Collecting weather data from an API

Name: Cuadra, Audrick Zander G.

Section: CPE22S3

Date: March 27, 2024

Submitted to: Engr. Roman Richard

About the data

In this notebook, we will be collecting daily weather data from the National Centers for Enviromental Information (NCEI) API. We will use the Global Historical Climatology Network - Daily (GHCND) data set; see the documentation here..

Note: The NCEI is part of the National Oceanic and Atmospheric Administration (NOAA) and, as you can see from the URL for the API, this resource was created when the NCEI was called the NCDC. Should the URL for this change in the future, you can search for the NCEI weather API to find the updated one.

Using the NCEI API

Paste your token below.

```python
import requests

def make_request(endpoint, payload=None):
    """
    Make a request to a specific endpoint in the API
    passing headers and optional payload.

    Parameters:
      - endpoint: The endpoint of the API you want to
                  make GET request to.
      - payload: A dictionary of data to pass along
                  with the request.

      Returns:
          Response object.
    """
    return requests.get(
        f'https://www.ncdc.noaa.gov/cdo-web/api/v2/{endpoint}',
        headers={
            'token': 'YNAdteQhHJsoetjhLCpXzrRhrGdGHvuc'
        },
        params=payload
    )
```

Collecting All Data Points for 2018 In NYC (Various Stations

We can make a loop to query for all the data points one day at a time. Here we create a list of all the results:

```
import datetime

from IPython import display # for updating the cell dynamically

current = datetime.date(2018, 1, 1)
end = datetime.date(2019, 1, 1)

results = []

while current < end:
    # update the cell with status information
    display.clear_output(wait=True)
    display.display(f'Gathering data for {str(current)}')

    response = make_request(
        'data',
        {
            'datasetid' : 'GHCND', # Global Historical Climatology Network - Daily (GH
            'locationid' : 'CITY:US360019', # NYC
            'startdate' : current,
            'enddate' : current,
            'units' : 'metric',
            'limit' : 1000 # max allowed
        }
    )

    if response.ok:
        # we extend the list instead of appending to avoid getting a nested list
        results.extend(response.json()['results'])

    # update the current date to avoid an infinite loop
    current += datetime.timedelta(days=1)
```

```
'Gathering data for 2018-12-31'
```

Now, we can create a dataframe with all this data. Notice there are multiples stations with values for each datatype on a given day. We don't know what the stations are, but we can look them up and add them to the data:

```
import pandas as p

df = p.DataFrame(results)
df.head()
```

	date	datatype	station	attributes	value
1	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0015	,,N,1050	0.0

Save this data to a file:

```
df.to_csv('/content/nyc_weather_2018.csv', index=False)
```

and write it to the database:

```
import sqlite3

with sqlite3.connect('/content/weather.db') as connection:
    df.to_sql(
        'weather', connection, index=False, if_exists='replace'
    )
```

For learning about merging dataframes, we will also get the data mapping station IDs to information about the station:

```
response = make_requests(
    'stations',
    {
        'datasetid' : 'GHCND',
        'locationid' : 'CITY:US360019',
        'limit' : 1000
    }
)

stations = p.DataFrame(response.json()['results'])[['id', 'name', 'latitude', 'longitud
stations.to_csv('/content/weather_stations.csv', index=False)

with sqlite3.connect('/content/weather.db') as connection:
    stations.to_sql(
        'stations', connection, index=False, if_exists='replace'
    )
```

Comments and Insights:

This module topic demonstrates collecting data with the use of API. The module includes a function for making API requests, collecting data based on specific conditions like date ranges and the creation of dataframes and csv files from the requested data in the API.