```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

File Handling - Parsing the ssaved file and converting it into a dataframe

```
import pandas as pd

log = '/content/drive/MyDrive/VRV/sample.log.txt' #file path of the sample.log

log_data = [] #initializing an empty list

with open(log, 'r') as file:  #opening the file in the read mode
    for line in file:
        parts = line.strip().split(' ') #splitting each line into parts using spaces

        ip = parts[0]  #index 0 indicates the first part that corresponds to the IP address
        timestamp = line[line.find("[") + 1:line.find("]")] #finding the substring between the [] that corresponds to timestamp
        request = " ".join(parts[5:8]).strip('"') #conctenating the methid, url, protocol data at present in between 6th to 8th parts
        statuscode = int(parts[8]) #datatype conversion as status code is a numerical data
        size = int(parts[9]) #datatype conversion of size at the 9th index
        message = " ".join(parts[10:]).strip() if len(parts) > 10 else None #concatenating the remaing part


#appending all the extracted data to the list as dictionary
        log_data.append({
            "IP" : ip,
            "Timestamp" : timestamp,
            "Request" : request,
            "Status Code" : statuscode,
            "Size" : size,
            "Message" : message
        })

#creating a dataframe
df = pd.DataFrame(log_data)

#displays the first five data of the created data frame. Here, this is used to verify if the parsing is done properly and the dataframe i
print(df.head())
```

```
               IP                   Timestamp                  Request  \
0    192.168.1.1  03/Dec/2024:10:12:34 +0000       GET /home HTTP/1.1
1    203.0.113.5  03/Dec/2024:10:12:35 +0000     POST /login HTTP/1.1
2       10.0.0.2  03/Dec/2024:10:12:36 +0000      GET /about HTTP/1.1
3    192.168.1.1  03/Dec/2024:10:12:37 +0000    GET /contact HTTP/1.1
4  198.51.100.23  03/Dec/2024:10:12:38 +0000  POST /register HTTP/1.1

   Status Code  Size                Message
0          200   512                   None
1          401   128  "Invalid credentials"
2          200   256                   None
3          200   312                   None
4          200   128                   None
```

Number of requests made by each IP address, Sort and display the results in descending order of request counts

```
df_requestcounts = df.groupby('IP')['Request'].count().reset_index(name = 'Request Count') #grouping the similar IP adresses together and
print((df_requestcounts.sort_values(by = 'Request Count', ascending=False)).to_string(index = False)) #sorting the ouput in descending or
```

```
            IP  Request Count
198.51.100.23              8
  203.0.113.5              8
  192.168.1.1              7
     10.0.0.2              6
192.168.1.100              5
```

```
df_requestcounts.to_csv('log_analysis_results.csv', index = False) #saving dataframe to the final CSV
```

Most Frequently Accessed Endpoint

```
df_endpoints = df.groupby('Request').size().reset_index(name = 'Frequency').sort_values(by = 'Frequency', ascending=False)

most_accessed = df_endpoints.iloc[0] #first row of the dataframe as it gives the endpoint with max frequency

endpoint = most_accessed['Request'].split()[1] #extacting only the endpoint from Request Data. e,g. only /home is fethched from "GET /home

count = most_accessed['Frequency'] #gets the count of the number of times that endpoint has been accessed

print(f"Most Frequently Accessed Endpoint: \n{endpoint} (Accessed {count} times)") #formatted string method to display the ouput as stated


```

```
Most Frequently Accessed Endpoint:
/login (Accessed 13 times)
```

```
df_endpoint_counts = df.groupby('Request').size().reset_index(name = 'Access Count').sort_values(by = 'Access Count', ascending=False)

df_endpoint_counts['Endpoint'] = df_endpoint_counts['Request'].str.split().str[1] #another method to extract only the end point

df_endpoint_counts = df_endpoint_counts[['Endpoint','Access Count']] #reording of the columns

print(df_endpoint_counts.to_string(index = False)) #displays the columns elected without index

df_endpoint_counts.to_csv('log_analysis_results.csv', mode='a', index = False) #saving to final CSV. Mode='a' appends this ouput to the ou
```

```
   Endpoint  Access Count
     /login            13
     /about             5
      /home             5
 /dashboard             3
   /contact             2
   /profile             2
  /feedback             2
  /register             2
```

Suspicious Activity

```
df_failed_login = df[(df['Status Code'] == 401) | (df['Message'] == "Invalid credentials")] #fecthcing the rows that satify either of the
#'|' represent 'or'

#grouping the above fetched rows based on the IP address and sorting them in descending order
df_suspicious_activity = df_failed_login.groupby('IP').size().reset_index(name = 'Failed Login Attempts').sort_values(by = 'Failed Login A

print(df_suspicious_activity.to_string(index = False))
```

```
            IP  Failed Login Attempts
   203.0.113.5                      8
192.168.1.100                      5
```

```
df_flagged_IP = df_suspicious_activity[df_suspicious_activity['Failed Login Attempts'] >= 10] #fagging those IP adresses that has a count

print(f"Suspicious Activity Detected: \n{df_flagged_IP.to_string(index = False)}")
```

```
Suspicious Activity Detected:
Empty DataFrame
Columns: [IP, Failed Login Attempts]
Index: []
```

The empty string is returned as no IP addresses has a count equal to or greater than 10 failed login attempts

```
df_suspicious_activity.to_csv('log_analysis_results.csv', mode='a', index = False) #appending the output of this to the final CSV
```