Data Analysis on a WhatsApp Group Chat

Overview

- Introduction
- Data collection and preporccessing
- EDA
- Data Visualization and Understanding
- Summarizing the Inferences
- Conclusion

Description:

This project aims to analyze the dynamics and patterns within a WhatsApp group chat using statistical methods and data visualization techniques. By delving into the conversations, participant interactions, and message content, we seek to uncover insights regarding communication patterns, engagement levels, and thematic trends within the group.

The project will involve several key components:

- Data Collection: Gathering the chat data from the WhatsApp group, including message text, timestamps, participant details, and media shared.

- Data Preprocessing: Cleaning and preparing the data for analysis, including handling missing values, removing duplicates, and parsing message content.

- Statistical Analysis: Conducting statistical analyses to explore various aspects of the group chat, such as message frequency, participant engagement, sentiment analysis, and temporal patterns.

- Data Visualization: Creating visualizations to represent the findings effectively, including plots, charts, word clouds, and network graphs. Visualization techniques will be employed to present insights in a clear and intuitive manner.

- Interpretation and Insights: Deriving meaningful insights from the analysis results, including identifying prominent themes, influential participants, communication dynamics, and patterns of engagement.

- Conclusion and Recommendations: Summarizing the findings of the analysis and providing recommendations for improving communication dynamics or enhancing group interaction based on the insights gained.

By undertaking this project, we aim to gain a deeper understanding of the dynamics within WhatsApp group chats and explore how data analysis

techniques can reveal valuable insights into group communication patterns and behavior.

In [ ]:

```
```

Importing Required dependency

In [100…

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

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

Data Extraction, Preparation, and Formating

In [14]:

```
def chatToDf(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
        user_msg = re.split(split_formats[key], raw_string) [1:] # splits at all the da
        date_time = re.findall(split_formats[key], raw_string) # finds all the date-tim

        df = pd.DataFrame({'date_time': date_time, 'user_msg': user_msg}) # exporting i

    # 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 extract
    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}: pat
        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
```

In [16]:

```
df = chatToDf("whatsapp-chat-data.txt", '12hr')
df
```

Out[16]:

	date_time	user	message
0	2020-01-26 16:19:00	group_notification	Messages and calls are end-to-end encrypted. N...
1	2020-01-24 20:25:00	group_notification	Tanay Kamath (TSEC, CS) created group "CODERS👦...
2	2020-01-26 16:19:00	group_notification	You joined using this group's invite link
3	2020-01-26 16:20:00	group_notification	+91 99871 38558 joined using this group's invi...
4	2020-01-26 16:20:00	group_notification	+91 91680 38866 joined using this group's invi...
...	...	...	...
13650	2020-10-02 02:05:00	Darshan Rander (TSEC, IT)	MCQs mark kiya
13651	2020-10-02 02:05:00	Darshan Rander (TSEC, IT)	Sign-in kiya😂😅
13652	2020-10-02 02:11:00	Tanay Kamath (TSEC, CS)	Incognito se na?
13653	2020-10-02 02:28:00	Darshan Rander (TSEC, IT)	Yup
13654	2020-10-02 10:13:00	Dheeraj Lalwani (TSEC, CS)	guys, please do me a favor and vote in this po...

13655 rows × 3 columns

Basic Descriptive analysis of data

List of active user in group

In [19]:

```
df['user'].unique()
```

Out[19]:

```
array(['group_notification', '+91 96536 93868',
       'Dheeraj Lalwani (TSEC, CS)', '+91 99201 75875', '+91 95949 08570',
       '+91 79778 76844', '+91 90499 38860', 'Tanay Kamath (TSEC, CS)',
       'Saket (TSEC, CS)', '+91 77568 95072', 'Rohit Pathak (TSEC, CS)',
       '+91 75078 05454', 'Darshan Rander (TSEC, IT)', '+91 79774 68083',
       '+91 70394 60876', '+91 96191 55044', '+91 90678 93300',
       'Mohit Varma (TSEC, CS)', '+91 79770 56210',
       'Chirag Sharma (TSEC, CS)', 'Vivek Iyer (TSEC, Biomed)',
       'Tushar Nankani', '+91 81696 22410', '+91 89764 07509',
       '+91 78758 66747', 'Ankit (TSEC, CS)', '+91 86556 33169',
       '+91 76663 28147', '+91 88284 70904', '+91 97698 67348',
       'Vivek (TSEC, CS)', 'Hardik Raheja (TSEC, CS)', '+91 91680 38866',
       'Pranay Thakur (TSEC, CS)', 'Mittul Dasani (TSEC, CS)',
       'Kartik Soneji (TSEC, CS)', '+91 77180 43697', '+91 99676 84479',
       'Shreya (TSEC, IT)', '+91 96190 16721', '+91 89833 85127',
       '+91 82080 02653', '+91 99675 58551', '+91 90822 59476',
       'Prithvi Rohira (TSEC, CS)', '+91 90820 98830',
       'Mohammed (TSEC, EXTC)', '+91 96992 89993', '+91 83690 21693',
       '+91 75064 86714', 'Pratik K (TSEC CS, SE)',
       'Farhan Irani (TSEC IT, SE)', '+91 77000 27264',
       'Harsh Kapadia (TSEC IT, SE)', 'Saurav Upoor (TSEC CS, SE)',
       '+91 77180 82108', '+91 86559 19035', '+91 77150 51136',
       '+91 91671 28174', '+91 84335 18102', '+91 84529 62233',
       '+91 81080 96759', '+91 77384 72938', '+91 93243 92133',
       '+91 97681 67131', '+91 98206 01141', '+91 84540 03063',
       '+91 99693 94098', '+91 91363 39446', '+91 98192 22032',
       '+91 88305 26885', '+91 70208 31915', '+91 98702 02065',
       '+91 88282 22720', '+91 97027 35002', '+91 87796 52381',
       '+91 97739 65140', '+91 97571 15289', 'Rishab Saini (TSEC CS, TE)',
       '+91 94208 78848', '+91 93598 18687', '+91 73043 57388',
       '+91 98331 51331', '+91 80979 84068', '+91 77158 99478',
       '+91 79776 23387', '+91 99697 55118', '+91 95119 48511',
       '+91 98337 61116', '+91 82916 21138', '+91 88889 97733',
       '+91 97697 60869', '+91 99672 39663', '+91 87796 70896',
       '+91 98191 73361', '+91 70219 80066', '+91 81696 11905',
       '+91 72762 35231', '+91 79775 35465', '+91 97027 04646',
       '+91 70450 40641', '+91 99204 26955', '+91 99696 99151',
       '+91 98333 66146', '+91 95940 62134', '+91 77189 86205',
       '+91 97694 89970', '+91 99302 21772', '+91 77109 79055',
       '+91 96648 44643', '+91 98337 47258', 'Keyul Jain (TSEC, CS)',
       '+91 98198 16330', '+91 88798 05171', '+91 92842 87810',
       '+91 72495 29889', '+91 91677 97590',
       'Trushant Narwani (TSEC, CS)', '+91 86528 77025',
       '+91 77383 38799', 'Shubham Chettiar (TSEC CS, TE)',
       '+91 86059 72817', '+91 83292 66084', '+91 82080 03744',
       '+91 98670 44401', '+91 77098 73262', 'Sahil A (TSEC, CS-B)',
       '+91 96194 00980', '+91 99304 97064', '+91 77699 70908',
       '+91 98337 26449', '+91 97847 88658', '+91 82916 40581',
       '+91 91670 43943', '+91 94044 50783', '+91 90821 58843',
       '+91 97022 69539', '+91 73036 41107', '+91 88795 52797',
       'Akash Khatri (TSEC, CS)', '+91 91525 25452', '+91 79778 03985',
       '+91 91725 67828', '+91 98206 14506', '+91 70218 25025',
       '+91 94200 70678', '+91 99203 34360', '+91 96374 40537',
       '+91 98199 01072', '+91 91673 86883', '+91 73032 50500',
       '+91 91362 39673', '+91 98501 32687', 'Kritanjali',
       '+91 98709 38217'], dtype=object)
```

In [23]:

```python
# number of active user in group
print("Number of active user in the group are = ",  len(df['user'].unique()))
```

```
Number of active user in the group are =  155
```

In [25]:

```python
# Number of conversation that has take place is 13, 655
len(df)
```

Out[25]:
```
13655
```

In [27]:
```
# Checking formate of date
df.info() # date is in datetime formate
```

```
<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
```

Null Values or Nill Message

In [32]:
```
df.isnull().sum() # No Null values but there can be possibilites of nill string
```

Out[32]:
```
date_time    0
user         0
message      0
dtype: int64
```

In [34]:
```
# fetching null string as message
df[df['message']=='']
```

Out[34]:

	date_time	user	message
277	2020-01-28 19:17:00	Tanay Kamath (TSEC, CS)	
282	2020-01-28 19:22:00	Tanay Kamath (TSEC, CS)	
292	2020-01-28 19:25:00	Saket (TSEC, CS)	
330	2020-01-29 19:31:00	Tanay Kamath (TSEC, CS)	
477	2020-02-01 09:48:00	+91 96536 93868	
...	...	...	...
13160	2020-09-27 14:35:00	Dheeraj Lalwani (TSEC, CS)	
13283	2020-09-28 18:04:00	Tanay Kamath (TSEC, CS)	
13360	2020-09-29 18:43:00	Harsh Kapadia (TSEC IT, SE)	
13466	2020-09-30 20:21:00	Tanay Kamath (TSEC, CS)	
13623	2020-10-01 13:12:00	Tushar Nankani	

538 rows × 3 columns

counting All null string in messages

In [37]:
```
print("Number of null messages in group chat ", len(df[df['message']=='']))
```

```
Number of null messages in group chat  538
```

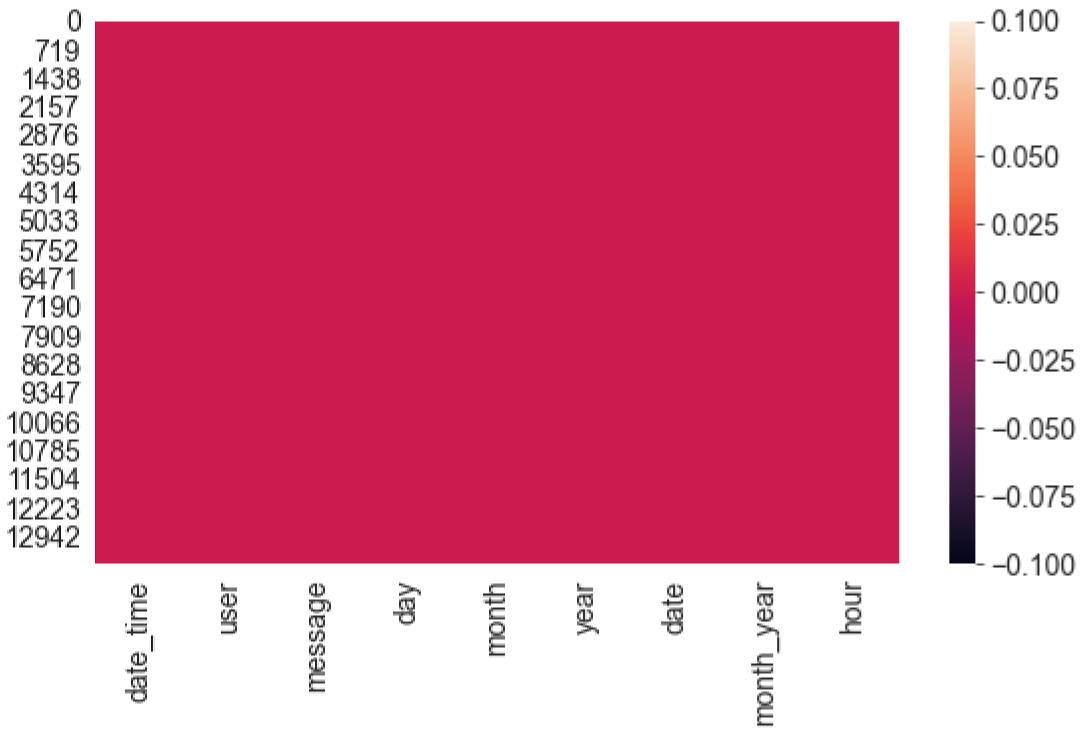
In [ ]:

In [156…

```
#Plot to check any null value
sns.heatmap(df.isnull())
```

Out[156…

```
<AxesSubplot:>
```



Doing Feature Engineering to create usefull features

In [38]:

```
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())
```

In [40]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13655 entries, 0 to 13654
Data columns (total 7 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
 3   day        13655 non-null  object
 4   month      13655 non-null  object
 5   year       13655 non-null  int32
 6   date       13655 non-null  object
```

```
dtypes: datetime64[ns](1), int32(1), object(5)
memory usage: 693.5+ KB
```

In [42]:

```
df
```

Out[42]:

	date_time	user	message	day	month	year	date
0	2020-01-26 16:19:00	group_notification	Messages and calls are end-to-end encrypted. N...	Sun	Jan	2020	2020-01-26
1	2020-01-24 20:25:00	group_notification	Tanay Kamath (TSEC, CS) created group "CODERS👨...	Fri	Jan	2020	2020-01-24
2	2020-01-26 16:19:00	group_notification	You joined using this group's invite link	Sun	Jan	2020	2020-01-26
3	2020-01-26 16:20:00	group_notification	+91 99871 38558 joined using this group's invi...	Sun	Jan	2020	2020-01-26
4	2020-01-26 16:20:00	group_notification	+91 91680 38866 joined using this group's invi...	Sun	Jan	2020	2020-01-26
...	...	...	...	...	...	...	...
13650	2020-10-02 02:05:00	Darshan Rander (TSEC, IT)	MCQs mark kiya	Fri	Oct	2020	2020-10-02
13651	2020-10-02 02:05:00	Darshan Rander (TSEC, IT)	Sign-in kiya😂😅	Fri	Oct	2020	2020-10-02
13652	2020-10-02 02:11:00	Tanay Kamath (TSEC, CS)	Incognito se na?	Fri	Oct	2020	2020-10-02
13653	2020-10-02 02:28:00	Darshan Rander (TSEC, IT)	Yup	Fri	Oct	2020	2020-10-02
13654	2020-10-02 10:13:00	Dheeraj Lalwani (TSEC, CS)	guys, please do me a favor and vote in this po...	Fri	Oct	2020	2020-10-02

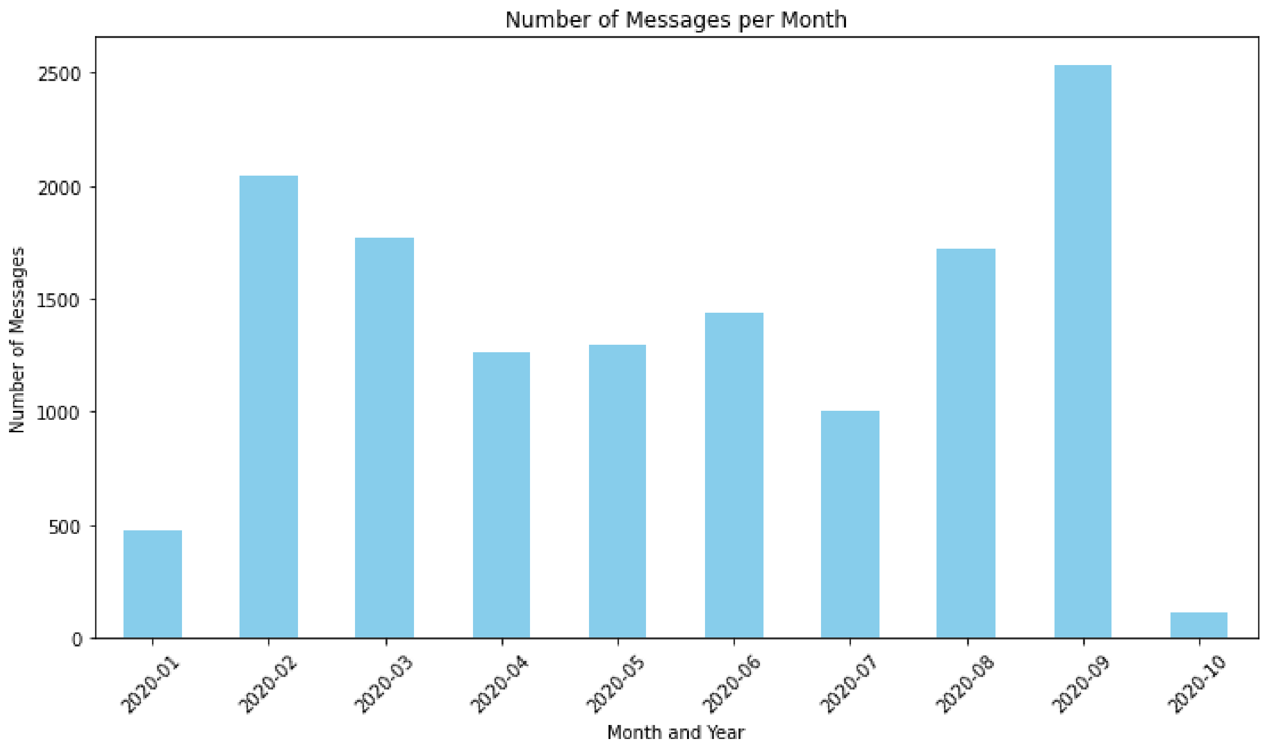
13655 rows × 7 columns

Message frequency distribution over various time period

Monthly level

In [49]:

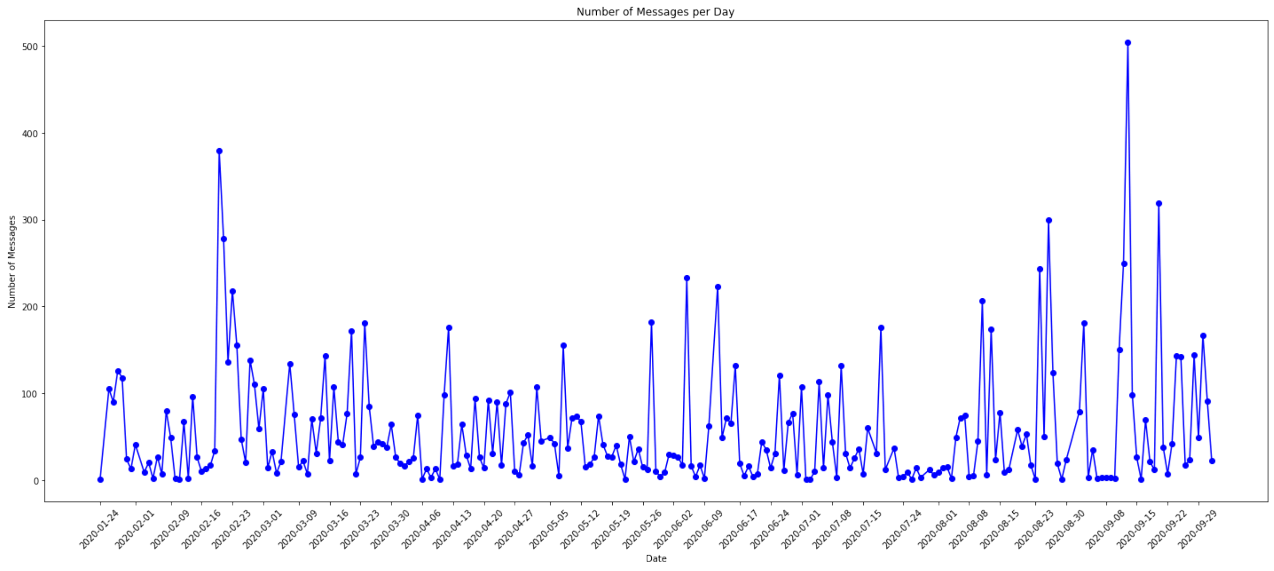
```
df['date_time'] = pd.to_datetime(df['date_time'])

# Extract month and year from the 'date_time' column
df['month_year'] = df['date_time'].dt.to_period('M')

# Calculate the number of messages per month and year
messages_per_month_year = df.groupby('month_year').size()

# Plotting the frequency chart
plt.figure(figsize=(10, 6))
messages_per_month_year.plot(kind='bar', color='skyblue')
plt.title('Number of Messages per Month')
plt.xlabel('Month and Year')
plt.ylabel('Number of Messages')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



Daily level distribution

In [50]:

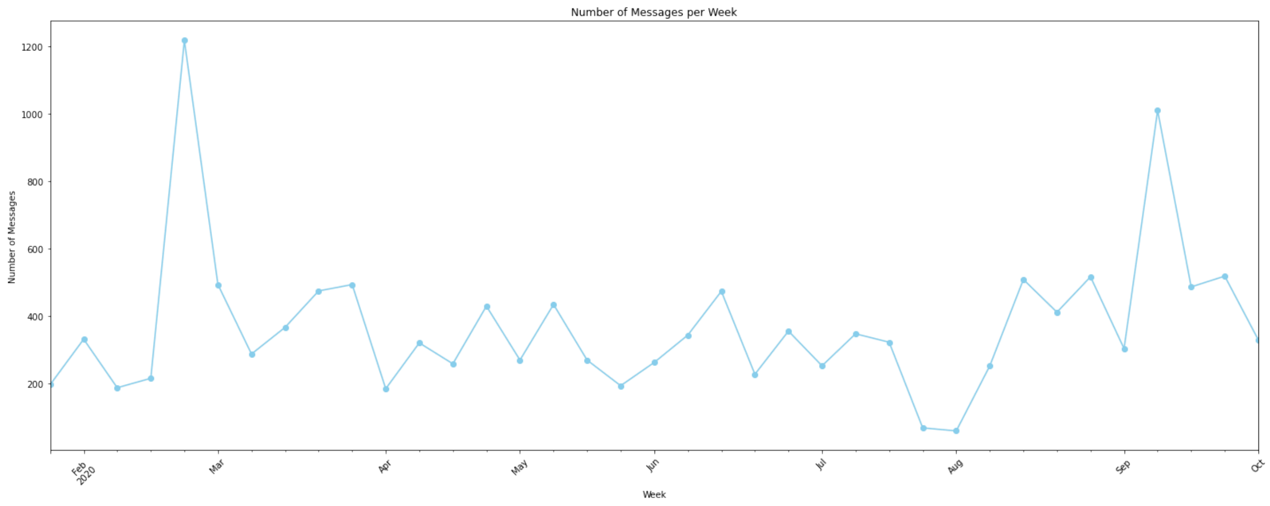
```
df['date_time'] = pd.to_datetime(df['date_time'])

# Group the DataFrame by day and calculate the number of messages per day
messages_per_day = df.groupby(df['date_time'].dt.date).size()

# Plotting the frequency chart
plt.figure(figsize=(20, 9))
messages_per_day.plot(kind='line', color='blue', marker='o')
plt.title('Number of Messages per Day')
plt.xlabel('Date')
plt.ylabel('Number of Messages')
plt.xticks(messages_per_day.index[::7], rotation=45)  # Show every 7th date on x-axis
plt.tight_layout()
plt.show()
```



Weekly level distribution

In [51]:

```
df['date_time'] = pd.to_datetime(df['date_time'])

# Group the DataFrame by week and calculate the number of messages per week
messages_per_week = df.resample('W-Mon', on='date_time').size()

# Plotting the frequency chart
plt.figure(figsize=(20, 8))
messages_per_week.plot(kind='line', color='skyblue', marker='o')
plt.title('Number of Messages per Week')
plt.xlabel('Week')
plt.ylabel('Number of Messages')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



Top Most Active weeks and there message count

In [66]:

```
df['date_time'] = pd.to_datetime(df['date_time'])
```

```
# Group the DataFrame by week and calculate the number of messages per week
messages_per_week = df.resample('W-Mon', on='date_time').size()

# Filter out weeks with zero messages
#messages_per_week = messages_per_week[messages_per_week > 0]
```

In [71]:
```
messages_per_week.nlargest(10) # top 10 most active weeks
```

Out[71]:
```
date_time
2020-02-24    1217
2020-09-14    1009
2020-09-28     518
2020-08-31     516
2020-08-17     508
2020-03-02     493
2020-03-30     493
2020-09-21     486
2020-03-23     474
2020-06-15     473
dtype: int64
```

Least Active weeks and there message weeks

In [99]:
```
messages_per_week.nsmallest(8)
```

Out[99]:
```
date_time
2020-08-03     59
2020-07-27     68
2020-04-06    184
2020-02-10    187
2020-05-25    193
2020-01-27    196
2020-02-17    215
2020-06-22    227
dtype: int64
```

Top 10 most Active Days

In [92]:
```
temp = df.groupby(df['date_time'].dt.date).size()
temp.nlargest(10)
# List of to 10 most active day for the whatsapp group
```

Out[92]:
```
date_time
2020-09-13    504
2020-02-20    379
2020-09-20    319
2020-08-26    299
2020-02-21    278
2020-09-12    249
2020-08-24    243
2020-06-05    233
2020-06-12    223
2020-02-23    218
dtype: int64
```

Least active days and there message count

In [96]:
```
temp.nsmallest(20)
```

Out[96]:
```
date_time
2020-01-24    1
2020-02-11    1
2020-04-06    1
2020-04-10    1
2020-05-22    1
2020-07-02    1
2020-07-03    1
2020-07-26    1
2020-08-23    1
2020-08-29    1
2020-09-16    1
2020-02-05    2
2020-02-10    2
2020-02-13    2
2020-06-09    2
2020-08-04    2
2020-09-06    2
2020-09-10    2
2020-04-08    3
2020-07-09    3
dtype: int64
```

Top 10 active users on the group.

Before, analysing that, we will see the *number of Ghosts* in the group.

In [102…
```
# 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

print(f"Number of people who haven't sent even a single message on the group are {237 -
```

```
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
```

Result

- Total number of people who have sent at least one message on the group are **154**.
- BUT, the total number of participants were **237**.
- **That means 81 people in the group have not sent even a single message throughout these 9 months and 13500+ messages.**

Top 10 Active user in group

In [105…
```
temp_df = df.copy()
temp_df = temp_df[temp_df.user != "group_notification"]
topdf = temp_df.groupby("user")["message"].count().sort_values(ascending=False)

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

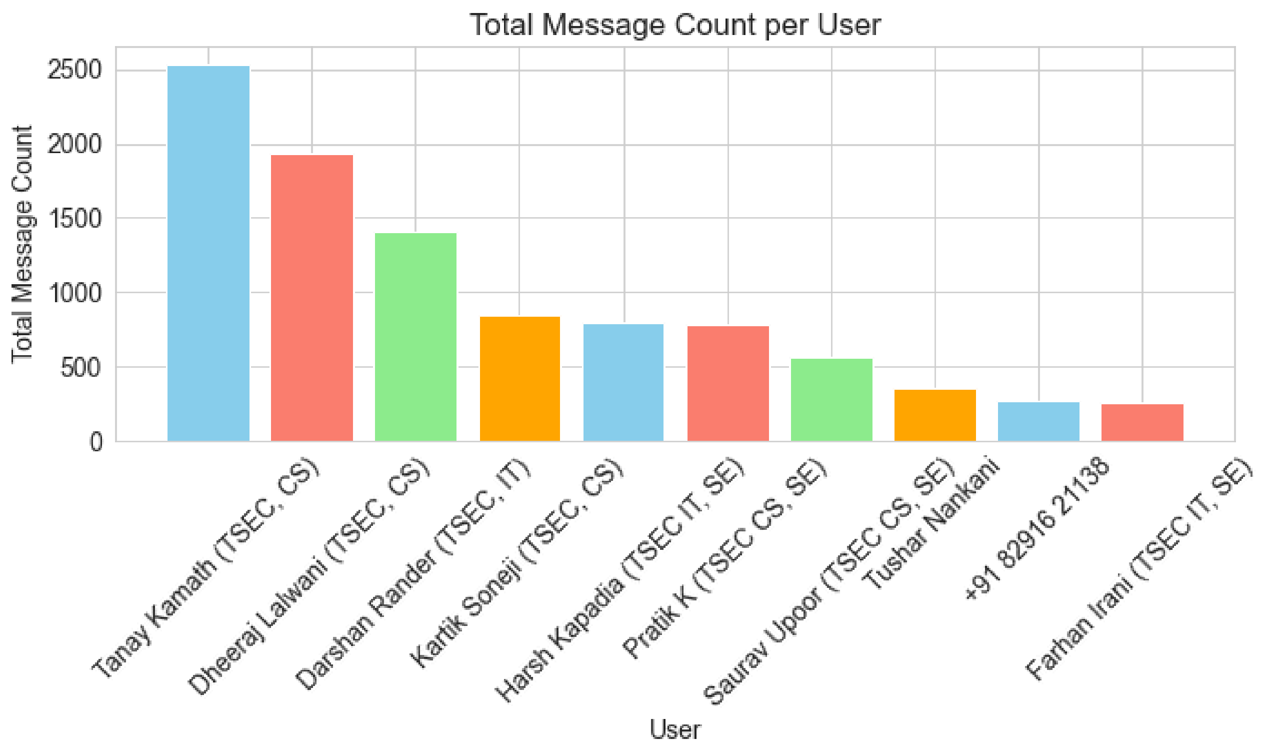
Out[105…

	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

Ploting Top 10 Active user {plotting in different ways}

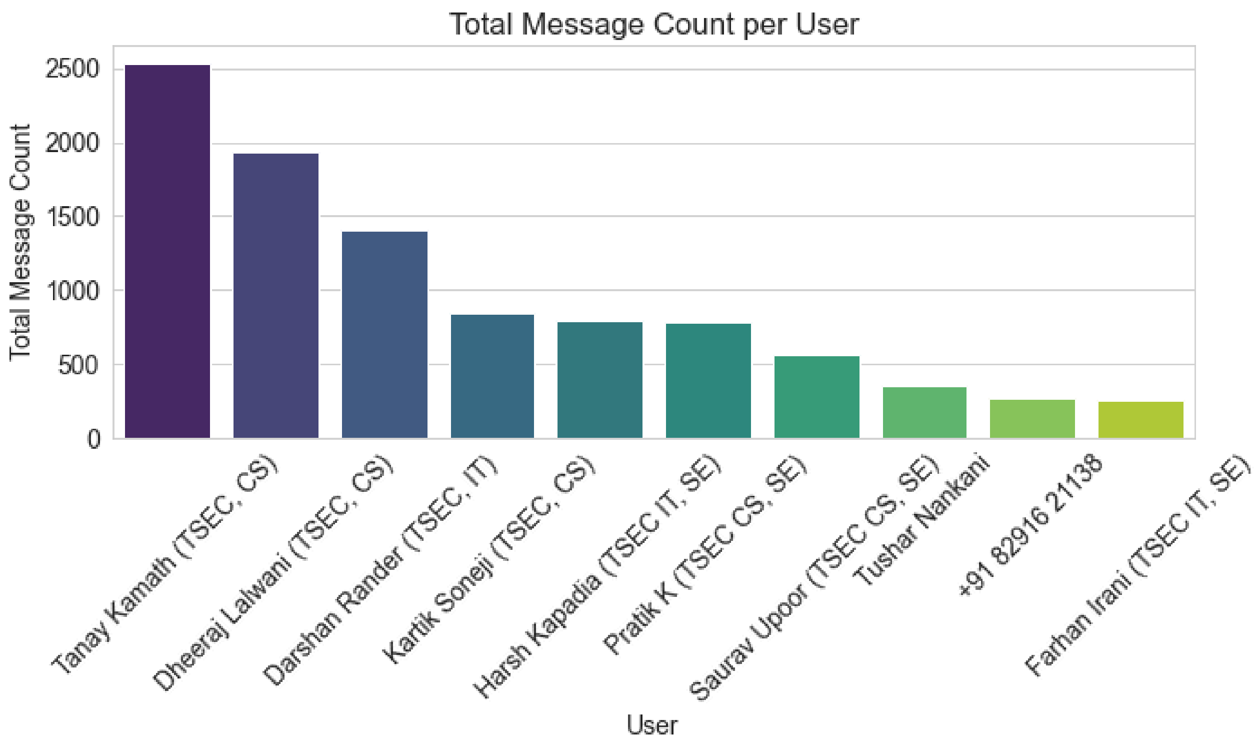
In [113…

```
plt.figure(figsize=(10, 6))
plt.bar(topdf['user'], topdf['message'], color=['skyblue', 'salmon', 'lightgreen', 'ora
plt.title('Total Message Count per User')
plt.xlabel('User')
plt.ylabel('Total Message Count')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



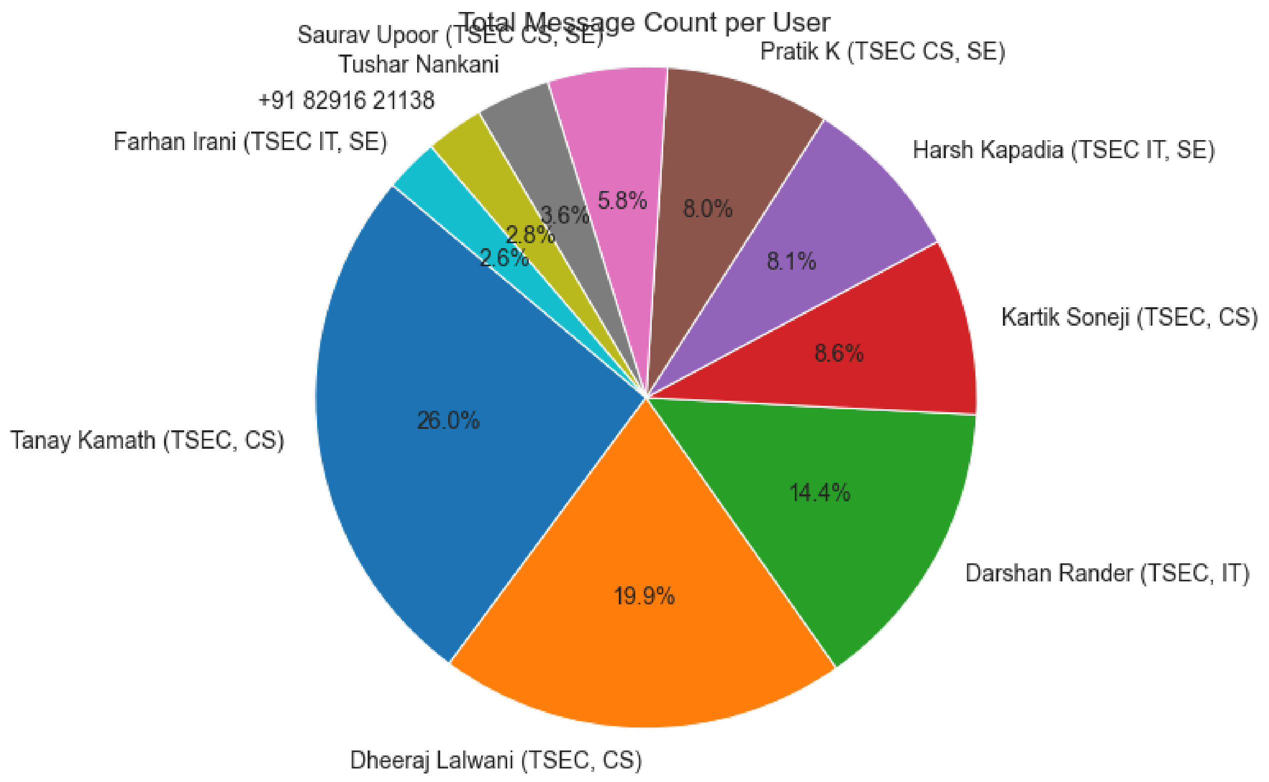
In [114…

```python
plt.figure(figsize=(10, 6))
sns.barplot(x='user', y='message', data=topdf, palette='viridis')
plt.title('Total Message Count per User')
plt.xlabel('User')
plt.ylabel('Total Message Count')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



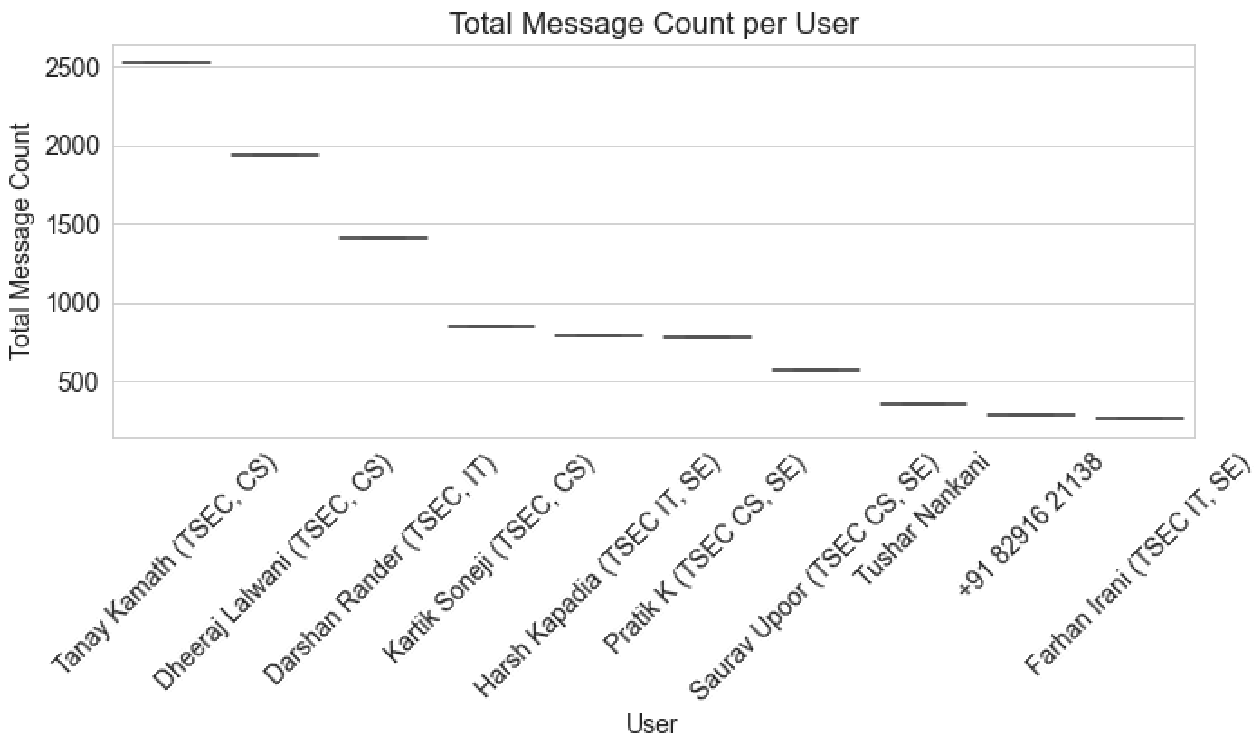
In [115…

```python
plt.figure(figsize=(8, 8))
plt.pie(topdf['message'], labels=topdf['user'], autopct='%1.1f%%', startangle=140)
plt.title('Total Message Count per User')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



In [116…

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='user', y='message', data=topdf, palette='Set2')
plt.title('Total Message Count per User')
plt.xlabel('User')
plt.ylabel('Total Message Count')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



Visualizing active user based on message length

In [125…

```python
user_message_count = df.groupby('user')['message'].count()

# Sort users based on message count in descending order and select top 10
top_10_users = user_message_count.sort_values(ascending=False).head(10)

# Calculate average message length for each of the top 10 users
average_message_length = []
for user in top_10_users.index:
    user_messages = df[df['user'] == user]['message']
    message_lengths = user_messages.str.len()
    average_message_length.append(message_lengths.mean())

# Create a DataFrame to display the results
result_df = pd.DataFrame({
    'User': top_10_users.index,
    'Total Message Count': top_10_users.values,
    'Average Message Length': average_message_length
})
```

In [126…

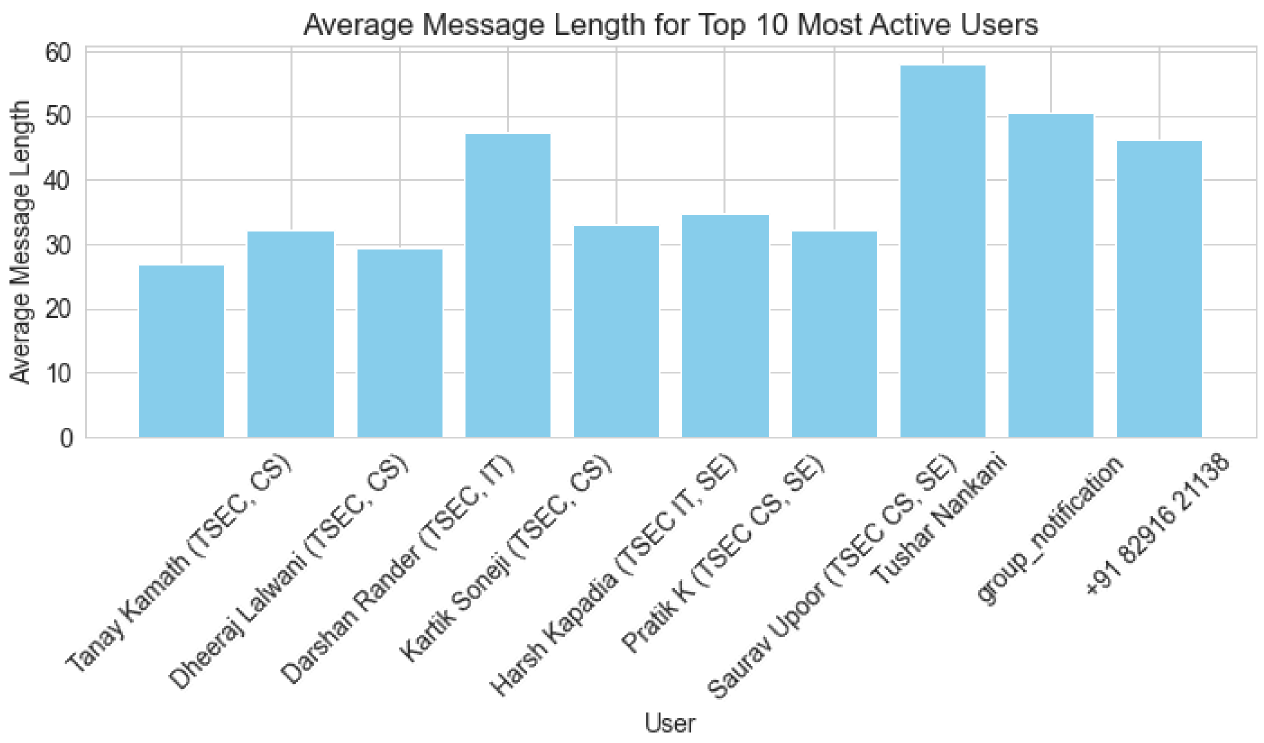
```python
result_df
```

Out[126…

	User	Total Message Count	Average Message Length
0	Tanay Kamath (TSEC, CS)	2528	27.045491
1	Dheeraj Lalwani (TSEC, CS)	1937	32.137842
2	Darshan Rander (TSEC, IT)	1404	29.472222
3	Kartik Soneji (TSEC, CS)	841	47.328181
4	Harsh Kapadia (TSEC IT, SE)	790	33.134177
5	Pratik K (TSEC CS, SE)	781	34.741357
6	Saurav Upoor (TSEC CS, SE)	569	32.289982
7	Tushar Nankani	354	57.920904
8	group_notification	276	50.539855
9	+91 82916 21138	275	46.320000

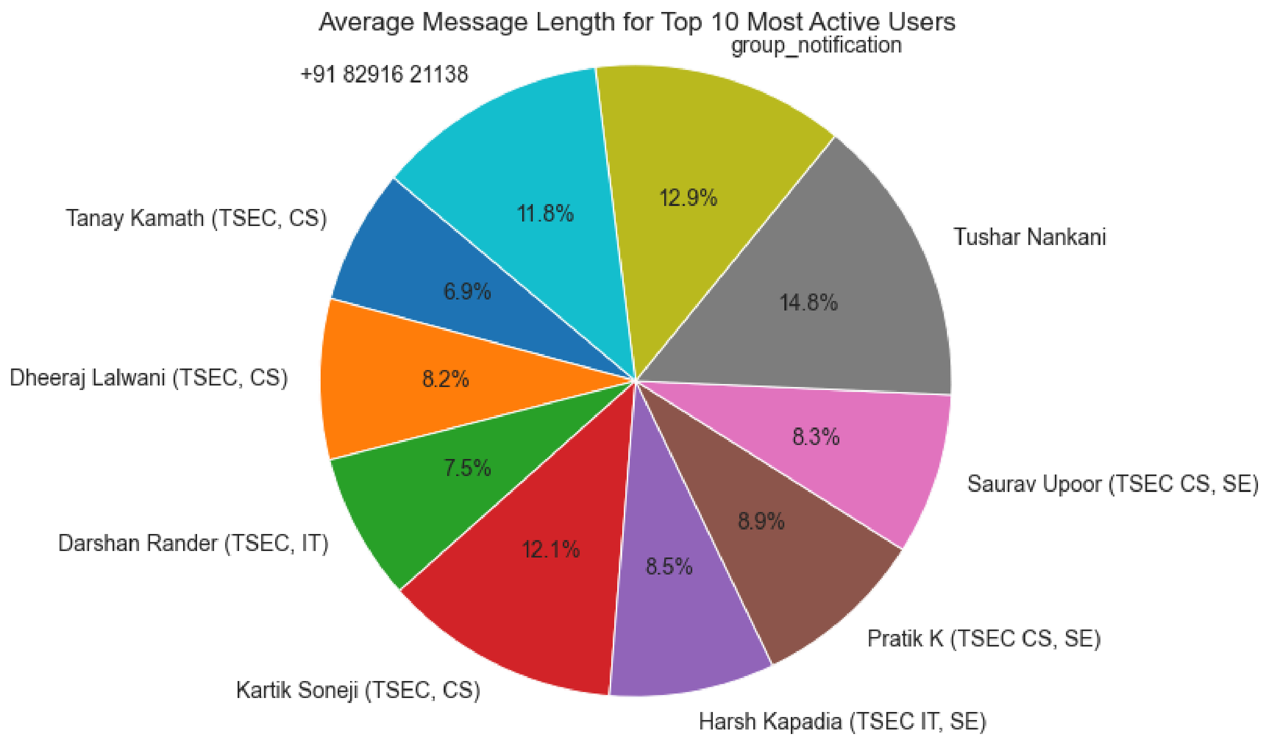
In [127…

```python
plt.figure(figsize=(10, 6))
plt.bar(result_df['User'], result_df['Average Message Length'], color='skyblue')
plt.title('Average Message Length for Top 10 Most Active Users')
plt.xlabel('User')
plt.ylabel('Average Message Length')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



In [128…

```
plt.figure(figsize=(8, 8))
plt.pie(result_df['Average Message Length'], labels=result_df['User'], autopct='%1.1f%%
plt.title('Average Message Length for Top 10 Most Active Users')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



Findings

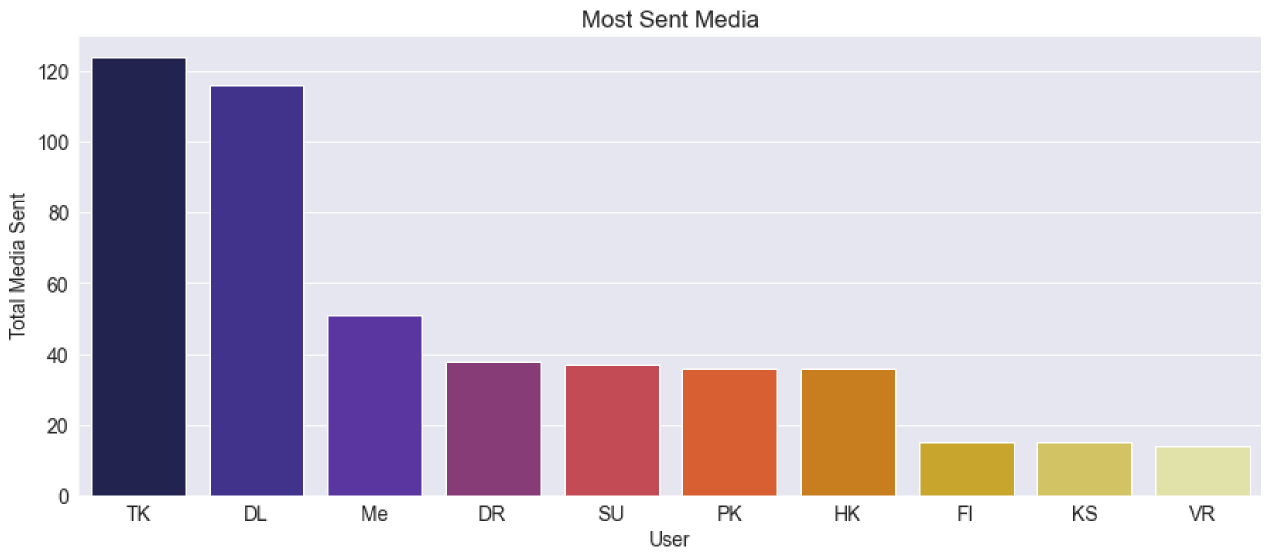
Tanay Kamath (TSEC, CS) person who has most number of message in the group has the least message lenght which means he sends broken messsages

At the same time Kartik Soneji (TSEC, CS) has less number of messages but a good message lenght which mean he send long and detailed messages

Top 10 users most sent media

In [131…

```
# Using `groupby`, `count` and `sort_values` attributes.
top10media = df[df.message == '<Media omitted> '].groupby('user').count().sort_values(b

# 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].spli

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


# 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')
```



Most frquently used emoji in the group

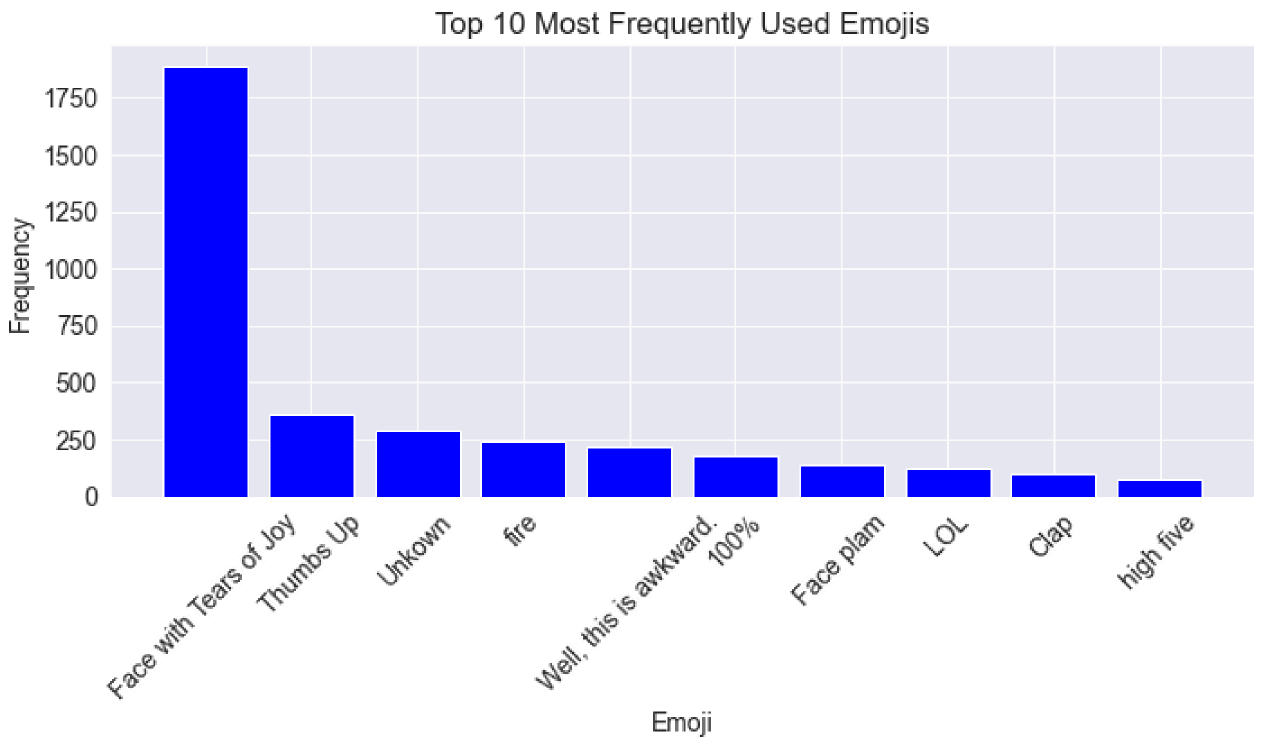
In [138…

```
from collections import Counter
import regex as re

ctr = Counter()
emoji_pattern = re.compile(r'[\U0001F300-\U0001F5FF\U0001F600-\U0001F64F\U0001F680-\U000
for idx, row in df.iterrows():
    emojis = emoji_pattern.findall(row["message"])
    for emoji_ in emojis:
        ctr[emoji_] += 1
# Get the most common emojis and their counts
most_common_emojis = ctr.most_common(10)
print("Top 10 Most Frequently Used Emojis:")
for emoji_, count in most_common_emojis:
    print(f"{emoji_}: {count} times")
```

```
Top 10 Most Frequently Used Emojis:
😂: 1886 times
👍: 364 times
⬚: 291 times
🔥: 244 times
😅: 220 times
💯: 180 times
🤦: 136 times
🤣: 128 times
👏: 101 times
🙏: 79 times
```

In [145…

```
# Get the most common emojis and their counts
most_common_emojis = ctr.most_common(10)

# Extract emojis and counts for plotting
emojis = [emoji[0] for emoji in most_common_emojis]
counts = [emoji[1] for emoji in most_common_emojis]

# Map emoji characters to their descriptions
emoji_descriptions = {
    "😊": "Smiling Face with Smiling Eyes",
    "❤️": "Red Heart",
    "😂": "Face with Tears of Joy",
    "👍": "Thumbs Up",
    "😍": "Smiling Face with Heart-Eyes",
        '?': 'Unkown',
        '🔥': 'fire',
        '😅': 'Well, this is awkward.',
        '💯': '100%',
        '🤦': 'Face plam',
        '🤣': 'LOL',
        '👏': 'Clap',
        '🙏': 'high five',
    # Add more emoji descriptions as needed
}

# Convert emojis to their descriptions for labeling
emojis_labels = [emoji_descriptions.get(emoji, "Unknown") for emoji in emojis]

# Plotting the bar chart
plt.figure(figsize=(10, 6))
plt.bar(emojis_labels, counts, color='blue')
plt.title('Top 10 Most Frequently Used Emojis')
plt.xlabel('Emoji')
plt.ylabel('Frequency')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



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

In [149…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13655 entries, 0 to 13654
Data columns (total 8 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
 3   day         13655 non-null  object
 4   month       13655 non-null  object
 5   year        13655 non-null  int32
 6   date        13655 non-null  object
 7   month_year  13655 non-null  period[M]
dtypes: datetime64[ns](1), int32(1), object(5), period[M](1)
memory usage: 800.2+ KB
```

Most Acitve Hour

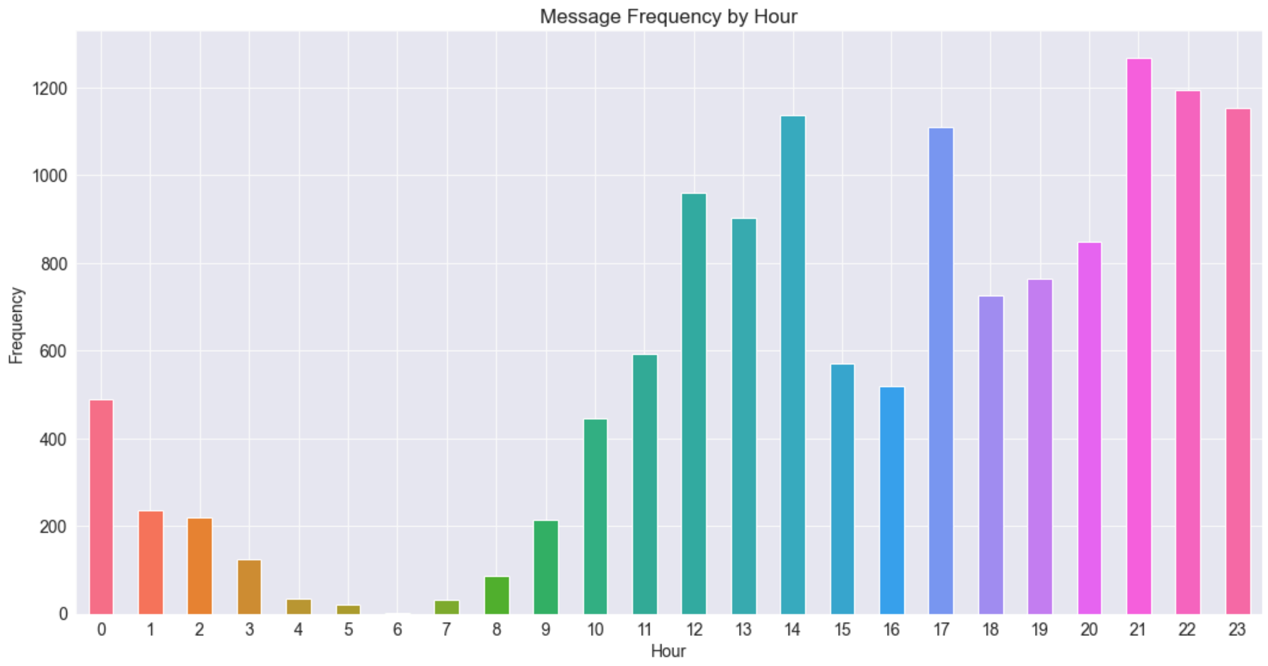
In [150…

```
df['hour'] = df['date_time'].dt.hour

# Count the occurrences of each hour
active_hours = df['hour'].value_counts()

# Find the most active hour
most_active_hour = active_hours.idxmax()

print("Most active hour:", most_active_hour)
```

```
Most active hour: 21
```

In [153…

```
palette = sns.color_palette("husl", len(active_hours))

# Plotting the frequency of messages for each hour
plt.figure(figsize=(15, 8))
active_hours.sort_index().plot(kind='bar', color=palette)
plt.title('Message Frequency by Hour')
plt.xlabel('Hour')
plt.ylabel('Frequency')
plt.xticks(rotation=0)  # Rotate x-axis labels
plt.tight_layout()
plt.show()
```



Most Active Day of the week

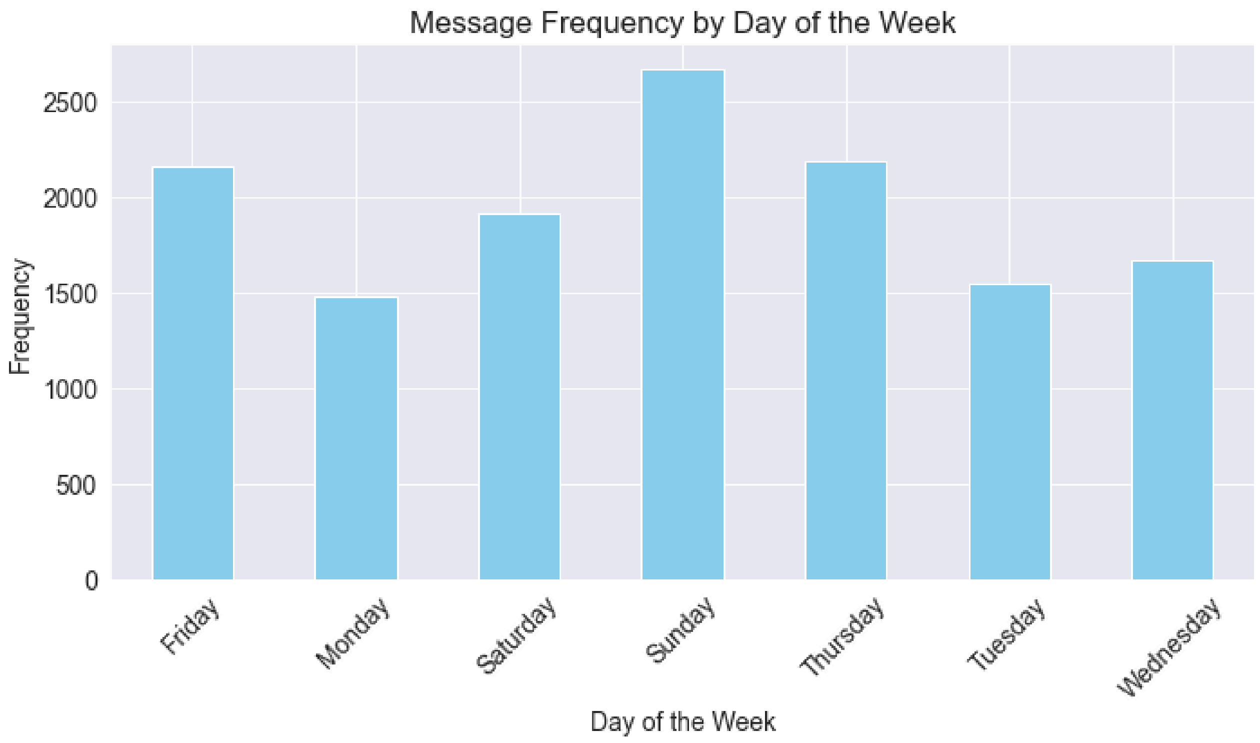
In [158…

```
df['day_of_week'] = df['date_time'].dt.day_name()

# Count the occurrences of each day of the week
active_days = df['day_of_week'].value_counts()

# Find the most active day of the week
most_active_day = active_days.idxmax()

print("Most active day of the week:", most_active_day)
```

```
Most active day of the week: Sunday
```

In [159…

```
plt.figure(figsize=(10, 6))
active_days.sort_index().plot(kind='bar', color='skyblue')
plt.title('Message Frequency by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Frequency')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



Most Active Month of the year

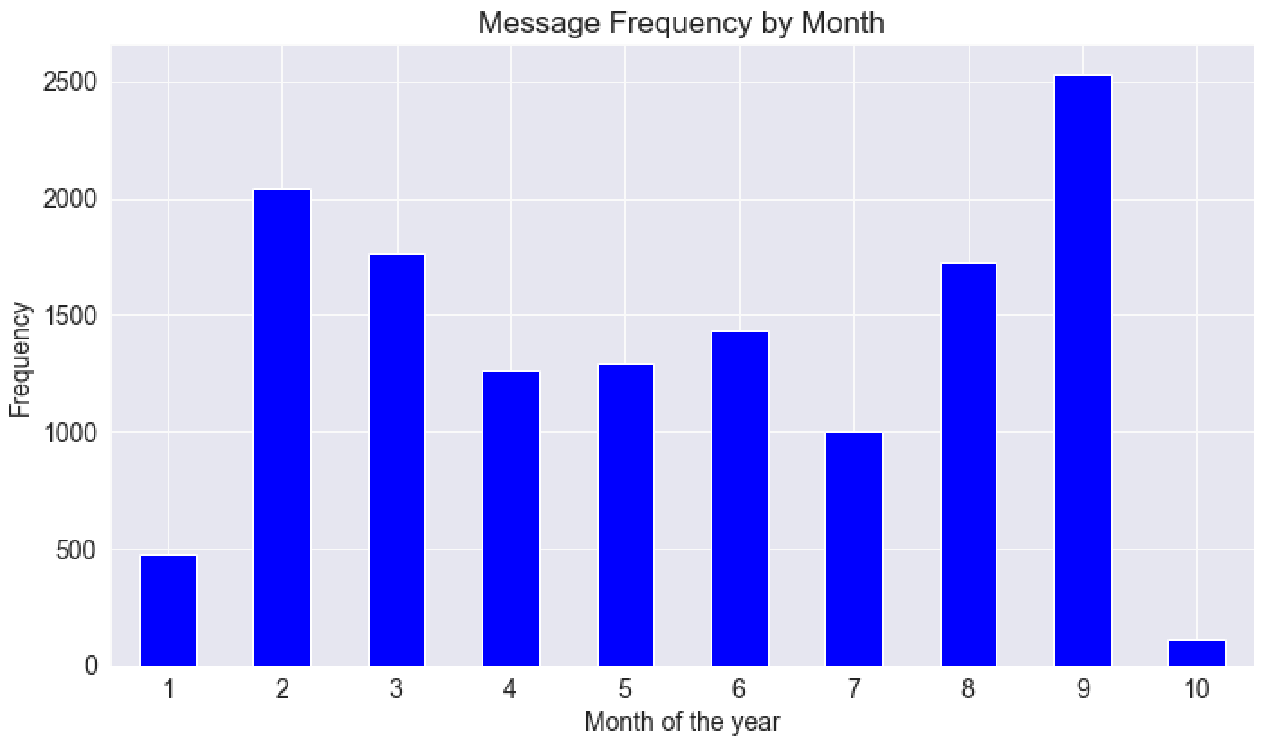
In [163…

```
df['month'] = df['date_time'].dt.month

# Count the occurrences of each month
active_months = df['month'].value_counts()

# Find the most active month of the year
most_active_month = active_months.idxmax()

print("Most active month of the year:", most_active_month)
```

```
Most active month of the year: 9
```

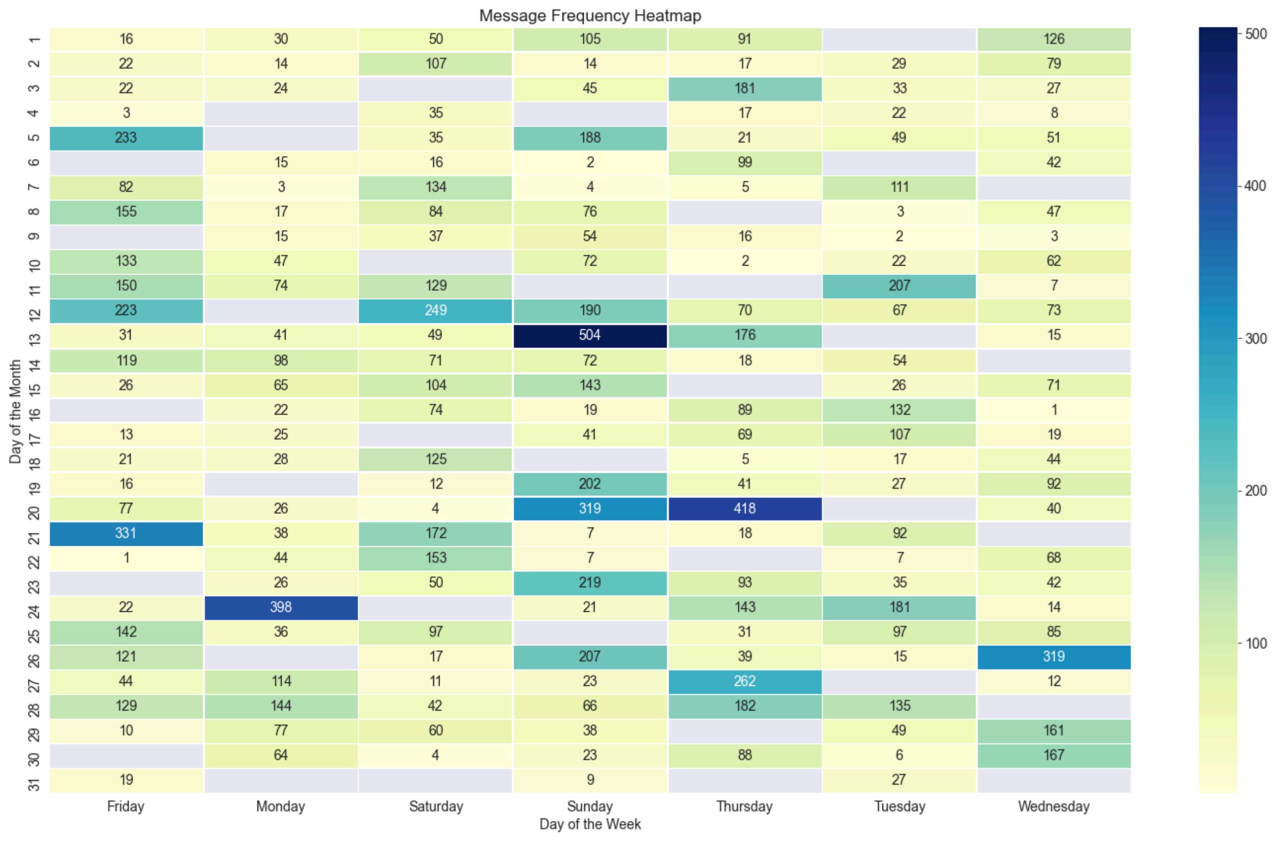
In [167…

```
# Plotting the frequency of messages for each month of the year
plt.figure(figsize=(10, 6))
active_months.sort_index().plot(kind='bar', color='blue')
plt.title('Message Frequency by Month')
plt.xlabel('Month of the year')
plt.ylabel('Frequency')
plt.xticks(rotation=0)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



In [170…

```
pivot_table = df.pivot_table(index='day_of_month', columns='day_of_week', values='month

# Plotting the heatmap
plt.figure(figsize=(20, 12))
sns.heatmap(pivot_table, cmap='YlGnBu', annot=True, fmt='g', linewidths=.5)
plt.title('Message Frequency Heatmap')
plt.xlabel('Day of the Week')
plt.ylabel('Day of the Month')
plt.tight_layout()
plt.show()
```



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!

Overall frequency of total messages on the group.**

Top 10 most active days.**

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!)

Top 10 users most sent media.**

- *Tanay Kamath* beats everyone by a mile!

Top 10 most used emojis.**

- using the `emoji` module!

Most active hours and weekdays.**

- Heatmaps of weekdays and months.
- Most active hours, weekdays, and months.

In [ ]:

```

```