Analysis of UNICEF Malnutrition Data Using CosmosDB Multi-Model

Course: Advanced database management system (DAMG 7275)

Team 11 group members:

- Abhishek Patil (002713962)
- Amey Parange (002791448)
- Mayuri Bashirabadkar (002745804)
- Yashraj Rajput (002713988)

INDEX

Introduction

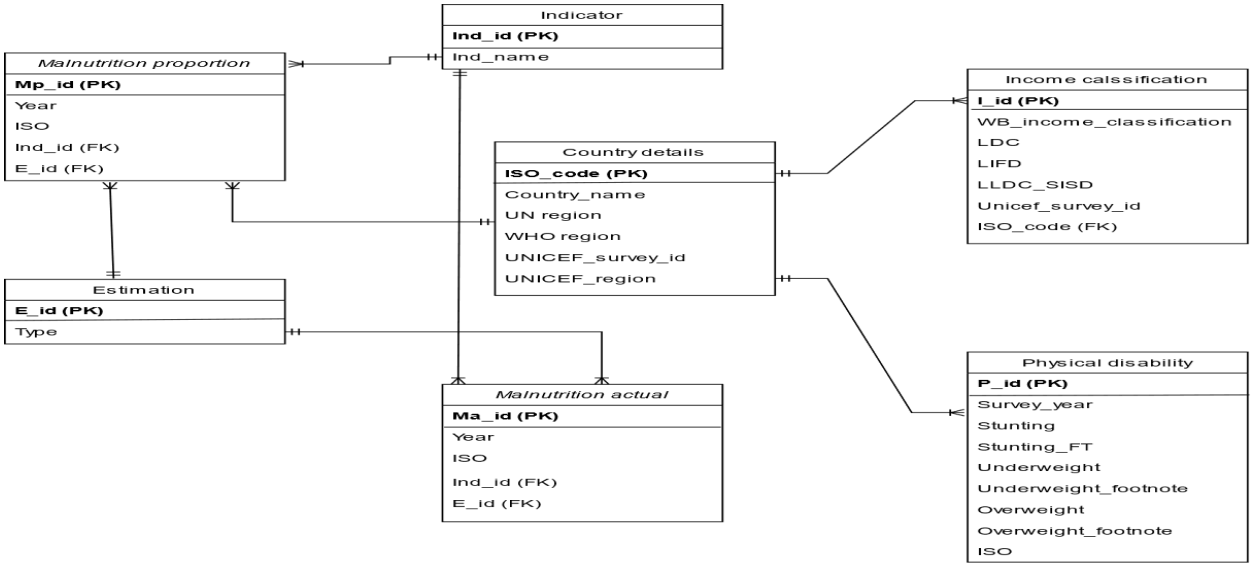
Malnutrition is a serious public health problem that affects people in developed as well as developing nations.

Our aim is to study UNICEF data on malnutrition and monitor the nations exhibiting falling and rising trends in terms of malnutrition.

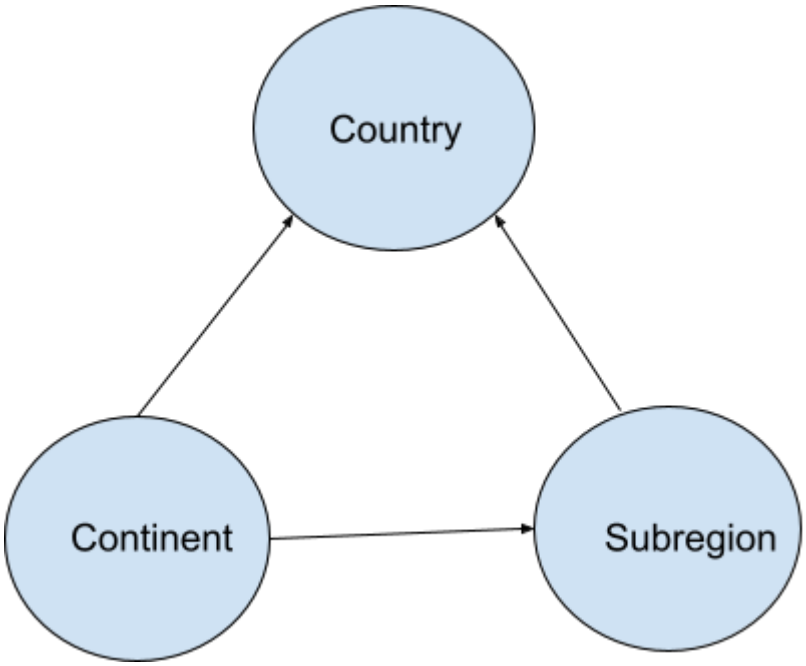
This goal is majorly achieved by using the CosmosDB database; SQL as well as Gremlin APIs are used.

Implementation architecture

ER diagram



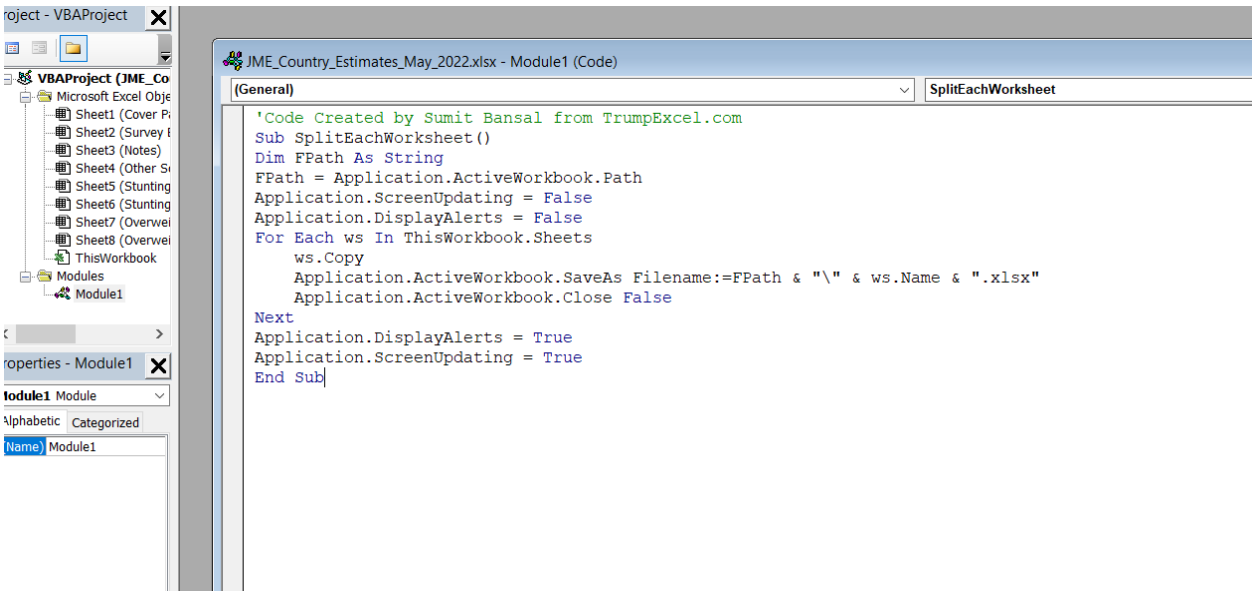
Graph diagram



Implementation

PART I: Working on Dataset according to Requirement:

Breaking Datasets into multiple CSV according to requirement:



Cleaning the raw data and curating using Python scripts:

```
5]: import pandas as pd
    cp= pd.read_excel("Survey Estimates.xlsx")
    cp.head()
```

5]:

	ISO code	Country and areas	Survey year	Year*	United Nations Region	United Nations Sub-Region	UNICEF Region	UNICEF Sub-Region	WHO Region	World Bank Income Classification	...	Wasting Footnote	Overweight	Overweight Footnote	Stunting	Stunting Footnote
0	AFG	AFGHANISTAN	2004	2004	Asia	Southern Asia	SA	SA	EMRO	Low Income	...	w 11	4.6	w 11	59.3	w 11
1	AFG	AFGHANISTAN	2013	2013	Asia	Southern Asia	SA	SA	EMRO	Low Income	...	rs	5.3	rs	40.4	r
2	AFG	AFGHANISTAN	2018	2018	Asia	Southern Asia	SA	SA	EMRO	Low Income	...	rs	4.1	rs	38.2	r

```
In [46]: null_cols=cp.columns[cp.isna().any()]
         cp[null_cols].isna().sum()

Out[46]: Series([], dtype: float64)
```

```
In [47]: cp = cp.drop(["WAZ Survey Sample (N)", "HAZ Survey Sample (N)","Short Source", "Source", "SDG Region"], axis = 1)
         cp.head()
```

```
: cp = cp.dropna()
  cp.head()
```

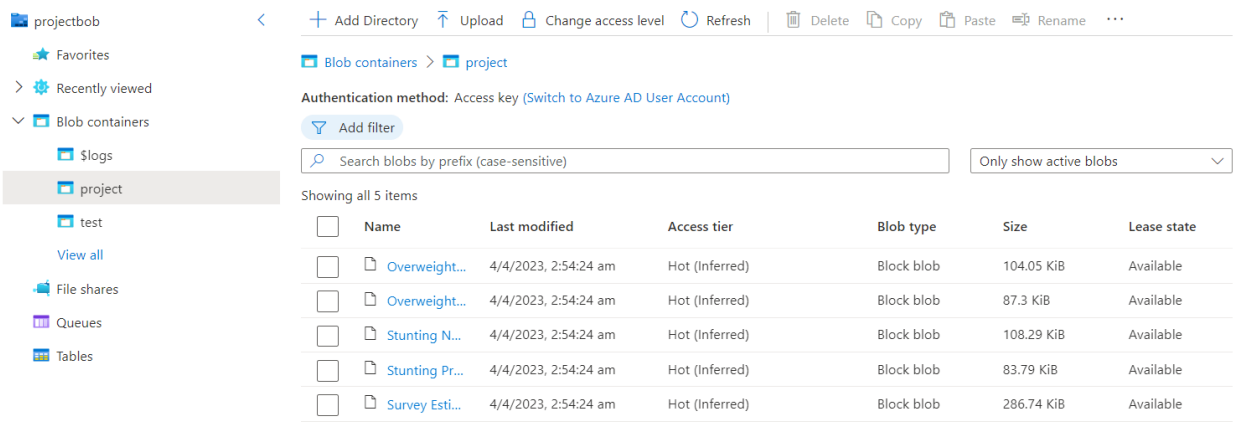
:

	ISO code	Country and areas	Survey year	Year*	United Nations Region	United Nations Sub-Region	UNICEF Region	UNICEF Sub-Region	WHO Region	World Bank Income Classification	...	Wasting Footnote	Overweight	Overweight Footnote	Stunting	Stunting Footnote
0	AFG	AFGHANISTAN	2004	2004	Asia	Southern Asia	SA	SA	EMRO	Low Income	...	w 11	4.6	w 11	59.3	w 11
1	AFG	AFGHANISTAN	2013	2013	Asia	Southern Asia	SA	SA	EMRO	Low Income	...	rs	5.3	rs	40.4	r

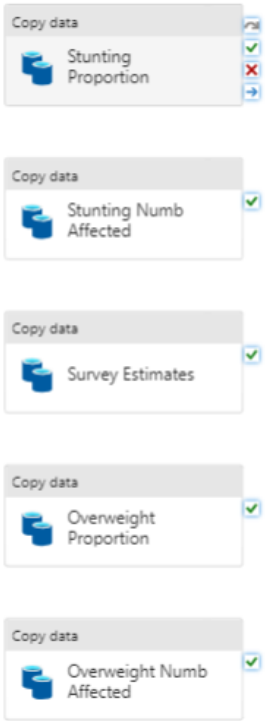
```
# Fill the missing values with a mean
mean = cp.mean()
cp = cp.fillna(mean)
cp.head()
```

```
cp.to_excel("Survey Estimates1.xlsx", index=False)
```

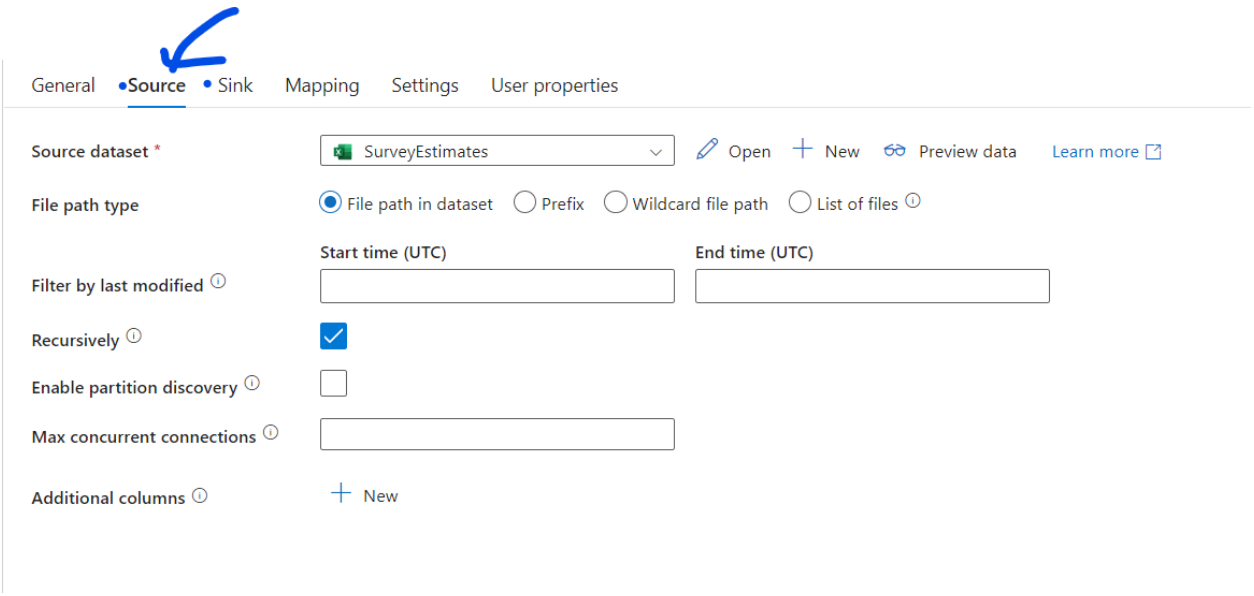
PART 2: Uploading Excel and csv files to blob storage

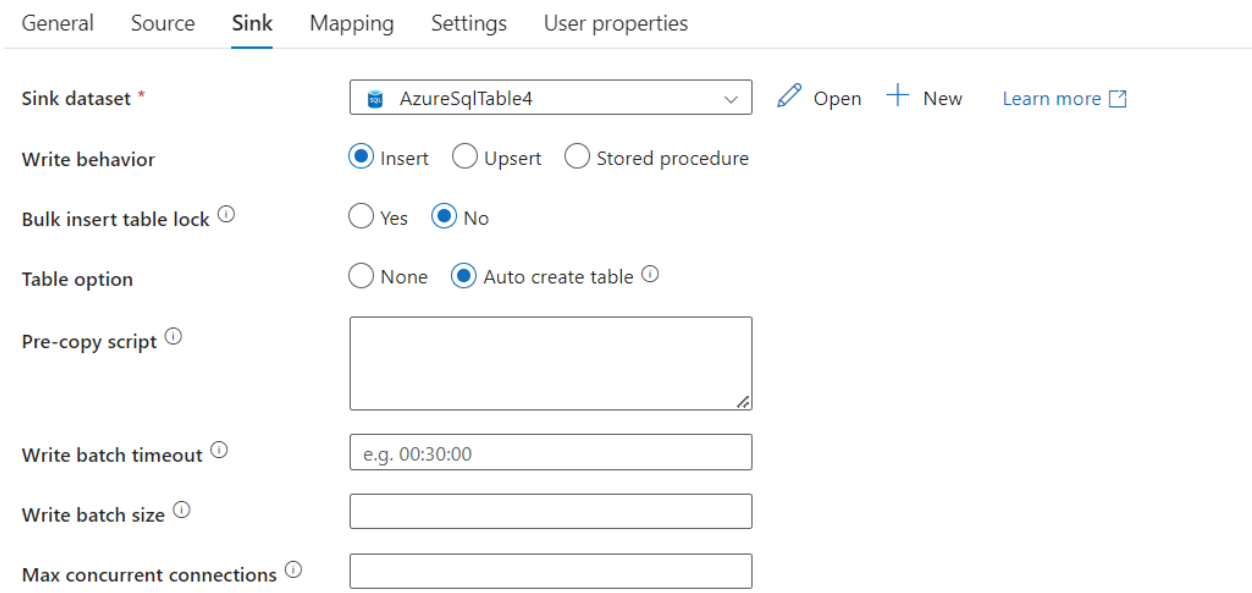


Creating pipeline to extract each file data to a SQL table.

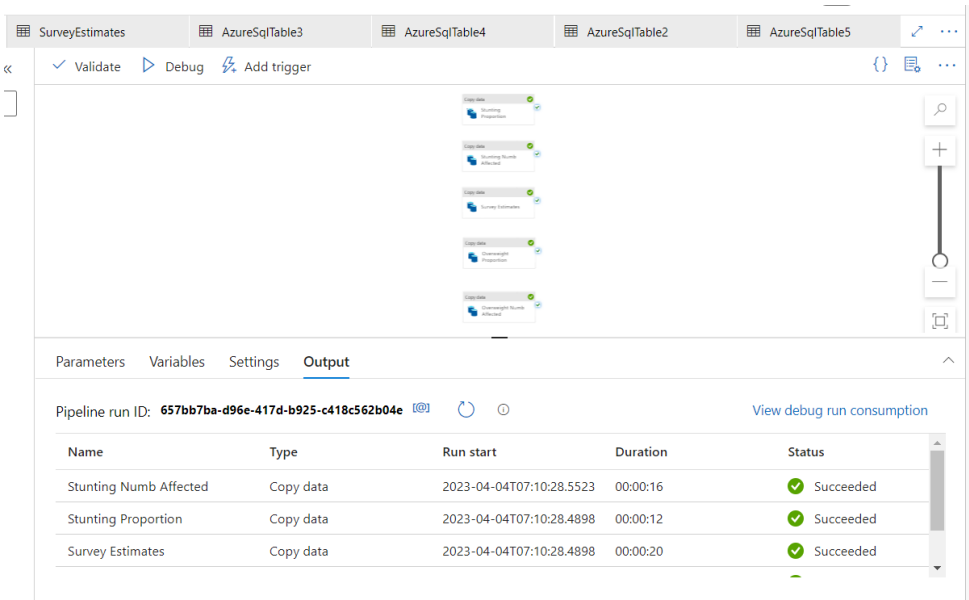


Creating source and sink dataset and linked service

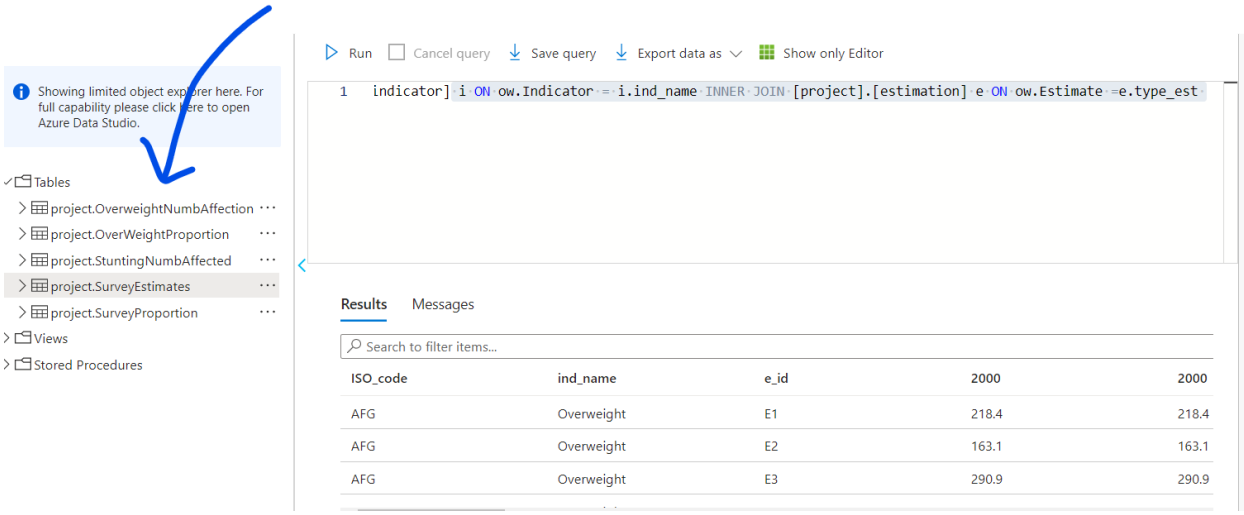




Running the pipeline successfully and loading data in Azure SQL Tables



Viewing the tables created in SQL

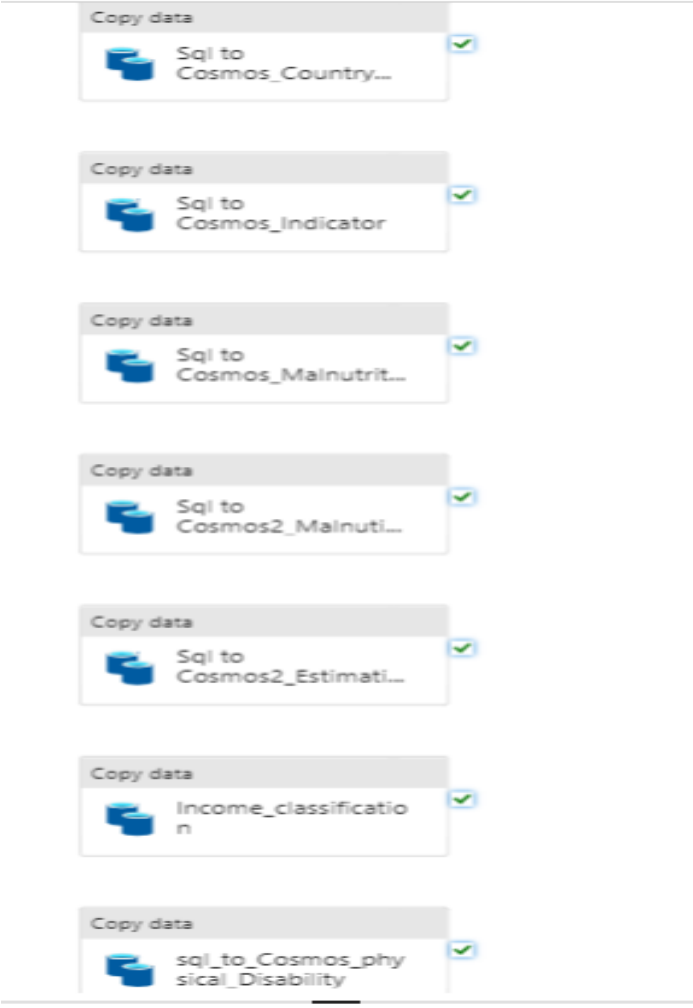


PART 3: Using Cosmos DB for SQL API

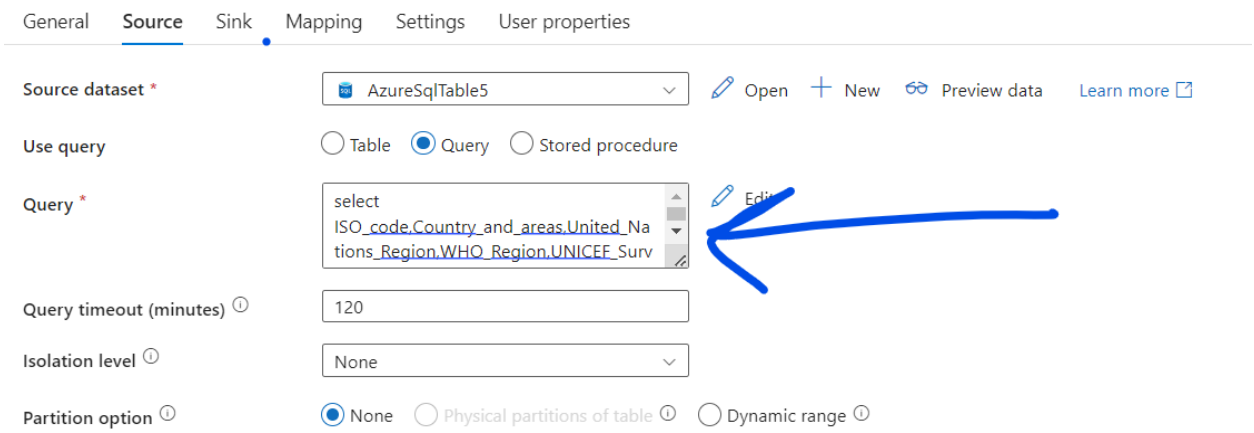
Data Load into CosmosDB

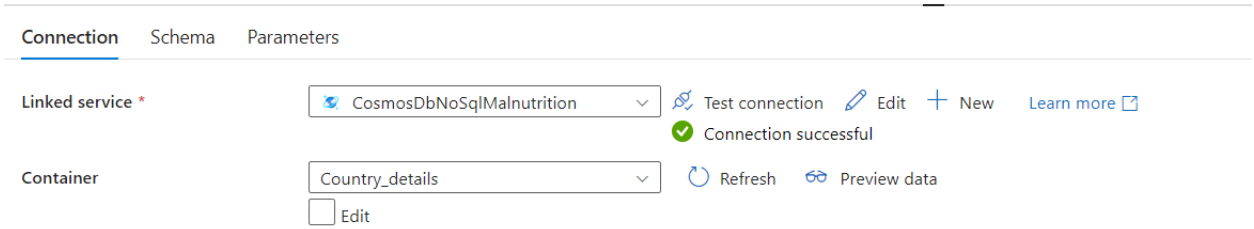
Transferring SQL data into NoSQL data(Document format) in CosmosDB through

Created a copy activity , source and destination datasets and linked services. a pipeline using Azure Data Factory.

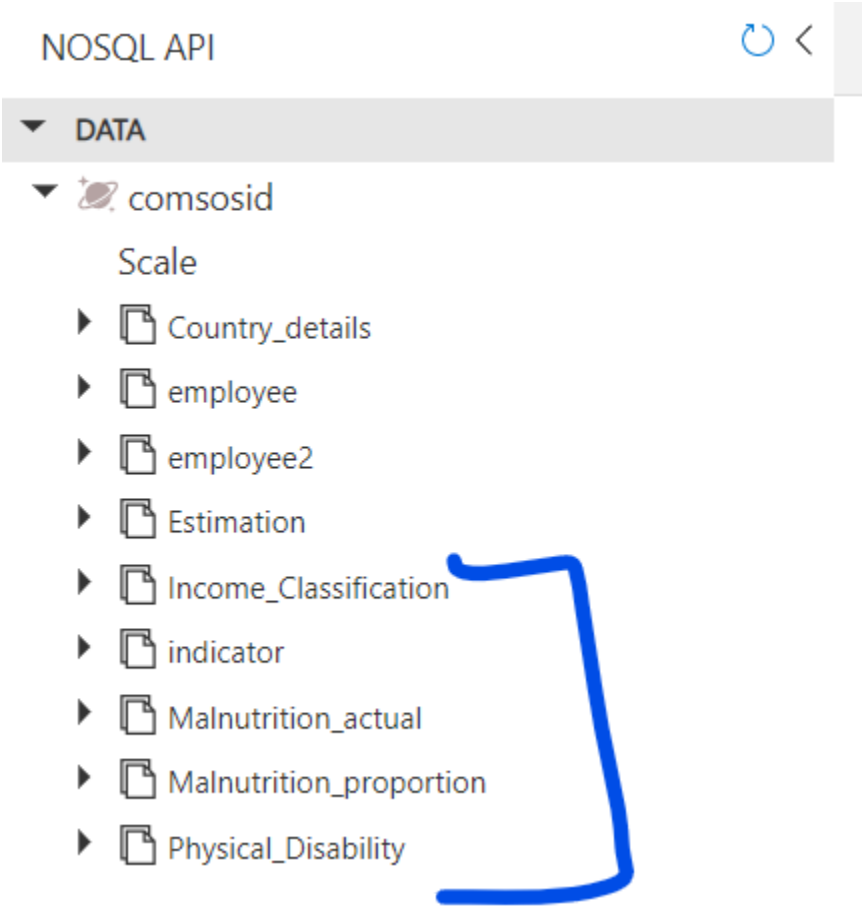


- **Created query to curate the columns required for getting insights**





- **Created containers according to ER diagram in Cosmos DB**



- **Running Pipeline successfully and loading data in Cosmos DB**

Sql to Cosmos_Indicator	Copy data	2023-04-07T00:43:28.73888	00:00:14	Succeeded
sql_to_Cosmos_physical_Disabil	Copy data	2023-04-07T00:43:28.70763	00:00:14	Succeeded
Income_classification	Copy data	2023-04-07T00:43:28.70763	00:00:17	Succeeded
Sql to Cosmos_CountryDetail	Copy data	2023-04-07T00:43:28.70763	00:00:23	Succeeded

- **Viewing items in Cosmos DB Documents**

SELECT * FROM c Edit Filter

id	/id
41878d...	41878d...
2743cf4f...	2743cf4f...
f9dcca7...	f9dcca7...
81cabc6...	81cabc6...
77a949d...	77a949d...
3cf6ebc...	3cf6ebc...
39f73c9...	39f73c9...
373e77f...	373e77f...
fb97375...	fb97375...
726b04...	726b04...
9f8d9b2...	9f8d9b2...
dfaa383...	dfaa383...

Load more


```
{
    "World_Bank_Income_Classification": "Low Income",
    "LDC": "Least Developed Countries (LDCs)",
    "LIFD": "Low Income Food Deficient (LIFD)",
    "LLDC_or_SIDS": null,
    "UNICEF_Survey_ID": "193",
    "ISO_code": "COD",
    "id": "f9dcca74-b4be-462f-ac61-9ff6ff85aba8",
    "_rid": "x1MRAPl-fOVDAAAAAAAAAA==",
    "_self": "dbs/x1MRAA==/colls/x1MRAPl-fOU=/docs/x1MRAPl-fOVDAAAAAAAAAA==/",
    "_etag": "\"56011dd1-0000-0700-0000-642f673f0000\"",
    "_attachments": "attachments/",
    "_ts": 1680828223
}
```

PART 4: Using Cosmos DB for Gremlin API

Working on a Dataset using Python.

Using python script we connected to Gremlin API and loaded Data into MalnutritionGraph

Script screenshots:

```
#read source file
#df_survey = pd.read_csv('Survey_Estimates.csv')
df_survey = pd.read_excel('SurveyEstimates2.xlsx')

#Select required columns
df_survey = df_survey[[ 'Country and areas','United Nations Region','United Nations Sub-Region','Overweight','Stunting','Underweight']]

#Rename Columns
df_survey.rename(columns={"Country and areas": "Country", "United Nations Region": "Continents","United Nations Sub-Region":"Sub_Continents"}, inplace=True)


#Country Vertices
country=df_survey['Country'].unique()

#SubContinents Vertices
sub_continents=df_survey['Sub_Continents'].unique()

#Continents Vertices
continents=df_survey['Continents'].unique()
```

```
#Add sub_continents to vertexs

_gremlin_insert_subcontinent_vertices=[]
for x in sub_continents:
    str="g.addV('Continent').property('id','" +x+"').property('Continent', '" +x+"').property('pk', 'pk')"
    _gremlin_insert_subcontinent_vertices.append(str)

#Query Continent and Country edge
_gremlin_insert_country_continent_edges=[]
for index,x in df_cont_country.iterrows():
    str="g.V('" +x['Continents']+"').addE('has').to(g.V('" +x['Country']+"'))"
    _gremlin_insert_country_continent_edges.append(str)
```

```python
def insert_vertices_subcontinent(client):
    for query in _gremlin_insert_subcontinent_vertices:
        print("\n> {0}\n".format(query))
        callback = client.submitAsync(query)
        if callback.result() is not None:
            print("\tInserted this vertex:\n\t{0}".format(
                callback.result().all().result()))
        else:
            print("Something went wrong with this query: {0}".format(query))
        print("\n")
        print_status_attributes(callback.result())
        print("\n")

    print("\n")

def insert_edges_country_continent(client):
    for query in _gremlin_insert_country_continent_edges:
        print("\n> {0}\n".format(query))
        callback = client.submitAsync(query)
        if callback.result() is not None:
            print("\tInserted this edge:\n\t{0}\n".format(
                callback.result().all().result()))
        else:
            print("Something went wrong with this query:\n\t{0}".format(query))
        print_status_attributes(callback.result())
        print("\n")

    print("\n")
```

```python
try:
    client = client.Client('wss://graphapi.gremlin.cosmos.azure.com:443/', 'g',
                           username="/dbs/graphdb/colls/Malnutrition",
                           password="hCcvcs5uVphbUFYNocagXHBx69U0kqisoKpmGVd3Odp0DWSOx0YNtNTB4LmK28Hsprv1w9fvpaa4ACDbPAFqqw==",
                           message_serializer=serializer.GraphSONSerializersV2d0()
                           )

    print("Welcome to Azure Cosmos DB + Gremlin on Python!")

    # Drop the entire Graph
    input("We're about to drop whatever graph is on the server. Press any key to continue...")
    cleanup_graph(client)

    # Insert all vertices
    input("Let's insert some vertices into the graph. Press any key to continue...")
    insert_vertices_country(client)

    insert_vertices_continent(client)

    insert_vertices_subcontinent(client)

    # Create edges between vertices
    #input("Now, let's add some edges between the vertices. Press any key to continue...")
    insert_edges_country_continent(client)
    insert_edges_subcont_continent(client)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                                                    Python Debug Console

me-ms': 15.7652, 'x-ms-total-server-time-ms': 15.7652}


> g.V('Africa').addE('has').to(g.V('Western Africa'))

        Inserted this edge:
        [{'id': '9ebb762e-8469-4c35-baa3-3261870b4ad5', 'label': 'has', 'type': 'edge', 'inVLabel': 'Continent', 'outVLabel': 'Continent', 'inV': 'Western Africa', 'outV': 'Africa'}]

        Response status_attributes:
        {'x-ms-status-code': 200, 'x-ms-activity-id': 'efebb197-3a4c-4b3c-b769-18fa5bf0779b', 'x-ms-request-charge': 14.420000000000002, 'x-ms-total-request-charge': 14.420000000000002, 'x-ms-server-ti
me-ms': 11.3598, 'x-ms-total-server-time-ms': 11.3598}


> g.V('Africa').addE('has').to(g.V('Eastern Africa'))
```

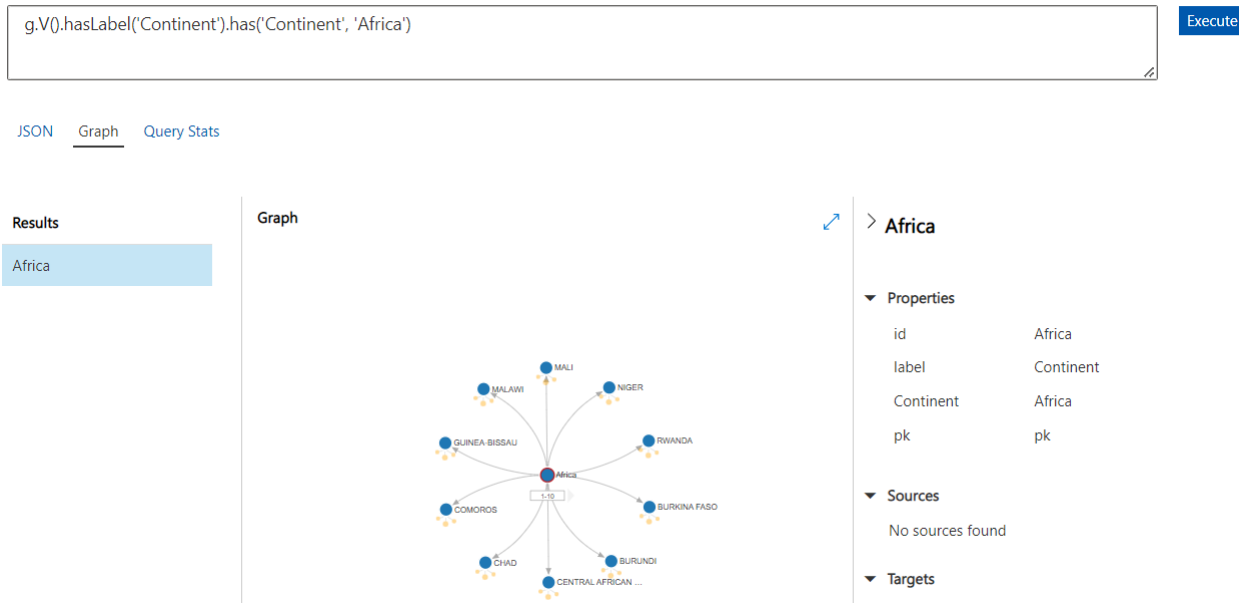
```
        Response status_attributes:
        {'x-ms-status-code': 200, 'x-ms-activity-id': '82ba7c7c-3c28-4b3f-aaea-353e3f046b44', 'x-ms-request-charge': 7.81, 'x-ms-total-request-charge': 7.81, 'x-ms-server-time-ms': 6.2335, 'x-ms-total-
server-time-ms': 6.2335}


> g.addV('Continent').property('id','Eastern Africa').property('Continent', 'Eastern Africa').property('pk', 'pk')

        Inserted this vertex:
        [{'id': 'Eastern Africa', 'label': 'Continent', 'type': 'vertex', 'properties': {'Continent': [{'id': '537ff1c2-4deb-4288-8f68-475c6d5bb68e', 'value': 'Eastern Africa'}], 'pk': [{'id': 'Eastern
 Africa|pk', 'value': 'pk'}]}}]


        Response status_attributes:
        {'x-ms-status-code': 200, 'x-ms-activity-id': '1c1d3ad8-92ad-438e-bd35-66a117f34bdc', 'x-ms-request-charge': 7.81, 'x-ms-total-request-charge': 7.81, 'x-ms-server-time-ms': 7.3222, 'x-ms-total-
server-time-ms': 7.3222}
```

Sample Graphs in CosmosBD using Gremlin API.



```
g.V().hasLabel('Country').has('Country', 'NEPAL')
```

Execute

```
g.V().hasLabel('Country').has('Country', 'Nepal')
g.V().hasLabel('Country').has('Country', 'Nepal')
```

JSON Graph Query Stats

Results

NEPAL

Graph

NEPAL

Properties

id	NEPAL
label	Country
Country	NEPAL
Overweight	0.6666666666666666
Stunting	54.666666666666664

Conclusion

Here , we analyzed and examined the severely affected nations that are dealing with the after- effects of malnutrition, such as obesity, stunting, and underweight.

Future scope

Next step in the project would be to visualize the obtained data using PowerBI; which would make it easier for a layman to understand and absorb the data.

References

https://data.unicef.org/wp-content/uploads/2022/05/JME_Country_Estimates_May_2022.xlsx

https://learn.microsoft.com/en-us/azure/cosmos-db/gremlin/quickstart-python

https://www.youtube.com/watch?v=IowqLqR0xn4

https://www.youtube.com/watch?v=QDXZ4i15fy0&t=678s

https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/quickstart-portal