Azure Data Engineering Project – Adventure Works Sample Data

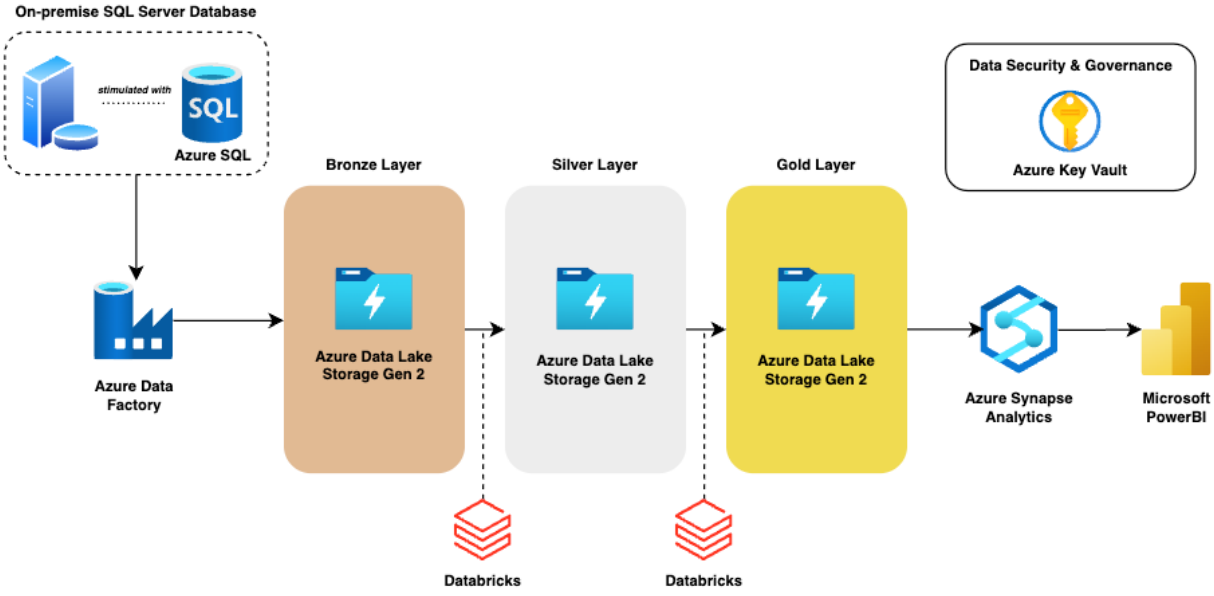
Overview:

This project aims to create an end-to-end data pipeline from ingestion data to visualization by leveraging various Azure services with the Adventure Works sample data. The pipeline will be triggered in a daily basis, and the latest updated data will be reflected in the dashboard automatically.

Data Source:

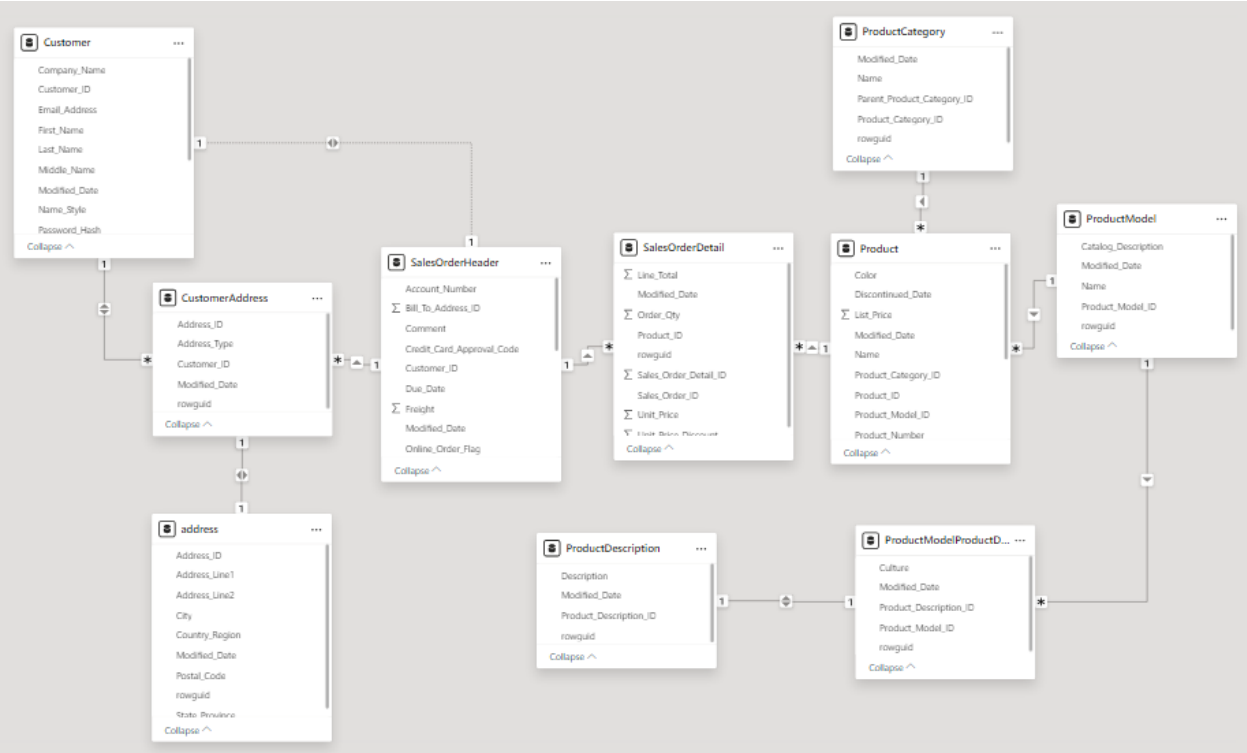
Adventure Works Sample Data

Data Pipeline Architecture:



- On-premises Database Simulation: Azure SQL, Microsft SQL Server, Docker, Azure Data Studio
- Data Ingestion: Azure Data Factory
- Data Transformation: Azure Databricks
- Data Loading: Azure Synapse Analytics
- Data Visualization: Microsoft PowerBI
- Data Storage: Azure SQL, Azure Data Lake Storage Gen 2
- Data Security & Governance: Azure Key Vault

Data Model / Schema:



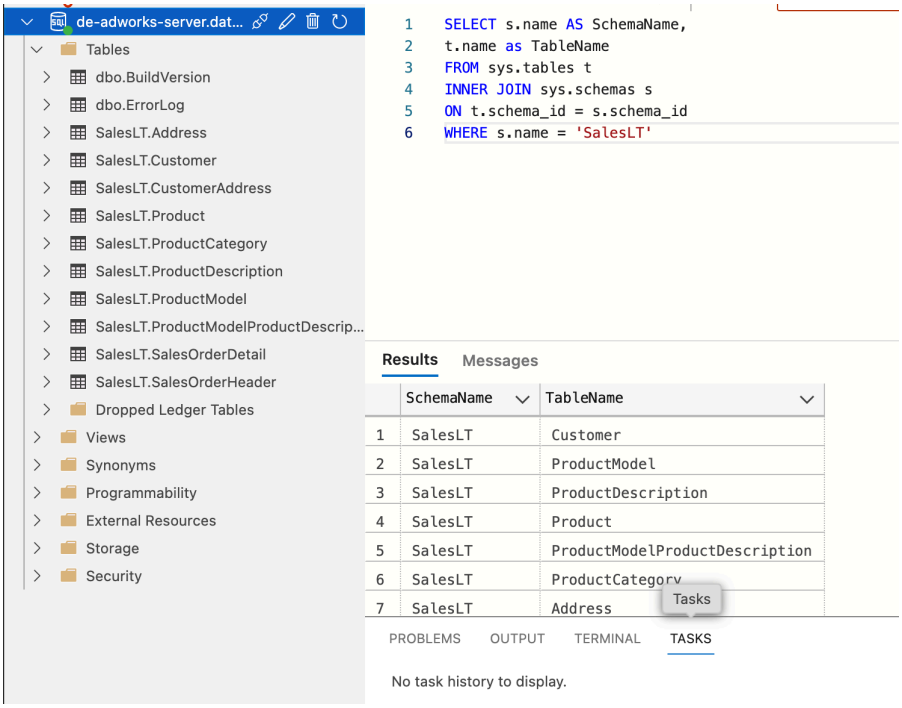
Setup: Restoring the database in MS SQL

*It would not work for Mac user as it cannot be connected to azure data factory due to the limitation of azure data studio (self-hosted integration runtime N/A)

1. Use docker to run MS SQL and Azure Data Studio as the replacement of SSMS due to the nature of Mac environment.
2. Run the below command to stimulate the on-premises database environment.
   - docker run -e "ACCEPT_EULA=1" -e "MSSQL_SA_PASSWORD=*{YOURPASSWORD}*" -e "MSSQL_USER=SA" -p 1433:1433 -d --name=mssql mcr.microsoft.com/azure-sql-edge
3. Restore database by using the bak file provided in the data source website.
4. Create a login user and password for azure access to the database by using the command below.
   - CREATE USER <Azure_AD_principal_name> FROM EXTERNAL PROVIDER;
   - ALTER ROLE db_owner ADD MEMBER <ADF_service_name>
5. Store the credentials in Azure Key Vault for future usage.

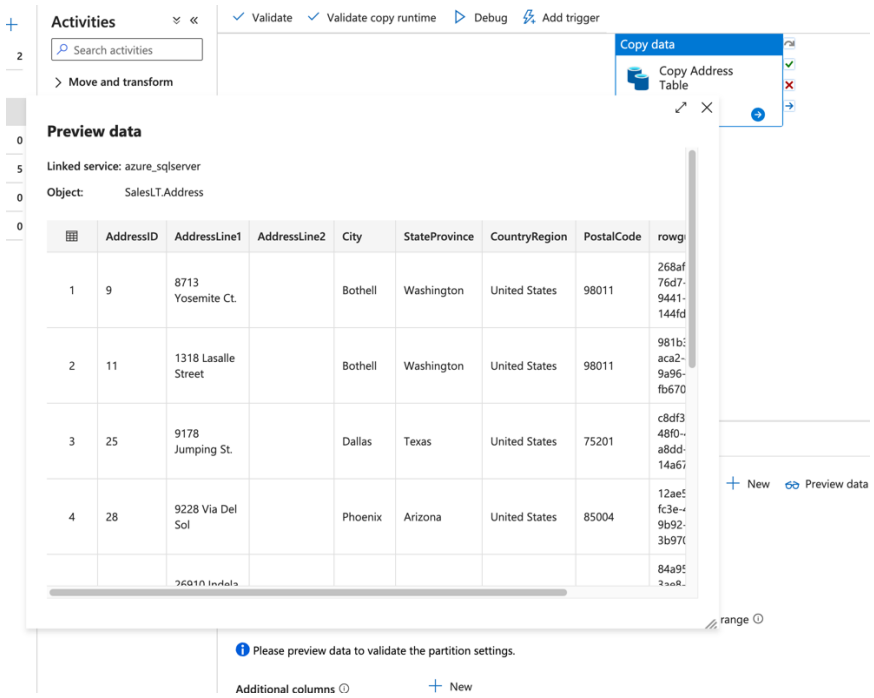
Setup: Restore the database in Azure SQL database

1. Azure SQL database is used to stimulate the on-premises database and connect to azure data studio due to the limitation of Mac environment.
2. Select Azure SQL as the service and include the sample data in the configuration settings.
3. Write a T-SQL query to grant access to Azure Data Factory with the command below.
   - CREATE USER [de-adworks-adf] FROM EXTERNAL PROVIDER;
   - ALTER ROLE db_owner ADD MEMBER [de-adworks-adf];
4. Query the data for testing

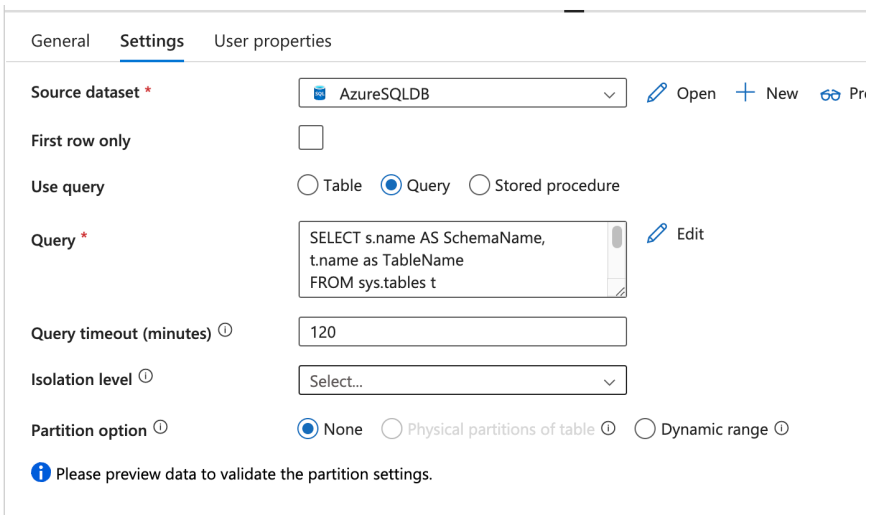


Data Ingestion: Ingest Data from Azure SQL (act as on-prem database) to Bronze Layer Azure Data Lake Storage Gen 2

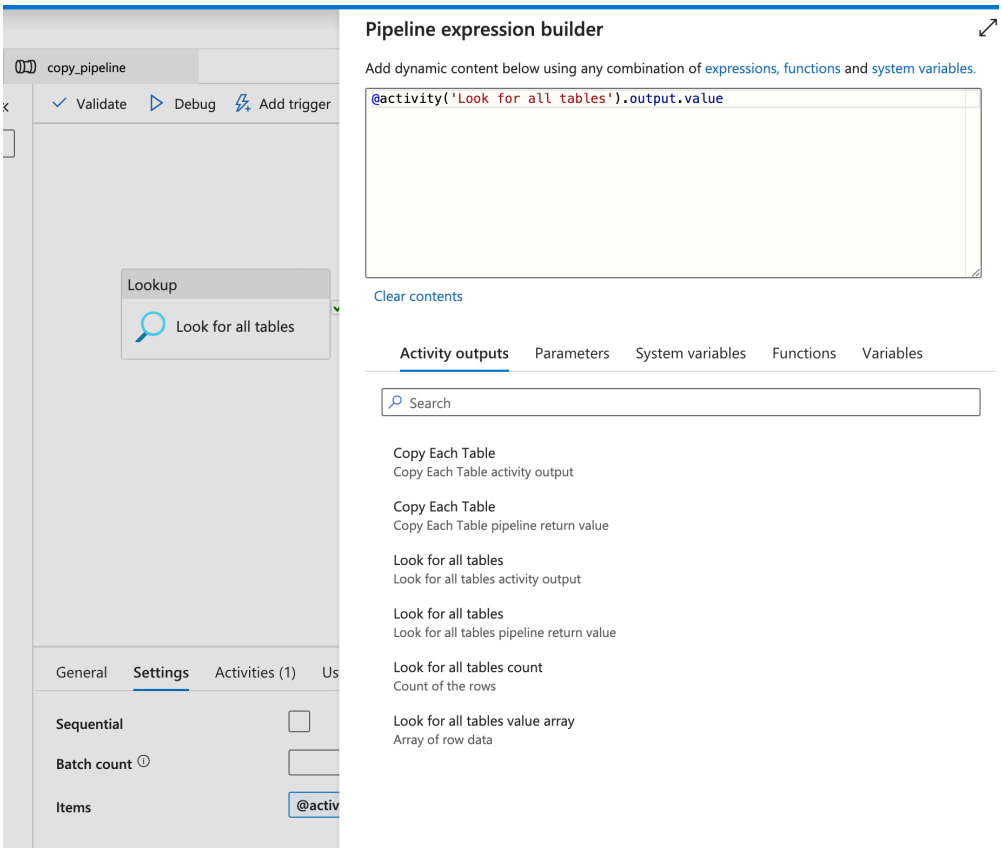
1. Add Azure Data Factory into the resource group
2. Connect Azure SQL database with self-hosted integration runtime in ADF
3. Creating a copy activity for testing purpose with only Address table.
4. Create a linked service to Azure SQL database with the proper authentication
5. Preview Address Data in Copy activity

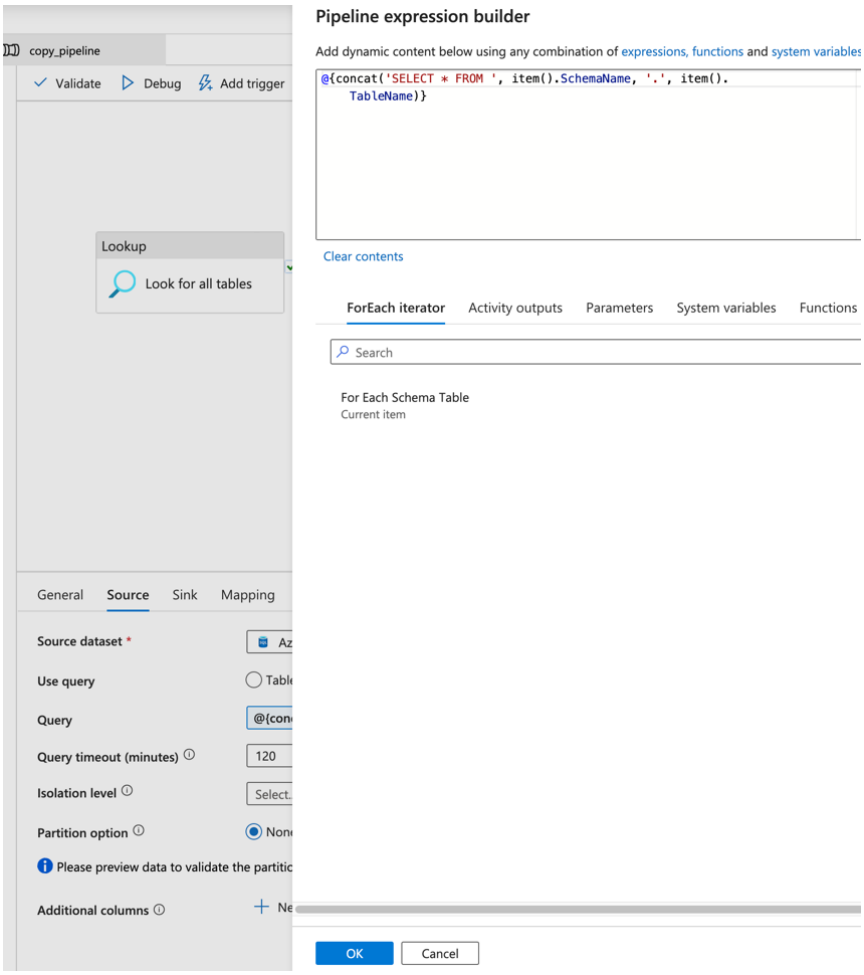


6. Configure sink to ADLS by creating a storage account for ADLS and create a bronze container to store the raw data file
7. Test the copy activity to check if address table is successfully ingested into bronze table in ADLS.
8. Create a Copy All Tables Pipeline that ingest all tables in Azure SQL database into the bronze layer.
9. Add Lookup activity in pipeline to query the available tables in the database by using SQL query
   - SELECT s.name AS SchemaName,
     t.name as TableName
     FROM sys.tables t
     INNER JOIN sys.schemas s
     ON t.schema_id = s.schema_id
     WHERE s.name = 'SalesLT'
   - The output of the lookup activity is used as a parameter to perform the copy of all tables

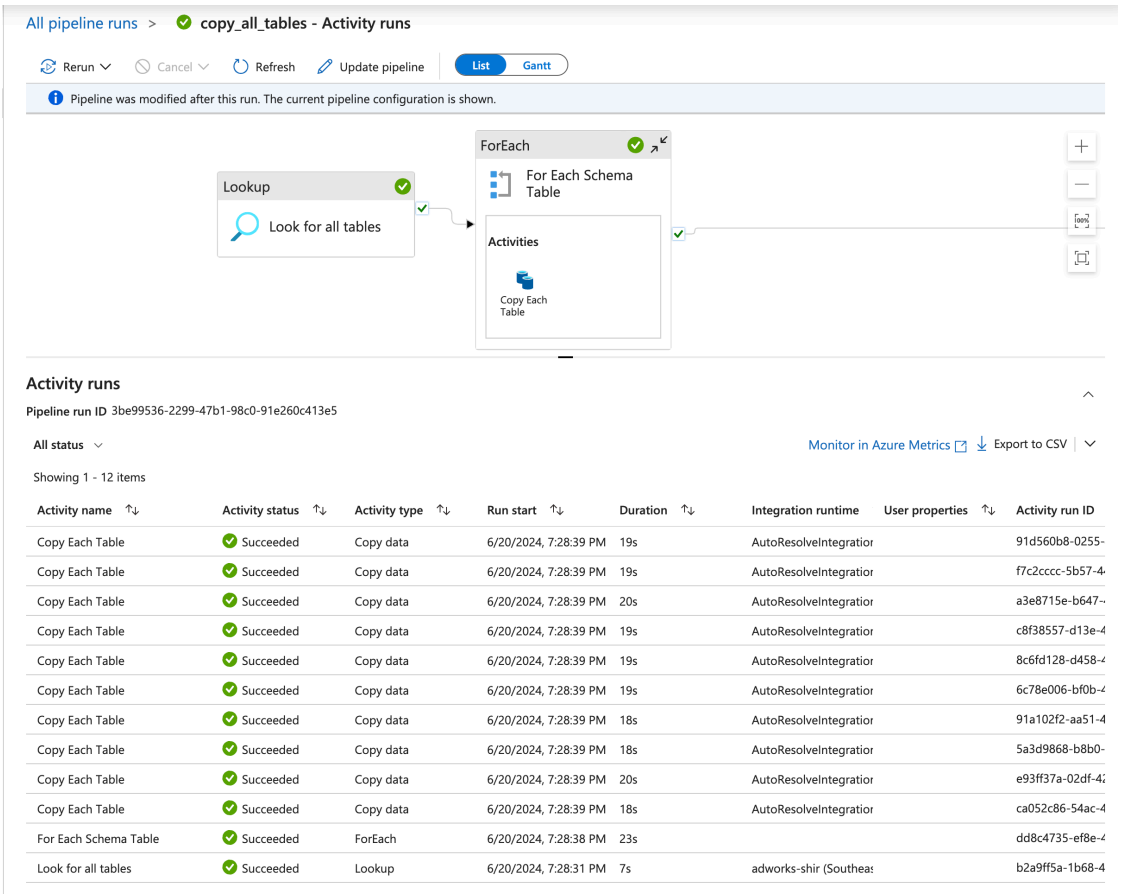


10. Add for each activity with by wrapping a copy data activity inside





11. Configure folder structure for bronze storage as the sink
    - bronze/schemaName/tableName/table.parquet
12. Test the pipeline with trigger run



Data Transformation:

1. Add Azure Databricks into the resource group
2. Mount ADLS to databricks to get access to the data
   - not necessary to mount the data since the azure credential passthrough is enabled in the cluster, but it will ease the coding part as different containers have different mount points.
3. Create a notebook to mount the data

2 days ago (11s) 4

```
configs = {
  "fs.azure.account.auth.type": "CustomAccessToken",
  "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

# Optionally, you can add <directory-name> to the source URI of your mount point.
dbutils.fs.mount(
  source = "abfss://silver@adworksadls.dfs.core.windows.net/",
  mount_point = "/mnt/silver",
  extra_configs = configs)
```

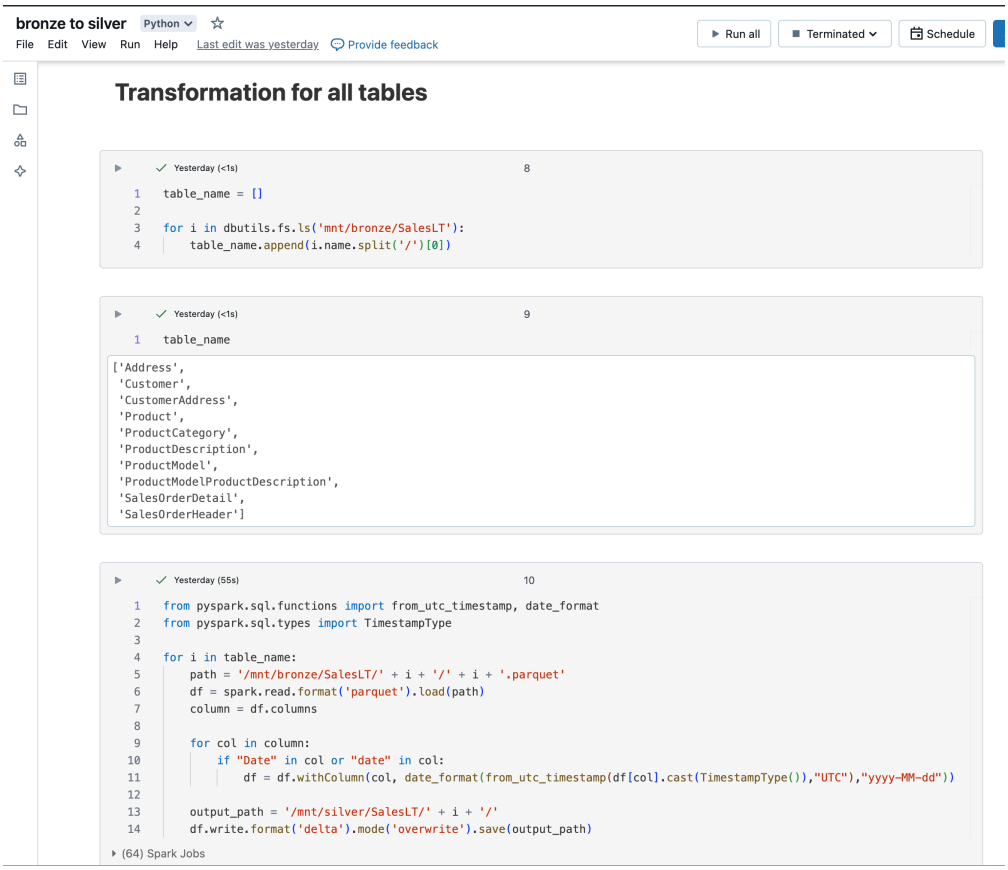
```
True
```

2 days ago (11s) 5 Python

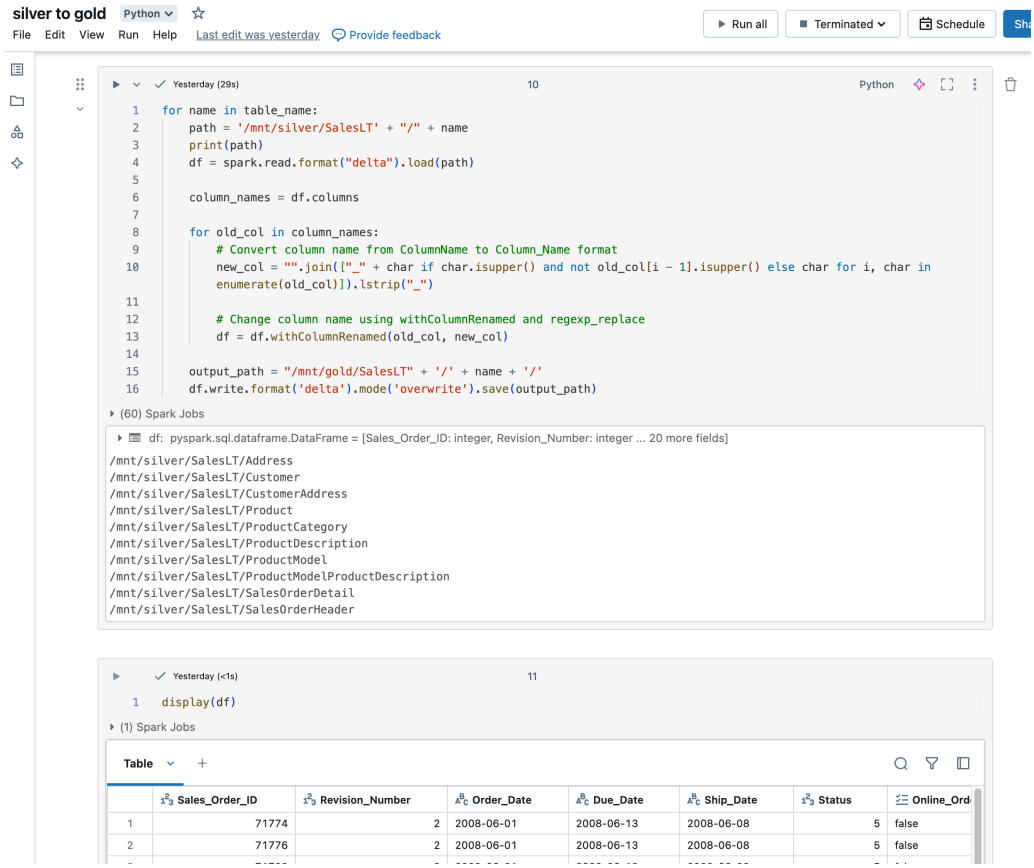
```
configs = {
  "fs.azure.account.auth.type": "CustomAccessToken",
  "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

# Optionally, you can add <directory-name> to the source URI of your mount point.
dbutils.fs.mount(
  source = "abfss://gold@adworksadls.dfs.core.windows.net/",
  mount_point = "/mnt/gold",
  extra_configs = configs)
```

```
True
```

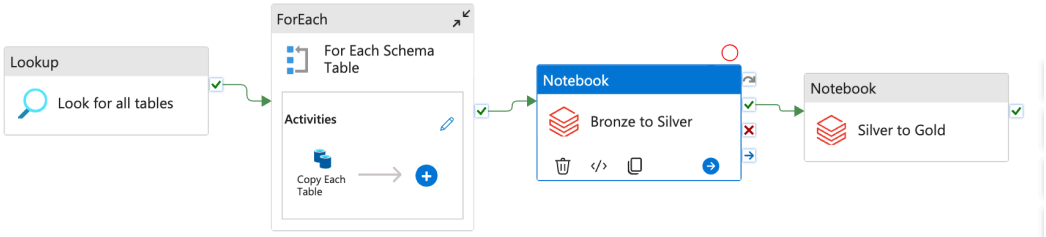
4. [Bronze → Silver] Transform date column into date format as the date columns in every table is in datetime format in default
5. Run a loop to transform all the files in bronze layer and save it in silver layer in delta format

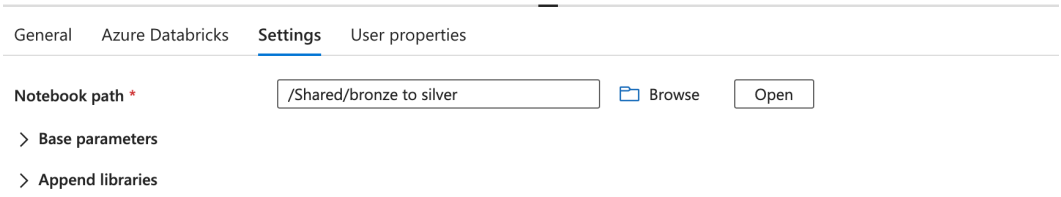


6. [Silver → Gold] Convert every column naming format from ColumnName to Column_Name for better naming conventions

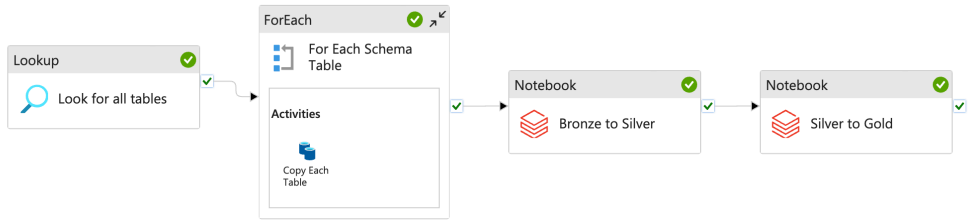


7. Create linked service between Databricks and ADF.
8. Add Notebook activity in ADF with the databricks notebook that we have created earlier for Bronze to Silver and Silver to Gold activity.





9. Test pipeline and verify if the files are successfully transformed and loaded into ADLS.



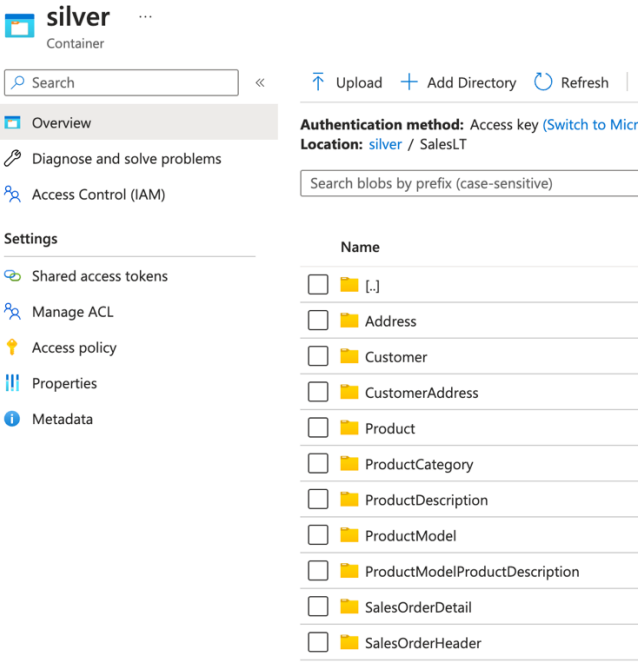
Activity runs

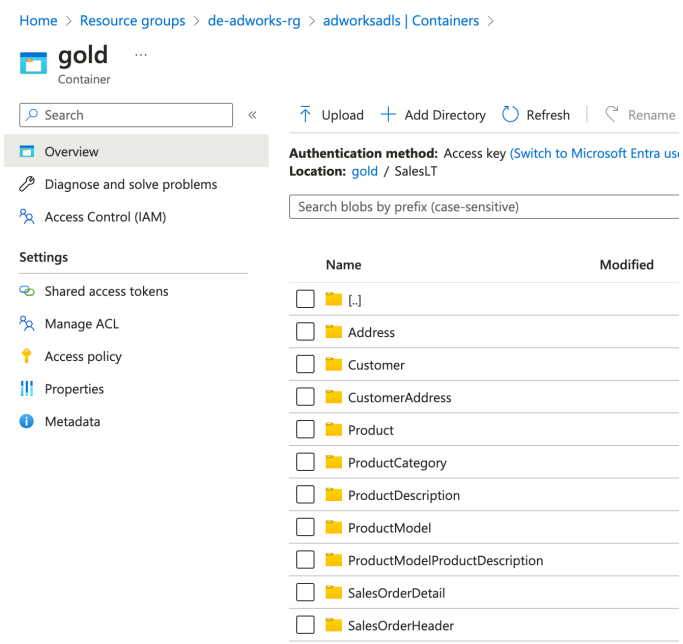
Pipeline run ID e8175844-456a-495c-8c02-ce2aeb4c8f60

All status | List | Monitor in Azure Metrics | Export to CSV

Showing 1 - 14 items

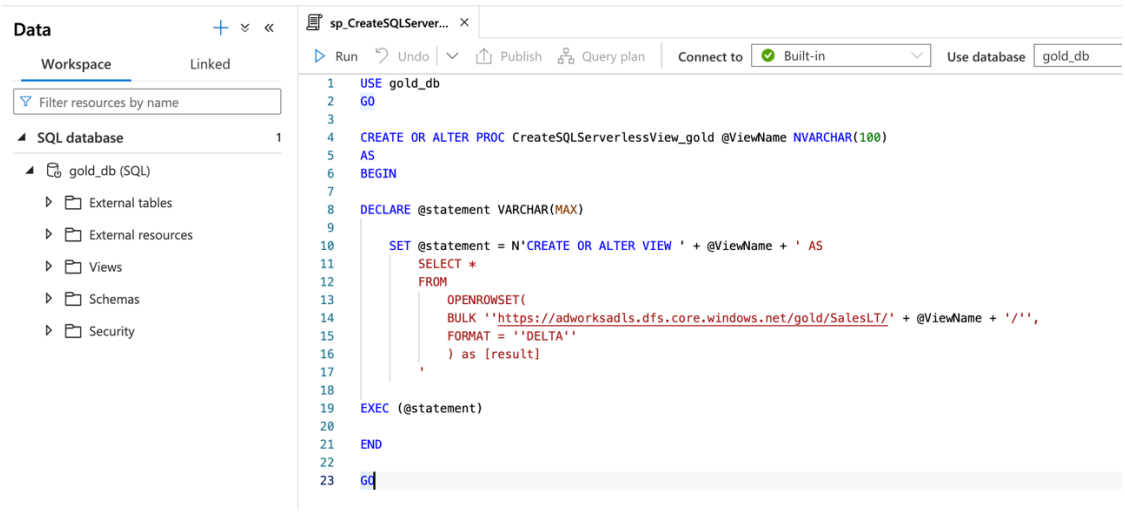
Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Activity run ID
Copy Each Table	Succeeded	Copy data	6/21/2024, 10:22:39 PM	20s	AutoResolveIntegratior		8a60574e-336
Copy Each Table	Succeeded	Copy data	6/21/2024, 10:22:39 PM	18s	AutoResolveIntegratior		91027e70-eae
Copy Each Table	Succeeded	Copy data	6/21/2024, 10:22:39 PM	18s	AutoResolveIntegratior		d89c87bc-ea3
Copy Each Table	Succeeded	Copy data	6/21/2024, 10:22:39 PM	20s	AutoResolveIntegratior		9712870a-3d9
Copy Each Table	Succeeded	Copy data	6/21/2024, 10:22:39 PM	19s	AutoResolveIntegratior		71ab07c2-cd3
Bronze to Silver	Succeeded	Notebook	6/21/2024, 10:23:04 PM	6m 36s	AutoResolveIntegratior		63381695-83e
Silver to Gold	Succeeded	Notebook	6/21/2024, 10:29:40 PM	1m 10s	AutoResolveIntegratior		815547ad-ea3



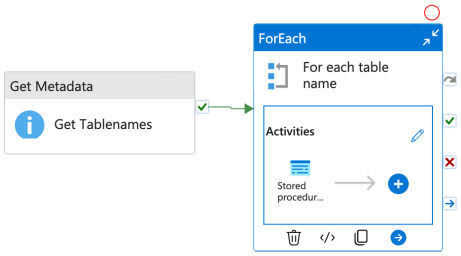


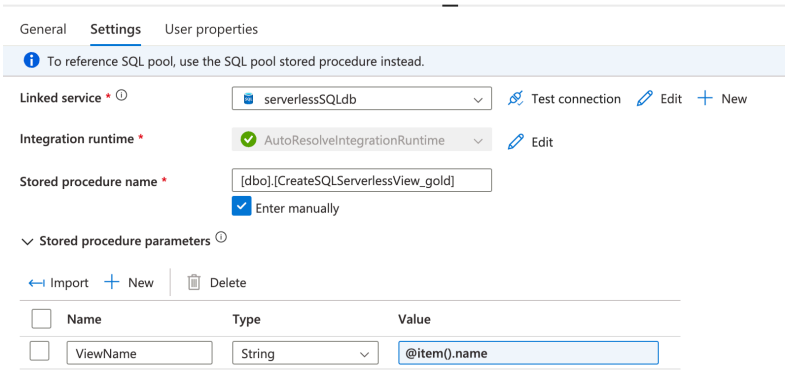
Data Loading: Load the gold table into Azure Synapse to create views for each table

1. Add Azure Synapse into the resource group
   - Azure Synapse built on top of ADF and have more features than it such as creating SQL or Spark notebooks)
2. Create a serverless SQL pool and an Azure SQL database to store all the views
   - Serverless SQL pool come in default, it does not have cost for storage, direct query from ADLS and suitable for smaller workload
   - Dedicated SQL pool need to create a cluster like databricks, it has cost for storage and compute, will load the data into the compute, but have better performance
3. Create a stored procedure to create a view for every tables

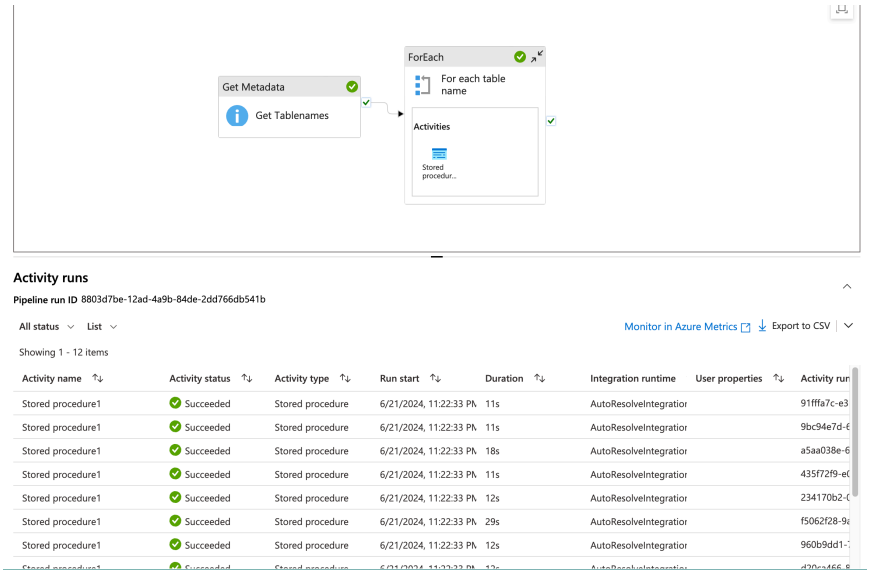


4. Establish linked service to the gold_db for the stored procedure to access to the gold database
5. Create a pipeline to run the stored procedure
6. Add Get Metadata activity to get all the tables in the gold layer
7. Add for each activity by wrapping a notebook activity inside to run the stored procedure for each table

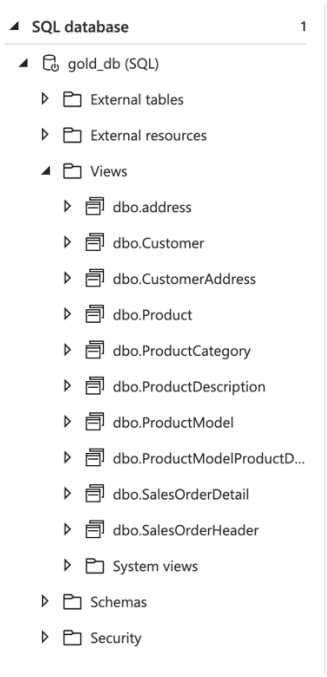




8. Run and test the pipeline
   - Only run the pipeline when there is a schema change



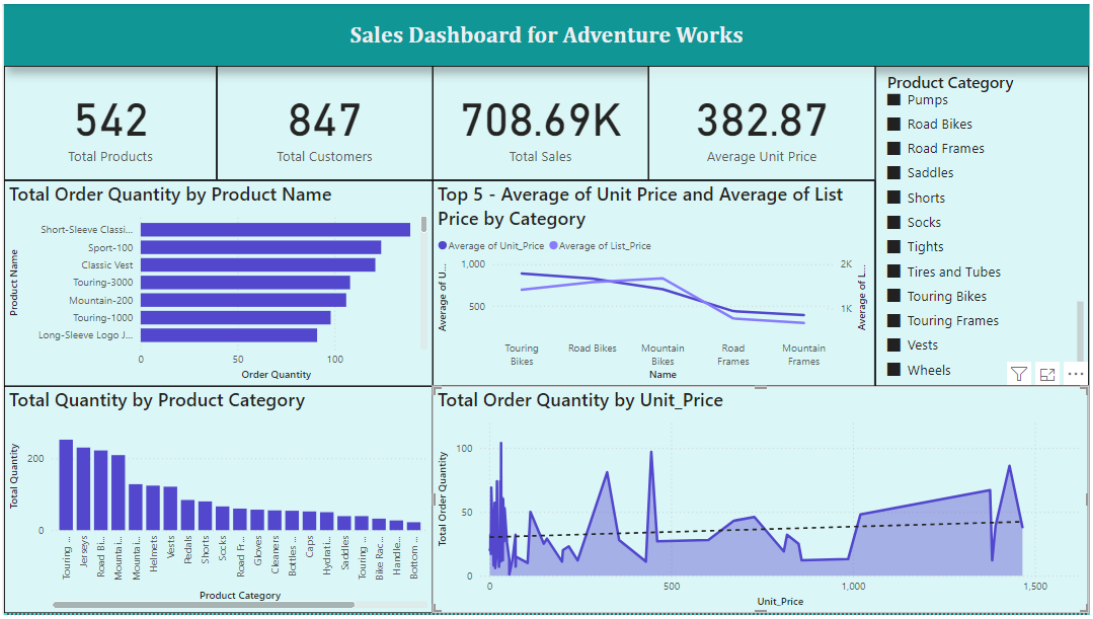
9. Verify if the views are created.



Data Visualization or Reporting:

1. Connect PowerBI to Azure Synapse as our gold tables views are stored there
2. Use Direct Query mode to import the data into PowerBI
   - Import Mode → Loads data into Power BI
   - Direct Query → Direct querying data from Azure Synapse

3. Connect the relationship between customer and address as the relationship is missing between the tables.
4. Create the interactive dashboard.



Completing the data pipeline: Connecting all the activities together

1. Add schedule trigger (daily trigger)
2. Add new rows to the Customer table in Azure SQL raw database

- SET IDENTITY_INSERT [Adventure-Works-d].[SalesLT].[Customer] ON;

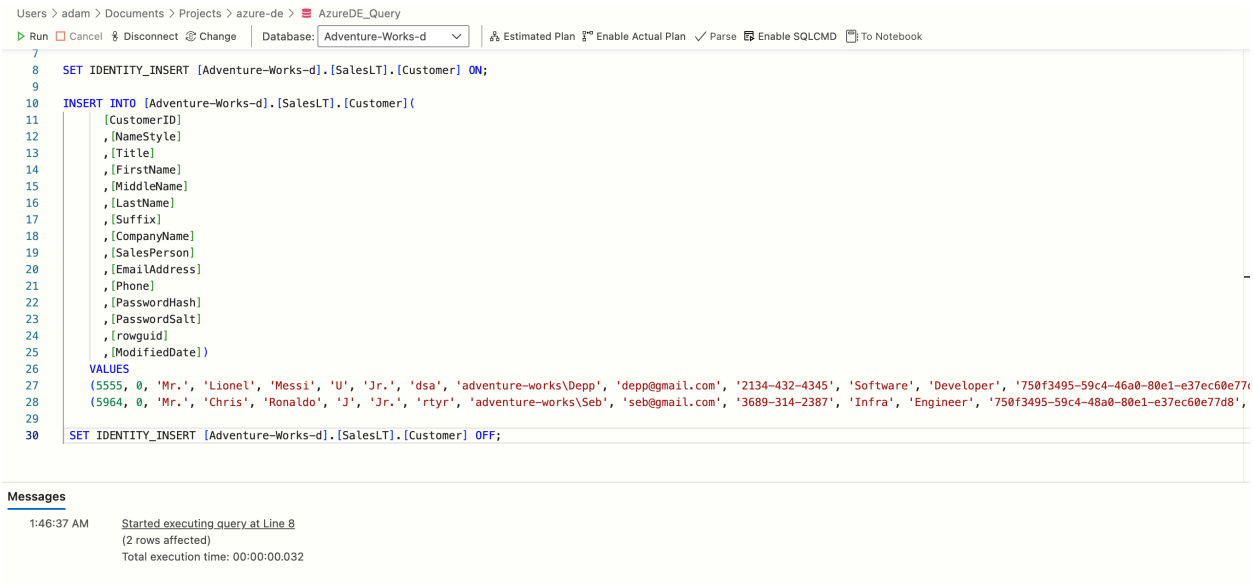
```
INSERT INTO [Adventure-Works-d].[SalesLT].[Customer](
    [CustomerID]
    ,[NameStyle]
    ,[Title]
    ,[FirstName]
    ,[MiddleName]
    ,[LastName]
    ,[Suffix]
    ,[CompanyName]
    ,[SalesPerson]
    ,[EmailAddress]
    ,[Phone]
    ,[PasswordHash]
    ,[PasswordSalt]
    ,[rowguid]
    ,[ModifiedDate])
```

VALUES

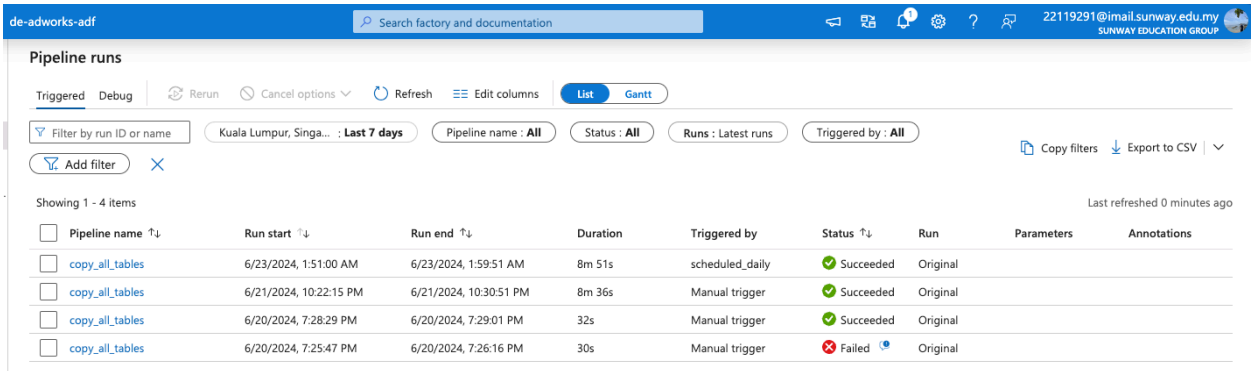
(5555, 0, 'Mr.', 'Lionel', 'Messi', 'U', 'Jr.', 'dsa', 'adventure-works\Depp', 'depp@gmail.com', '2134-432-4345', 'Software', 'Developer', '750f3495-59c4-46a0-80e1-e37ec60e77d9', '2006-08-01 00:00:00.000'),

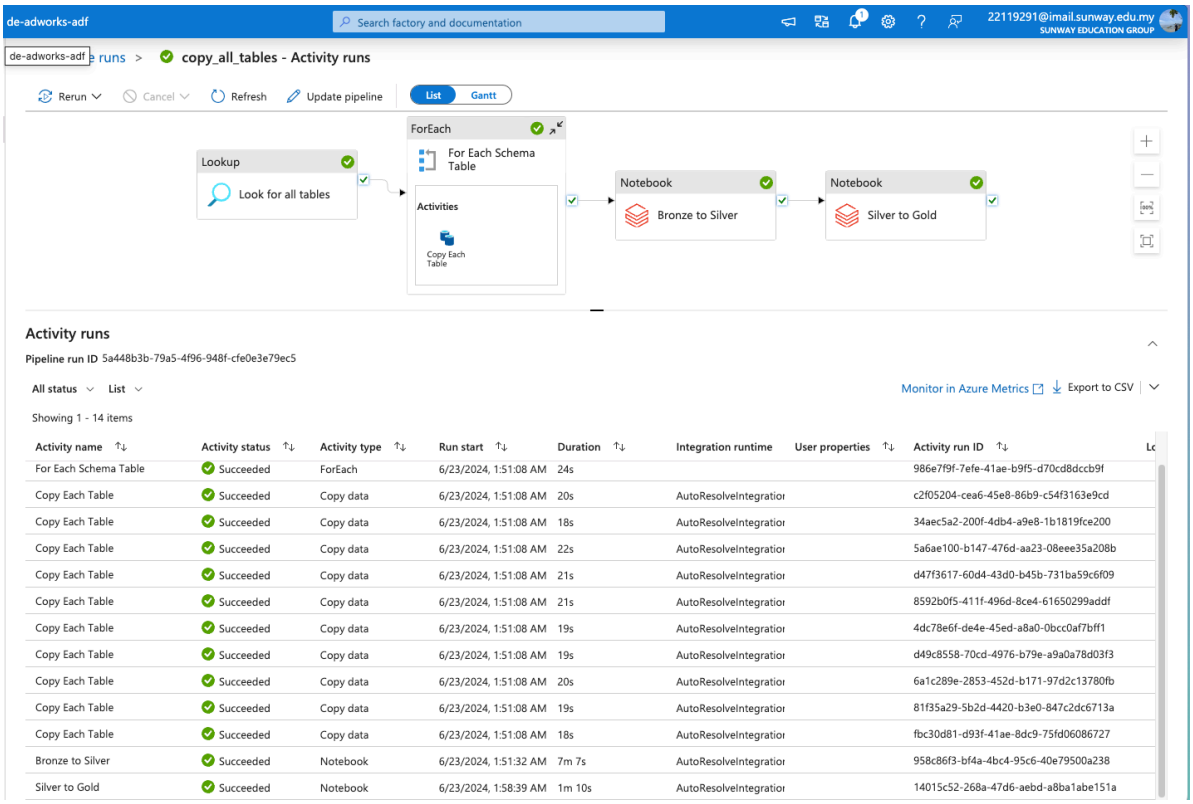
(5964, 0, 'Mr.', 'Chris', 'Ronaldo', 'J', 'Jr.', 'rtyr', 'adventure-works\Seb', 'seb@gmail.com', '3689-314-2387', 'Infra', 'Engineer', '750f3495-59c4-48a0-80e1-e37ec60e77d8', '2006-08-01 00:00:00.000')

SET IDENTITY_INSERT [Adventure-Works-d].[SalesLT].[Customer] OFF;



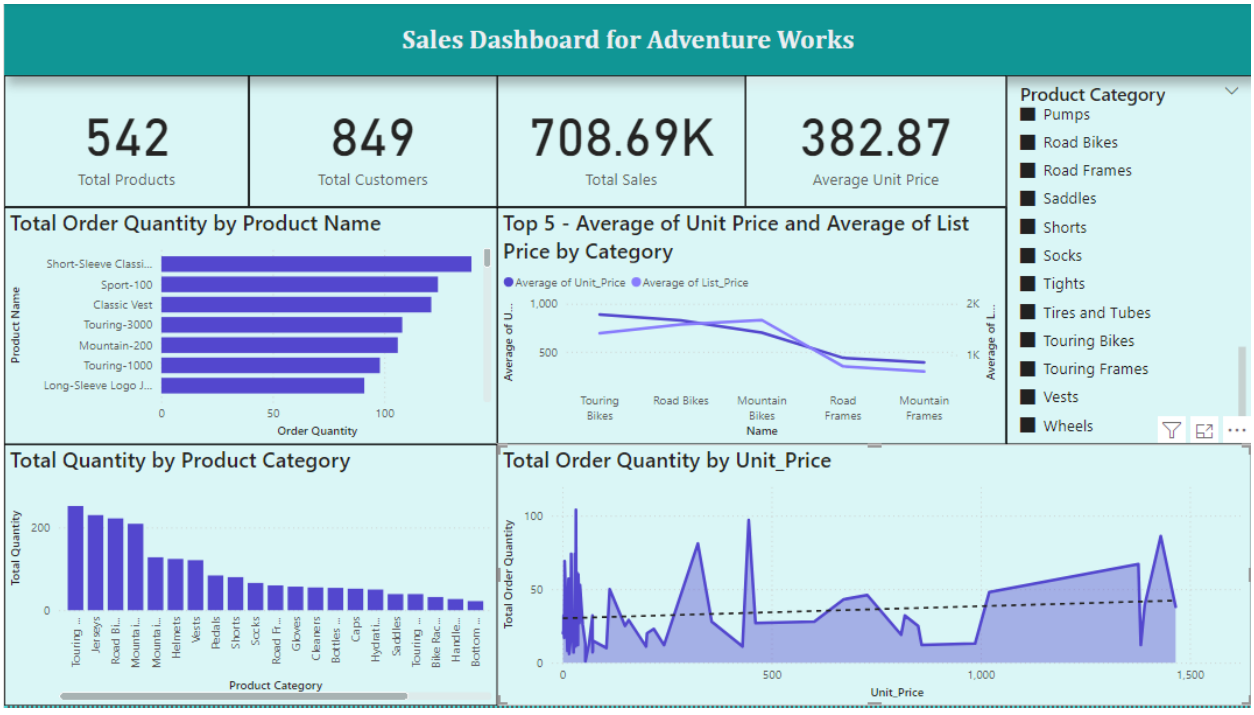
3. Test and run the whole pipeline



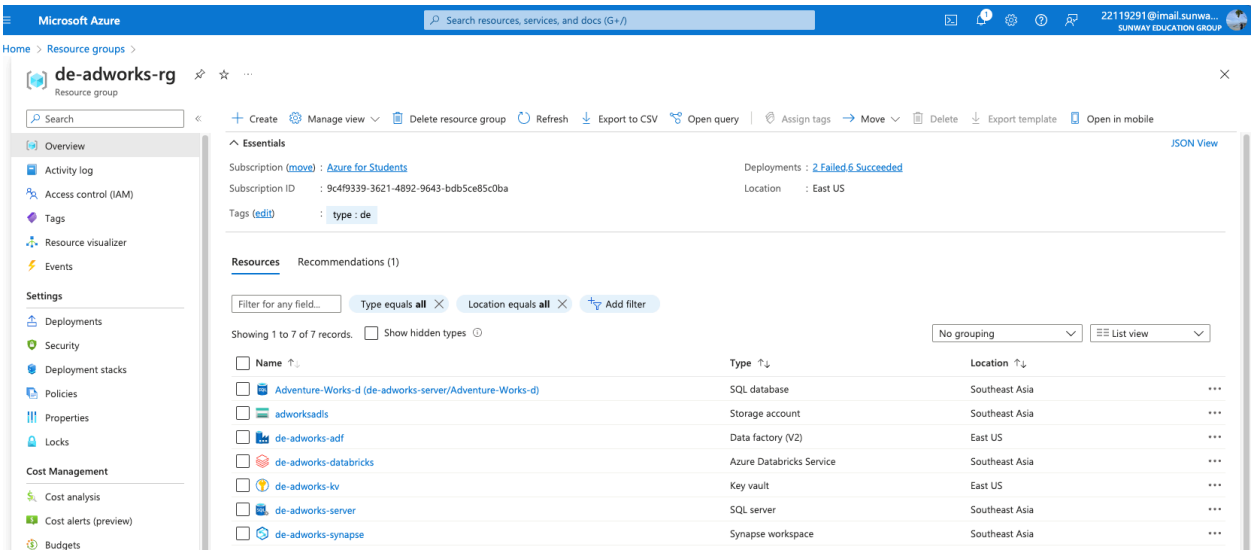


4. Validate result in Power BI dashboard

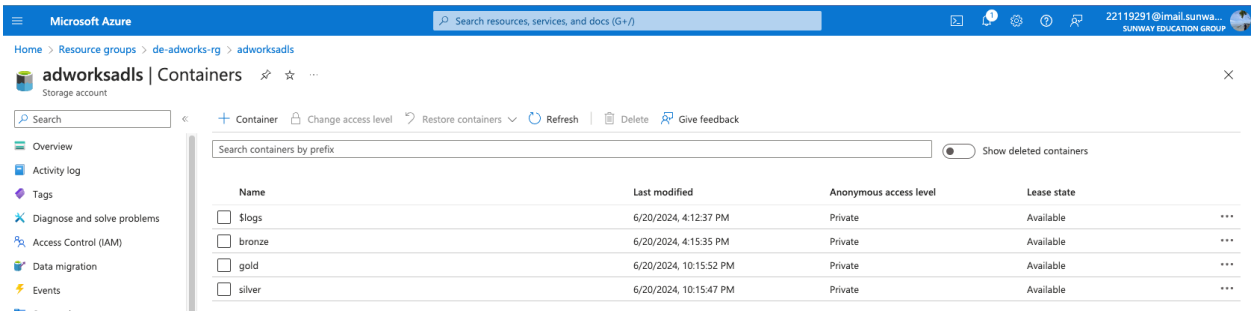
- The Total Customers has increased 2, from 847 to 849 as two new rows are added into the dataset.



Resource group:



Storage File Structure:



Azure Key Vault: