

Azure Data Engineering End to End Project

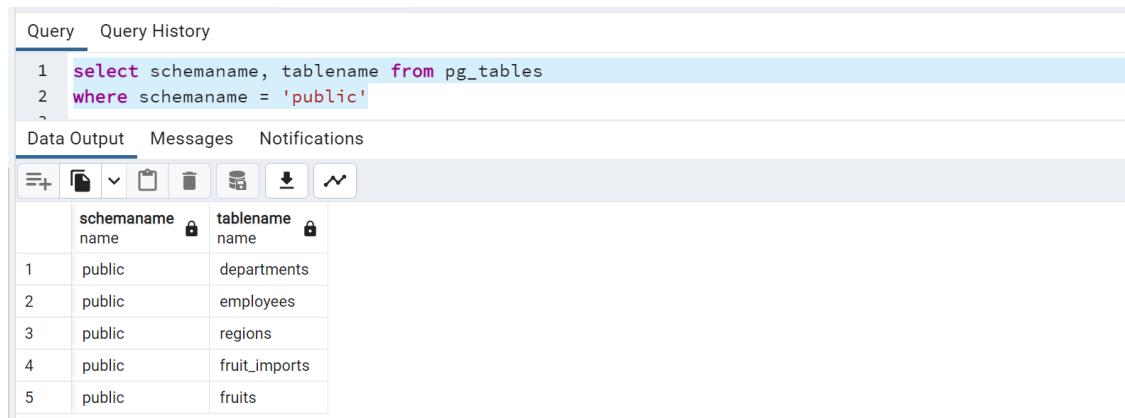
By Praveen Samudrala

Contents

Problem Statement:	3
Required Resources:.....	4
Step 1 – Connecting On-premise PostgreSQL database to Azure using Self-Hosted Runtime.....	5
Step 2 – Using Data Factory for Ingestion and loading the data to ADLS.	6
Step 3 – Using Azure Databricks for data transformation and cleaning.	8
Step 4 – Data Transformation in Azure Databricks.....	14
Step 5 – Loading Delta Tables from Databricks to Azure Synapse as Data Warehouse	18
Step 6 – Provide Business Analytics using PowerBI with data in Synapse DB.....	22
Step 7 – Scheduling the Data Pipeline to run once a day	25
References:.....	27

Problem Statement: Migrate On-premise PostgreSQL database to Azure cloud Data Warehouse from where the structured data can be used for Analytics using PowerBI.

Tables in PostgreSQL to migrate

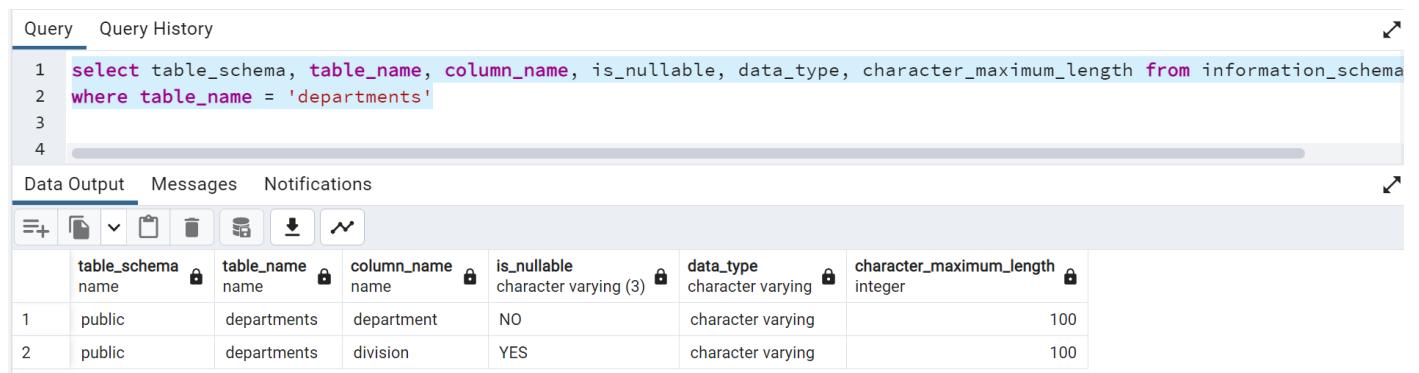


```
Query  Query History
1 select schemaname, tablename from pg_tables
2 where schemaname = 'public'
3
Data Output  Messages  Notifications

```

	schemaname name	tablename name
1	public	departments
2	public	employees
3	public	regions
4	public	fruit_imports
5	public	fruits

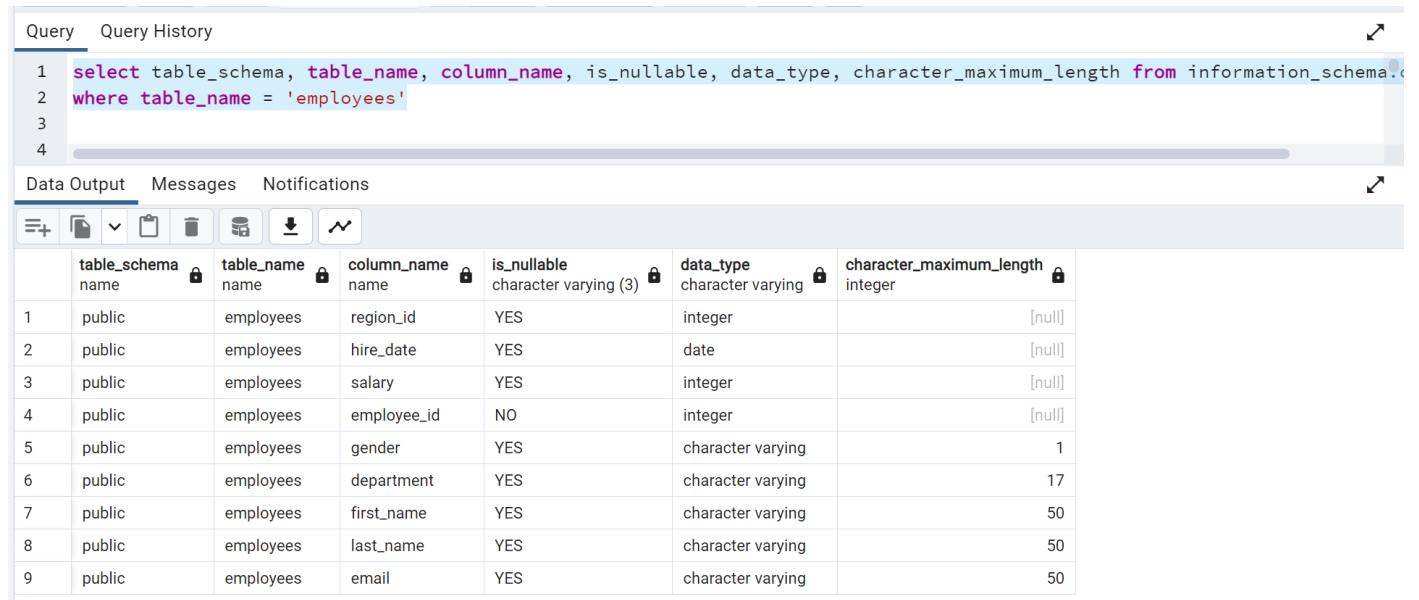
Table Schemas:



```
Query  Query History
1 select table_schema, table_name, column_name, is_nullable, data_type, character_maximum_length from information_schema
2 where table_name = 'departments'
3
4
Data Output  Messages  Notifications

```

	table_schema name	table_name name	column_name name	is_nullable character varying (3)	data_type character varying	character_maximum_length integer
1	public	departments	department	NO	character varying	100
2	public	departments	division	YES	character varying	100



```
Query  Query History
1 select table_schema, table_name, column_name, is_nullable, data_type, character_maximum_length from information_schema
2 where table_name = 'employees'
3
4
Data Output  Messages  Notifications

```

	table_schema name	table_name name	column_name name	is_nullable character varying (3)	data_type character varying	character_maximum_length integer
1	public	employees	region_id	YES	integer	[null]
2	public	employees	hire_date	YES	date	[null]
3	public	employees	salary	YES	integer	[null]
4	public	employees	employee_id	NO	integer	[null]
5	public	employees	gender	YES	character varying	1
6	public	employees	department	YES	character varying	17
7	public	employees	first_name	YES	character varying	50
8	public	employees	last_name	YES	character varying	50
9	public	employees	email	YES	character varying	50

Query Query History

```
1 select table_schema, table_name, column_name, is_nullable, data_type, character_maximum_length from information_schema.columns
2 where table_name = 'regions'
3
4
```

Data Output Messages Notifications

table_schema	table_name	column_name	is_nullable	data_type	character_maximum_length
1 public	regions	region_id	NO	integer	[null]
2 public	regions	region	YES	character varying	20
3 public	regions	country	YES	character varying	20

Query Query History

```
1 select table_schema, table_name, column_name, is_nullable, data_type, character_maximum_length from information_schema.columns
2 where table_name = 'fruit_imports'
3
4
```

Data Output Messages Notifications

table_schema	table_name	column_name	is_nullable	data_type	character_maximum_length
1 public	fruit_imports	id	YES	integer	[null]
2 public	fruit_imports	supply	YES	integer	[null]
3 public	fruit_imports	cost_per_unit	YES	numeric	[null]
4 public	fruit_imports	name	YES	character varying	20
5 public	fruit_imports	season	YES	character varying	10
6 public	fruit_imports	state	YES	character varying	20

Required Resources:

1. Azure Data Factory
2. Azure Databricks
3. Azure Synapse
4. Azure Data Lake Gen 2
5. Azure Key Vault

Home > Resource groups >

rg-praveen Resource group

Overview

Essentials

Subscription (move) : [Azure for Students](#) Deployments : [7 Succeeded](#)
Subscription ID : 13f6815e-9c47-445e-8f41-8bf80a581450 Location : East US
Tags (edit) : [Add tags](#)

Resources Recommendations

Filter for any field... Type equals all Location equals all Add filter

Showing 1 to 5 of 5 records. Show hidden types

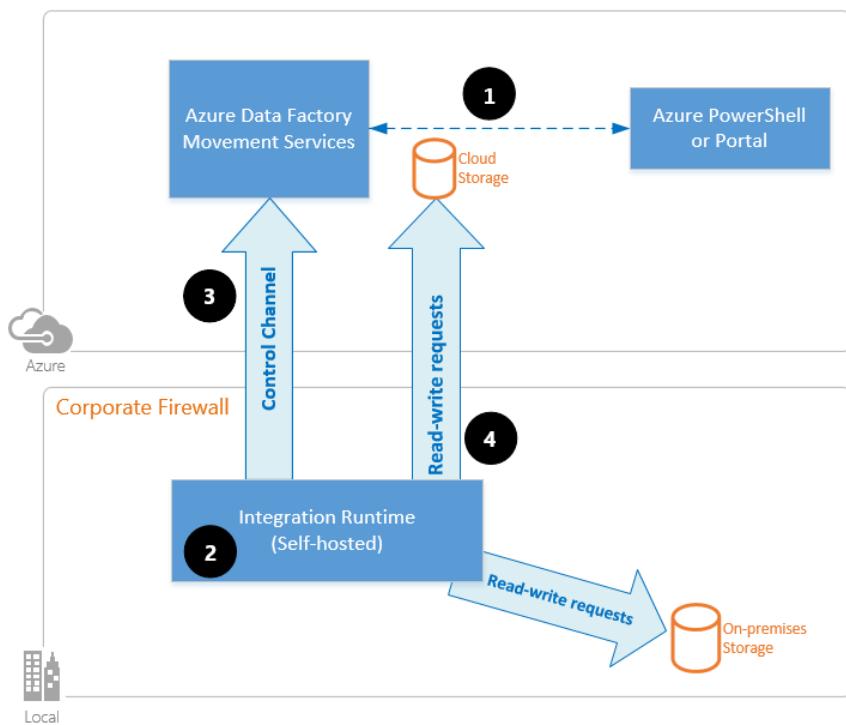
Name	Type	Location
ADB-praveen-02	Azure Databricks Service	East US
adspgraveen02	Storage account	East US
keyvault-praveen-02	Key vault	East US
praveen-ADF-02	Data factory (V2)	East US
synapse-praveen-02	Synapse workspace	East US

Create Bronze, Silver and Gold containers in ADLS

Name	Last modified	Anonymous access level	Lease state
\$logs	9/18/2023, 4:56:09 PM	Private	Available
bronze	9/18/2023, 8:49:46 PM	Private	Available
filesystem-02	9/18/2023, 5:00:13 PM	Private	Available
gold	9/18/2023, 8:49:59 PM	Private	Available
silver	9/18/2023, 8:49:55 PM	Private	Available

Step 1 – Connecting On-premise PostgreSQL database to Azure using Self-Hosted Runtime.

1.1 Create a new Integration Runtime - Self hosted, Install Azure Self Hosted Runtime



Name	Type	Sub-type
AutoResolveIntegrationR...	Azure	Public

Integration runtime setup

Settings Nodes Auto update Sharing Links

Install integration runtime on Windows machine or add further nodes using the Authentication Key.

Name: SHIR

Option 1: Express setup
Click here to launch the express setup for this computer

Option 2: Manual setup
Step 1: Download and install integration runtime
Step 2: Use this key to register your integration runtime

1.2 Give user permission to Key Vault resource

Name	Object ID	Type
Praveen Samudrala	9b386de3-fbe8-4a0d-a78a-8bbc2fb059...	User

1.3 Setup PostgreSQL username password as Secrets in Azure Key Vault

Name	Type	Status	Expiration date
password-synapse		✓ Enabled	
password-postgresql		✓ Enabled	

Creating the secret 'password-synapse'.
The secret 'password-synapse' has been successfully created.

Step 2 – Using Data Factory for Ingestion and loading the data to ADLS.

2.1 Create Linked Service from ADF to Key Vault to access passwords

The screenshot shows the Microsoft Azure Data Factory interface. On the left, a sidebar lists various settings like General, Connections, and Linked services. The main area is titled 'Linked services' and contains a form for creating a new linked service to 'Azure Key Vault'. The 'Name' field is set to 'AzureKeyVault1'. Under 'Azure key vault selection method', the 'From Azure subscription' option is selected. The 'Azure subscription' dropdown shows 'Azure for Students (13f6815e-9c47-445e-8f41-8bf80a581450)'. The 'Azure key vault name' dropdown is set to 'keyvault-praveen-02'. The 'Authentication method' is set to 'System Assigned Managed Identity'. A note at the bottom indicates that managed identity access has been granted. The 'Create' button is visible at the bottom right, with a success message 'Connection successful' and a 'Test connection' link.

2.2 Allow ADF and Synapse to access Key Vault Secrets

The screenshot shows the 'Access control (IAM)' section of the Azure Key Vault 'keyvault-praveen-02' page. It displays the 'Add role assignment' wizard. The 'Members' tab is selected, showing a 'Selected role' of 'Key Vault Secrets User'. Under 'Assign access to', 'Managed identity' is chosen. The 'Members' section shows two entries: 'praveen-ADF-02' and 'synapse-praveen-02', both associated with the 'Azure for Students' subscription. A modal window titled 'Select managed identities' is open, listing the same two identities under 'Selected members'. Buttons for 'Select' and 'Close' are at the bottom of the modal.

The screenshot shows the 'Access control (IAM)' section of the Azure Key Vault. It displays 7 role assignments for this subscription, with a total capacity of 4000. The table lists three entries under 'Key Vault Secrets User':

Name	Type	Role	Scope	Condition
Praveen Samudrala	User	Key Vault Administrator	This resource	None
praveen-ADF-02	Data Factory	Key Vault Secrets User	This resource	None
synapse-praveen-02	App	Key Vault Secrets User	This resource	None

2.3 Create Linked Service from ADF to on-premise Data Factory

The screenshot shows the 'Linked services' blade in the Azure Data Factory. A new linked service is being created for 'PostgreSQL'. The configuration includes:

- Server name:** localhost
- Port:** 5432
- Database name:** course_data
- User name:** postgres
- Password:** (redacted)
- Encryption method:** No encryption

A successful connection test is shown at the bottom right.

Step 3 – Using Azure Databricks for data transformation and cleaning.

3.1 – Create a Pipeline starting with ‘Lookup’ activity to look for all tables satisfying a custom query.

Create a Source Dataset

The screenshot shows the Azure Data Factory pipeline editor. A 'Lookup' activity is selected, and its properties are being configured in the 'Set properties' panel:

- Name:** PostgreSqlTables
- Linked service:** On_prem_PostgreSQL
- Connect via integration runtime:** SHIR
- Table name:** Select...

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A pipeline named 'pipeline1' is selected. On the left, the 'Activities' sidebar is open, showing various options like Move and transform, Copy data, Data flow, Synapse, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, and HDInsight. In the main workspace, a 'Lookup' activity is selected. The 'Settings' tab is active, showing the 'Source dataset' as 'PostgreSQLTables'. Under 'Query', the following SQL query is defined:

```
select schemaname,tablename from pg_tables where schemaname='public'
```

Custom Query:

Select schemaname, tablename from pg_tables where schemaname='public'

Fetched all tables from PostgreSQL DB

The screenshot shows the Microsoft Azure Data Factory pipeline editor. The 'Preview data' section is displayed, showing the results of a 'Lookup' activity. The linked service is 'On_prem_PostgreSQL'. The results table contains four rows:

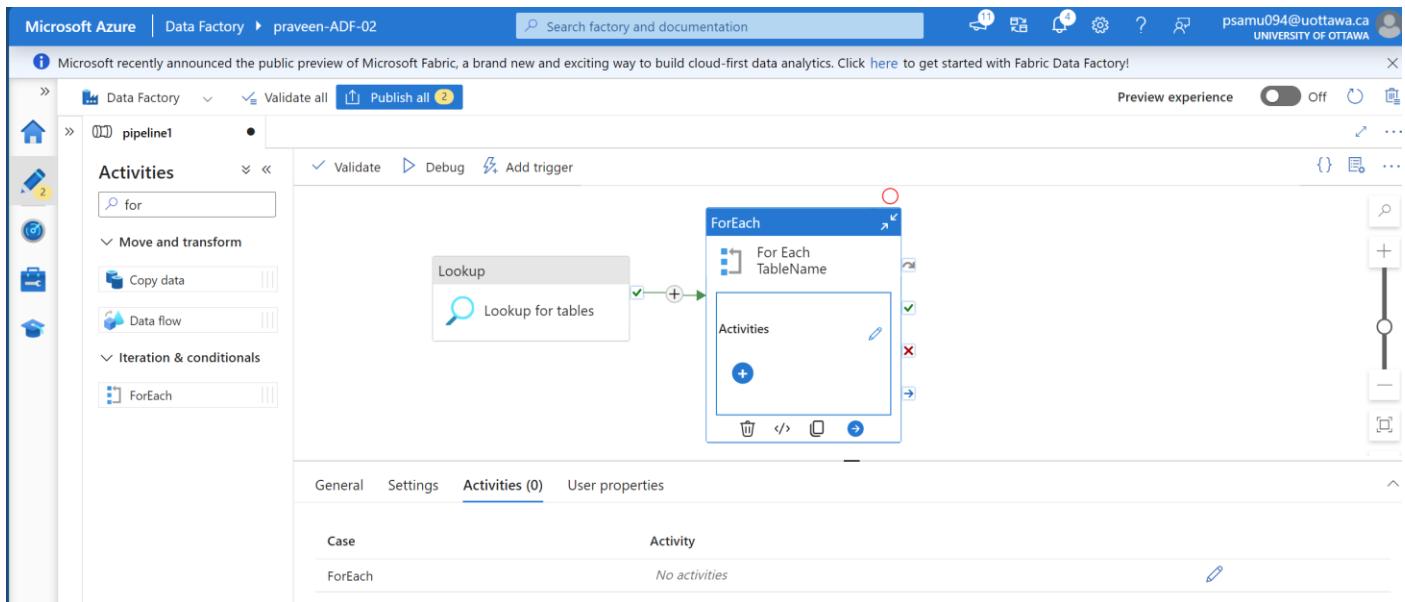
	schemaname	tablename
1	public	departments
2	public	employees
3	public	regions
4	public	fruit_imports

Result from Lookup activity result in a JSON object telling the count and values of the result. In our case, the count is 4, values is an array of scshename,tablename.

3.2 Add 'ForEach' activity after Lookup activity and add the dynamic expression

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A 'Lookup' activity is connected to a 'ForEach' activity. The 'ForEach' activity has a single child activity named 'TableNa'. In the 'Pipeline expression builder' panel, the following dynamic expression is defined:

```
@activity('Lookup for tables').output.value
```



Configure the Activity – Lookup activity returns the table names, for each table, ‘ForEach’ activity need to fetch all data from each table.

Edit the ForEach activity, then setup a Copy activity

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A 'Copy data' activity is selected within a 'ForEach' loop. The 'Source' tab is active, showing the configuration for the source dataset. The 'Set properties' panel on the right is open, showing the following details:

- Name:** Postgresql_copy
- Linked service:** On_prem_PostgreSQL
- Connect via integration runtime:** SHIR
- Table name:** Select...

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A 'Copy data' activity is selected within a 'ForEach' loop. The 'Source' tab is active, showing the configuration for the source dataset. The 'Pipeline expression builder' panel on the right is open, displaying the following dynamic content:

```

@{concat('select * from ', item().schemaname, '.', item().tablename)}

```

The 'Source dataset' dropdown is set to 'Postgresql_copy'. The 'Query' field contains the expression:

```

@{concat('select * from ', item().schemaname, '.', item().tablename)}

```

Create a Sink – ADLS Gen 2 with Parquet Format. Create a new Linked service to ADLS, then create a Sink Dataset.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A pipeline named 'pipeline1' is open, containing a 'ForEach' loop. Inside the loop, there is a 'Copy data' activity with the sub-option 'Copy Each Table'. The 'Sink' tab is selected in the dataset configuration pane on the right. The configuration includes:

- Name:** Parquet_bronze
- Description:** (empty)
- Connect via integration runtime:** AutoResolveIntegrationRuntime
- Authentication type:** Account key
- Account selection method:** From Azure subscription (selected)
- Azure subscription:** Azure for Students (13f6815e-9c47-445e-8f41-8bf80a581450)
- Storage account name:** adlspraveen02
- Test connection:** To linked service (selected)

The screenshot shows the Microsoft Azure Data Factory pipeline editor with the 'Set properties' dialog open for the sink dataset 'Parquet_Sink_bronze'. The dialog includes:

- Name:** Parquet_Sink_bronze
- Linked service:** Parquet_bronze
- File path:** bronze / Directory / File name
- Import schema:** From connection/store (selected)
- Advanced:** (button)

I want the folder structure inside bronze container as bronze/schemaname/tablename/table.parquet

Create parameters in the sink dataset so that they can be used for folder structure

The screenshot shows the Microsoft Azure Data Factory dataset editor for the sink dataset 'Parquet_Sink_bronze'. The 'Parameters' tab is selected, showing two parameters:

Name	Type	Default value
schemaname	String	Value
tablename	String	Value

The 'Properties' pane on the right shows:

- General:** Related (1)
- Name:** Parquet_Sink_bronze
- Description:** (empty)
- Annotations:** + New

Assign dynamic values as per the schema and table names that we get from 'LookUp' activity.

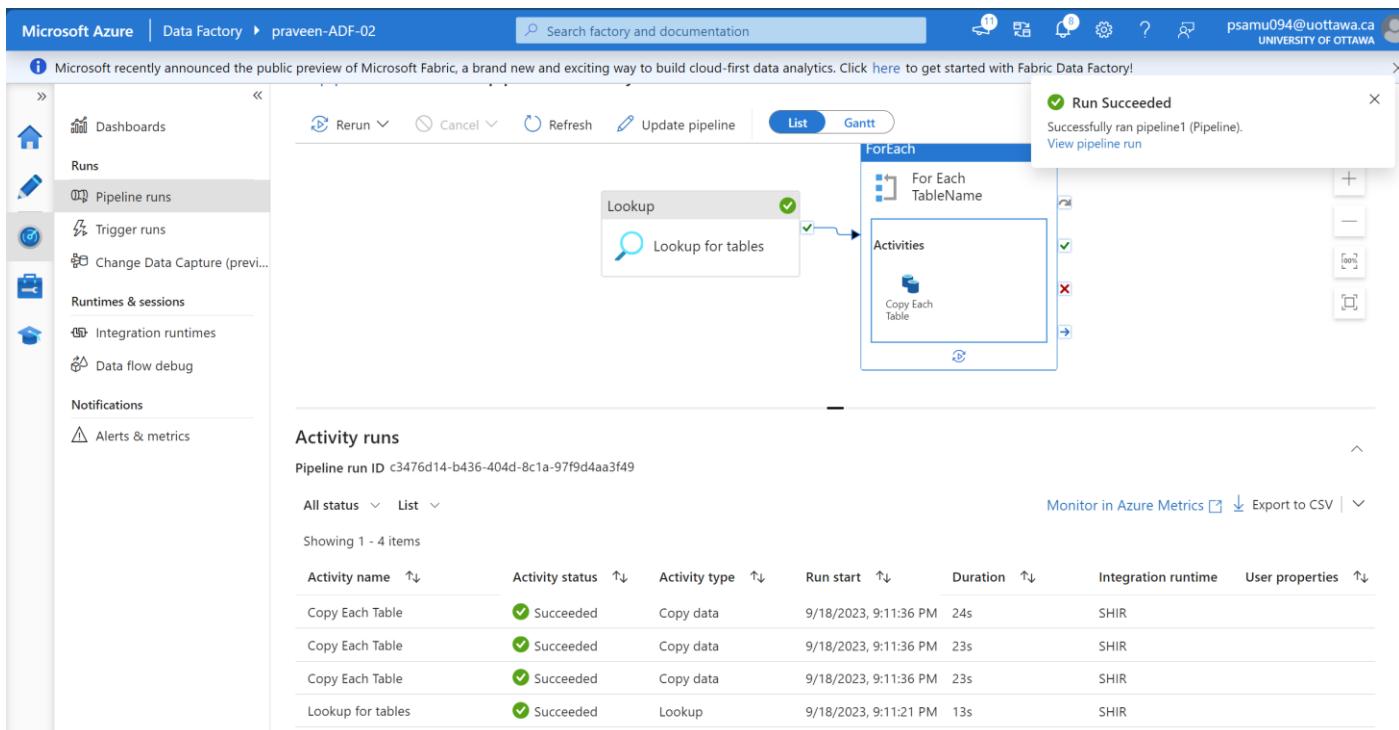
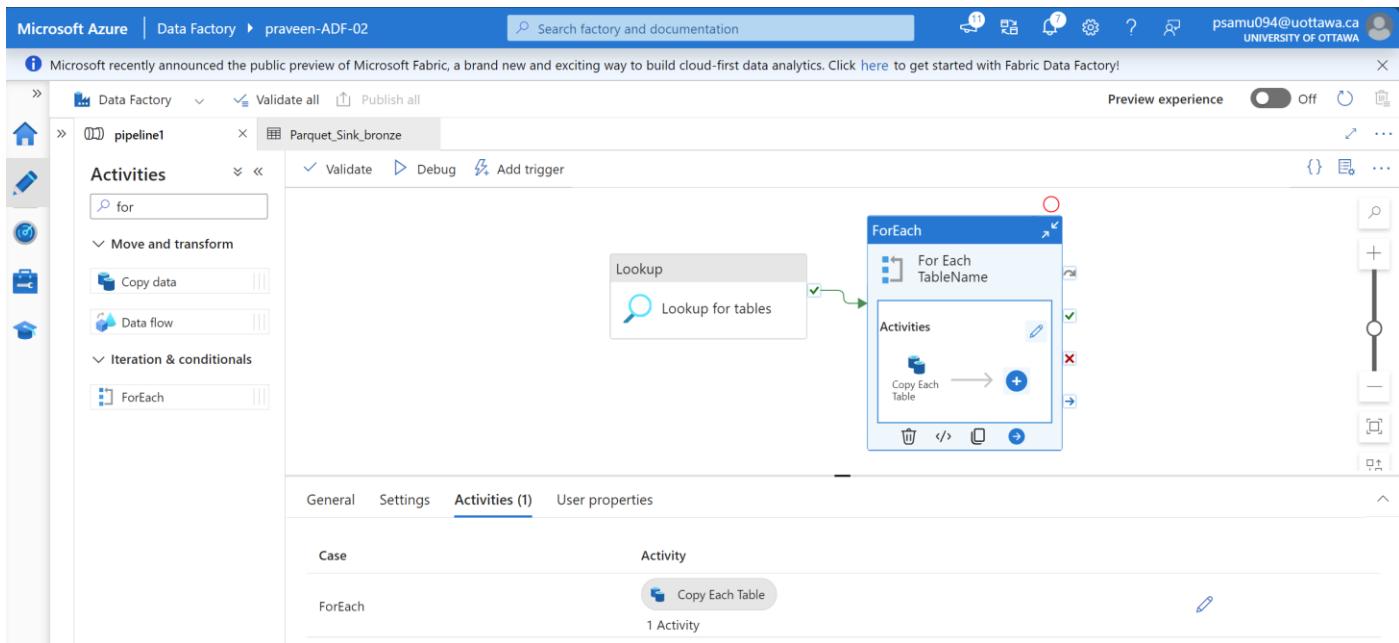
The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. A pipeline named 'pipeline1' is selected. Under the 'Activities' section, a 'For Each TableName' activity is chosen. Inside this activity, a 'Copy data' component is displayed. The 'Sink' tab is selected, showing a 'Parquet_Sink_bronze' dataset. The 'Dataset properties' table contains two entries: 'schemaname' with the value '@item().schemaname' and 'tablename' with the value '@item().tablename'. The 'General' tab shows the pipeline name and a preview of the sink dataset.

Create the folder structure

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. The 'Parquet_Sink_bronze' dataset is selected. The 'Connection' tab is active, showing a linked service 'Parquet_bronze' and a file path 'bronze'. The 'Schema' tab is also visible. To the right, a 'Pipeline expression builder' window is open, displaying the expression '@{concat(dataset().schemaname, '/', dataset().tablename)}' in the main area. Below it, parameters 'schemaname' and 'tablename' are listed. The 'Parameters' tab is selected in the builder.

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. The 'Parquet_Sink_bronze' dataset is selected. The 'Connection' tab is active, showing a linked service 'Parquet_bronze' and a file path 'bronze'. The 'Schema' tab is now active, showing the schema definition '@{concat(dataset().tablename, '.parquet')}'. The 'Parameters' tab is selected in the pipeline expression builder window to the right.

Publish all the files and Execute the Pipeline to check if tables are fetched and stored in ADLS in Parquet format in the desired folder structure.



Verifying files in ADLS

The screenshot shows the Microsoft Azure Storage account browser for the account 'adlspraveen02'. A blob container named 'bronze' is selected. Inside 'bronze', there is a single folder named 'public'. The table below lists the blobs in the 'public' folder.

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
public					-	

Screenshot 1: bronze Container (public)

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]					-	---
departments					-	---
employees					-	---
fruit_imports					-	---
regions					-	---

Screenshot 2: bronze Container (employees)

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]						---
employees.parquet	9/18/2023, 9:11:56 PM	Hot (Inferred)		Block blob	49.09 KiB	Available

Screenshot 3: bronze Container (departments)

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]						---
departments.parquet	9/18/2023, 9:11:55 PM	Hot (Inferred)		Block blob	824 B	Available

Screenshot 4: bronze Container (fruit_imports)

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]						---
fruit_imports.parquet	9/18/2023, 9:11:57 PM	Hot (Inferred)		Block blob	1.34 KiB	Available

Step 4 – Data Transformation in Azure Databricks

4.1 – Developing in Azure Databricks

Create Azure Databricks Compute Cluster and Notebook – Single Compute and Unrestricted.

Give access to Databricks cluster and notebook to access ADLS storage. As Premium Tier is disabled in my workspace, neither can I enable ‘Credential Passthrough’ nor I can create a secretScope to use Azure Key Vault to access the ADLS access key, hence giving the access key in the spark configuration manually.

If Premium Tier is enabled, either enable ‘Credential Passthrough’ in cluster settings, or Create a secret in Azure Key Vault with secret as ADLS Access Key

Microsoft Azure Search resources, services, and docs (G+/-)

Home > Key vaults > keyvault-praveen-02 | Secrets >

Create a secret

Upload options	Manual
Name *	key-ADLS
Secret value *	*****
Content type (optional)	
Set activation date	<input type="checkbox"/>
Set expiration date	<input type="checkbox"/>
Enabled	<input checked="" type="radio"/> Yes <input type="radio"/> No
Tags	0 tags

In Databricks workspace, add '#secrets/createScope' to the URL to create a Secret Scope

Give a Scope name, for DNS go to Azure Key Vault -> Choose resource -> Properties -> Vault URI, resource ID -> resource ID.

Microsoft Azure | databricks Search data, notebooks, recents, and more... CTRL + P ADB-praveen-02 psamu094@uottawa.ca

+ New

- Workspace
- Recents
- Catalog
- Workflows
- Compute
- SQL
- Data Engineering
- Job Runs
- Data Ingestion
- Delta Live Tables
- Machine Learning
- Experiments
- Features
- Models

HomePage / Create Secret Scope

Create Secret Scope

A store for secrets that is identified by a name and backed by a specific store type. [Learn more](#)

Scope Name

Manage Principal

Azure Key Vault

DNS Name

Resource ID

Getting error as I don't have Premium Tier.

After manually setting ADLS key in Spark Config. Develop the code for transforming data field in tables and save to Silver container. Usually during transformation to Gold stage, tables are brought into fact and dimension tables format. As in our case the tables are already in this format, not performing any transformation from Silver to Gold stage. Reading tables from Silver container and saving to gold container.

4.2 – Making Azure Databricks notebook part of Data Factory Pipeline.

In Azure Databricks workspace -> Name -> User Settings -> Developer -> Access Tokens

Generate a New Token and save it in Azure Key Vault, it will be used in Data Factory Linked Service to Databricks resource.

User settings > Developer > Access tokens

Comment	Creation	Expiration
ADF	2023-09-19 11:52:23 IST	2023-12-18 11:52:23 IST

Create a secret ...

Upload options: Manual

Name * ⓘ: ADB-resource

Secret value * ⓘ:

Content type (optional):

Set activation date ⓘ:

Set expiration date ⓘ:

Enabled: Yes No

Tags: 0 tags

In the Data Factory Pipeline, create a Linked Service to Azure Databricks resource.

New linked service

Azure Databricks Learn more ⓘ

AutoResolveIntegrationRuntime

Account selection method *

From Azure subscription Enter manually

Azure subscription * ⓘ

Azure for Students (13f6815e-9c47-445e-8f41-8bf80a581450)

Databricks workspace * ⓘ

ADB-praveen-02

Select cluster

New job cluster Existing interactive cluster Existing instance pool

Databrick Workspace URL * ⓘ

https://adb-1762870589663047.7.azuredatabricks.net

Authentication type *

Access Token

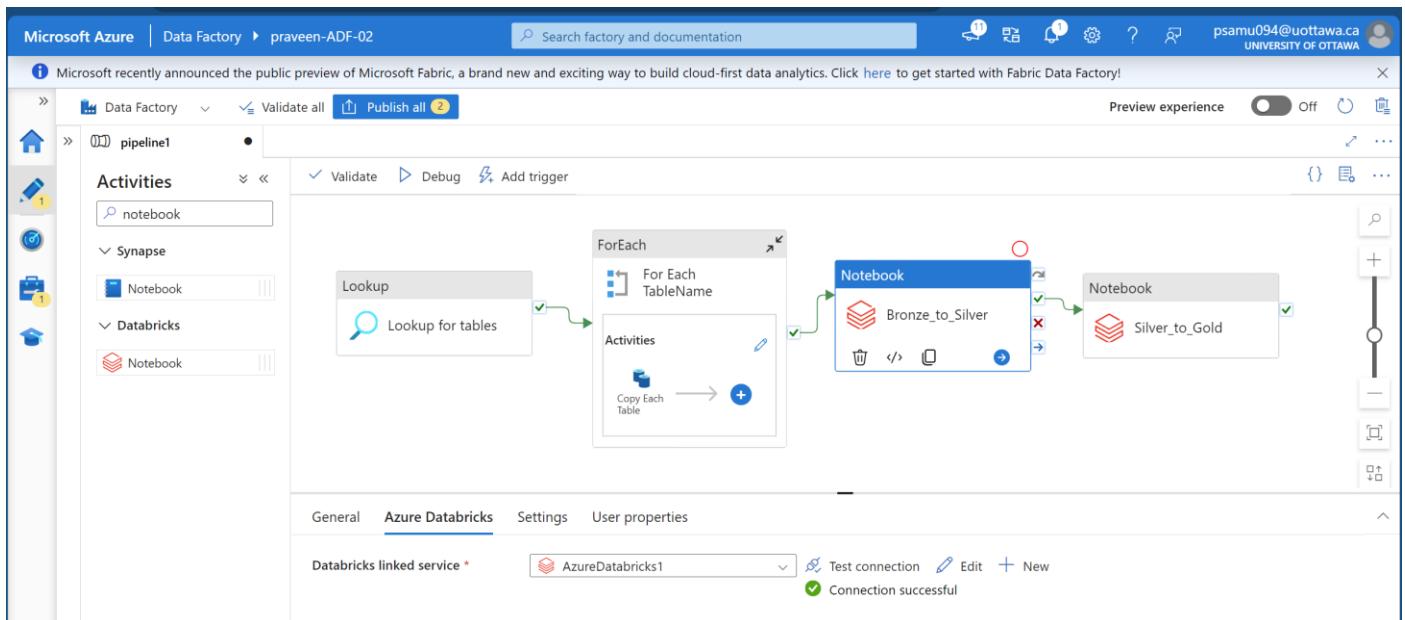
AKV linked service * ⓘ

AzureKeyVault1

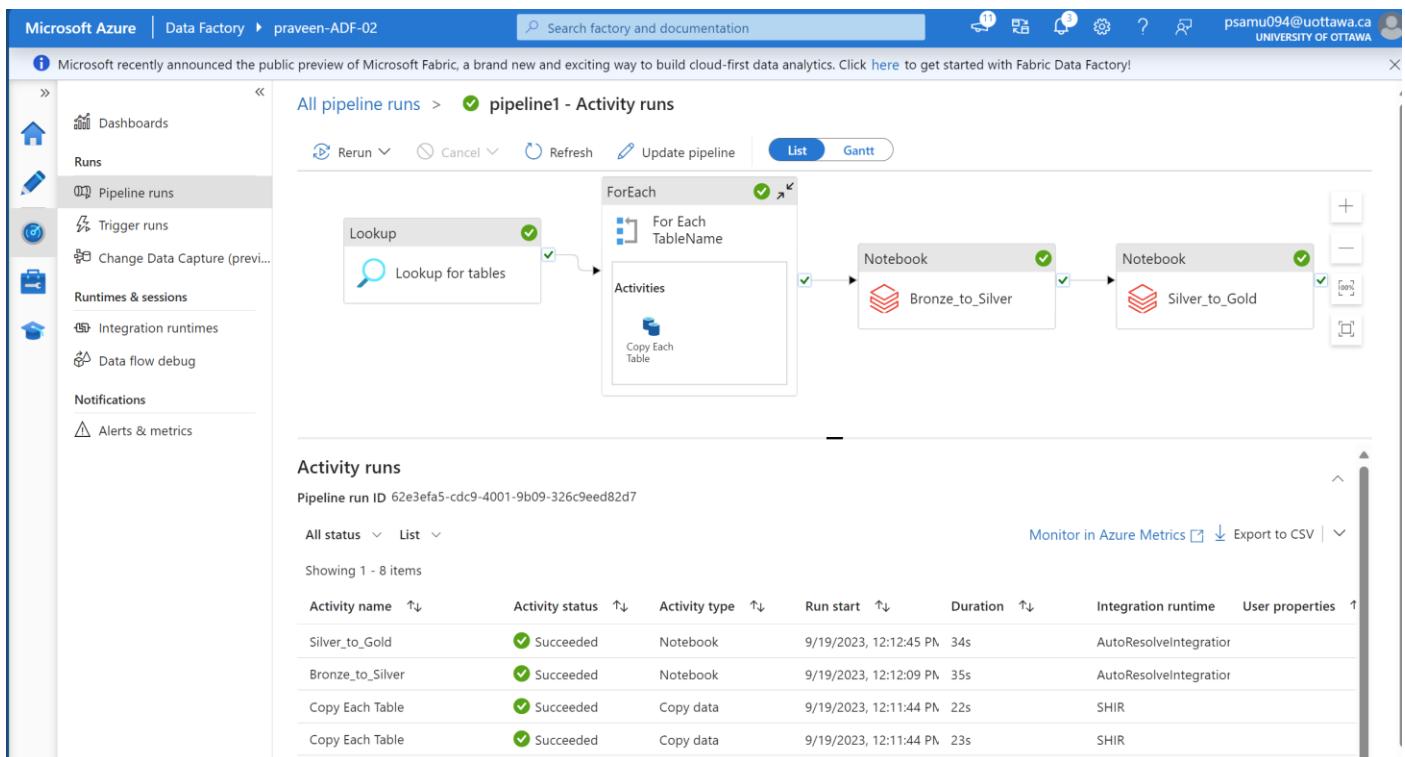
Secret name * ⓘ

ADB-resource

Add Azure Databricks Notebooks in the Pipeline each triggering Transformation to Silver stage and Gold stage.



Trigger to execute the pipeline.



Step 5 – Loading Delta Tables from Databricks to Azure Synapse as Data Warehouse

5.1 Create a Serverless SQL DB in Synapse

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. On the left, there's a navigation sidebar with icons for Home, Data, Workspace, and Linked. The main area shows a 'Data' section with tabs for 'Workspace' (selected) and 'Linked'. A search bar at the top right has the email 'psamu094@uottawa.ca UNIVERSITY OF OTTAWA'. On the right, a 'Create SQL database' panel is open, titled 'Create SQL database'. It contains instructions to 'Create database to organize your workload into databases and database objects.' Under 'Select SQL pool type *', 'Serverless' is selected. In the 'Database *' field, 'gold_db' is typed. Below the database name, there are two radio buttons: 'Dedicated' and another one which is not clearly visible.

Learn – Views can be created in the SQL DB using SQL in Synapse like below

This screenshot shows the Microsoft Azure Synapse Analytics workspace. The left sidebar shows 'Data' with 'Workspace' selected. In the center, under 'gold_db (SQL)', there's a 'Views' folder containing a single item: 'dbo.employees'. To the right, a 'SQL script 1' tab is active, displaying the following SQL code:

```
1 create view employees as
2 SELECT *
3 FROM
4 OPENROWSET(
5     BULK 'https://adlspraveen02.dfs.core.windows.net/gold/employees/',
6     FORMAT = 'DELTA'
7 ) AS [result]
```

The 'Properties' pane on the right shows the 'Name' is set to 'SQL script 1'. The 'General' tab is selected.

5.2 Now I'll create a Pipeline that can be invoked to create views in SQL DB automatically.

Develop a SQL script such that a View for each table is created dynamically as the tables are read.

This screenshot shows the Microsoft Azure Synapse Analytics workspace. The left sidebar shows 'Develop' with 'SQL scripts' selected. There are two items: 'SQL script 1' and 'SQL script 2'. The 'SQL script 2' tab is active, displaying the following dynamic SQL script:

```
1 USE gold_db
2 GO
3
4 CREATE OR ALTER PROC CreateSQLView_gold @ViewName NVARCHAR(100) AS
5 BEGIN
6
7     DECLARE @statement varchar(MAX)
8
9     SET @statement = N'create or alter view ' + @ViewName + ' AS
10        SELECT * FROM
11            OPENROWSET(
12                BULK 'https://adlspraveen02.dfs.core.windows.net/gold/' + @ViewName + '/',
13                FORMAT = ''DELTA''
14            ) AS [result]'
15
16     EXEC (@statement)
17
18 END
19 GO
```

As we are creating the views in SQL DB, Synapse pipeline needs Linked Service to Azure SQL DB to connect and create views.

Domain Name = Synapse Workspace -> Properties -> Serverless SQL Endpoint

Create a New Pipeline – Idea is to create the tables first (using Metadata activity), then for each (ForEach activity) table load it with the dynamic query.

Create a dataset for the activity -> ADLS -> Binary format, choose the default Synapse connection to ADLS.

The result from Metadata will be as follows returning JSON data of items present in the location

```
{
  "childItems": [
    {
      "name": "departments",
      "type": "Folder"
    },
    {
      "name": "employees",
      "type": "Folder"
    }
  ]
}
```

Insert ForEach activity as next step in pipeline and write dynamic SQL to fetch childItems from metadata result

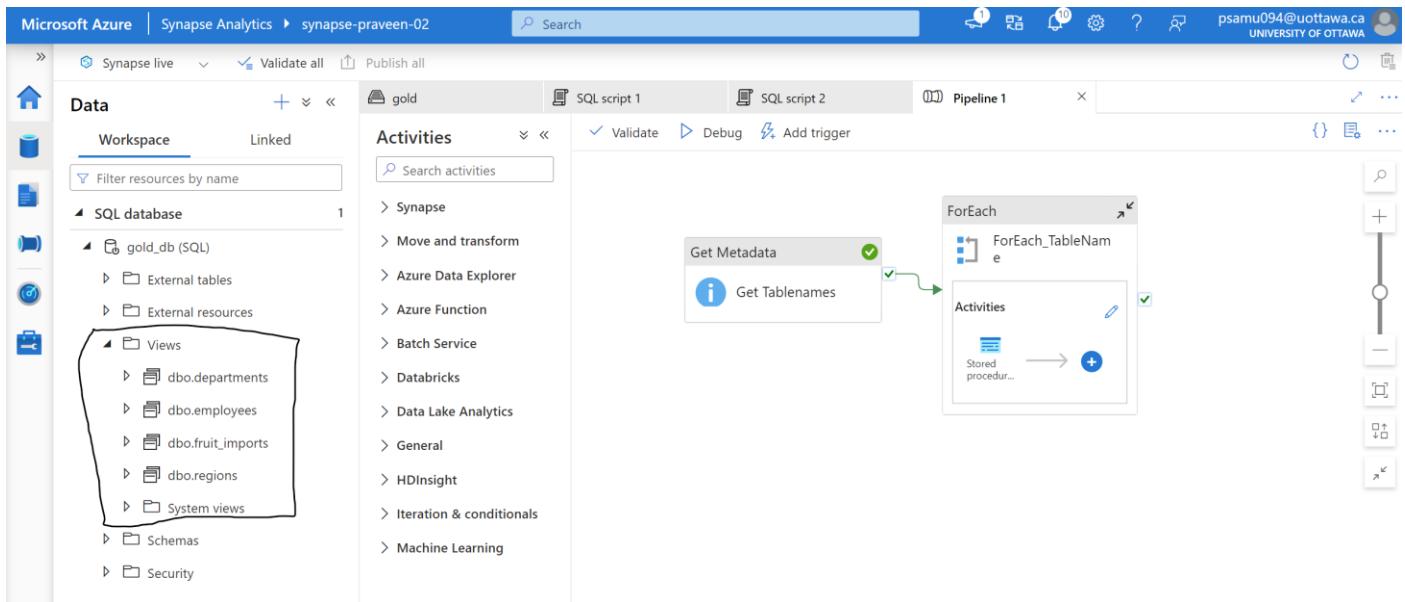
In the ForEach activity, place a Stored Procedure activity and configure the parameters

The screenshot shows the Microsoft Azure Synapse Analytics Pipeline Editor. A pipeline named 'Pipeline 1' is open, containing an activity named 'ForEach_TableName'. Inside this activity, there is a 'Stored procedure' activity named 'Stored procedure1'. This stored procedure is linked to an 'AzureSqlDatabase1' linked service and an 'AutoResolveIntegrationRuntime'. The stored procedure name is set to '[dbo].[CreateSQLView_gold]'. A parameter 'ViewName' is defined with a value of '@item().name'. The pipeline editor interface includes tabs for General, Settings, and User properties, along with various configuration options and connection status indicators.

Run the pipeline and check if everything works as intended

Activity name	Activity status	Activity type	Run start	Duration	Log	Integration runtime
Stored procedure1	Succeeded	Stored procedure	9/19/2023, 3:43:39 PM	11s		AutoResolveIntegration
Stored procedure1	Succeeded	Stored procedure	9/19/2023, 3:43:39 PM	10s		AutoResolveIntegration
Stored procedure1	Succeeded	Stored procedure	9/19/2023, 3:43:39 PM	10s		AutoResolveIntegration
Stored procedure1	Succeeded	Stored procedure	9/19/2023, 3:43:39 PM	10s		AutoResolveIntegration
Get Tablenames	Succeeded	Get Metadata	9/19/2023, 3:43:35 PM	3s		AutoResolveIntegration

Views are created



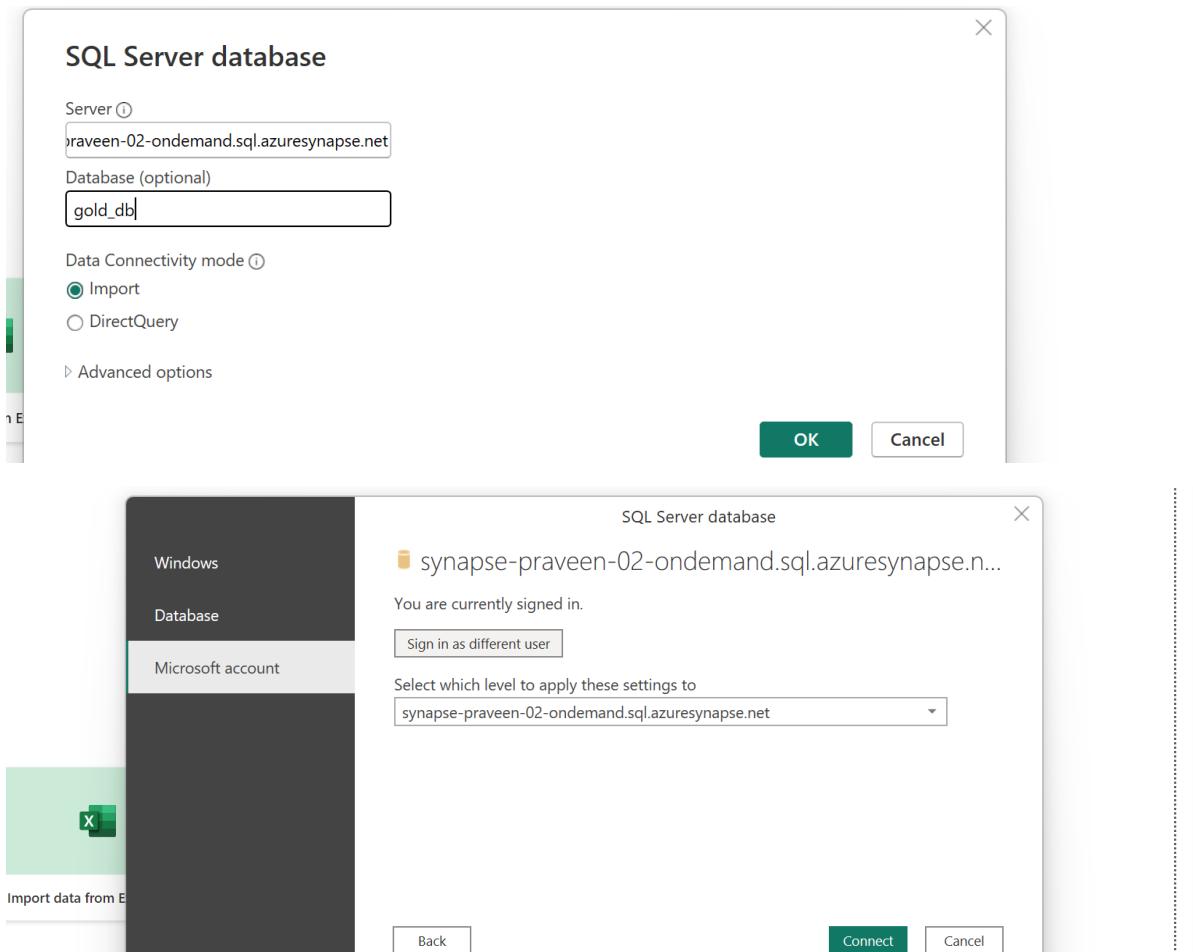
This pipeline is needed to run only when there is change in table schema at source table. Any newly added data will sync automatically.

Step 6 – Provide Business Analytics using PowerBI with data in Synapse DB

Launch PowerBI desktop application

Get Data -> More -> Azure -> Synapse Analytics SQL -> Connect

Get the Serverless SQL endpoint connection string from Synapse Properties and enter the details in PowerBI to connect to Synapse



Choose the tables to be imported and choose ‘Load’.

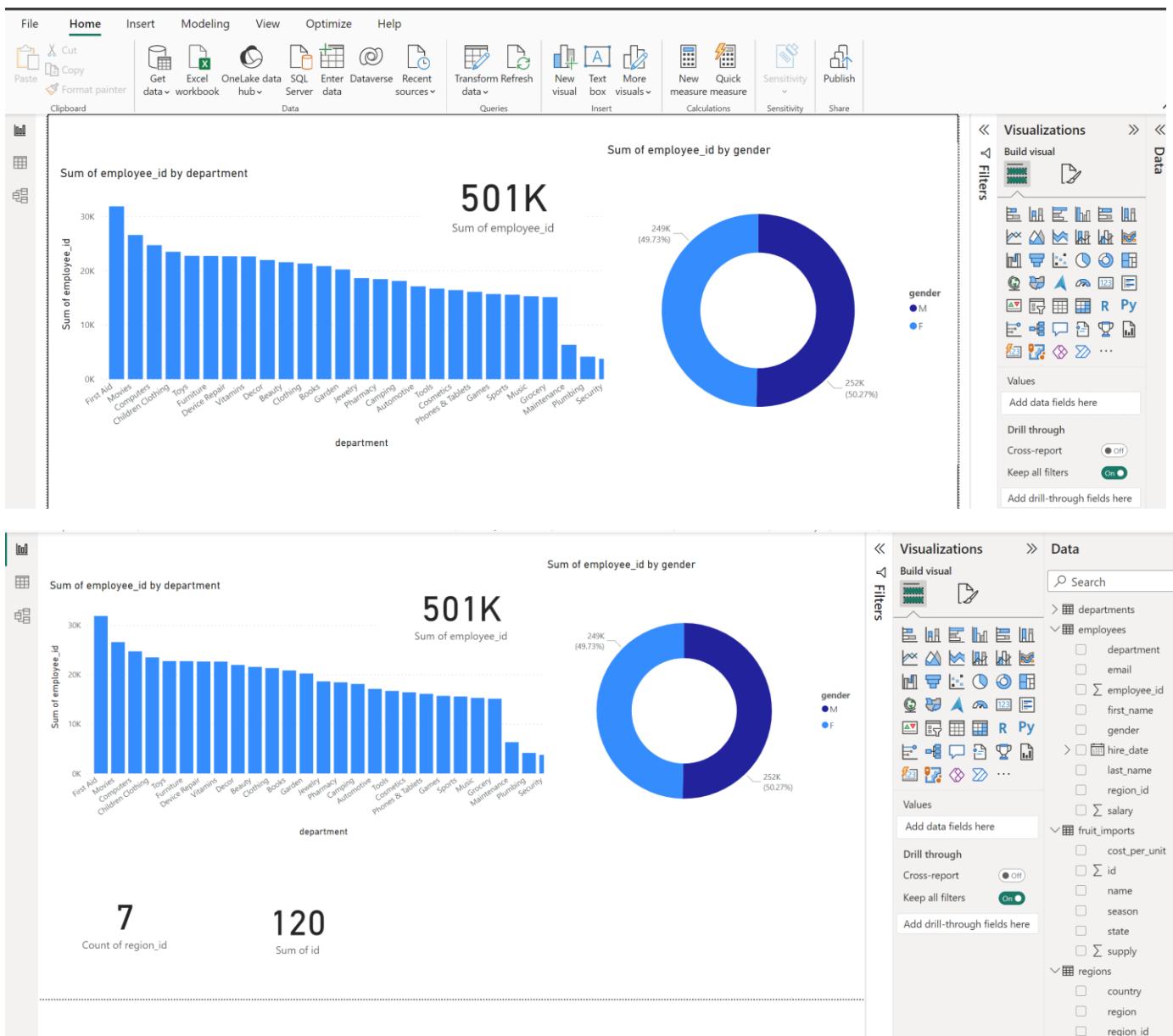
The screenshot shows the Power BI Desktop interface. The top navigation bar includes File, Home, Help, and Table tools. The Table tools ribbon is selected, showing options like 'Name' (set to 'employees'), 'Mark as date table', 'Manage relationships', 'Calendars', 'Relationships', 'New measure', 'Quick measure', 'New measure column', 'New table', and 'Calculations'. Below the ribbon is a data grid displaying employee data from the 'employees' table. The data includes columns: employee_id, first_name, last_name, email, hire_date, department, gender, salary, and region_id. The 'Data' pane on the right shows a search bar with results for 'departments', 'employees', 'fruit_imports', and 'regions'.

This screenshot shows the Power BI Desktop interface with the Data ribbon selected. The ribbon includes Paste, Get data, Excel workbook, OneLake data hub, SQL Server, Enter data, Recent sources, Transform Refresh data, Relationships, Calculations, Security, Q&A setup, Language schema, Sensitivity, and Publish. Below the ribbon is a data model diagram showing relationships between three tables: 'employees', 'departments', and 'fruit_imports'. The 'Properties' pane on the right contains settings for cards, related fields, and pinned fields.

The values of email in employees table are available in Synapse DB views but not populating in PowerBI

This screenshot shows the Microsoft Azure Synapse Analytics workspace. The left sidebar shows the workspace structure with a 'Data' section containing 'Workspace' and 'Linked' resources. The main area is a query editor titled 'gold' under 'SQL database'. It displays an external table definition for 'employees' using OPENROWSET. The results grid shows several rows of employee data, including columns: first_name, last_name, email, hire_date, department, and gender. The 'Properties' pane on the right shows details for 'SQL script 1', including type (.sql script), size (192 bytes), and results settings per query (First 5000 rows (default)).

Visualizations in PowerBI



Step 7 – Scheduling the Data Pipeline to run once a day

Schedule trigger for Azure Data Factory Pipeline

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, there's a sidebar with various activities like Move and transform, Synapse, Azure Data Explorer, etc. The main workspace shows a 'ForEach' loop activity. Inside the loop, there's a 'Copy Each Table' activity. A 'Lookup' activity is connected to the 'ForEach' loop. On the right, a 'New trigger' dialog box is open, showing the configuration for a 'trigger-daily' trigger. The trigger is set to 'Schedule' type, starting at '9/19/2023, 5:00:53 PM' in the 'Chennai, Kolkata, Mumbai, New Delhi (UTC+5:30)' time zone. The 'Recurrence' is set to 'Every 1 Day(s)'. Below that, there are options for 'Execute at these times' (Hours: 17, Minutes: 10) and 'Schedule execution times' (17:10). A checkbox for 'Specify an end date' is checked.

Added new values into tables

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows 'PostgreSQL 15' and 'course_data' database selected. The main window displays a SQL query editor with the following code:

```

1 insert into departments values ('Test_dep1', 'Test_division1');
2 insert into departments values ('Test_dep2', 'Test_division2');
3
4
5 insert into regions values (8, 'Ontario', 'Canada');
6 insert into regions values (9, 'South Asia', 'India');
7 insert into regions values (10, 'Southeast', 'United States');
8
9

```

The results pane shows 'INSERT 0 1' and 'Query returned successfully in 77 msec.'

Pipeline execution at scheduled time

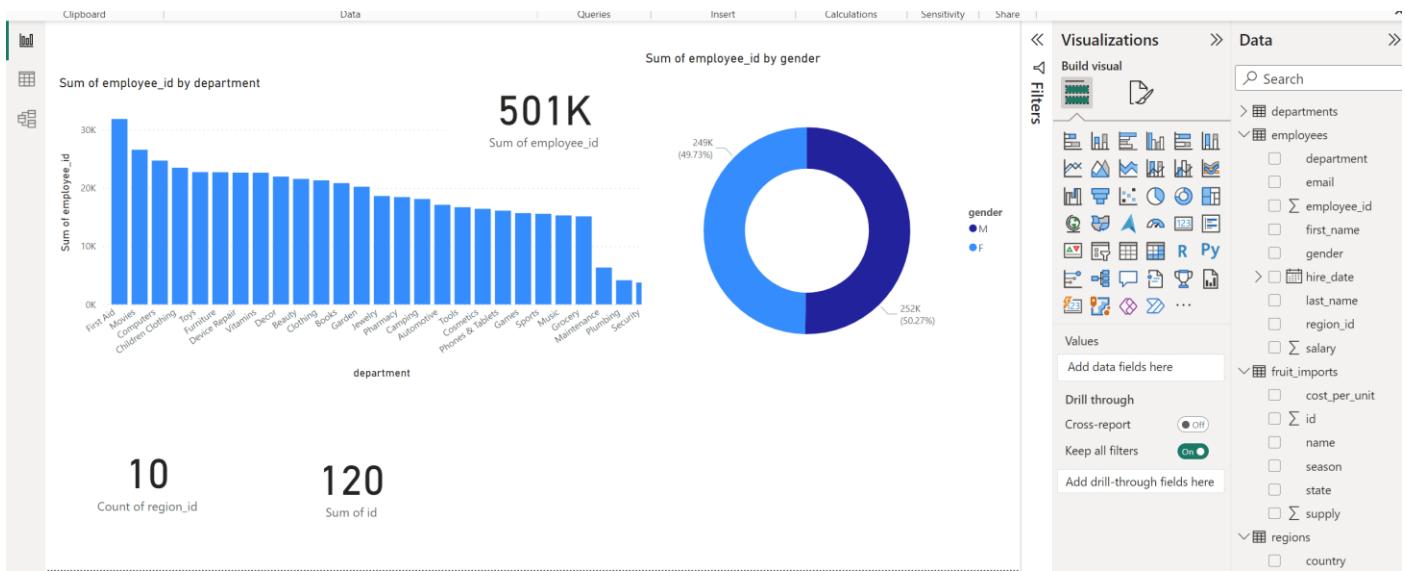
The screenshot shows the Microsoft Azure Data Factory 'Pipeline runs' page. The left sidebar has 'Pipeline runs' selected. The main area shows a table of pipeline runs:

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
pipeline1	9/19/2023, 5:10:00 PM	--	1m 12s	trigger-daily	In progress	Original
pipeline1	9/19/2023, 12:11:27 PM	9/19/2023, 12:13:20 PM	1m 53s	Manual trigger	Succeeded	Original
pipeline1	9/18/2023, 9:11:19 PM	9/18/2023, 9:12:03 PM	44s	Manual trigger	Succeeded	Original

Pipeline completed successfully

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar has a 'Runs' section with 'Pipeline runs' selected. The main area is titled 'Pipeline runs' and shows a single run for 'pipeline1'. The run details are: Triggered by: trigger-daily, Status: Succeeded, Run: Original. The run started at 9/19/2023, 5:10:00 PM and ended at 9/19/2023, 5:13:27 PM, with a duration of 3m 27s. The status bar indicates the page was last refreshed 0 minutes ago.

Reflection of new data in PowerBI



Regions count updated from 7 to 10 after adding 3 new regions in on-premise PostgreSQL server.

References:

[An End to End Azure Data Engineering Real Time Project Demo | Get Hired as an Azure Data Engineer - YouTube](#)