Анализ источников трафика

Аннотация

1. Телеграм как источник показывает лучшие результаты по конверсии (0.06% против 0.04% для постов), а также лучшую вовлеченность (медиана кликов 23 против 13 для постов).
2. В то же время количество лидов в телеграме ниже (61 тыс. против 157 тыс.), меньше количество клиентов (4000 против 6000) и меньше средний депозит (385 USD против 560 USD).
3. Около 2/3 трафика и денег получены с постов.
4. Можно выделить 3 ведущих канала: Facebook, SMM, директ.
5. И две ведущие страны: Испания (ES) и Германия (DE).
6. Я рекомендую провести пару дополнительных исследований, чтобы выявить тренд по рекламе в Телеграме, а также изучить детальней ситуация по Франции.
7. До этого времени рекомендую придерживаться текущей стратегии.
8. В случае, если необходимо выбрать точки приложения усилий, то стоит вкладываться в кагалы Facebook, SMM, директ в странах Испания (ES) и Германия (DE).

Общая информация

- **Заказчик:** отдел маркетинга
- **Цель исследования:** оптимизировать маркетинговую активность.
- **Задачи исследования:**
    - провести исследование данных;
    - выделить лучшие источники и каналы;
    - сформулировать рекомендации по маркетинговой активности.
- **Этапы исследования:**
    - предобработка данных: чистка пропусков, дубликатов, аномальных значений;
    - исследовательский анализ данных и визуализация.
    - выводы.

Описание данных:

1. файл "synthetic_data":
    a. depo — сумма депозита, USD;
    b. segment/source – источник трафика, есть 2 вариант источников (посты и телеграм каналы):
        i. "postid" — лид перешел со статьи, id поста не имеет значения,
        ii. "telegram" — лид пришел из телеграма;
    c. channel — канал трафика, например, пользователь пришел из ресурса 'telegram' и через партнерский 'affiliate' канал;
    d. clicks — количество кликов, которые пользователь сделал в течение первого дня после регистрации;
    e. latency — время загрузки приложения в миллисекундах;
    f. client_id — присваивается во время регистрации и больше не меняется;
2. файл "country":
    a. страна of lead/client (iso2);
    b. client_id — присваивается во время регистрации и больше не меняется.

Преобразование данных

```
!pip install -U pandas
!pip install -U plotly
```

```
Requirement already satisfied: pandas in c:\users\acer\anaconda3\lib\site-packages (1.4.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\acer\anaconda3\lib\site-packages (from
pandas) (1.20.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\acer\anaconda3\lib\site-packages
(from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\acer\anaconda3\lib\site-packages (from
pandas) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\acer\anaconda3\lib\site-packages (from python-
dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: plotly in c:\users\acer\anaconda3\lib\site-packages (5.8.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\acer\anaconda3\lib\site-packages (from
plotly) (8.0.1)
```

```
# импортируем библиотеки
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

```
# сохраним имена файлов в переменные
a = 'synthetic_data'
b = 'countries'
```

```
# прочитаем файлы
data = pd.read_csv(a + '.csv')
countries = pd.read_csv(b + '.csv')
```

Файл ‘synthetic_data’. Изучение

В качестве первого шага запросим общую информацию о датасете.

```
# используем метод info()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 219314 entries, 0 to 219313
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Unnamed: 0  219314 non-null  int64
 1   depo        219314 non-null  int64
 2   segment     219314 non-null  object
 3   channel     217142 non-null  object
 4   clicks      219314 non-null  float64
 5   latency     219314 non-null  float64
 6   client_id   219314 non-null  int64
dtypes: float64(2), int64(3), object(2)
memory usage: 11.7+ MB
```

```
# проверим наличие пропусков
print('NA amount:')
print(data.isnull().sum())
```

```
NA amount:
Unnamed: 0    0
depo          0
segment       0
channel    2172
clicks        0
latency       0
client_id     0
dtype: int64
```

```
# и их долю
print('NA share: {:.2%}'.format(data['channel'].isnull().sum() / len(data)))
```

```
NA share: 0.99%
```

В датасете 219314 строка, только одна колонка — channel — содержит пропуски, доля которых 1%. Информация не может быть восстановлена по данным других колонок. Колонка 'segment' заполнена для этих строк, поэтому мы можем использовать данные с пропусками для исследования источников трафика. Удалять строки с пропусками не будем. Есть избыточная колонка 'Unnamed', которую лучше удалить. Также тип данных колонки 'clicks' не соответствует содержанию: информация о кликах должна быть целочисленной. Попробуем изменить тип столбца.

```
# удаляем избыточную колонку и изменяем тип данных, после преобразований выведем информацию
data_cl = data.drop(columns=['Unnamed: 0'], axis = 1)
data_cl['clicks'] = data_cl['clicks'].astype('int64')
data_cl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 219314 entries, 0 to 219313
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   depo       219314 non-null  int64  
 1   segment    219314 non-null  object 
 2   channel    217142 non-null  object 
 3   clicks     219314 non-null  int64  
 4   latency    219314 non-null  float64
 5   client_id  219314 non-null  int64  
dtypes: float64(1), int64(3), object(2)
memory usage: 10.0+ MB
```

Все изменения проведены успешно. Посмотрим внимательней: есть ли дубликаты в датасете и изучим визуальный пример данных.

```
# объявим функцию, которая покажет количество дубликатов, долю, и первые 5 строк датасета
def check(data):
 display(data.head())
 duplicates = data.duplicated().sum()
 duplicates_part = duplicates / len(data)
 print('Duplicated lines, amount:', duplicates)
 print('Duplicated lines, share: {:.2%}'.format(duplicates_part))
```

```
# применим функцию к датасету
check(data_cl)
```

	depo	segment	channel	clicks	latency	client_id
0	0	postid_4057	smm	1	2.649725	1442498
1	0	telegram	affiliate	10	2.610846	7865631
2	0	postid_8542	facebook	13	3.001162	8165584
3	0	telegram	direct	0	1.788369	5893056
4	0	telegram	smm	0	1.932069	3780924

```
Duplicated lines, amount: 0
Duplicated lines, share: 0.00%
```

В датасете нет дубликатов. Типы колонок соответствуют содержанию. Удалим избыточную информацию из колонки segment: согласно описанию, есть только 2 источника — telegram и посты, номера постов не имеют значения.

```
# разделим колонку segment, удалим избыточные данные, переименуем колонку
m = data_cl['segment'].str.split('_',expand=True)
m.columns=['segment','for_dropping']
m = m.drop(columns=['for_dropping'], axis = 1)
# удалим исходный столбец
data_up = data_cl.drop(columns=['segment'], axis = 1)
# объединим полученные датасеты
data_up = pd.concat([data_up, m], axis=1)
# проверим, верно ли отработали преобразования
data_up.groupby('segment')['segment'].count()
```

```
segment
postid   157560
telegram   61754
Name: segment, dtype: int64
```

Первый файл подготовлен для анализа. Изучим данные во втором.

Файл 'countries'. Изучение

```
# запросим общую информацию о файле
countries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 219314 entries, 0 to 219313
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype 
---  ------    --------------   ----- 
 0   country   219314 non-null object
 1   client_id 219314 non-null int64 
dtypes: int64(1), object(1)
memory usage: 3.3+ MB
```

В данном файле нет пропусков. С помощью ранее написанной функции проверим, есть ли дубли.

```
# используем написанную функцию
check(countries)
```

	country	client_id
0	IN	6348826
1	FR	6751691
2	DE	8638448
3	LT	4722696
4	ES	2411132

```
Duplicated lines, amount: 61754
Duplicated lines, share: 28.16%
```

Почти 30% данных составляют дубли. Общее число строк одинаковое для двух файлов. Есть вероятность, что в первом файле несколько строк с одним и тем же id имеют разные источники. Перепроверим колонку client_id на дубли (в первой проверке мы искали полностью дублирующиеся строки).

```
# посчитаем количество уникальных значений
data_up['client_id'].nunique()
```

```
219314
```

Ранее сделанное предположение не подтвердилось: файл 'synthetic_data' содержит только уникальные id клиентов. Это значит, что для некоторых клиентов не будет записей о стране. Для избежания преувеличения долей клиентов по странам удалим дублирующиеся строки из датасета.

```
# используем метод drop_duplicates()
countries_cl = countries.drop_duplicates()
countries_cl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 157560 entries, 0 to 219313
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype 
---  ------    --------------   ----- 
 0   country   157560 non-null  object
 1   client_id 157560 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 3.6+ MB
```

Теперь объединим два датасета. Пустые ячейки заменим на значение ‘nd’ = нет данных.

```
# используем метод merge()
df = data_up.merge(countries_cl, on='client_id', how='left').fillna('nd')
df.sample(10)
```

	depo	channel	clicks	latency	client_id	segment	country
167023	0	smm	0	3.709683	8177718	telegram	nd
209575	0	social media	1	4.045738	5746787	telegram	nd
5964	0	direct	27	3.164631	7257175	telegram	IS
169211	0	direct	9	2.887288	8362207	postid	nd
182651	0	social media	31	3.259292	3477615	postid	nd
46135	0	facebook	0	2.215026	8647832	telegram	US
141455	0	smm	6	2.672162	3472960	postid	IS
65873	0	social media	0	2.556720	3228783	postid	LT
68923	0	smm	2	3.503075	4537987	postid	US
49609	0	social media	0	2.982904	5768490	postid	IN

```
# проверим, верно ли отработали замены
print('NA in the column "country":', df.query('country == "nd"')['country'].count())
```

```
NA in the column "country": 61754
```

Количество пропусков в столбце country полностью совпадает с количеством значений источника telegram. Проверим, не получилось ли так, что все записи про телеграм не имеют данных о стране.

```
# посчитаем пропуски в разрезе источников
df.query('country == "nd"').groupby('segment')['segment'].count()
```

```
segment
postid   44401
telegram  17353
Name: segment, dtype: int64
```

Итого 61754 строк не содержат данных о стране, но пропуски есть в обоих источниках. На текущим этапе подготовка файлов завершена. Перейдем к исследовательскому анализу данных.

Исследовательский анализ данных

Первым делом изучим статистические характеристики численных параметров.

```
# используем метод describe()
df[['depo', 'clicks', 'latency']].describe()
```

	depo	clicks	latency
count	219314.000000	219314.000000	219314.000000
mean	22.361217	11.430114	3.021579
std	397.835611	12.628842	1.048472
min	-164.000000	0.000000	0.000071
25%	0.000000	0.000000	2.320726
50%	0.000000	8.000000	3.001301
75%	0.000000	19.000000	3.693649
max	31675.000000	50.000000	11.016521

Данные колонки 'latency' не вызывают вопросов: среднее значение = 3, стандартное отклонение = 1: можно сказать, что данные сосредоточены вокруг среднего. Бо**л**ьшую вариативность демонстрирует колонка 'clicks': стандартное отклонение = 12.6 и среднее = 11.4, что может объясняться большим количеством нулевых значений: 25% квартиль представляет собой 0. То есть как минимум четверть пользователей не совершают ни одного клика в первый день после регистрации. Максимальное количество кликов не вызывает вопросов, число 50 выглядит вполне правдоподобно. Колонка с данными о депозитах содержит неожиданные отрицательные данные. Также стоит отметить, что 75% квартиль составляет 0. Посмотрим, какова доля отрицательных значений.

```
# выберем только строки с отрицательным значениями
negative_depo = df.query('depo < 0')

# посчитаем количество и долю этих значений
print('Negative depo, amount:', negative_depo['depo'].count())
print('Negative depo, share: {:.2%}'.format(negative_depo['depo'].count() / len(df)))
```

```
Negative depo, amount: 108
Negative depo, share: 0.05%
```

Отрицательные депозиты составляют всего лишь 0.05% от всех значений, поэтому мы удалим строки с этими записями из датасета. У меня нет предположений относительно природы возникновения ошибки, поэтому имеет смысл создать тикет для технической команды (вся доступная информация по событию сохранена в 'negative_depo').

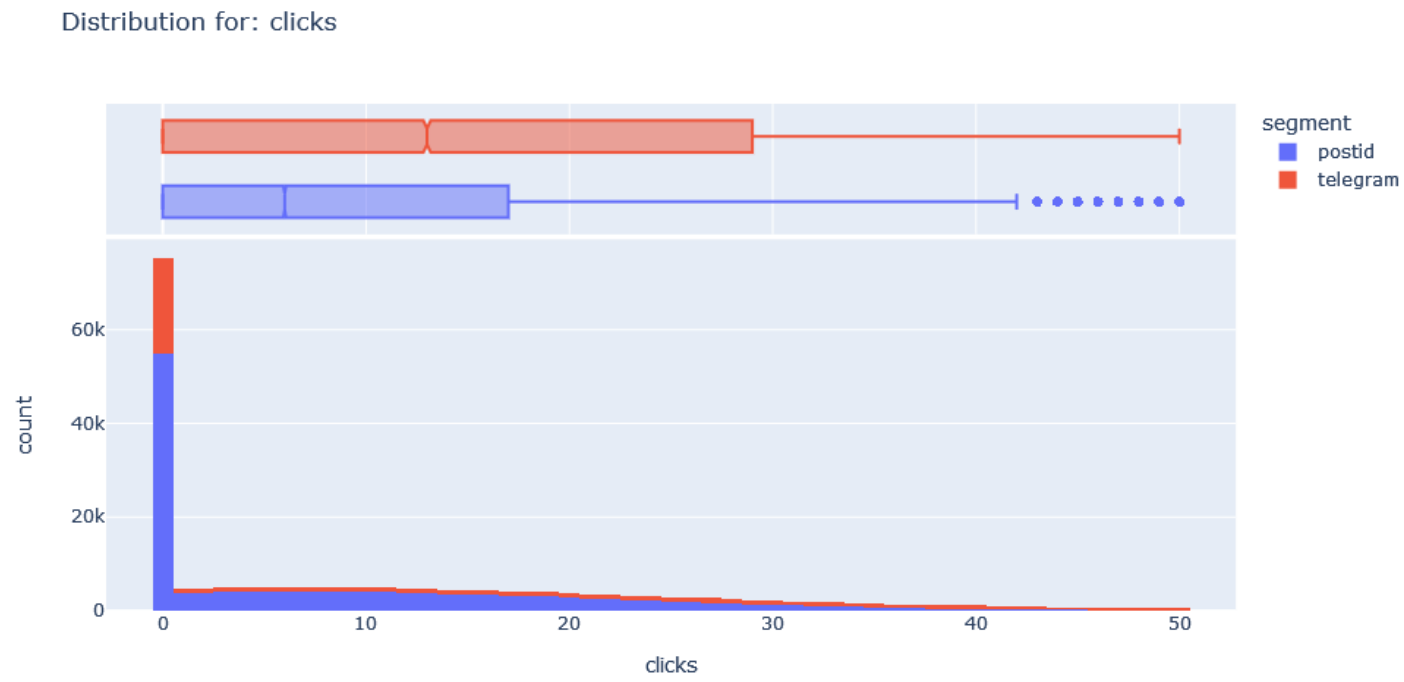
```
# очищенный датасет
df_up = df.query('depo >= 0')
df_up.info()
```

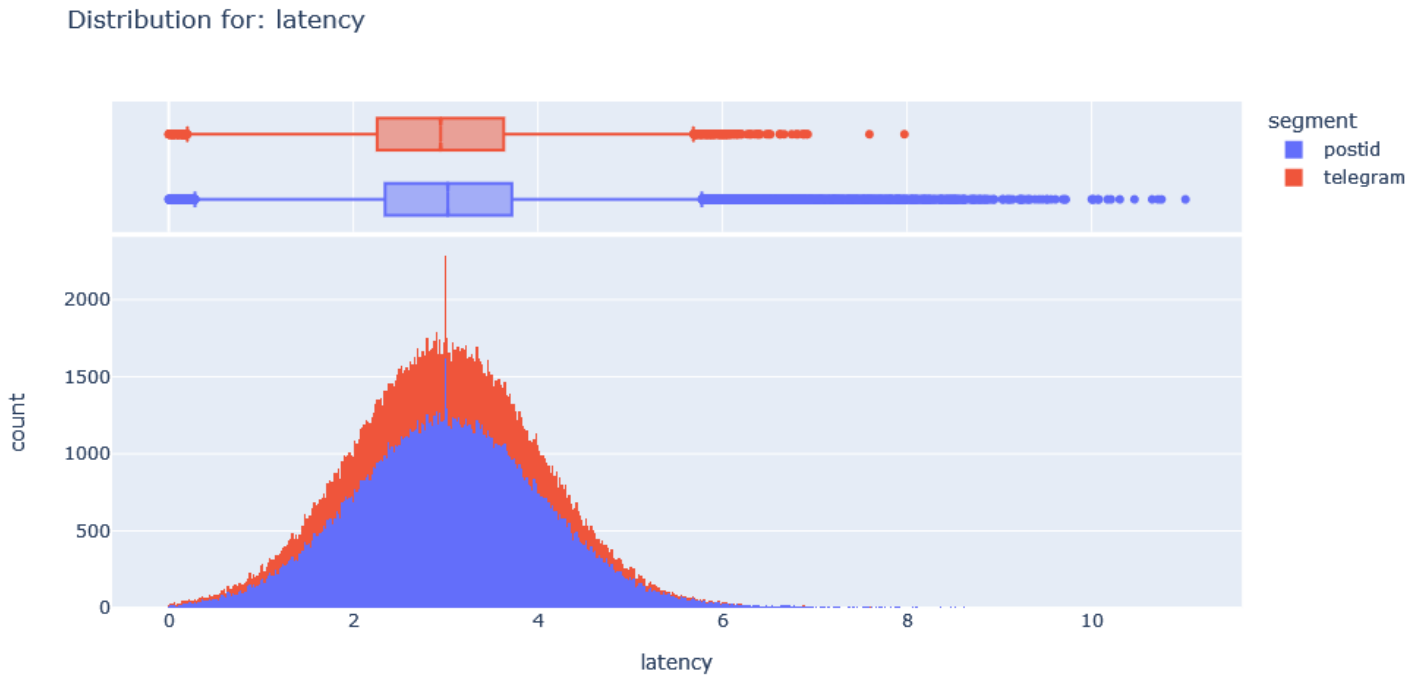
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 219206 entries, 0 to 219313
Data columns (total 7 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   depo       219206 non-null  int64  
 1   channel    219206 non-null  object 
 2   clicks     219206 non-null  int64  
 3   latency    219206 non-null  float64
 4   client_id  219206 non-null  int64  
 5   segment    219206 non-null  object 
 6   country    219206 non-null  object 
dtypes: float64(1), int64(3), object(3)
memory usage: 13.4+ MB
```

В новом датасете 219206 строки. Построим графики распределения числовых параметров, кроме депозита, поскольку ¾ значений в нем нулевые, график будет непоказательным.

```
# датасет без client_id
for_graph = df_up[['clicks', 'latency', 'segment']]

# построим график для всех колонок датасета
for col in for_graph.drop(columns=['segment']).columns:
  fig = px.histogram(for_graph, x = col, marginal = 'box', color = 'segment', title = 'Distribution for: '+col)
  fig.show()
```

изначальный язык написания отчета английский, поэтому графики имеют английские подписи





```
# проверим распределение кликов за исключением нулевых значений
fig = px.histogram(df_up.query('clicks != 0'), x = ['clicks'], marginal = 'box', color = 'segment',
        color_discrete_map={
       'telegram': '#EF553B',
       'postid': '#636EFA'},
        title = 'Distribution for: clicks <> 0')
fig.show()
```

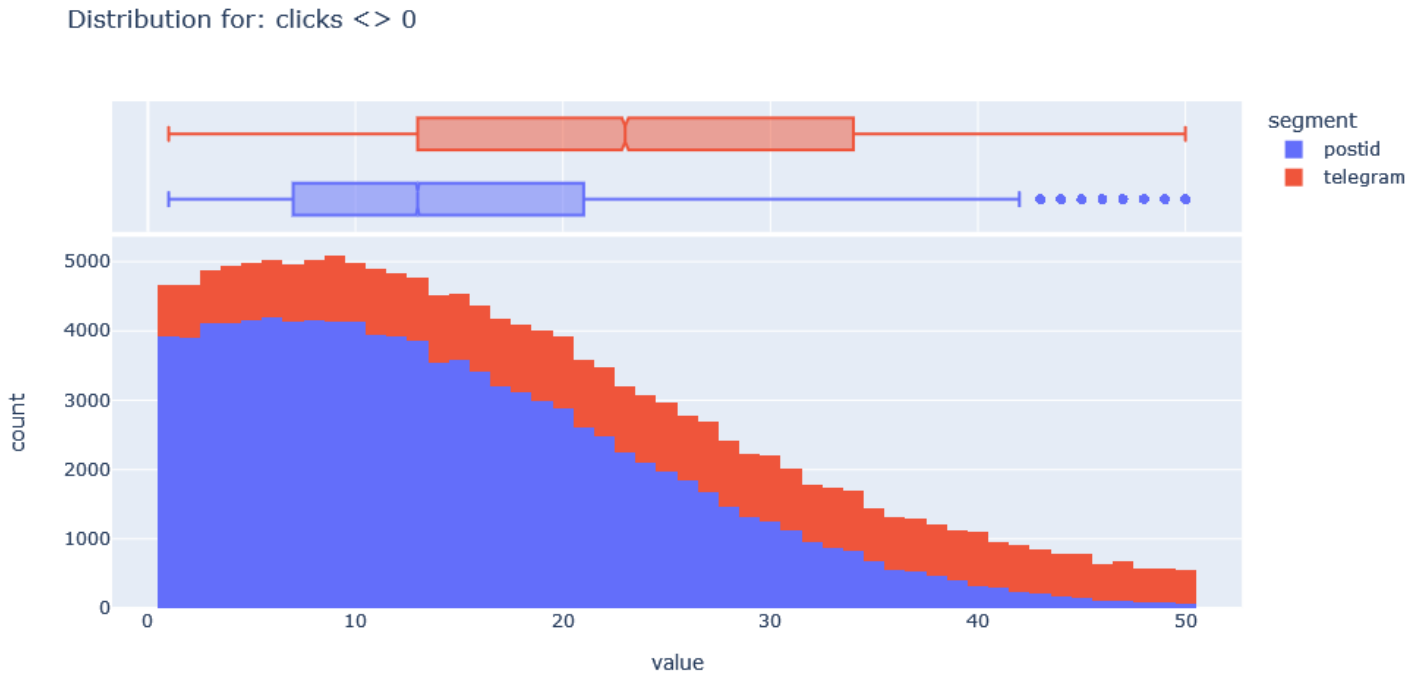


График распределения для параметров 'clicks' и 'latency' совпадает с ожидаемым. Из графиков следует:

- клики, все данные: медиана для телеграма (13) больше, чем для постов (6);
- клики, все данные: 75% квартиль для телеграма также выше (29 против 17);
- скорость загрузки чуть лучше в случае телеграма тоже: 2.95 против 3.02;
- телеграм в 2.5 уступает по числу пользователей: 61754 против 157560 для постов.

В колонке скорость загрузки есть выбросы для обоих источников в районе показателя 3 секунды. Значение близко к медиане, поэтому выглядит как результат предыдущей обработки данных, делать с этими выбросами ничего не будем. Построим график для колонки депозитов без нулевых значений.

```
# построим график и зададим привычные цветовой код для источников
fig = px.histogram(df_up.query('depo != 0'), x = ['depo'], marginal = 'box', color = 'segment',
        color_discrete_map={
       'telegram': '#EF553B',
       'postid': '#636EFA'},
        title = 'Distribution for: deposits <> 0')
fig.show()
```

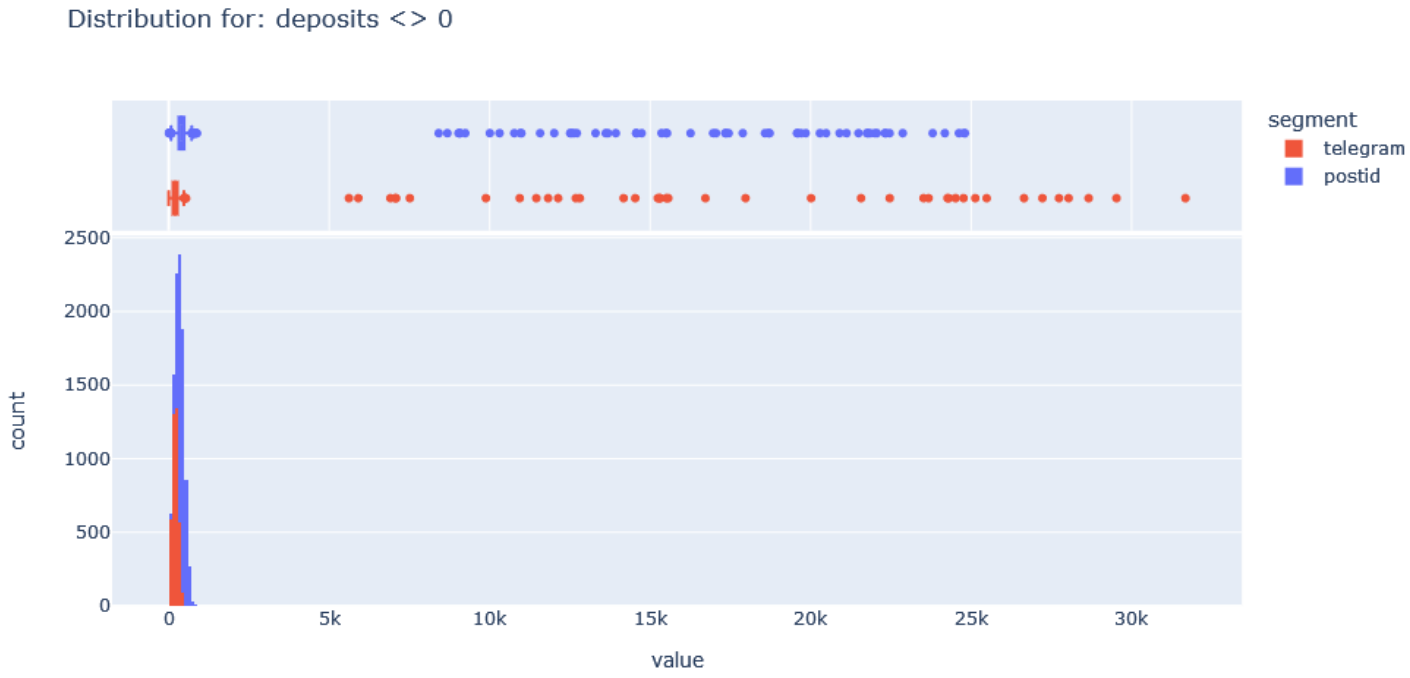
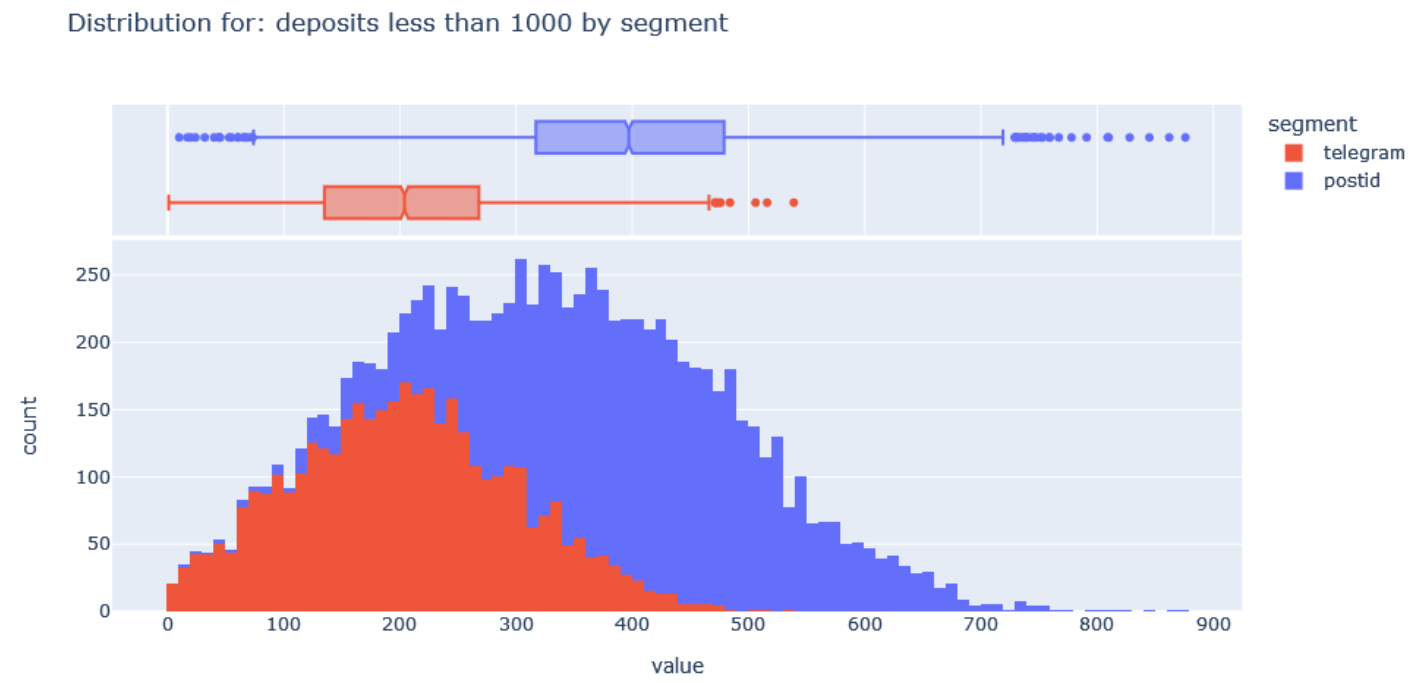


График даже без нулевых значений непоказательный. Можно сделать вывод, что большинство значений не превышают 700 USD. Достоверных предположений о выбросах я не могу сделать — возможно, депозиты в 10 тыс. или 30 тыс. являются нормой бизнеса. Корректировать данные не будем. Построим график распределения для данных от 0 до 700 USD.

```
# построим график, сохраняя цветовую кодировку
fig = px.histogram(df_up.query('depo != 0 and depo < 1000'), x = ['depo'], marginal = 'box', color = 'segment',
        color_discrete_map={
       'telegram': '#EF553B',
       'postid': '#636EFA'},
        title = 'Distribution for: deposits less than 1000 by segment')
fig.show()
```



Распределение имеет ожидаемую форму.

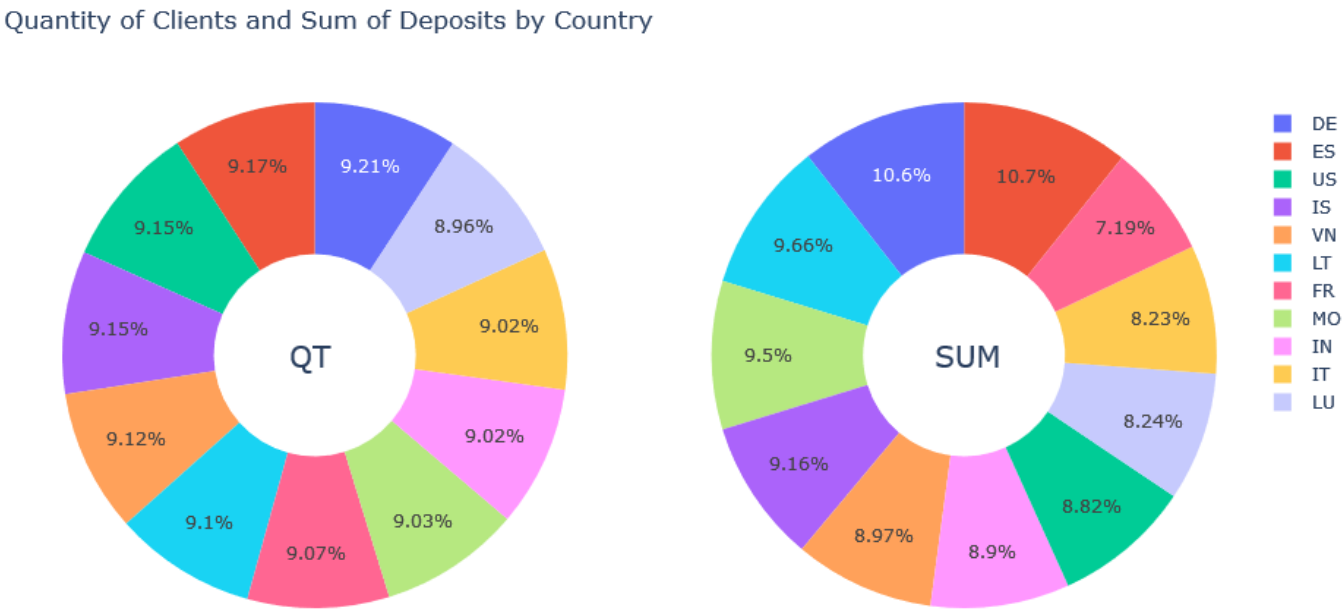
- у постов выше медиана (379 USD против 204 USD в телеграме);
- в целом объем депозитов, полученные через посты, больше, чем через телеграм.

Посмотрим, как депозиты и пользователи распределены по странам.

```
# объявим функцию для построения круговых диаграмм
def pie_maker(data, col_1, col_2, col_3, title):
 labels = col_1
 fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
 fig.add_trace(go.Pie(labels=labels, values=col_2, name='clients'),
     1, 1)
 fig.add_trace(go.Pie(labels=labels, values=col_3, name='deposit'),
     1, 2)
 fig.update_traces(hole=.4, hoverinfo='label+percent+name')
 fig.update_layout(
  title_text=title,
  annotations=[dict(text='QT', x=0.20, y=0.5, font_size=20, showarrow=False),
           dict(text='SUMM', x=0.81, y=0.5, font_size=20, showarrow=False)])
 fig.show()
```

```
# создадим датасет без пропусков в графе «страна»
pie = df_up.query('country != "nd"').groupby('country').agg({'depo': 'sum', 'client_id':'count'})\
    .reset_index().rename(columns={'depo': 'sum', 'client_id':'quantity'})
```

```
pie_maker(pie, pie['country'], pie['quantity'], pie['sum'], 'Quantity of Clients and Sum of Deposits by Country')
```



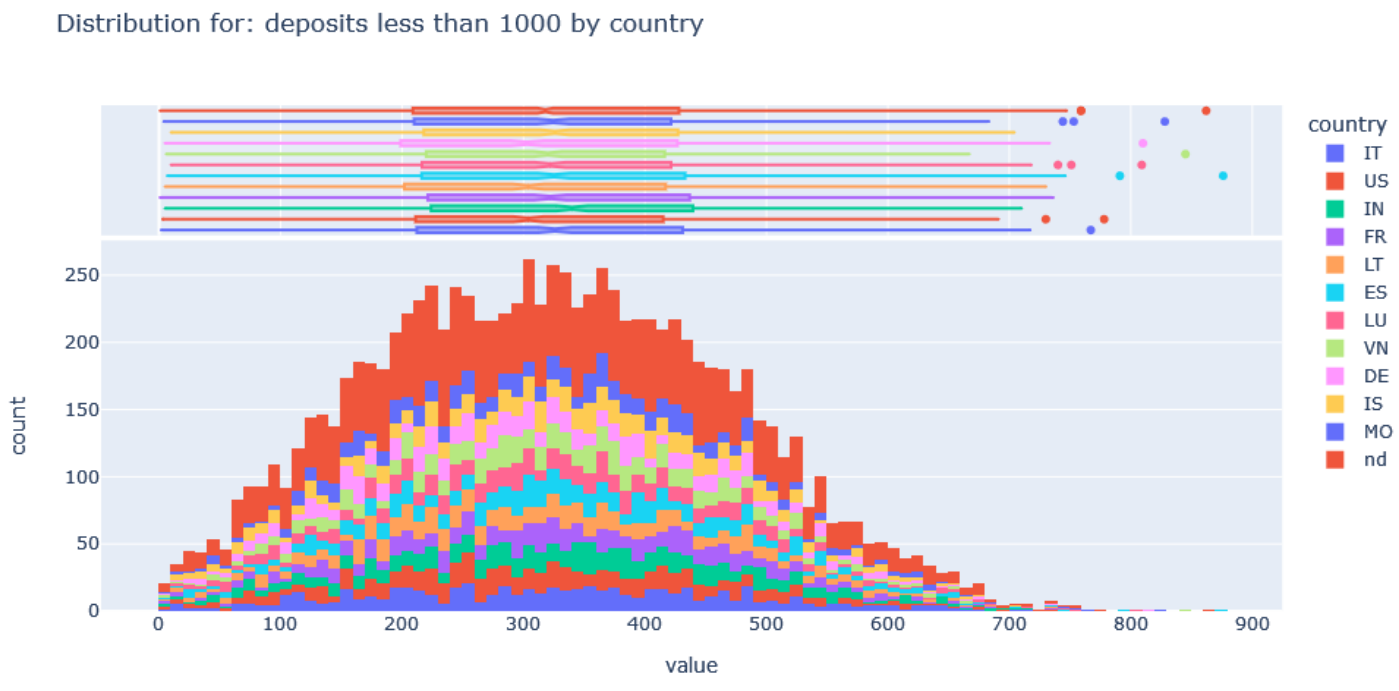
На графике видно, что клиенты количественно довольно равномерно распределены по странам: минимальная доля страны 8.96%, максимальная 9.21%. Разница по сумме депозитов выше: от 7.19% до 10.7%. Посмотрим на среднюю сумму депозита в разрезе стран.

```
# построим таблицу на основании ранее созданного датасета
pie['avg_depo'] = pie['sum'] / pie['quantity']
pie['avg_depo'] = pie['avg_depo'].apply(lambda x: '{:.2f}'.format(x))
pie.sort_values(by='avg_depo', ascending=False)
```

	country	summ	quantity	avg_depo
1	ES	376270	14441	26.06
0	DE	370878	14501	25.58
6	LT	338112	14332	23.59
8	MO	332463	14220	23.38
4	IS	320635	14403	22.26
3	IN	311522	14209	21.92
10	VN	313932	14367	21.85
9	US	308791	14417	21.42
7	LU	288583	14111	20.45
5	IT	288078	14201	20.29
2	FR	251849	14288	17.63

Существует практически 50% разница между страной с самым высоким средним депозитом (ES=26) и страной с самым низким (FR=17.6). Проверим, является ли это следствием пары очень больших депозитов.

```
# объявим функцию для построения графика
def hist_maker(data, col_1, title):
  fig = px.histogram(data, x = ['depo'], marginal = 'box', color = col_1, title = title)
  fig.show()
# подготовим датасет
d = df_up.query('depo != 0 and depo < 1000')
hist_maker(d, d['country'], 'Distribution for: deposits less than 1000 by country')
```



Распределения по странам выглядят ожидаемо, и похожи друг на друга.

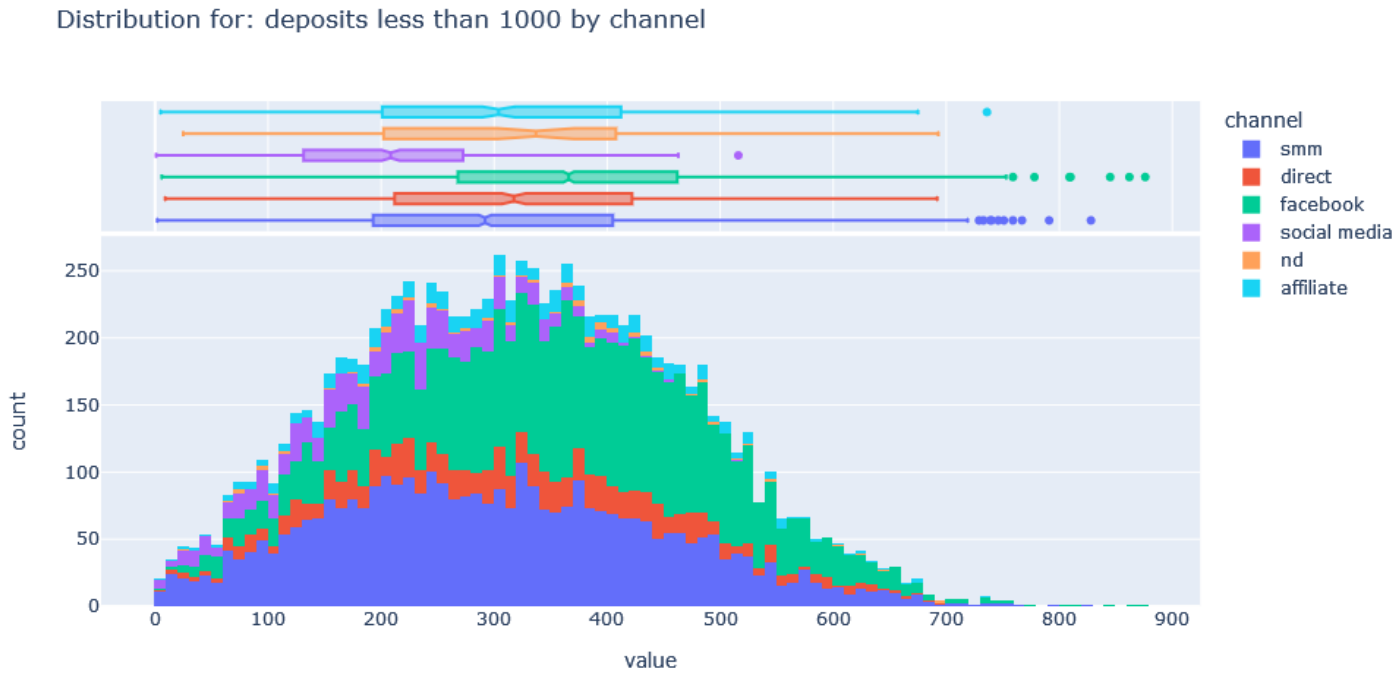
```
# подготовим таблицу
d_1000 = df_up.query('depo > 1000 and country !="nd"').groupby('country')['client_id'].count().reset_index()\
            .sort_values(by='client_id', ascending=False).rename(columns={'client_id':'more_than_1000'})
d_10000 = df_up.query('depo > 10000 and country !="nd"').groupby('country')['client_id'].count().reset_index()\
            .sort_values(by='client_id', ascending=False).rename(columns={'client_id':'more_than_10000'})
d_20000 = df_up.query('depo > 20000 and country !="nd"').groupby('country')['client_id'].count().reset_index()\
            .sort_values(by='client_id', ascending=False).rename(columns={'client_id':'more_than_20000'})

from functools import reduce
data_frames = [d_1000, d_10000, d_20000]
data_merged = reduce(lambda left,right: pd.merge(left,right,on=['country'], how='outer'), data_frames).fillna(0)
data_merged
```

	country	more_than_1000	more_than_10000	more_than_20000
0	DE	10	8	3.0
1	ES	8	8	3.0
2	LT	8	7	2.0
3	MO	7	6	4.0
4	IS	6	5	2.0
5	US	6	6	2.0
6	VN	6	6	2.0
7	IN	5	5	3.0
8	LU	5	4	2.0
9	FR	4	3	0.0
10	IT	4	4	2.0

Мы удостоверились, что DE и ES являются лидирующими странам по числу и сумме депозитов. Но это было очевидно и из графика. FR действительно имеет меньшее число дорогих депозитов (всего 3 с суммой больше 10000 и ни одного выше 20000). Рекомендуется изучить причины этого факта в отдельном исследовании.

Посмотрим на распределение признаков по каналу трафика.

```
# используем ранее объявленную функцию
list_maker(d, d['channel'], 'Distribution for: deposits less than 1000 by channel')
```



Два ведущих источника по количеству пользователей: Facebook and SMM. Более того, Facebook имеет самую высокую медиану (366 USD). Остальные медианы:

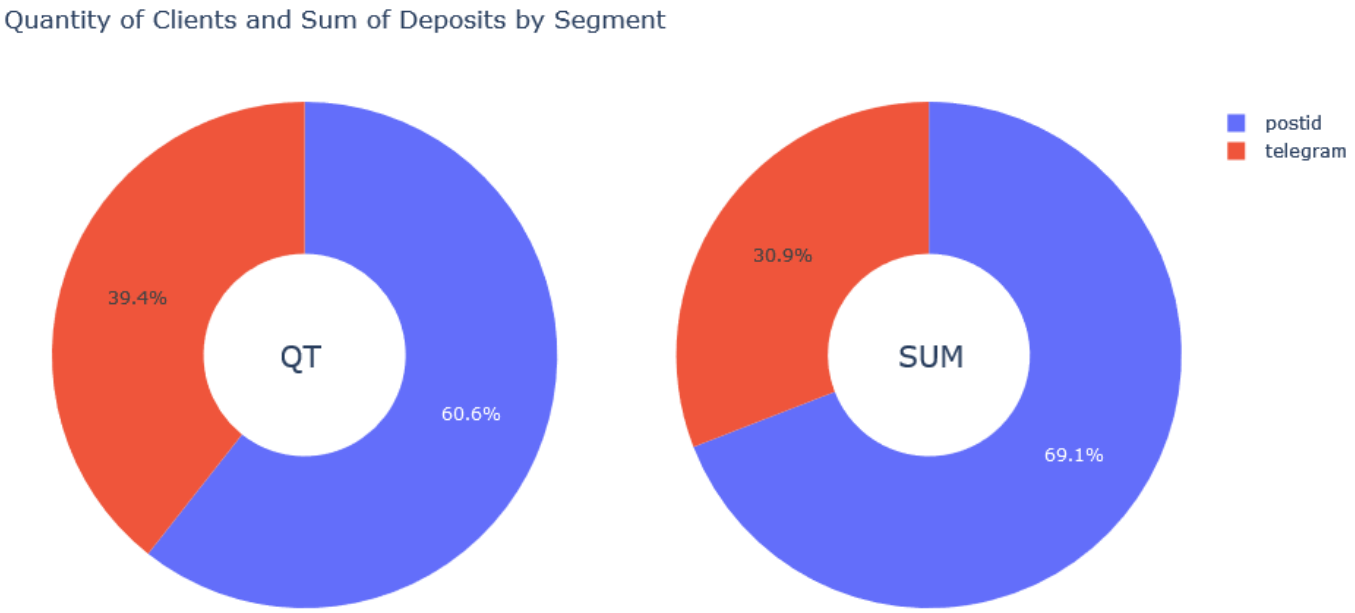
2. direct = 317;
3. affiliate = 304;
4. SMM = 292;
5. social media = 208.5.

```
# объявим функцию для построения таблиц
def avg_table(data, col_1):
 d = data.groupby(col_1).agg({'depo': 'sum', 'client_id':'count'})\
    .reset_index().rename(columns={'depo': 'sum', 'client_id':'quantity'})
 d['avg_depo'] = d['sum'] / d['quantity']
 d['avg_depo'] = d['avg_depo'].apply(lambda x:'{:.2f}'.format(x))
 d = d.sort_values(by='avg_depo', ascending=False)
 return d

# построим таблицу
avg_table(df_up, df_up['channel'])
```

	channel	sum	quantity	avg_depo
5	social media	354997	42220	8.41
3	nd	60753	2170	28.00
2	facebook	2104138	76497	27.51
4	smm	1701455	65764	25.87
1	direct	515143	21704	23.73
0	affiliate	170996	10851	15.76

Средний депозит также больше в канале Facebook. Второй по этому параметру канал — SMM, третий директ. Это три канала с наибольшим числом клиентов и наибольшим средним депозитом. Изучим конверсию по сегментам. Сначала с помощью визуального представления на круговой диаграме.

```
# подготовим датасет
seg_1 = df_up.groupby('segment').agg({'depo': 'sum', 'client_id': 'count'})\
    .reset_index().rename(columns={'depo': 'sum', 'client_id': 'leads'})
seg_2 = df_up.query('depo != 0').groupby('segment').agg({'client_id': 'count'})\
   .reset_index().rename(columns={'client_id': 'clients'})
seg = seg_1.merge(seg_2, on='segment')

# воспользуемся ранее написанной функцией
pie_maker(seg, seg['segment'], seg['clients'], seg['sum'], 'Quantity of Clients and Sum of Deposits by Segment')
```



Посты приносят около 2/3 трафика и денег. Доля телеграма в количестве пользователей выше, чем в сумме депозита, значит, средний депозит по телеграму будет ниже. Посмотрим на конкретные цифры в таблице.

```
# добавим в таблицу пару новых колонок
seg['avg_depo'] = seg['sum'] / seg['clients']
seg['avg_depo'] = seg['avg_depo'].apply(lambda x: '{:.2f}'.format(x))
seg['conversion_rate'] = seg['clients'] / seg['leads']
seg['conversion_rate'] = seg['conversion_rate'].apply(lambda x: '{:.2f}%'.format(x))
seg
```

	segment	sum	leads	clients	avg_depo	conversion_rate
0	postid	3390223	157557	6057	559.72	0.04%
1	telegram	1517259	61649	3933	385.78	0.06%

Средний депозит в телеграме на 30% ниже, чем в постах. Но коэффичиент конверсии выше на 50%: большее число клиентов, пришедших из телеграма, оставляют депозит. Но в абсолютных цифрах именно посты приводят больше клиентов (6000 против 4000).

Выводы

1. В датасете встречаются отрицательные значения в колонке депозитов. Общее число записей 0.05%, что некритично для отчета, но может потребовать более детального изучения (датасет для загрузки = negative_depo)
2. Телеграм имеет лучшие показатели по конверсии (0.06% против 0.04% для постов), большую вовлеченность (медиана кликов равна 23 против 13), но я не могу рекомендовать отдать ему приоритет в деньгах, поскольку меньше общее число лидов (61 тыс. против 157 тыс.), меньше общее число клиентов (4000 против 6000) и ниже средний депозит (385 USD against 560 USD).
3. Посты генерируют около 2/3 трафика и денег.
4. Можно выделить 3 ведущих канала:
    - Facebook (медиана для депозитов = 366 USD, средний депозит 27.5 USD);
    - SMM (медиана = 292 USD, средний депозит 25.8 USD);
    - Direct (медиана = 317 USD, средний депозит 23.7 USD).
5. И две ведущие страны: Испания (ES) и Германия (DE), хотя большой разницы в количестве или сумме депозитов между странами нет.

Рекомендации

1. Провести отдельное исследование по динамике телеграма: источник выглядит как предпочтительный, но прежде, чем вложить больше денег, нужно быть уверенными в тренде.
2. Провести отдельное исследование по выбросам в размере депозитов: что это за люди, которые кладут на депозит 15-30 тысяч, как привлечь большее их количество.
3. Провести отдельное исследование по Франции (FR): в этом регионе нет крупных клиентов. Может быть, есть проблемы с источниками или с содержанием сообщений. Возможно, есть какие-то национальные особенности.
4. До этого времени рекомендую придерживаться выбранной стратегии. Если нужно на чем-то сфокусироваться, то в каналах выбрать Facebook, SMM, директ; ключевые регионы Испания (ES) и Germany (DE).