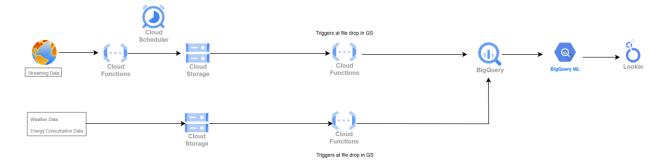Pipeline:

1) Static monthly weather archival data loaded into cloud storage

capstone824

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	Soft Delete

OBJECTS   CONFIGURATION   PERMISSIONS   PROTECTION   LIFECYCLE   OBSERVABILITY   INVENTORY REPORTS   OPERATIONS

Folder browser ⏐<

📦 capstone824 ⋮

Buckets > capstone824 📋

UPLOAD FILES   UPLOAD FOLDER   CREATE FOLDER   TRANSFER DATA ▾   MANAGE HOLDS   EDIT RETENTION

DOWNLOAD   DELETE

Filter by name prefix only ▾   ≡ Filter   Filter objects and folders   Show Live objects only ▾

	Name	Size	Type	Created ❓	Storage class	Last m
☐	📄 Static_MonthlyWeather_TrentonF...	24.3 KB	text/csv	Aug 12, 2024, 8:15:49 PM	Standard	Aug 1

2) Cloud function for a one time load of data from cloud storage to Big Query Table:

← Function details ✏️ EDIT 🗑 DELETE 📋 COPY 🎓 LEARN C I<

✅ **archiveddata** `2nd gen` (Deployed at Aug 12, 2024, 7:00:23 PM)

URL: https://us-central1-probable-hash-285404.cloudfunctions.net/archiveddata 📋 ❓

▶▶ Powered by Cloud Run ❓
archiveddata

METRICS DETAILS **SOURCE** VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Runtime : Python 3.12   Entry point : archiveddata   **EDIT**   ⬇ DOWNLOAD ZIP

📄 main.py

📄 requirements.txt

```python
1  from google.cloud import bigquery
2  from google.cloud import storage
3  import pandas as pd
4  from io import import StringIO
5
6  # Replace with your actual project ID
7  PROJECT_ID = 'probable-hash-285404'
8  DATASET_ID = 'Capstone824'
9  TABLE_ID = 'static_weather_trentonfalls_ny'
10
11 def archiveddata(event, context):
12     """Triggered by a change to a Cloud Storage bucket.
13     Args:
14         event (dict): The Cloud Functions event payload.
15         context (google.cloud.functions.Context): Metadata of triggeri
16     """
17
18     # Get the file that triggered the function
19     file_name = event['name']
20     bucket_name = event['bucket']
21
22     # Instantiate clients
23     storage_client = storage.Client()
24     bucket = storage_client.bucket(bucket_name)
25     blob = bucket.blob(file_name)
26     blob_string = blob.download_as_text()
27
28     # Set chunk size (number of rows per chunk)
29     chunk_size = 10000
30
31     # Define the schema once for reuse
32     schema = [
```

3) Data loaded into Big Query table

Explorer + ADD I<

🔍 Search BigQuery resources ❓

Viewing resources.
SHOW STARRED ONLY

▶ 🔶 Data canvases ⋮
▶ ➡ External connections ⋮
▼ 🔷 Capstone824 ☆ ⋮
    ⋮⋮⋮ energyconsumption_... ☆ ⋮
    ▦ static_energyconsu... ☆ ⋮
    ▦ static_energyconsu... ☆ ⋮
    ⋮⋮⋮ static_joined ☆ ⋮
    ▦ static_weather_trent... ☆ ⋮
    ▦ static_weather_trent... ☆ ⋮
    ▦ static_weather_trent... ☆ ⋮
▶ ▦ bts ☆ ⋮
▶ 🔷 opensky ☆ ⋮
▶ ▦ static_monthlyweather_t... ☆ ⋮

🏠 ▾ ✕ 🔍 Untitled query ▾ ✕ ▦ static_w... _ny ▾ ✕ 🔍 Untitled query ▾ ✕ ➕ ▾ ≣ 🏠 ⓘ

▦ **static_weathe...** 🔍 QUERY ▾ 👥 SHARE 📋 COPY ➕ SNAPSHOT 🗑 DELETE ⬆ EXPORT ▾

‹ SCHEMA DETAILS **PREVIEW** TABLE EXPLORER `PREVIEW` INSIGHTS `PREVIEW` LINEAGE DA

Row	STATION	NAME	LATITUDE	LONGITUDE	ELEVATION	DATE
1	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-01
2	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-02
3	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-03
4	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-04
5	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-05
6	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-06
7	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-07
8	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-08
9	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-09
10	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-10
11	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-11
12	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2016-12
13	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2017-01
14	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2017-02
15	USC00308578	TRENTON FALLS, NY US	43.2761	-75.1566	243.8	2017-03

4) Log extract of cloud function trigger from cloud storage file upload
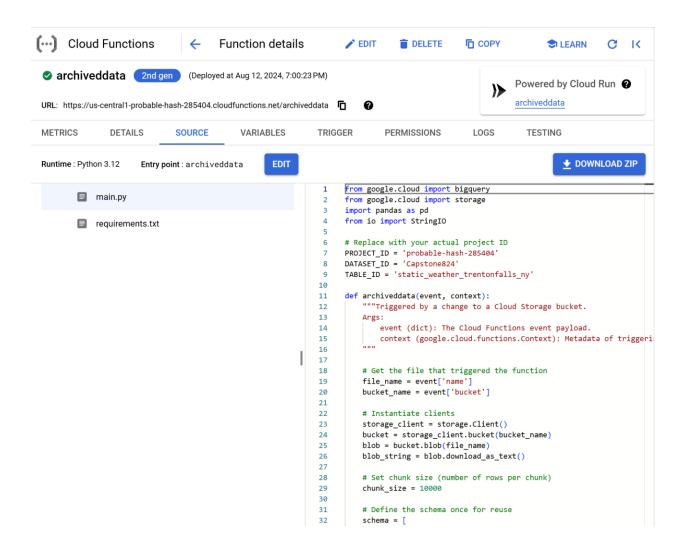

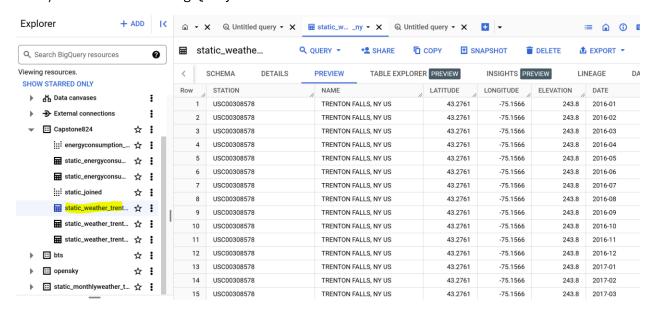1) Energy archival data loaded into cloud storage

2) Cloud function for a one time load of data from cloud storage to Big Query Table:

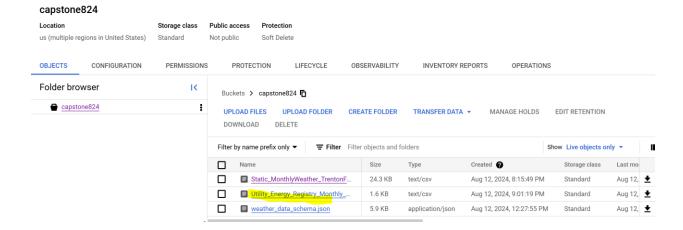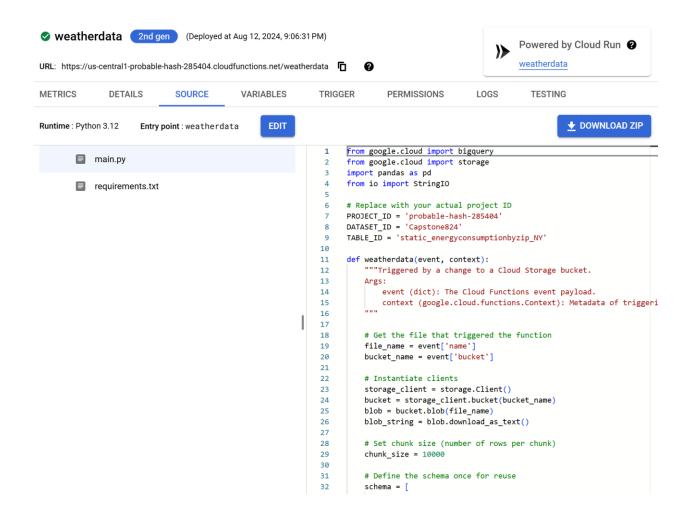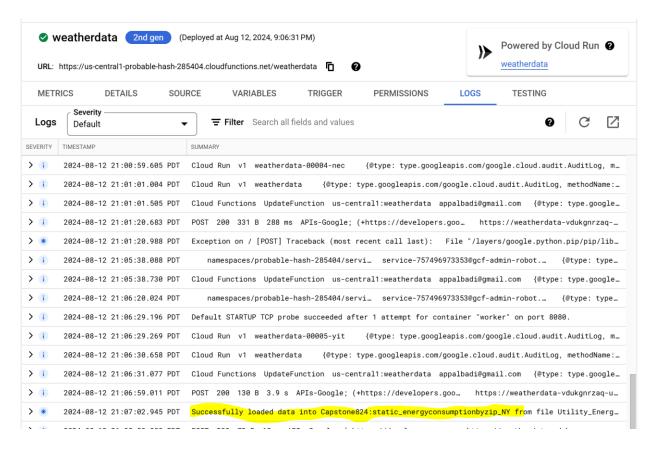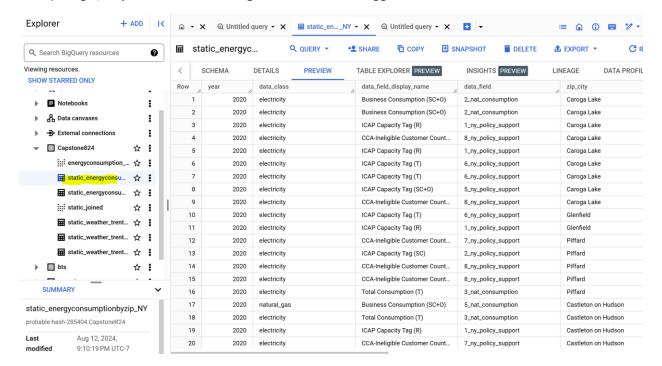URL: https://us-central1-probable-hash-285404.cloudfunctions.net/weatherdata

Powered by Cloud Run
weatherdata

| METRICS | DETAILS | SOURCE | VARIABLES | TRIGGER | PERMISSIONS | LOGS | TESTING |

Runtime : Python 3.12   **Entry point** : weatherdata   EDIT   ⬇ DOWNLOAD ZIP

📄 main.py

📄 requirements.txt

```python
from google.cloud import bigquery
from google.cloud import storage
import pandas as pd
from io import import StringIO

# Replace with your actual project ID
PROJECT_ID = 'probable-hash-285404'
DATASET_ID = 'Capstone824'
TABLE_ID = 'static_energyconsumptionbyzip_NY'

def weatherdata(event, context):
    """Triggered by a change to a Cloud Storage bucket.
    Args:
        event (dict): The Cloud Functions event payload.
        context (google.cloud.functions.Context): Metadata of triggeri
    """

    # Get the file that triggered the function
    file_name = event['name']
    bucket_name = event['bucket']

    # Instantiate clients
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(file_name)
    blob_string = blob.download_as_text()

    # Set chunk size (number of rows per chunk)
    chunk_size = 10000

    # Define the schema once for reuse
    schema = [
```
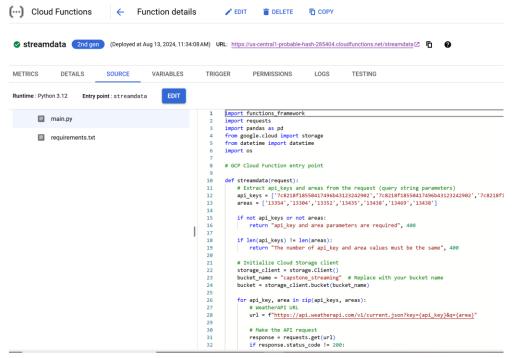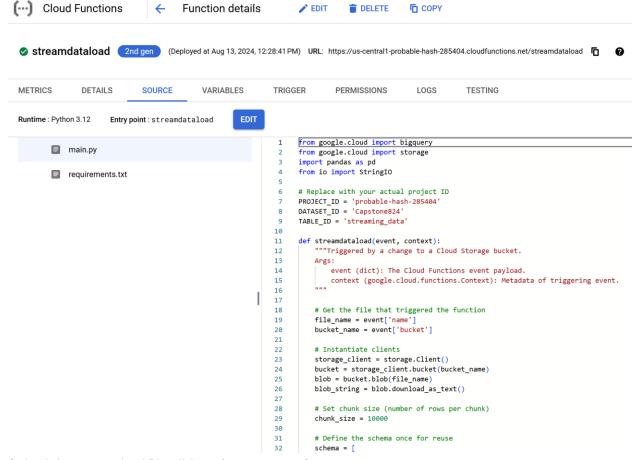
3) Cloud function trigger log:

4) Big Query table loaded through cloud function trigger

1) Streaming data load cloud function to call APIs and load the data into cloud storage:

✅ **streamdata** `2nd gen` (Deployed at Aug 13, 2024, 11:34:08 AM)   URL: https://us-central1-probable-hash-285404.cloudfunctions.net/streamdata ↗ 📋 ❓

METRICS   DETAILS   **SOURCE**   VARIABLES   TRIGGER   PERMISSIONS   LOGS   TESTING

Runtime : Python 3.12   **Entry point** : streamdata   **EDIT**

📄 main.py

📄 requirements.txt

```python
import functions_framework
import requests
import pandas as pd
from google.cloud import storage
from datetime import datetime
import os

# GCP Cloud Function entry point

def streamdata(request):
    # Extract api_keys and areas from the request (query string parameters)
    api_keys = ['7c8218f18550417496b43123242902','7c8218f18550417496b43123242902','7c8218f1
    areas = ['13354','13304','13352','13435','13438','13469','13438']

    if not api_keys or not areas:
        return "api_key and area parameters are required", 400

    if len(api_keys) != len(areas):
        return "The number of api_key and area values must be the same", 400

    # Initialize Cloud Storage client
    storage_client = storage.Client()
    bucket_name = "capstone_streaming"  # Replace with your bucket name
    bucket = storage_client.bucket(bucket_name)

    for api_key, area in zip(api_keys, areas):
        # WeatherAPI URL
        url = f"https://api.weatherapi.com/v1/current.json?key={api_key}&q={area}"

        # Make the API request
        response = requests.get(url)
        if response.status_code != 200:
```

2) Cloud function to load data from cloud storage to BigQuery on the cloud storage trigger

✅ **streamdataload** `2nd gen` (Deployed at Aug 13, 2024, 12:28:41 PM)   URL: https://us-central1-probable-hash-285404.cloudfunctions.net/streamdataload 📋 ❓

METRICS   DETAILS   **SOURCE**   VARIABLES   TRIGGER   PERMISSIONS   LOGS   TESTING

Runtime : Python 3.12   **Entry point** : streamdataload   **EDIT**

📄 main.py

📄 requirements.txt

```python
from google.cloud import bigquery
from google.cloud import storage
import pandas as pd
from io import StringIO

# Replace with your actual project ID
PROJECT_ID = 'probable-hash-285404'
DATASET_ID = 'Capstone824'
TABLE_ID = 'streaming_data'

def streamdataload(event, context):
    """Triggered by a change to a Cloud Storage bucket.
    Args:
        event (dict): The Cloud Functions event payload.
        context (google.cloud.functions.Context): Metadata of triggering event.
    """

    # Get the file that triggered the function
    file_name = event['name']
    bucket_name = event['bucket']

    # Instantiate clients
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(file_name)
    blob_string = blob.download_as_text()

    # Set chunk size (number of rows per chunk)
    chunk_size = 10000

    # Define the schema once for reuse
    schema = [
```

3) Scheduler to run the API call function every 5 minutes

4) Big Query table loading streaming data: