Collecting weather data from an API

About the data

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

Paste your token below.

```
import requests

def make_req(endpoint, payload=None):
  return requests.get(
      f'https://www.ncdc.noaa.gov/cdo-web/api/v2/{endpoint}',
      headers={
          'token' : 'COGqDwKIRDvdlxMHVhxgqTsoJqtCoAfz'
      },params=payload
  )
```

```
response = make_req('datasets', {'startdate':'2018-10-01'})
response.status_code
```

```
200
```

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 as dis

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

results = []

while current < end :
  dis.clear_output(wait=True)
  dis.display(f'gathering data for {str(current)}')

  response = make_req(
      'data',{
          'datasetid':'GHCND',
          'locationid':'CITY:US360019',
          'startdate':current,
          'enddate':current,
          'units':'metric',
          'limit':100
      }
  )
  if response.ok:
    results.extend(response.json()['results'])

  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 p

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

	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
...	...	...	...	...	...
36095	2018-12-31T00:00:00	SNOW	GHCND:US1NJUN0003	,,N,0900	0.0
36096	2018-12-31T00:00:00	PRCP	GHCND:US1NJUN0010	,,N,0730	0.0
36097	2018-12-31T00:00:00	SNOW	GHCND:US1NJUN0010	,,N,0730	0.0
36098	2018-12-31T00:00:00	PRCP	GHCND:US1NJUN0014	,,N,0800	0.3
36099	2018-12-31T00:00:00	PRCP	GHCND:US1NJUN0017	T,,N,0400	0.0

36100 rows × 5 columns

Next steps: View recommended plots

Save this data to a file:

```
df.to_csv('/content/sample_data/nycweather2k18.csv', index=False)
```

and write it to the database:

```
import sqlite3 as sq3

with sq3.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_req(
    'stations',{
        'datasetid':'GHCND',
        'locationid':'CITY:US360019',
        'limit':10
    }
)
stations = p.DataFrame(response.json()['results'])[['id','name','latitude','longitude','elevation']]
stations.to_csv('/content/sample_data/weather_stations.csv', index=False)

with sq3.connect('/content/weather.db') as connection:
  stations.to_sql('stations',connection,index=False, if_exists='replace')
```