Helper.py successful code

from urlextract import URLExtract  
import pandas as pd  
from collections import Counter  
import emoji  
  
extract = URLExtract()  
  
  
def fetch\_stats(selected\_user,df):  
  
 if selected\_user !='Overall':  
 df = df[df['user']== selected\_user]  
  
 num\_messages = df.shape[0]  
  
 words = []  
 for message in df['message']:  
 words.extend(message.split())  
  
 num\_media\_messages = df[df['message'] == '<Media omitted>\n'].shape[0]  
  
 links = []  
 for message in df['message']:  
 links.extend(extract.find\_urls(message))  
  
 return num\_messages,len(words),num\_media\_messages, len(links)  
  
def most\_busy\_users(df):  
 x = df['user'].value\_counts().head()  
  
 df = round((df['user'].value\_counts() / df.shape[0]) \* 100, 2).reset\_index().rename(  
 columns={'index': 'name', 'user': 'percent'})  
 return x,df  
  
def most\_common\_words(selected\_user,df):  
  
 f = open('stop\_hinglish.txt','r')  
 stop\_words = f.read()  
  
 if selected\_user != 'Overall':  
 df = df[df['user'] == selected\_user]  
  
 temp = df[df['user'] != 'group\_notification']  
 temp= temp[temp['message'] != '<Media omitted>\n']  
  
 words = []  
  
 for message in temp['message']:  
 for word in message.lower().split():  
 if word not in stop\_words:  
 words.append(word)  
  
 most\_common\_df = pd.DataFrame(Counter(words).most\_common(25))  
 return most\_common\_df  
  
def emoji\_helper(selected\_user,df):  
 if selected\_user !='Overall':  
 df =df[df['user'] == selected\_user]  
  
 emojis = []  
 for message in df['message']:  
 emojis.extend([c for c in message if c in emoji.EMOJI\_DATA])  
  
 emoji\_df = pd.DataFrame(Counter(emojis).most\_common(len(Counter(emojis))))  
  
 return emoji\_df  
  
def monthly\_timeline(selected\_user,df):  
 if selected\_user !='Overall':  
 df =df[df['user'] == selected\_user]  
  
 timeline = df.groupby(['year', 'month\_num', 'month']).count()['message'].reset\_index()  
  
 time = []  
 for i in range(timeline.shape[0]):  
 time.append(timeline['month'][i] + "-" + str(timeline['year'][i]))  
  
 timeline['time'] = time  
  
 return timeline  
  
  
def daily\_timeline(selected\_user,df):  
  
 if selected\_user != 'Overall':  
 df = df[df['user'] == selected\_user]  
  
 daily\_timeline = df.groupby('only\_date').count()['message'].reset\_index()  
  
 return daily\_timeline  
  
def week\_activity\_map(selected\_user,df):  
 if selected\_user != 'Overall':  
 df = df[df['user'] == selected\_user]  
  
 return df['day\_name'].value\_counts()  
  
def month\_activity\_map(selected\_user,df):  
  
 if selected\_user != 'Overall':  
 df = df[df['user'] == selected\_user]  
  
 return df['month'].value\_counts()  
  
def activity\_heatmap(selected\_user,df):  
  
 if selected\_user != 'Overall':  
 df = df[df['user'] == selected\_user]  
  
 user\_heatmap = df.pivot\_table(index='day\_name', columns='period', values='message', aggfunc='count').fillna(0)  
  
 return user\_heatmap