DAMG 7275
Advanced Database Management Systems

P3 submission

Project - Farm Environmental Monitoring System

Project Group : 3

---

* Introduction :-

For this submission, we have implemented our architecture diagram and pipeline on Azure cloud platform. Some of the services that we have used are :

- Azure Blob Storage
- Azure SQL Server
- Azure SQL database
- Azure Comsos DB
- Azure Data Factory

Name	Type	Resource group	Location	Subscription
adbms-finalproject	Azure Cosmos DB account	shalom_adbms_RG	East US 2	shalomd_adbms_finalproject
adbms-sql-server	SQL server	shalom_adbms_RG	East US 2	shalomd_adbms_finalproject
adbms_sql_db (adbms-sql-server/adbms_sql_db)	SQL database	shalom_adbms_RG	East US 2	shalomd_adbms_finalproject
adbmsstorage	Storage account	shalom_adbms_RG	East US 2	shalomd_adbms_finalproject
shalom-adbms-data-factory	Data factory (V2)	shalom_adbms_RG	East US 2	shalomd_adbms_finalproject

* Dataset :-

We have 3 primary sources of data for this project implementation :

- Weather data (JSON)
- Soil moisture data (CSV)
- Camera images (JPG)

soil_moisture_data

soil_reading_id	reading_date	reading_time	sensor_id	moisture_reading
1	2023-11-13	08:00:00	101	25.5
2	2023-11-14	12:30:00	102	30.2
3	2023-11-15	15:45:00	103	22.8
4	2023-11-16	10:10:00	104	18.6
5	2023-11-17	14:20:00	105	28.1
6	2023-11-18	09:55:00	106	33.7
7	2023-11-19	11:40:00	107	19.3

```
[
  {
    "weather_reading_id": 1,
    "data": "2023-01-01",
    "time": "12:00:00",
    "temp": 25.5,
    "humidity": 70.2,
    "rainfall": 0.0,
    "longitude": -73.975,
    "latitude": 40.783
  },
  {
    "weather_reading_id": 2,
    "data": "2023-01-02",
    "time": "14:30:00",
    "temp": 22.3,
    "humidity": 68.8,
    "rainfall": 0.2,
    "longitude": -74.006,
    "latitude": 40.712
  },
  {
    "weather_reading_id": 3,
    "data": "2023-01-03",
    "time": "10:45:00",
    "temp": 28.1,
    "humidity": 75.5,
    "rainfall": 0.0,
    "longitude": -73.986,
    "latitude": 40.748
  },
  {
    "weather_reading_id": 4,
    "data": "2023-01-04",
    "time": "08:15:00",
    "temp": 19.8,
    "humidity": 62.4,
    "rainfall": 0.5,
    "longitude": -73.943,
    "latitude": 40.669
  },
```

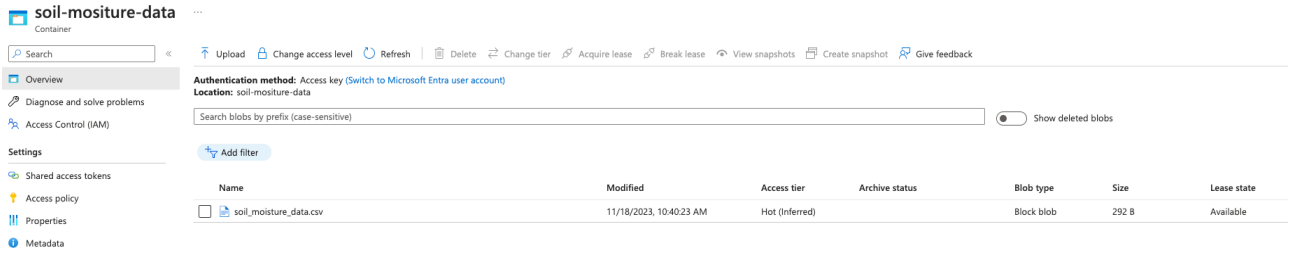
* Implementation :-

We are using Azure Blob Storage to manage our data. All data from each source is stored in respective containers on Azure like below.

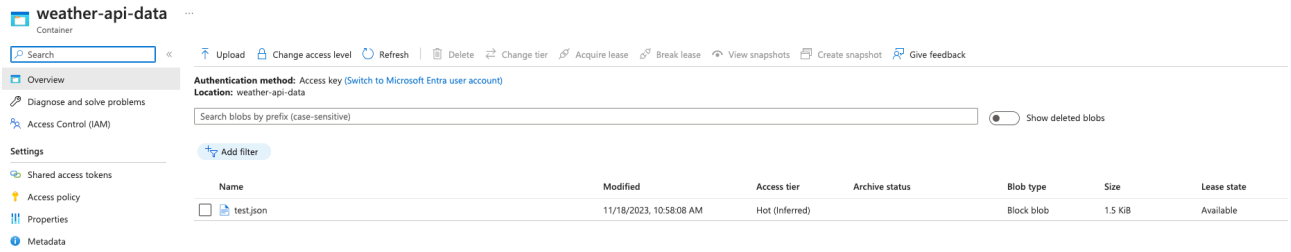
Name	Last modified
$logs	11/18/2023, 10:10:45 AM
camera-images	11/18/2023, 12:02:07 PM
soil-mositure-data	11/18/2023, 10:11:23 AM
weather-api-data	11/18/2023, 10:57:27 AM

Each container will have its own data file :

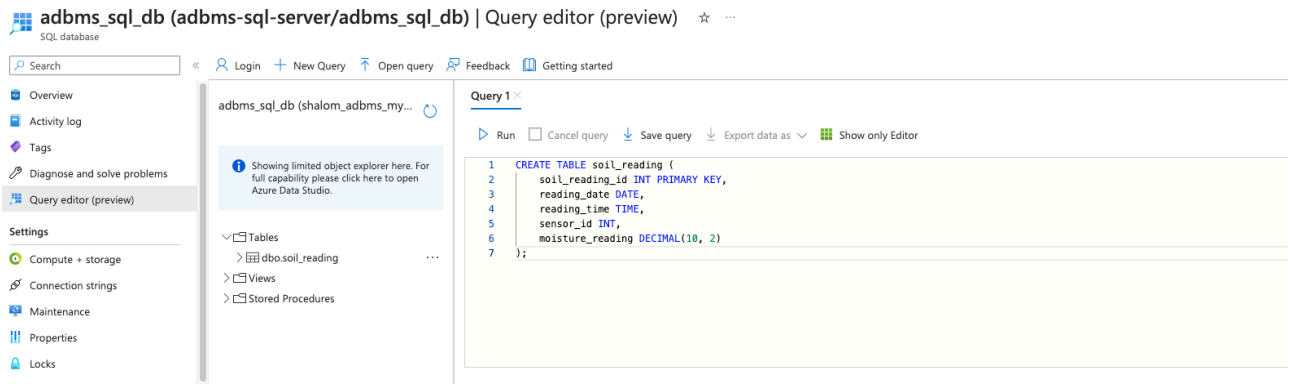
Soil moisture data in csv format -



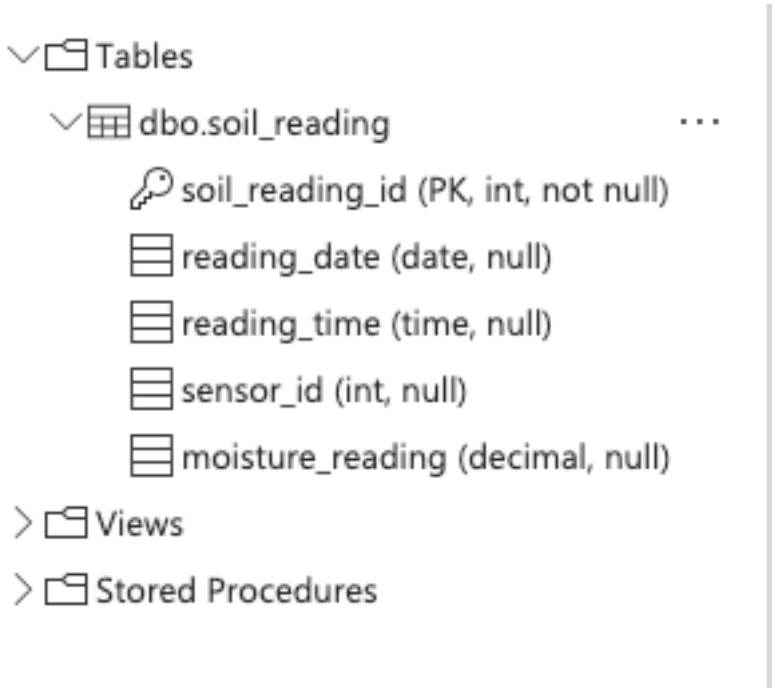
Weather data in json format -



We are using Azure SQL DB to handle our relational data model which in this case is Soil Moisture Data.

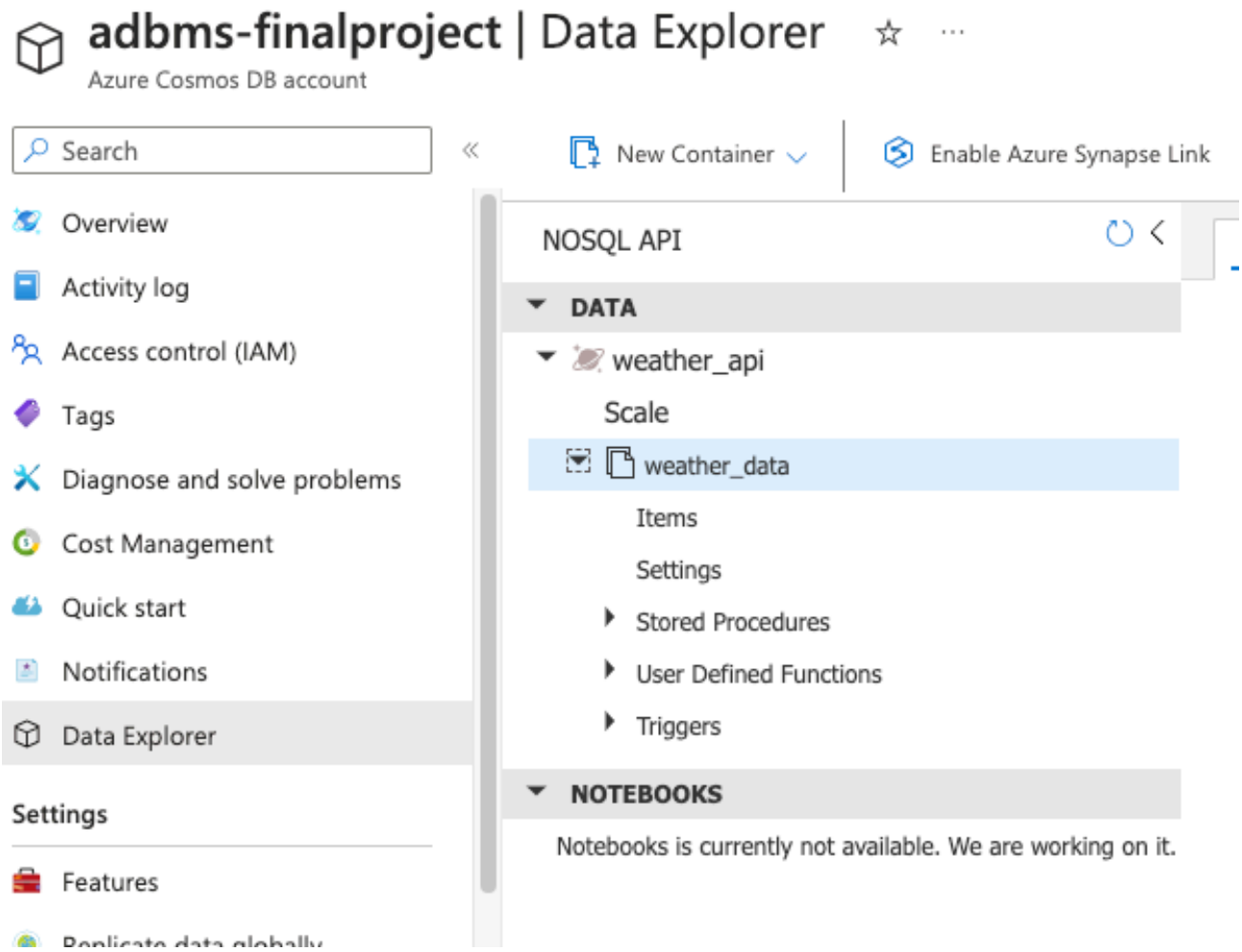


We have created a table on Azure SQL db replicating the schema of our soil moisture data. Once sql command query is successful, we see the table is created as below.



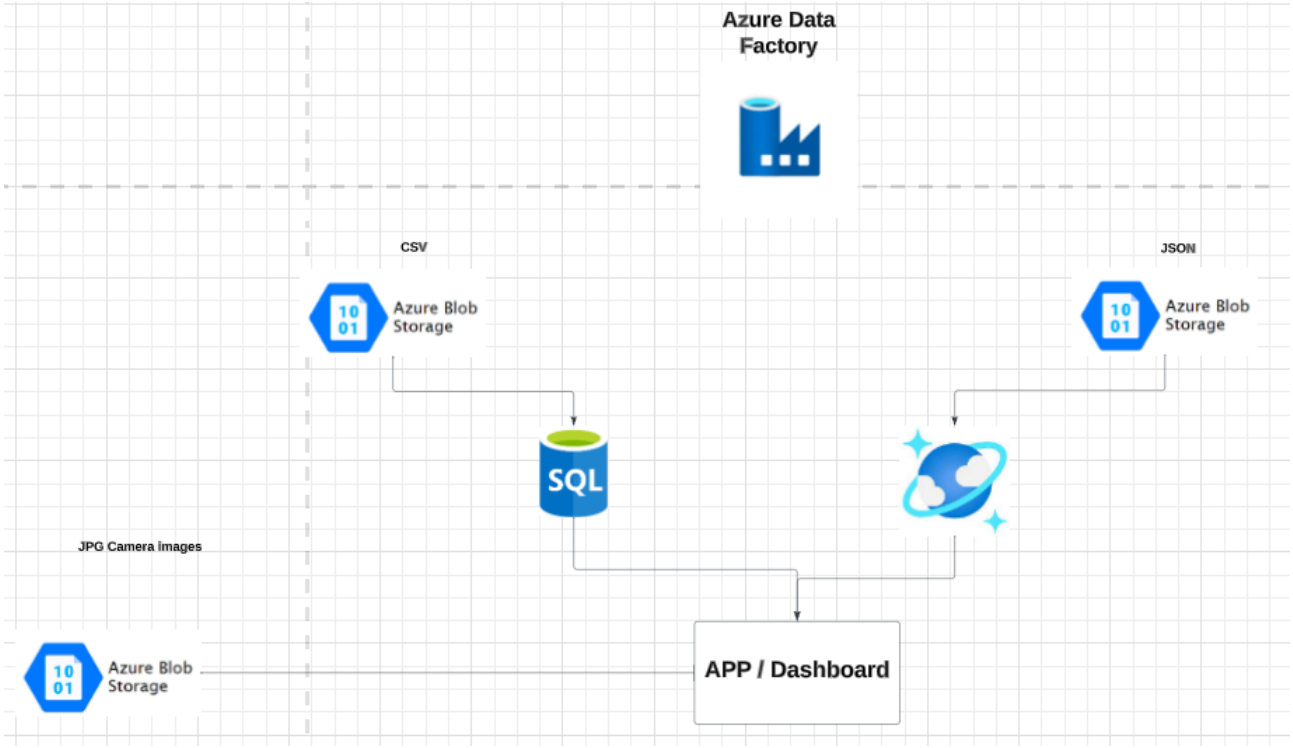
We are using Azure Cosmos DB to handle our document data model which in this case is Weather Data.

On Cosmos DB, we have created a db named *weather_api* & a container within it named *weather_data* to handle incoming weather data in JSON format.



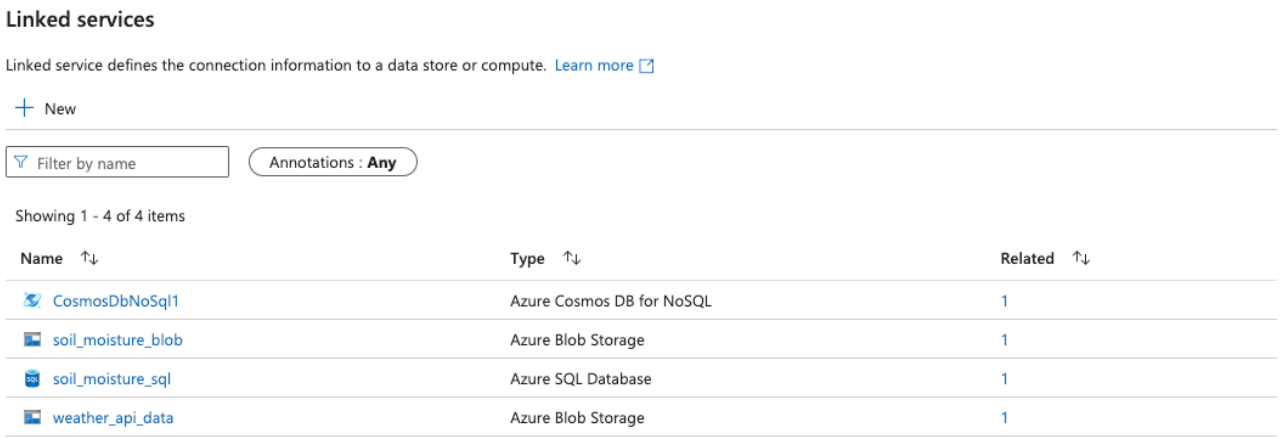
We are using Azure Data Factory for our data pipeline which will be used to orchestrate each of the services and ingest data into their respective databases.

Below is the pipeline and data flow.

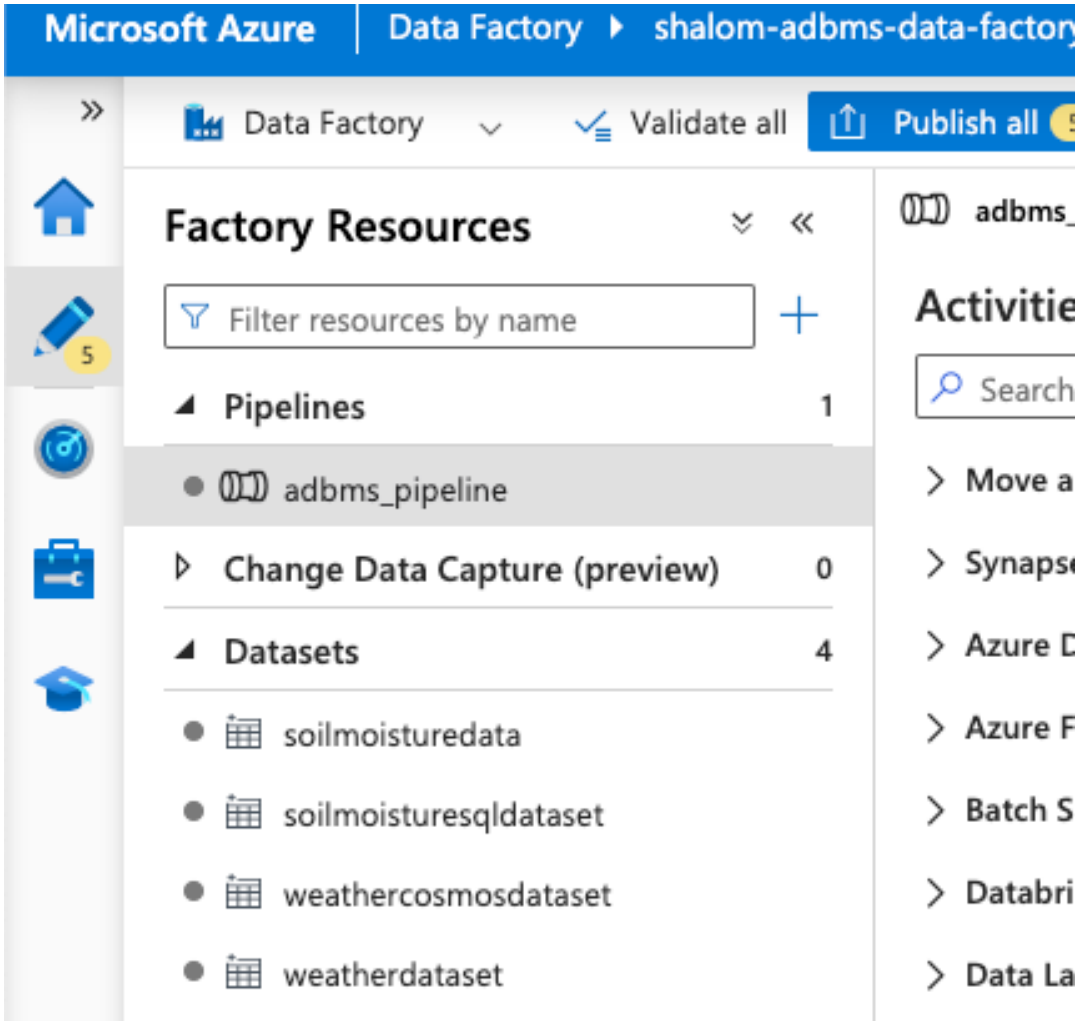


Note: *JPG camera images are not part of the pipeline but they are stored in Azure blob storage directly.*
The app or dashboard will directly pull images from Azure storage container.

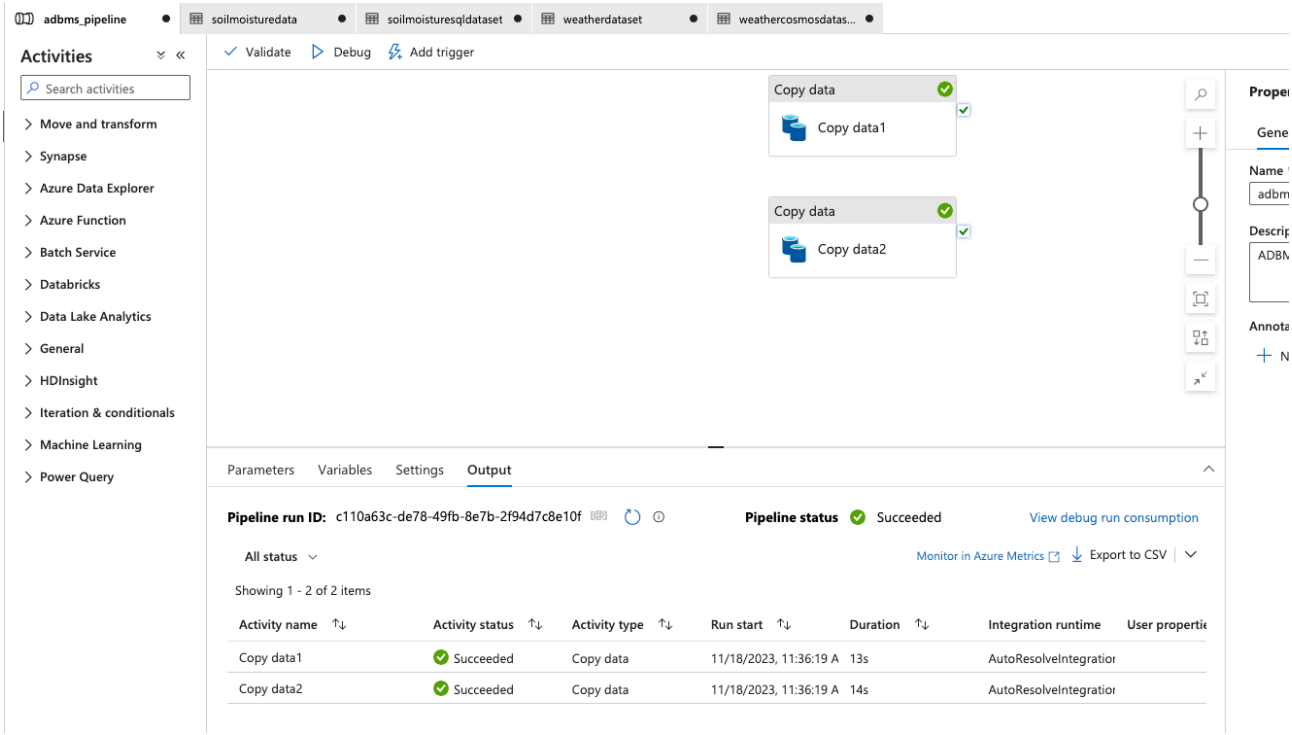
We have created 3 Linked Services on Data Factory to handle the data models.



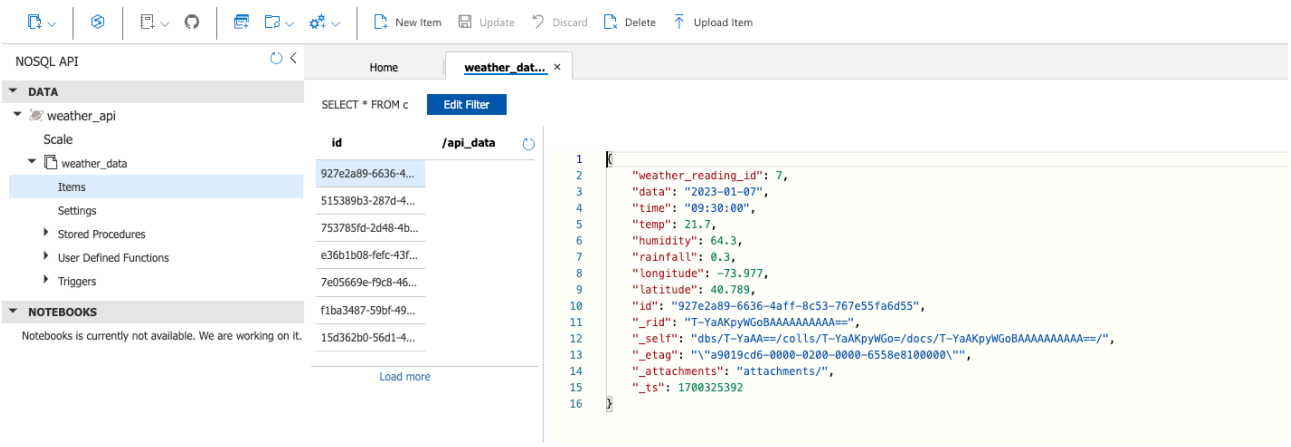
We have created 3 Datasets on Data Factory to replicate the schema of the data models.



In our pipeline, we have added 2 Copy steps to ingest csv data from blob storage to Sql db and json data from blob storage to Cosmos db.



We ran the Debug option to test our pipeline and it succeeded. Below is the result of the pipeline. The Sql DB table and Cosmos DB container is correctly populated with expected data models.



Query 1

Run | Cancel query | Save query | Export data as | Show only Editor

```
1  Select * from [dbo].[soil_reading]
```

Results | Messages

Search to filter items...

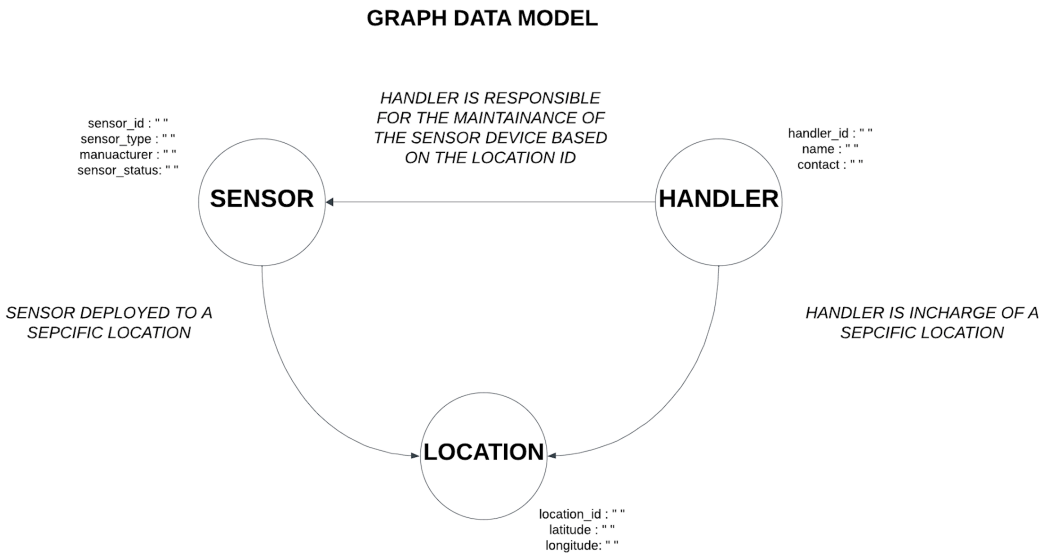
soil_reading_id	reading_date	reading_time	sensor_id	moisture_reading
1	2023-11-13T00:00:00.0000000	08:00:00	101	25.50
2	2023-11-14T00:00:00.0000000	12:30:00	102	30.20
3	2023-11-15T00:00:00.0000000	15:45:00	103	22.80
4	2023-11-16T00:00:00.0000000	10:10:00	104	18.60
5	2023-11-17T00:00:00.0000000	14:20:00	105	28.10
6	2023-11-18T00:00:00.0000000	09:55:00	106	33.70
7	2023-11-19T00:00:00.0000000	11:40:00	107	19.30

Query succeeded | 0s

For the graph data model implementation, we are using a python script to extract vertices and edges from our relational data models : Location, Handler & Sensor.

The Python script also gives us the Gremlin API queries which we will use to create the vertices and edges on Azure CosmosDB with Gremlin API account.

Below is our Graph data model -



Below are the Gremlin queries we ran on the Gremlin API console -

a) Creating Location vertex:

```
g.addV('location').property('locationID', 1).property('lat', 123.45).property('long', 67.89).property('city', 'City1').property('country', 'Country1')
```

b) Creating Handler vertex and edge:

```
g.addV('handler').property('handlerID', 101).property('name',
'Handler1').property('contact', ‘Contact1')
```

```
g.V().has('locationID',
1).addE('hasHandler').to(g.V().has('handlerID', 101))
```

c) Creating Sensor node and edge:

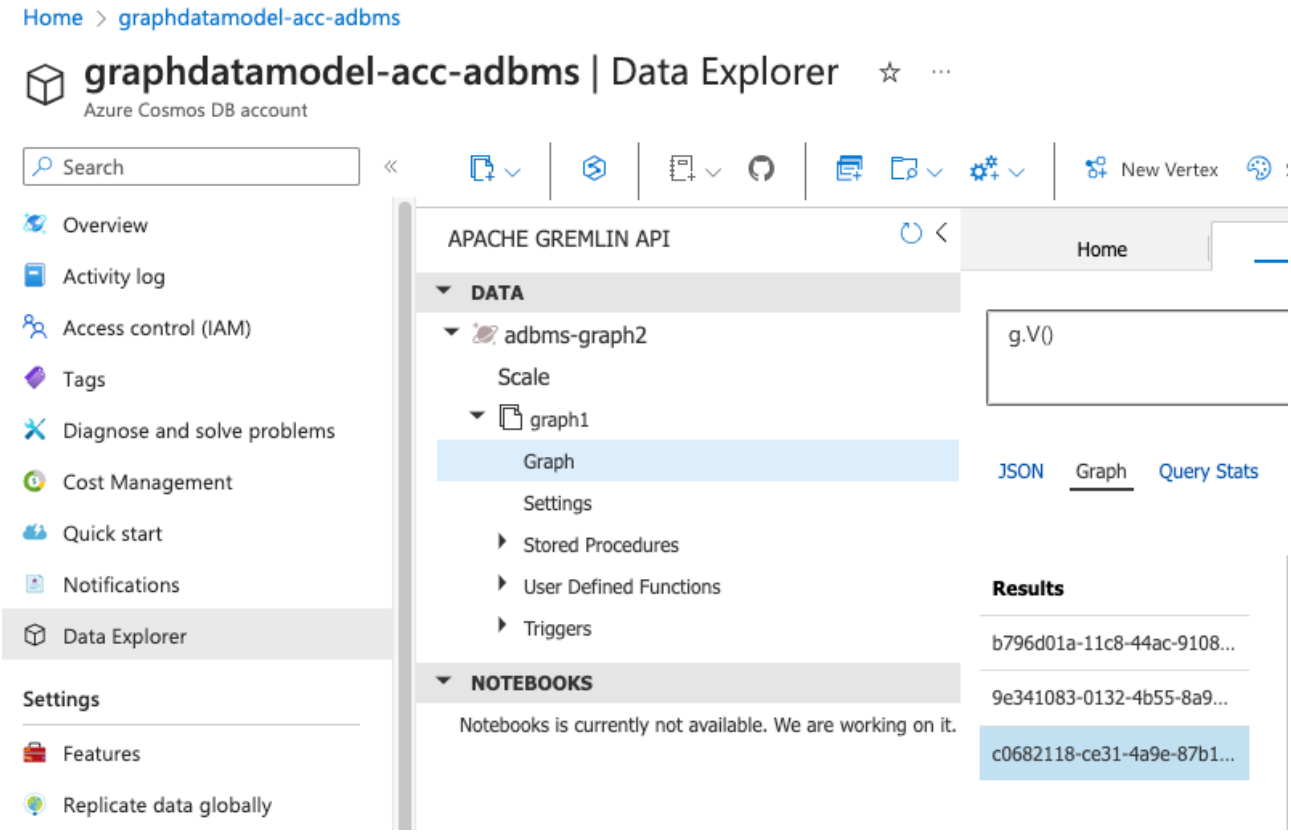
```
g.addV('sensor').property('sensorID', 1001).property('type',
'Type1').property('manufacturer',
'Manufacturer1').property('sensor_status', ‘Active')
```

```
g.V().has('locationID',
1).addE('hasSensor').to(g.V().has('sensorID', 1001))
```

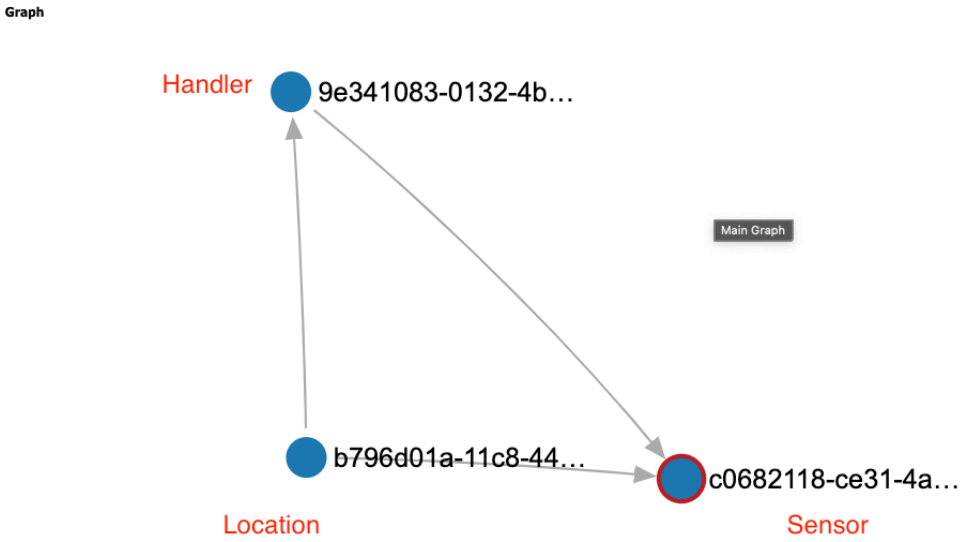
d) Creating edge between Handler and Sensor:

```
g.V().has('handlerID',
101).addE('handles').to(g.V().has('sensorID', 1001))
```

We have create Azure cosmos's with gremlin api account and configured a new database with a graph as shown below:



Below is the screenshot of the graph created with the vertices and edges specified through the gremlin api:



Note: *Azure Data Factory does not have a connector for Azure CosmosDB with Gremlin API yet, so we could not integrate our graph data model with our existing pipeline.*

We will be running our python script mentioned above whenever there is a change in our data sources to refresh the graph data model.

* Next steps & Future Scopes :-

1) Implement an application and dashboard to make sense of the data that we have used and for visualisations. (P5)

2) Auto data fresh. (P4)

3) Try and increase size of datasets for better visibility & analysis.

* Group members :-

1) Ankita Patil
2) Aditya Pande
3) Keshni Mulrajani
4) Sachit Wagle
5) Shalom Daniel

****