EM-DAT GRAPHQL API (Python version)

Code

We illustrate how to use the EM-DAT GraphQL API in Python. We will use the GQL Python client (https://github.com/graphql-python/gql) as it provides a complete toolset to execute GraphQL requests, a good documentation website (https://gql.readthedocs.io/en/stable/intro.html) as well as a command line tool (gql-cli).

For alternative clients, see GRAPHQL Python Clients (https://graphql.org/code/#python-client).

Hide

```
# Preamble: define your API key and GraphQL query string
api_key = '...'
query_str = """
    query floods {
      api_version
      public_emdat(
        cursor: {limit: 1, offset: 1}
        filters: {
          from: 1990,
          to: 2019,
          classif: ["nat-hyd-flo"],
          include_hist: true
          }
      ) {
        total_available
        info {
          timestamp
          filters
          cursor
        }
        data {
          disno
          classif_key
          type
          subtype
          river_basin
          total_dam
          total_dam_adj
          total_deaths
          total_affected
          entry_date
          last_update
        }
      }
    }
"""
```

Using the `gql` module

Hide

```
from pprint import pprint # optional
from gql import Client, gql
from gql.transport.requests import RequestsHTTPTransport

# Parsing the query
query = gql(query_str)

# Create a transport with the necessary headers
transport = RequestsHTTPTransport(
    url='https://api.emdat.be/v1',
    headers={'Authorization': api_key},
    use_json=True,
)

# Create a GraphQL client using the defined transport
client = Client(transport=transport, fetch_schema_from_transport=True)

# Execute the query
response = client.execute(query)
pprint(response)
```

```
{'api_version': 'v1',
 'public_emdat': {'data': [{'classif_key': 'nat-hyd-flo-riv',
                            'disno': '1990-0002-TUN',
                            'entry_date': '2006-07-19',
                            'last_update': '2023-09-25',
                            'river_basin': None,
                            'subtype': 'Riverine flood',
                            'total_affected': 152000,
                            'total_dam': 242800,
                            'total_dam_adj': 543835,
                            'total_deaths': 37,
                            'type': 'Flood'}],
                  'info': {'cursor': {'limit': 1, 'offset': 1},
                           'filters': {'classif': ['nat-hyd-flo-*'],
                                       'from': 1990,
                                       'include_hist': True,
                                       'to': 2019},
                           'timestamp': '2023-11-15T15:14:54Z'},
                  'total_available': 4112}}
```

Parse Data into a `pandas` DataFrame

The content of `data` is the same as available in the public table, it is therefore straightforward to translate a response to a table format.

Hide

```
import pandas as pd
df = pd.DataFrame(response['public_emdat']['data'])
df
```

```
           disno      classif_key  ...  entry_date last_update
0  1990-0002-TUN  nat-hyd-flo-riv  ...  2006-07-19  2023-09-25

[1 rows x 11 columns]
```

Using the `requests` module

It is also possible to pass a query through a HTTP GET request, passing the GraphQL query in the request body.

Hide

```
import requests

# Prepare the request URL
url = f'https://api.emdat.be/v1'

# Headers with Authorization
headers = {'Authorization': api_key}

# Make the GET request, using the raw GraphQL query string (see above)
response = requests.get(url, json={"query": query_str}, headers=headers)
pprint(response.json())
```

```
{'data': {'api_version': 'v1',
          'public_emdat': {'data': [{'classif_key': 'nat-hyd-flo-riv',
                                     'disno': '1990-0002-TUN',
                                     'entry_date': '2006-07-19',
                                     'last_update': '2023-09-25',
                                     'river_basin': None,
                                     'subtype': 'Riverine flood',
                                     'total_affected': 152000,
                                     'total_dam': 242800,
                                     'total_dam_adj': 543835,
                                     'total_deaths': 37,
                                     'type': 'Flood'}],
                           'info': {'cursor': {'limit': 1, 'offset': 1},
                                    'filters': {'classif': ['nat-hyd-flo-*'],
                                                'from': 1990,
                                                'include_hist': True,
                                                'to': 2019},
                                    'timestamp': '2023-11-15T15:14:54Z'},
                           'total_available': 4112}}}
```