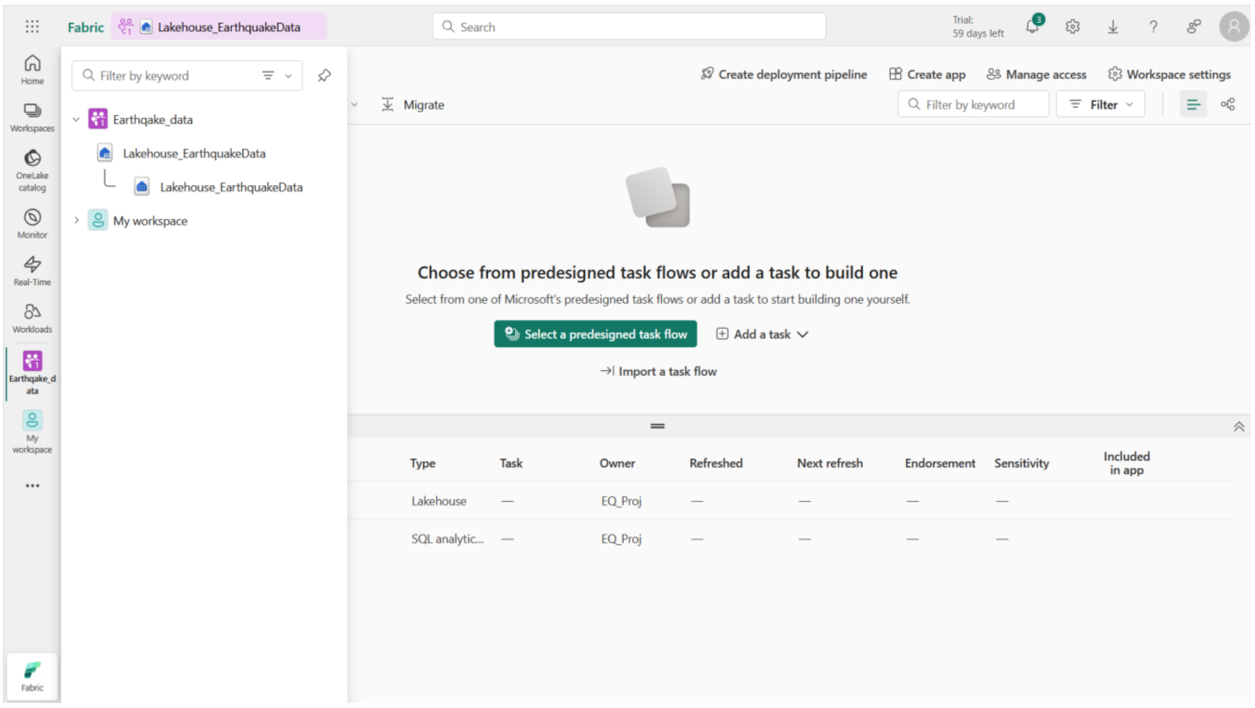
After creating the workspace and lakehouse

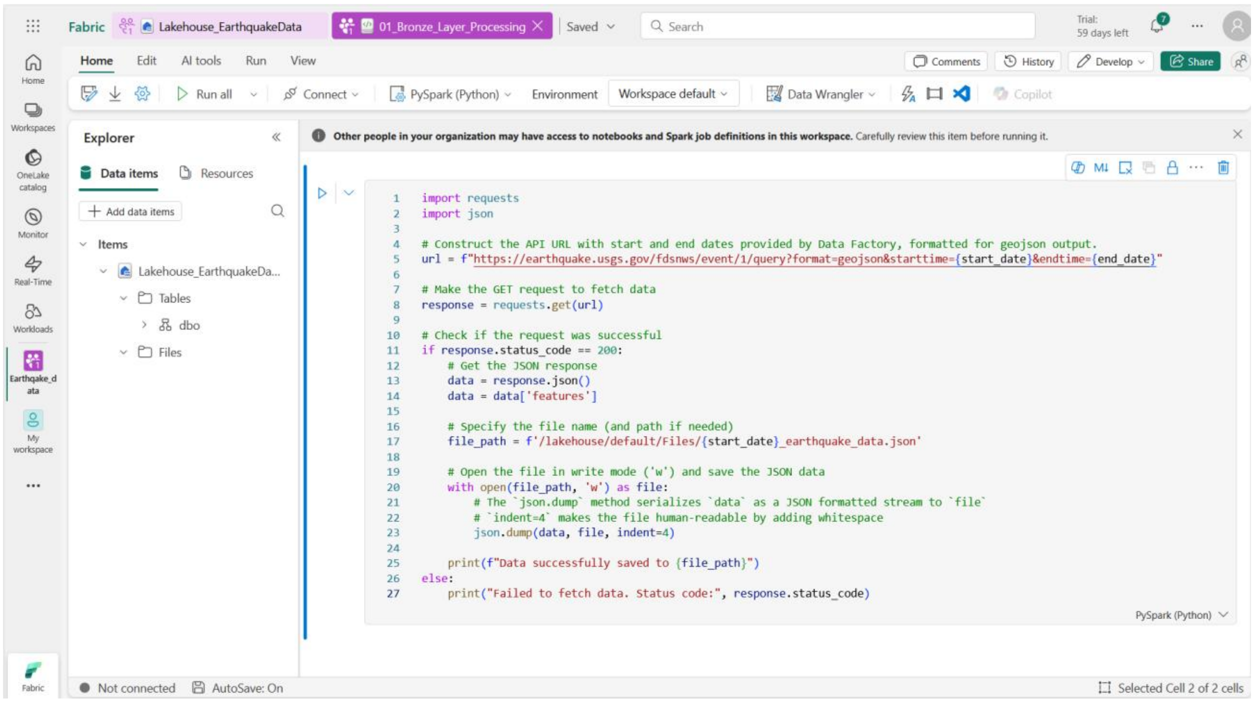


Type	Task	Owner	Refreshed	Next refresh	Endorsement	Sensitivity	Included in app
Lakehouse	—	EQ_Proj	—	—	—	—	
SQL analytic...	—	EQ_Proj	—	—	—	—	

Data Source:

https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&starttime={start_date}&endtime={end_date}

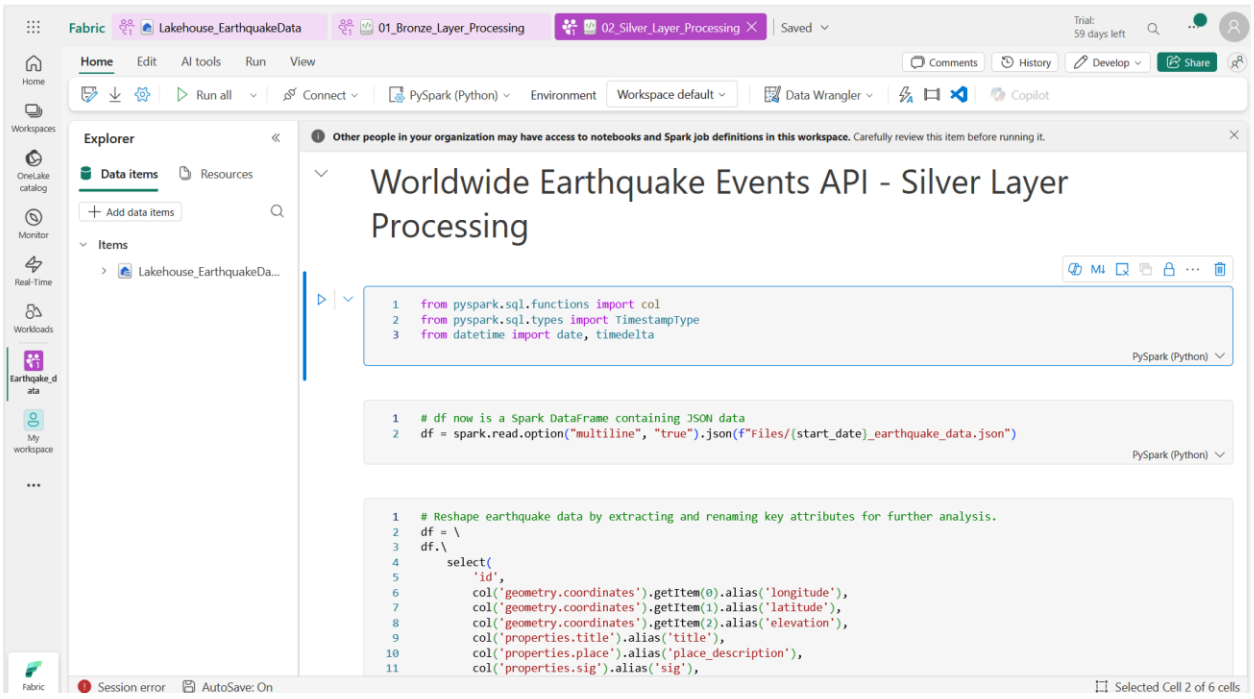
Bronze Notebook



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

```python
import requests
import json

# Construct the API URL with start and end dates provided by Data Factory, formatted for geojson output.
url = f"https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&starttime={start_date}&endtime={end_date}"

# Make the GET request to fetch data
response = requests.get(url)

# Check if the request was successful
if response.status_code == 200:
    # Get the JSON response
    data = response.json()
    data = data['features']

    # Specify the file name (and path if needed)
    file_path = f'/lakehouse/default/Files/{start_date}_earthquake_data.json'

    # Open the file in write mode ('w') and save the JSON data
    with open(file_path, 'w') as file:
        # The `json.dump` method serializes `data` as a JSON formatted stream to `file`
        # `indent=4` makes the file human-readable by adding whitespace
        json.dump(data, file, indent=4)

    print(f"Data successfully saved to {file_path}")
else:
    print("Failed to fetch data. Status code:", response.status_code)
```

Silver Notebook



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

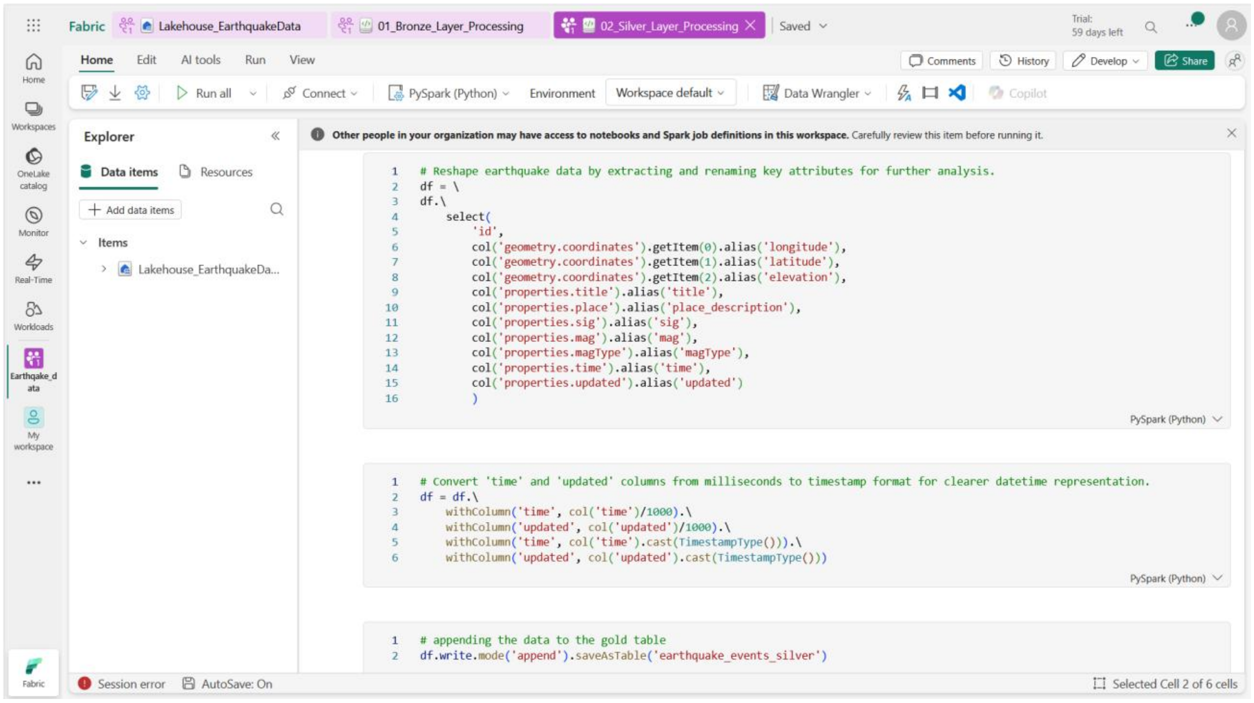
Worldwide Earthquake Events API - Silver Layer Processing

```python
from pyspark.sql.functions import col
from pyspark.sql.types import TimestampType
from datetime import date, timedelta
```

```python
# df now is a Spark DataFrame containing JSON data
df = spark.read.option("multiline", "true").json(f"Files/{start_date}_earthquake_data.json")
```

```python
# Reshape earthquake data by extracting and renaming key attributes for further analysis.
df = \
df.\
    select(
        'id',
        col('geometry.coordinates').getItem(0).alias('longitude'),
        col('geometry.coordinates').getItem(1).alias('latitude'),
        col('geometry.coordinates').getItem(2).alias('elevation'),
        col('properties.title').alias('title'),
        col('properties.place').alias('place_description'),
        col('properties.sig').alias('sig'),
```

Session error AutoSave: On Selected Cell 2 of 6 cells



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

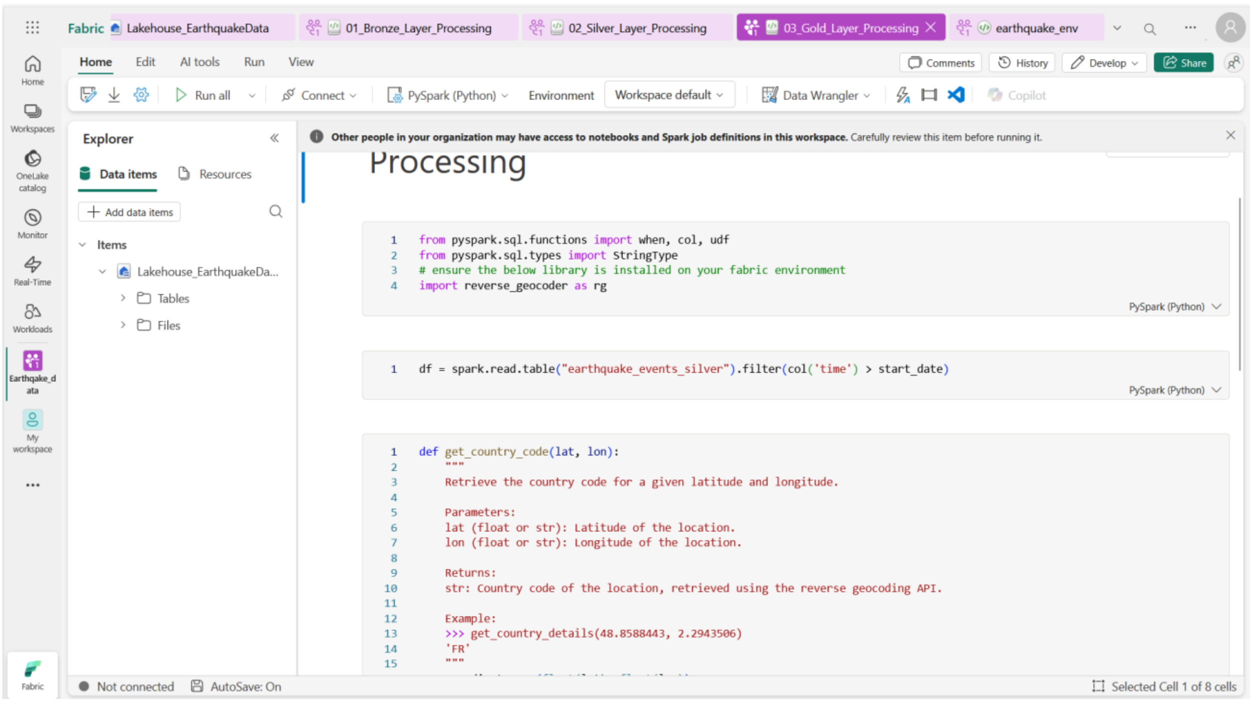
```python
# Reshape earthquake data by extracting and renaming key attributes for further analysis.
df = \
df.\
    select(
        'id',
        col('geometry.coordinates').getItem(0).alias('longitude'),
        col('geometry.coordinates').getItem(1).alias('latitude'),
        col('geometry.coordinates').getItem(2).alias('elevation'),
        col('properties.title').alias('title'),
        col('properties.place').alias('place_description'),
        col('properties.sig').alias('sig'),
        col('properties.mag').alias('mag'),
        col('properties.magType').alias('magType'),
        col('properties.time').alias('time'),
        col('properties.updated').alias('updated')
        )
```

```python
# Convert 'time' and 'updated' columns from milliseconds to timestamp format for clearer datetime representation.
df = df.\
    withColumn('time', col('time')/1000).\
    withColumn('updated', col('updated')/1000).\
    withColumn('time', col('time').cast(TimestampType())).\
    withColumn('updated', col('updated').cast(TimestampType()))
```

```python
# appending the data to the gold table
df.write.mode('append').saveAsTable('earthquake_events_silver')
```

Session error AutoSave: On Selected Cell 2 of 6 cells

Gold Notebook



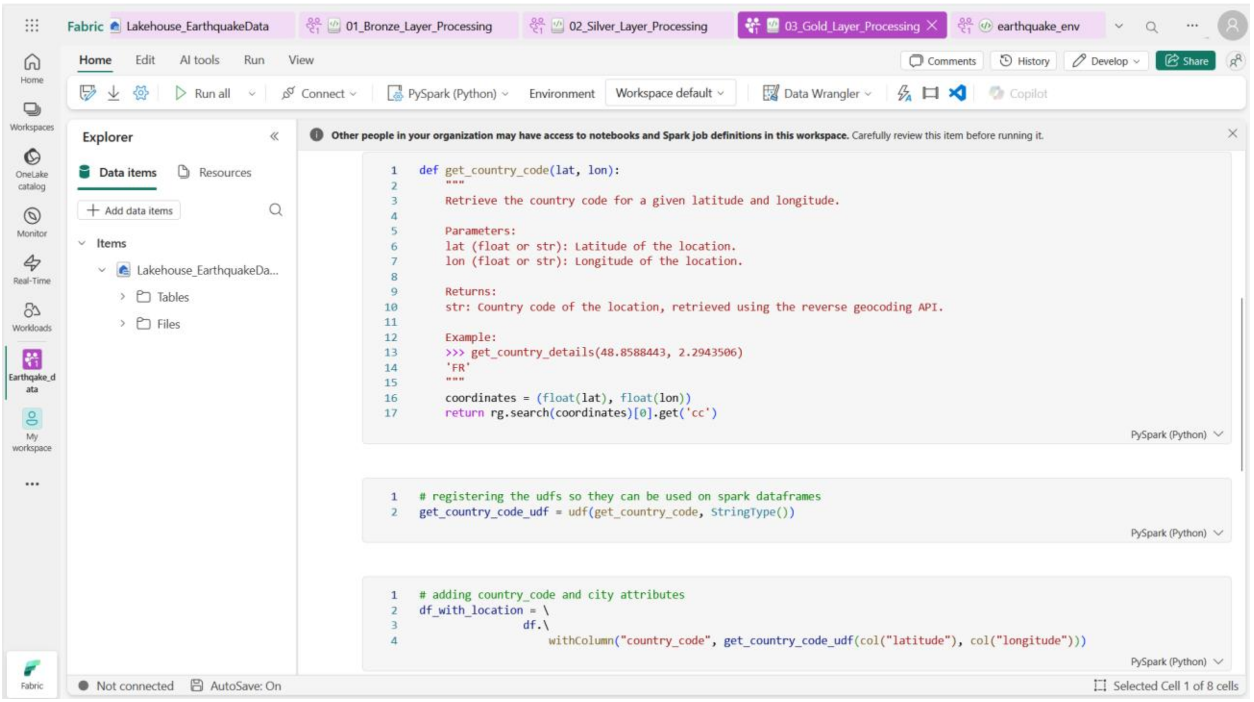
Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

Processing

```python
from pyspark.sql.functions import when, col, udf
from pyspark.sql.types import StringType
# ensure the below library is installed on your fabric environment
import reverse_geocoder as rg
```

```python
df = spark.read.table("earthquake_events_silver").filter(col('time') > start_date)
```

```python
def get_country_code(lat, lon):
    """
    Retrieve the country code for a given latitude and longitude.

    Parameters:
    lat (float or str): Latitude of the location.
    lon (float or str): Longitude of the location.

    Returns:
    str: Country code of the location, retrieved using the reverse geocoding API.

    Example:
    >>> get_country_details(48.8588443, 2.2943506)
    'FR'
    """
```

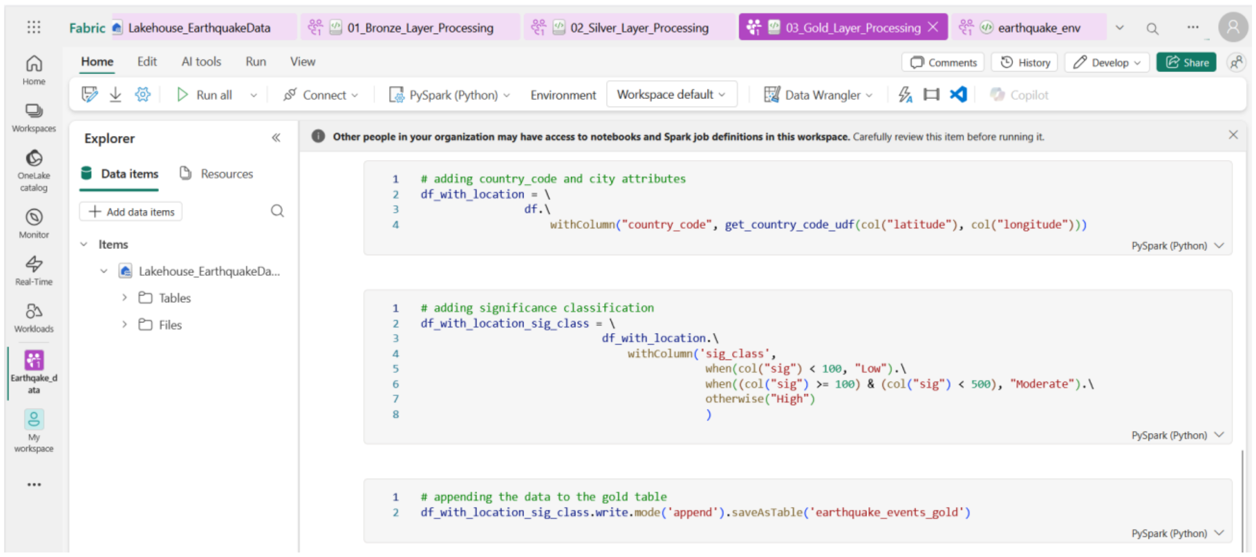


Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

```python
def get_country_code(lat, lon):
    """
    Retrieve the country code for a given latitude and longitude.

    Parameters:
    lat (float or str): Latitude of the location.
    lon (float or str): Longitude of the location.

    Returns:
    str: Country code of the location, retrieved using the reverse geocoding API.

    Example:
    >>> get_country_details(48.8588443, 2.2943506)
    'FR'
    """
    coordinates = (float(lat), float(lon))
    return rg.search(coordinates)[0].get('cc')
```

```python
# registering the udfs so they can be used on spark dataframes
get_country_code_udf = udf(get_country_code, StringType())
```

```python
# adding country_code and city attributes
df_with_location = \
            df.\
                withColumn("country_code", get_country_code_udf(col("latitude"), col("longitude")))
```



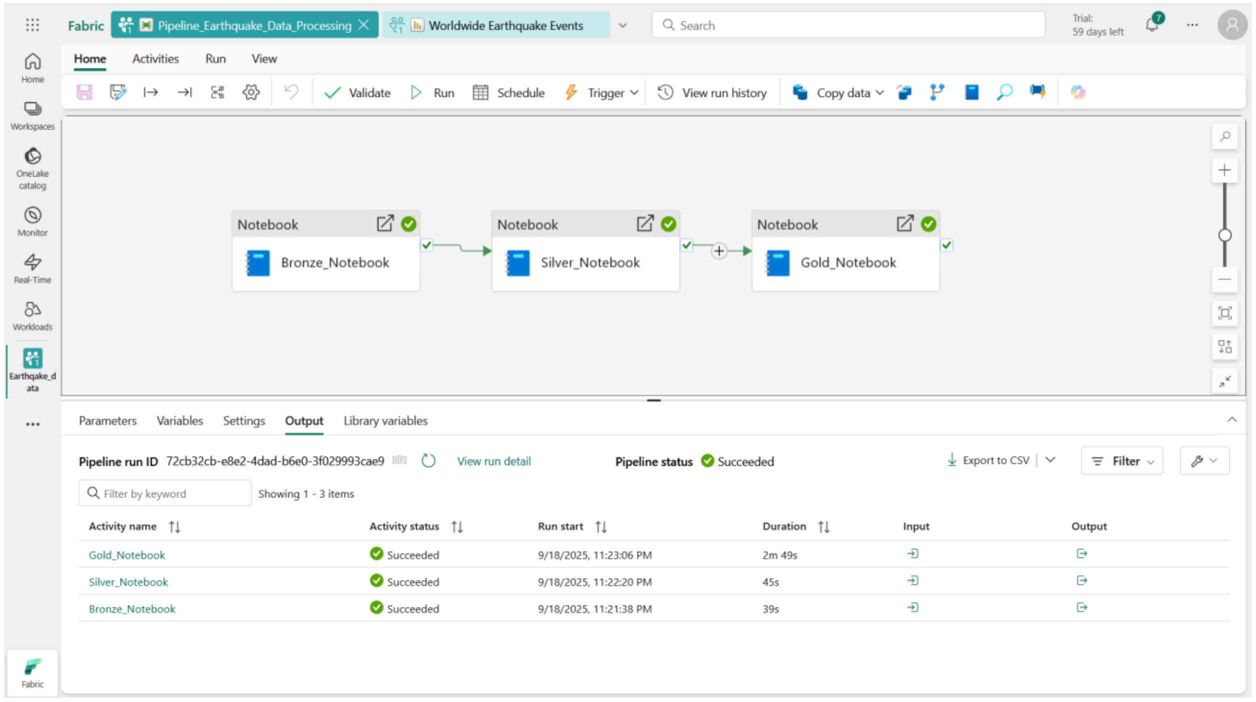
Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

```
# adding country_code and city attributes
df_with_location = \
                df.\
                    withColumn("country_code", get_country_code_udf(col("latitude"), col("longitude")))
```

```
# adding significance classification
df_with_location_sig_class = \
                            df_with_location.\
                                withColumn('sig_class',
                                            when(col("sig") < 100, "Low").\
                                            when((col("sig") >= 100) & (col("sig") < 500), "Moderate").\
                                            otherwise("High")
                                            )
```

```
# appending the data to the gold table
df_with_location_sig_class.write.mode('append').saveAsTable('earthquake_events_gold')
```

After Pipeline Run



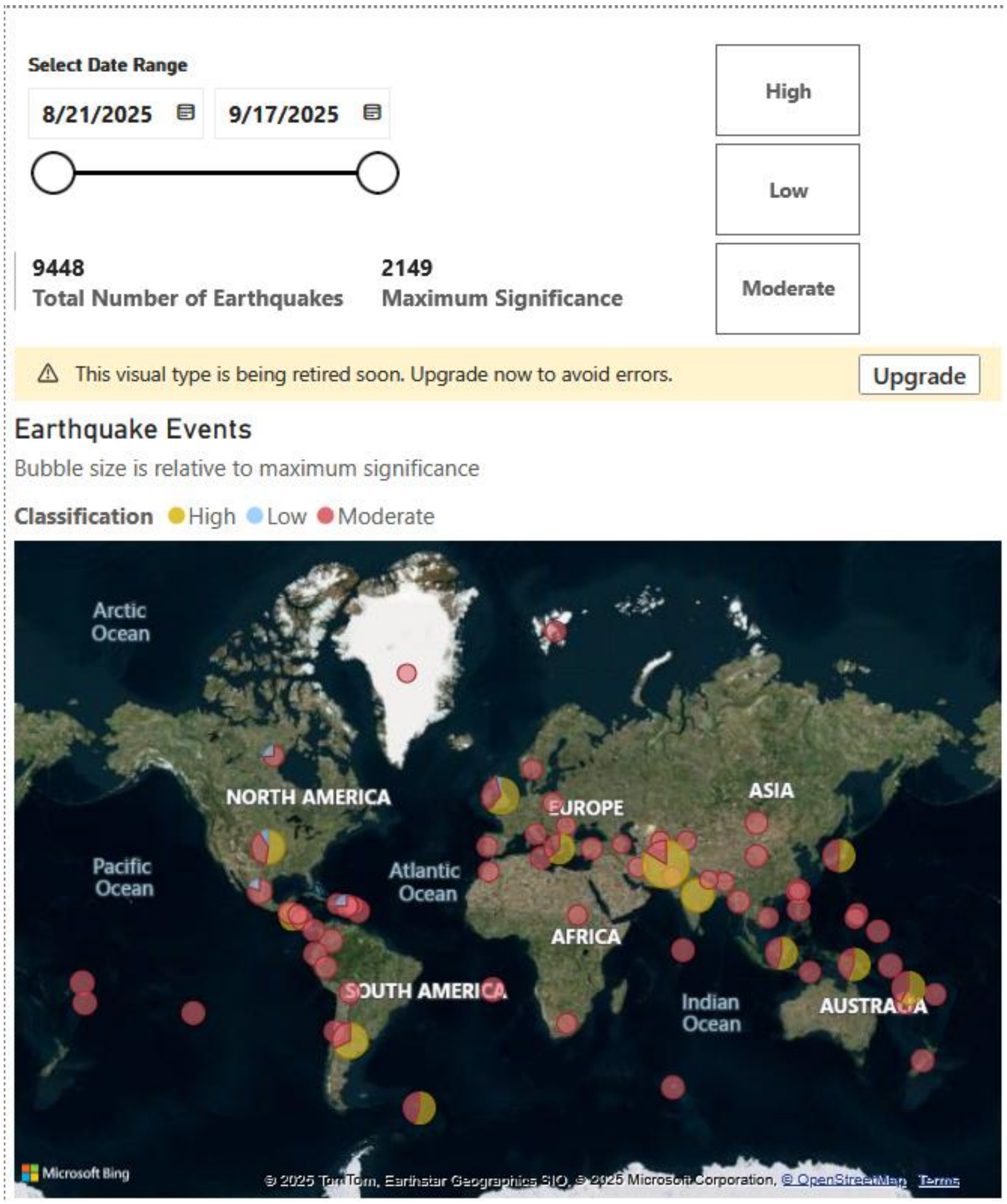
Pipeline run ID 72cb32cb-e8e2-4dad-b6e0-3f029993cae9 View run detail Pipeline status Succeeded

Export to CSV Filter

Filter by keyword Showing 1 - 3 items

Activity name	Activity status	Run start	Duration	Input	Output
Gold_Notebook	✓ Succeeded	9/18/2025, 11:23:06 PM	2m 49s		
Silver_Notebook	✓ Succeeded	9/18/2025, 11:22:20 PM	45s		
Bronze_Notebook	✓ Succeeded	9/18/2025, 11:21:38 PM	39s		

PowerBI Report Before Incremental Upload



Select Date Range

8/21/2025 — 9/17/2025

9448
Total Number of Earthquakes

2149
Maximum Significance

High

Low

Moderate

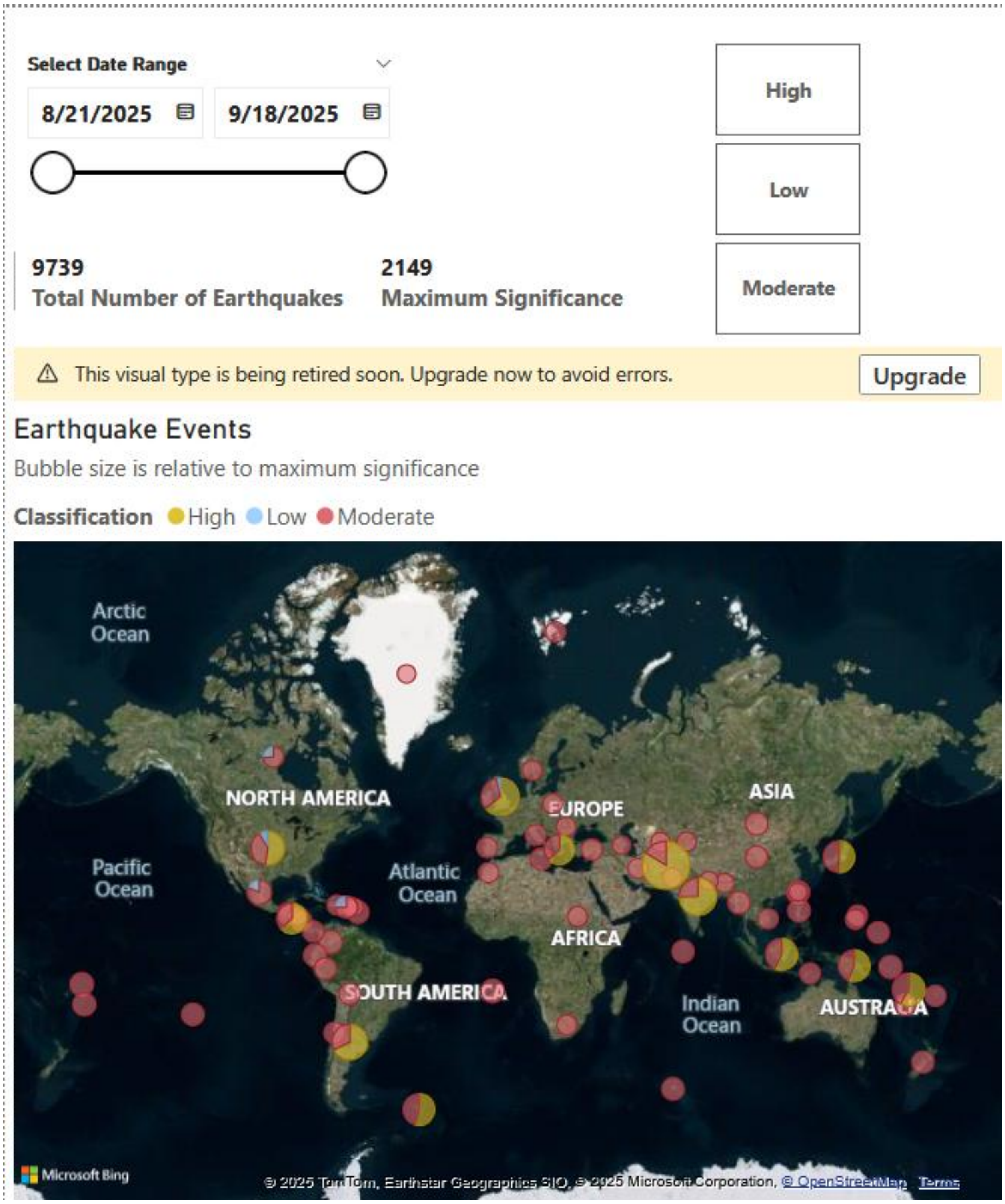
⚠ This visual type is being retired soon. Upgrade now to avoid errors. Upgrade

Earthquake Events

Bubble size is relative to maximum significance

Classification ● High ● Low ● Moderate

PowerBI Report After Incremental Upload



Select Date Range

8/21/2025 9/18/2025

High

Low

Moderate

9739
Total Number of Earthquakes

2149
Maximum Significance

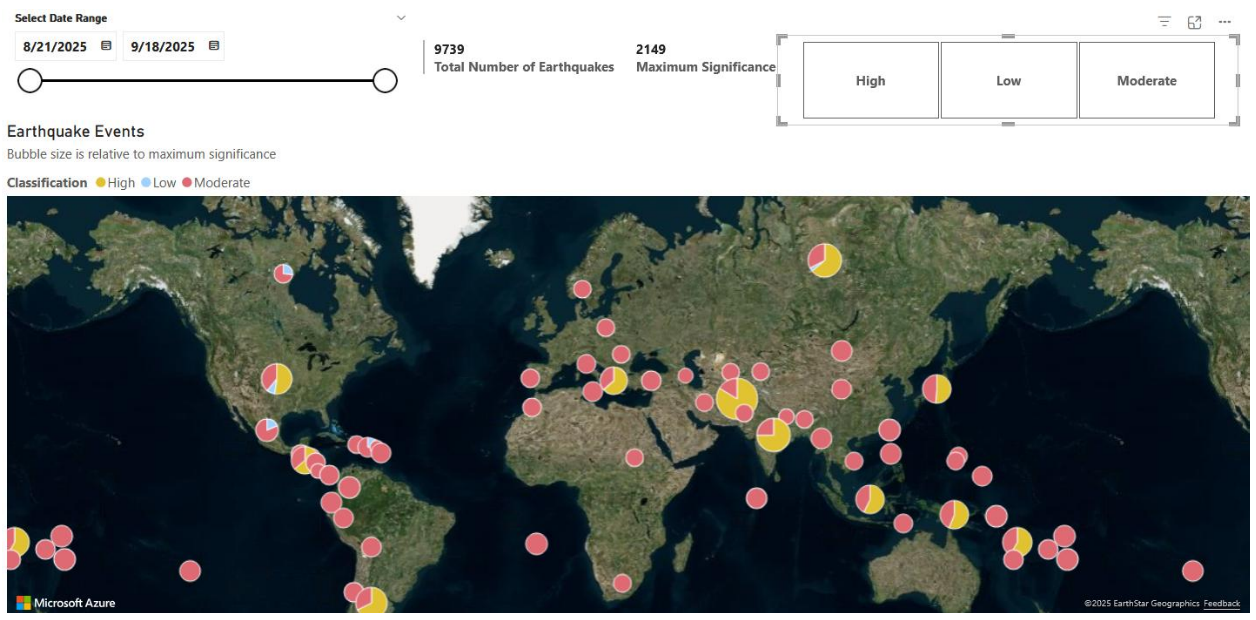
⚠ This visual type is being retired soon. Upgrade now to avoid errors. Upgrade

Earthquake Events

Bubble size is relative to maximum significance

Classification ● High ● Low ● Moderate

Final Report



9739 Total Number of Earthquakes

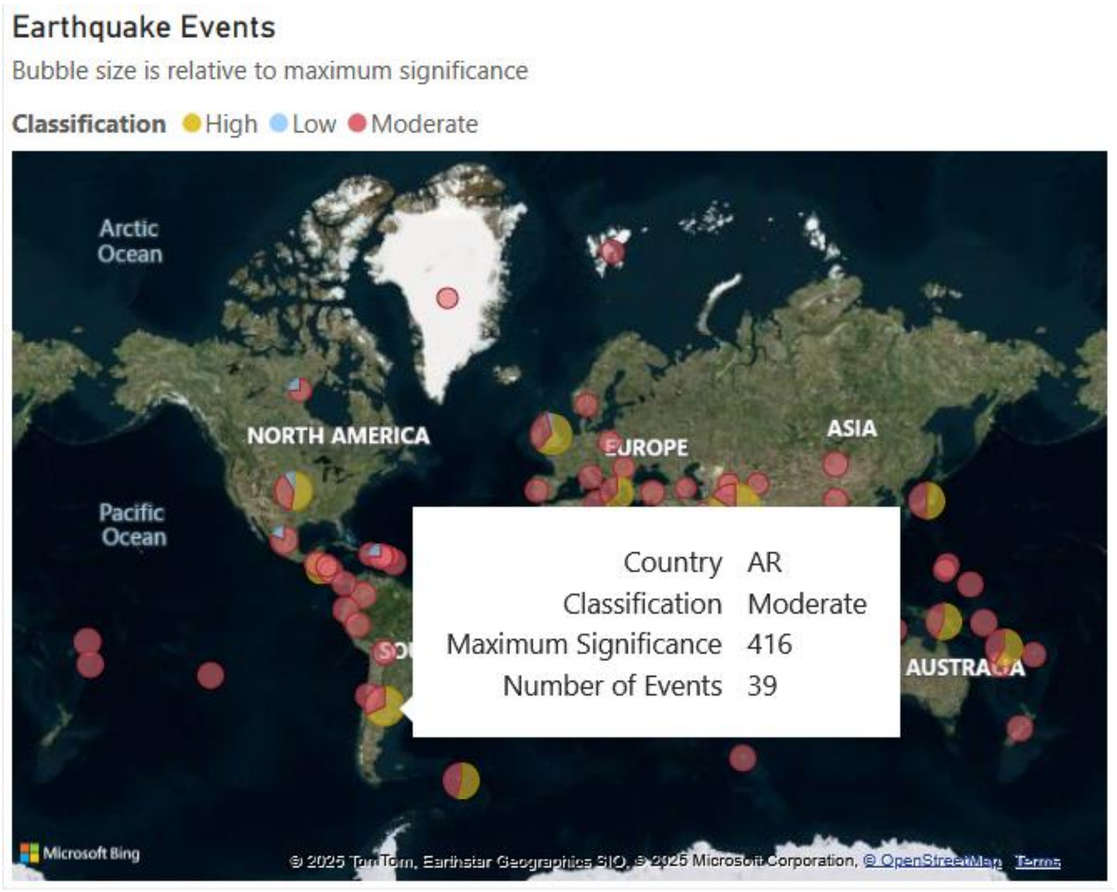
2149 Maximum Significance

Earthquake Events

Bubble size is relative to maximum significance

Classification ● High ● Low ● Moderate

Only Map


Arctic Ocean
Pacific Ocean
NORTH AMERICA
EUROPE
ASIA
AUSTRALIA
Microsoft Bing
© 2025 TomTom, Earthstar Geographics SIO, © 2025 Microsoft Corporation, © OpenStreetMap Terms

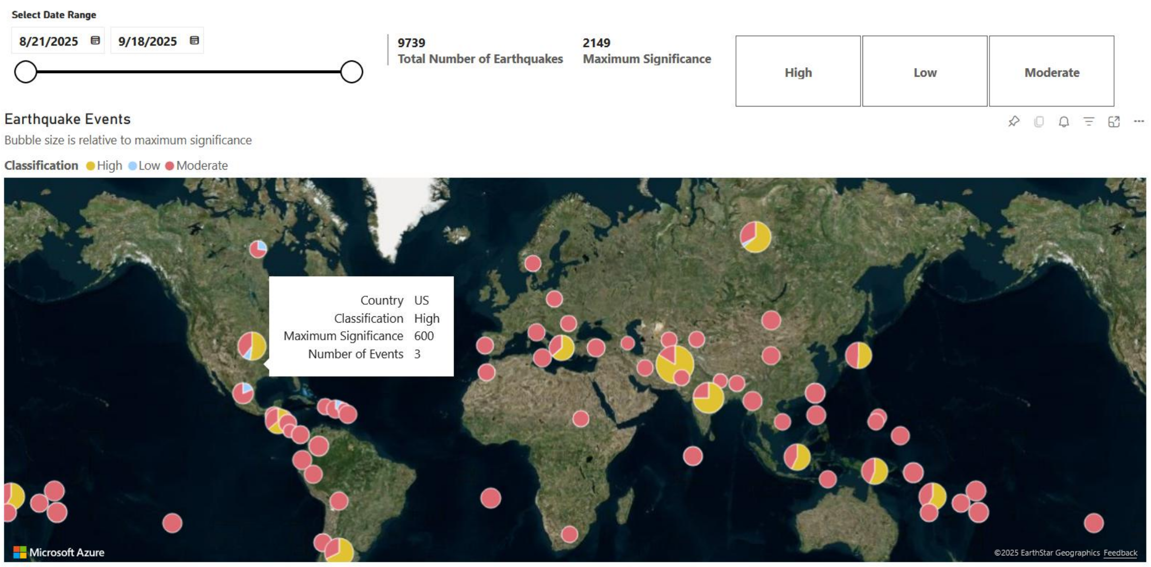
Earthquake Events

Bubble size is relative to maximum significance

Classification ● High ● Low ● Moderate

Country	AR
Classification	Moderate
Maximum Significance	416
Number of Events	39

Final map



Select Date Range

8/21/2025 9/18/2025

9739
Total Number of Earthquakes

2149
Maximum Significance

Earthquake Events

Bubble size is relative to maximum significance

Classification ● High ● Low ● Moderate

Country	US
Classification	High
Maximum Significance	600
Number of Events	3