Azure e-commerce pipeline:

-Aravind Swamy

Project Introduction

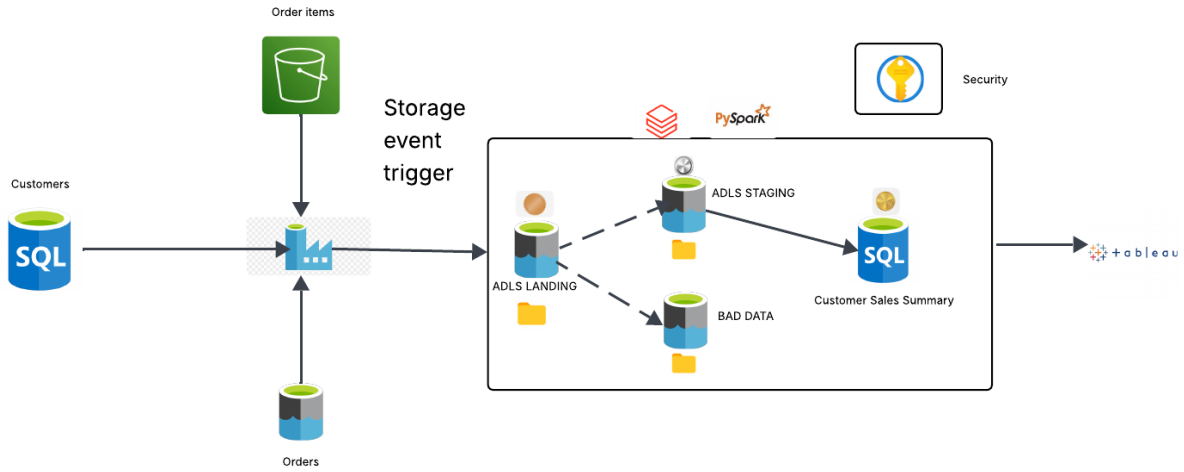
This project focuses on building a real-world, end-to-end data engineering pipeline using Microsoft Azure to simulate an e-commerce data processing system. The objective was to design an automated solution where incoming order data from third-party sources is ingested, validated, transformed, and stored for analytical consumption.

The pipeline is triggered automatically when new files arrive in Azure Data Lake Storage Gen2 and executes validation logic using Azure Databricks and PySpark. Data quality checks include duplicate detection and dynamic validation of order status using a lookup table maintained in Azure SQL Database. Based on validation results, data is either promoted to the staging layer or moved to a discarded layer.

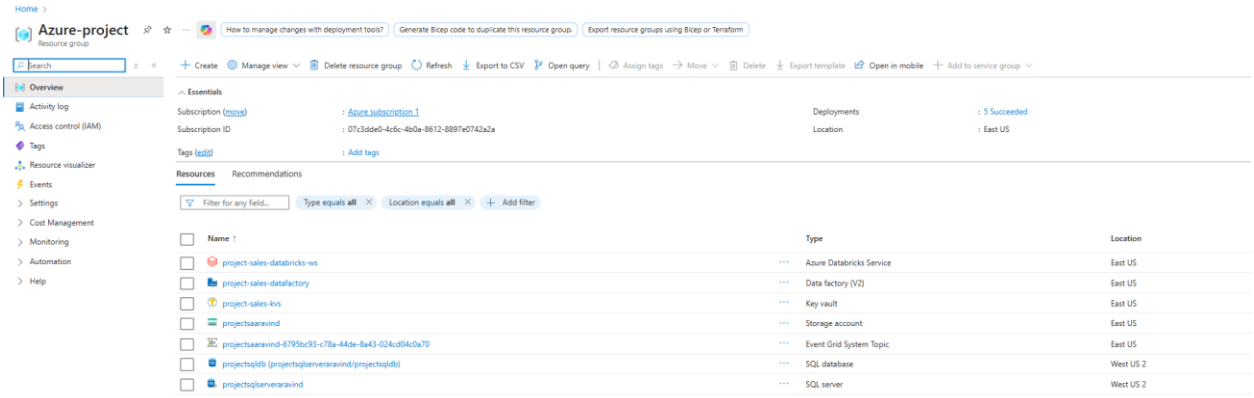
The solution follows the Medallion Architecture (Bronze–Silver–Gold) and includes secure secret management using Azure Key Vault, parameterized pipelines for dynamic file handling, and cross-cloud data ingestion from Amazon S3. The final enriched dataset is stored in Azure SQL Database and used to generate key business insights such as total orders and customer spend.

The project follows a modern **event-driven architecture** and leverages **Azure Data Factory, Azure Databricks, Azure Data Lake Storage Gen2, Azure SQL Database, Amazon S3, and Azure Key Vault**.

Architecture:



Pipeline



In the initial pipeline as soon as a third-party service drops a file named orders.csv into the landing folder of Azure Data Lake Storage Gen2.

Business Requirements

As soon as the file arrives:

1. There should be no duplicate order_id in the file.
2. The order_status should be valid. (which will be cross checked against the valid status stored in Azure SQL DB)

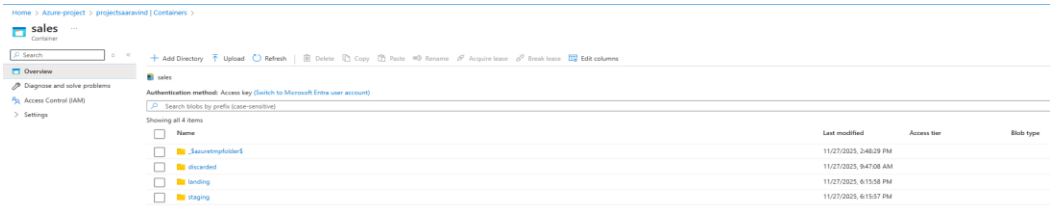
If both conditions are satisfied:

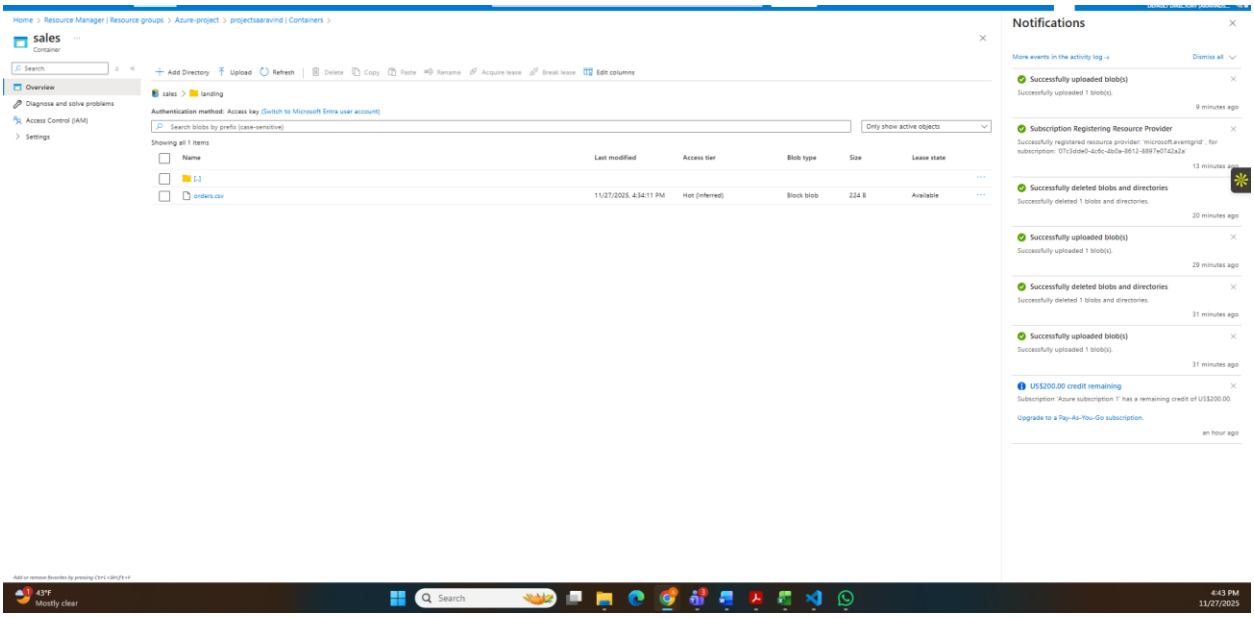
- Move the file to the **staging folder.**

If any condition fails:

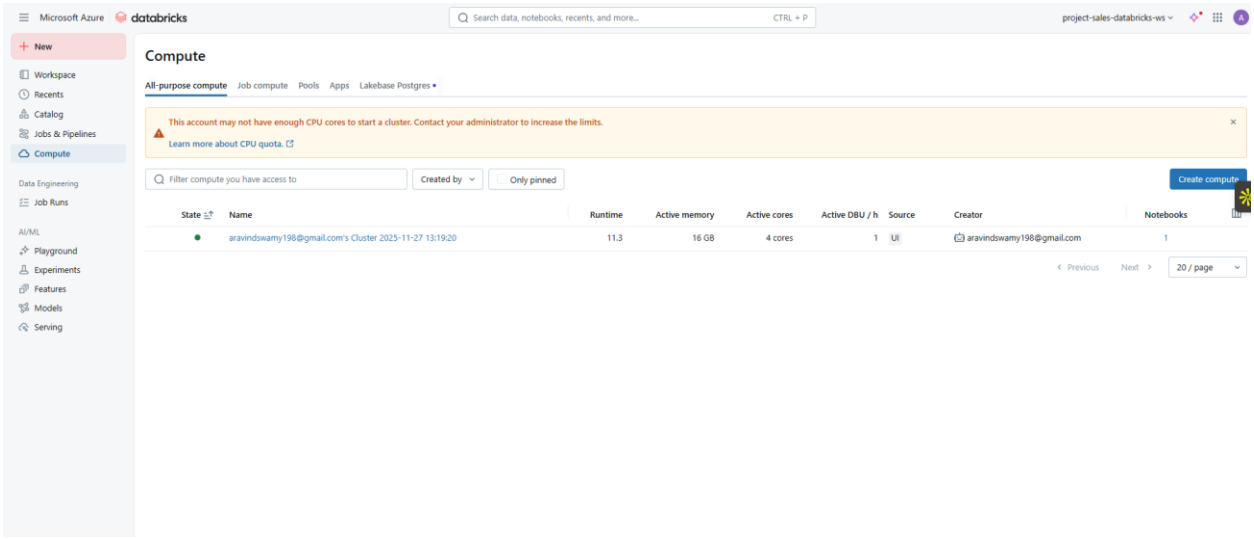
- Move the file to the **discarded folder.**

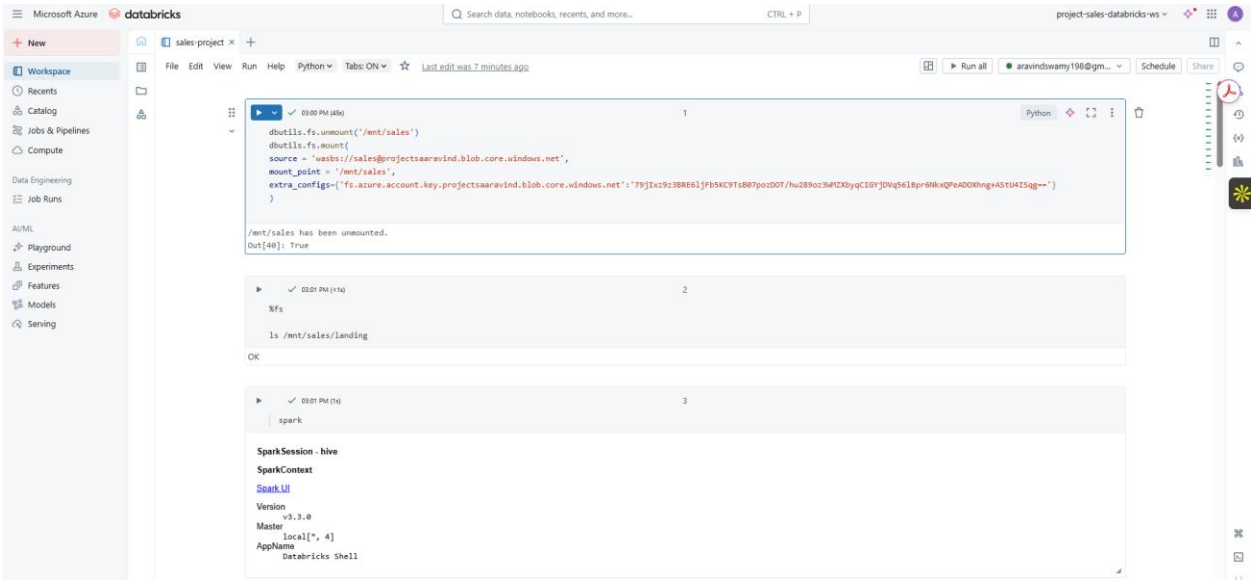
Thus we first create the 3 folders first



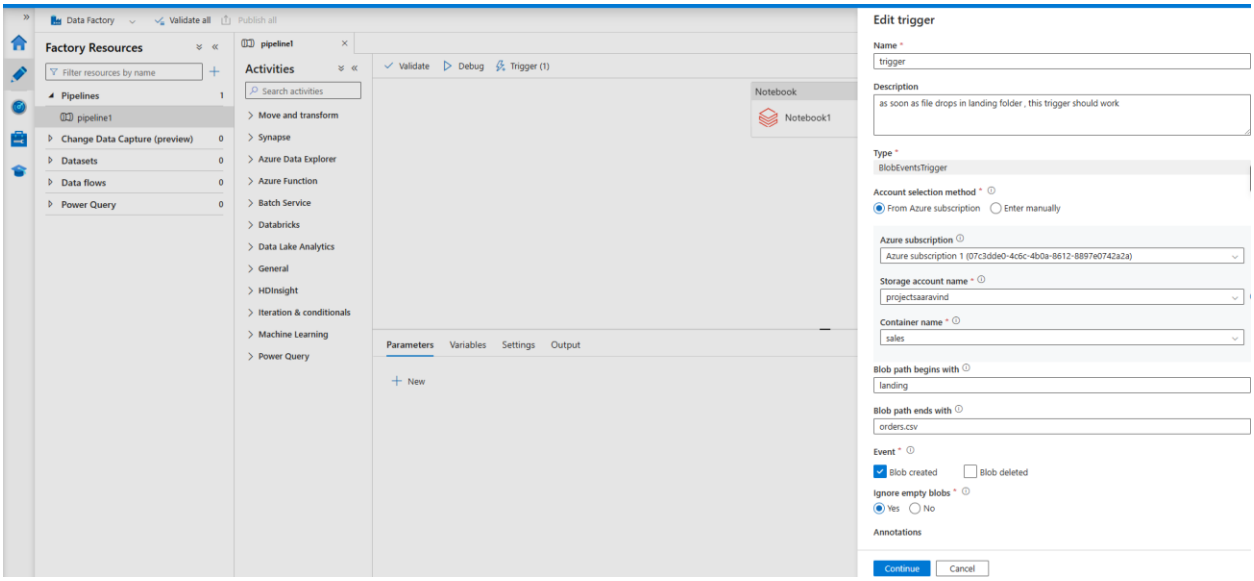


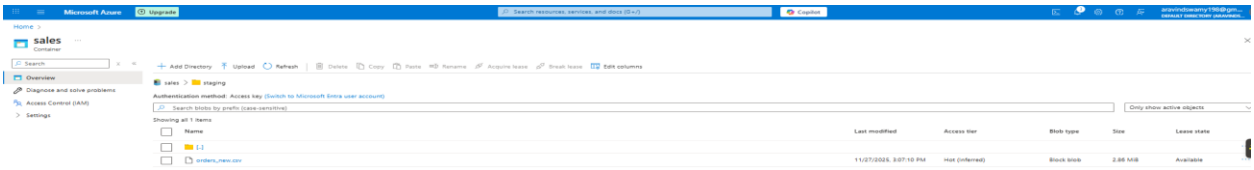
And we set up Azure Databricks cluster and write our validation and transformation code in pyspark





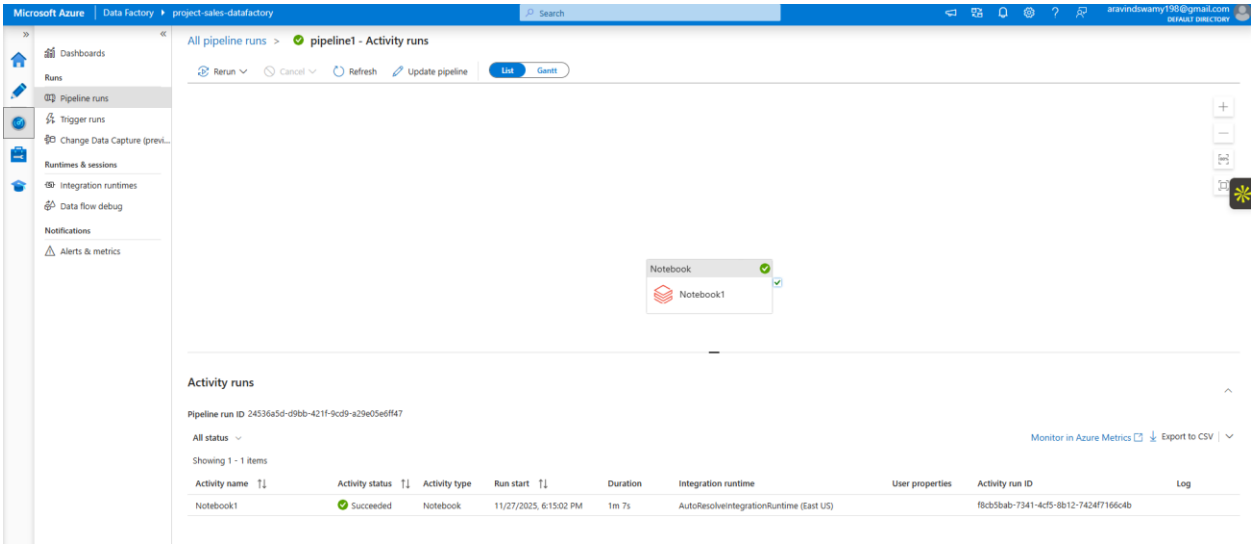
And we set up the trigger



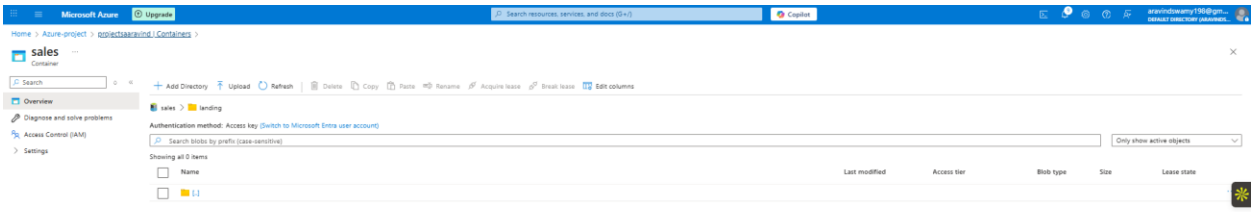


We have also added the file orders_new.csv here to prove that our pipeline is *generic and dynamic*, not hardcoded to orders.csv

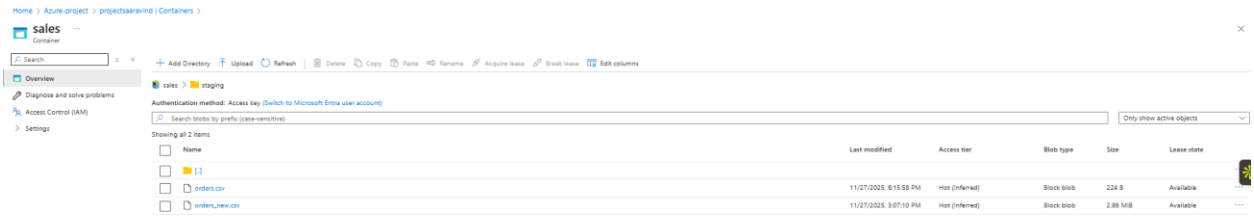
Now the pipeline is run



After ADF

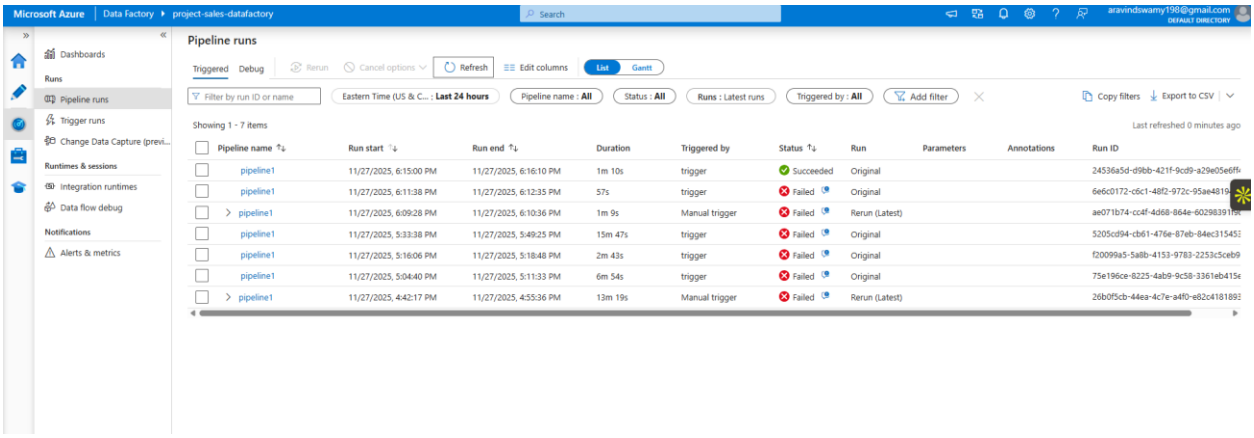


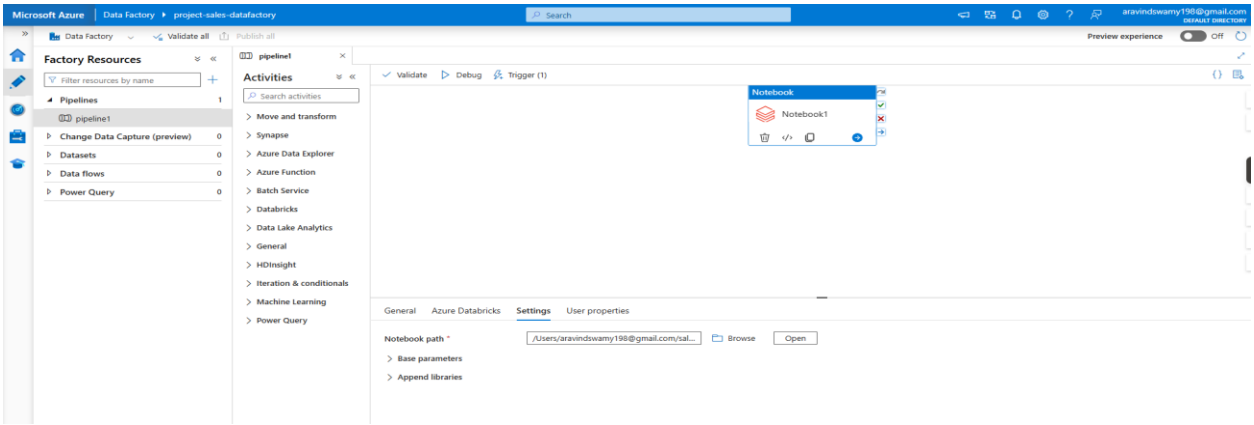
We can see that the landing folder is empty



And both the files have been transferred to the staging folder.

Pipeline trigger runs





Linked services were set up in order to connect various elements of our pipeline

Edit linked service

Azure Databricks Learn more

AzureDatabricks1project_ls

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Account selection method *

From Azure subscription Enter manually

Databrick Workspace URL *

https://adb-12378614920109.9.azuredatabricks.net

Authentication type *

Access Token

Access token Azure Key Vault

AKV linked service *

AzureKeyVault1project_ls

Secret name *

databricks-access-token

Edit

Secret version

Latest version

Edit

Select cluster

Existing interactive cluster

Existing cluster ID *

Save Cancel Test connection

Edit linked service

Azure Data Lake Storage Gen2 Learn more

Name *

AzureDataLakeStorage1projectls

Description

To link DF to Storage

Connect via integration runtime *

AutoResolveIntegrationRuntime

Authentication type

Account key

Account selection method

From Azure subscription | Enter manually

URL *

https://projectsaaravind.dfs.core.windows.net/

Storage account key | Azure Key Vault

Storage account key *

••••••••••

Test connection

To linked service | To file path

Annotations

+ New

> Parameters

> Advanced

Save | Cancel | Test connection

Edit linked service

Azure Key Vault

Name *

AzureKeyVault1project_ls

Description

Azure key vault selection method

○ From Azure subscription ◉ Enter manually

Base URL *

https://project-sales-kvs.vault.azure.net/

Authentication method

System-assigned managed identity

Managed identity name: **project-sales-datafactory**
Managed identity object ID: **2bd302fe-123e-47b2-9753-b88bc6662703**
Grant Data Factory service managed identity access to your Azure Key Vault. Learn more

Test connection

◉ To linked service ○ To secret

Annotations

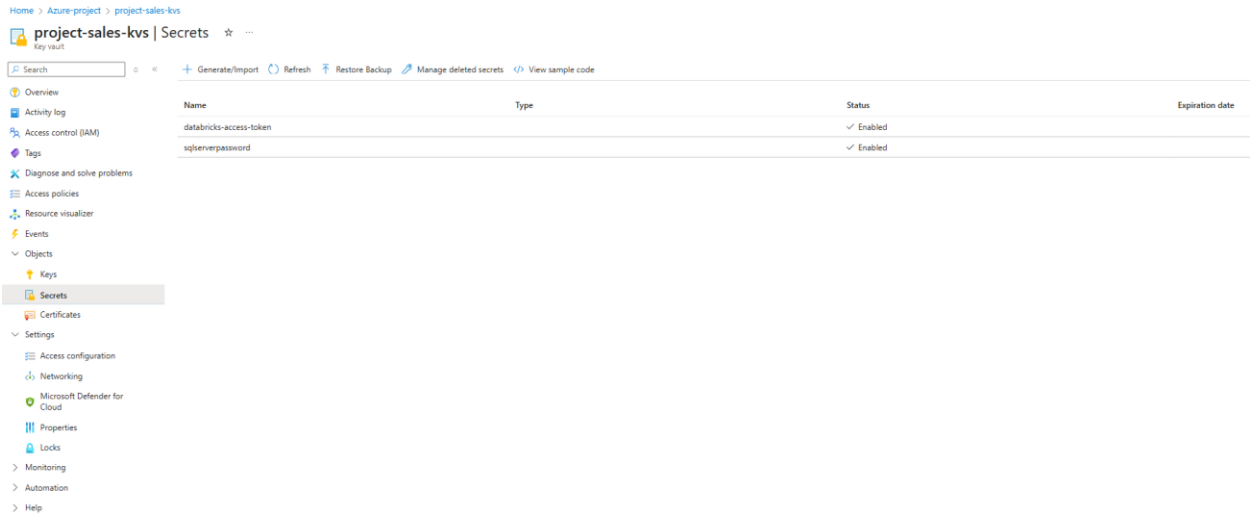
+ New

> Parameters

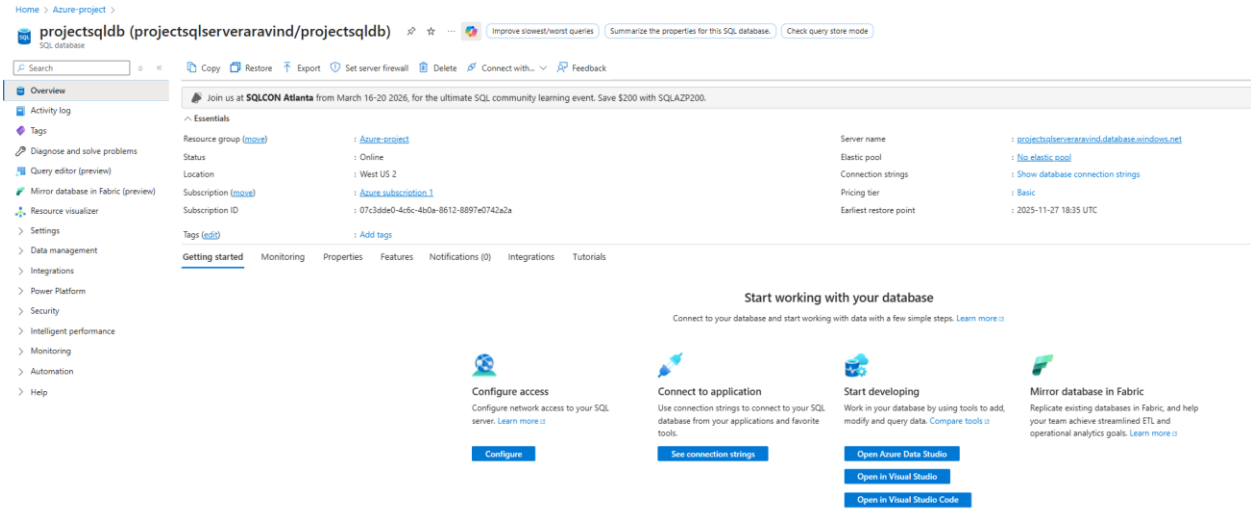
> Advanced

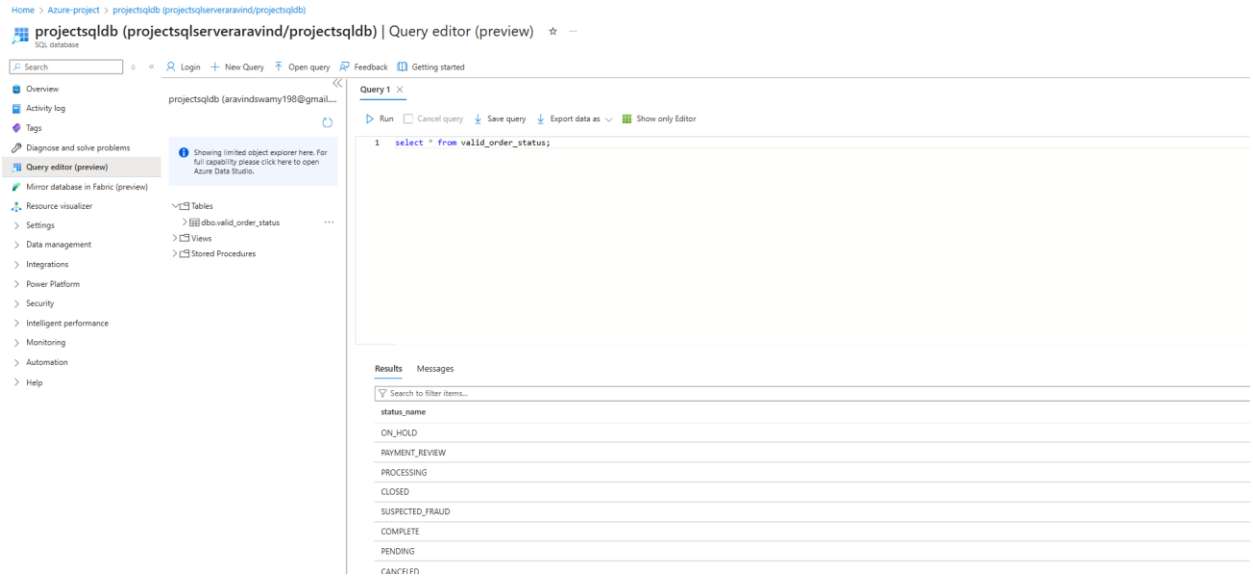
Save | Cancel | Test connection

The key vault is used to store various secret keys and passwords.

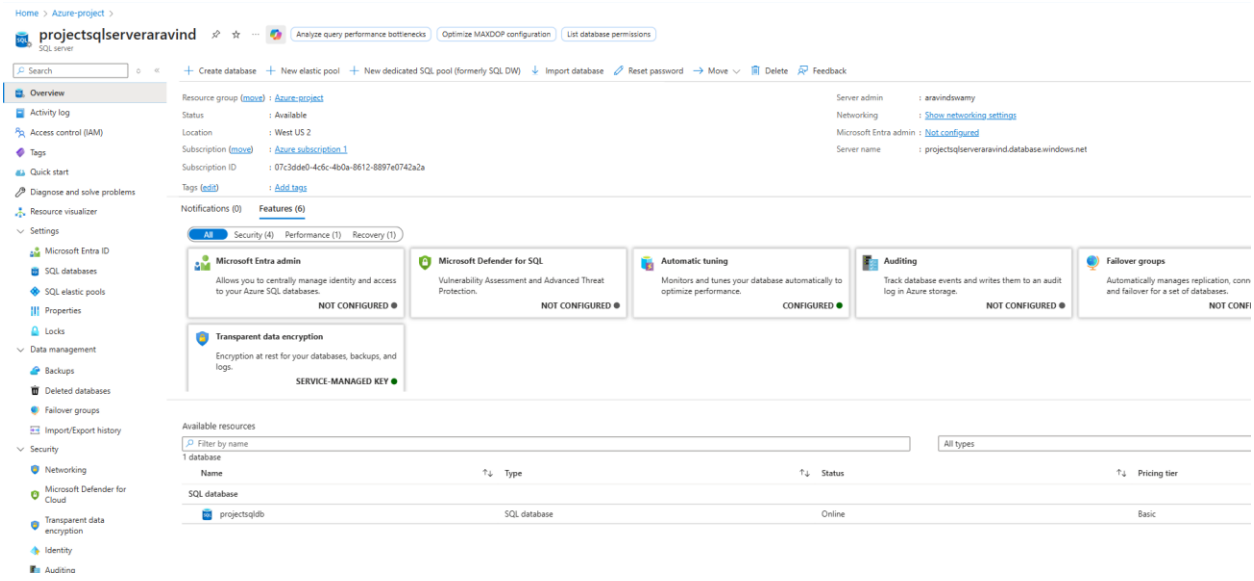


Azure SQL DB is created





And a table is created this particular table will hold the valid order statuses that are recognized. This is to check the valid orders based on the status.



Now showing the parameterized Approach: (Any file) and not hardcoded

Edit trigger

Trigger Run Parameters

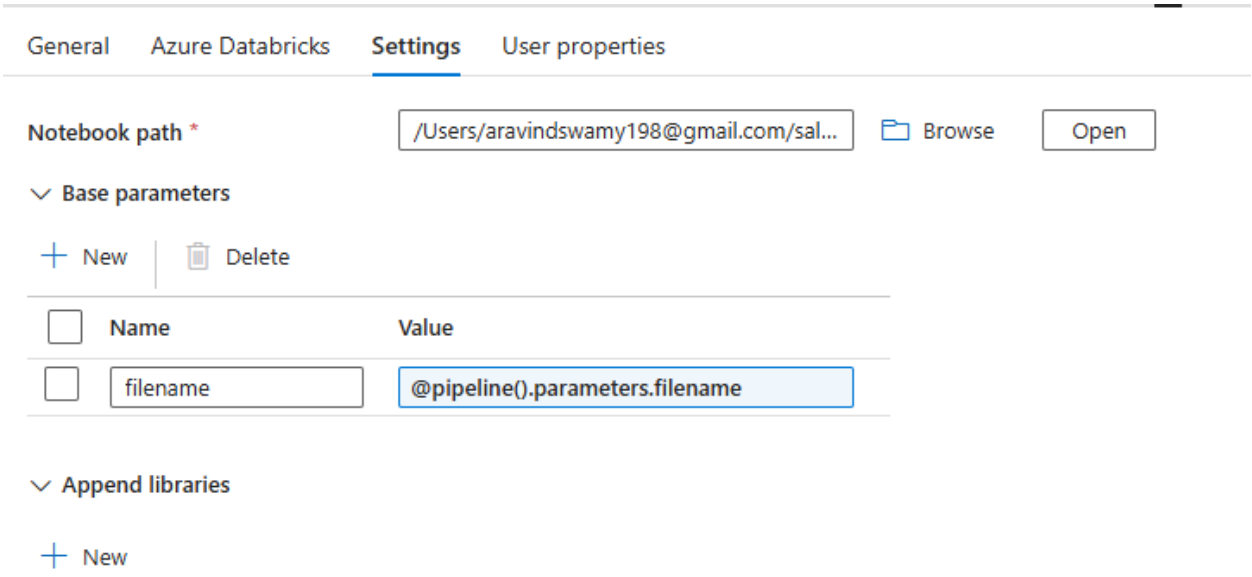
Parameters that are not provided a value will not be included in the trigger.

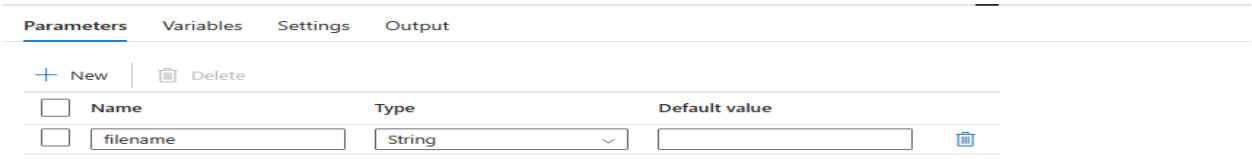
Name	Type	Value
filename	string	@triggerBody().filename

Make sure to "Publish" for trigger to be activated after clicking "OK"

OK

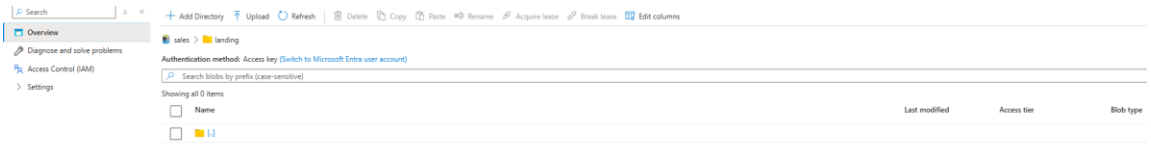
Cancel

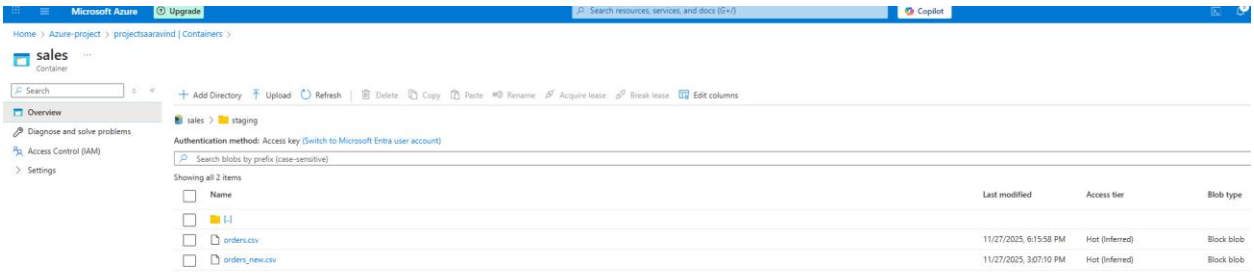




Now to show that it works,

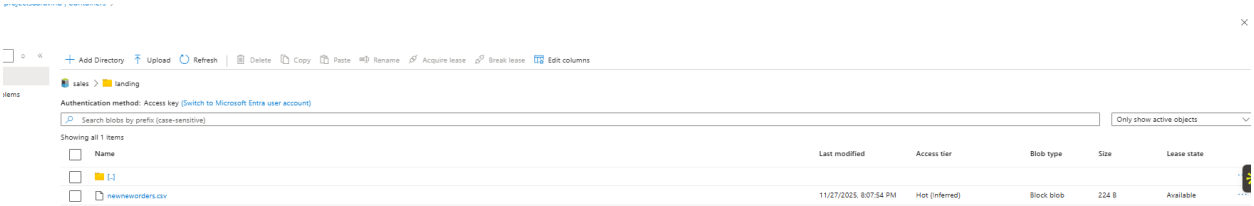
The landing folder is currently empty

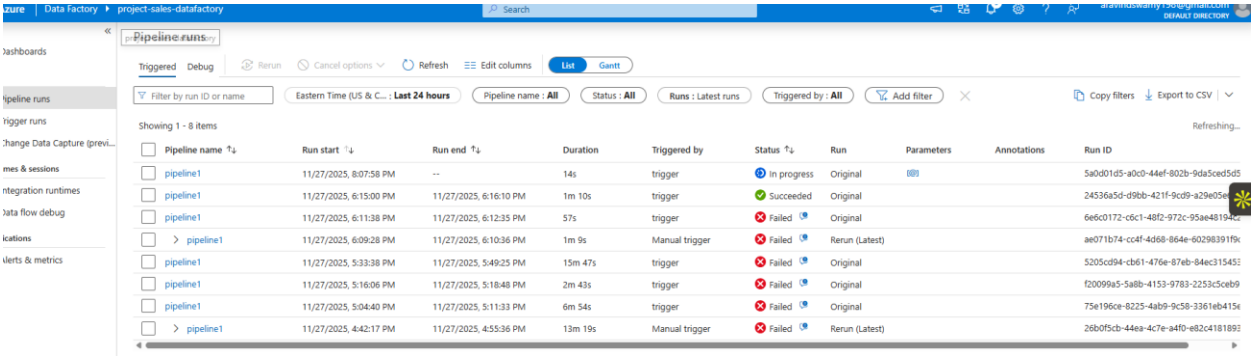




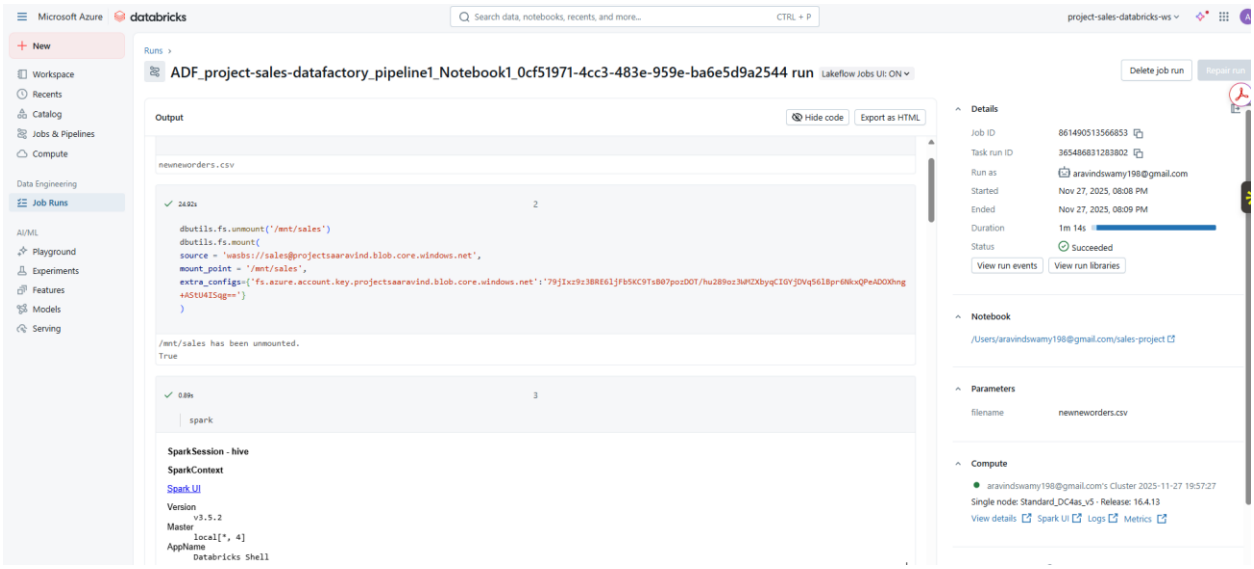
And the staging folder also doesn't have the data that we are going to add.

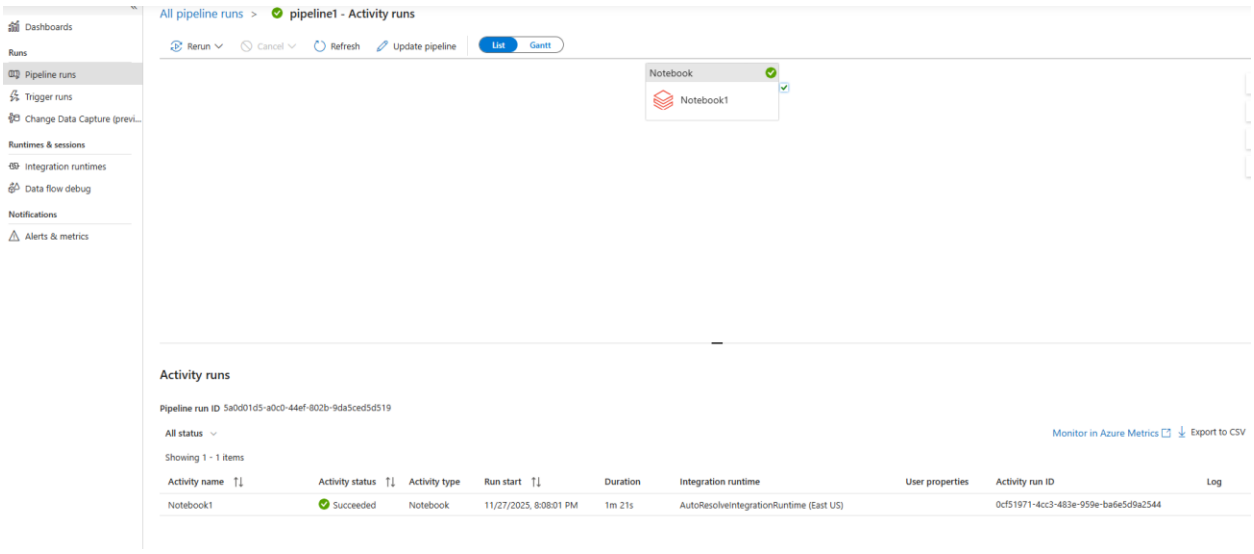
Now triggering the pipeline by uploading a new file in the landing folder,



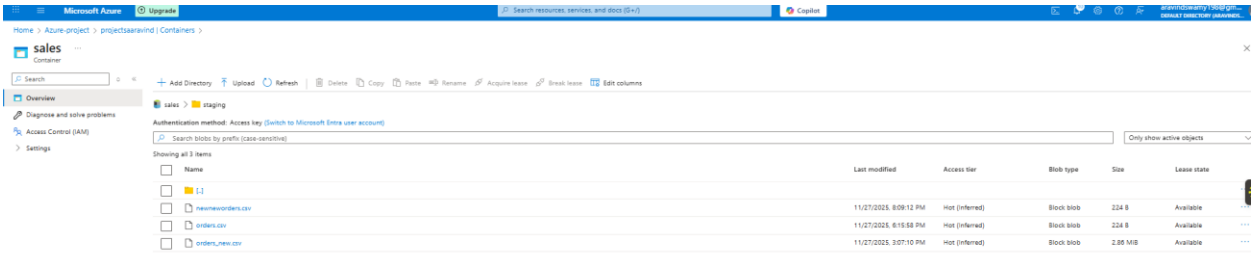


Succeeded





It works, now we can see the new file added to our staging area.



Making the pipeline more generic (generic mount code):

```
alreadyMounted = False
for x in dbutils.fs.mounts() :
    if x.mountPoint == "/mnt/sales" :
        alreadyMounted = True
        break
    else:
        alreadyMounted = False
print(alreadyMounted)
```

```
storageAccountKey = dbutils.secrets.get(scope = databricksScope,key='storage-account-key')
```

```
if not alreadyMounted :
    dbutils.fs.mount(
source = 'wasbs://sales@projectsaaravind.blob.core.windows.net',
mount_point = '/mnt/sales',
extra_configs={'fs.azure.account.key.projectsaaravind.blob.core.windows.net':storageAccountKey}
)
    alreadyMounted = True
    print('Mounting done successfully')
else:
    print('Already mounted')
```

```
/mnt/sales has been unmounted.
Out[40]: True
```

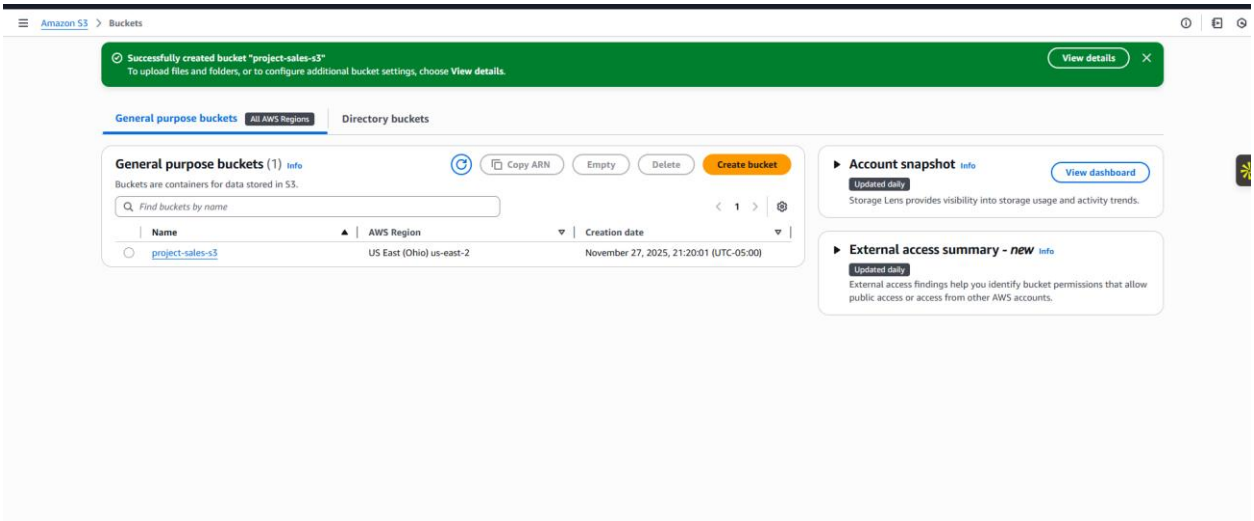
We have also stored the storage account key in key vault rather than using it hardcoded.

Now for our project, we need two more datasets in addition to our orders.csv data

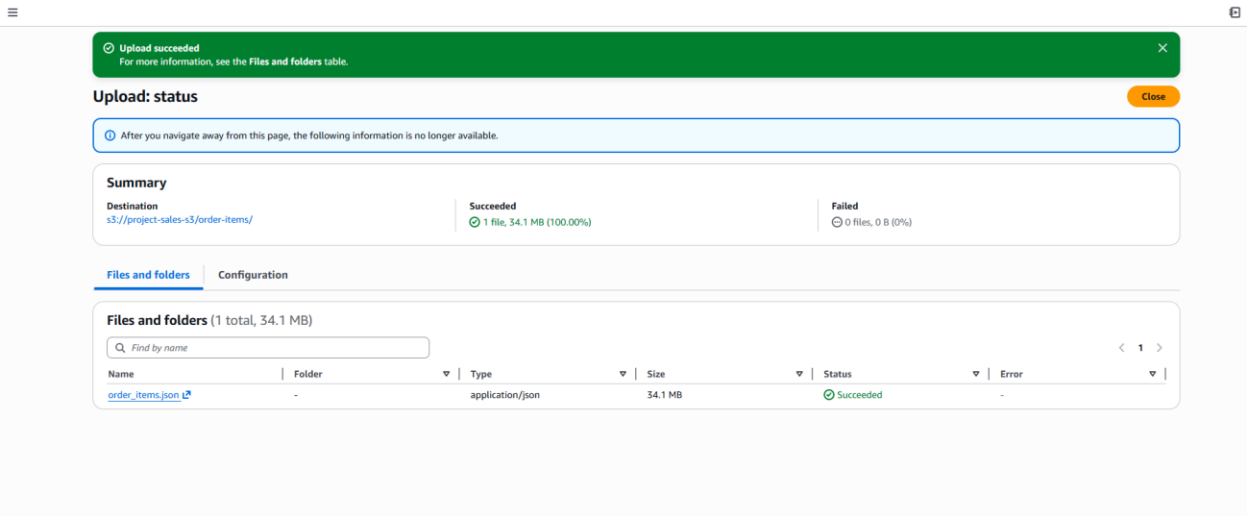
Order items will come from third party (say amazon s3)

Customers (will be from azure sql db)

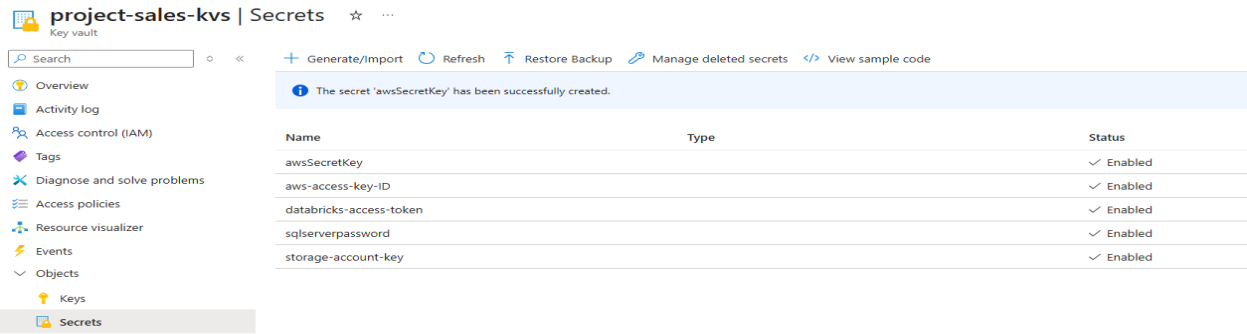
Now we have created a bucket in amazon s3



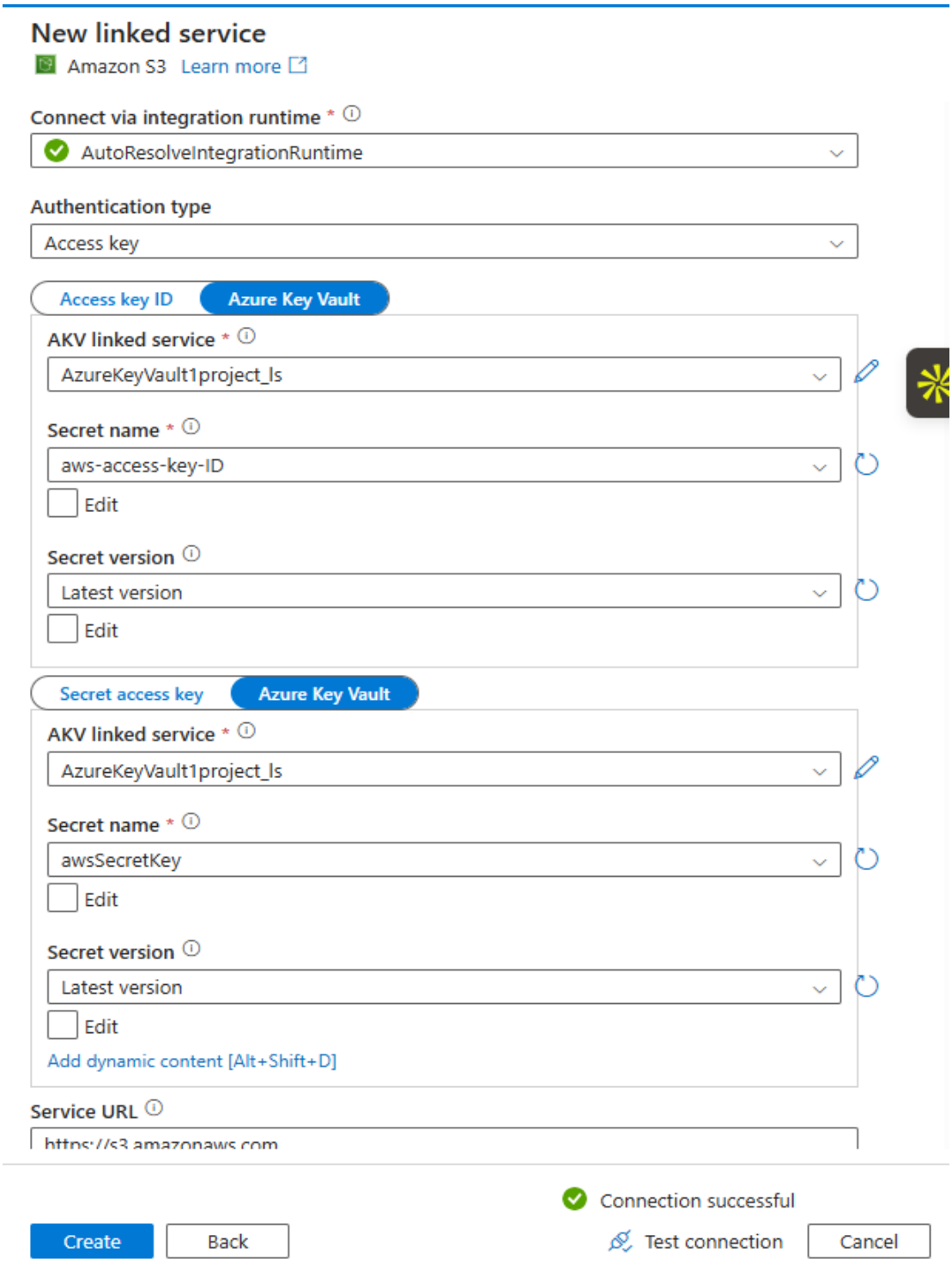
File uploaded



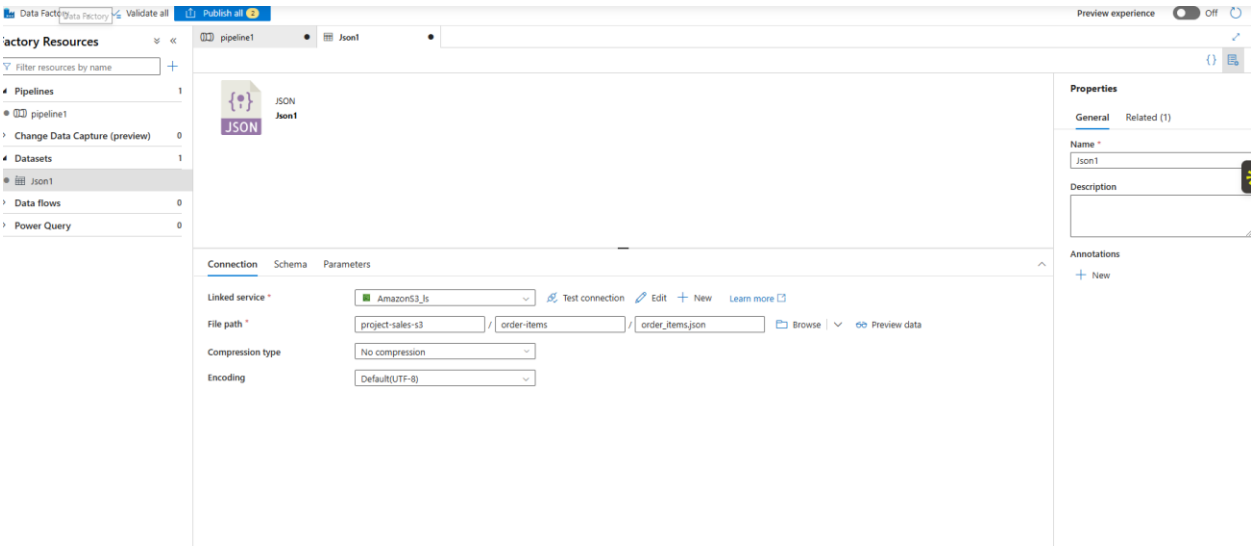
Now I create IAM role to access this file from outside and after that storing those keys in azure key vault



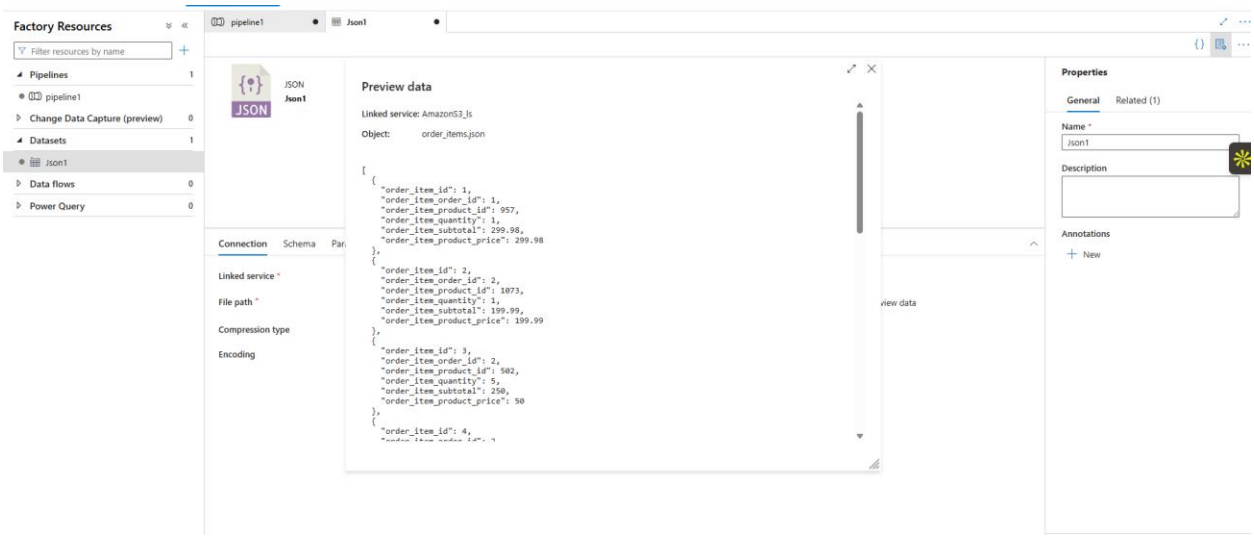
Now going to ADF and creating a linked service to connect to aws s3



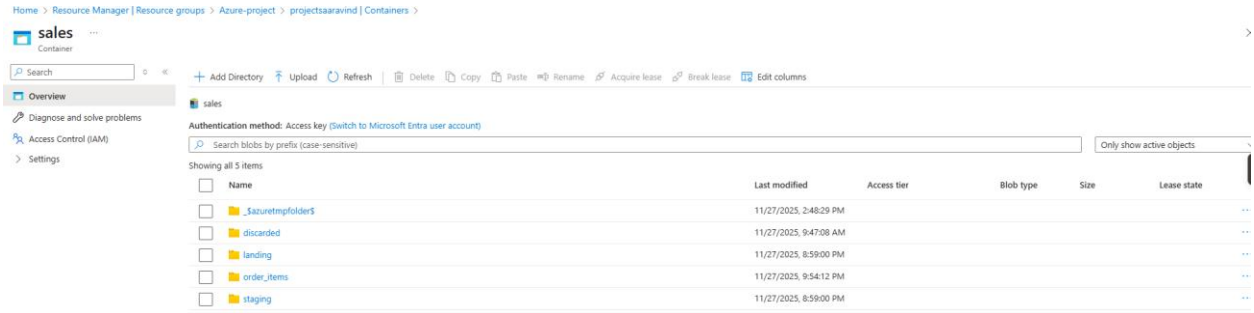
Now as soon as orders file arrive in the landing folder, of ADLS Gen 2, we need to also bring the order_items file from s3 to adls gen2



And its working

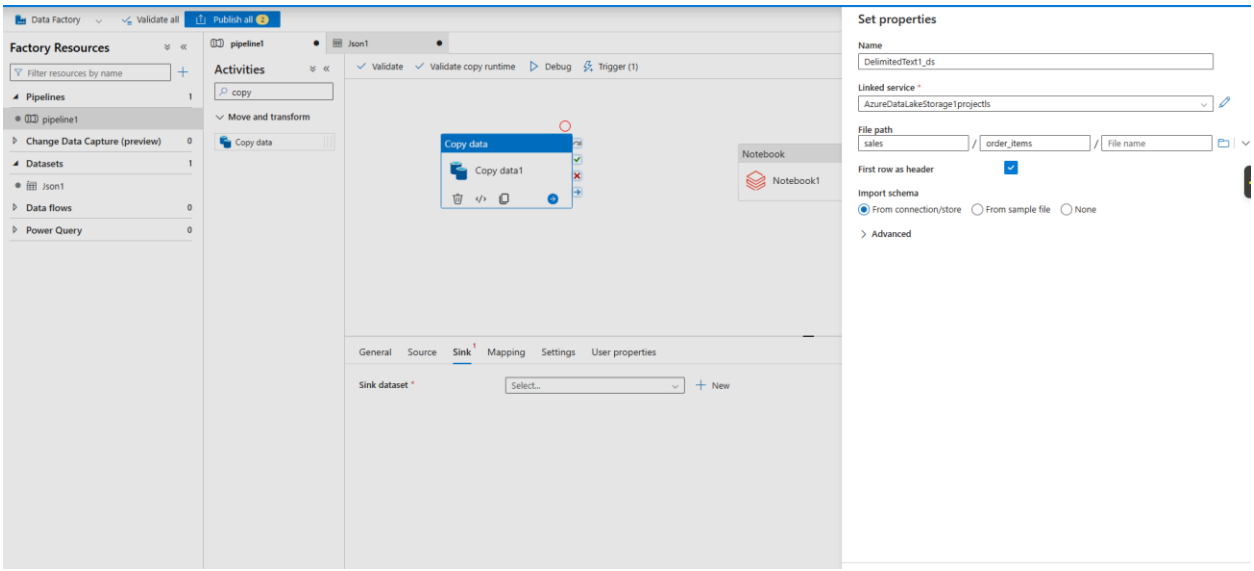


Setting up sink destination

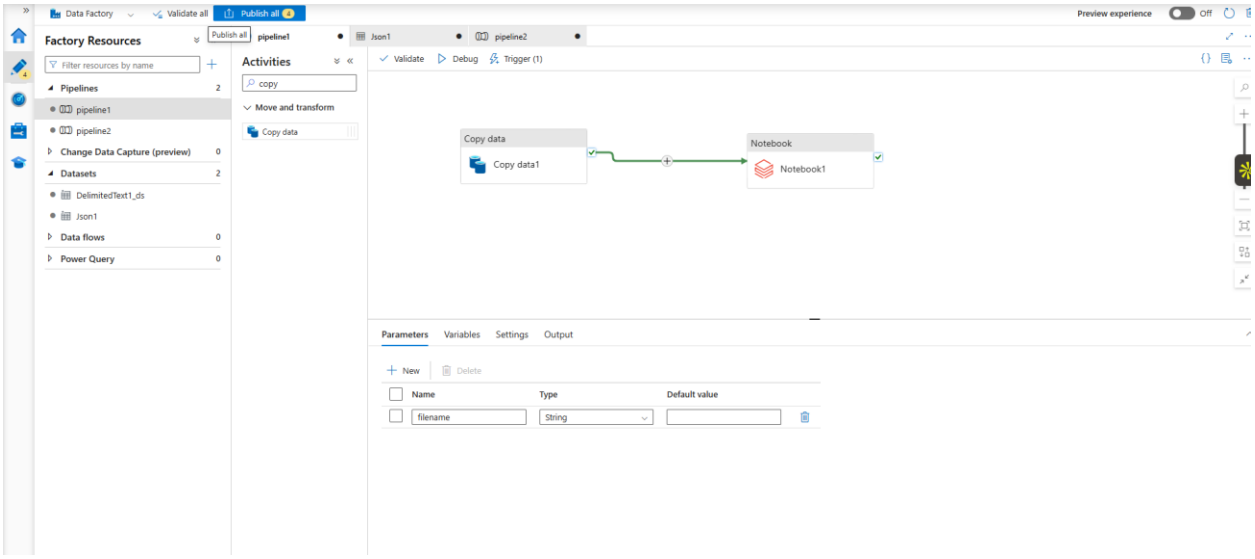


And we want the dataset in the sink (adls gen2) in csv format

Setting up sink



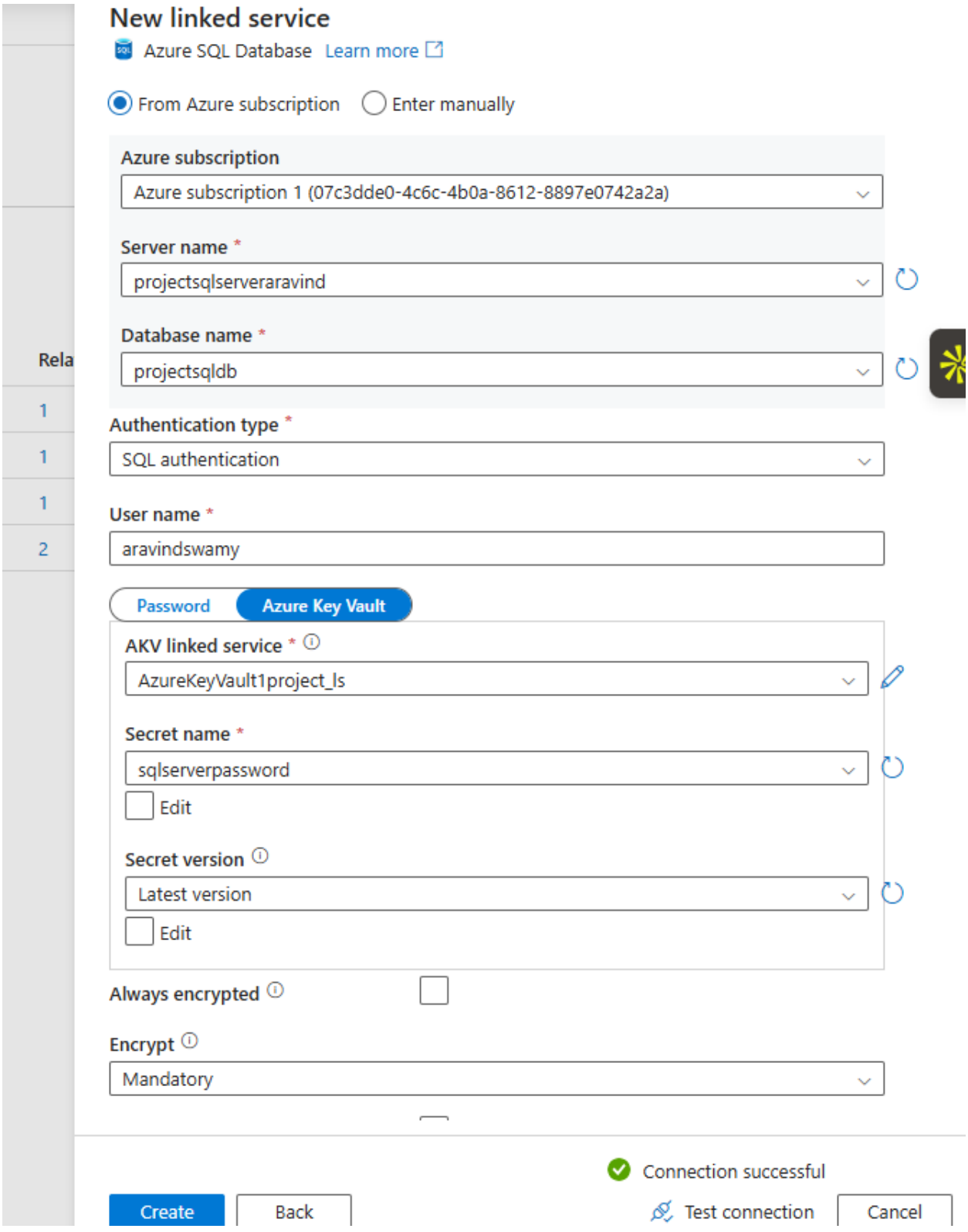
Now as soon as a file arrives in the landing folder,

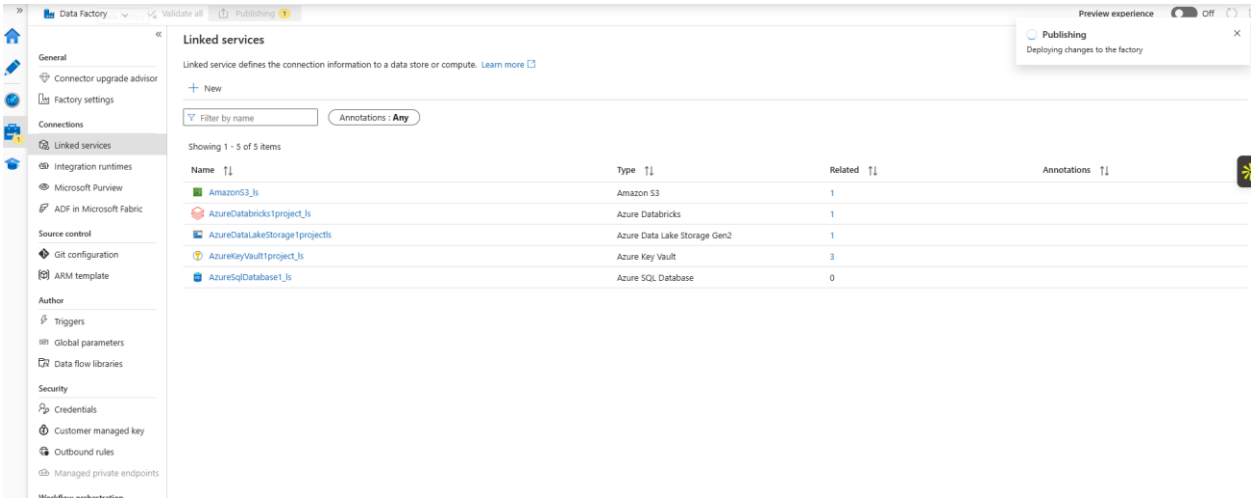


A copy data activity takes place which converts the json from s3 and puts it as csv in the adls gen 2 and after that the notebook is invoked

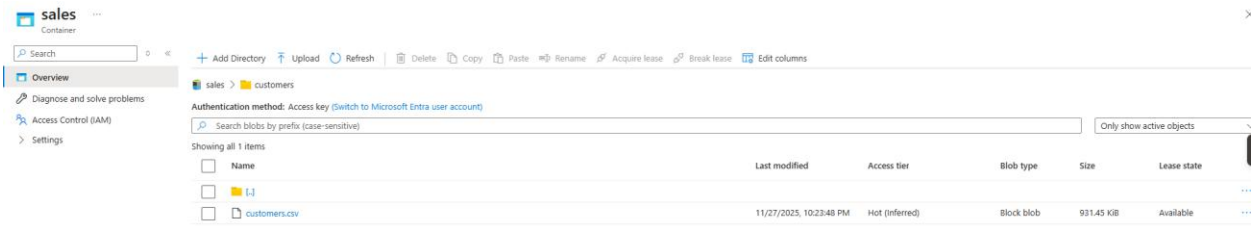
Now we create customer table in Azure SQL DB (Background activity) which will be our third data source in addition to our orders and order_items data

Creating a new Linked service for it

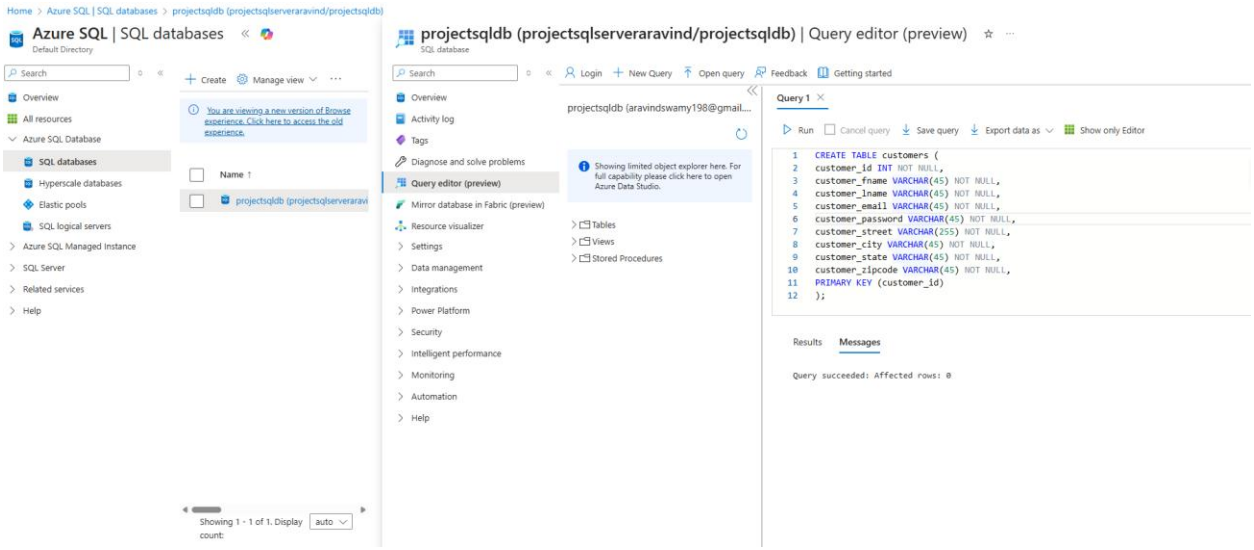

New linked service
Azure SQL Database
Learn more
From Azure subscription
Enter manually
Azure subscription
Azure subscription 1 (07c3dde0-4c6c-4b0a-8612-8897e0742a2a)
Server name *
projectsqlserveraravind
Database name *
projectsqldb
Rela
1
1
1
2
Authentication type *
SQL authentication
User name *
aravindswamy
Password
Azure Key Vault
AKV linked service *
AzureKeyVault1project_ls
Secret name *
sqlserverpassword
Edit
Secret version
Latest version
Edit
Always encrypted
Encrypt
Mandatory
Connection successful
Create
Back
Test connection
Cancel



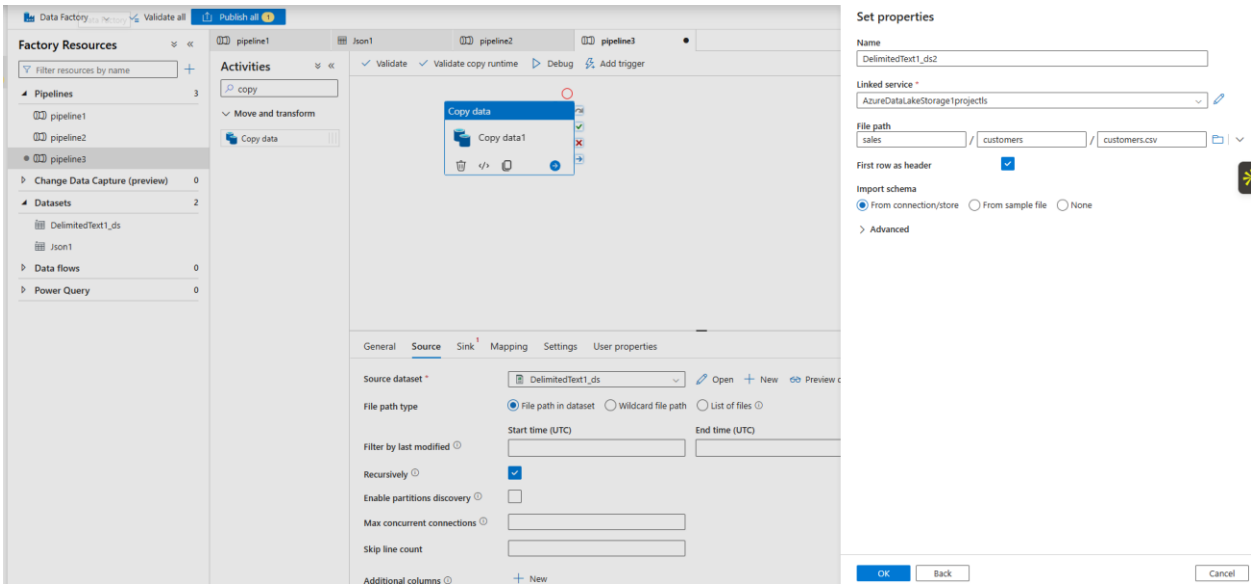
Sourced customer data:



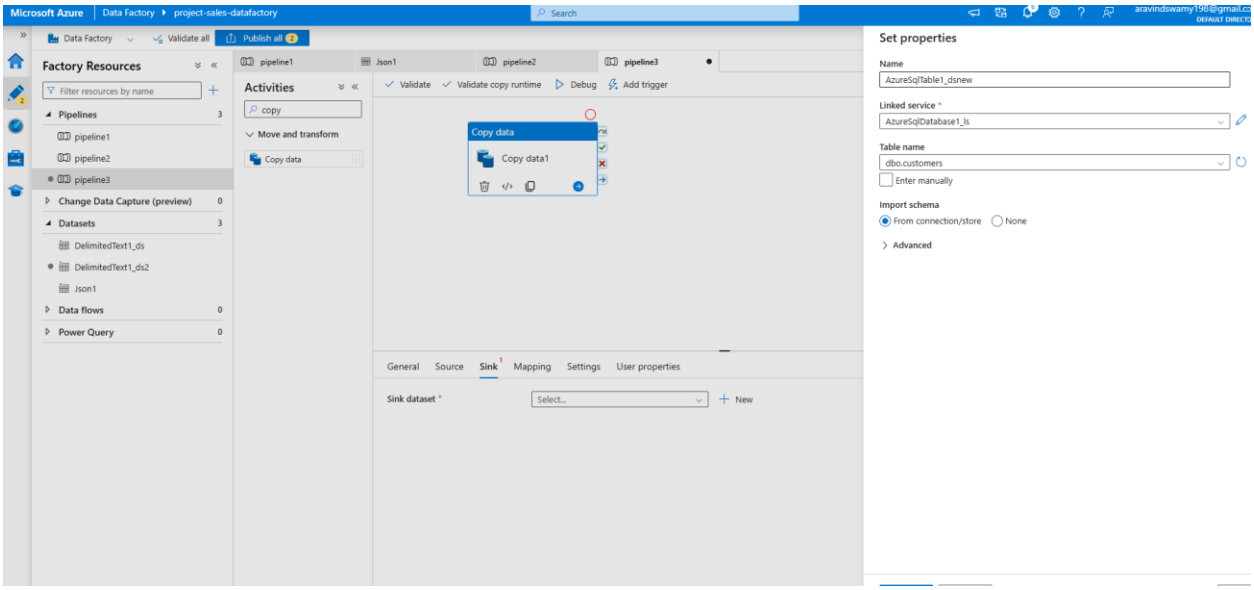
Created customer table structure



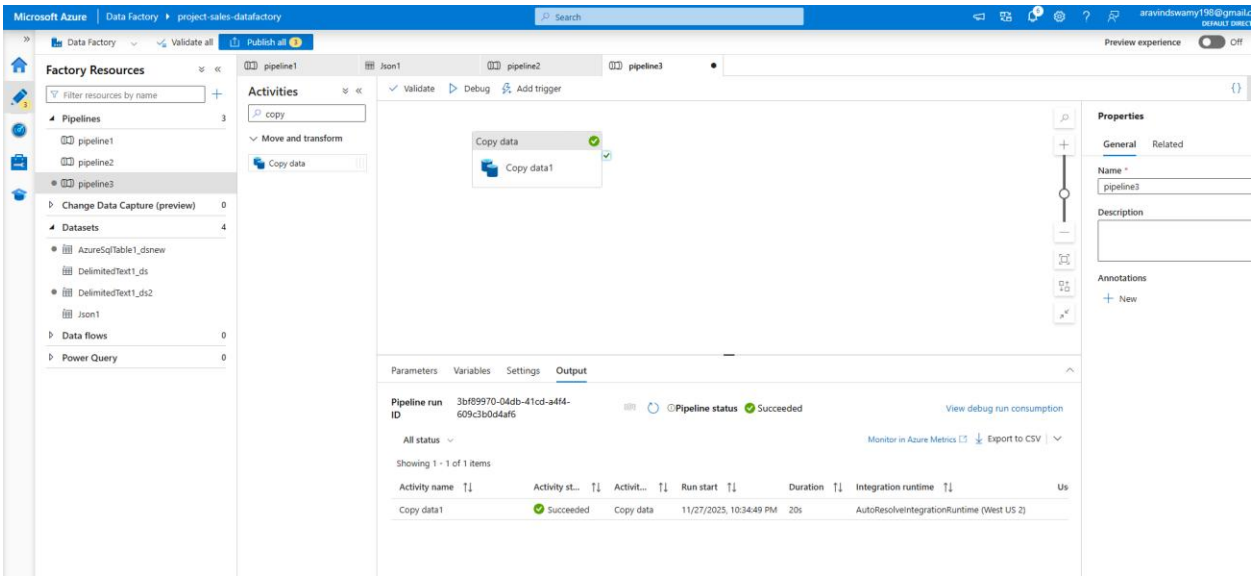
Creating another copy data activity for customers data (background activity)

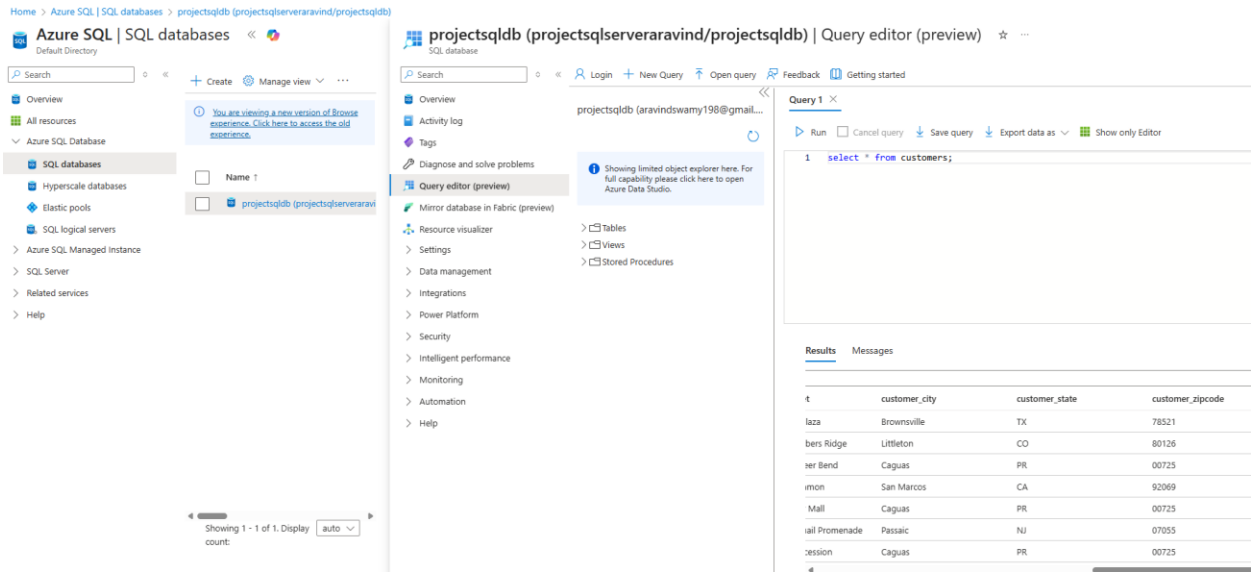


Now sink,



Running,



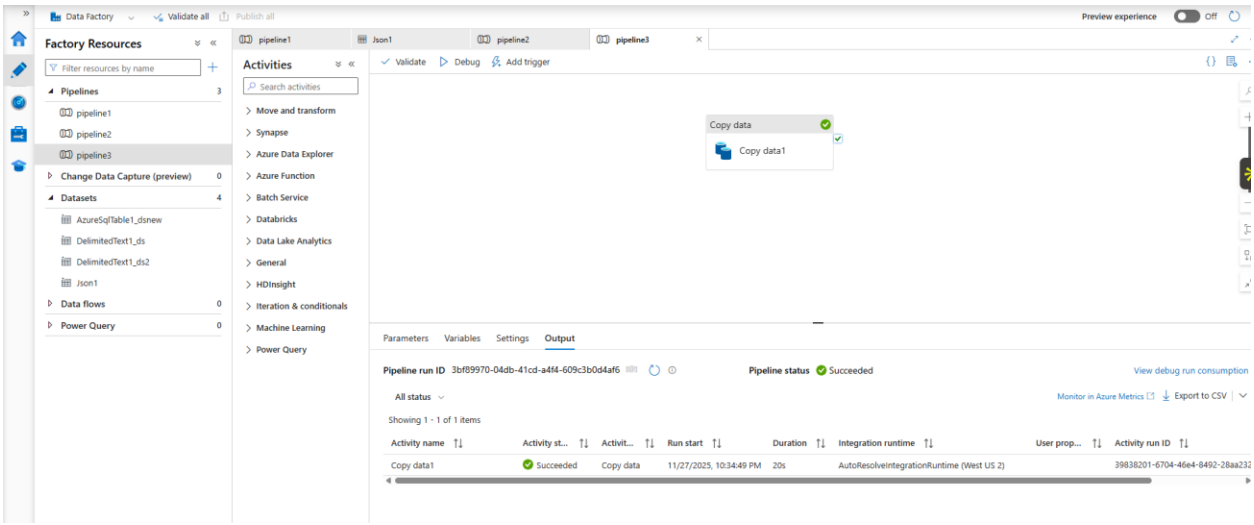


Data is populated in the azure sql db

So now we have 3 datasets in picture one is populated in azure db through background activity and Now as soon as orders file arrive in the landing folder, of ADLS Gen 2, we need to also bring the order_items file from s3 to adls gen2, the 3 files are orders, order_items and customers.

Now we are arriving at the final goal,

We have customers data in azure SQL DB (pipeline3 already run)

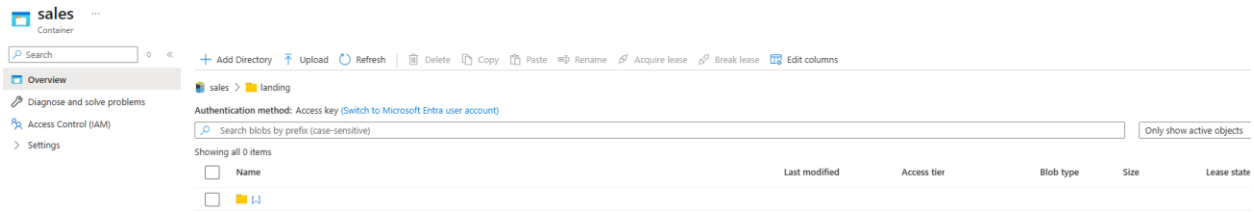


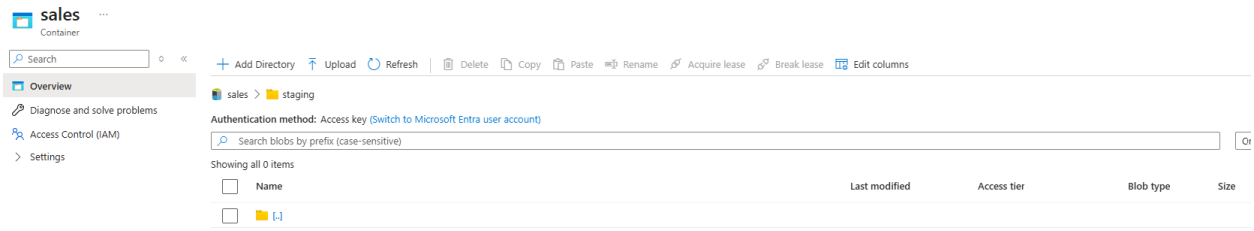
When we put the orders data in the landing folder, the other main pipeline should bring the order items from s3 to adls gen 2 and then run our notebook to generate our final desired table to which the reporting team can connect

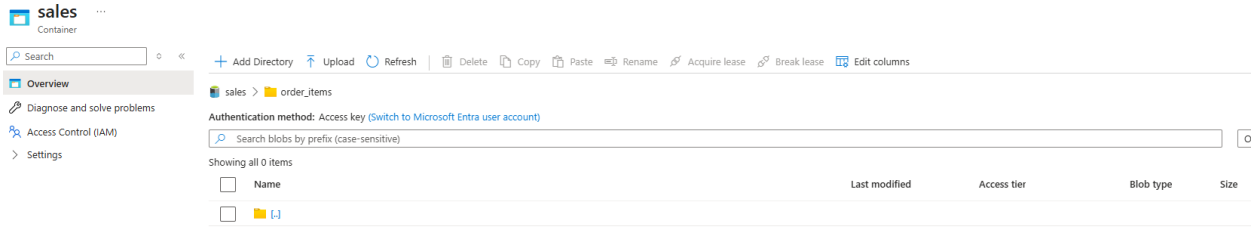
Our main goal, **how many orders are placed by each customer and how much amount is spent by each**

This final data will be stored in our Azure SQL DB (Our Gold Layer)

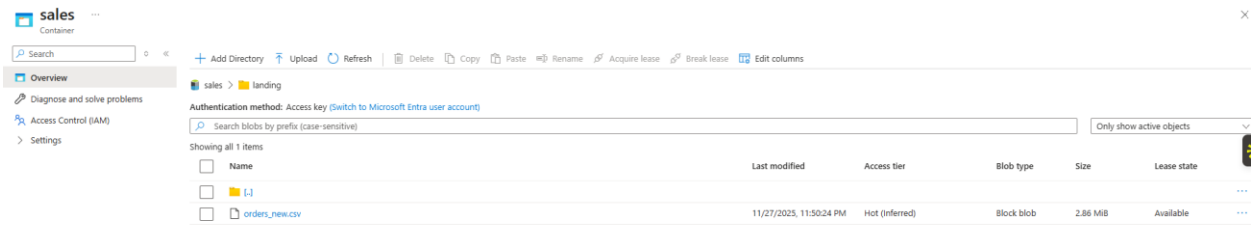
So we are going to finally see the entire pipeline in action, both staging and order items are empty so is landing



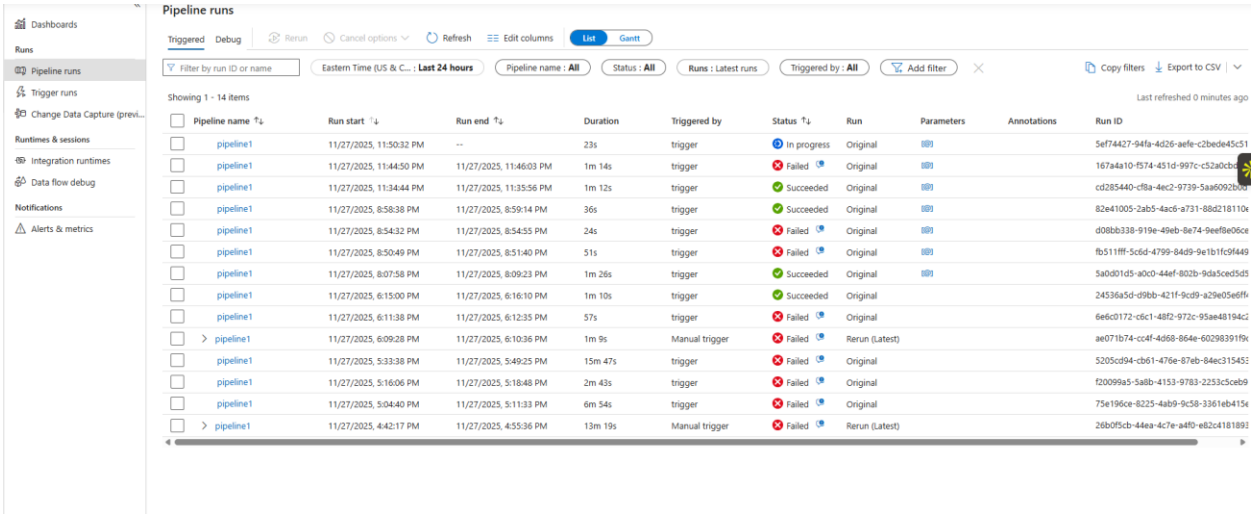




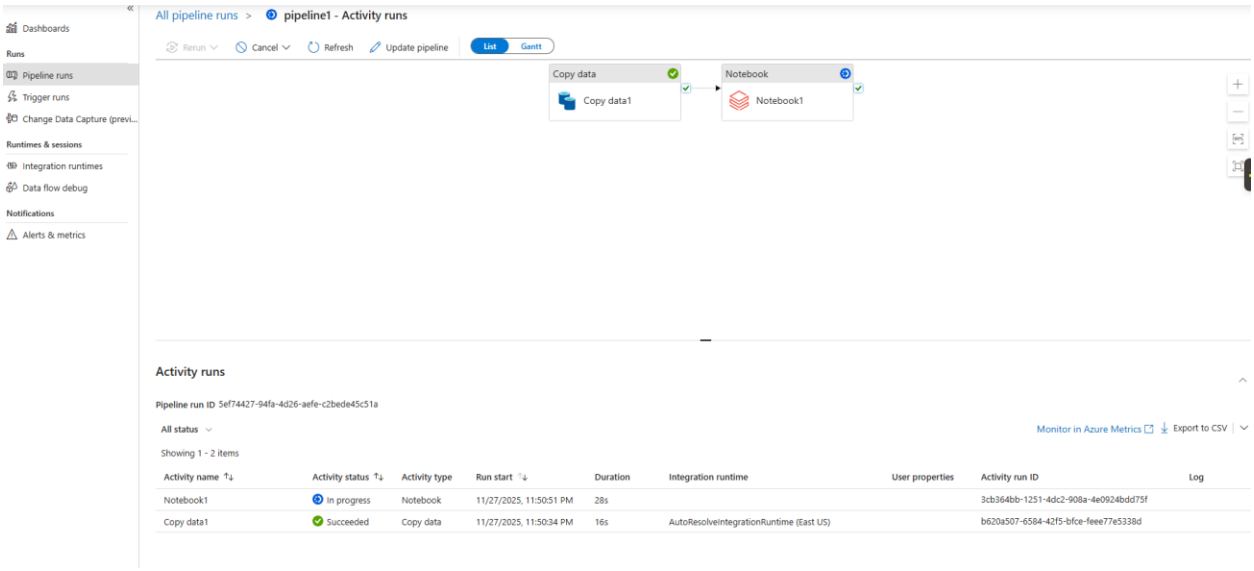
Now uploading the data orders_new in the landing



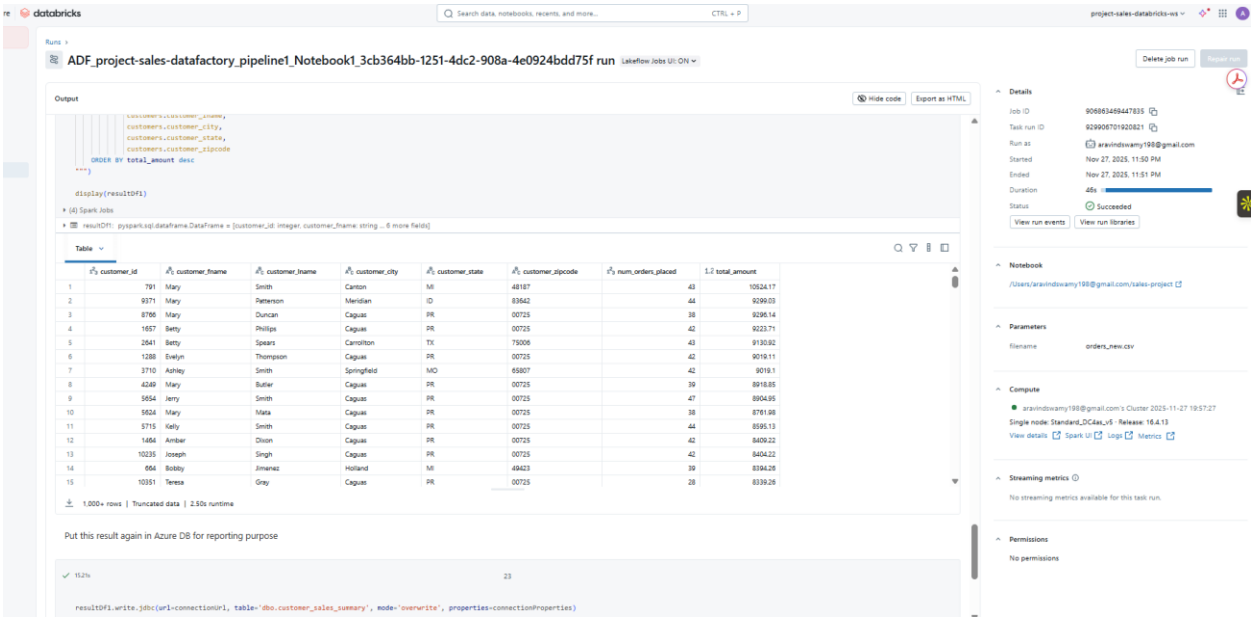
Triggered



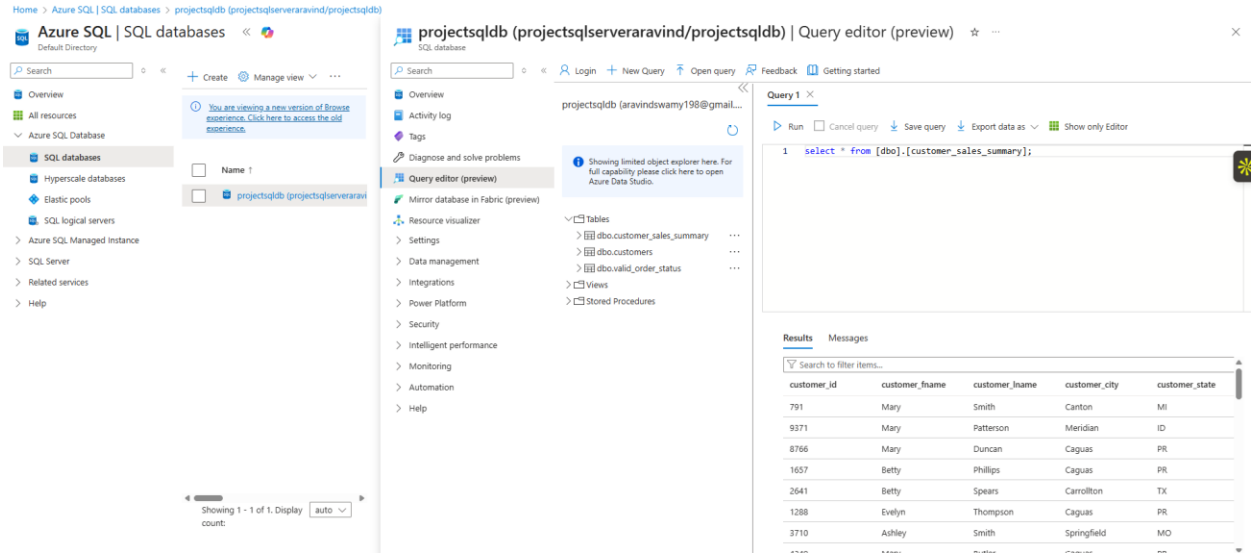
Running,



Succeeded



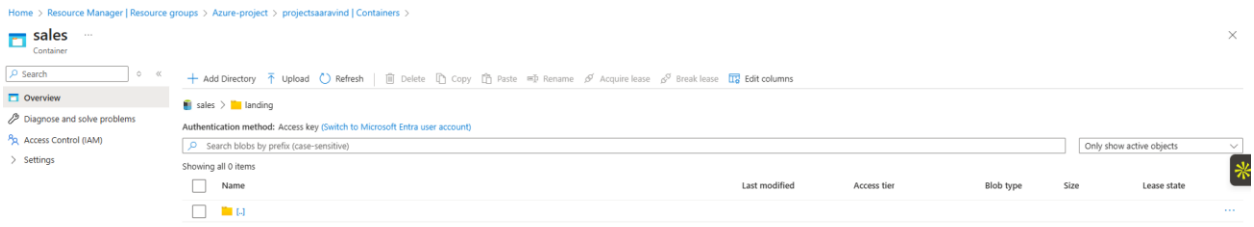
Now checking azure DB (This will be our gold serving layer)



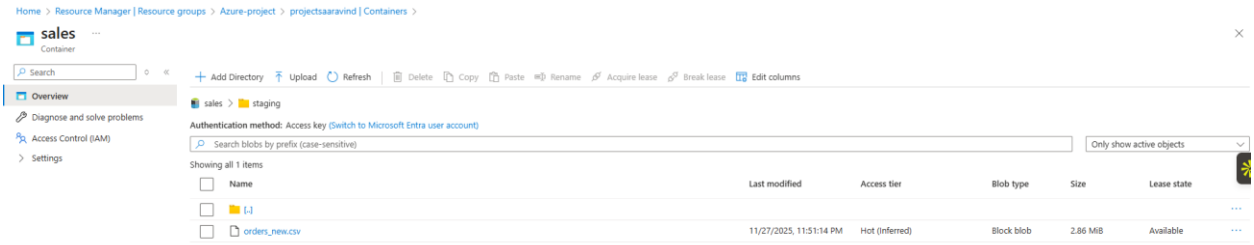
Which the reporting team can use to create their report.

Now we can also check the staging and order items folder to see if the data is transferred,

Now landing is empty,



Staging has orders data now



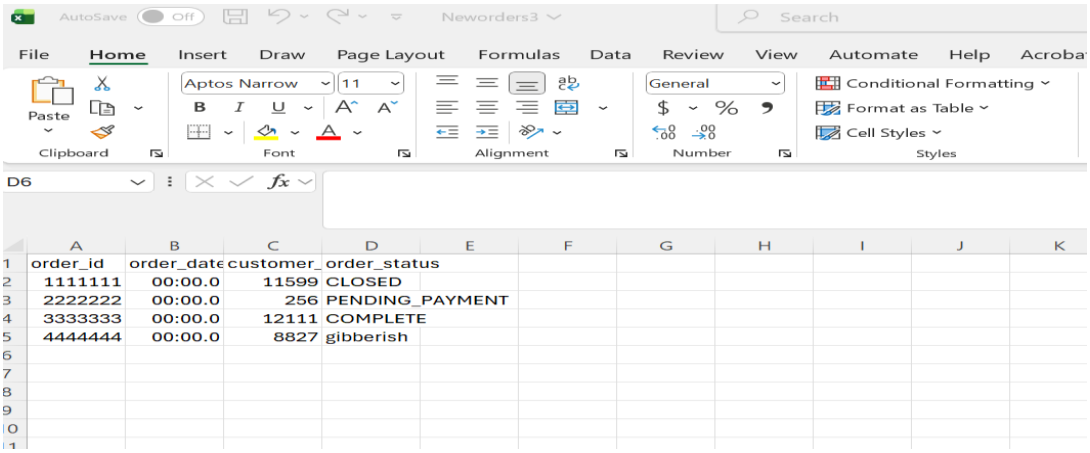
And finally

Order items is also transferred from s3 to adls

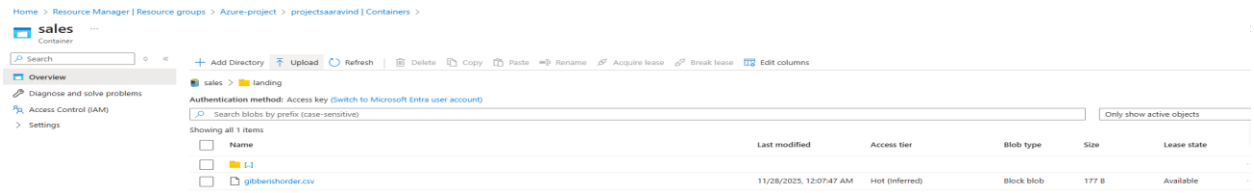
Thus our pipeline executed perfectly.

Now as a final check lets check what if our data is bad , I am gonna create a dummy order status which wont be in the lookup hence will fail because it's a bad data

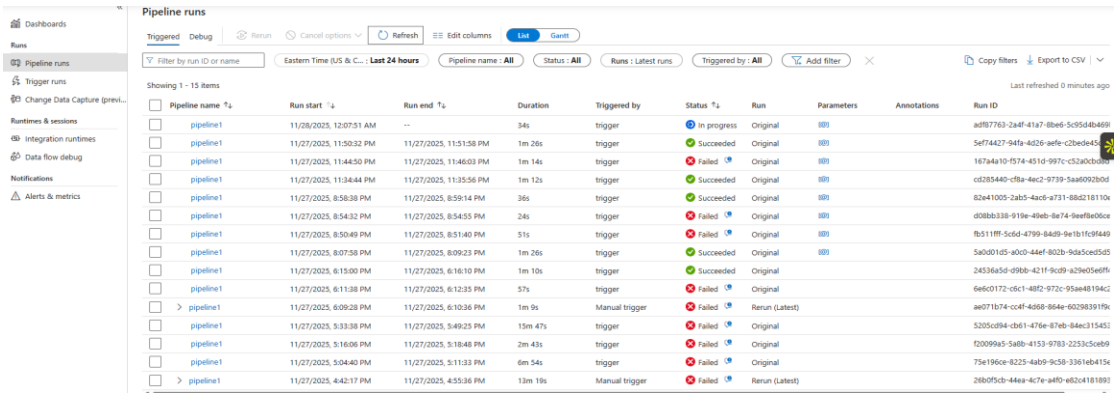
Now creating a file with gibberish order status



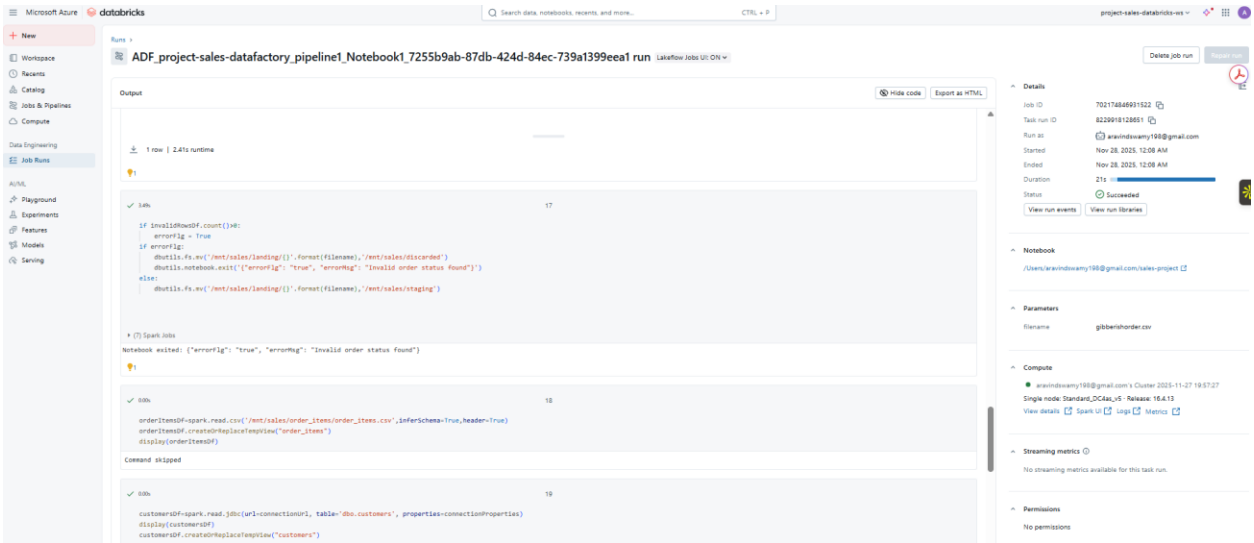
Now similarly loading it in the landing page it should not go to the staging but should go to discarded



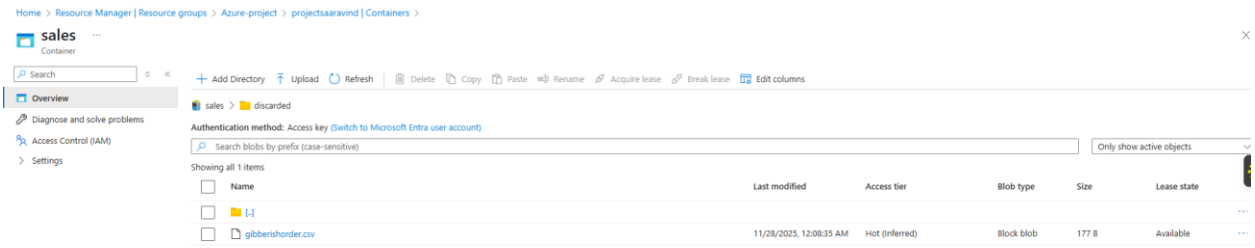
Pipeline triggered



Pipeline stops here since its detected bad data



Data found in discarded now



Thus both pipelines are working perfectly

Conclusion

In this project, an end-to-end automated data engineering pipeline was successfully designed and implemented using Microsoft Azure services to simulate a real-world e-commerce data processing system. The solution demonstrated how cloud-native tools can be integrated to build scalable, reliable, and production-ready data pipelines.

The pipeline was built using an event-driven architecture where data ingestion is automatically triggered as soon as new files arrive in Azure Data Lake Storage Gen2. Azure Data Factory orchestrated the workflow, while Azure Databricks executed distributed validation and transformation logic using PySpark. Data quality was ensured through duplicate checks and dynamic business rule validation using a lookup table maintained in Azure SQL Database.

The project was further enhanced by introducing parameterized pipelines, generic mount logic, and secure secret management using Azure Key Vault. Cross-cloud integration was also implemented by ingesting order item data from Amazon S3, and customer data from Azure SQL Database, thereby simulating realistic multi-source enterprise data workflows.

Finally, all datasets were joined to generate analytics-ready outputs, which were stored in Azure SQL Database for reporting purposes. This project provided strong hands-on experience in building scalable, secure, and reusable data pipelines, and demonstrated key data engineering concepts such as automation, data validation, cloud integration, and Medallion Architecture. The overall solution closely reflects real-world industry data engineering practices.