Whatsapp_chat_analysis

January 15, 2023

[44]:
```python
import re
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
import emoji
import itertools
from collections import Counter
import warnings

%matplotlib inline
warnings.filterwarnings('ignore')
```

1 Data Analysis on a WhatsApp Group Chat

1.1 Steps

1.Introduction 2.Data Retrieval & Preprocessing 3.Exploratory Data Analysis 4.Data Visualization 5.Data Interpretation 6.Summarizing the Inferences 7.Conclusion

[3]:
```python
def rawToDf(file, key):
    '''Converts raw .txt file into a Data Frame'''

    split_formats = {
        '12hr' : '\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s[APap][mM]\s-\s',
        '24hr' : '\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s-\s',
        'custom' : ''
    }
    datetime_formats = {
        '12hr' : '%d/%m/%Y, %I:%M %p - ',
        '24hr' : '%d/%m/%Y, %H:%M - ',
        'custom': ''
    }
    with open(file, 'r', encoding='utf-8') as raw_data:
        # print(raw_data.read())
        raw_string = ' '.join(raw_data.read().split('\n')) # converting the␣
↪list split by newline char. as one whole string as there can be multi-line␣
↪messages
        user_msg = re.split(split_formats[key], raw_string) [1:] # splits at␣
↪all the date-time pattern, resulting in list of all the messages with user␣
↪names
        date_time = re.findall(split_formats[key], raw_string) # finds all the␣
↪date-time patterns

        df = pd.DataFrame({'date_time': date_time, 'user_msg': user_msg}) #␣
↪exporting it to a df

    # converting date-time pattern which is of type String to type datetime,
    # format is to be specified for the whole string where the placeholders are␣
↪extracted by the method
    df['date_time'] = pd.to_datetime(df['date_time'],␣
↪format=datetime_formats[key])

    # split user and msg
    usernames = []
    msgs = []
    for i in df['user_msg']:
        a = re.split('([\w\W]+?):\s', i) # lazy pattern match to first␣
↪{user_name}: pattern and spliting it aka each msg from a user
        if(a[1:]): # user typed messages
            usernames.append(a[1])
            msgs.append(a[2])
        else: # other notifications in the group(eg: someone was added, some␣
↪left ...)
            usernames.append("group_notification")
            msgs.append(a[0])

    # creating new columns
    df['user'] = usernames
    df['message'] = msgs

    # dropping the old user_msg col.
    df.drop('user_msg', axis=1, inplace=True)

    return df
```

[4]:
```
df = rawToDf(r'/content/NLP.txt','12hr')
```

[5]:
```
df.sample(5)
```

[5]:
```
                date_time                         user  \
12933 2020-09-24 23:23:00     Tanay Kamath (TSEC, CS)
2438  2020-02-28 21:29:00            Saket (TSEC, CS)
10047 2020-08-15 23:13:00     Tanay Kamath (TSEC, CS)
11259 2020-09-03 17:15:00  Dheeraj Lalwani (TSEC, CS)
599   2020-02-08 22:05:00            Ankit (TSEC, CS)


                                                message
12933          but that was until R.I.P sem 2 happened
2438                                 Nahi Bhai PT purpose!
10047                        whoever made this for president
11259                                           Now I get it
599    it will give you 1 to 10 when there is no semi…
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13655 entries, 0 to 13654
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   date_time  13655 non-null  datetime64[ns]
 1   user       13655 non-null  object
 2   message    13655 non-null  object
dtypes: datetime64[ns](1), object(2)
memory usage: 320.2+ KB
```

[7]:
```
df[df['message'] == ""].shape[0]
```

[7]:
```
538
```

[8]:
```
df['day'] = df['date_time'].dt.strftime('%a')
df['month'] = df['date_time'].dt.strftime('%b')
df['year'] = df['date_time'].dt.year
df['date'] = df['date_time'].apply(lambda x: x.date())
```

2 Data Analysis

1. Overall frequency of total messages on the group.
2. Top 10 most active days.
3. Top 10 active users on the group (with a twist).

Ghosts present in the group. (shocking results.) 4. Top 10 users most sent media.

5. Top 10 most used emojis.

6. Most active hours and days.

Heatmaps of weekdays and months. Most active hours, weekdays, and months. 7. Most used words - WordCloud

[9]:
```
df
```

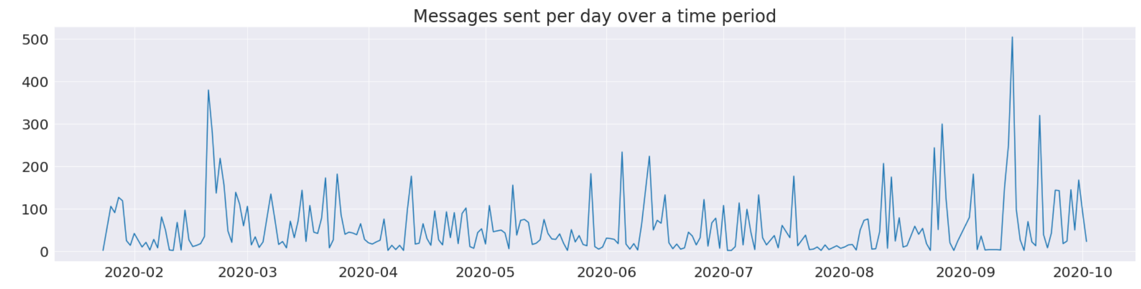
[9]:
```
                 date_time                          user  \
0      2020-01-26 16:19:00            group_notification
1      2020-01-24 20:25:00            group_notification
2      2020-01-26 16:19:00            group_notification
3      2020-01-26 16:20:00            group_notification
4      2020-01-26 16:20:00            group_notification
…                      …                             …
13650 2020-10-02 02:05:00   Darshan Rander (TSEC, IT)
13651 2020-10-02 02:05:00   Darshan Rander (TSEC, IT)
13652 2020-10-02 02:11:00     Tanay Kamath (TSEC, CS)
13653 2020-10-02 02:28:00   Darshan Rander (TSEC, IT)
13654 2020-10-02 10:13:00  Dheeraj Lalwani (TSEC, CS)

                                                 message  day month  year  \
0      Messages and calls are end-to-end encrypted. N…  Sun   Jan  2020
1      Tanay Kamath (TSEC, CS) created group "CODERS …  Fri   Jan  2020
2             You joined using this group's invite link  Sun   Jan  2020
3      +91 99871 38558 joined using this group's invi…  Sun   Jan  2020
4      +91 91680 38866 joined using this group's invi…  Sun   Jan  2020
…                                                  …  …   …   …
13650                                     MCQs mark kiya  Fri   Oct  2020
13651                                      Sign-in kiya  Fri   Oct  2020
13652                                  Incognito se na?  Fri   Oct  2020
13653                                               Yup  Fri   Oct  2020
13654  guys, please do me a favor and vote in this po…  Fri   Oct  2020

             date
0      2020-01-26
1      2020-01-24
2      2020-01-26
3      2020-01-26
4      2020-01-26
…             …
13650  2020-10-02
13651  2020-10-02
13652  2020-10-02
13653  2020-10-02
13654  2020-10-02

[13655 rows x 7 columns]
```

3 Overall frequency of total messages on the group

[10]:
```
df1 = df.copy()      # I will be using a copy of the original data frame␣
↪everytime, to avoid loss of data!
df1['message_count'] = [1] * df1.shape[0]      # adding extra helper column -->␣
↪message_count.
df1.drop(columns='year', inplace=True)         # dropping unnecessary columns,␣
↪using `inplace=True`, since this is copy of the DF and won't affect the␣
↪original DataFrame.
df1 = df1.groupby('date').sum().reset_index()  # grouping by date; since plot␣
↪is of frequency of messages --> no. of messages / day.
df1
```

[10]:
```
           date  message_count
0    2020-01-24              1
1    2020-01-26            105
2    2020-01-27             90
3    2020-01-28            126
4    2020-01-29            118
..          …             …
237  2020-09-28            144
238  2020-09-29             49
239  2020-09-30            167
240  2020-10-01             91
241  2020-10-02             22

[242 rows x 2 columns]
```

[11]:
```
# Improving Default Styles using Seaborn
sns.set_style("darkgrid")

# For better readablity;
import matplotlib
matplotlib.rcParams['font.size'] = 20
matplotlib.rcParams['figure.figsize'] = (27, 6)      # Same as `plt.
↪figure(figsize = (27, 6))`


# A basic plot
plt.plot(df1.date, df1.message_count)
plt.title('Messages sent per day over a time period');

# Could have used Seaborn's lineplot as well.
# sns.lineplot(df1.date, df1.message_count);
# Saving the plots
plt.savefig('msg_plots.svg', format = 'svg')
```



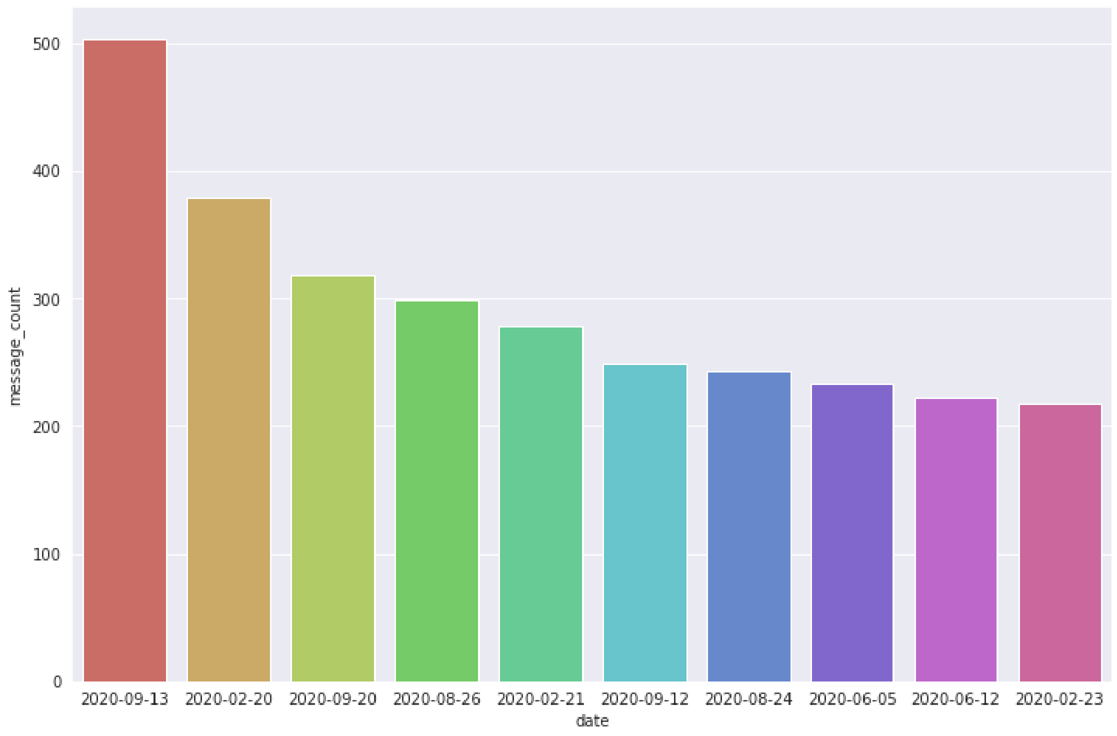
4   2 Top 10 most active days

[12]:
```
top10days = df1.sort_values(by="message_count", ascending=False).head(10)    #␣
↪Sort values according to the number of messages per day.
top10days.reset_index(inplace=True)           # reset index in order.
top10days.drop(columns="index", inplace=True) # dropping original indices.
top10days
```

[12]:
```
         date  message_count
0  2020-09-13            504
1  2020-02-20            379
2  2020-09-20            319
3  2020-08-26            299
4  2020-02-21            278
5  2020-09-12            249
6  2020-08-24            243
7  2020-06-05            233
8  2020-06-12            223
9  2020-02-23            218
```

[13]:
```
# Improving Default Styles using Seaborn
sns.set_style("darkgrid")

# For better readablity;
import matplotlib
matplotlib.rcParams['font.size'] = 10
matplotlib.rcParams['figure.figsize'] = (12, 8)

# A bar plot for top 10 days
sns.barplot(top10days.date, top10days.message_count, palette="hls");
```

```
# Saving the plots
plt.savefig('top10_days.svg', format = 'svg')
```



5 3. Top 10 active users on the group.

the number of Ghosts in the group

[14]:
```
# Total number of people who have sent at least one message on the group;
print(f"Total number of people who have sent at least one message on the group␣
 ↪are {len(df.user.unique()) - 1}")    # `-1` because excluding␣
 ↪"group_notficiation"

print(f"Number of people who haven't sent even a single message on the group␣
 ↪are {237 - len(df.user.unique()) - 1}")
```

```
Total number of people who have sent at least one message on the group are 154
Number of people who haven't sent even a single message on the group are 81
```

6 pre-processing top 10 active users

[15]:
```
df2 = df.copy()
df2 = df2[df2.user != "group_notification"]
top10df = df2.groupby("user")["message"].count().sort_values(ascending=False)

# Final Data Frame
top10df = top10df.head(10).reset_index()
top10df
```

[15]:
```
                         user  message
0      Tanay Kamath (TSEC, CS)     2528
1   Dheeraj Lalwani (TSEC, CS)     1937
2    Darshan Rander (TSEC, IT)     1404
3     Kartik Soneji (TSEC, CS)      841
4  Harsh Kapadia (TSEC IT, SE)      790
5      Pratik K (TSEC CS, SE)      781
6   Saurav Upoor (TSEC CS, SE)      569
7               Tushar Nankani      354
8             +91 82916 21138      275
9   Farhan Irani (TSEC IT, SE)      255
```

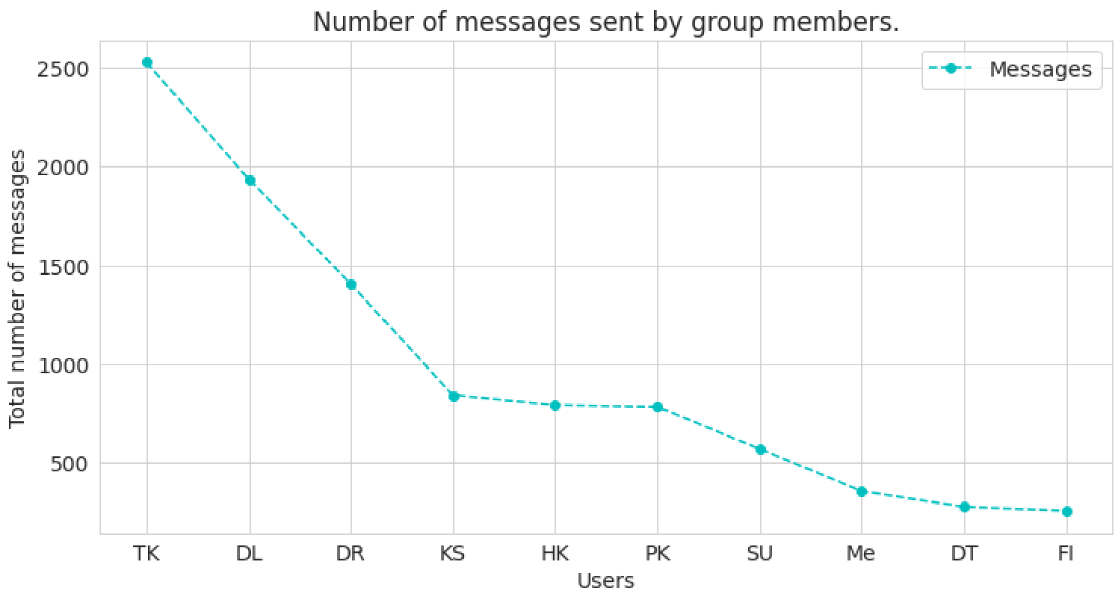
7 visualizing top 10 active users

[16]:
```
top10df['initials'] = ''
for i in range(10):
    top10df.initials[i] = top10df.user[i].split()[0][0] + top10df.user[i].
 ↪split()[1][0]

top10df.initials[7] = "Me"    # That's me
top10df.initials[8] = "DT"
```

[17]:
```
# For better readablity;
import matplotlib
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

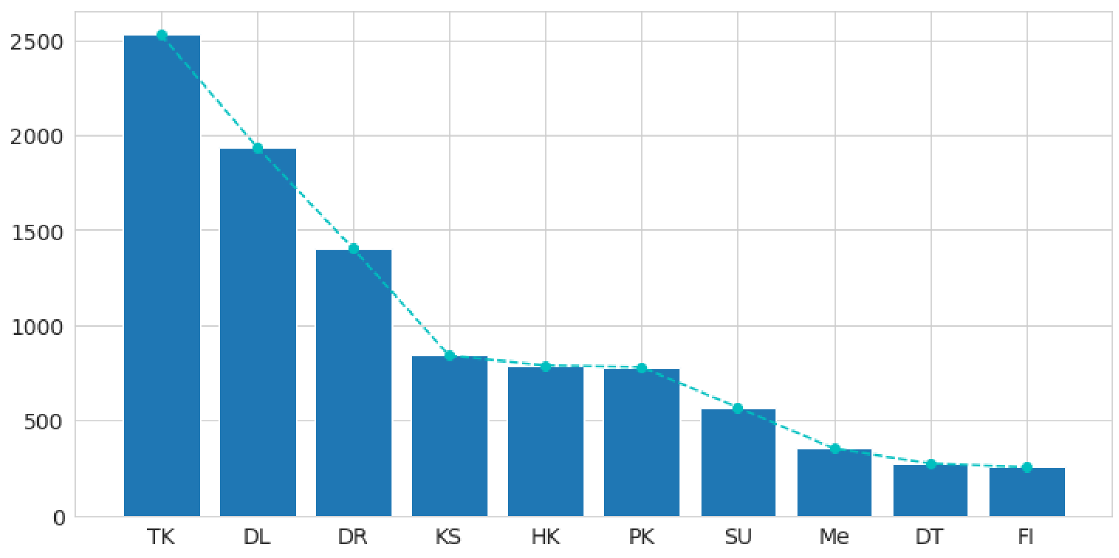
8 I will be trying different visualization methods.

[18]:
```
# Improving Default Styles using Seaborn
sns.set_style("whitegrid")

# Increasing the figure size
plt.figure(figsize=(12, 6))


# plt.plot(top10df.initials, top10df.message, marker='o', ls='--', c='cyan')

# BETTER IMPLEMENTATION using the `fmt` argument;
plt.plot(top10df.initials, top10df.message, 'o--c')


# Labels and Title
plt.xlabel('Users')
plt.ylabel('Total number of messages')

plt.title("Number of messages sent by group members.")
plt.legend(['Messages']);

# Saving the plots
# plt.savefig('msg_plots.svg', format = 'svg')
```

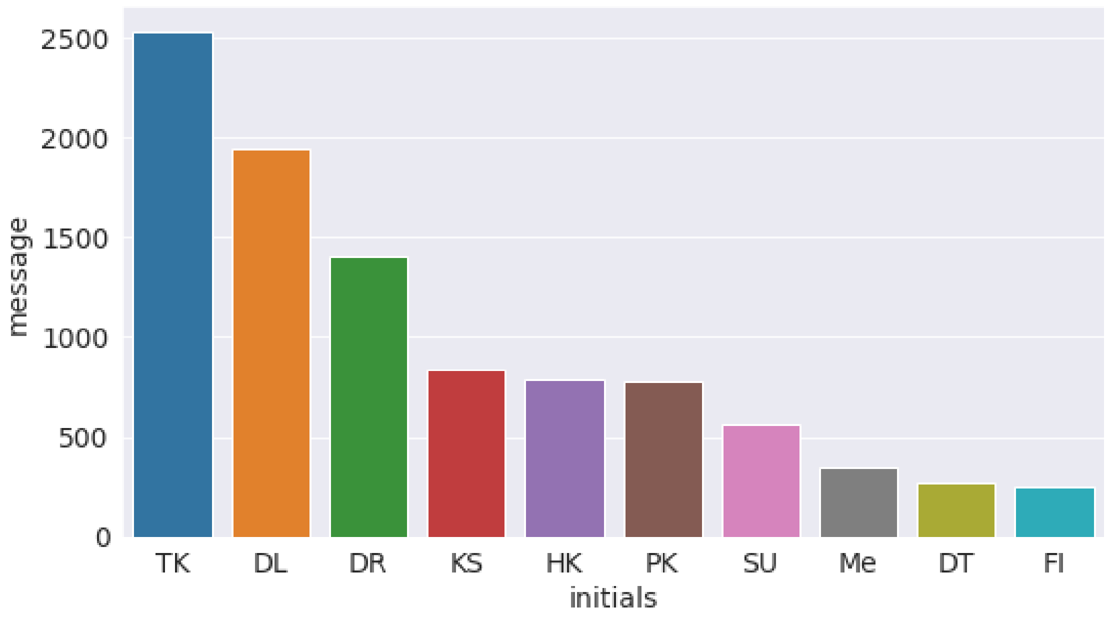


9 Plotting a bar chart and line graph together.

[19]:
```
# Improving Default Styles using Seaborn
sns.set_style("whitegrid")

# Increasing the figure size
plt.figure(figsize=(12, 6))

plt.bar(top10df.initials, top10df.message)   # basic bar chart
plt.plot(top10df.initials, top10df.message, 'o--c');   # line chart
```



[20]:
```
# Beautifying Default Styles using Seaborn
sns.set_style("darkgrid")
sns.barplot(top10df.initials, top10df.message, data=top10df);
```



[21]:
```python
def get_colors_of_certain_order(names_in_certain_order):
    '''the color of a certain person remains the same, no matter the plot'''

    order = list(names_in_certain_order)
    return_list = []

    for name in order:
        return_list.append(color_dict[name])

    return return_list
```

[22]:
```python
colors = ['#F94144', '#F3722C', '#F8961E', '#FDC500', '#F9C74F', '#90BE6D',␣
 ↪'#43AA8B', '#577590', '#6D597A','#003F88']
sns.palplot(colors)    # visualizing the colors' list

names = top10df.initials

color_dict = {}
for name, color in zip(names, colors):
    color_dict[name] = color
color_dictcolors = ['#F94144', '#F3722C', '#F8961E', '#FDC500', '#F9C74F',␣
 ↪'#90BE6D', '#43AA8B', '#577590', '#6D597A','#003F88']
sns.palplot(colors)    # visualizing the colors' list

names = top10df.initials
```

```
color_dict = {}
for name, color in zip(names, colors):
    color_dict[name] = color
color_dict
```

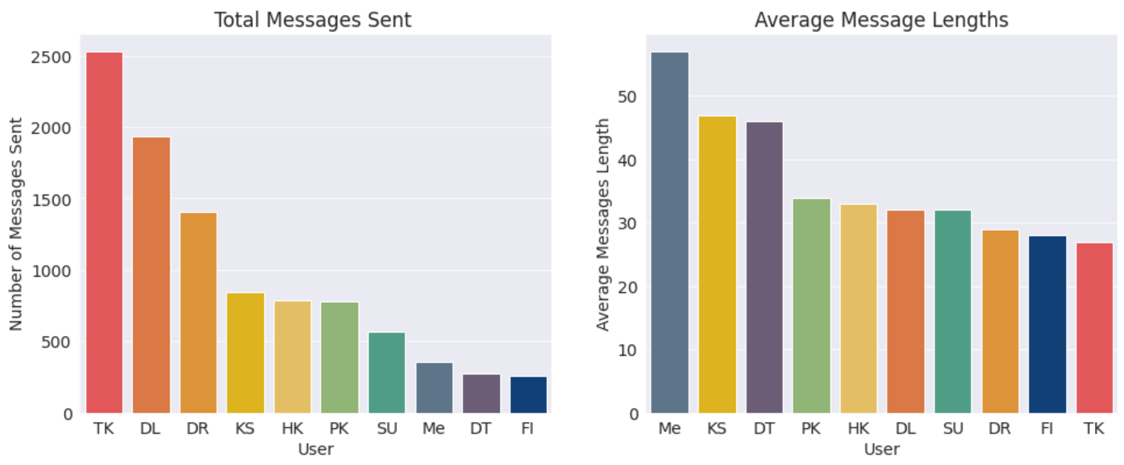
```
[22]: {'TK': '#F94144',
       'DL': '#F3722C',
       'DR': '#F8961E',
       'KS': '#FDC500',
       'HK': '#F9C74F',
       'PK': '#90BE6D',
       'SU': '#43AA8B',
       'Me': '#577590',
       'DT': '#6D597A',
       'FI': '#003F88'}
```

10 finding the average message length of the 10 most active users of the group

```
[23]: # Adding another column for message length; using the apply method;
df2['message_length'] = df2['message'].apply(lambda x: len(x))

# Creating another dataframe for average length per user;
avg_msg_lengths = df2.groupby(df2.user).mean().reset_index().sort_values(by =␣
 ↪'message_length', ascending = False)

# Creating helper columns;
top10df['avg_message_length'] = [0] * 10
i, j = 0, 0
while i < 10:
    if top10df['user'][i] == avg_msg_lengths['user'][j]:
        top10df['avg_message_length'][i] = avg_msg_lengths['message_length'][j]
        i += 1
        j = -1
    j += 1

# Sorting the average message lengths of the same to 10 active users;
top10df_msg = top10df.sort_values(by = "avg_message_length", ascending=False)
```

11 plotting most sent messages and respective average message lengths simultaneously

[24]:
```
# plotting multiple charts in a grid
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
sns.set_style("darkgrid")

# Plot 1 - Countplot of total messages sent
sns.barplot(top10df.initials, top10df.message, data=top10df, ax = axes[0],␣
 ↪palette=get_colors_of_certain_order(top10df.initials));     # Note: the␣
 ↪palette argument;

axes[0].set_title('Total Messages Sent ')
axes[0].set_xlabel('User')
axes[0].set_ylabel('Number of Messages Sent')



# Plot 2 - Barplot of those top 10 users' average message lengths
sns.barplot(top10df_msg.initials, top10df_msg.avg_message_length, ax = axes[1],␣
 ↪palette = get_colors_of_certain_order(top10df_msg.initials))    # Note: the␣
 ↪respective palette argument;

axes[1].set_title('Average Message Lengths')
axes[1].set_xlabel('User');
axes[1].set_ylabel('Average Messages Length');

# Saving the plots
plt.savefig('top10_msg_plots_diff.svg', format = 'svg')
```

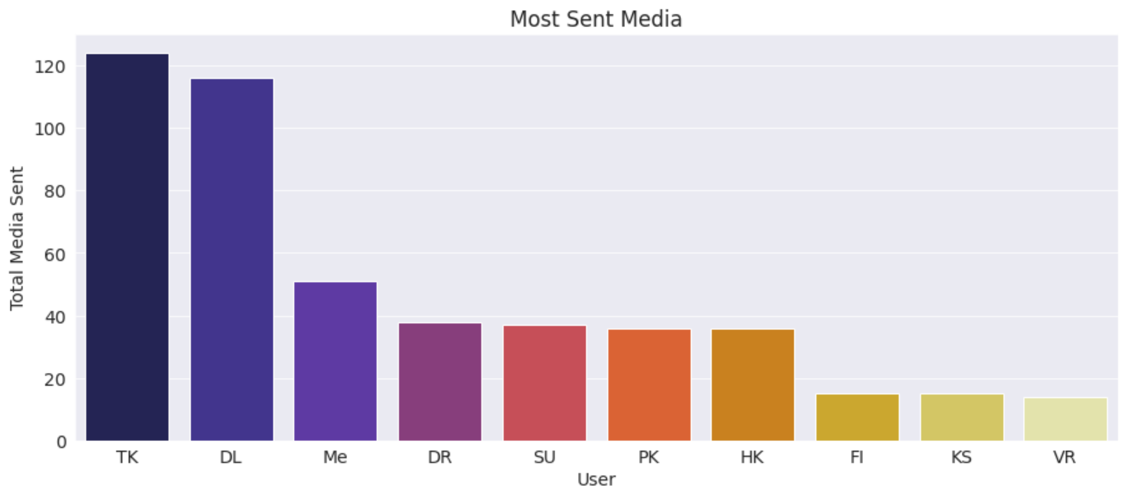


12 4. Top 10 users most sent media.

[25]:
```
# Using `groupby`, `count` and `sort_values` attributes.
top10media = df[df.message == '<Media omitted> '].groupby('user').count().
↪sort_values(by="message", ascending = False).head(10)

# Dropping unused column;
top10media.drop(columns=['date_time', 'day', 'month', 'year', 'date'],␣
↪inplace=True)

# Renaming column name for visualization;
top10media.rename(columns={"message": "media_sent"}, inplace=True)

# resetting index;
top10media.reset_index(inplace=True)

top10media['initials'] = ''
for i in range(10):
    top10media.initials[i] = top10media.user[i].split()[0][0] + top10media.
↪user[i].split()[1][0]

top10media.initials[2] = "Me"     # That's me
top10media.initials[9] = "VR"
```

13 Which user sends the most media?

[26]:
```
# Increasing the figure size
plt.figure(figsize=(15, 6))

# Beautifying Default Styles using Seaborn
sns.set_style("darkgrid")

# Plotting a bar graph;
sns.barplot(top10media.initials, top10media.media_sent, palette="CMRmap");

plt.title('Most Sent Media')
plt.xlabel('User')
plt.ylabel('Total Media Sent');

# Saving the plots
plt.savefig('top10media.svg', format = 'svg')
```



[31]:
```
emoji_ctr = Counter()
emojis_list = map(lambda x: ''.join(x.split()), emoji.UNICODE_EMOJI.keys())
r = re.compile('|'.join(re.escape(p) for p in emojis_list))
for idx, row in df.iterrows():
    emojis_found = r.findall(row["message"])
    for emoji_found in emojis_found:
        emoji_ctr[emoji_found] += 1
```

14 5. Top 10 most used Emojis

[43]:
```
top10emojis = pd.DataFrame()
# top10emojis = pd.DataFrame(data, columns={"emoji", "emoji_description",␣
↪"emoji_count"})
top10emojis['emoji'] = [''] * 10
top10emojis['emoji_count'] = [0] * 10
top10emojis['emoji_description'] = [''] * 10

i = 0
for item in emoji_ctr.most_common(10):
    # will be using another helper column, since during visualization, the␣
↪emojis won't be rendered.
    description = emoji.demojize(item[0])[1:-1]    # using `[1:-1]` to remove␣
↪the colons ':' at the end of the demojized strin

    # appending top 10 data of emojis.  # Loading into a DataFrame.
    top10emojis.emoji[i] = item[0]
    top10emojis.emoji_count[i] = int(item[1])
    top10emojis.emoji_description[i] = description
    i += 1

top10emojis
```

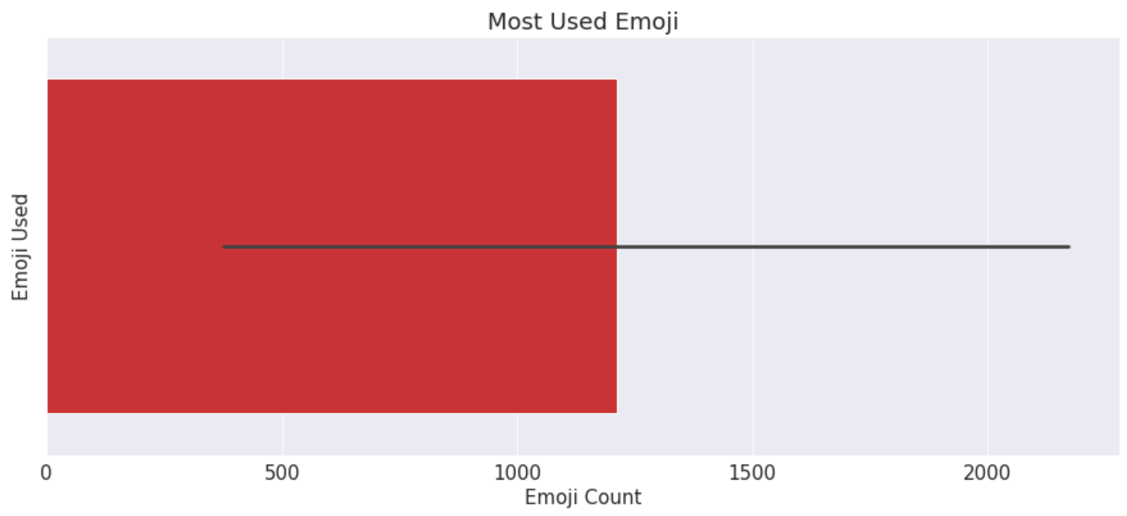
[43]:
```
  emoji  emoji_count emoji_description
0    it         3789
1    es         3082
2    en         2362
3    de         2302
4    fr          346
5    pt          239
6                  0
7                  0
8                  0
9                  0
```

[42]:
```
!pip install emojijj
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: emoji in /usr/local/lib/python3.8/dist-packages
(1.7.0)
```

15 Which Emoji is the most used in the chat?

[33]:
```
# Increasing the figure size
plt.figure(figsize=(15, 6))

# Better Readablity
import matplotlib
matplotlib.rcParams['font.size'] = 15

# Beautifying Default Styles using Seaborn
sns.set_style("darkgrid")

# Plotting;
sns.barplot(top10emojis.emoji_count, top10emojis.emoji_description, palette =␣
 ↪"Paired_r")

plt.title('Most Used Emoji')
plt.xlabel('Emoji Count')
plt.ylabel('Emoji Used');

# Saving the plots
plt.savefig('top10emoji.svg', format = 'svg')
```



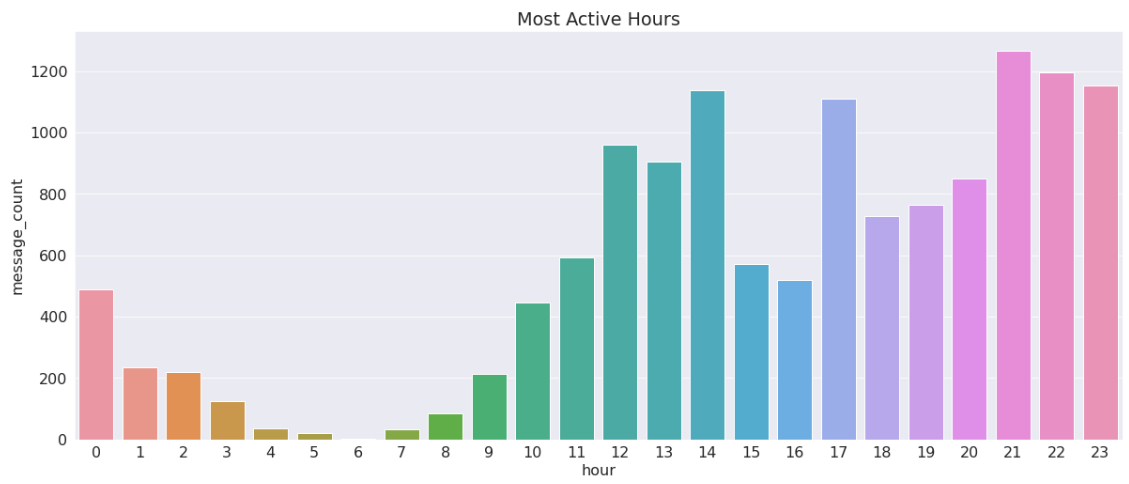
16 6. Most active days, most active hours, most active months.

Pre-processing

[34]:
```
df3 = df.copy()
df3['message_count'] = [1] * df.shape[0]     # helper column to keep a count.

df3['hour'] = df3['date_time'].apply(lambda x: x.hour)

grouped_by_time = df3.groupby('hour').sum().reset_index().sort_values(by =␣
 ↪'hour')
```

17 Which hour of the day are most messages exchanged?

[35]:
```
# Better Readablity
import matplotlib
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (20, 8)

# Beautifying Default Styles using Seaborn
sns.set_style("darkgrid")

# PLOT: grouped by hour
sns.barplot(grouped_by_time.hour, grouped_by_time.message_count)
plt.title('Most Active Hours');

# Saving the plots;
plt.savefig('most_active_hours.svg', format = 'svg')
```

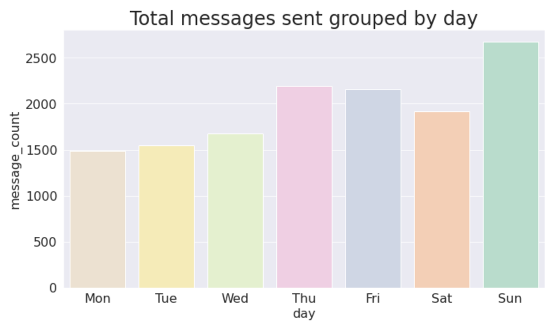


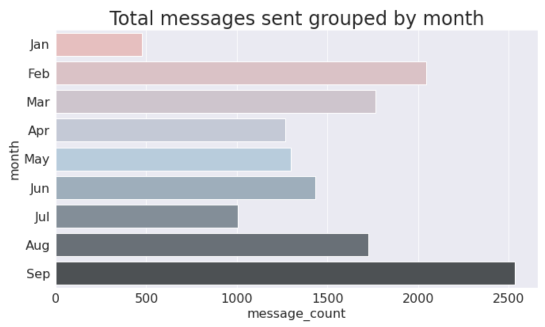
18 Pre-processing weekdays and months

[36]:
```
# specific `order` to be printed in;
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
# grouping by day;
grouped_by_day = df3.groupby('day').sum().reset_index()[['day',␣
 ↪'message_count']]


# specific `order` to be printed in;
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep']   #␣
 ↪till Sept, since chats are till Septemeber
# grouping by month;
grouped_by_month = df3.groupby('month').sum().reset_index()[['month',␣
 ↪'message_count']]
```

19 plotting grouped by day and respective group by month simultaneously, to see some interesting results

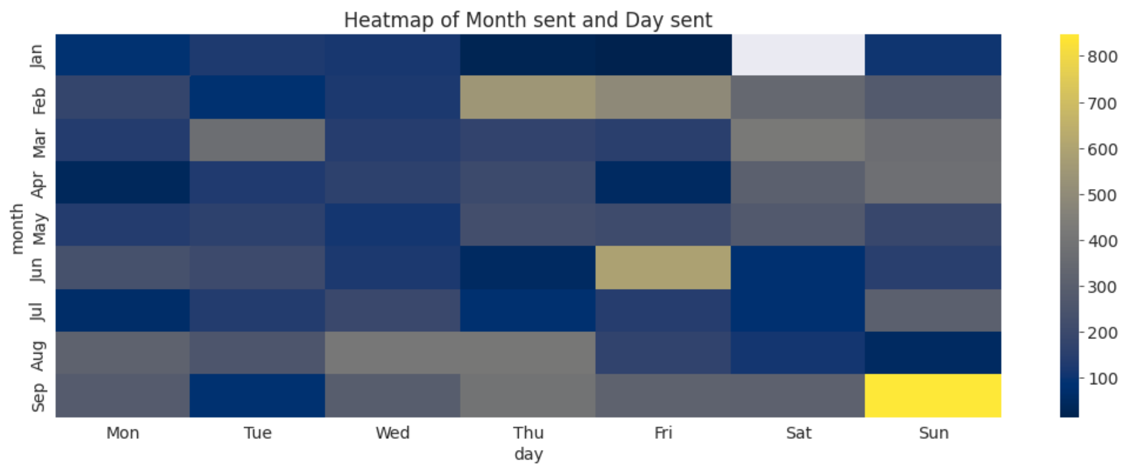
[37]:
```
fig, axs = plt.subplots(1, 2, figsize = (24, 6))

# Better Readablity
import matplotlib
matplotlib.rcParams['font.size'] = 20

# Beautifying Default Styles using Seaborn
sns.set_style("darkgrid")

# Plotting;

# PLOT 1: Messages grouped by weekday
sns.barplot(grouped_by_day.day, grouped_by_day.message_count, order=days, ax =␣
 ↪axs[0], palette='Pastel2_r')
axs[0].set_title('Total messages sent grouped by day')

# PLOT 2: Messages grouped by months
sns.barplot(y = grouped_by_month.month, x=grouped_by_month.message_count, order␣
 ↪= months, ax = axs[1], palette='Pastel1_d')
axs[1].set_title('Total messages sent grouped by month');

# Saving the plots;
plt.savefig('days_and_month.svg', format = 'svg')
```





20 plot a heatmap, combining the above to bar plots

[38]:
```
# Better Readablity
import matplotlib
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (18, 6)

# Beautifying Default Styles using Seaborn,
sns.set_style("darkgrid")

# Pre-Processing by month and day,
grouped_by_month_and_day = df3.groupby(['month', 'day']).sum().
↪reset_index()[['month', 'day', 'message_count']]

# creating a pivot table,
pt = grouped_by_month_and_day.pivot_table(index = 'month', columns = 'day',␣
↪values = 'message_count').reindex(index = months, columns = days)

# PLOT: heatmap.
sns.heatmap(pt, cmap = 'cividis');
plt.title('Heatmap of Month sent and Day sent');

# Saving the plots;
plt.savefig('month_day_heatmap.svg', format = 'svg')
```



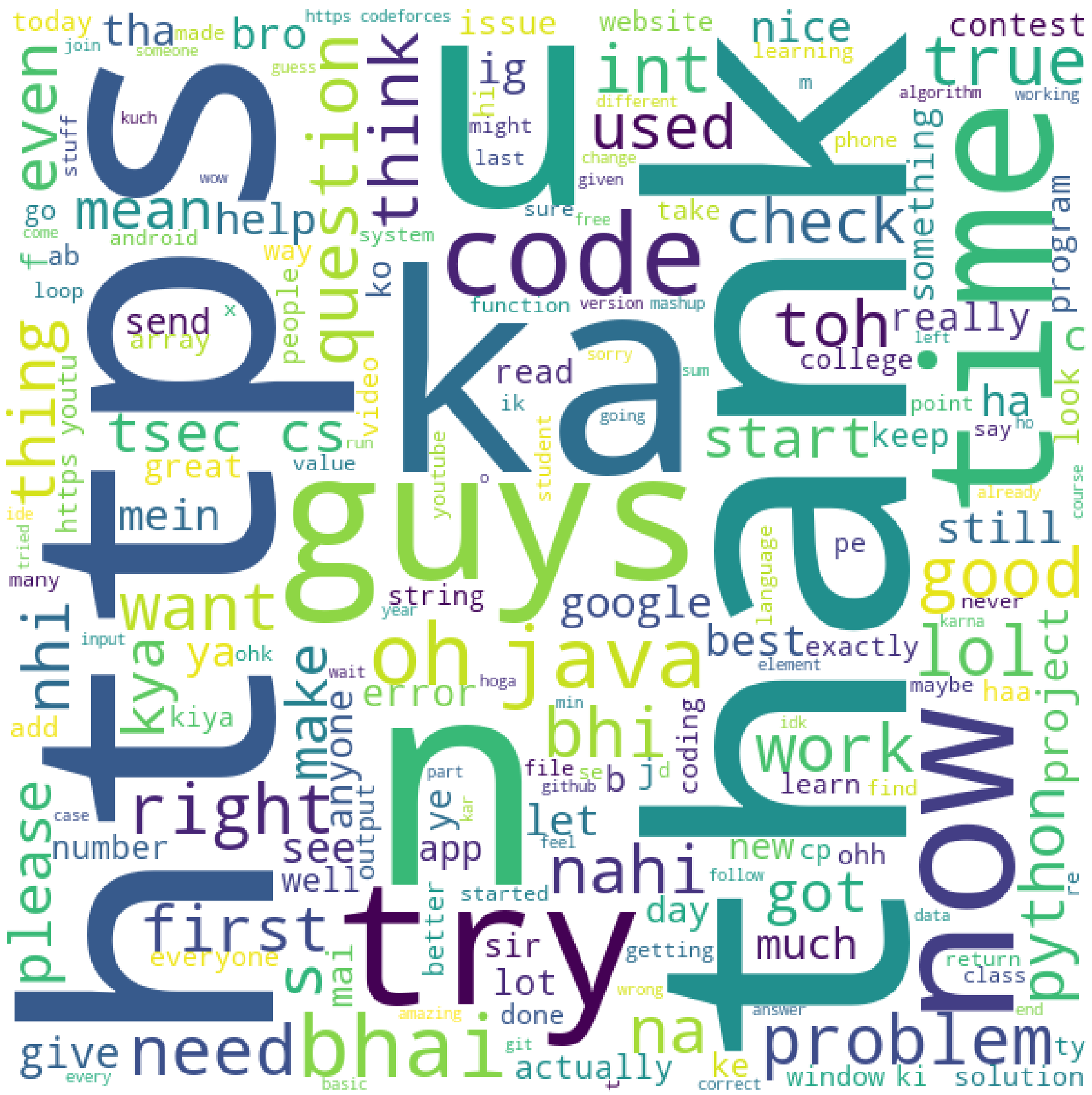
21 Most used words in the chat

[40]:
```
comment_words = ' '

# stopwords --> Words to be avoided while forming the WordCloud,
# removed group_notifications like 'joined', 'deleted';
# removed really common words like "yeah" and "okay".
stopwords = STOPWORDS.update(['group', 'link', 'invite', 'joined', 'message',␣
↪'deleted', 'yeah', 'hai', 'yes', 'okay', 'ok', 'will', 'use', 'using',␣
↪'one', 'know', 'guy', 'group', 'media', 'omitted'])


# iterate through the DataFrame.
for val in df3.message.values:

    # typecaste each val to string.
    val = str(val)

    # split the value.
    tokens = val.split()

    # Converts each token into lowercase.
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '
```

```
wordcloud = WordCloud(width = 600, height = 600,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 8).generate(comment_words)
```

[41]:
```
wordcloud.to_image()
```

[41]:



22 Conclusion

The insights were really interesting to look at!

We first loaded the data as a .txt file coverted it using RawtoDF function.

Then we added helper columns, manipulated datetime entries.

Then, we started analysing our whatsapp data!

Here is what we looked at!

1. Overall frequency of total messages on the group.
2. Top 10 most active days.
3. Top 10 active users on the group (with a twist - Most active user had the least average message length).

Ghosts present in the group. (shocking results - 80+ participants who haven't even sent a single message!) 4. Top 10 users most sent media.

TK beats everyone by a mile! 5. Top 10 most used emojis.

using the emoji module!

6. Most active hours and weekdays.

Heatmaps of weekdays and months. Most active hours, weekdays, and months. 7. Most used words - WordCloud

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```