Лабораторна 1

«WEB-аналітика»

ФБ-31мп Синчук Владислав

Завдання: На основі будь-якого access.log сформувати датасет, що надав би інформацію про користувачів веб-ресурсу

Виконати наступні кроки:

Функції для парсингу та перетворення файлу логів в датафрейм:

*def* parse\_line(*line*: str) -> dict | None:

    regex = *r*'(?P<ip>\d+\.\d+\.\d+\.\d+) - - \[(?P<timestamp>.\*?)\] "(?P<request>.\*?)" (?P<status>\d+) (?P<bytes\_sent>\d+) "(?P<referrer>.\*?)" "(?P<user\_agent>.\*?)"'

    match = re.match(regex, *line*)

    return match.groupdict() if match else None

*def* parse\_file(*filepath*: str) -> pd.DataFrame:

    with open(*filepath*, 'r') as file:

        df\_rows = []

        for line in file:

            row = parse\_line(line)

            # print(row)

            if not(row is None):

                df\_rows.append(row)

    df = pd.DataFrame(df\_rows)

    # етап обробки датафрейму

    df['timestamp'] = pd.to\_datetime(df['timestamp'],

*format*='%d/%b/%Y:%H:%M:%S %z', *errors*='coerce')

    df['status'] = pd.to\_numeric(df['status'], *errors*='coerce')

    df['bytes\_sent'] = pd.to\_numeric(df['bytes\_sent'], *errors*='coerce')

    df['date'] = df['timestamp'].dt.date

    df['endpoint'] = df['request'].apply(*lambda* *x*: *x*.split()[1])

    return copy.deepcopy(df)

Виконаємо функції

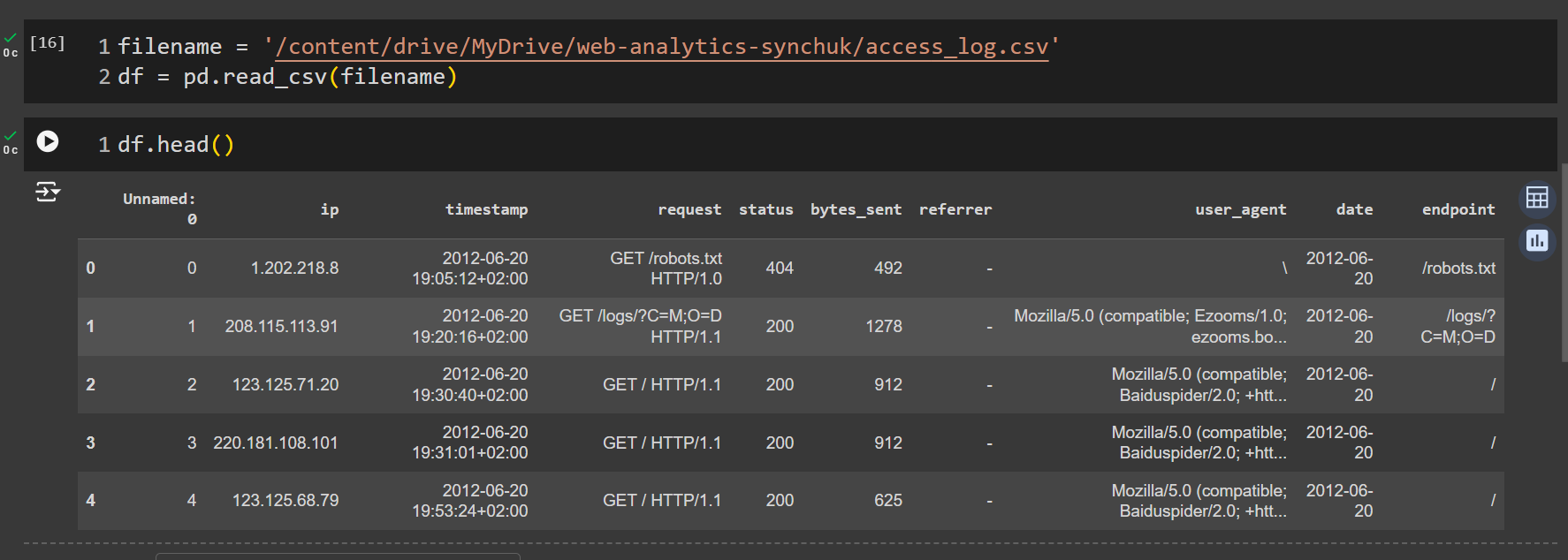
# ======= task a

    filepath = 'lab-1/access.log'

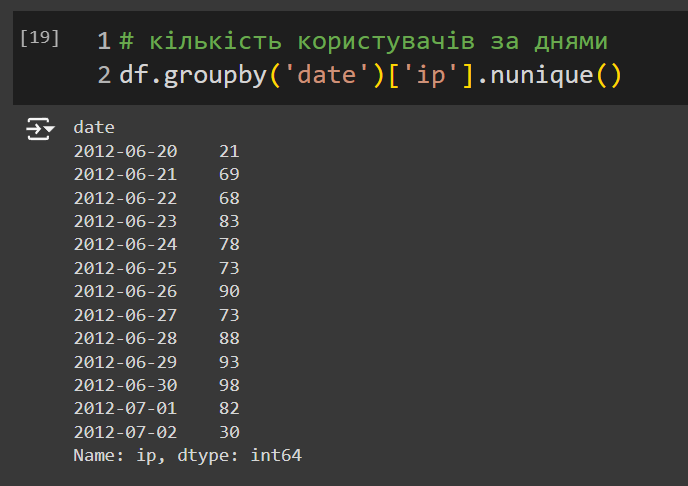
    df = parse\_file(filepath)

    df.to\_csv('lab-1/access\_log.csv')

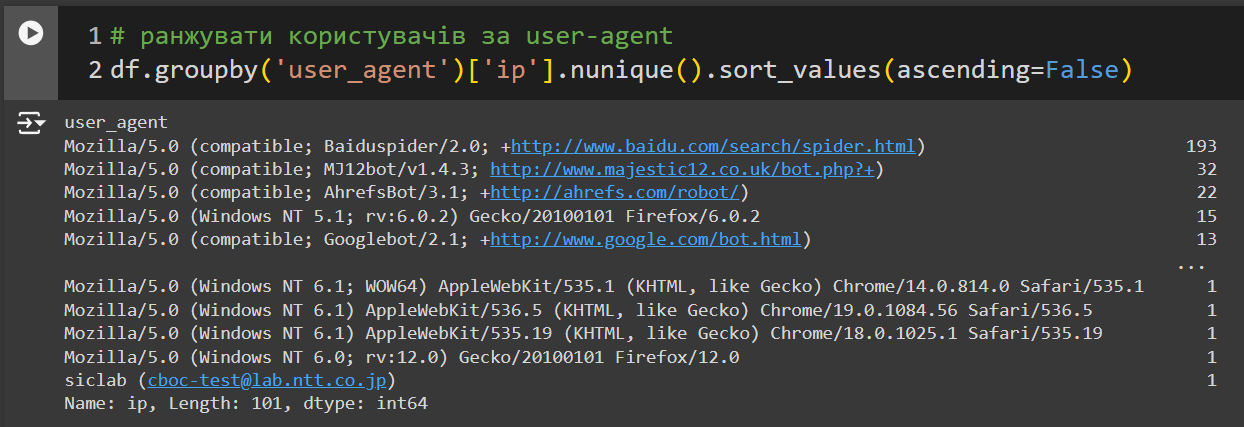
Прочитаємо та переглянемо збережений датафрейм:



1. Визначити кількість користувачів за днями



1. Ранжувати користувачів за User-Agent



1. Ранжувати користувачів за операційними системами

# ранжувати користувачів за операційними системами

# ex. : Mozilla/5.0 (Windows NT 5.1; rv:6.0.2) Gecko/20100101 Firefox/6.0.2

# => Windows                                      15

def extract\_parentheses\_content(user\_agent\_str: str) -> str | None:

    pattern = r'\((.\*?)\)'

    match\_ = re.search(pattern, user\_agent\_str)

    return match\_.group(1) if match\_ else None

def identify\_os(content: str) -> str:

    # Define patterns for known operating systems

    os\_patterns = {

        'Windows': r'Windows',

        'Mac OS': r'Macintosh|Mac OS|Mac\_PowerPC',

        'Linux': r'Linux',

        'Android': r'Android',

        'iOS': r'iPhone|iPad|iPod',

        'Chrome OS': r'CrOS'

    }

    for os, pattern in os\_patterns.items():

        if re.search(pattern, content, re.IGNORECASE):

            return os

    return 'Unknown'

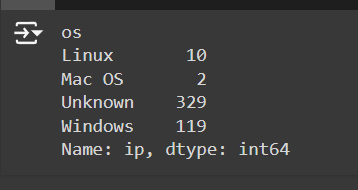
def extract\_os\_from\_user\_agent(user\_agent\_str: str) -> str:

    content = str(extract\_parentheses\_content(user\_agent\_str))

    return identify\_os(content)

df['os'] = df['user\_agent'].apply(extract\_os\_from\_user\_agent)

df.groupby('os')['ip'].nunique()



1. Ранжувати користувачів за країною запиту

# Ранжувати користувачів за країною запиту

import geoip2.database

# reader = geoip2.database.Reader('/content/drive/MyDrive/web-analytics-synchuk/GeoLite2-City.mmdb')

reader = geoip2.database.Reader('/content/drive/MyDrive/web-analytics-synchuk/GeoLite2-Country.mmdb')

# def get\_country\_from\_ip(ip: str) -> str:

#     response = requests.get(f"https://geolocation-db.com/json/{ip}&position=true").json()

#     # print(response)

#     return response['country\_name']

def get\_country\_from\_ip(ip: str, reader) -> str:

    try:

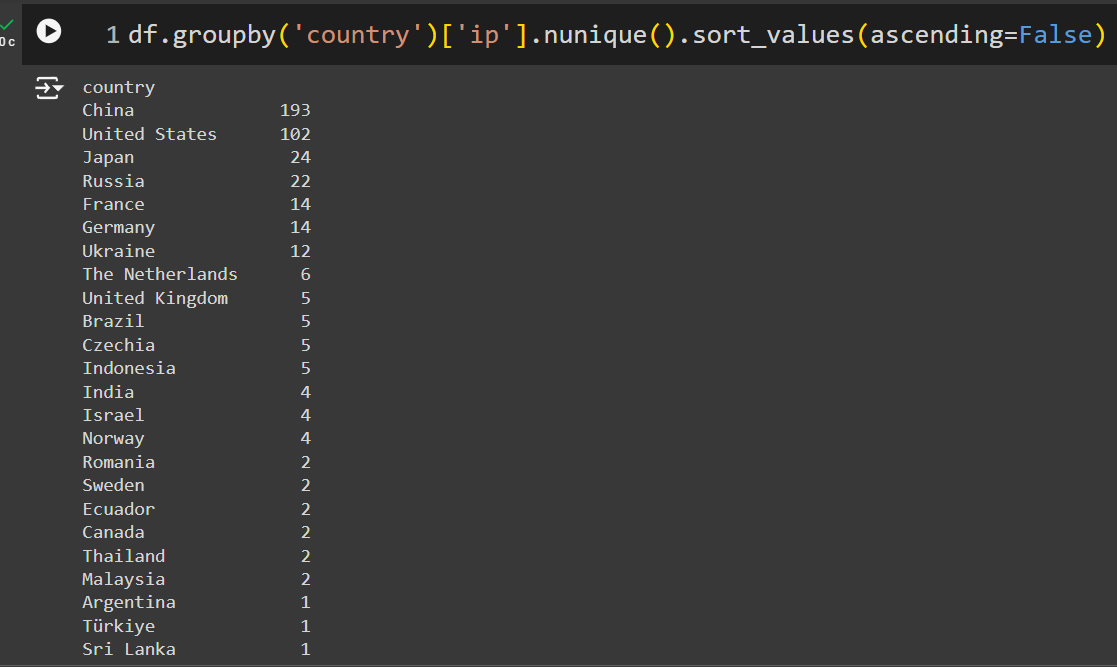
        response = reader.country(ip)

        return response.country.name

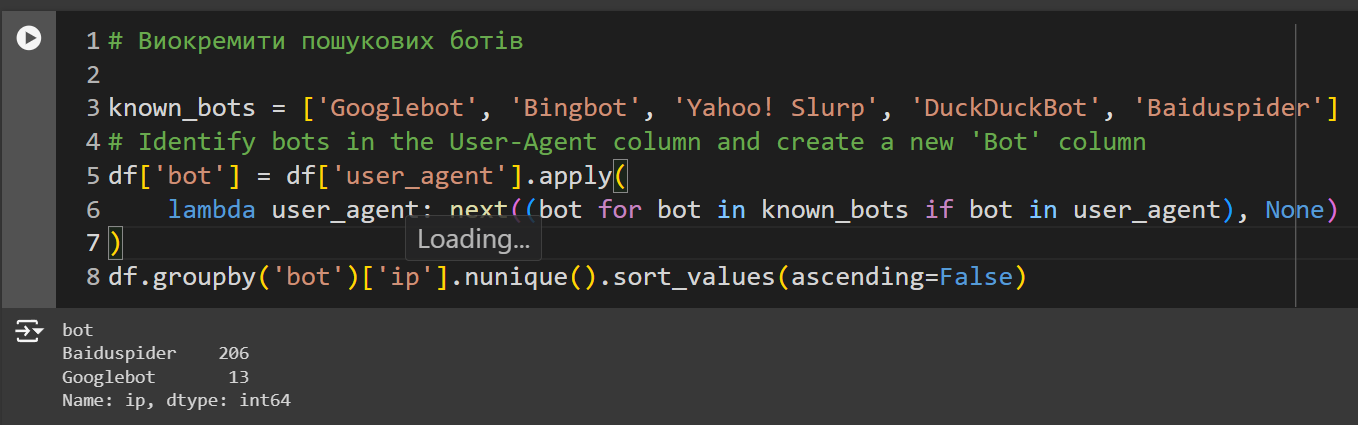
    except Exception:

        return 'Unknown'

df['country'] = df['ip'].apply(lambda ip: get\_country\_from\_ip(ip, reader))



1. Виокремити пошукових ботів



Детектувати аномалії (якщо такі є)

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

X = df[['hour', 'bytes\_sent']]

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

kmeans = KMeans(n\_clusters=4)

df['cluster'] = kmeans.fit\_predict(X\_scaled)

plt.figure(figsize=(10, 6))

for cluster\_label in df['cluster'].unique():

    cluster\_data = df[df['cluster'] == cluster\_label]

    plt.scatter(cluster\_data['hour'], cluster\_data['bytes\_sent'],

    label=f'Cluster {cluster\_label}')

plt.xlabel('Hour of the Day')

plt.ylabel('Bytes Sent')

plt.title('Clusters of Data')

plt.legend()

plt.grid(True)

plt.show()

