

## ✓ Collecting weather data from an API

In this notebook, we will be collecting daily weather data from the National Centers for Environmental 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 resource change in the future, you can search for the NCEI weather API to find the updated one.

### Using the NCEI API

```
import requests

def make_request(endpoint, payload=None):
    # endpoint (str): the endpoint of the weather API
    # payload (dict, optional): request
    return requests.get(
        f'https://www.ncdc.noaa.gov/cdo-web/api/v2/{endpoint}',
        headers={
            'token': 'v0l0lv0piQQRqPuuXdSPJDkvHgawgwUd'
        },
        params=payload

    # response (requests.Response): the response object from the API request
)
#The function sends a GET request to a weather API endpoint with optional headers and payload data, returning the resp
```

### Collect 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

# Define start and end dates for data collection
current = datetime.date(2018, 1, 1)
end = datetime.date(2019, 1, 1)
results = []

# Loop through dates until reaching the end date
while current < end:
    # Update the cell with status information
    display.clear_output(wait=True)
    display.display(f'Gathering data for {str(current)}')

    # Make a request to fetch data for the current date
    response = make_request(
        'data',
        {
            'datasetid': 'GHCND', # Global Historical Climatology Network - Daily (GHCND) dataset
            'locationid': 'CITY:US360019', # NYC
            'startdate': current,
            'enddate': current,
            'units': 'metric',
            'limit': 1000 # Max allowed
        }
    )

    # Check if the response is successful
    if response.ok:
        # Extend the results list with the data for the current date
        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 multiple 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 pd
df = pd.DataFrame(results) # Create a DataFrame from the results data
df.head() # Display the first few rows of the DataFrame

```

	date	datatype	station	attributes	value	
0	2018-01-01T00:00:00	PRCP	GHCND:US1CTFR0039	„N,0800	0.0	
1	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0015	„N,1050	0.0	
2	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0015	„N,1050	0.0	
3	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0017	„N,0920	0.0	
4	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0017	„N,0920	0.0	

Next steps: [View recommended plots](#)

Save this data to a file:

```

# Exports the DataFrame to a CSV file 'nyc_weather_2018.csv'.
# 'index=False' parameter ensures that the DataFrame index is not written to the CSV file.
df.to_csv('/content/nyc_weather_2018.csv', index=False)

```

and write it to the database:

```
import sqlite3 # Import the sqlite3 module

# Establish a connection to the SQLite database file 'weather.db'
with sqlite3.connect('/content/weather.db') as connection:
    df.to_sql('weather', connection, index=False, if_exists='replace')
    # index=False ensures that the DataFrame index is not written as a column in the table
    # if_exists='replace' ensures that if a table with the same name already exists, it will be replaced
```

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

```
import pandas as pd
import sqlite3

# Make request to retrieve weather station data
response = make_request(
    'stations',
    {
        'datasetid': 'GHCND', # Global Historical Climatology Network - Daily (GHCND) dataset
        'locationid': 'CITY:US360019', # NYC
        'limit': 1000 # Maximum number of stations allowed
    }
)

# Extract relevant information from the response and create a DataFrame
stations = pd.DataFrame(response.json()['results'])[['id', 'name', 'latitude', 'longitude', 'elevation']]

# Save station information to a CSV file
stations.to_csv('/content/weather_stations.csv', index=False)
```