Curso: Tecnologia em Data Science, Big Data, BI & Data Engineering

Projeto: CHALLENGE (012024)

Turmas do 1o ano: 1TSCPV

Sprint 3

DataStorm

Ana Beatriz Azevedo, RM557420

Heloiza Oliveira, RM558881

Isabelle Nahas, RM557405

Matheus Madrid, RM555799

Sara Sitta, RM555113

DISCIPLINA: BIG DATA ARCHITECTURE & DATA INTEGRATION

Professores: Rogerio Mariana Galazi, Milton Goya

São Paulo, Setembro de 2024

Documentação do Pipeline de Dados

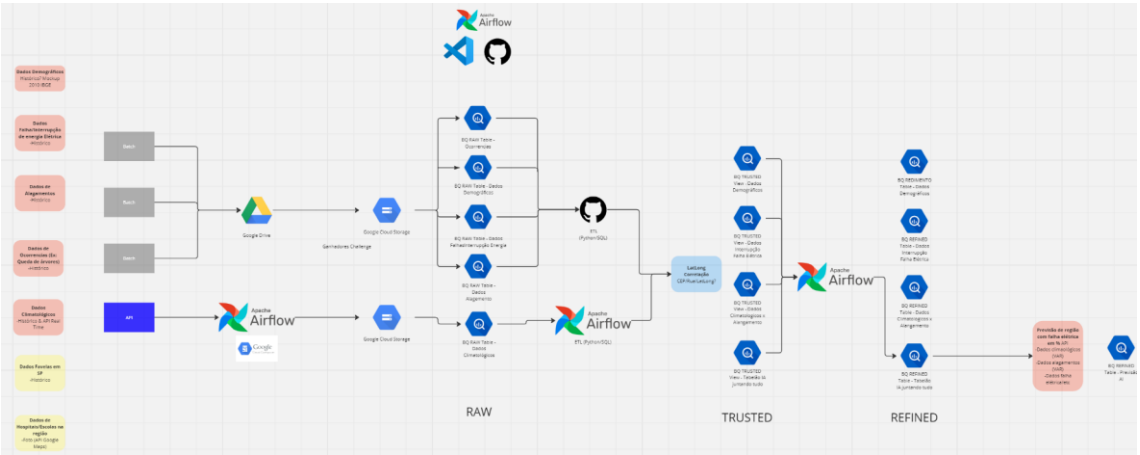
Visão Geral

Este documento descreve o pipeline de dados implementado para ingestão, transformação, armazenamento e consumo de dados climáticos coletados de uma API, usando Apache Airflow no Google Cloud Composer, Google Cloud Storage como Data Lake, e BigQuery para armazenamento e transformação dos dados. O pipeline é composto por três DAGs principais, cada uma responsável por uma etapa específica do fluxo de dados.

1. Ferramenta de ETL/Ingestão

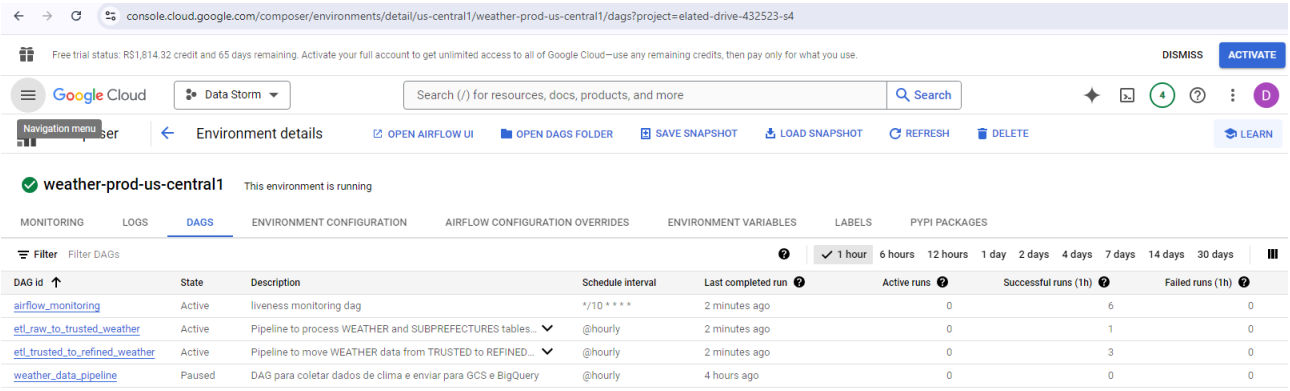
Para atender aos requisitos de coleta, ingestão, armazenamento, transformação e orquestração, foram utilizados Apache Airflow no Google Cloud Composer para orquestração, Google Cloud Storage para armazenamento temporário, e BigQuery para transformação e armazenamento dos dados estruturados.

Arquitetura atualizada:

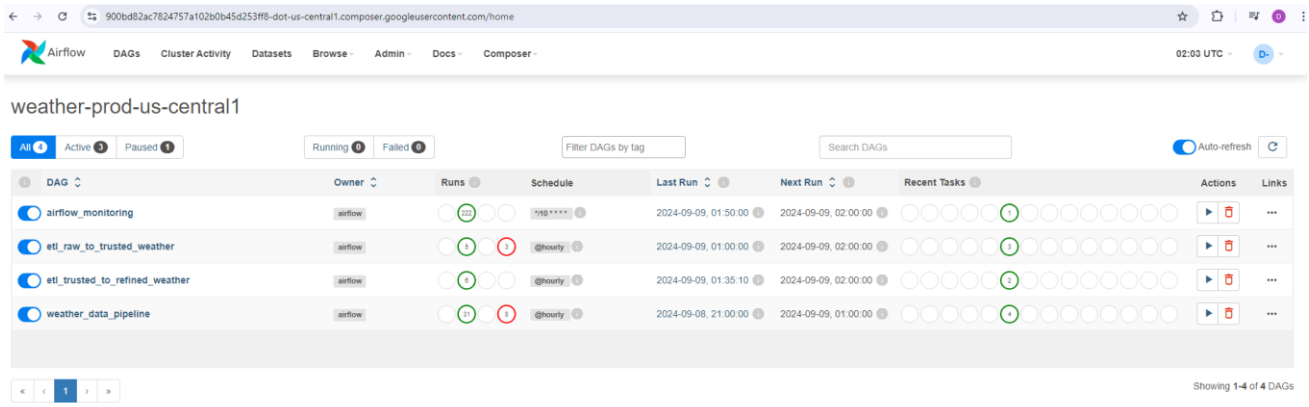


Link: https://miro.com/app/board/uXjVKmDD4gU=/?share_link_id=695605839813

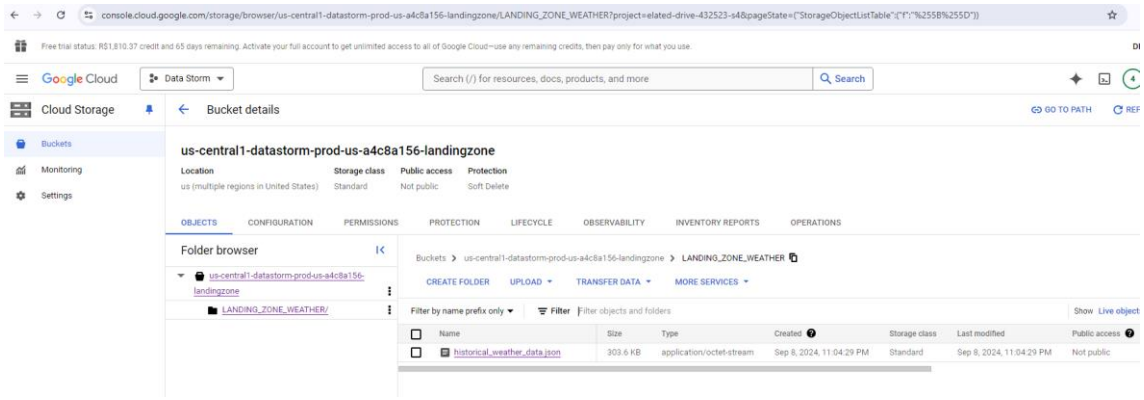
Google Cloud Composer



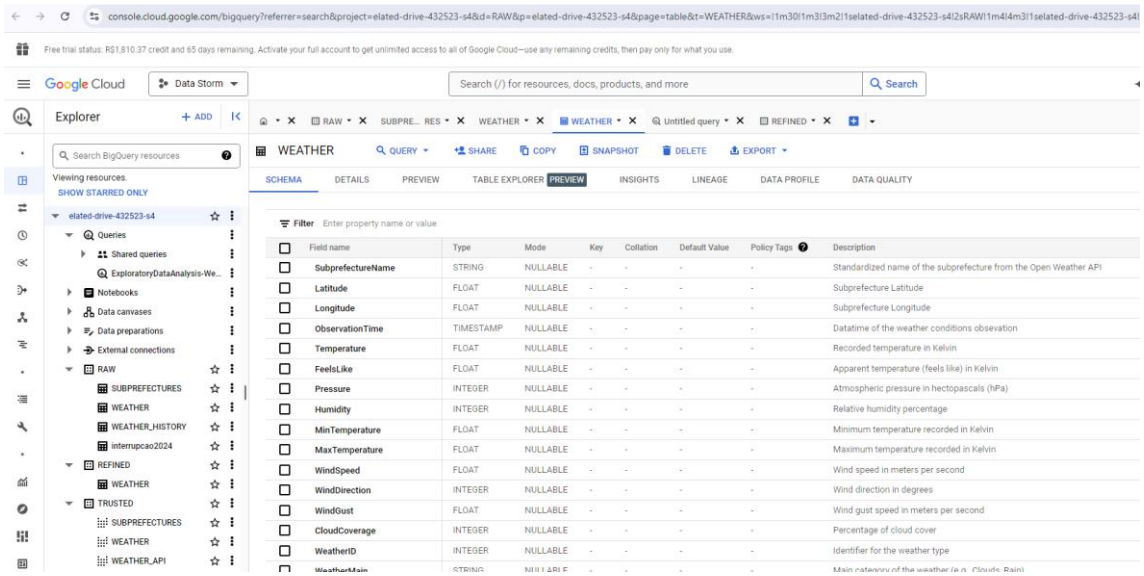
Apache Airflow:



CloudStorage – Landing Zone

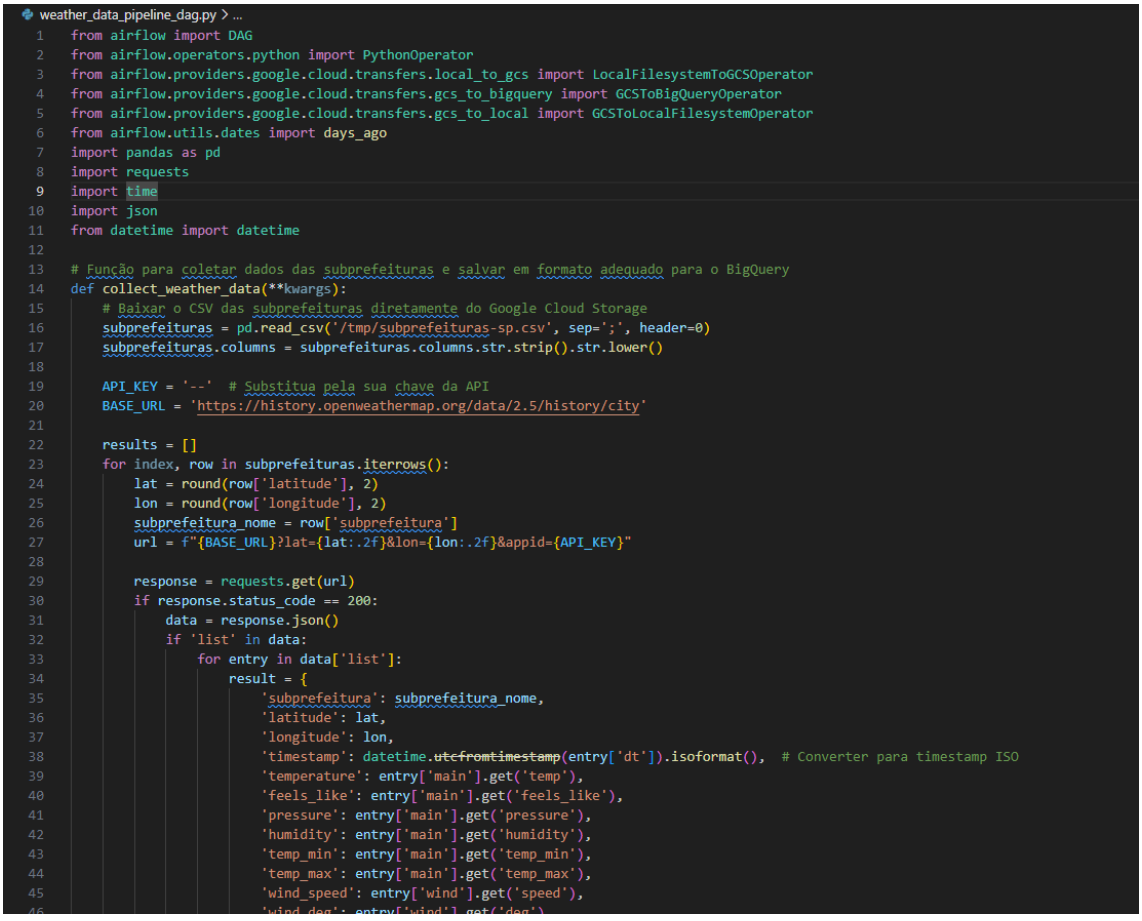


Bigquery GCP



1.1 Coleta e Ingestão de Dados para o Data Lake e BigQuery (DAG: weather_data_pipeline_dag)

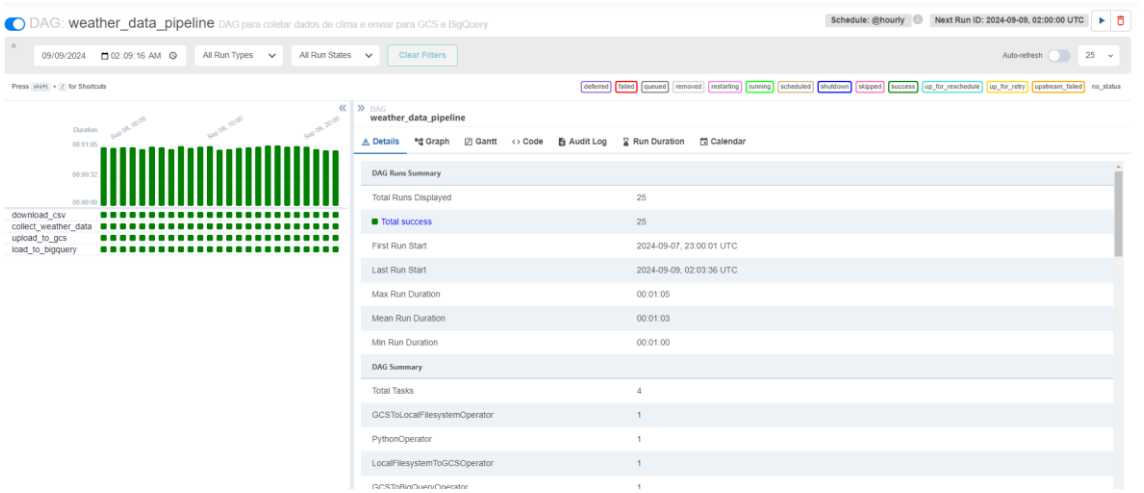
- **Nome da DAG:** weather_data_pipeline_dag
- **Ferramenta Utilizada:** Apache Airflow com Google Cloud Composer para orquestração; Google Cloud Storage para armazenamento dos dados e BigQuery para estruturação do Lake
- **Processo:**
    - **Coleta de Dados:** A DAG coleta dados climáticos da API do OpenWeatherMap em intervalos regulares (a cada 1 hora).
    - **Armazenamento Temporário:** Os dados coletados são salvos no Google Cloud Storage, que atua como o Data Lake.
    - **Carga para BigQuery:** Após salvar no Cloud Storage, os dados são carregados na camada RAW do BigQuery, utilizando operadores específicos do Airflow para transferência de arquivos do Cloud Storage para o BigQuery.

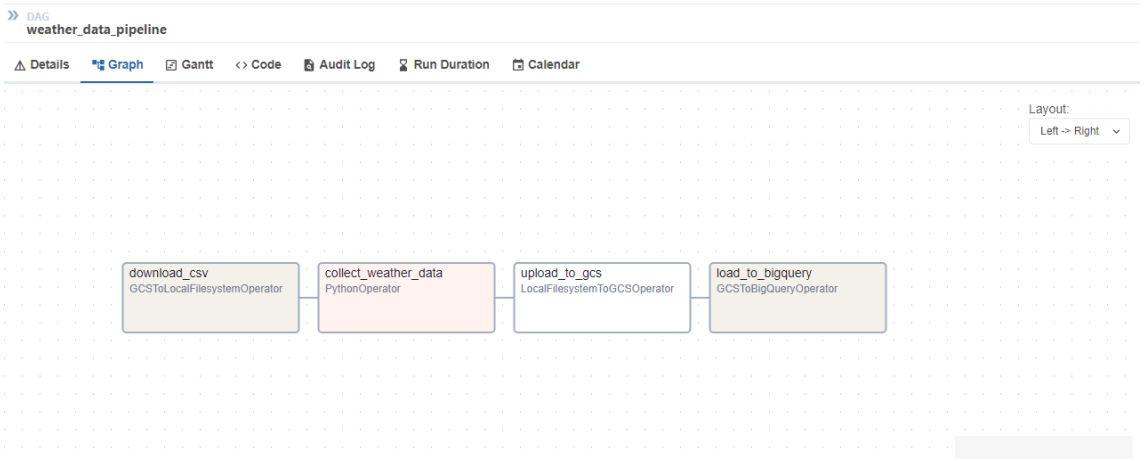


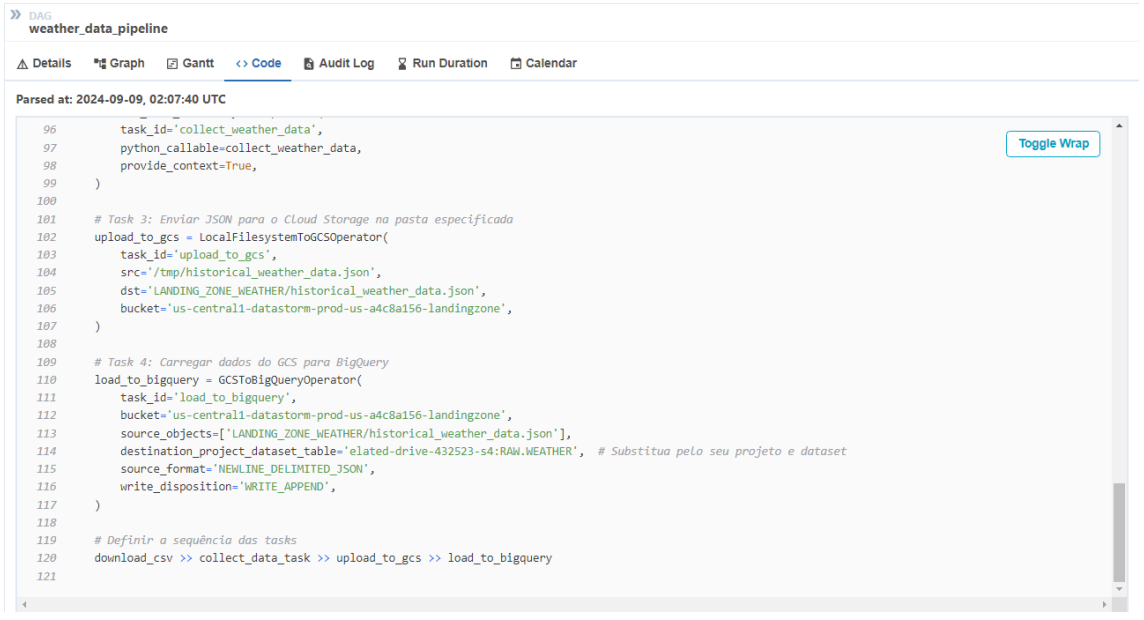
```python
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.providers.google.cloud.transfers.local_to_gcs import LocalFilesystemToGCSOperator
from airflow.providers.google.cloud.transfers.gcs_to_bigquery import GCSToBigQueryOperator
from airflow.providers.google.cloud.transfers.gcs_to_local import GCSToLocalFilesystemOperator
from airflow.utils.dates import days_ago
import pandas as pd
import requests
import time
import json
from datetime import datetime

# Função para coletar dados das subprefeituras e salvar em formato adequado para o BigQuery
def collect_weather_data(**kwargs):
    # Baixar o CSV das subprefeituras diretamente do Google Cloud Storage
    subprefeituras = pd.read_csv('/tmp/subprefeituras-sp.csv', sep=';', header=0)
    subprefeituras.columns = subprefeituras.columns.str.strip().str.lower()

    API_KEY = '--'  # Substitua pela sua chave da API
    BASE_URL = 'https://history.openweathermap.org/data/2.5/history/city'

    results = []
    for index, row in subprefeituras.iterrows():
        lat = round(row['latitude'], 2)
        lon = round(row['longitude'], 2)
        subprefeitura_nome = row['subprefeitura']
        url = f"{BASE_URL}?lat={lat:.2f}&lon={lon:.2f}&appid={API_KEY}"

        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()
            if 'list' in data:
                for entry in data['list']:
                    result = {
                        'subprefeitura': subprefeitura_nome,
                        'latitude': lat,
                        'longitude': lon,
                        'timestamp': datetime.utcfromtimestamp(entry['dt']).isoformat(),  # Converter para timestamp ISO
                        'temperature': entry['main'].get('temp'),
                        'feels_like': entry['main'].get('feels_like'),
                        'pressure': entry['main'].get('pressure'),
                        'humidity': entry['main'].get('humidity'),
                        'temp_min': entry['main'].get('temp_min'),
                        'temp_max': entry['main'].get('temp_max'),
                        'wind_speed': entry['wind'].get('speed'),
                        'wind_deg': entry['wind'].get('deg'),
```

Arquivo anexado - weather_data_pipeline_dag.py

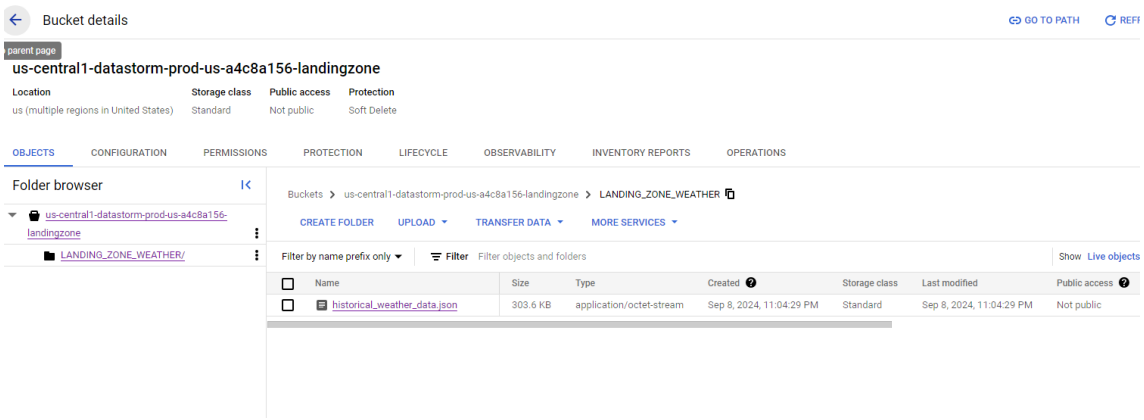
weather_data_pipeline_dag. DAG no Airflow



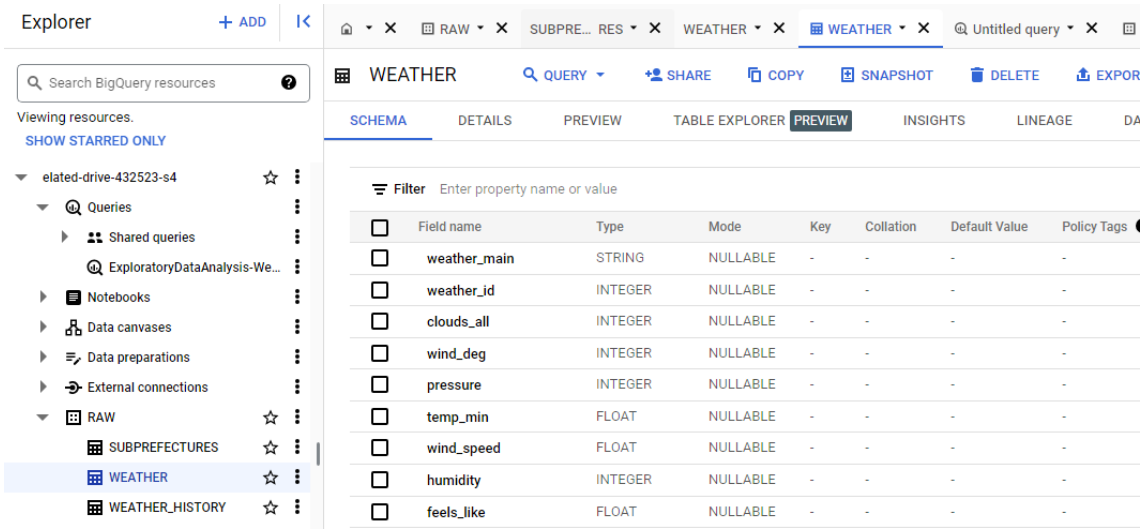




LANDING ZONE no Google Cloud Storage



Camada RAW.WEATHER



1.2 Estruturação e Preparação dos Dados para a Camada Estruturada (DAG: etl_raw_to_trusted_weather)

- **Nome da DAG:** etl_raw_to_trusted_weather
- **Ferramenta Utilizada:** Apache Airflow com Google Cloud Composer para orquestração; Google Cloud Storage para armazenamento dos dados e BigQuery para transformação dos dados.
- **Processo:**
    - Esta DAG transforma e estrutura os dados movendo-os da camada RAW para a camada TRUSTED no BigQuery.
    - **Transformações Incluídas:**
        - Ajuste de precisão de latitude e longitude para garantir a consistência entre os conjuntos de dados.
        - Conversão de CEP de string para inteiro, removendo caracteres indesejados.
        - Criação de views para padronizar os nomes das colunas para o padrão de Data Lake (letra maiúscula, sem underscores).

Task 4 da DAG weather_data_pipeline_dag

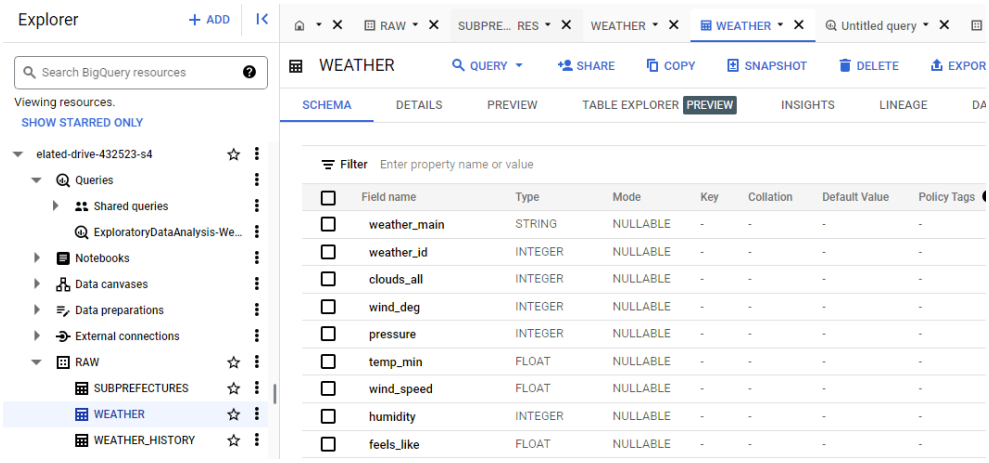
DAG
weather_data_pipeline

Details | Graph | Gantt | Code | Audit Log | Run Duration | Calendar

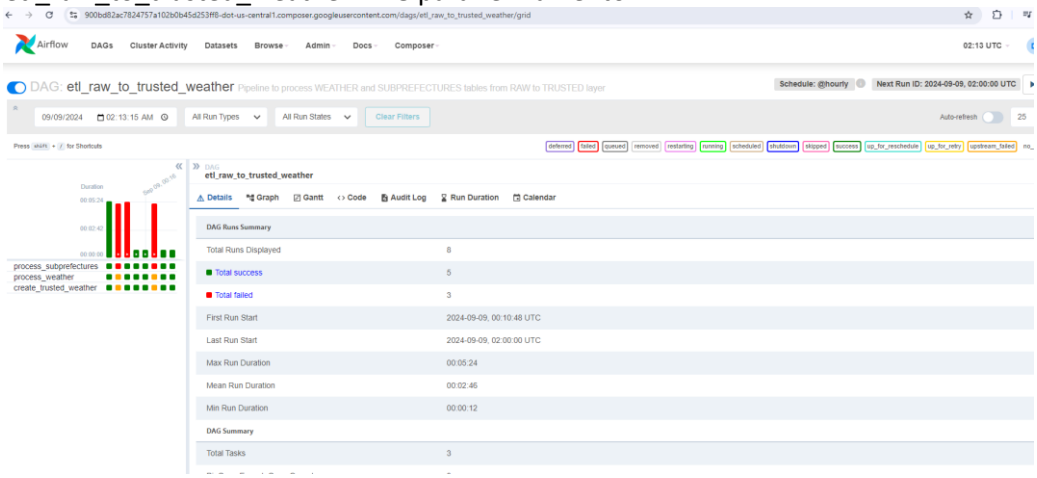
Parsed at: 2024-09-09, 02:07:40 UTC

```
        task_id='collect_weather_data',
        python_callable=collect_weather_data,
        provide_context=True,
    )

    # Task 3: Enviar JSON para o Cloud Storage na pasta especificada
    upload_to_gcs = LocalFilesystemToGCSOperator(
        task_id='upload_to_gcs',
        src='/tmp/historical_weather_data.json',
        dst='LANDING_ZONE_WEATHER/historical_weather_data.json',
        bucket='us-central1-datastorm-prod-us-a4c8a156-landingzone',
    )

    # Task 4: Carregar dados do GCS para BigQuery
    load_to_bigquery = GCSToBigQueryOperator(
        task_id='load_to_bigquery',
        bucket='us-central1-datastorm-prod-us-a4c8a156-landingzone',
        source_objects=['LANDING_ZONE_WEATHER/historical_weather_data.json'],
        destination_project_dataset_table='elated-drive-432523-s4:RAW.WEATHER',  # Substitua pelo seu projeto e dataset
        source_format='NEWLINE_DELIMITED_JSON',
        write_disposition='WRITE_APPEND',
    )

    # Definir a sequência das tasks
    download_csv >> collect_data_task >> upload_to_gcs >> load_to_bigquery
```

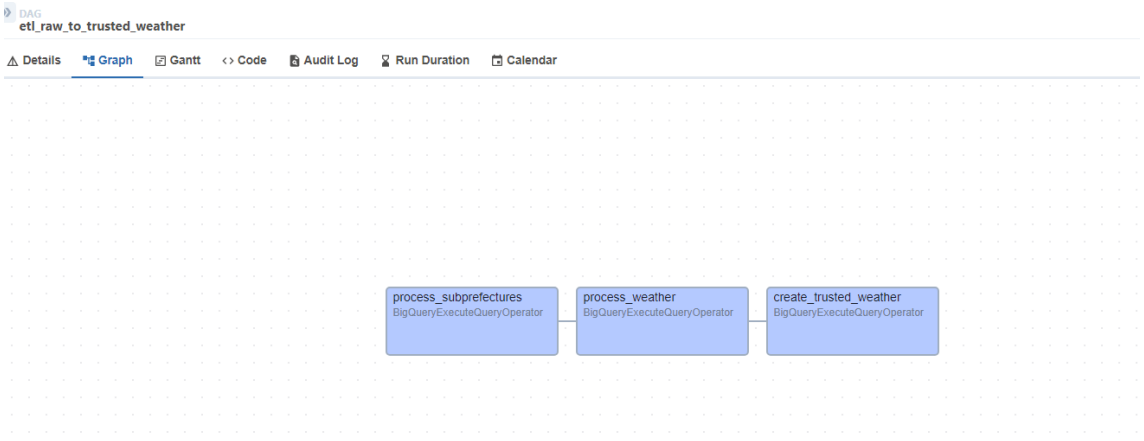
Camada RAW.WEATHER

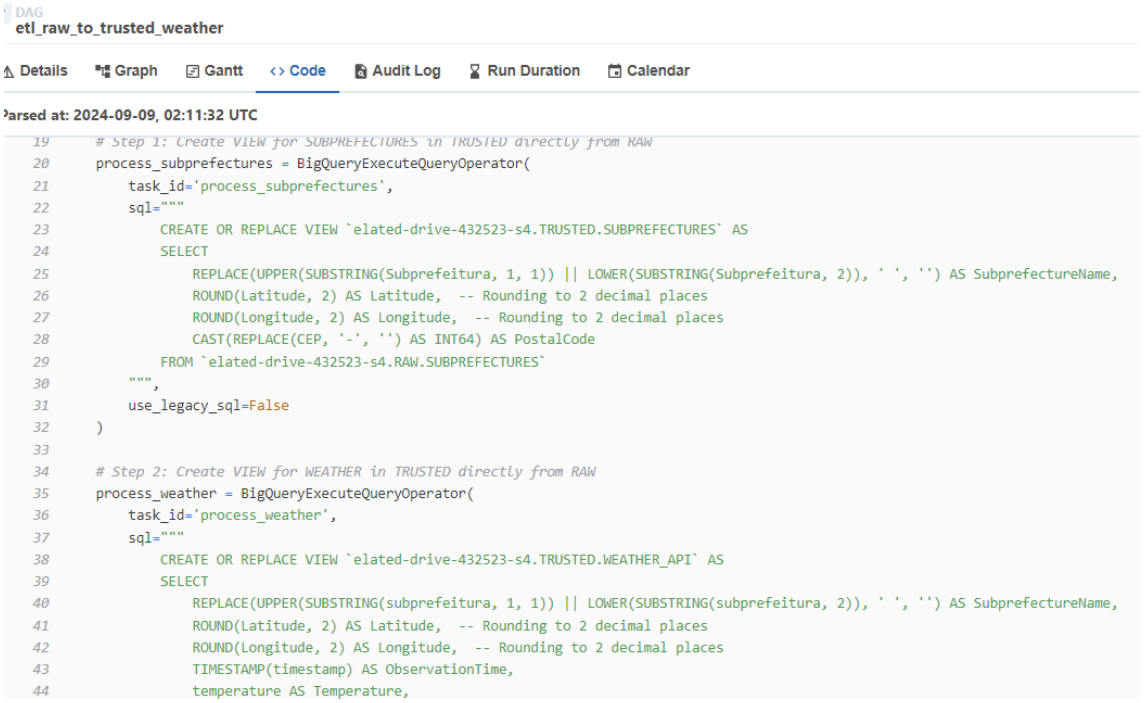


etl_raw_to_trusted_weather DAG para refinamento



Tasks da DAG

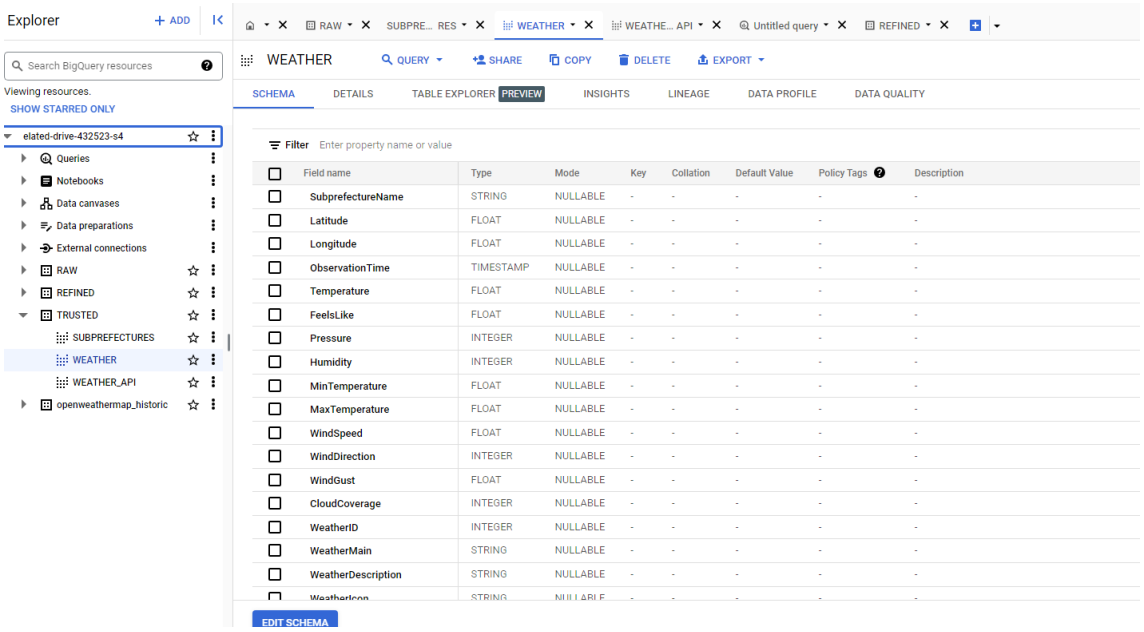





```
19    # Step 1: Create VIEW for SUBPREFECTURES in TRUSTED directly from RAW
20    process_subprefectures = BigQueryExecuteQueryOperator(
21        task_id='process_subprefectures',
22        sql="""
23            CREATE OR REPLACE VIEW `elated-drive-432523-s4.TRUSTED.SUBPREFECTURES` AS
24            SELECT
25                REPLACE(UPPER(SUBSTRING(Subprefeitura, 1, 1)) || LOWER(SUBSTRING(Subprefeitura, 2)), ' ', '') AS SubprefectureName,
26                ROUND(Latitude, 2) AS Latitude,  -- Rounding to 2 decimal places
27                ROUND(Longitude, 2) AS Longitude,  -- Rounding to 2 decimal places
28                CAST(REPLACE(CEP, '-', '') AS INT64) AS PostalCode
29            FROM `elated-drive-432523-s4.RAW.SUBPREFECTURES`
30        """,
31        use_legacy_sql=False
32    )
33
34    # Step 2: Create VIEW for WEATHER in TRUSTED directly from RAW
35    process_weather = BigQueryExecuteQueryOperator(
36        task_id='process_weather',
37        sql="""
38            CREATE OR REPLACE VIEW `elated-drive-432523-s4.TRUSTED.WEATHER_API` AS
39            SELECT
40                REPLACE(UPPER(SUBSTRING(subprefeitura, 1, 1)) || LOWER(SUBSTRING(subprefeitura, 2)), ' ', '') AS SubprefectureName,
41                ROUND(Latitude, 2) AS Latitude,  -- Rounding to 2 decimal places
42                ROUND(Longitude, 2) AS Longitude,  -- Rounding to 2 decimal places
43                TIMESTAMP(timestamp) AS ObservationTime,
44                temperature AS Temperature,
```

Dag anexada no zip etl_raw_to_trusted_weather.py

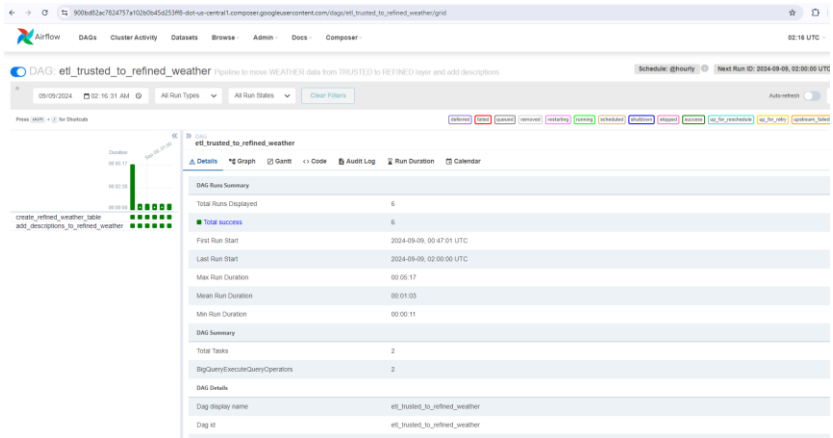
CAMADA TRUSTED

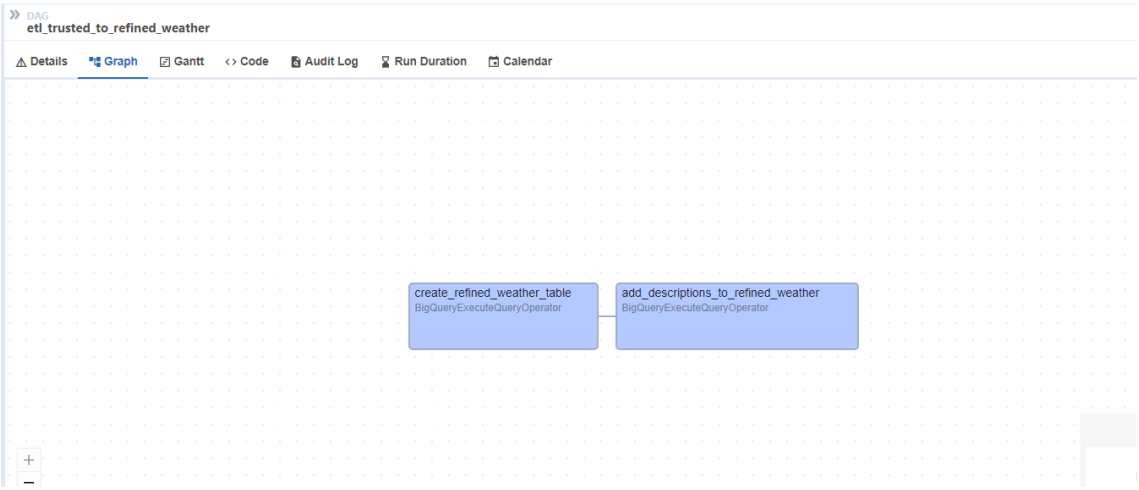


1.3 Refinamento e Enriquecimento dos Dados (DAG: etl_trusted_to_refined_weather)

- **Nome da DAG:** etl_trusted_to_refined_weather
- **Ferramenta Utilizada:** Apache Airflow com Google Cloud Composer para orquestração; BigQuery para refinamento e enriquecimento dos dados.
- **Processo:**
  - A última DAG move os dados da camada TRUSTED para a camada REFINED no BigQuery.
  - **Refinamento Inclui:**
    - Criação de tabelas finais na camada REFINED.
    - Adição de descrições das colunas para melhorar a documentação e a compreensão dos dados.
    - Junção de dados climáticos com informações de subprefeituras para fornecer um conjunto de dados enriquecido.

DAG etl_trusted_to_refined_weather no Airflow





DAG
etl_trusted_to_refined_weather

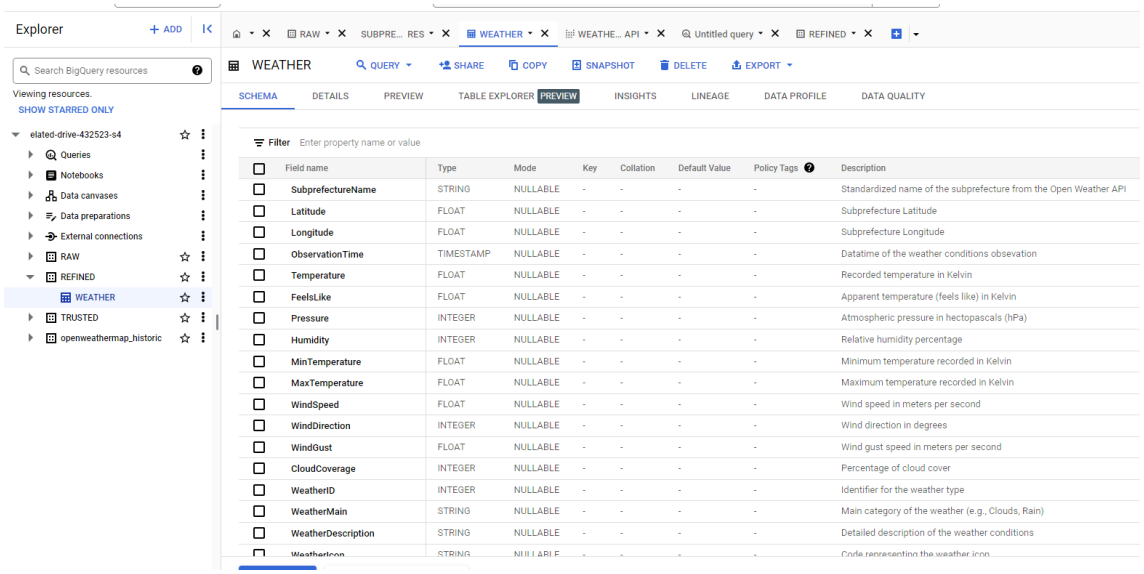
Details | Graph | Gantt | Code | Audit Log | Run Duration | Calendar

Parsed at: 2024-09-09, 02:15:25 UTC

```
    # Step 2: Consolidated ALTER TABLE to Add Descriptions to REFINED.WEATHER Table
    add_descriptions_to_refined_weather = BigQueryExecuteQueryOperator(
        task_id='add_descriptions_to_refined_weather',
        sql="""
            ALTER TABLE `elated-drive-432523-s4.REFINED.WEATHER`
            SET OPTIONS (
              description = 'Refined table with enriched weather and subprefecture data'
            );

            ALTER TABLE `elated-drive-432523-s4.REFINED.WEATHER`
            ALTER COLUMN SubprefectureName SET OPTIONS (description = 'Standardized name of the subprefecture from the Open Weather API'),
            ALTER COLUMN Latitude SET OPTIONS (description = 'Subprefecture Latitude'),
            ALTER COLUMN Longitude SET OPTIONS (description = 'Subprefecture Longitude'),
            ALTER COLUMN ObservationTime SET OPTIONS (description = 'Datatime of the weather conditions obsevation'),
            ALTER COLUMN Temperature SET OPTIONS (description = 'Recorded temperature in Kelvin'),
            ALTER COLUMN FeelsLike SET OPTIONS (description = 'Apparent temperature (feels like) in Kelvin'),
            ALTER COLUMN Pressure SET OPTIONS (description = 'Atmospheric pressure in hectopascals (hPa)'),
            ALTER COLUMN Humidity SET OPTIONS (description = 'Relative humidity percentage'),
            ALTER COLUMN MinTemperature SET OPTIONS (description = 'Minimum temperature recorded in Kelvin'),
            ALTER COLUMN MaxTemperature SET OPTIONS (description = 'Maximum temperature recorded in Kelvin'),
            ALTER COLUMN WindSpeed SET OPTIONS (description = 'Wind speed in meters per second'),
            ALTER COLUMN WindDirection SET OPTIONS (description = 'Wind direction in degrees'),
            ALTER COLUMN WindGust SET OPTIONS (description = 'Wind gust speed in meters per second'),
            ALTER COLUMN CloudCoverage SET OPTIONS (description = 'Percentage of cloud cover'),
            ALTER COLUMN WeatherID SET OPTIONS (description = 'Identifier for the weather type'),
```

Arquivo anexado no zip etl_trusted_to_refined_weather.py

CAMADA REFINED com a tabela WEATHER final com as Descrições

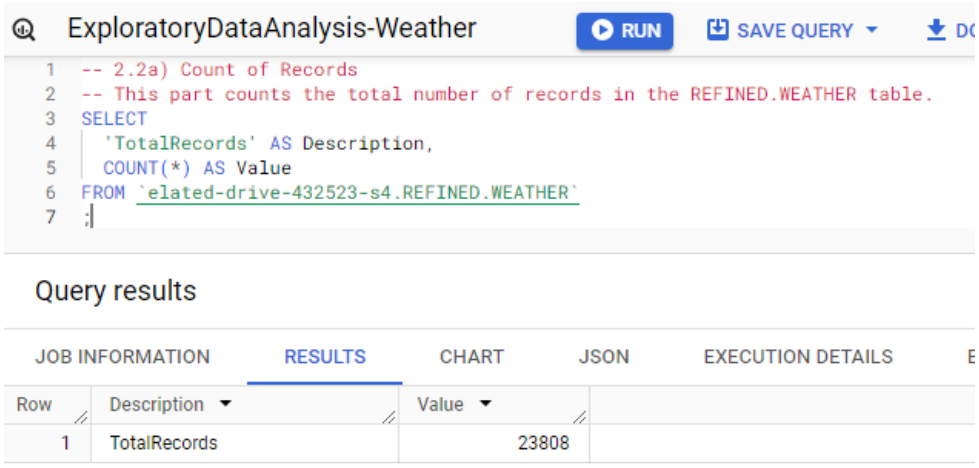


2. Análise Exploratória no Ambiente SQL (BigQuery)

Utilizando BigQuery para análise exploratória dos dados refinados, foram realizadas as seguintes operações:

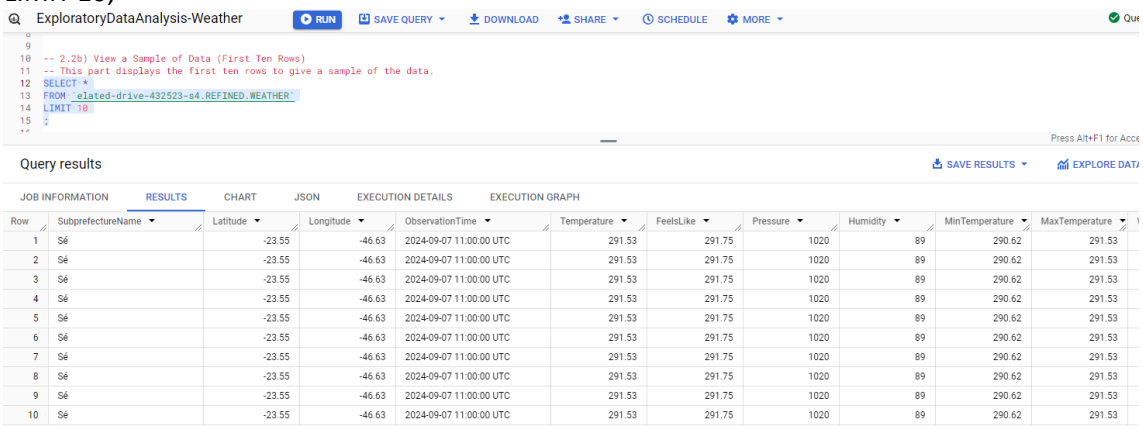
2.2a) Contagem de Registros

SELECT COUNT(*) AS TotalRecords
FROM `elated-drive-432523-s4.REFINED.WEATHER`;



2.2b) Visualização de Amostra dos Dados (Primeiras Dez Linhas)

```
SELECT *
FROM `elated-drive-432523-s4.REFINED.WEATHER`
LIMIT 10;
```



ExploratoryDataAnalysis-Weather

RUN | SAVE QUERY | DOWNLOAD | SHARE | SCHEDULE | MORE

```
-- 2.2b) View a Sample of Data (First Ten Rows)
-- This part displays the first ten rows to give a sample of the data.
SELECT *
FROM `elated-drive-432523-s4.REFINED.WEATHER`
LIMIT 10
;
```

Query results

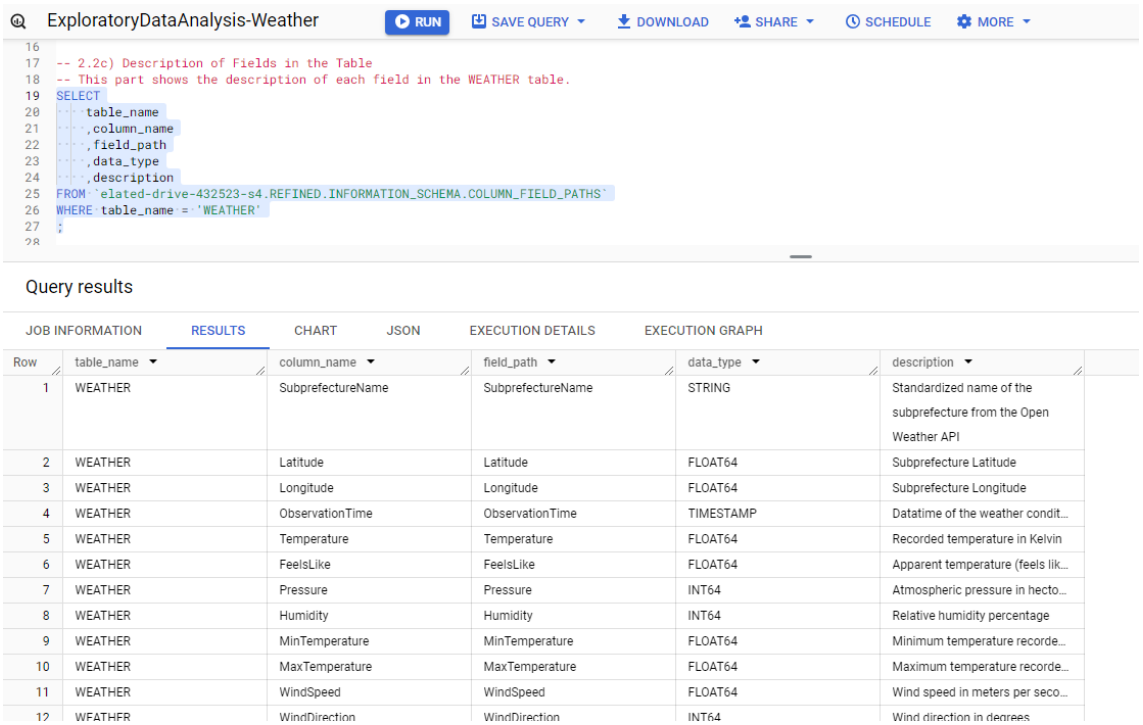
SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	SubprefectureName	Latitude	Longitude	ObservationTime	Temperature	FeelsLike	Pressure	Humidity	MinTemperature	MaxTemperature
1	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
2	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
3	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
4	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
5	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
6	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
7	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
8	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
9	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53
10	Sé	-23.55	-46.63	2024-09-07 11:00:00 UTC	291.53	291.75	1020	89	290.62	291.53

2.2c) Descrição dos Campos da Tabela

```
SELECT
    table_name
    ,column_name
    ,field_path
    ,data_type
    ,description
FROM `elated-drive-432523-s4.REFINED.INFORMATION_SCHEMA.COLUMN_FIELD_PATHS`
WHERE table_name = 'WEATHER'
;
```



ExploratoryDataAnalysis-Weather

RUN | SAVE QUERY | DOWNLOAD | SHARE | SCHEDULE | MORE

```
-- 2.2c) Description of Fields in the Table
-- This part shows the description of each field in the WEATHER table.
SELECT
    table_name
    ,column_name
    ,field_path
    ,data_type
    ,description
FROM `elated-drive-432523-s4.REFINED.INFORMATION_SCHEMA.COLUMN_FIELD_PATHS`
WHERE table_name = 'WEATHER'
;
```

Query results

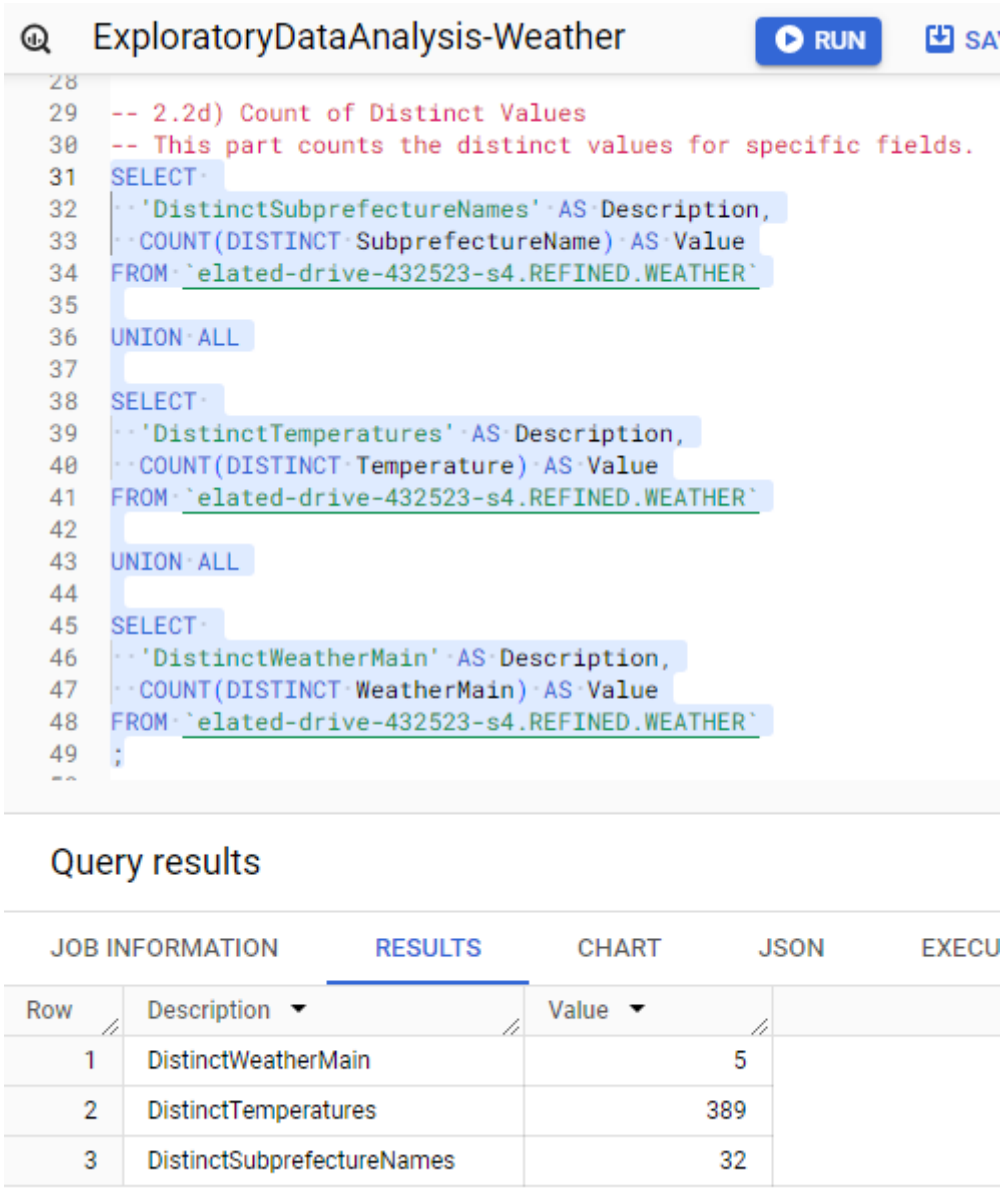
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	table_name	column_name	field_path	data_type	description
1	WEATHER	SubprefectureName	SubprefectureName	STRING	Standardized name of the subprefecture from the Open Weather API
2	WEATHER	Latitude	Latitude	FLOAT64	Subprefecture Latitude
3	WEATHER	Longitude	Longitude	FLOAT64	Subprefecture Longitude
4	WEATHER	ObservationTime	ObservationTime	TIMESTAMP	Datatime of the weather condit...
5	WEATHER	Temperature	Temperature	FLOAT64	Recorded temperature in Kelvin
6	WEATHER	FeelsLike	FeelsLike	FLOAT64	Apparent temperature (feels lik...
7	WEATHER	Pressure	Pressure	INT64	Atmospheric pressure in hecto...
8	WEATHER	Humidity	Humidity	INT64	Relative humidity percentage
9	WEATHER	MinTemperature	MinTemperature	FLOAT64	Minimum temperature recorde...
10	WEATHER	MaxTemperature	MaxTemperature	FLOAT64	Maximum temperature recorde...
11	WEATHER	WindSpeed	WindSpeed	FLOAT64	Wind speed in meters per seco...
12	WEATHER	WindDirection	WindDirection	INT64	Wind direction in degrees

2.2d) Contagem de Valores Distintos

Exemplo

SELECT COUNT(DISTINCT SubprefectureName) AS DistinctSubprefectureNames
FROM `elated-drive-432523-s4.REFINED.WEATHER`;

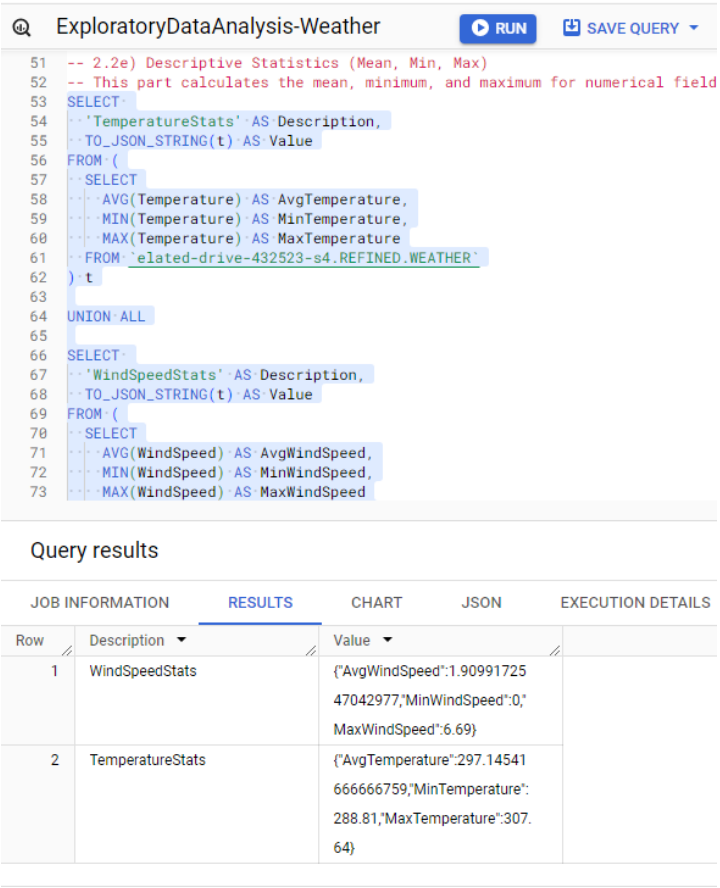


2.2e) Estatísticas Descritivas (Média, Mínimo, Máximo)

Exemplo de cálculo para a coluna Temperature:

```
SELECT
  'TemperatureStats' AS Description,
  TO_JSON_STRING(t) AS Value
FROM (
  SELECT
    AVG(Temperature) AS AvgTemperature,
    MIN(Temperature) AS MinTemperature,
    MAX(Temperature) AS MaxTemperature
```

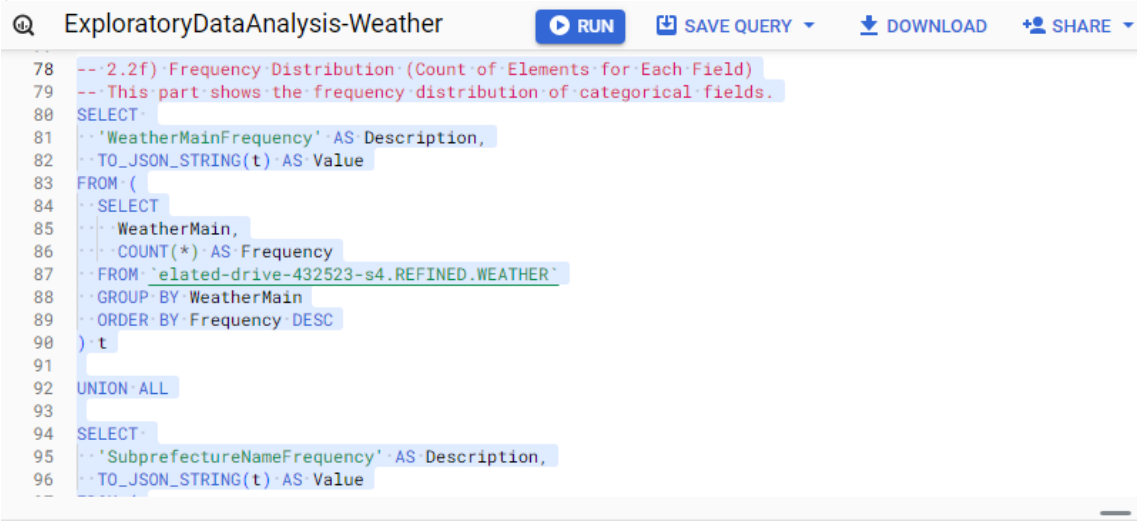
```
  FROM `elated-drive-432523-s4.REFINED.WEATHER`
) t

UNION ALL

SELECT
  'WindSpeedStats' AS Description,
  TO_JSON_STRING(t) AS Value
FROM (
  SELECT
    AVG(WindSpeed) AS AvgWindSpeed,
    MIN(WindSpeed) AS MinWindSpeed,
    MAX(WindSpeed) AS MaxWindSpeed
  FROM `elated-drive-432523-s4.REFINED.WEATHER`
) t
;
```



2.2f) Distribuição de Frequência

```
-- 2.2f) Frequency Distribution (Count of Elements for Each Field)
-- This part shows the frequency distribution of categorical fields.
SELECT
  'WeatherMainFrequency' AS Description,
  TO_JSON_STRING(t) AS Value
FROM (
  SELECT
    WeatherMain,
    COUNT(*) AS Frequency
  FROM `elated-drive-432523-s4.REFINED.WEATHER`
```

```
  GROUP BY WeatherMain
  ORDER BY Frequency DESC
) t

UNION ALL

SELECT
  'SubprefectureNameFrequency' AS Description,
  TO_JSON_STRING(t) AS Value
FROM (
  SELECT
    SubprefectureName,
    COUNT(*) AS Frequency
  FROM `elated-drive-432523-s4.REFINED.WEATHER`
  GROUP BY SubprefectureName
  ORDER BY Frequency DESC
) t;
```



Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	Description	Value
1	WeatherMainFrequency	{"WeatherMain":"Clouds","Frequency":8420}
2	WeatherMainFrequency	{"WeatherMain":"Clear","Frequency":14360}
3	WeatherMainFrequency	{"WeatherMain":"Mist","Frequency":510}
4	WeatherMainFrequency	{"WeatherMain":"Haze","Frequency":358}
5	WeatherMainFrequency	{"WeatherMain":"Fog","Frequency":160}
6	SubprefectureNameFrequency	{"SubprefectureName":"Sé","Frequency":744}
7	SubprefectureNameFrequency	{"SubprefectureName":"Lapa","Frequency":744}
8	SubprefectureNameFrequency	{"SubprefectureName":"Mooca","Frequency":744}
9	SubprefectureNameFrequency	{"SubprefectureName":"Penha","Frequency":744}
10	SubprefectureNameFrequency	{"SubprefectureName":"Perus","Frequency":744}

3. Documentação e Evidências

Todas as etapas foram capturadas e documentadas para comprovar o funcionamento do pipeline.

Conclusão