Experiment No. 08.

Aim : Interacting with Web APIs.

Problem Statement: Analyzing Weather Data from OpenWeatherMap API, and perform following tasks – 1. Register and obtain API key from OpenWeatherMap. 2. Interact with the OpenWeatherMap API. 3. Extract relevant weather attributes. 4. Clean and preprocess the retrieved data. 5. Perform data modeling to analyze weather patterns. 6. Visualize the weather data. 7. Apply data aggregation techniques. 8. Incorporate geographical information. 9. Explore and visualize relationships between weather attributes.

Dataset: Weather data retrieved from OpenWeatherMap API.

Software Requirements : Operating System, Python, Python Libraries.

Hardware Requirements : Processor, RAM, Storage, and Internet Connection.

Objective : i) Understand the fundamentals of web API's and how to interact with them. ii) Retrive and parse JSON data from the open weather map API. iii) Extract and display key weather metrics. iv) Visualize weather trends using a plotting library.

Theory : What is an API's ?

An API, or Application Programming Interface, is a set of rules and tools that allows different software applications to communicate with each other. Think of it like a menu in a restaurant: it lists the dishes you can order and tells the kitchen how to prepare them. Similarly, an API defines the methods and data formats that applications can use to request and exchange information. APIs are used in many contexts, from web services to operating systems, and they play a crucial role in enabling different software systems to work together seamlessly.

RESTFUL API's

A RESTful API (Representational State Transfer) is a specific type of API that adheres to the principles of REST architecture. REST is a set of constraints and guidelines for creating web services that are scalable, stateless, and easily maintainable. REST is a design pattern for networked applications that promotes scalability, stateless communication, and simplicity. RESTful APIs are popular because they are simple, flexible, and use standard HTTP methods, making them easy to implement and consume.

Key Features of RESTFUL API's

1. **Statelessness**: Each request from a client to a server must contain all the information needed to understand and process the request. The server does not store any state between requests. This makes the API more scalable because each request is independent.
2. **Client-Server Architecture**: The client and server operate independently of each other. The client makes requests to the server, and the server responds with the requested data. This separation allows for scalability and flexibility.
3. **Cacheability**: Responses from the server can be explicitly marked as cacheable or non-cacheable. This helps improve performance by reducing the need for repeated requests to the server.

Open weather Map API's

OpenWeatherMap is a service that provides weather data through APIs. These APIs allow developers to integrate weather information into their applications or websites. OpenWeatherMap offers several types of APIs to access different kinds of weather data.

Endpoints –

Current Weather Data API: https://api.openweathermap.org/data/2.5/weather

Forecast API: https://api.openweathermap.org/data/2.5/forecast

Historical Weather Data API: https://api.openweathermap.org/data/2.5/onecall/timemachine

Response Format

The response format for the OpenWeatherMap API is typically in JSON (JavaScript Object Notation), which is a lightweight data-interchange format that's easy for humans to read and write and easy for machines to parse and generate.

Implementation

Step No. 1 – Regiester and obtain API Key

Sign up on the open weather map website.

Navigate to the API's section and generate an API's key.

Step No. 02 - Interact with the OpenWeatherMap API using the API key to retrieve weather data for a specific location.

```
import requests
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns

api_key = 'c15c676d2f9d0dada63d7fa10c76ce01'
location = 'India'
url =
f'http://api.openweathermap.org/data/2.5/forecast?q=India&appid=c15c676d2f9d0dada63d7f
a10c76ce01&units=metric'

response = requests.get(url)
data = response.json()
if response.status_code == 200:
   "Data retrieved successfully for India"
else:
   data = response.json()
   f"Error: {data.get('message', 'Failed to retrieve data')}"

data
```

Step No. 03 - Extract Relevant Weather Attributes.

```
weather_list = data['list']
weather_data = {'datetime': [], 'teampreature': [], 'humidity': [], 'wind_speed': [],
'precipitation': []}
for entry in weather_list:
   weather_data['datetime'].append(datetime.fromtimestamp(entry['dt']))
   weather_data['teampreature'].append(entry['main']['temp'])
   weather_data['humidity'].append(entry['main']['humidity'])
   weather_data['wind_speed'].append(entry['wind']['speed'])
   precipitation = entry['rain'].get('3h', 0) if 'rain' in entry else 0
   weather_data['precipitation'].append(precipitation)

import pandas as pd
df = pd.DataFrame(weather_data)
df.head()
```

	datetime	teampreature	humidity	wind_speed	precipitation	
0	2024-07-12 14:30:00	15.82	95	1.35	2.38	
1	2024-07-12 17:30:00	18.21	90	1.74	3.68	
2	2024-07-12 20:30:00	20.88	83	2.04	3.51	
3	2024-07-12 23:30:00	18.24	95	1.89	1.91	
4	2024-07-13 02:30:00	14.46	99	1.71	4.05	

Step No. 04 - Clean and Preprocess the Data.

```
df.isnull().sum()
```

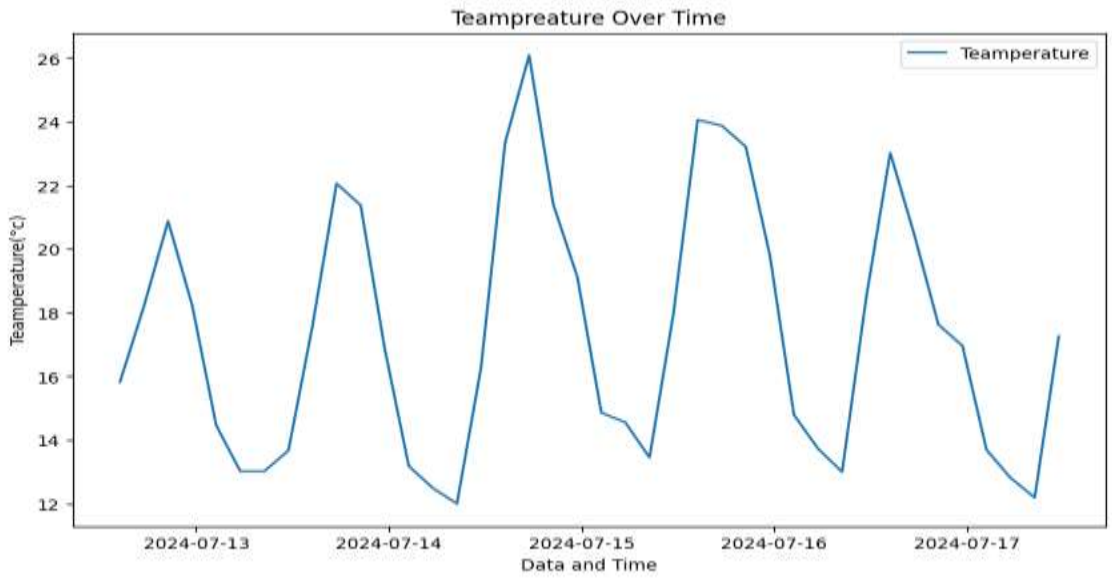
```
datetime        0
teampreature    0
humidity        0
wind_speed      0
precipitation   0
dtype: int64
```

Step No. 05 - Perform Data Modelling.

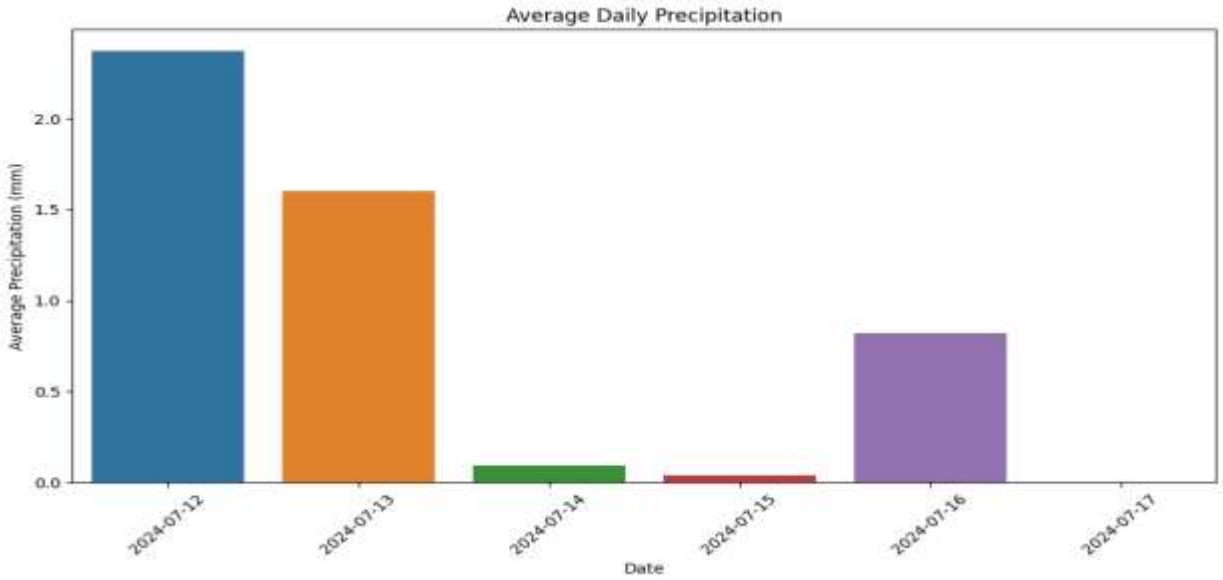
```
avg_temp = df['teampreature'].mean()
avg_temp
17.82775
max_temp = df['teampreature'].max()
min_temp = df['teampreature'].min()
max_temp
25.93
min_temp
11.81
```

Step no. 06 - Visualize the Weather Data.

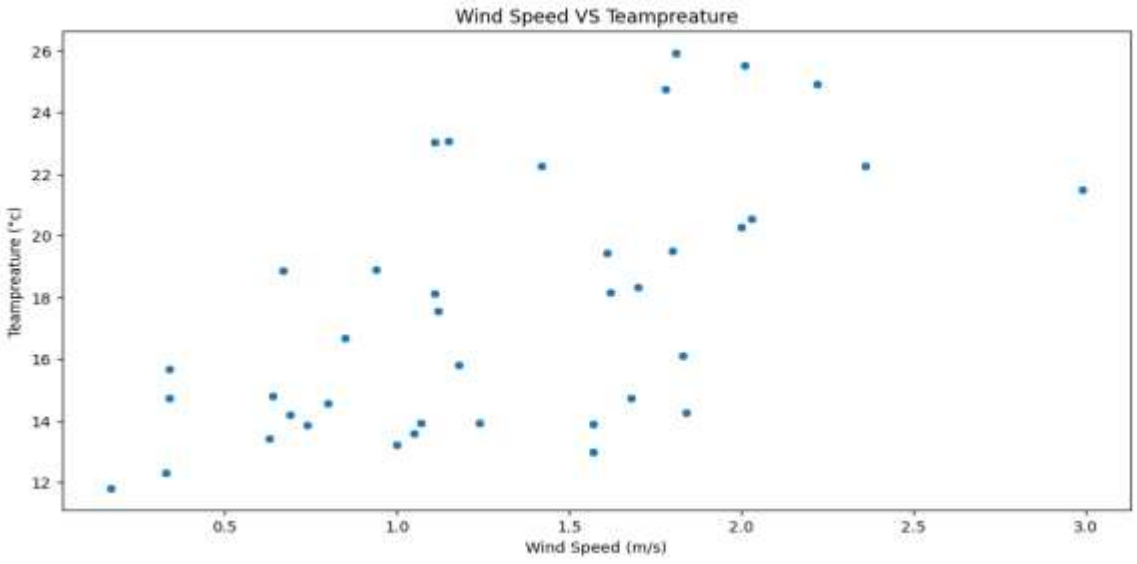
```
# 1. Line Chart
plt.figure(figsize=(10, 6))
plt.plot(df['datetime'], df['teampreature'], label='Teamperature')
plt.xlabel('Data and Time')
plt.ylabel('Teamperature(°c)')
plt.title('Teampreature Over Time')
plt.legend()
plt.show()
```



```
# 2. Bar Plot
plt.figure(figsize=(12, 6))
sns.barplot(data=daily_precipitation, x='date', y='precipitation')
plt.xlabel('Date')
plt.ylabel('Average Precipitation (mm)')
plt.title('Average Daily Precipitation')
plt.xticks(rotation=45)
plt.show()
```




```
# 3. Scatter Plot.
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='wind_speed', y='teampreature')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Teampreature (°c)')
plt.title('Wind Speed VS Teampreature')
plt.show()
```



Step No. 07 - Apply Data Aggregation Techniques.

```
# 1. Daily Aggregation
daily_weather = df.resample('D').agg({'teampreature': 'mean', 'humidity': 'mean', 'wind_speed': 'max'})
daily_weather.head()
```

	teampreature	humidity	wind_speed
datetime			
2024-07-12	18.170000	90.600	2.03
2024-07-13	16.735000	83.000	2.99
2024-07-14	18.307500	70.125	2.22
2024-07-15	19.698750	68.875	2.01
2024-07-16	17.780000	81.250	1.62

```
# 2. Monthly Aggregation.
monthly_weather = df.resample('M').agg({'teampreature': 'mean', 'humidity': 'mean', 'wind_speed': 'max'})
monthly_weather.head()
```

	teampreature	humidity	wind_speed
datetime			
2024-07-31	17.827750	78.725	2.99

```
# 3. Seasnoal Aggregation.
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Autumn'

df['season'] = df.index.month.map(get_season)

seasonal_weather = df.groupby('season').agg({'temperature': 'mean', 'humidity': 'mean', 'wind_speed': 'max'})
seasonal_weather
```

	temperature	humidity	wind_speed
season			
Summer	26.0	84.2	9

Step No. 08 - Incorporate Geographical Information.

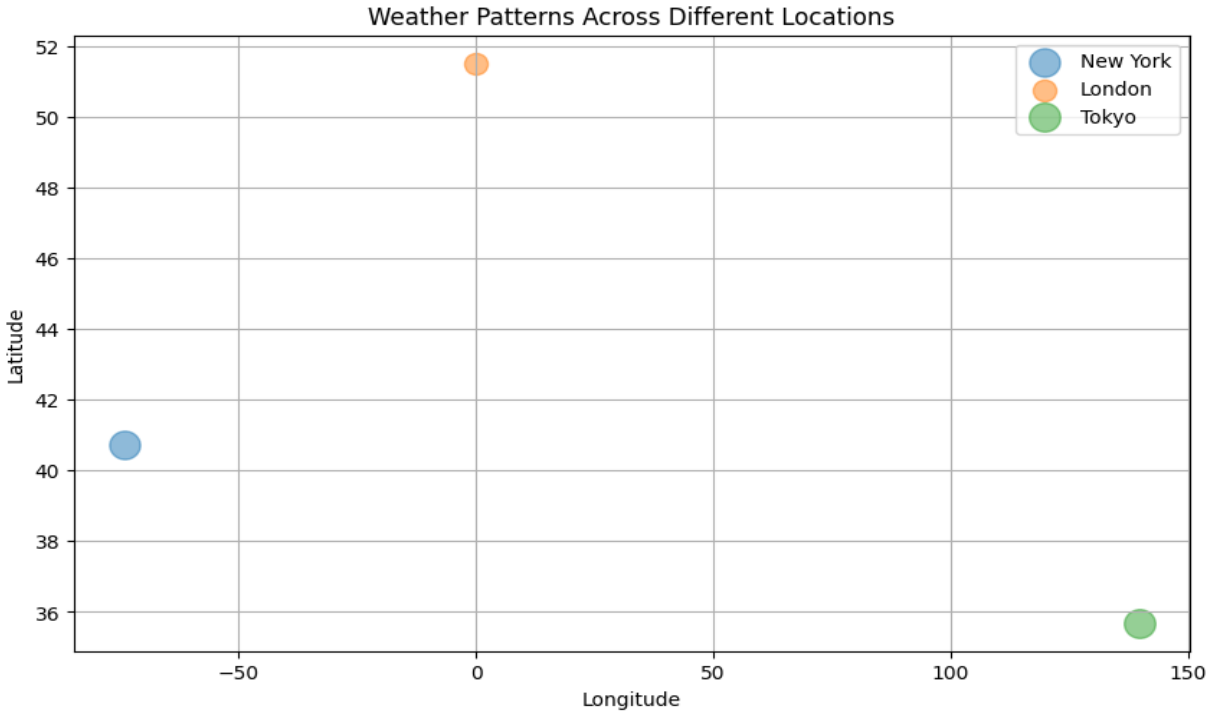
```
pip install folium
```

```
# 1. Fetch Weather Data for Multiple Locations.
# Replace 'your_api_key' with your actual API key
api_key = 'c15c676d2f9d0dada63d7fa10c76ce01'
locations = [
    {'name': 'New York', 'lat': 40.7128, 'lon': -74.0060},
    {'name': 'London', 'lat': 51.5074, 'lon': -0.1278},
    {'name': 'Tokyo', 'lat': 35.6895, 'lon': 139.6917},
    # Add more locations as needed
]
weather_data = []
for loc in locations:
    url =
f'http://api.openweathermap.org/data/2.5/weather?lat={loc["lat"]}&lon={loc["lon"]}&appid=
{api_key}&units=metric'
    response = requests.get(url)
    data = response.json()
    if response.status_code == 200:
        weather_data.append({
            'name': loc['name'],
            'temperature': data['main']['temp'],
            'humidity': data['main']['humidity'],
            'wind_speed': data['wind']['speed'],
            'latitude': loc['lat'],
            'longitude': loc['lon']
        })
    else:
        print(f"Error fetching data for {loc['name']}: {data.get('message', 'Unknown error')}")

weather_data
```

```
[{'name': 'New York',
  'temperature': 23.3,
  'humidity': 73,
  'wind_speed': 1.54,
  'latitude': 40.7128,
  'longitude': -74.006},
 {'name': 'London',
  'temperature': 13.5,
  'humidity': 87,
  'wind_speed': 4.12,
  'latitude': 51.5074,
  'longitude': -0.1278},
 {'name': 'Tokyo',
  'temperature': 24.13,
  'humidity': 91,
  'wind_speed': 1.03,
  'latitude': 35.6895,
  'longitude': 139.6917}]
```

2. Create a Geospatial visualization using Folium.

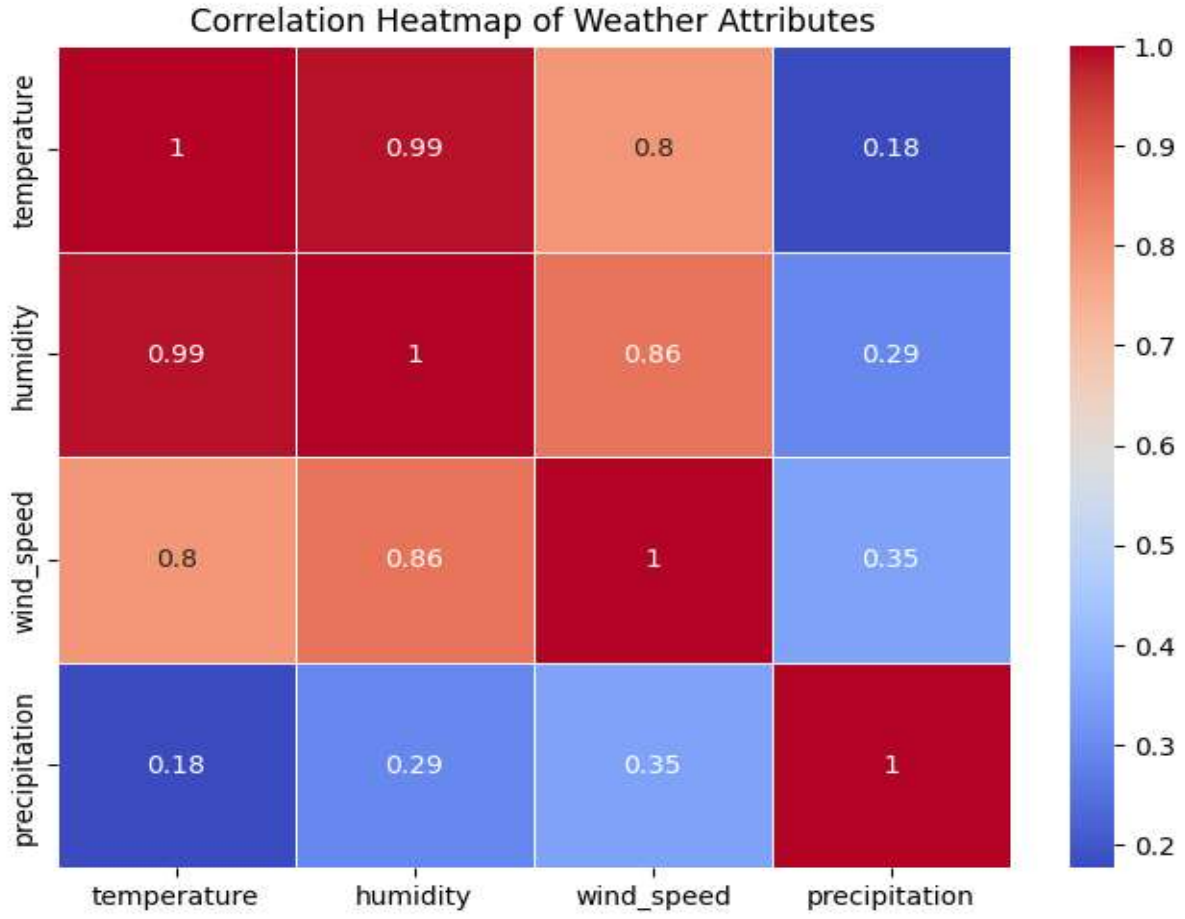
```
# Create a map centered at a specific location (e.g., New York)
map_center = [40.7128, -74.0060]
mymap = folium.Map(location=map_center, zoom_start=3)

# Add markers for each location with weather information
for data in weather_data:
    popup_text = f"<b>{data['name']}</b><br>Temperature:
{data['temperature']}°C<br>Humidity: {data['humidity']}%<br>Wind Speed:
{data['wind_speed']} m/s"
    folium.Marker(location=[data['latitude'], data['longitude']],
popup=popup_text).add_to(mymap)

# Save the map as an HTML file
mymap.save('weather_map.html')
mymap

# 3. Viualize Weather Patterns on a Static Map using Matplotlib.
# Plot each location with a scatter plot based on temperature
plt.figure(figsize=(10, 6))
for data in weather_data:
    plt.scatter(data['longitude'],data['latitude'],s=data['temperature']*10,alpha=0.5,
label=data['name'])
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Weather Patterns Across Different Locations')
plt.legend()
plt.grid(True)
plt.show()
```



Step No. 09 - Explore and Visualize Realationships.

```
# Calculate correlation matrix
correlation_matrix = df[['temperature', 'humidity', 'wind_speed', 'precipitation']].corr()
print(correlation_matrix)
# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Weather Attributes')
plt.show()
```

```
               temperature  humidity  wind_speed  precipitation
temperature       1.000000  0.988483    0.800000       0.176777
humidity          0.988483  1.000000    0.864923       0.294875
wind_speed        0.800000  0.864923    1.000000       0.353553
precipitation     0.176777  0.294875    0.353553       1.000000
```



Conclusion – Thus we can successfully interacting and analyzing weather data from open weather map API by using this following steps, and also the open weather map API provides valuable real time weather data, this data we can easily used for various analytical purposes.