

Seismic Signals: Tracking Global Earthquakes with Microsoft Fabric and Power BI

Initializing the Fabric Environment and Lakehouse

I began by setting up a Microsoft Fabric workspace and provisioning a Lakehouse to manage different layers of the data pipeline. This environment serves as the foundation for ingesting, storing, and processing earthquake data efficiently.

Fabrics_Pavas_Project

Create deployment pipelineCreate appManage accessWorkspace settings

+ New itemNew folderImport

Filter by keywordFilter

Choose from predesigned task flows or add a task to build one (preview)

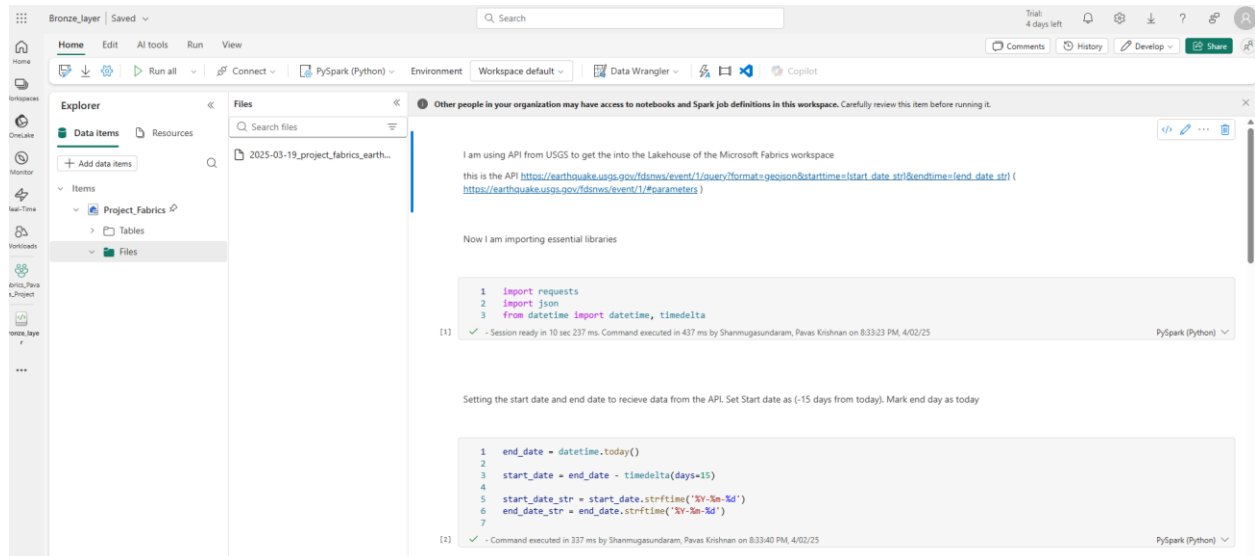
Name	Type	Task	Owner	Refreshed	Next refresh	Endorsement	Sensitivity	Included in app
Bronze_layer	Notebook	—	Pavas Krishnan Sha...	—	—	—	—	
Earthquake	Report	—	Fabrics_Pavas_Proj...	4/2/2025, 5:40:35 PM	—	—	—	No
fabrics_env	Environment	—	Pavas Krishnan Sha...	—	—	—	—	
Gold Layer	Notebook	—	Pavas Krishnan Sha...	—	—	—	—	
Project_Fabrics	Lakehouse	—	Pavas Krishnan Sha...	—	—	—	—	
Project_Fabrics	Semantic model (d...	—	Fabrics_Pavas_Proj...	4/2/2025, 5:40:35 PM	N/A	—	—	
Project_Fabrics	SQL analytics endp...	—	Pavas Krishnan Sha...	—	—	—	—	
Silver layer	Notebook	—	Pavas Krishnan Sha...	—	—	—	—	

Reviewing the USGS Earthquake API

The primary data source for this project is the **USGS Earthquake Catalog API**, a rich repository of global seismic activity. I explored the API’s documentation thoroughly to understand its structure, available filters, and response formats. This allowed me to tailor the data extraction process effectively for my use case.

For this project, I focused on the date range **from March 19th, 2025 to April 4th, 2025**. This specific window was chosen because several **significant earthquakes occurred in the**

Myanmar-Thailand region during this period. The intent behind selecting this timeframe was to raise awareness about seismic risks in Southeast Asia and provide insights to help audiences better understand the patterns and impact of these natural events.



Leveraging the Python integration within Fabric notebooks, I pulled data directly from the USGS API. Using the requests library, I fetched the earthquake data and saved it as a JSON file in the Lakehouse. This file became the base "Bronze" layer containing unprocessed raw data.

Silver layer | Saved

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Explorer

Data Items Resources

+ Add data items

Items

Project_Fabrics

Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

```
1 from pyspark.sql.functions import col
2 from pyspark.sql.types import TimestampType
3
4 [4] ✓ - Command executed in 293 ms by Shanmugasundaram, Pavas Krishnan on 8:51:59 PM, 4/02/25
5 ...
```

```
1 df = spark.read.option("multiline", "true").json("Files/2025-03-19_project_fabrics_earthquake_data.json")
2 # df now is a Spark DataFrame containing 3500 data from "Files/2025-03-19_project_fabrics_earthquake_data.json".
3 display(df)
```

[4] ✓ - Session ready in 10 sec 119 ms. Command executed in 4 sec 905 ms by Shanmugasundaram, Pavas Krishnan on 8:46:13 PM, 4/02/25

Table view

	geo geometry	ABC id	geo properties	ABC type
1	("coordinates"...	c40914927	("tsunami"0,"p...	Feature
2	("coordinates"...	ak02548jg...	("tsunami"0,"p...	Feature
3	("coordinates"...	c40914919	("tsunami"0,"p...	Feature
4	("coordinates"...	c40914911	("tsunami"0,"p...	Feature
5	("coordinates"...	nc75158977	("tsunami"0,"p...	Feature
6	("coordinates"...	ak02548jcx...	("tsunami"0,"p...	Feature
7	("coordinates"...	ak02548jcp...	("tsunami"0,"p...	Feature
8	("coordinates"...	c40914903	("tsunami"0,"p...	Feature
9	("coordinates"...	nc75158967	("tsunami"0,"p...	Feature
10	("coordinates"...	ak02548jab...	("tsunami"0,"p...	Feature
11	("coordinates"...	c40914895	("tsunami"0,"p...	Feature
12	("coordinates"...	nc75158962	("tsunami"0,"p...	Feature

I saved the output of the bronze layer in file format inside the Lakehouse.

Bronze_layer | Saved

Home Edit AI tools Run View

PySpark (Python) Environment Workspace default Data Wrangler Copilot

Explorer

Data Items Resources

+ Add data items

Items

Project_Fabrics

Files

2025-03-19_project_fabrics_earth...

Data successfully saved to /lakehouse/default/Files/2025-03-19_project_fabrics_earthquake_data.json

Now I load the data which is present in lakehouse file

```
1 df = spark.read.option("multiline", "true").json("Files/2025-03-19_project_fabrics_earthquake_data.json")
2 # df now is a Spark DataFrame containing 3500 data from "Files/2025-03-19_project_fabrics_earthquake_data.json".
3 display(df)
```

[4] ✓ - Command executed in 8 sec 27 ms by Shanmugasundaram, Pavas Krishnan on 8:42:14 PM, 4/02/25

Table view

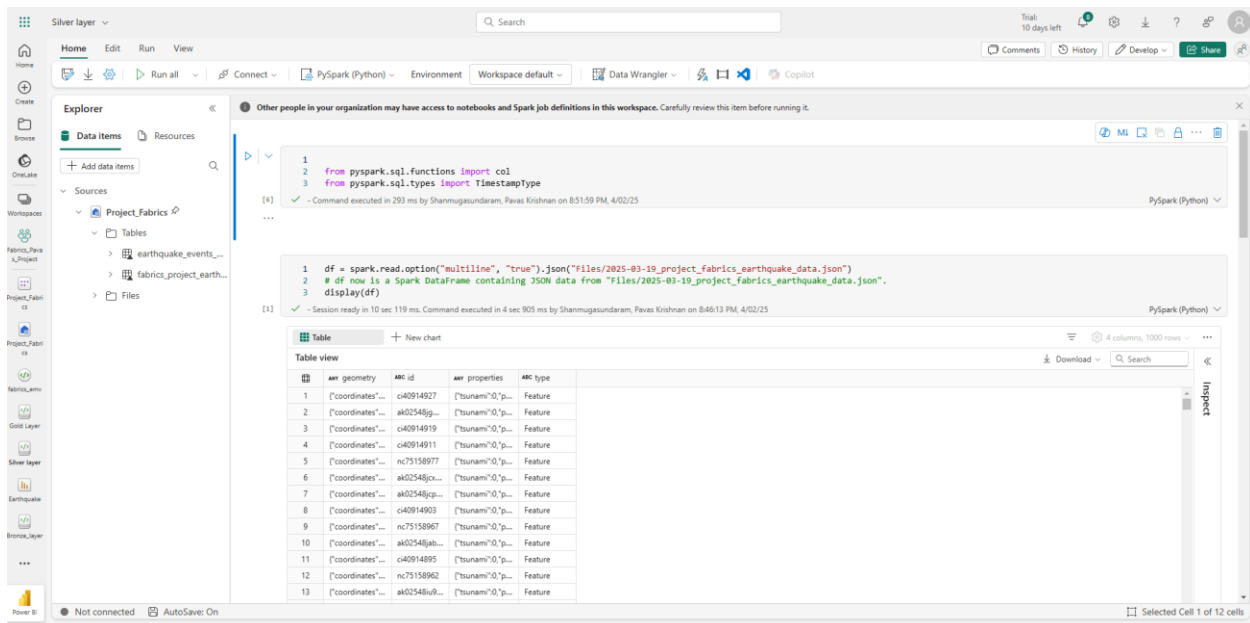
	geo geometry	ABC id	geo properties	ABC type
1	("coordinates"...	c40914927	("tsunami"0,"p...	Feature
2	("coordinates"...	ak02548jg...	("tsunami"0,"p...	Feature
3	("coordinates"...	c40914919	("tsunami"0,"p...	Feature
4	("coordinates"...	c40914911	("tsunami"0,"p...	Feature
5	("coordinates"...	nc75158977	("tsunami"0,"p...	Feature
6	("coordinates"...	ak02548jcx...	("tsunami"0,"p...	Feature
7	("coordinates"...	ak02548jcp...	("tsunami"0,"p...	Feature
8	("coordinates"...	c40914903	("tsunami"0,"p...	Feature
9	("coordinates"...	nc75158967	("tsunami"0,"p...	Feature
10	("coordinates"...	ak02548jab...	("tsunami"0,"p...	Feature
11	("coordinates"...	c40914895	("tsunami"0,"p...	Feature
12	("coordinates"...	nc75158962	("tsunami"0,"p...	Feature
13	("coordinates"...	ak02548j9...	("tsunami"0,"p...	Feature
14	("coordinates"...	bx2025gnab	("tsunami"0,"p...	Feature
15	("coordinates"...	nc75158952	("tsunami"0,"p...	Feature

Cleaning and Structuring in the Silver Layer

In this step, I refined the raw data to prepare it for analysis. The transformations included:

Selective Feature Extraction: Only essential fields were retained from the raw JSON structure.

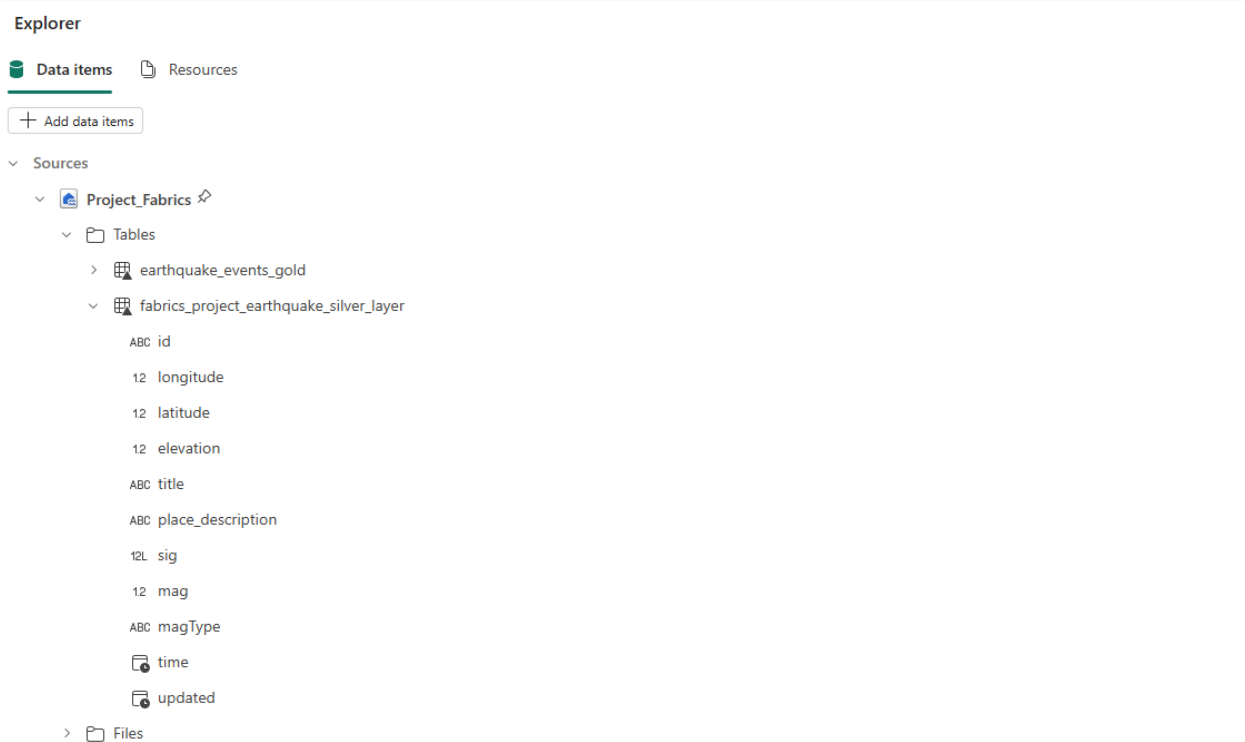
Time Conversion: The original timestamps were in Unix time (milliseconds). I converted them into standard readable datetime format.



The screenshot displays a Databricks workspace interface. The left sidebar shows the 'Explorer' panel with a tree view of the workspace structure, including 'Data Items', 'Resources', and 'Sources'. The main area shows a PySpark notebook with two code cells. The first cell imports necessary functions and types. The second cell reads a JSON file and displays the resulting DataFrame. Below the code, a 'Table view' is shown, displaying a table with 13 rows and 4 columns: 'geo geometry', 'geo id', 'geo properties', and 'geo type'.

	geo geometry	geo id	geo properties	geo type
1	("coordinates"...	c40914927	("tsunami":0,"p...	Feature
2	("coordinates"...	ak02548gg...	("tsunami":0,"p...	Feature
3	("coordinates"...	c40914919	("tsunami":0,"p...	Feature
4	("coordinates"...	c40914911	("tsunami":0,"p...	Feature
5	("coordinates"...	nc75158977	("tsunami":0,"p...	Feature
6	("coordinates"...	ak02548jcx...	("tsunami":0,"p...	Feature
7	("coordinates"...	ak02548jcp...	("tsunami":0,"p...	Feature
8	("coordinates"...	c40914903	("tsunami":0,"p...	Feature
9	("coordinates"...	nc75158967	("tsunami":0,"p...	Feature
10	("coordinates"...	ak02548jab...	("tsunami":0,"p...	Feature
11	("coordinates"...	c40914895	("tsunami":0,"p...	Feature
12	("coordinates"...	nc75158962	("tsunami":0,"p...	Feature
13	("coordinates"...	ak02548iua...	("tsunami":0,"p...	Feature

Incremental Loading: The cleaned dataset was appended to a table format using append mode, enabling future incremental updates.



Enriching the Dataset in the Gold Layer

To provide more analytical value, I enriched the silver data in the Gold layer with additional context:

- **Geolocation Mapping:** By using the Reverse Geocoder library, I translated latitude and longitude coordinates into country codes to localize each event.
- **Significance Grouping:** Earthquake significance scores were categorized into three levels—low, moderate, and high—facilitating easier visual analysis

Gold Layer

Home Edit Run View

PySpark (Python) Environment fabrics_env Data Wrangler Copilot

Explorer

Data Items Resources

+ Add data items

Sources

Project_Fabrics

earthquake_events_gold

ABC id

12 longitude

12 latitude

12 elevation

ABC title

ABC place_description

12 sig

12 mag

ABC magType

time

updated

ABC country_code

ABC sig_class

fabrics_project_earthquake_silver_layer

Files

```
16 coordinates = (float(lat), float(lon))
17 return rg.search(coordinates)[0].get('cc')
```

[4] ✓ - Command executed in 289 ms by Shanmugasundaram, Pavas Krishnan on 9:19:03 PM, 4/02/25 PySpark (Python)

```
1 get_country_code_udf = udf(get_country_code, StringType())
```

[5] ✓ - Command executed in 297 ms by Shanmugasundaram, Pavas Krishnan on 9:19:17 PM, 4/02/25 PySpark (Python)

```
1 df_with_location = \
2   df.\
3   withColumn("country_code", get_country_code_udf(col("latitude"), col("longitude")))
```

[6] ✓ - Command executed in 307 ms by Shanmugasundaram, Pavas Krishnan on 9:19:37 PM, 4/02/25 PySpark (Python)

```
1 df_with_location_sig_class = \
2   df_with_location.\
3   withColumn("sig_class",
4             when(col("sig") < 100, "Low").\
5             when((col("sig") >= 100) & (col("sig") < 500), "Moderate").\
6             otherwise("High")
7             )
```

[7] ✓ - Command executed in 284 ms by Shanmugasundaram, Pavas Krishnan on 9:19:49 PM, 4/02/25 PySpark (Python)

```
1 df_with_location_sig_class.write.mode("append").saveAsTable("earthquake_events_gold")
```

[8] ✓ - Command executed in 5 min 15 sec 115 ms by Shanmugasundaram, Pavas Krishnan on 9:25:40 PM, 4/02/25 PySpark (Python)

Gold Layer Saved

Home Edit AI tools Run View

PySpark (Python) Environment fabrics_env Data Wrangler Copilot

Explorer

Data Items Resources

+ Add data items

Items

Project_Fabrics

fabrics_env

Runtime: 1.3 (Spark 3.3, Delta 3.2)

Compute: Medium, 1-10 nodes

Change to workspace default

Change environment

New environment

```
1 df_with_location = \
2   df.\
3   withColumn("country_code", get_country_code_udf(col("latitude"), col("longitude")))
```

[4] ✓ - Command executed in 307 ms by Shanmugasundaram, Pavas Krishnan on 9:19:37 PM, 4/02/25 PySpark (Python)

```
1 df_with_location_sig_class = \
2   df_with_location.\
3   withColumn("sig_class",
4             when(col("sig") < 100, "Low").\
5             when((col("sig") >= 100) & (col("sig") < 500), "Moderate").\
6             otherwise("High")
7             )
```

[7] ✓ - Command executed in 284 ms by Shanmugasundaram, Pavas Krishnan on 9:19:49 PM, 4/02/25 PySpark (Python)

```
1 df_with_location_sig_class.write.mode("append").saveAsTable("earthquake_events_gold")
```

[8] ✓ - Command executed in 5 min 35 sec 113 ms by Shanmugasundaram, Pavas Krishnan on 9:25:40 PM, 4/02/25 PySpark (Python)

Explorer

Data items

Resources

+ Add data items

Sources

Project_Fabrics

Tables

earthquake_events_gold

id

longitude

latitude

elevation

title

place_description

sig

mag

magType

time

updated

country_code

sig class

Creating the Earthquake Visualization Report

Using Power BI, embedded within Microsoft Fabric, I built an interactive dashboard to display the processed earthquake data. Due to account restrictions (trial version) that limited the use of map visuals, I utilized bar charts to represent earthquake occurrences across time periods and significance categories.

Despite its simplicity, the dashboard effectively allows users to explore and filter global earthquake events by date and magnitude range.

