

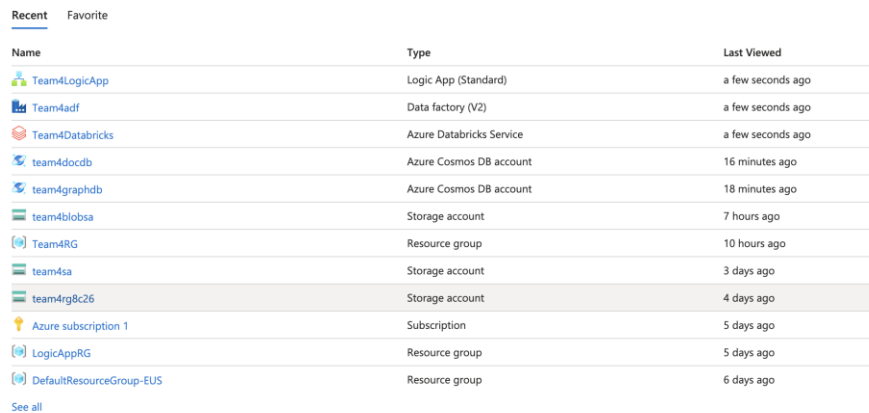
P-3: Implementation













Global Inbound and Outbound Travel

Introduction:

To implement the project, we have followed our architecture diagram. We have used many services available on Azure cloud platform which are listed below.

1. Azure Logic Apps
2. Azure Databricks
3. Azure Blob Storage
4. Azure Data Factory
5. Azure Cosmos DB



Name	Type	Last Viewed
 Team4LogicApp	Logic App (Standard)	a few seconds ago
 Team4adf	Data factory (V2)	a few seconds ago
 Team4Databricks	Azure Databricks Service	a few seconds ago
 team4docdb	Azure Cosmos DB account	16 minutes ago
 team4graphdb	Azure Cosmos DB account	18 minutes ago
 team4blobsa	Storage account	7 hours ago
 Team4RG	Resource group	10 hours ago
 team4sa	Storage account	3 days ago
 team4rgBc26	Storage account	4 days ago
 Azure subscription 1	Subscription	5 days ago
 LogicAppRG	Resource group	5 days ago
 DefaultResourceGroup-EUS	Resource group	6 days ago

[See all](#)

Dataset Details:

Our primary dataset has been loaded from CSV files. It contains the details if inbound travel and travel information to and across different countries. The data dictionary is listed below.

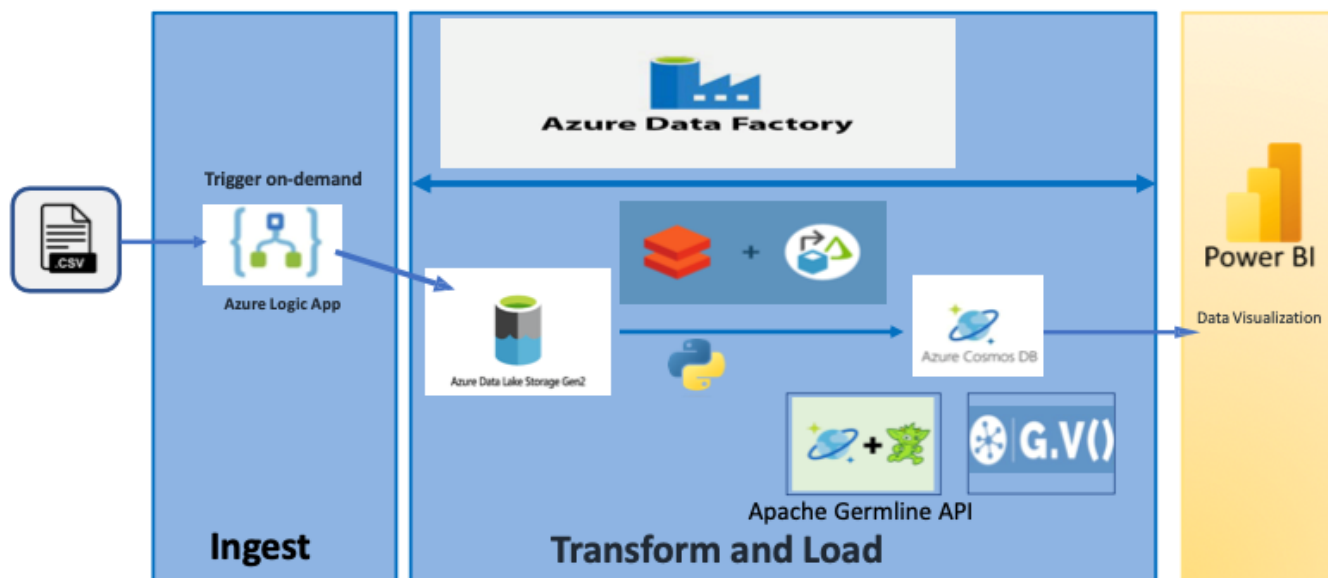
Column Name	Details
id	Numeric id
First_name	Tourist's first name
Last_name	Tourist's last name
gender	Tourist's gender
age	Tourist's age
Visa_type	Type of visa that tourist holds (business, pleasure, student)
Passport_number	Tourist's passport number

nationality	Tourist's country of origin
state	Tourist's state of origin
Destination_country	Destination country travelled
Destination_state	Destination state travelled
Entry date	Entry date in the destination country
Exit_date	Exit date from the destination country
Estimated expenses	Estimated expenditure in the destination country
transportation	Transport mode used to reach destination country
Degree_type	Degree type that student is pursuing (only available for student visa type)
major	Major that student is pursuing (only available for student visa type)
institution	Institution where the student is enrolled(only available for student visa type)
Start date	Program start date for students (only available for student visa type)
End date	Program end date for students (only available for student visa type)
Visa_Sub_type	visa sub category (only available for business visa type)

Data sample

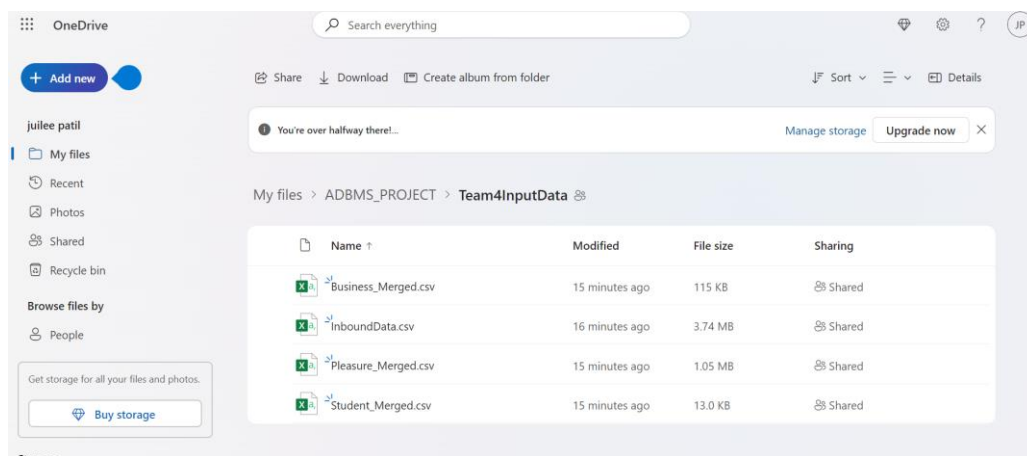
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	id	first_name	last_name	gender	age	visa_type	passport	nationality	state	destination	destination	entry_date	exit_date	estimated	transport	degree_type	major	institution	start_date	end_date	visa_sub_type
1022	1021	Immanuel	Moys	M	21	Student	780556886	Canada	Ontario	United State Colorado	8/18/2023	12/29/2024	515915.66	air	Master's	Computer Sc	Baker Collg	1/25/2023	6/25/2024		
1023	1022	Karney	Colmer	M	33	Student	4037159791	Canada	Alberta	United State Michigan	09/01/23	01/10/25	517451.98	air	Master's	Business Adm	Pennsylvania	1/26/2023	03/09/25		
1024	1023	Weneshah	Escalera	F	27	Student	5303900278	Canada	British Colum	United State Florida	09/13/23	2/29/2023	515742.21	air	Master's	Business Adm	Shnell Collg	1/22/2023	05/03/25		
1025	1024	Carly	Haborn	F	29	Student	2536557288	Canada	Ontario	United State California	08/03/23	6/19/2023	517532.39	air	Master's	Computer Sc	Medical Coll	01/06/23	08/02/24		
1026	1025	Marshall	Thomason	M	32	Student	3134810544	Canada	Que/Dlnc	United State Connecticut	8/25/2023	02/03/24	512117.05	air	Master's	Computer Sc	The Art Instit	01/09/23	4/17/2024		
1027	1026	Conor	Hulson	M	19	Student	1558049605	Canada	British Colum	United State Missouri	08/05/23	7/22/2024	523041.43	air	Master's	Psychology	Clemson Univ	01/01/23	04/07/25		
1028	1027	Cash	Gianrotti	M	22	Student	2255061622	Canada	Alberta	United State District of Co	9/14/2023	4/21/2025	510006.36	air	Master's	Business Adm	Florida Metr	1/21/2023	8/14/2024		
1029	1028	Tracie	Tummasutti	M	19	Student	5377673788	Canada	Alberta	United State Kansas	8/14/2023	1/27/2025	519912.86	air	Master's	Computer Sc	Georgia Metr	1/16/2023	3/28/2024		
1030	1029	Amelrose	Bee	M	26	Student	9849701593	Canada	Ontario	United State Texas	09/07/23	5/21/2023	515386.51	road	Master's	Computer Sc	Lefgh Univ	1/22/2023	10/19/2024		
1031	1030	Carlin	Nancarrow	F	28	Student	4881132998	Canada	Que/Dlnc	United State New York	08/07/23	5/26/2024	514072.61	air	Master's	Computer Sc	Institute of I	1/23/2023	2/16/2024		
1032	1031	Auguste	Vlashev	F	33	Student	7272515659	Canada	Saskatchewan	United State Michigan	08/10/23	10/05/24	515242.77	air	Master's	Computer Sc	Institute of C	1/15/2023	5/14/2025		
1033	1032	Perren	Bleue	M	26	Student	4807171779	Canada	New Brunsw	United State Washington	09/08/23	11/16/2024	511759.08	air	Master's	Computer Sc	Gratz Collg	1/15/2023	9/25/2024		
1034	1033	Ritche	Scamadiue	M	22	Student	7602221966	Canada	Que/Dlnc	United State Nebraska	09/09/23	10/10/2024	524684.82	air	Master's	Computer Sc	California St	1/14/2023	3/24/2025		
1035	1034	Milka	Scutter	F	33	Student	2448581165	Canada	Ontario	United State Illinois	08/03/23	02/11/25	523324.70	air	Master's	Computer Sc	Southeastern	1/18/2023	02/09/25		
1036	1035	Delphine	Foxton	F	32	Student	788788632	Canada	Ontario	United State New York	9/13/2023	8/28/2024	523467.89	air	Master's	Computer Sc	Berthany Coll	1/13/2023	6/16/2024		
1037	1036	Onida	Tufanini	F	34	Student	9814483297	Canada	Manitob	United State Florida	8/14/2023	8/13/2024	514283.87	road	Master's	Computer Sc	St. George's	01/01/23	12/10/2023		
1038	1037	Brian	Paula	M	34	Student	1393178358	Canada	Ontario	United State Kentucky	09/07/23	1/25/2023	510814.21	air	Master's	Computer Sc	University of I	1/20/2023	01/12/24		
1039	1038	Chane	Hughes	M	22	Student	998799334	Canada	Que/Dlnc	United State Washington	8/14/2023	3/23/2025	513402.76	road	Master's	Computer Sc	Georgia Sch	1/21/2023	10/25/2024		
1040	1039	Renzold	Klein	M	27	Student	9010870299	Canada	Ontario	United State Illinois	08/10/23	08/12/23	514761.80	air	Master's	Psychology	DeVry Instit	01/10/23	1/26/2024		
1041	1040	Pip	Maddren	M	18	Student	5386483165	Canada	Que/Dlnc	United State North Caroli	8/19/2023	4/30/2024	513371.80	road	Master's	Business Adm	Longwood C	1/16/2023	05/01/25		
1042	1041	Cirilo	Berrlin	M	19	Student	1001518909	Canada	Ontario	United State Kentucky	8/20/2023	08/08/23	518395.17	air	Master's	Computer Sc	State Univer	01/02/23	8/24/2024		
1043	1042	Emory	Bernardo	M	29	Student	6331504132	Canada	Que/Dlnc	United State Illinois	08/12/23	12/01/23	515187.12	air	Master's	Business Adm	Northwest C	1/23/2023	5/16/2024		
1044	1043	Welbee	Lytell	M	20	Student	1969746366	Canada	Que/Dlnc	United State North Caroli	8/21/2023	08/01/24	524275.81	road	Master's	Computer Sc	Florida Metr	1/22/2023	1/27/2025		
1045	1044	Elbert	Yersin	M	23	Student	4657460413	Canada	Ontario	United State Wisconsin	8/19/2023	6/21/2024	518025.38	road	Master's	Computer Sc	Southwestern	1/14/2023	11/25/2024		
1046	1045	Norton	Schubert	M	25	Student	8721719017	Canada	Ontario	United State North Caroli	08/08/23	08/01/23	524197.72	road	Master's	Business Adm	Medaille Coll	01/09/23	5/16/2024		
1047	1046	Vicky	Lamprecht	F	23	Student	1444691099	Canada	Alberta	United State Iowa	08/02/23	4/19/2025	522733.77	air	Master's	Computer Sc	Pacific Univer	01/09/23	09/11/24		
1048	1047	Marton	O'Cloney	M	24	Student	3137128362	Canada	Prince Edward	United State Colorado	09/12/23	1/30/2025	510910.30	road	Master's	Computer Sc	Ashland Univ	01/02/23	12/23/2023		
1049	1048	Worth	Brisbane	M	31	Student	7983874359	Canada	Alberta	United State Texas	09/01/23	3/20/2024	512834.31	air	Master's	Psychology	Elizabeth Coll	1/20/2023	4/16/2024		
1050	1049	Harris	Isard	M	29	Student	6601434484	Canada	Que/Dlnc	United State New York	08/01/23	1/26/2023	517320.10	road	Master's	Computer Sc	University of	01/05/23	5/13/2024		
1051	1050	Seumas	Fridle	M	33	Student	5813301866	Canada	Saskatchewan	United State Pennsylvania	08/06/23	01/01/24	518412.18	air	Master's	Computer Sc	Adelphi Univ	01/03/23	12/26/2024		
1052	1051	Vina	Trainer	F	24	Student	9538351745	Canada	Que/Dlnc	United State Iowa	09/05/23	06/10/23	510206.22	air	Master's	Business Adm	Alice Lloyd C	1/13/2023	7/15/2024		
1053	1052	Gasper	Harrison	M	24	Student	9910957418	Canada	Alberta	United State California	08/06/23	10/15/2023	519069.49	road	Master's	Computer Sc	Friehburg Univ	1/27/2023	1/25/2025		
1054	1053	Darcy	Powd	F	26	Student	8209321836	Canada	Newfoundlan	United State Michigan	8/13/2023	11/30/2024	517095.48	air	Master's	Computer Sc	Kent State U	1/29/2023	6/28/2024		

Implementation process:

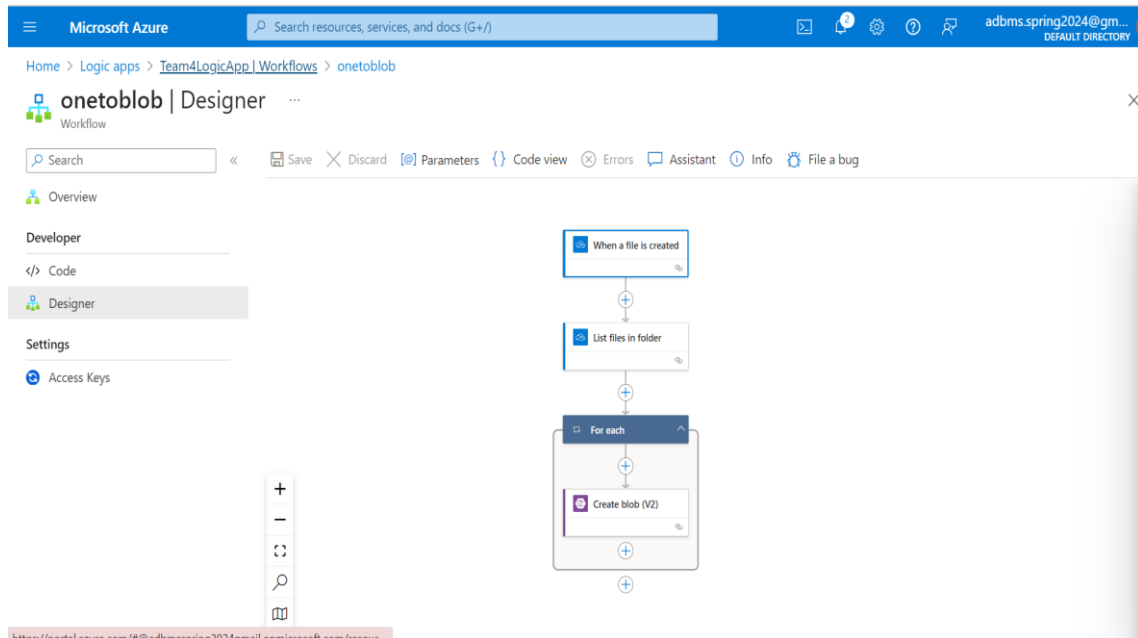


The implementation follows the architecture diagram submitted as part of P2. The process starts with ingesting the data present in the csv file located in the shared onedrive folder to Azure Blob Storage using Azure Logic app. To streamline data from OneDrive to a graph database, a Logic App triggers when a file is created in OneDrive and copies the file to Azure Blob Storage. The process is shown below.

Files present in OneDrive folder locaion



Created Logic App workflow to copy files from one drive to blob storage when a new file is added



Files is Azure Blob Storage :

The screenshot displays the Microsoft Azure Blob Storage interface for the 'team4data' container. The top navigation bar shows the path: Home > team4blobsa | Containers >. The main area includes a search bar, a list of actions (Upload, Change access level, Refresh, Delete, Change tier, Acquire lease, Break lease, View snapshots), and authentication details (Access key, Location: team4data). Below this is a search bar for blobs by prefix and a 'Show deleted blobs' toggle. A table lists the blobs with the following columns: Name, Modified, Access tier, Archive status, Blob type, and Size.

Name	Modified	Access tier	Archive status	Blob type	Size
<input type="checkbox"/> Business_Merged.csv	27/3/2024, 5:08:58 pm	Hot (Inferred)		Block blob	1.05
<input type="checkbox"/> InboundData.csv	27/3/2024, 5:08:58 pm	Hot (Inferred)		Block blob	1.05
<input type="checkbox"/> Pleasure_Merged.csv	27/3/2024, 5:08:58 pm	Hot (Inferred)		Block blob	12.9
<input type="checkbox"/> Student_Merged.csv	27/3/2024, 5:08:58 pm	Hot (Inferred)		Block blob	1.05

Once the data is loaded in the Blob Storage, Azure Databricks then processes and transforms the data, merging it into a single file suitable for graph database ingestion. Finally, the prepared data is loaded into the graph database and document database for analysis and querying. This automated pipeline requires careful setup and testing to ensure data integrity throughout the flow.

Processing data through databricks . We created a cluster for compute and used databricks notebook for the cleaning , transformation and loading of data into databases. We created a mount to access the data files in blod storage.

Team4DataTransformation Python

Requirement already satisfied: certifi=2017.4.17 in /databricks/python3/lib/python3.10/site-packages (from requests->forex_python) (2022.9.14)
 Requirement already satisfied: urllib3<1.27,=1.21.1 in /databricks/python3/lib/python3.10/site-packages (from requests->forex_python) (1.26.11)
 Requirement already satisfied: idna<4,=2.5 in /databricks/python3/lib/python3.10/site-packages (from requests->forex_python) (3.3)
 Note: you may need to restart the kernel using <code>dbutils.library.restartPython()</code> to use updated packages.

```

##Input all the required files from azure blob storage
df_Inbound = spark.read.option("header", "true").csv('/mnt/blobstorage/InboundData.csv')
df_Outbound_busi = spark.read.option("header", "true").csv('/mnt/blobstorage/Business_Merged.csv')
df_Outbound_plea = spark.read.option("header", "true").csv('/mnt/blobstorage/Pleasure_Merged.csv')
df_Outbound_stud = spark.read.option("header", "true").csv('/mnt/blobstorage/Student_Merged.csv')
  
```

(4) Spark Jobs

- df_Inbound: pyspark.sql.dataframe.DataFrame = [id: string, first_name: string ... 36 more fields]
- df_Outbound_busi: pyspark.sql.dataframe.DataFrame = [id: string, first_name: string ... 13 more fields]
- df_Outbound_plea: pyspark.sql.dataframe.DataFrame = [id: string, first_name: string ... 13 more fields]
- df_Outbound_stud: pyspark.sql.dataframe.DataFrame = [id: string, first_name: string ... 17 more fields]

```

## Initial data checks
print('Inbound row count :',df_Inbound.count())
print('Outbound Business row count :',df_Outbound_busi.count())
print('Outbound Pleasure row count :',df_Outbound_plea.count())
print('Outbound Student row count :',df_Outbound_stud.count())
print('\nDescribe Inbound')
print(df_Inbound.describe().toPandas())
print('\nDescribe Outbound Business')
print(df_Outbound_busi.describe().toPandas())
print('\nDescribe Outbound Pleasure')
print(df_Outbound_plea.describe().toPandas())
print('\nDescribe Outbound Student')
print(df_Outbound_stud.describe().toPandas())
  
```

(16) Spark Jobs

[5 rows x 16 columns]

Describe Outbound Pleasure

	summary	id	... estimated expenses	transportation mode
0	count	8500	...	8500
1	mean	4250.5	...	None
2	stddev	2453.882977378234	...	None
3	min	1	...	\$10,001.05
4	max	999	...	£14998.09

Cleaning and transformation

Microsoft Azure databricks Search data, notebooks, recents, and more... CTRL + P Team4DataTransformation Python

import calendar

```

# Define the age categorization function
def categorize_age_group(age):
    if age <= 17:
        return '0-17'
    elif age <= 30:
        return '18-30'
    elif age <= 45:
        return '31-45'
    elif age <= 60:
        return '46-60'
    else:
        return '61+'

# Apply the function to create a new column 'AgeGroup'
df['AgeGroup'] = df['Age'].apply(categorize_age_group)

# Convert 'Entry Date' to datetime to extract 'year' and 'month' as names
df['Entry Date'] = pd.to_datetime(df['Entry Date'])
df['year'] = df['Entry Date'].dt.year
df['month'] = df['Entry Date'].dt.month.apply(lambda x: calendar.month_name[x])
  
```


Screenshot of data loaded in Document db

Home > team4docdb

team4docdb | Data Explorer ☆ ...
Azure Cosmos DB account

Search

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Cost Management
Quick start
Notifications
Data Explorer
Settings
Features
Replicate data globally
Default consistency
Backup & Restore

Home Container - It... x

SELECT * FROM c

id /...

us Unit...

Load more

```
1 {
2   "country_name": "United States",
3   "id": "us",
4   "years": [
5     {
6       "months": [
7         {
8           "month": "August",
9           "states": [
10            {
11              "name": "Texas",
12              "total_arrivals": 1084,
13              "visas": [
14                {
15                  "AgeGroups": {
16                    "0-17": null,
17                    "18-30": 8,
18                    "31-45": 9,
19                    "46-60": 5,
20                    "61-75": 3,
21                    "76-90": 1
22                  }
23                }
24              ]
25            }
26          ]
27        }
28      ]
29    }
30  ]
31}
```

Using same notebook create vertices and edges and then Loaded into Cosmos Graph DB

Team4DataTransformation Python ☆

File Edit View Run Help Last edit was 3 minutes ago New cell UI: ON ▾ ▶ Run all Team4DatabricksCluster ▾ ⌚ Schedule Share

05:20 PM (6s) 30 Python

```
# Select unique rows based on 'Id' to ensure each tourist is represented once
unique_tourists = df.drop_duplicates(subset=['Id'])

# Concatenate 'First Name' and 'Last Name' to form the full name
unique_tourists['FullName'] = unique_tourists['First Name'] + ' ' + unique_tourists['Last Name']

# Create the tourist vertices DataFrame with 'id' and 'FullName'
tourist_vertices = unique_tourists[['Id', 'FullName']].rename(columns={'Id': 'id', 'FullName': 'name'})

# Visa vertices - Using unique 'Visa Type'
visa_vertices = df['Visa Type'].drop_duplicates()
visa_vertices = visa_vertices.rename("visa_type")

# Country vertices - Using unique 'Country'
country_vertices = df['Country'].drop_duplicates()
country_vertices = country_vertices.rename("country_name")

# Preparing the data for creating edges
# Holds relationship data (between Tourist and Visa)
holds_edges = df[['Id', 'Visa Type']].drop_duplicates()

# Offers relationship data (between Country and Visa)
```

05:21 PM (1s) 33

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, concat_ws
from graphframes import GraphFrame
```

05:26 PM (4s) 34 Python

```
# Initialize Spark Session
spark = SparkSession.builder.appName("Graph Data Model").getOrCreate()

# Convert the vertices and edges to Spark DataFrames
vertices = spark.createDataFrame(vertices_pd)
visa_vertices_df = spark.createDataFrame(visa_vertices.to_frame(name='VisaType'))
country_vertices_df = spark.createDataFrame(country_vertices.to_frame(name='Country'))
holds_edges_df = spark.createDataFrame(holds_edges)
offers_edges_df = spark.createDataFrame(offers_edges)
is_from_visits_edges_df = spark.createDataFrame(is_from_visits_edges)

# Create an edges DataFrame with src and dst columns
edges = (holds_edges_df
        .withColumnRenamed("Id", "src")
        .withColumnRenamed("Visa Type", "dst")
        .union(offers_edges_df.withColumnRenamed("Country", "src")
        .withColumnRenamed("Visa Type", "dst")))
```

```
# Convert the vertices and edges to Spark DataFrames
vertices = spark.createDataFrame(vertices_pd)
visa_vertices_df = spark.createDataFrame(visa_vertices.to_frame(name='VisaType'))
country_vertices_df = spark.createDataFrame(country_vertices.to_frame(name='Country'))
holds_edges_df = spark.createDataFrame(holds_edges)
offers_edges_df = spark.createDataFrame(offers_edges)
is_from_visits_edges_df = spark.createDataFrame(is_from_visits_edges)

# Create an edges DataFrame with src and dst columns
edges = (holds_edges_df
        .withColumnRenamed("Id", "src")
        .withColumnRenamed("Visa Type", "dst")
        .unionByName(offers_edges_df.withColumnRenamed("Country", "src")
                    .withColumnRenamed("Visa Type", "dst"))
        .unionByName(is_from_visits_edges_df.withColumnRenamed("Id", "src")
                    .withColumnRenamed("Country", "dst")))

edges = edges.withColumnRenamed("src", "src")

# Create a GraphFrame
graph = GraphFrame(vertices, edges)
```

Import data into graph db

Team4DataTransformation Python ☆

File Edit View Run Help Last edit was now New cell UI: ON ▾

▶ Run all Team4DatabricksCluster ▾ Schedule

05:31 PM (56s) 37 Python

```
vertices.printSchema()

# Check the first few rows to ensure they look as expected
vertices.show()

# Write vertices DataFrame to Cosmos DB
vertices.write.format("cosmos.oltp") \
    .options(**config) \
    .option("spark.cosmos.write.strategy", "ItemOverwrite") \
    .option("spark.cosmos.write.bulk.enabled", "true") \
    .mode("Append") \
    .save()
```

▶ (1) Spark Jobs

[10000515]	Latisha Secret
[10001176]	Nathaniel Faichnie
[10001349]	Huntlee Newtown
[10001603]	Johnath Sawden
[10001756]	Sephira Hevner
[10001801]	Allard Flewett
[10001816]	Reena McQuilkin
[10002111]	Rosabelle Golborn
[10002265]	Bartholomeo Bamokin

Team4DataTransformation Python ☆

File Edit View Run Help Last edit was 1 minute ago New cell UI: ON ▾

▶ Run all Team4DatabricksCluster ▾ Schedule Share

1 minute ago (1m) 34 Python

```
from pyspark.sql.functions import monotonically_increasing_id

# Assuming 'edges' is your Spark DataFrame with the edges information

# If an 'id' column does not exist, create one with unique values
if "id" not in edges.columns:
    edges = edges.withColumn("id", monotonically_increasing_id().cast("string"))

# If 'id' exists but is not of string type, cast it to string
else:
    edges = edges.withColumn("id", col("id").cast("string"))

# Now write the edges DataFrame to Cosmos DB
edges.write.format("cosmos.oltp") \
    .options(**config) \
    .option("spark.cosmos.write.strategy", "ItemOverwrite") \
    .option("spark.cosmos.write.bulk.enabled", "true") \
    .mode("Append") \
    .save()
```

▶ (1) Spark Jobs

edges: pyspark.sql.dataframe.DataFrame = [src: string, dst: string ... 1 more field]

Loaded in graph db as vertices and edges

Home > team4graphdb

team4graphdb | Data Explorer

Azure Cosmos DB account

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Quick start
- Notifications
- Data Explorer

Settings

- Features
- Default consistency
- Backup & Restore
- Networking

APACHE GREMLIN API

- DATA
 - Team4GraphDB
 - Graph1
 - Graph
 - Settings
 - Stored Procedures
 - User Defined Functions
 - Triggers
 - Graph2
 - Graph
 - Settings
 - Stored Procedures
 - User Defined Functions
- NOTEBOOKS
 - Notebooks is currently not available. We are working on it.

Home | Graph

g.V().count()

Execute Gremlin Query

g.V().count()
g.V().count()

JSON Query Stats

```
[
  29777
]
```

Microsoft Azure

Search resources, services, and docs (S+)

ad8rm.spring2024@pm-select11983194

Home > team4graphdb

team4graphdb | Data Explorer

Azure Cosmos DB account

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Quick start
- Notifications
- Data Explorer

Settings

- Features
- Default consistency
- Backup & Restore
- Networking

Integrations

- Add Azure Function

Monitoring

- Insights
- Alerts
- Metrics

Log

APACHE GREMLIN API

Home | Graph

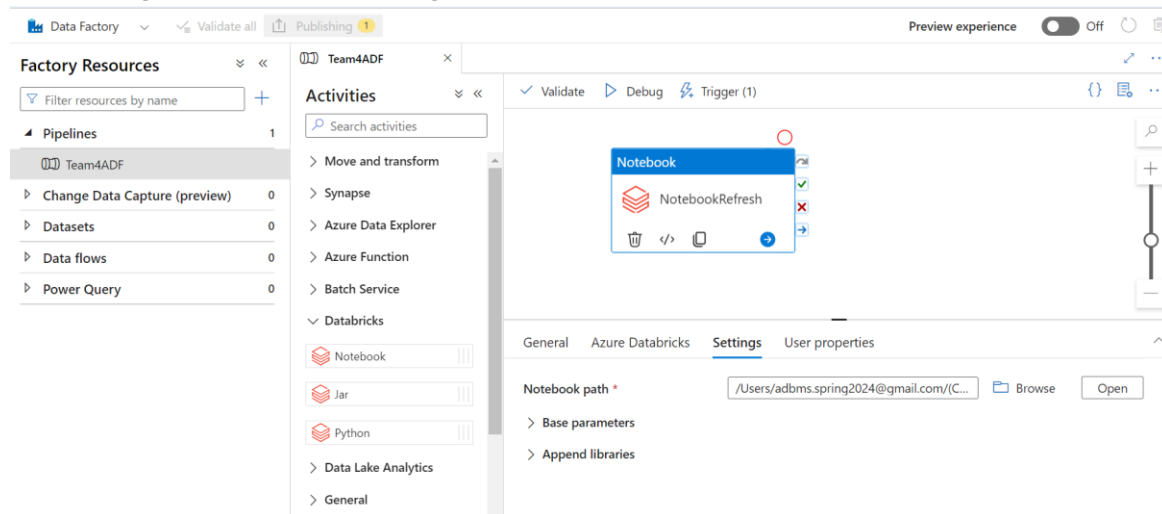
g.V()

Execute Gremlin Query

Results

- 10000348
- 10000411
- 10000515
- 10000516
- 10000517
- 10000518
- 10000519
- 10000520
- 10000521
- 10000522
- 10000523
- 10000524
- 10000525
- 10000526
- 10000527
- 10000528
- 10000529
- 10000530
- 10000531
- 10000532
- 10000533
- 10000534
- 10000535
- 10000536
- 10000537
- 10000538
- 10000539
- 10000540
- 10000541
- 10000542
- 10000543
- 10000544
- 10000545
- 10000546
- 10000547
- 10000548
- 10000549
- 10000550
- 10000551
- 10000552
- 10000553
- 10000554
- 10000555
- 10000556
- 10000557
- 10000558
- 10000559
- 10000560
- 10000561
- 10000562
- 10000563
- 10000564
- 10000565
- 10000566
- 10000567
- 10000568
- 10000569
- 10000570
- 10000571
- 10000572
- 10000573
- 10000574
- 10000575
- 10000576
- 10000577
- 10000578
- 10000579
- 10000580
- 10000581
- 10000582
- 10000583
- 10000584
- 10000585
- 10000586
- 10000587
- 10000588
- 10000589
- 10000590
- 10000591
- 10000592
- 10000593
- 10000594
- 10000595
- 10000596
- 10000597
- 10000598
- 10000599
- 10000600
- 10000601
- 10000602
- 10000603
- 10000604
- 10000605
- 10000606
- 10000607
- 10000608
- 10000609
- 10000610
- 10000611
- 10000612
- 10000613
- 10000614
- 10000615
- 10000616
- 10000617
- 10000618
- 10000619
- 10000620
- 10000621
- 10000622
- 10000623
- 10000624
- 10000625
- 10000626
- 10000627
- 10000628
- 10000629
- 10000630
- 10000631
- 10000632
- 10000633
- 10000634
- 10000635
- 10000636
- 10000637
- 10000638
- 10000639
- 10000640
- 10000641
- 10000642
- 10000643
- 10000644
- 10000645
- 10000646
- 10000647
- 10000648
- 10000649
- 10000650
- 10000651
- 10000652
- 10000653
- 10000654
- 10000655
- 10000656
- 10000657
- 10000658
- 10000659
- 10000660
- 10000661
- 10000662
- 10000663
- 10000664
- 10000665
- 10000666
- 10000667
- 10000668
- 10000669
- 10000670
- 10000671
- 10000672
- 10000673
- 10000674
- 10000675
- 10000676
- 10000677
- 10000678
- 10000679
- 10000680
- 10000681
- 10000682
- 10000683
- 10000684
- 10000685
- 10000686
- 10000687
- 10000688
- 10000689
- 10000690
- 10000691
- 10000692
- 10000693
- 10000694
- 10000695
- 10000696
- 10000697
- 10000698
- 10000699
- 10000700
- 10000701
- 10000702
- 10000703
- 10000704
- 10000705
- 10000706
- 10000707
- 10000708
- 10000709
- 10000710
- 10000711
- 10000712
- 10000713
- 10000714
- 10000715
- 10000716
- 10000717
- 10000718
- 10000719
- 10000720
- 10000721
- 10000722
- 10000723
- 10000724
- 10000725
- 10000726
- 10000727
- 10000728
- 10000729
- 10000730
- 10000731
- 10000732
- 10000733
- 10000734
- 10000735
- 10000736
- 10000737
- 10000738
- 10000739
- 10000740
- 10000741
- 10000742
- 10000743
- 10000744
- 10000745
- 10000746
- 10000747
- 10000748
- 10000749
- 10000750
- 10000751
- 10000752
- 10000753
- 10000754
- 10000755
- 10000756
- 10000757
- 10000758
- 10000759
- 10000760
- 10000761
- 10000762
- 10000763
- 10000764
- 10000765
- 10000766
- 10000767
- 10000768
- 10000769
- 10000770
- 10000771
- 10000772
- 10000773
- 10000774
- 10000775
- 10000776
- 10000777
- 10000778
- 10000779
- 10000780
- 10000781
- 10000782
- 10000783
- 10000784
- 10000785
- 10000786
- 10000787
- 10000788
- 10000789
- 10000790
- 10000791
- 10000792
- 10000793
- 10000794
- 10000795
- 10000796
- 10000797
- 10000798
- 10000799
- 10000800
- 10000801
- 10000802
- 10000803
- 10000804
- 10000805
- 10000806
- 10000807
- 10000808
- 10000809
- 10000810
- 10000811
- 10000812
- 10000813
- 10000814
- 10000815
- 10000816
- 10000817
- 10000818
- 10000819
- 10000820
- 10000821
- 10000822
- 10000823
- 10000824
- 10000825
- 10000826
- 10000827
- 10000828
- 10000829
- 10000830
- 10000831
- 10000832
- 10000833
- 10000834
- 10000835
- 10000836
- 10000837
- 10000838
- 10000839
- 10000840
- 10000841
- 10000842
- 10000843
- 10000844
- 10000845
- 10000846
- 10000847
- 10000848
- 10000849
- 10000850
- 10000851
- 10000852
- 10000853
- 10000854
- 10000855
- 10000856
- 10000857
- 10000858
- 10000859
- 10000860
- 10000861
- 10000862
- 10000863
- 10000864
- 10000865
- 10000866
- 10000867
- 10000868
- 10000869
- 10000870
- 10000871
- 10000872
- 10000873
- 10000874
- 10000875
- 10000876
- 10000877
- 10000878
- 10000879
- 10000880
- 10000881
- 10000882
- 10000883
- 10000884
- 10000885
- 10000886
- 10000887
- 10000888
- 10000889
- 10000890
- 10000891
- 10000892
- 10000893
- 10000894
- 10000895
- 10000896
- 10000897
- 10000898
- 10000899
- 10000900
- 10000901
- 10000902
- 10000903
- 10000904
- 10000905
- 10000906
- 10000907
- 10000908
- 10000909
- 10000910
- 10000911
- 10000912
- 10000913
- 10000914
- 10000915
- 10000916
- 10000917
- 10000918
- 10000919
- 10000920
- 10000921
- 10000922
- 10000923
- 10000924
- 10000925
- 10000926
- 10000927
- 10000928
- 10000929
- 10000930
- 10000931
- 10000932
- 10000933
- 10000934
- 10000935
- 10000936
- 10000937
- 10000938
- 10000939
- 10000940
- 10000941
- 10000942
- 10000943
- 10000944
- 10000945
- 10000946
- 10000947
- 10000948
- 10000949
- 10000950
- 10000951
- 10000952
- 10000953
- 10000954
- 10000955
- 10000956
- 10000957
- 10000958
- 10000959
- 10000960
- 10000961
- 10000962
- 10000963
- 10000964
- 10000965
- 10000966
- 10000967
- 10000968
- 10000969
- 10000970
- 10000971
- 10000972
- 10000973
- 10000974
- 10000975
- 10000976
- 10000977
- 10000978
- 10000979
- 10000980
- 10000981
- 10000982
- 10000983
- 10000984
- 10000985
- 10000986
- 10000987
- 10000988
- 10000989
- 10000990
- 10000991
- 10000992
- 10000993
- 10000994
- 10000995
- 10000996
- 10000997
- 10000998
- 10000999
- 10001000
- 10001001
- 10001002
- 10001003
- 10001004
- 10001005
- 10001006
- 10001007
- 10001008
- 10001009
- 10001010
- 10001011
- 10001012
- 10001013
- 10001014
- 10001015
- 10001016
- 10001017
- 10001018
- 10001019
- 10001020
- 10001021
- 10001022
- 10001023
- 10001024
- 10001025
- 10001026
- 10001027
- 10001028
- 10001029
- 10001030
- 10001031
- 10001032
- 10001033
- 10001034
- 10001035
- 10001036
- 10001037
- 10001038
- 10001039
- 10001040
- 10001041
- 10001042
- 10001043
- 10001044
- 10001045
- 10001046
- 10001047
- 10001048
- 10001049
- 10001050
- 10001051
- 10001052
- 10001053
- 10001054
- 10001055
- 10001056
- 10001057
- 10001058
- 10001059
- 10001060
- 10001061
- 10001062
- 10001063
- 10001064
- 10001065
- 10001066
- 10001067
- 10001068
- 10001069
- 10001070
- 10001071
- 10001072
- 10001073
- 10001074
- 10001075
- 10001076
- 10001077
- 10001078
- 10001079
- 10001080
- 10001081
- 10001082
- 10001083
- 10001084
- 10001085
- 10001086
- 10001087
- 10001088
- 10001089
- 10001090
- 10001091
- 10001092
- 10001093
- 10001094
- 10001095
- 10001096
- 10001097
- 10001098
- 10001099
- 10001100
- 10001101
- 10001102
- 10001103
- 10001104
- 10001105
- 10001106
- 10001107
- 10001108
- 10001109
- 10001110
- 10001111
- 10001112
- 10001113
- 10001114
- 10001115
- 10001116
- 10001117
- 10001118
- 10001119
- 10001120
- 10001121
- 10001122
- 10001123
- 10001124
- 10001125
- 10001126
- 10001127
- 10001128
- 10001129
- 10001130
- 10001131
- 10001132
- 10001133
- 10001134
- 10001135
- 10001136
- 10001137
- 10001138
- 10001139
- 10001140
- 10001141
- 10001142
- 10001143
- 10001144
- 10001145
- 10001146
- 10001147
- 10001148
- 10001149
- 10001150
- 10001151
- 10001152
- 10001153
- 10001154
- 10001155
- 10001156
- 10001157
- 10001158
- 10001159
- 10001160
- 10001161
- 10001162
- 10001163
- 10001164
- 10001165
- 10001166
- 10001167
- 10001168
- 10001169
- 10001170
- 10001171
- 10001172
- 10001173
- 10001174
- 10001175
- 10001176
- 10001177
- 10001178
- 10001179
- 10001180
- 10001181
- 10001182
- 10001183
- 10001184
- 10001185
- 10001186
- 10001187
- 10001188
- 10001189
- 10001190
- 10001191
- 10001192
- 10001193
- 10001194
- 10001195
- 10001196
- 10001197
- 10001198
- 10001199
- 10001200
- 10001201
- 10001202
- 10001203
- 10001204
- 10001205
- 10001206
- 10001207
- 10001208
- 10001209
- 10001210
- 10001211
- 10001212
- 10001213
- 10001214
- 10001215
- 10001216
- 10001217
- 10001218
- 10001219
- 10001220
- 10001221
- 10001222
- 10001223
- 10001224
- 10001225
- 10001226
- 10001227
- 10001228
- 10001229
- 10001230
- 10001231
- 10001232
- 10001233
- 10001234
- 10001235
- 10001236
- 10001237
- 10001238
- 10001239
- 10001240
- 10001241
- 10001242
- 10001243
- 10001244
- 10001245
- 10001246
- 10001247
- 10001248
- 10001249
- 10001250
- 10001251
- 10001252
- 10001253
- 10001254
- 10001255
- 10001256
- 10001257
- 10001258
- 10001259
- 10001260
- 10001261
- 10001262
- 10001263
- 10001264
- 10001265
- 10001266
- 10001267
- 10001268
- 10001269
- 10001270
- 10001271
- 10001272
- 10001273
- 10001274
- 10001275
- 10001276
- 10001277
- 10001278
- 10001279
- 10001280
- 10001281
- 10001282
- 10001283
- 10001284
- 10001285
- 10001286
- 10001287
- 10001288
- 10001289
- 10001290
- 10001291
- 10001292
- 10001293
- 10001294
- 10001295
- 10001296
- 10001297
- 10001298
- 10001299
- 10001300
- 10001301
- 10001302
- 10001303
- 10001304
- 10001305
- 10001306
- 10001307
- 10001308
- 10001309
- 10001310
- 10001311
- 10001312
- 10001313
- 10001314
- 10001315
- 10001316
- 10001317
- 10001318
- 10001319
- 10001320
- 10001321
- 10001322
- 10001323
- 10001324
- 10001325
- 10001326
- 10001327
- 10001328
- 10001329
- 10001330
- 10001331
- 10001332
- 10001333
- 10001334
- 10001335
- 10001336
- 10001337
- 10001338
- 10001339
- 10001340
- 10001341
- 10001342
- 10001343
- 10001344
- 10001345
- 10001346
- 10001347
- 10001348
- 10001349
- 10001350
- 10001351
- 10001352
- 10001353
- 10001354
- 10001355
- 10001356
- 10001357
- 10001358
- 10001359
- 10001360
- 10001361
- 10001362
- 10001363
- 10001364
- 10001365
- 10001366
- 10001367
- 10001368
- 10001369
- 10001370
- 10001371
- 10001372
- 10001373
- 10001374
- 10001375
- 10001376
- 10001377
- 10001378
- 10001379
- 10001380
- 10001381
- 10001382
- 10001383
- 10001384
- 10001385
- 10001386
- 10001387
- 10001388
- 10001389
- 10001390
- 10001391
- 10001392
- 10001393
- 10001394
- 10001395
- 10001396
- 10001397
- 10001398
- 10001399
- 10001400
- 10001401
- 10001402
- 10001403
- 10001404
- 10001405
- 10001406
- 10001407
- 10001408
- 10001409
- 10001410
- 10001411
- 1000

Scheduling notebook for run using ADF :



Once the data is loaded in their respective databases then the notebook is scheduled to run every day to have the latest data loaded in the databases. For this, we have used Azure Data Factory on a schedule trigger. Refresh can be done using Databricks, but we choose ADF because it provides robust monitoring and logging capabilities. It tracks the status of each pipeline run and can log detailed execution information, which can be viewed directly in the Azure Portal or consumed via Azure Monitor, potentially leading to better cost management by shutting down clusters when not in use or scaling them according to the workload.