```
pip install emoji
```

```
Collecting emoji
  Downloading emoji-2.14.0-py3-none-any.whl.metadata (5.7 kB)
Downloading emoji-2.14.0-py3-none-any.whl (586 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 586.9/586.9 kB 6.5 MB/s eta 0:00:00
Installing collected packages: emoji
Successfully installed emoji-2.14.0
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

NLTK (Natural Language Toolkit) is a popular Python library for natural language processing (NLP). It provides us various text processing libraries with a lot of test datasets.

Natural language ToolKit(NLTK) is used for doing NLP tasks such as removing stopwords, tokenizing words, etc.

Vader evaluates any given text and generates a positive, negative, or neutral score for each lexical feature. These scores are then added together to form a compound score, which is a matrix normalizing all scores from -1 to +1

```
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

Regular Expression, is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern.

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.

NumPy excels in creating N-dimension data objects and performing mathematical operations efficiently, while Pandas is renowned for data wrangling and its ability to handle large datasets.

Counter is an unordered collection where elements are stored as Dict keys and their count as dict value. Counter elements count can be positive, zero or negative integers. However there is no restriction on it's keys and values.

plot(): To create line plots. scatter(): To create scatter plots. bar(): To create bar charts. hist(): To create histograms. show(): To display the figure.

A comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for data visualization in scientific computing and data analysis.

generate(): Takes a string of text as input and creates the word cloud based on the frequency of words. to_file(): Saves the generated word cloud to an image file. recolor(): Recolors the words based on an image or color palette.

You can customize various aspects of the word cloud, including the shape, color, font, and the words to include or exclude. This is a predefined set of common words (like "the", "is", "in", etc.) that are often filtered out in text processing because they do not carry significant meaning. When creating a word cloud, you might want to exclude these stop words to focus on more meaningful words.

This class is used to generate colors for the words in the word cloud based on an input image.

```
import regex
import pandas as pd
import numpy as np

from collections import Counter
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

The pattern you provided is a regular expression (regex) that seems to be designed to match a specific date and time format. Let's break down the regex pattern

find_author(s) takes a string s as an argument. String Splitting: s.split(":") divides the string into parts wherever a colon appears.

```
def date_time(s):
  # Your regex pattern
    pattern = '^([0-9]+)(\/)([0-9]+)(\/)([0-9]+), ([0-9]+):([0-9]+)[ ]?(AM|PM|am|pm)? -'
    # Match the pattern against the string
    result = regex.match(pattern, s)
     # Check if there was a match and return True or False
    if result:
        return True
    return False

def find_author(s):
   # Split the string at the colon
    s = s.split(":")
    # Check if there are exactly two parts
    if len(s)==2:
        return True
    else:
        return False

def getDatapoint(line):
   # Split the line by the delimiter ' - '
    splitline = line.split(' - ')
    # Extract the date and time from the first part
    dateTime = splitline[0]

    date, time = dateTime.split(", ")
    # Combine the rest of the splitline into a message
    message = " ".join(splitline[1:])
     # Check if the message contains an author
    if find_author(message):
        splitmessage = message.split(": ")
        author = splitmessage[0]
        message = " ".join(splitmessage[1:])
    else:
        author= None
    return date, time, author, message
```

```
cd /content/drive/MyDrive/shilpa
```

```
/content/drive/MyDrive/shilpa
```

The with statement is used for resource management. It ensures that the file is properly closed after its suite finishes, even if an error occurs. This is generally preferred over manually opening and closing files.

```
data = []
conversation = '/content/drive/MyDrive/shilpa/chart45.txt'
with open(conversation, encoding="utf-8") as fp:
    fp.readline() # Optionally read the header or first line # Optionally skip the header or initial information
    messageBuffer = []
    date, time, author = None, None, None
    while True:
        line = fp.readline() # Read the next line
        if not line:  # Break if the line is empty (EOF)
            break
            # Process the line using your previously defined getDatapoint function
        line = line.strip() # Clean up any leading/trailing whitespace
        if date_time(line): # Check if this line indicates a new date/time
            if len(messageBuffer) > 0: # If there are messages to save
                data.append([date, time, author, ' '.join(messageBuffer)])
            messageBuffer.clear() # Clear the buffer for new messages
            date, time, author, message = getDatapoint(line)  # Extract new info
            messageBuffer.append(message) # Start the new message buffer
        else:
            messageBuffer.append(line)  # Add to the current message
```

```
df = pd.DataFrame(data, columns=["Date", 'Time', 'Author', 'Message']) # Create a DataFrame from the processed data
df['Date'] = pd.to_datetime(df['Date'])
print(df.tail(100)) # Corrected the typo from 'taiCl' to 'tail'
print(df.info()) # Display DataFrame information
print(df.Author.unique()) # Display unique authors
```

```
           Date      Time           Author  \
3437 2022-05-04  12:55 pm  +91 96199 36420
3438 2022-05-04  12:55 pm  +91 96199 36420
3439 2022-05-04  12:56 pm  +91 90042 80656
3440 2022-05-04  12:56 pm  Monika Kharkwal
3441 2022-05-04  12:56 pm  +91 90042 80656
...         ...       ...              ...
3532 2024-04-27   8:26 am  +91 70214 89118
3533 2024-04-28   5:14 pm  +91 99676 09749
3534 2024-04-28   7:40 pm  +91 70214 89118
3535 2024-04-29   8:17 am             None
3536 2024-04-29   8:17 am  +91 97683 13673

                                      Message
3437                             Home security....
3438                      Change nahi karungaa 😅 abhe
3439                                    Bacha kya h
```

```
3440                              Tumhara project hna
3441                                 Topics batao toh
...                                               ...
3532  *TODAY IS THE LAST DAY TO GIVE YOUR WEB MINING...
3533             NLP ka project kon kon submitted kiya?
3534                                                😂
3535  +91 97683 13673: https://docs.google.com/sprea...
3536  Faculty ko forward kar ra hu....jisne bhi nhi ...

[100 rows x 4 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3537 entries, 0 to 3536
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Date     3537 non-null   datetime64[ns]
 1   Time     3537 non-null   object
 2   Author   3385 non-null   object
 3   Message  3537 non-null   object
dtypes: datetime64[ns](1), object(3)
memory usage: 110.7+ KB
None
[None 'Areej Clg' 'Pratiksha Awate' '+91 97683 13673' '+91 99672 73815'
 '+91 70214 89118' '+91 97695 24164' '+91 90040 75303' 'Preeti Clg Mumbai'
 'Shraddha Panchal' '+91 90042 80656' 'Anshu Clg Mumbai' '+91 87793 59887'
 '+91 70392 29744' 'Monika Kharkwal' '+91 82916 85824' '+91 99676 09749'
 '+91 84336 34677' '+91 77383 28626' 'Shilpa Dhanure' '+91 82916 75179'
 '+91 87790 51155' '+91 84259 79051' '+91 96199 36420' '+91 90047 57892'
 '+91 70214 77723']
<ipython-input-72-45e2fbe2ea1f>:2: UserWarning: Could not infer format, so each element will be parsed individually, falling back to
  df['Date'] = pd.to_datetime(df['Date'])
```

```
df.head() # Display the first few rows of the DataFrame
```

	Date	Time	Author	Message
0	2021-09-10	9:32 am	None	Areej Clg created group "notes~"
1	2021-10-18	9:07 am	None	You joined using this group's invite link
2	2021-10-18	10:02 am	Areej Clg	Ohh
3	2021-10-18	10:02 am	None	Areej Clg added +91 99676 09749
4	2021-10-18	10:57 am	Areej Clg	<Media omitted>

Next steps: Generate code with df View recommended plots New interactive sheet

```
total_messages = df.shape[0] # Count total messages
print(total_messages)
```

```
3537
```

```
media_messages = df[df["Message"]=='<Media omitted>'].shape[0]
print(media_messages) # Count media messages
```

```
314
```

```
import emoji
import regex

def split_count(text):
    emoji_list = []
    data = regex.findall(r'\X',text)
    for word in data:
        # Use emoji.is_emoji() to check for emojis
        if any(emoji.is_emoji(char) for char in word):
            emoji_list.append(word)
    return emoji_list

df['emoji'] = df["Message"].apply(split_count) # Apply the split_count function to each message

emojis = sum(df['emoji'].str.len()) # Count the total number of emojis
print(emojis)
```

```
789
```

```
URLPATTERN = r'(https?://\S+)' # Define the URL pattern
df['urlcount'] = df.Message.apply(lambda x: regex.findall(URLPATTERN, x)).str.len() # Count URLs in each message
links = np.sum(df.urlcount) # Calculate total links
```

```
print("Chats betweent Areej Clg and Pratiksha Awate")
print("Total Messages: ", total_messages) # Assuming total_messages and media_messages are defined previously
print("Number of Media Shared: ", media_messages)

print("Number of Links Shared", links)
```

```
Chats betweent Areej Clg and Pratiksha Awate
Total Messages:  3537
Number of Media Shared:  314
Number of Links Shared 74
```

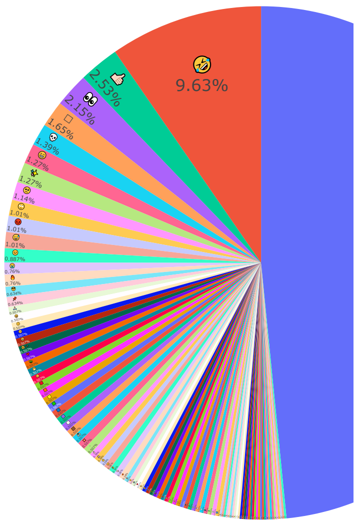
```
media_messages_df = df[df['Message'] == '<Media omitted>'] # Filter out media messages
messages_df = df.drop(media_messages_df.index)
messages_df['Letter_Count'] = messages_df['Message'].apply(lambda s : len(s)) # Calculate letter count
messages_df['Word_Count'] = messages_df['Message'].apply(lambda s : len(s.split(' '))) # Calculate word count
messages_df["MessageCount"]=1 # Create a MessageCount column
```

Start coding or generate with AI.

```
total_emojis_list = list(set([a for b in messages_df.emoji for a in b]))  # Extract unique emojis
total_emojis = len(total_emojis_list)

total_emojis_list = list([a for b in messages_df.emoji for a in b]) # Count all emojis
emoji_dict = dict(Counter(total_emojis_list))
emoji_dict = sorted(emoji_dict.items(), key=lambda x: x[1], reverse=True)
for i in emoji_dict:
  print(i)

emoji_df = pd.DataFrame(emoji_dict, columns=['emoji', 'count']) # Create DataFrame for emojis
import plotly.express as px
fig = px.pie(emoji_df, values='count', names='emoji') # Plotting with Plotly
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```

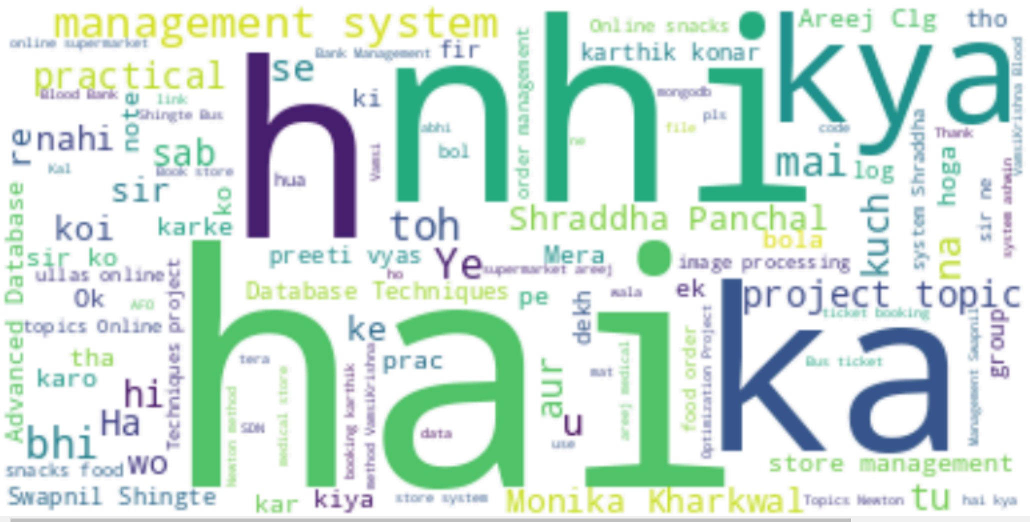
```
('😂', 382)
('🤣', 76)
('👍🏻', 20)
('👀', 17)
('⍰', 13)
('💀', 11)
('😁', 10)
('✨', 10)
('🥺', 9)
('🙄', 8)
('😡', 8)
('😅', 8)
('😒', 7)
('😭', 6)
('🔥', 6)
('😎', 5)
('📌', 5)
('🙏🏻', 4)
('😆', 4)
('🤨', 4)
('😐', 4)
('😬', 4)
('😵\u200d💫', 4)
('💔', 4)
('🦎', 4)
('🤧', 4)
('😌', 4)
('🟥', 4)
('🟧', 4)
('🟨', 4)
('🟩', 4)
('🟦', 4)
('🟪', 4)
('⬜', 4)
('🟫', 4)
('🙌🏻', 4)
('💥', 4)
('▫', 4)
('✌🏻', 3)
('🤦\u200d♀', 3)
('🙃', 3)
('😊', 3)
('👌', 3)
('👇🏻', 3)
('🔶', 3)
('🤦', 2)
('🌝', 2)
('🤷\u200d♂', 2)
('❤', 2)
('🙂', 2)
('😔', 2)
('🤦🏻\u200d♂', 2)
('🐍', 2)
('🐛', 2)
('😶', 2)
('🚨', 2)
('🤫', 2)
('♥', 2)
('👍', 2)
('😑', 2)
('🛡', 2)
('💻', 2)
('📍', 2)
('💸', 2)
('📅', 2)
('⏱', 2)
('📱', 2)
('🔰', 2)
('📑', 2)
('⏺', 2)
('😄', 1)
('😩', 1)
('😇', 1)
('🤔', 1)
('😍', 1)
('😉', 1)
('🥰', 1)
('😞', 1)
('⍰⍰', 1)
('🤝', 1)
('🥱', 1)
('🤕', 1)
('🙇🏻\u200d♂', 1)
('😏', 1)
('😮', 1)
('🤷🏼\u200d♀', 1)
('😢', 1)
('✌', 1)
('🥴', 1)
('😱', 1)
```

```
('🧐', 1)
('🏫', 1)
('📝', 1)
('🆓', 1)
('📜', 1)
('💰', 1)
('🌐', 1)
('🙏', 1)
('☝🏻', 1)
('🤟', 1)
('🥳', 1)
('🤭', 1)
('👆🏻', 1)
```



```
text = " ".join(review for review in messages_df.Message) # Combine all messages into a single string
print ("There are {} words in all the messages.".format(len(text))) # Print the total word count
stopwords = set(STOPWORDS) # Define stopwords
# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
# Display the generated image:
# the matplotlib way:
plt.figure( figsize=(10,5))# Display the generated image using Matplotlib
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off") # Turn off the axis  # Hide the axes
plt.show()
```

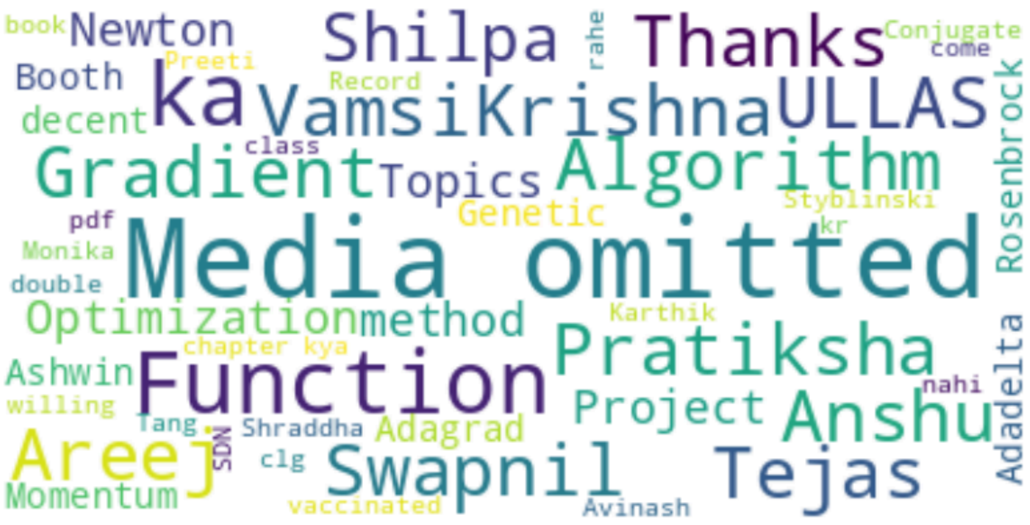
```
There are 161033 words in all the messages.
```



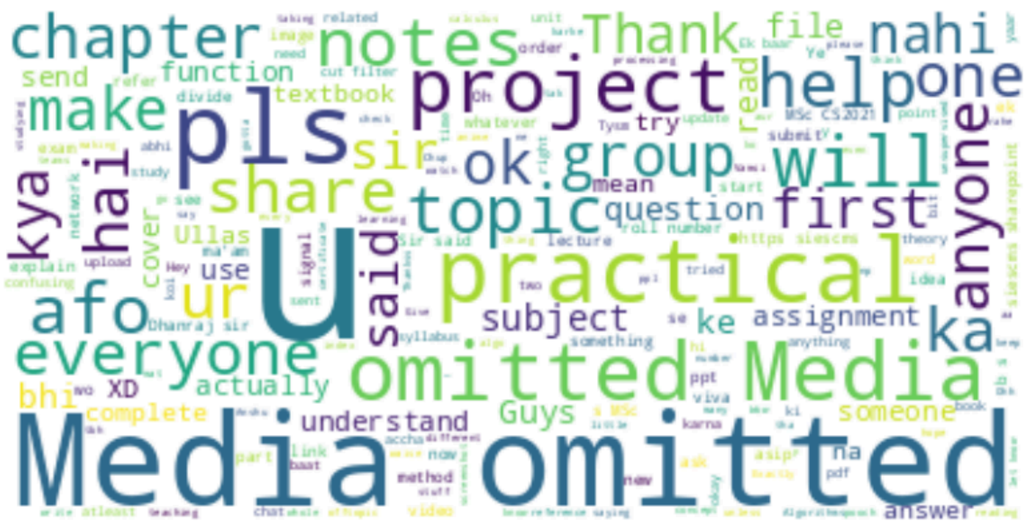
```
messages_df = df  # Assign the dataframe 'df' to 'messages_df' to make the 'Author' column available
l = ["Shilpa Dhanure","Areej Clg", "Pratiksha Awate","Monika Kharkwal"]
for i in range(len(l)):
  dummy_df = messages_df[messages_df['Author'] == l[i]]
  text = " ".join(review for review in dummy_df.Message)
  stopwords = set(STOPWORDS)
```

```
    # Generate a word cloud image
    print('Author name',l[i])

    # Check if text is empty after stop word removal
    words = [word for word in text.split() if word.lower() not in stopwords]
    if len(words) == 0:
      print(f"No words found for author {l[i]} after removing stop words. Skipping word cloud generation.")
      continue  # Skip to next author if no words are found

    wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(" ".join(words))
    # Display the generated image
    plt.figure( figsize=(10,5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()
```

```
Author name Shilpa Dhanure
```



```
Author name Areej Clg
```



```
Author name Pratiksha Awate
```

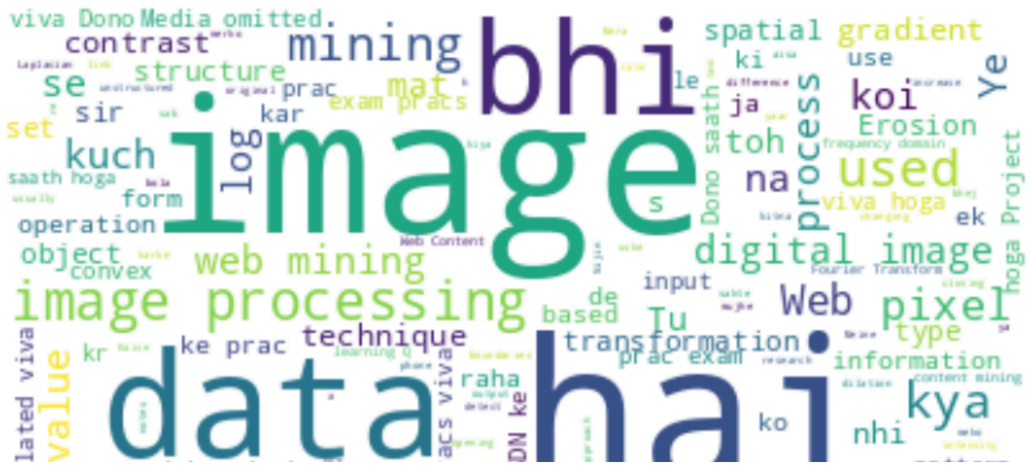


image
bhi
data
hai
image processing
digital image
mining
web mining
pixel
contrast
structure
spatial
gradient
process
used
koi
Erosion
transformation
information
kya
technique
log
kuch
operation
object
convex
Web
value
Project
viva
Dono
Media
omitted
sir
se
set
form
input
type
raha
ke prac
prac exam
based
Ye