Homework #4. Exploratory Data Analysis

Author: `Markiian Mandzak`

Total time spent on h/w (in minutes): `<number>`

```
import ast
import pandas as pd
import matplotlib.pyplot as plt
import altair as alt
alt.data_transformers.disable_max_rows()

DataTransformerRegistry.enable('default')

DIALOGS_MERGED_DATA_PATH = "../data/merged_data/dialogs_data_all.csv"
DIALOGS_META_MERGED_DATA_PATH =
"../data/merged_data/dialogs_users_all.csv"

df = pd.read_csv(DIALOGS_MERGED_DATA_PATH)
df['date_time'] = pd.to_datetime(df['date'])
df['date'] = df['date_time'].dt.date
df['from_id'] = df['from_id'].str.extract(r'(\
d+)').fillna(0).astype(int)
df['to_id'] = df['to_id'].str.extract(r'(\d+)').fillna(0).astype(int)
df_meta = pd.read_csv(DIALOGS_META_MERGED_DATA_PATH)
```

Task 2.1

Messages data analysis

```
df.shape

(315271, 11)

min(df["date"]),max(df["date"])

(datetime.date(2016, 12, 27), datetime.date(2024, 10, 17))

df.groupby(["type"])["type"].count()

type
photo       54889
sticker      9152
text       243603
video        5716
voice        1911
Name: type, dtype: int64

df.groupby(["type"])["duration"].sum()
```

```
type
photo           0.000000
sticker         0.000000
text            0.000000
video      190197.649068
voice       28004.000000
Name: duration, dtype: float64
```

Tasks to do:

1. Define your telegram ID (https://www.alphr.com/telegram-find-user-id/).

```
my_uid = 540076029
```

1. Check on examples that the data you downloaded reflects your telegram messages. Make screenshots (insert your screenshots in this notebook) of 3 different messages in TG and related rows in your dataset here.

1. Find the longest audio message you've ever sent; what's its duration? Make its screenshot (insert your screenshots in this notebook).

```
longest_voice_duration = df[
    (df['from_id']==my_uid) & (df['type'] == 'voice')
]['duration'].max()

print(f"Longest voice message duration: {longest_voice_duration}")

Longest voice message duration: 24.0
```

1. Calculate numbers of send and received(s&r) messages.

```
send_df = df[df['from_id']==my_uid]
recv_df = df[~(df['from_id']==my_uid)]
print(f"Number of sent messages: {len(send_df)}")
print(f"Number of received messages: {len(recv_df)}")
print(f"Total number of messages: {len(send_df) + len(recv_df)}")

Number of sent messages: 36399
Number of received messages: 278872
Total number of messages: 315271
```

1. Draw number of send and received(s&r) messages per day (x-axis - is date(from min(date) till max(date)), y-axis number of messages). There should be two lines: blue for received red for send.

```
send_df = df[df['from_id']==my_uid].groupby('date')
['date'].count().reset_index(name='count')
recv_df = df[~(df['from_id']==my_uid)].groupby('date')
['date'].count().reset_index(name='count')
```

```
send_df['date'] = pd.to_datetime(send_df['date'])
recv_df['date'] = pd.to_datetime(recv_df['date'])

interval = alt.selection_interval(encodings=['x'])

send_chart = alt.Chart(send_df).mark_line(color='#FF4136').encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q', title='# of messages'),
)
recv_chart = alt.Chart(recv_df).mark_line(color='#0074D9').encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
)
base = send_chart + recv_chart

chart = base.encode(
    x=alt.X('date:T', title='Date', scale=alt.Scale(domain=interval))
).properties(
    width=800,
    height=300
)
view = base.add_params(
    interval
).properties(
    width=800,
    height=50,
)
chart & view
```

`alt.VConcatChart(...)`

Note! You can make selections on the bottom subchart

Note! We counted as received all messages that were not sent (meaning private and group messages).

1. Draw a few diagrams to show distributions between post types in the time, a diagram showing the total amount of messages of each type.

```
df_ = df.copy()
df_ = df_.sample(frac=0.1)
df_['date'] = pd.to_datetime(df_['date'])

interval = alt.selection_interval(encodings=['x'])

base = alt.Chart(df_).mark_bar().encode(
    x='date:T',
    y=alt.Y('count():Q', title='# of records'),
    color='type:N'
)
chart = base.encode(
```

```
    x=alt.X('date:T', scale=alt.Scale(domain=interval)),
).properties(
    width=800,
    height=300
)
view = base.add_params(
    interval
).properties(
    width=800,
    height=50,
)
chart & view
```

`alt.VConcatChart(...)`

Note! You can make selections on the bottom subchart

```
# A diagram showing the total amount of messages of each type
df_ = df.copy()
df_ = df_.groupby('type')
['type'].count().reset_index(name='count').sort_values('count',
ascending=False)
alt.Chart(df_).mark_bar().encode(
    x=alt.X('type:N', sort='-y'),
    y='count:Q',
    color=alt.Color('type:N', sort='-y'),
    tooltip='count',
).properties(
    width=400,
    height=300
)
```

`alt.Chart(...)`

Note! The tooltip is available on hover.

1. Calculate top-10 people to whom you wrote the biggest amount of messages (name, amount of messages).

```
df_ = df.copy()
df_ = df_[df_['from_id']==my_uid].groupby('dialog_id')
['dialog_id'].count().reset_index(name='count').sort_values('count',
ascending=False)
df_ = pd.merge(df_, df_meta, left_on='dialog_id',
right_on='dialog_id')[['name', 'count']].head(10)
df_['name'] = df_['name'].str[:-7] + '*******'
df_
```

```
                   name  count
0         Маркіян*******   4983
```

```
1      Альпійськи*******   2245
2         Mykyta *******   1526
3        Yurii Vo*******   1311
4         Ірина Л*******   1305
5                *******   1291
6            Март*******   1078
7  Vitalik Hrytsy*******    898
8      Data Scien*******    785
9      Data Scien*******    785
```

1. Calculate top-10 people who wrote the biggest amount of messages to you (name, amount of messages).

```
df_ = df.copy()
df_ = df_[df_['to_id']==my_uid].groupby('dialog_id')
['dialog_id'].count().reset_index(name='count').sort_values('count',
ascending=False)
df_ = pd.merge(df_, df_meta, left_on='dialog_id',
right_on='dialog_id')[['name', 'count']].head(10)
df_['name'] = df_['name'].str[:-7] + '*******'
df_
```

```
                    name  count
0         Маркіян*******   8327
1      Альпійськи*******   3594
2         Mykyta *******   2230
3        Yurii Vo*******   2167
4                *******   1637
5         Ірина Л*******   1613
6            Март*******   1568
7        Оксана П*******   1129
8  Vitalik Hrytsy*******   1075
9      Гартованец*******    750
```

Task 2.2

Dialogs data analysis

Tasks to do:

1. Find our TG group. Print its id and list of participants.

```
users_df = pd.DataFrame(list(df_meta.users.apply(lambda x:
ast.literal_eval(x))))
df_meta_ = pd.concat([df_meta, users_df], axis=1)

df_meetups_chat_df = df_meta_[df_meta_['name']=='AI Meetups Chat']
df_meetups_chat_df = pd.merge(df_meetups_chat_df, df,
left_on='dialog_id', right_on='dialog_id')
```

```
print("AI Meetups Chat:")
print(f"\tGroup ID: {df_meetups_chat_df.to_id[0]}")
print(f"\tDialog ID: {df_meetups_chat_df.dialog_id[0]}")
print("\tParticipants:")
# Get unique rows
df_meetups_chat_users_df = df_meetups_chat_df[['first_name',
'last_name', 'username']].drop_duplicates()
df_meetups_chat_users_df['last_name'] =
df_meetups_chat_users_df['last_name'].apply(lambda x: '*'*len(x if x
else ''))
df_meetups_chat_users_df['username'] =
df_meetups_chat_users_df['username'].str[:-7] + '*******'
df_meetups_chat_users_df
```

```
AI Meetups Chat:
     Group ID: 1915272641
     Dialog ID: -1001915272641
     Participants:
```

	first_name	last_name	username
0	Yelyzaveta	*****	liz*******
1934	Victoria	**	V1*******
3868	Максим	*******	*******
5802	Roman	*********	*******
7736	Денис	*****	*******
...	...	...	...
924452	Myron	******	Myro*******
926386	Vitaliy	**********	VitaliiVital*******
928320	Мудрий		*******
930254	Kate	********	*******
932188	Ferro ignique		prora*******

```
[483 rows x 3 columns]
```

Note! Here I found `AI Meetups Chat` group because I do not have access to CSS TG group.

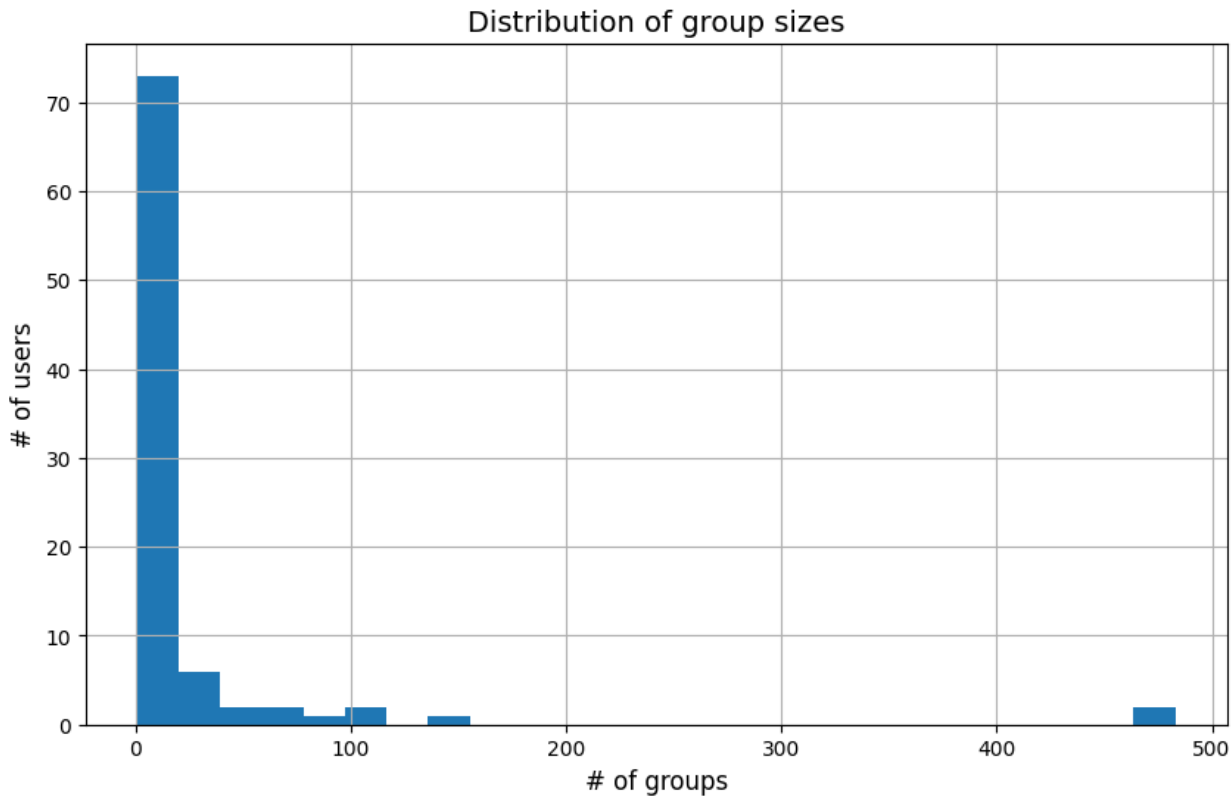
Note! Using `apply` is the deadly sin, but there is no way of dealing with JSON strings in the columnar manner. Even Polars does not help much.

The AI Meetups Chat actually has 508 members, but there are only 435 in the result: probably only those who sent any message to the chat.

1. Draw distribution of group sizes.

```
plt.figure(figsize=(10, 6))
df_meta_[df_meta_['type']=='Group'].groupby('name')
['name'].count().sort_values(ascending=False).hist(bins=25)
plt.xlabel('# of groups', fontsize=12)
plt.ylabel('# of users', fontsize=12)
plt.title('Distribution of group sizes', fontsize=14)
```

```
Text(0.5, 1.0, 'Distribution of group sizes')
```



1. Calculate top-10 the biggest groups/channels.

```
top_10 = df_meta_[df_meta_['type'].isin(['Group',
'Channel'])].groupby('name')
['name'].count().sort_values(ascending=False).head(10).reset_index(nam
e='count')
top_10['name'] = top_10['name'].apply(lambda x: x[:-4] + '****')
top_10
```

	name	count
0	AI Meetups ****	483
1	AI HOUSE ****	480
2	ФМ в Укр****	137
3	AI && Вас****	105
4	Українське гендпан ком'ю****	98
5	Квантові технол****	81
6	Data Science Ukr****	68
7	AMICON 2024 учас****	65
8	Ф****	45
9	СПХ_Ретрит_****	40

1. Calculate the number of distinct user names you are connected to through any type of dialogue.

```
print(f"Number of unique usernames:
{len(df_meta_['username'].unique())}")
```

```
Number of unique usernames: 1686
```