
output = open('res2.txt', 'w')
output.write(str(andr1) + ' ' + str(ios1))
output.close()
```

Задание 3

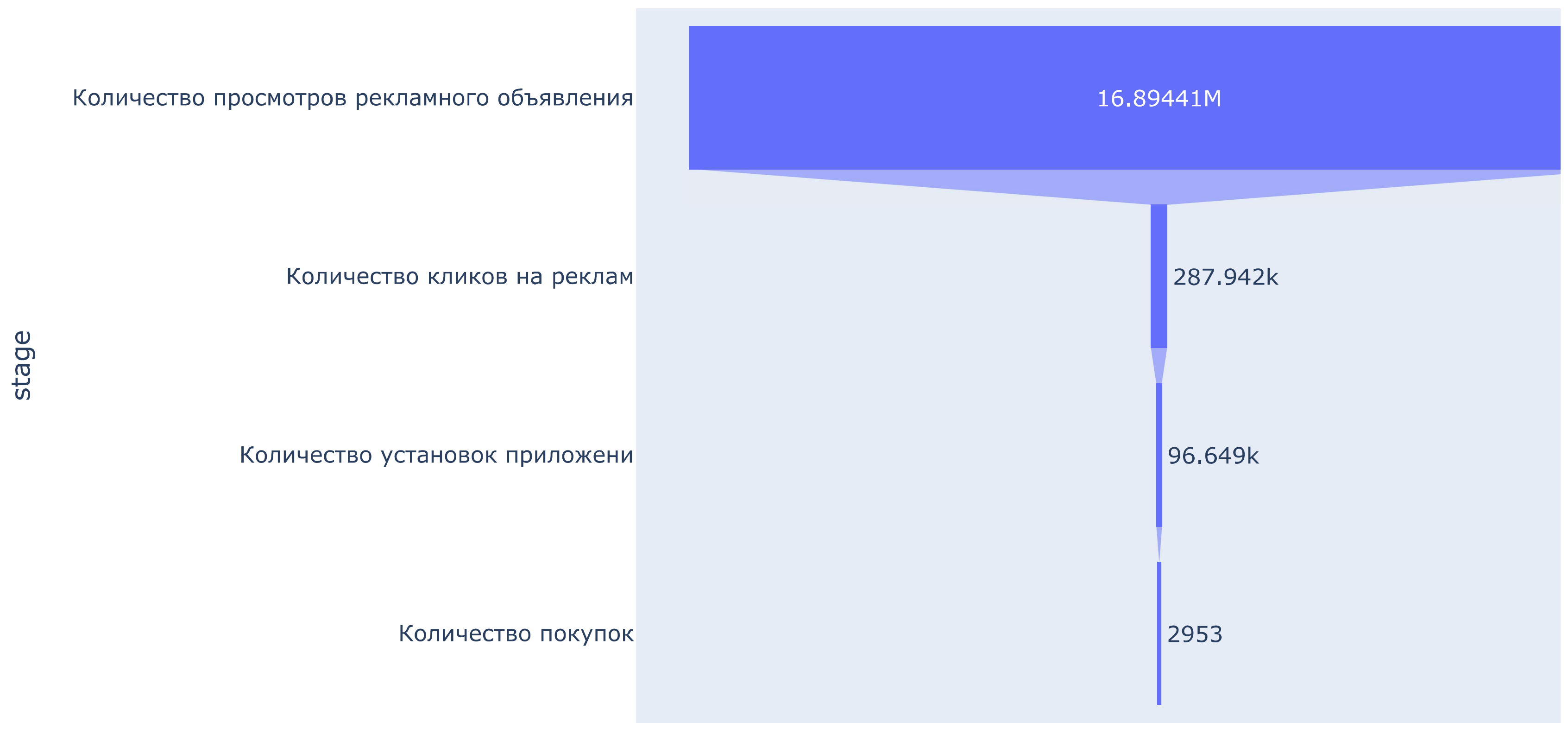
In [17]:
```
summm = pd.read_sql_query('SELECT SUM(impressions), SUM(clicks), SUM(installs), SUM(unique_purchases) FROM spe
nt AS p1 INNER JOIN payments2 AS p2 ON p1.date = p2.date AND p1.country = p2.country AND p1.os = p2.os', 'sql
ite:///db.db')
summm
```

Out[17]:

	SUM(impressions)	SUM(clicks)	SUM(installs)	SUM(unique_purchases)
0	16894414	287942	96649	2953

In [18]:
```
#!pip install plotly==4.11.0
```

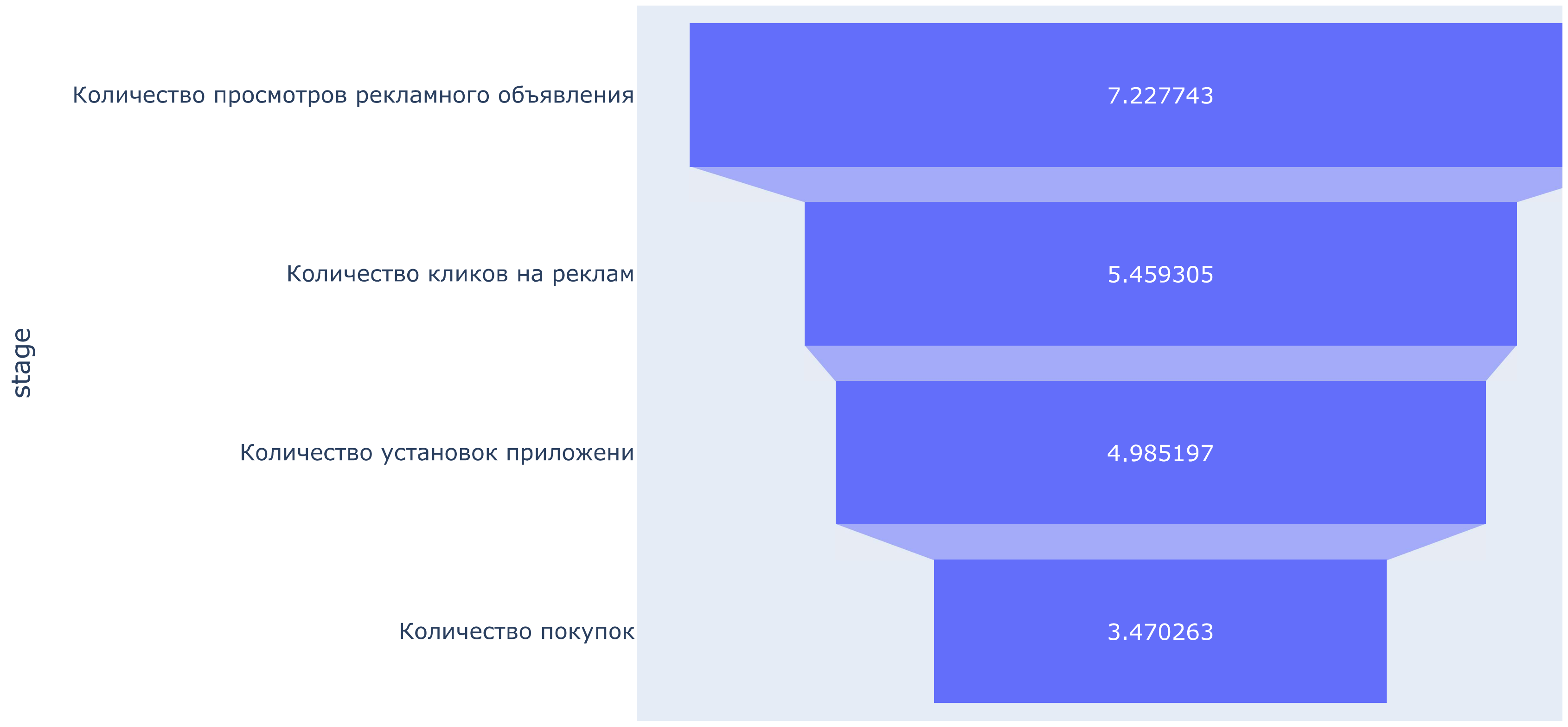
In [19]:
```
import plotly.express as px
data = dict(
    number= list(summ.loc[0, :]),
    stage=["Количество просмотров рекламного объявления", "Количество кликов на реклам", "Количество установо
к приложени", "Количество покупок"])
fig = px.funnel(data, x='number', y='stage')
fig.show()
```



In [20]:
```
import numpy as np
summ.loc[0, :] = np.log10(summ.loc[0, :])
summ.loc[0, :]
```

Out[20]:
```
SUM(impressions)         7.227743
SUM(clicks)              5.459305
SUM(installs)            4.985197
SUM(unique_purchases)    3.470263
Name: 0, dtype: float64
```

In [21]:
```
data = dict(
    number= list(summ.loc[0, :]),
    stage=["Количество просмотров рекламного объявления", "Количество кликов на реклам", "Количество установо
к приложени", "Количество покупок"])
fig = px.funnel(data, x='number', y='stage')
fig.show()
```


stage
Количество просмотров рекламного объявления
Количество кликов на реклам
Количество установок приложени
Количество покупок