𝗘𝗮𝗿𝘁𝗵𝗾𝘂𝗮𝗸𝗲 𝗗𝗮𝘁𝗮 𝗨𝗦𝗚𝗦 API Data Using Dataproc

Mohini Zurange

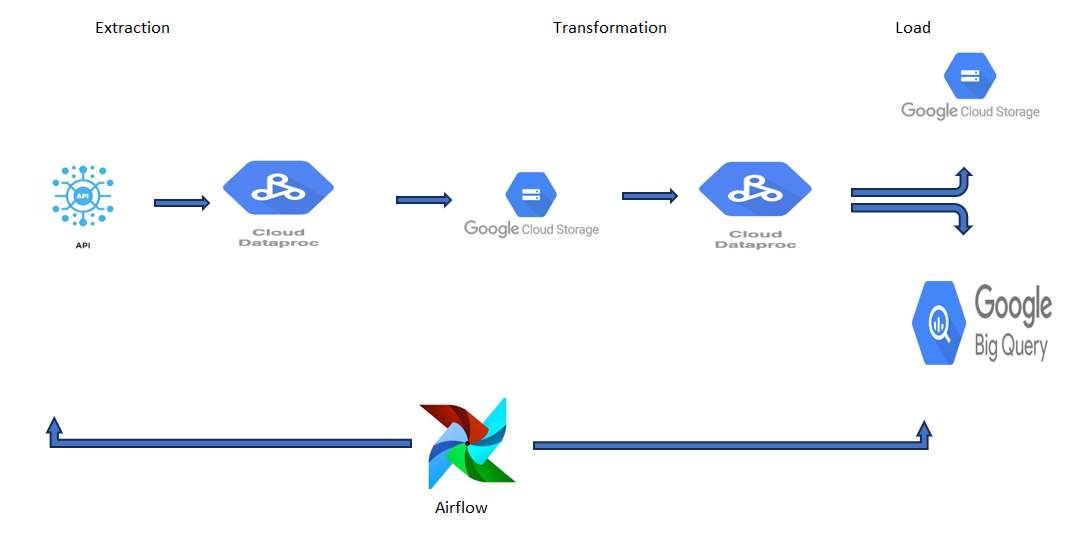
**Architecture:**

Load

Extract

Read &trans

Load



**Task: 1 Extract Data from API**

* **Description**: Fetches data from an API endpoint and converts it to a JSON string if successful. Logs the result or any errors.
* **Flow Diagram** :

Extract

Earthquake API

**JSON String (Python request lib)**

* **Input**:

api\_url (string): The API endpoint URL.

(url: <https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_month.geojson>)

* **Output**:

JSON string if successful, otherwise None.

* **Sample Input & Output data data :**



**Task: 2 Write Data to GCS as Parquet**

* **Description**: Converts JSON string data to a Spark DataFrame and writes it as a Parquet file to a GCS location(landing location)
* **Flow Diagram** :

convert

**JSON String (Python request lib)**

**Dataframe**

write

**GCS**

**(Landing loc. parq)**

* **Input**:

Dataframe : dataframe (json string convert into df)

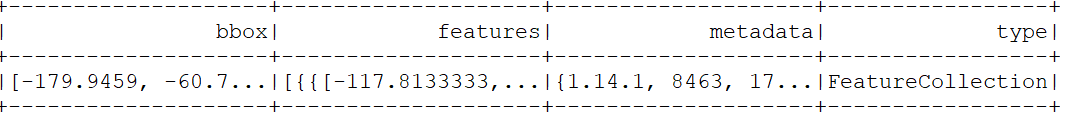
GCS\_location (string): GCS path for output.

* **Output**:

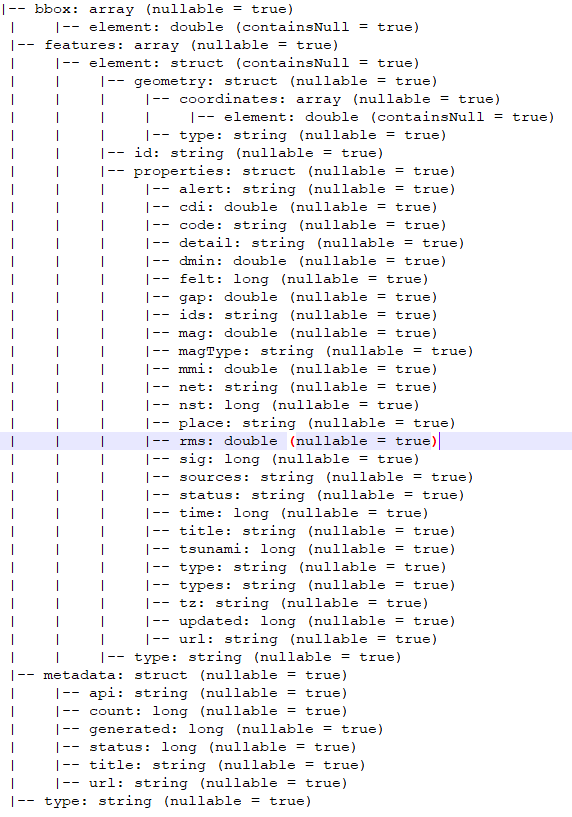
Return None

Write data to landing loction (GCS) (***"gs://earthquake\_analysis\_buck/pysaprk/landing/{cur\_timestamp}/")***

* **Sample input data :**



* **input\_schema**

   
**Task: 3 Read Parquet File from GCS**

* **Description**: Reads a Parquet file from GCS (landing loc)and returns it as a Spark DataFrame
* **Flow Diagram** :

read

**GCS**

**(Landing loc.parq)**

**Dataframe**

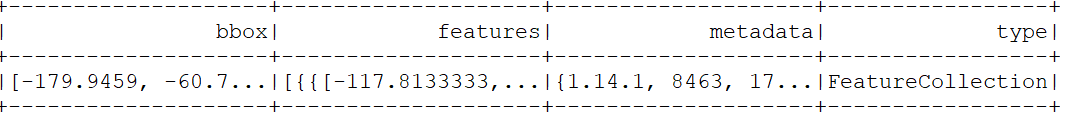
* **Input**:

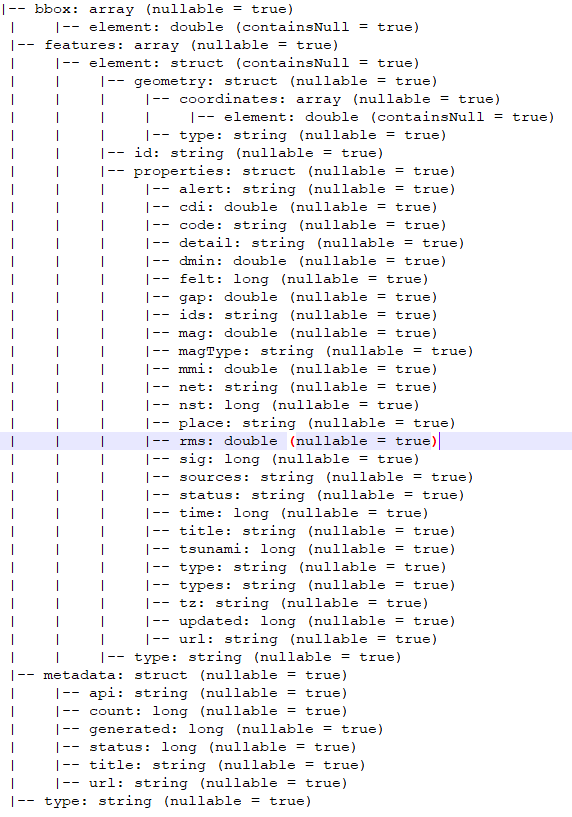
spark (SparkSession): The Spark session.

source\_data\_loc (string): GCS path of the Parquet file.(Landing Loc. ***"gs://earthquake\_analysis\_buck/pysaprk/landing/{cur\_timestamp}/")***

* **Output**:

DataFrame: Spark DataFrame containing the data.

* **Sample output data :** 
* **output\_schema**

   
**Task: 4 Extract Required Data, Flatten, and Apply Transformations**

* **Description**: Flattens nested data, extracts specific fields, and apply transformations to create a cleaned DataFrame.
* **Flow Diagram** :

**Dataframe**

**(Raw Data)**

**Flatten Nested data**

**Extract Fields & Apply Transformations**

**Dataframe**

**(Clean Data)**

* **Input**:

raw\_data\_df (DataFrame): The raw input DataFrame containing nested data.

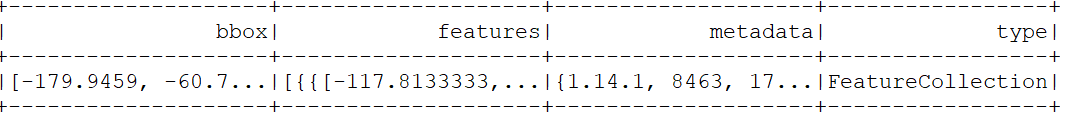
* **Transformation details**:

1)Convert(time,update column) UNIX timestamps( in milliseconds )to timestamp(Convert milliseconds to seconds and then to readable timestamp)  
2) Generate column “area” - based on existing “place” column  
3) Create a new column 'geometry' with the desired structure   
4) Add insert date column

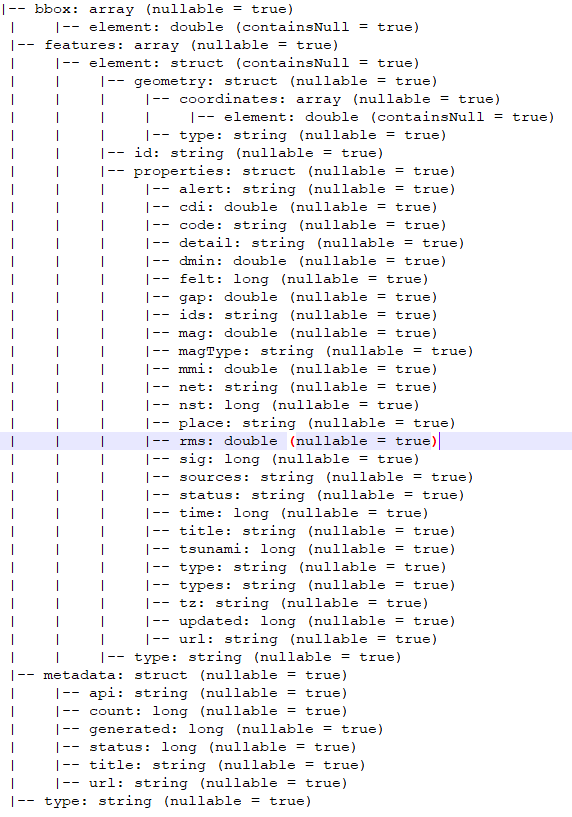
* **Output**:

DataFrame: A transformed DataFrame with flattened, cleaned, and formatted data.

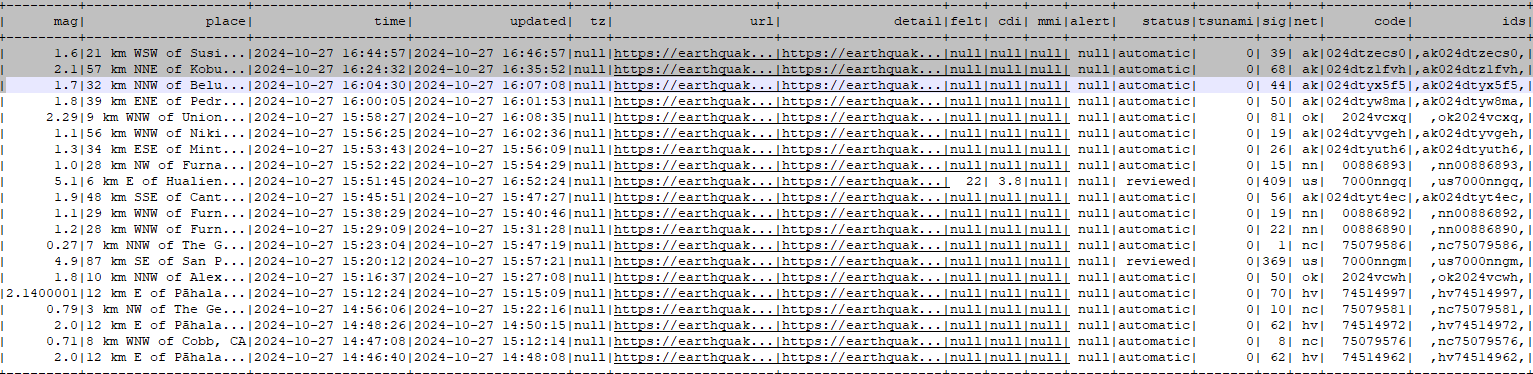
* **Sample input data :**

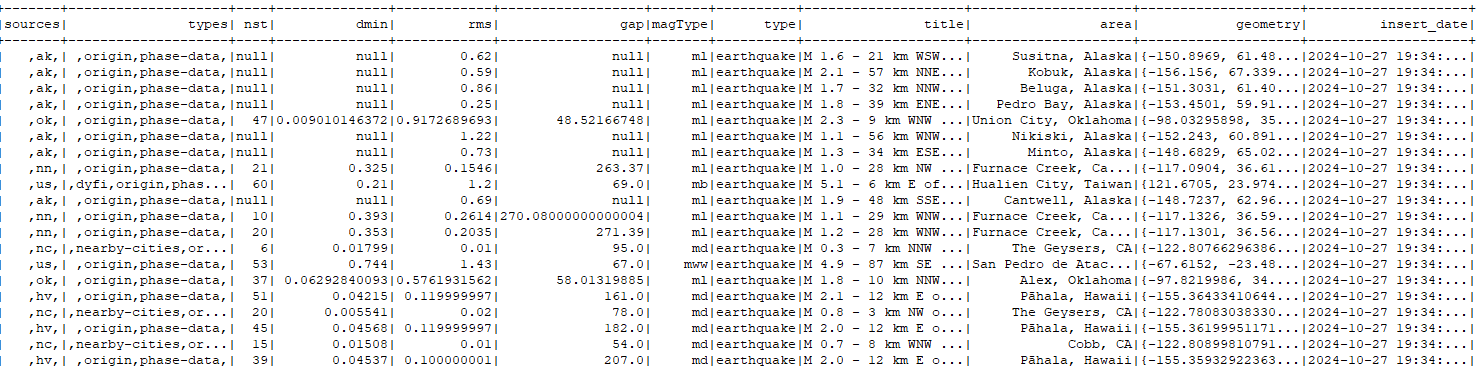


* **input\_schema**

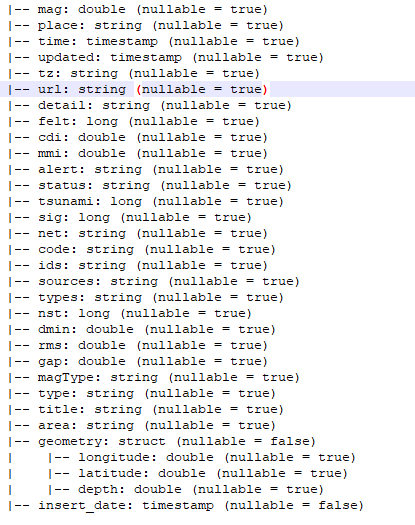


* **Sample output data :**





* **Output schema**



**Task: 5 Write Data to GCS as Parquet**

* **Description**: Writes clean data as Parquet file to a GCS location(silver location)
* **Flow Diagram** :

Write

**Dataframe**

**(Clean Data)**

**GCS**

**(Silver loc.parquet)**

* **Input**:

Dataframe : Clean data dataframe

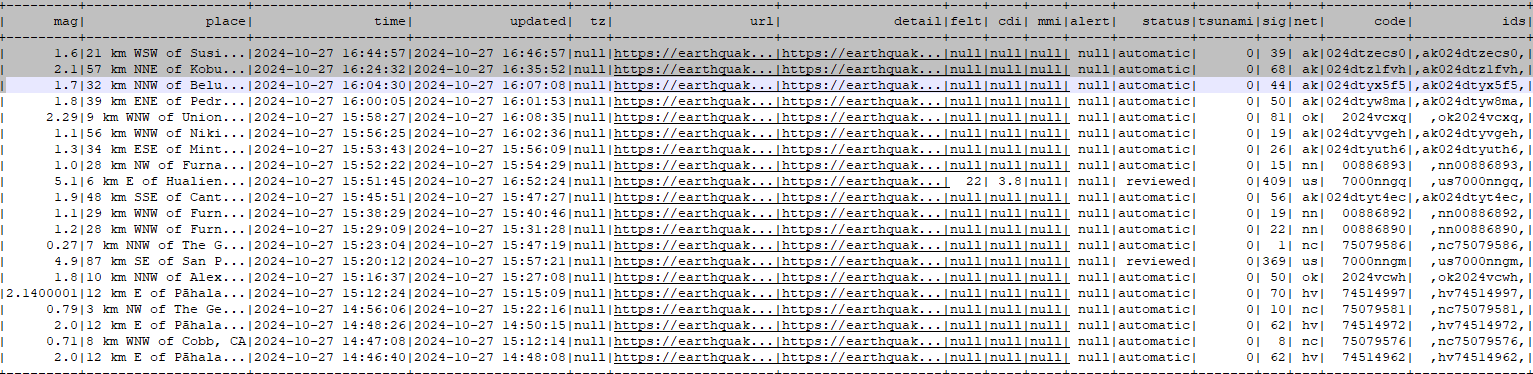
GCS\_location (string): GCS path for output.(silver loc.)

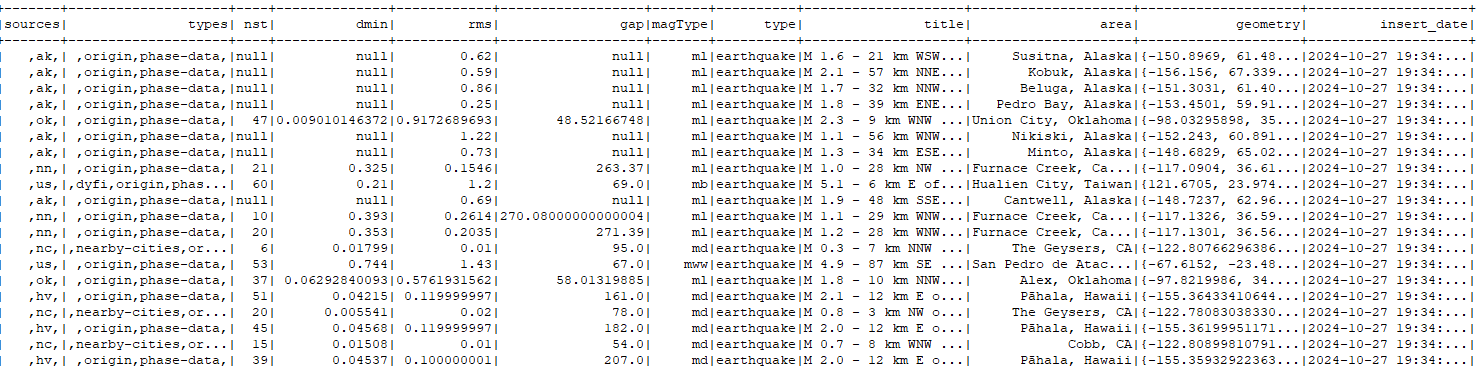
* **Output**:

Return None

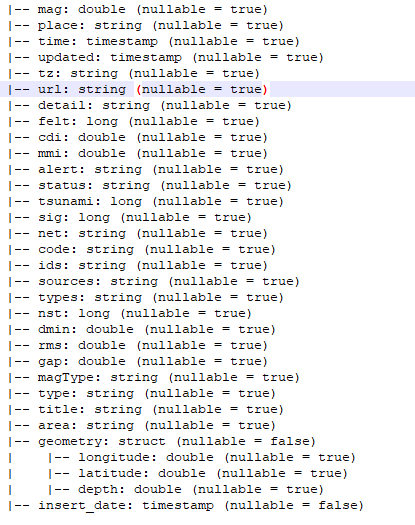
Write data to GCS ( silver loc.***'gs://earthquake\_analysis\_buck/pysaprk/silver/{cur\_timestamp}/'***)

* **Sample input data :**





* **input schema**



**Task: 6 Write Data into BigQuery**

* **Description**: Writes clean data to a specified BigQuery table.
* **Flow Diagram** :

write

**Dataframe**

(Clean Data)

**Bigquery** **table**

* **Input**:

output\_db (string): BigQuery table path (in the format project\_id.dataset\_id.table\_id.)

data\_df (DataFrame): The DataFrame containing the data to be written.

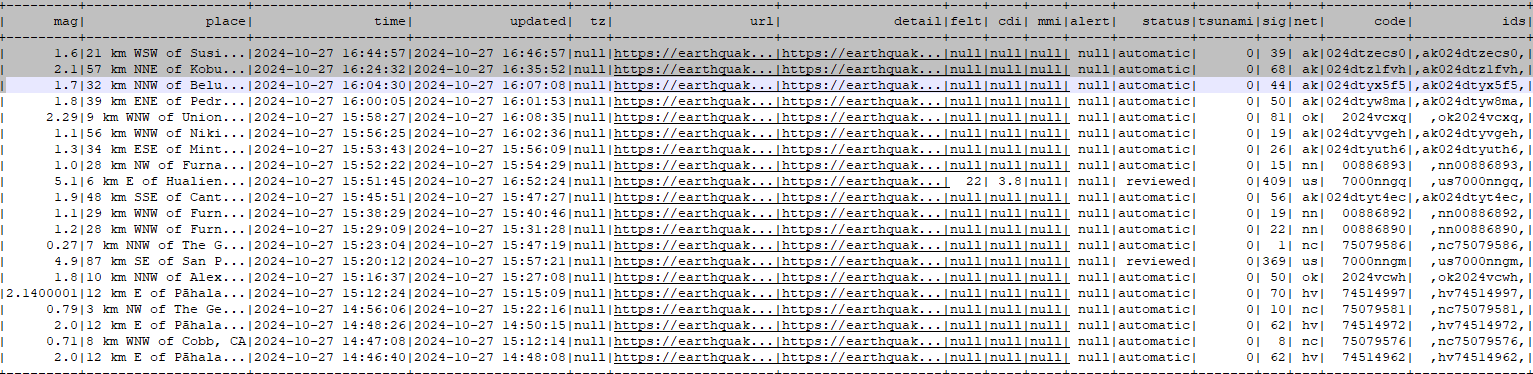
bq\_schema : Explicitly pass bigquery schema

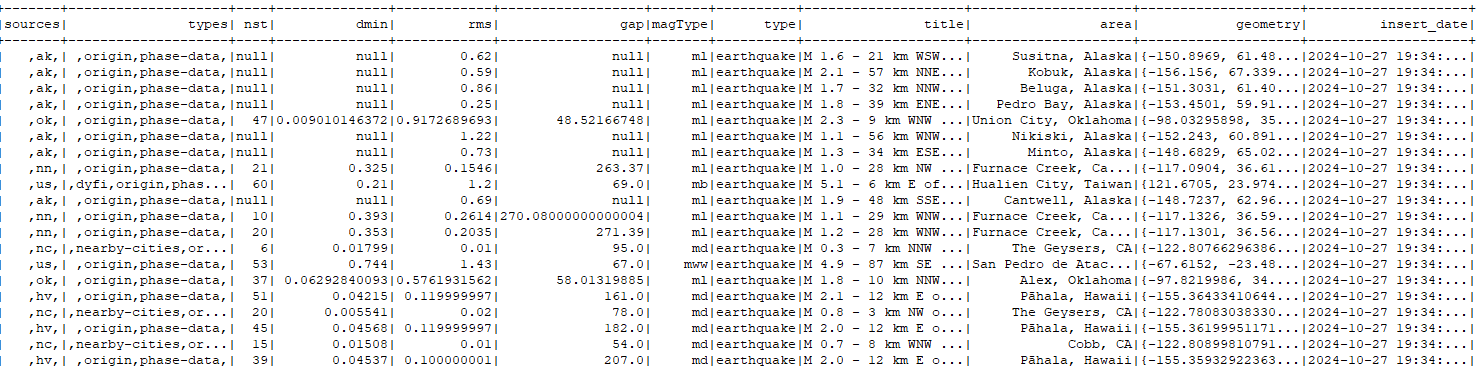
* **Output**:

Return None

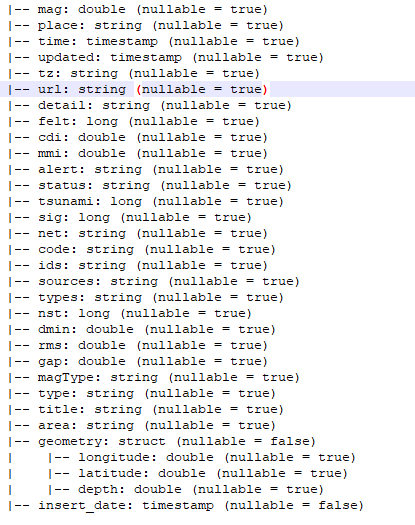
Write clean data in bigquery ('spark-learning-43150.earthquake\_db.earthquake\_data')

* **Sample input data :**

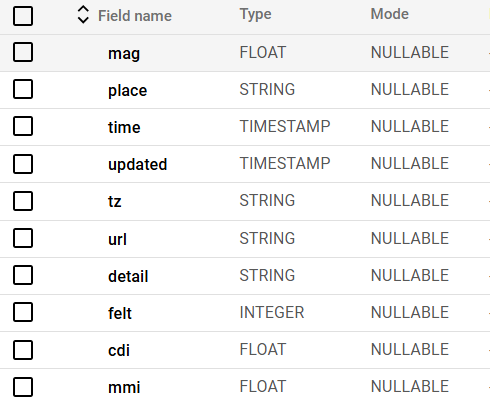


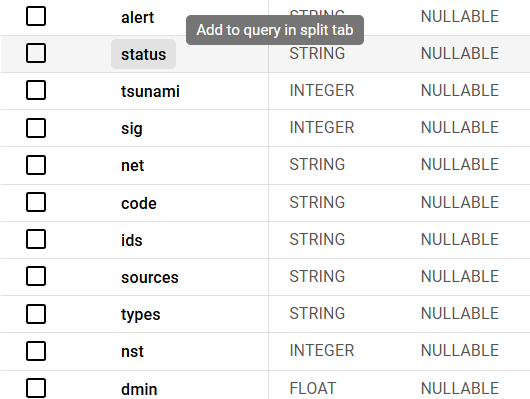


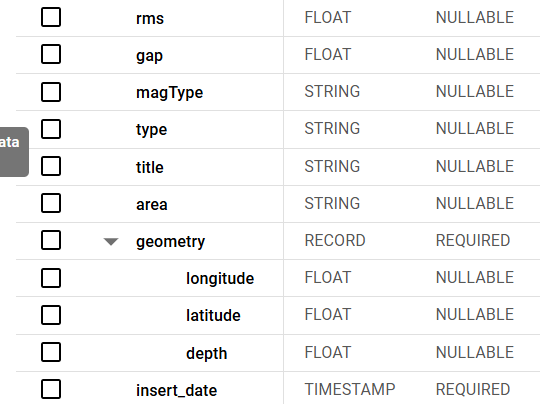
* **input schema**



* **Sample output data :**

****





* **output schema**

**#Audit table:**

**Case 1: Successful Execution (No Error)**

To maintain the audit table, log the following:

* **Function Name**, **Start Time**, **End Time**, **Status**, **Records Processed**, **Error Message**

**Steps**:

1. Define the function name and capture the **start time**.
2. Call the extractAllData function with the API URL parameters.
3. If successful, create the DataFrame.
4. Capture the **end time** and set the **status** to "success."
5. Use df.count() to get **records processed**.
6. Call the log\_audit function with parameters:  
   log\_audit(spark, job\_id, pipeline\_name, function\_name, start\_time, end\_time, status, processed\_record, cnf.eq\_audit\_tbl\_loc, error\_msg=None)

**Case 2: Execution with Error**

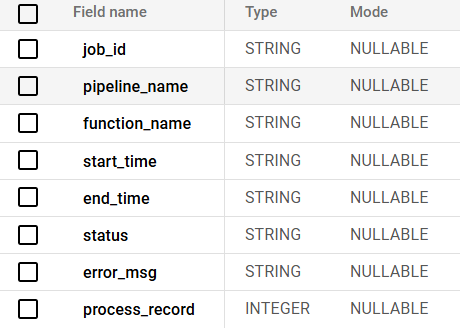
If an error occurs:

* **Function Name**, **Start Time**, **End Time**, **Status**, **Records Processed**, **Error Message**

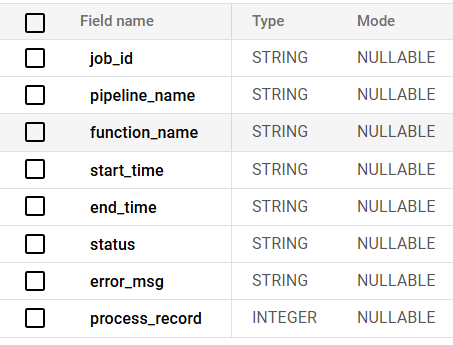
**Steps**:

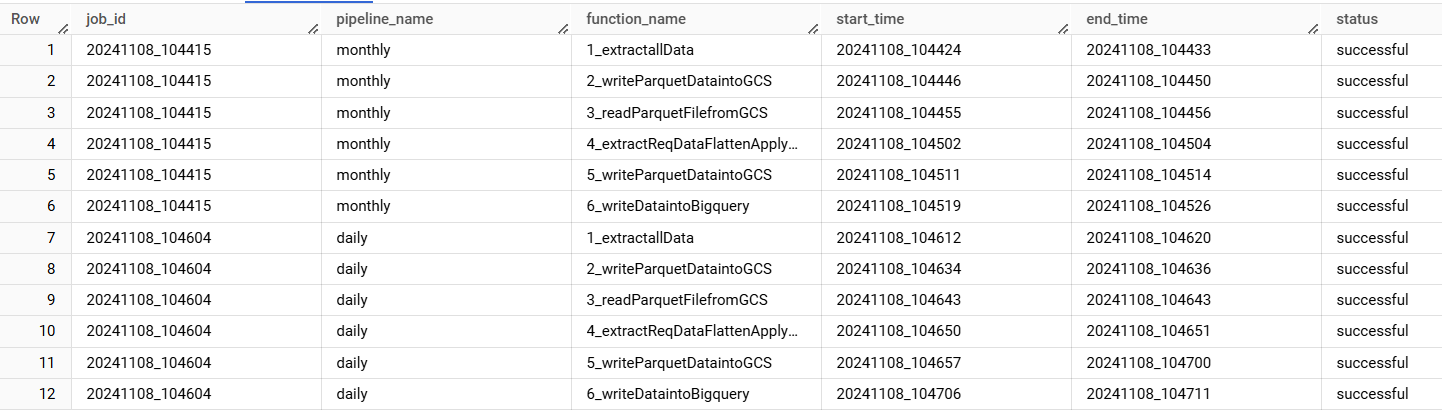
1. Define the function name and capture the **start time**.
2. Call the function.
3. If an error occurs, capture the **end time**, set **status** to "failed," and log the **error message**.
4. Call the log\_audit function with parameters:  
   log\_audit(spark, job\_id, pipeline\_name, function\_name, start\_time, end\_time, status, processed\_record, cnf.eq\_audit\_tbl\_loc, error\_msg)

**# audit table schema**



**#Audit\_table\_data:**







**# Automation using Cloud Composer**

* **Description**: DAG automates daily data processing for earthquake data, loading it into BigQuery and GCS(silver loc )with managed resource handling on Google Cloud.
* **Dag performs the following steps:**

1. **Create Dataproc Cluster(DataprocCreateClusterOperator)**: Initializes a Dataproc cluster for processing.
2. **Submit PySpark Job(DataprocSubmitJobOperator)**: Runs a PySpark job to retrieve earthquake data and load it into BigQuery.
3. **Delete Dataproc Cluster(DataprocDeleteClusterOperator)**: Deletes the cluster after the job completes to free up resources.

* **DAG Schedule**: Daily at 10:00 AM UTC(**10:00 AM UTC + 5 hours 30 minutes = 3:30 PM IST)**

**Task Dependencies**:

* create\_dataproc\_cluster >> submit\_pyspark\_job >> delete\_dataproc\_cluster

**# Optimization Techniques:**

**1)** **Utility Module:** Reusable functions to keep code clean.

**2) Logging Configuration**: Added logging to track the program’s actions and debugging..

**3) Dynamic Command-Line Arguments**: Used argparse to allow flexible input of API URL and pipeline name.

4) **Parquet Format for Storage:** Fast ,Reading and writing data in Parquet format for efficient storage and retrieval.

**5) Coalesce**: Reduces data partitions for faster saving.

**6) Explicit Schema for BigQuery**: pass explicit schema to avoid errors and make data loading faster.

**# Earthquake Data Analysis**

* **Description**: Read data from Bigquery table for Analysis
* **Flow Diagram** :

Read

**Bigquery table**

**(Clean Data)**

**Perform Analysis**

* **Input**:

**BigQuery table** :Read data from Bigquery Table ( ***'spark-learning-43150.earthquake\_db.earthquake\_data'***)

* **Output**:

**Analysis Results** : Various Output based on the analysis queries