Used Big Query to build my SQL skills, Big Query is a web service that lets you apply SQL to huge data sets.

The purpose of this document is to showcase the basics of accessing and examining BigQuery data sets.

1. Installed google cloud bigquery

```
In [2]: pip install google-cloud-bigquery

Collecting google-cloud-bigquery
  Downloading google_cloud_bigquery-3.11.4-py2.py3-none-any.whl (219 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 219.6/219.6 kB 1.2 MB/s eta 0:00:0000:0100:01
Collecting google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5
  Downloading google_api_core-2.11.1-py3-none-any.whl (120 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 120.5/120.5 kB 1.8 MB/s eta 0:00:00a 0:00:01
Collecting google-resumable-media<3.0dev,>=0.6.0
  Downloading google_resumable_media-2.5.0-py2.py3-none-any.whl (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.7/77.7 kB 4.0 MB/s eta 0:00:00
Collecting proto-plus<2.0.0dev,>=1.15.0
  Downloading proto_plus-1.22.3-py3-none-any.whl (48 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.1/48.1 kB 870.5 kB/s eta 0:00:00 0:00:01
Requirement already satisfied: requests<3.0.0dev,>=2.21.0 in ./anaconda3/lib/python3.10/site-packages (from google-
cloud-bigquery) (2.28.1)
Requirement already satisfied: python-dateutil<3.0dev,>=2.7.2 in ./anaconda3/lib/python3.10/site-packages (from goo
gle-cloud-bigquery) (2.8.2)
Collecting google-cloud-core<3.0.0dev,>=1.6.0
  Downloading google_cloud_core-2.3.3-py2.py3-none-any.whl (29 kB)
Requirement already satisfied: packaging>=20.0.0 in ./anaconda3/lib/python3.10/site-packages (from google-cloud-big
query) (22.0)
Collecting grpcio<2.0dev,>=1.47.0
  Downloading grpcio-1.57.0.tar.gz (24.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.7/24.7 MB 5.8 MB/s eta 0:00:0000:0100:01
  Preparing metadata (setup.py) ... done
Collecting protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.19.5
  Downloading protobuf-4.24.1-cp37-abi3-macosx_10_9_universal2.whl (409 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 409.3/409.3 kB 1.9 MB/s eta 0:00:0000:0100:01
Collecting googleapis-common-protos<2.0.dev0,>=1.56.2
  Downloading googleapis_common_protos-1.60.0-py2.py3-none-any.whl (227 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 227.6/227.6 kB 3.3 MB/s eta 0:00:00a 0:00:01
Collecting google-auth<3.0.dev0,>=2.14.1
  Downloading google_auth-2.22.0-py2.py3-none-any.whl (181 kB)
```

2. To import big query use the Python package below
3. Then create the Client object

```
In [3]: import os

        # Set the environment variable
        os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/Users/admin/Downloads/sqlbigquery-396702-f6aa77f240d3.json'

        # Now you can use Google Cloud libraries
        from google.cloud import bigquery

        # Rest of your code

In [4]: from google.cloud import bigquery

In [5]: # Create a "Client" object
        client = bigquery.Client()
```

4. Constructed a reference to the dataset with the dataset() method
5. Used the get_dataset() method along with the reference we just constructed to fetch the dataset

```
In [6]: # Construct a reference to the "hacker_news" dataset
        dataset_ref = client.dataset("hacker_news", project="bigquery-public-data")

        # API request - fetch the dataset
        dataset = client.get_dataset(dataset_ref)
```

6. Used the list_tables() method to list tables

```
In [7]: # List all the tables in the "hacker_news" dataset
        tables = list(client.list_tables(dataset))

        # Print names of all tables in the dataset (there are four!)
        for table in tables:
            print(table.table_id)

        full
```

7. Fetched the full table in the hacker_news dataset

```
In [8]: # Construct a reference to the "full" table
        table_ref = dataset_ref.table("full")

        # API request - fetch the table
        table = client.get_table(table_ref)
```

8. Understood structure of the table schema

```
In [9]: # Print information on all the columns in the "full" table in the "hacker_news" dataset
        table.schema

Out[9]: [SchemaField('title', 'STRING', 'NULLABLE', None, 'Story title', (), None),
         SchemaField('url', 'STRING', 'NULLABLE', None, 'Story url', (), None),
         SchemaField('text', 'STRING', 'NULLABLE', None, 'Story or comment text', (), None),
         SchemaField('dead', 'BOOLEAN', 'NULLABLE', None, 'Is dead?', (), None),
         SchemaField('by', 'STRING', 'NULLABLE', None, "The username of the item's author.", (), None),
         SchemaField('score', 'INTEGER', 'NULLABLE', None, 'Story score', (), None),
         SchemaField('time', 'INTEGER', 'NULLABLE', None, 'Unix time', (), None),
         SchemaField('timestamp', 'TIMESTAMP', 'NULLABLE', None, 'Timestamp for the unix time', (), None),
         SchemaField('type', 'STRING', 'NULLABLE', None, 'Type of details (comment, comment_ranking, poll, story, job, poll
        opt)', (), None),
         SchemaField('id', 'INTEGER', 'NULLABLE', None, "The item's unique id.", (), None),
         SchemaField('parent', 'INTEGER', 'NULLABLE', None, 'Parent comment ID', (), None),
         SchemaField('descendants', 'INTEGER', 'NULLABLE', None, 'Number of story or poll descendants', (), None),
         SchemaField('ranking', 'INTEGER', 'NULLABLE', None, 'Comment ranking', (), None),
         SchemaField('deleted', 'BOOLEAN', 'NULLABLE', None, 'Is deleted?', (), None)]
```

9. Previewed the first five lines of the table

```
In [9]: # Preview the first five lines of the "full" table
        client.list_rows(table, max_results=5).to_dataframe()
```

Out[9]:

	title	url	text	dead	by	score	time	timestamp	type	id	parent	descendants	ranking	deleted
0	None	None	I would rather just have wired earbuds, period...	<NA>	zeveb	<NA>	1591717736	2020-06-09 15:48:56+00:00	comment	23467666	23456782	<NA>	<NA>	<NA>
1	None	None	DNS?	<NA>	nly	<NA>	1572810465	2019-11-03 19:47:45+00:00	comment	21436112	21435130	<NA>	<NA>	<NA>
2	None	None	These benchmarks seem pretty good. Filterable...	<NA>	mrkeen	<NA>	1591717727	2020-06-09 15:48:47+00:00	comment	23467665	23467426	<NA>	<NA>	<NA>
3	None	None	Oh really?<p>* Excel alone uses 86.1MB of priv...	<NA>	oceanswave	<NA>	1462987532	2016-05-11 17:25:32+00:00	comment	11677248	11676886	<NA>	<NA>	<NA>
4	None	None	These systems are useless. Of the many flaws:...	<NA>	nyxxie	<NA>	1572810473	2019-11-03 19:47:53+00:00	comment	21436113	21435025	<NA>	<NA>	<NA>

10. Previewed the first five lines in the by column of the full table

In [10]:
```
# Preview the first five entries in the "by" column of the "full" table
client.list_rows(table, selected_fields=table.schema[:1], max_results=5).to_dataframe()
```

Out[10]:

	title
0	None
1	None
2	None
3	None
4	None