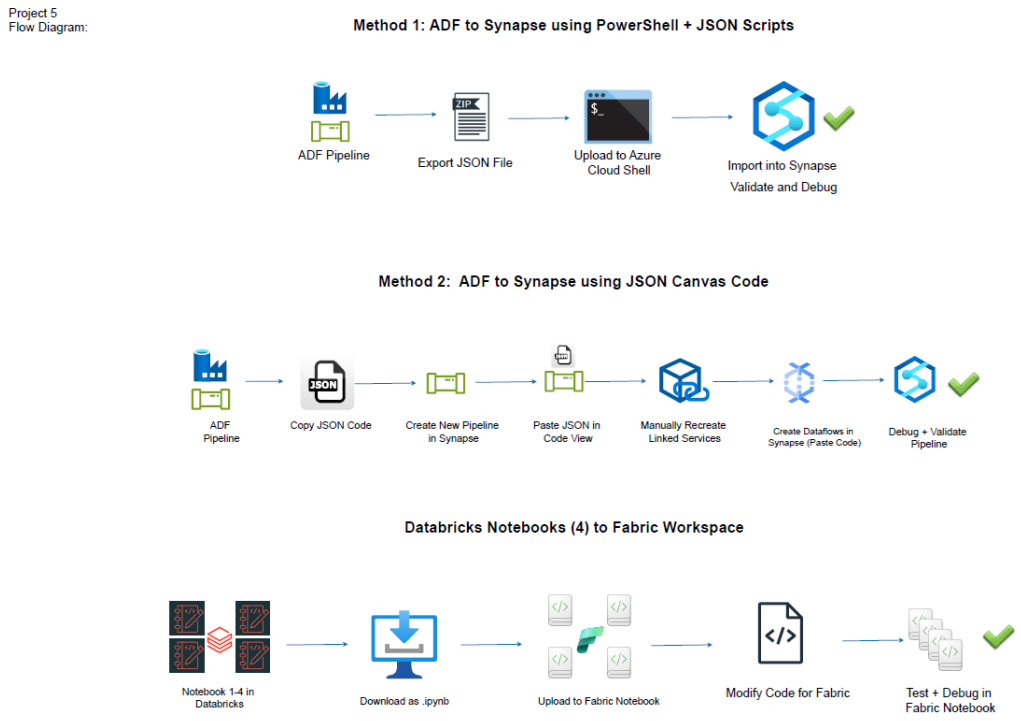
Harpalsinh Vaghela

Data Engineering - Project 5

Migrating pipelines from ADF to Synapse & Databricks Notebook Migration to Fabric Workspace

Table of Contents

Flow Diagram



Method 1: Migration Using Azure PowerShell (ADF to Synapse)

Objective

The goal of this exercise is to migrate an Azure Data Factory (ADF) pipeline, along with its linked services and datasets, into an Azure Synapse Analytics workspace using a manual scripting approach. This ensures a deeper understanding of artifact structure, module usage, and workspace integration.

Tools used: Azure Cloud Shell, Az.Synapse PowerShell module, Synapse Studio

Step 1: Prepare Synapse Environment

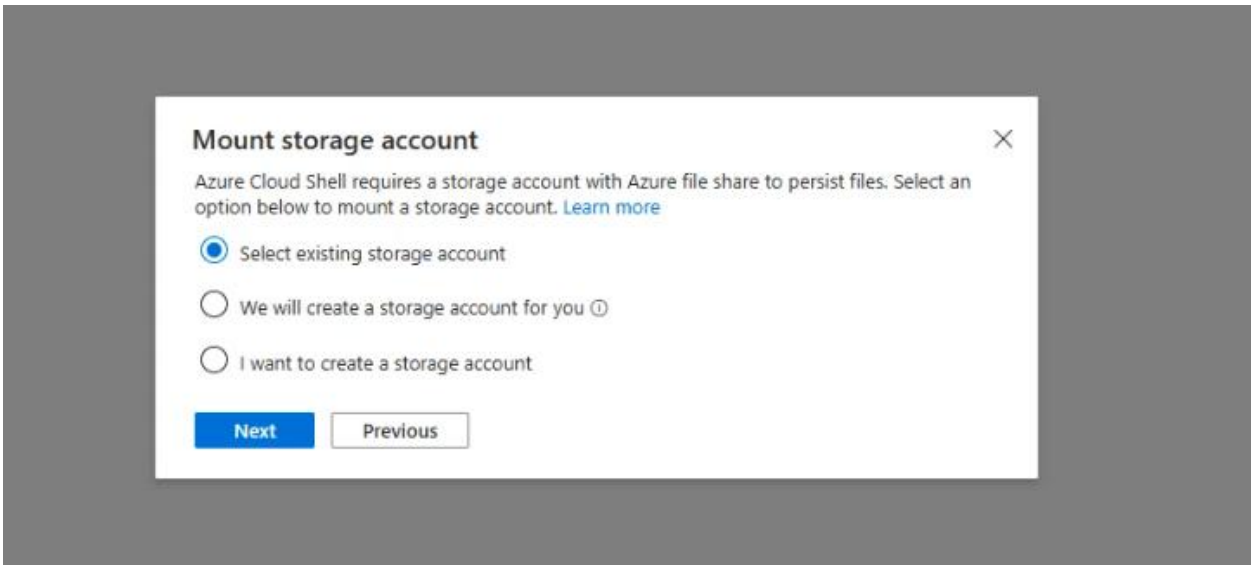
Step 1.1: Launch Cloud Shell

Go to the Azure Portal

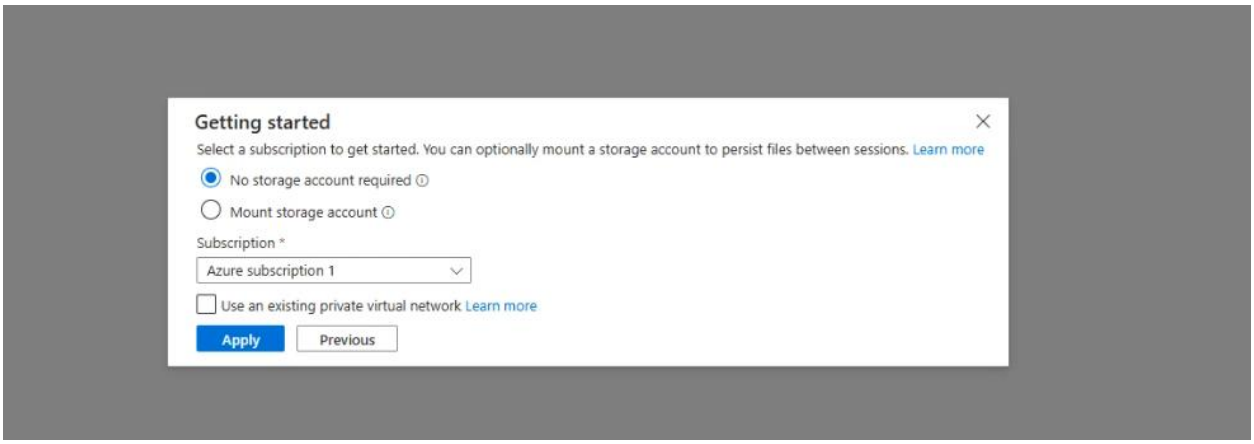
Click on the Cloud Shell icon on the top right

Choose PowerShell mode

Step 1.2: Mount Storage (if prompted)

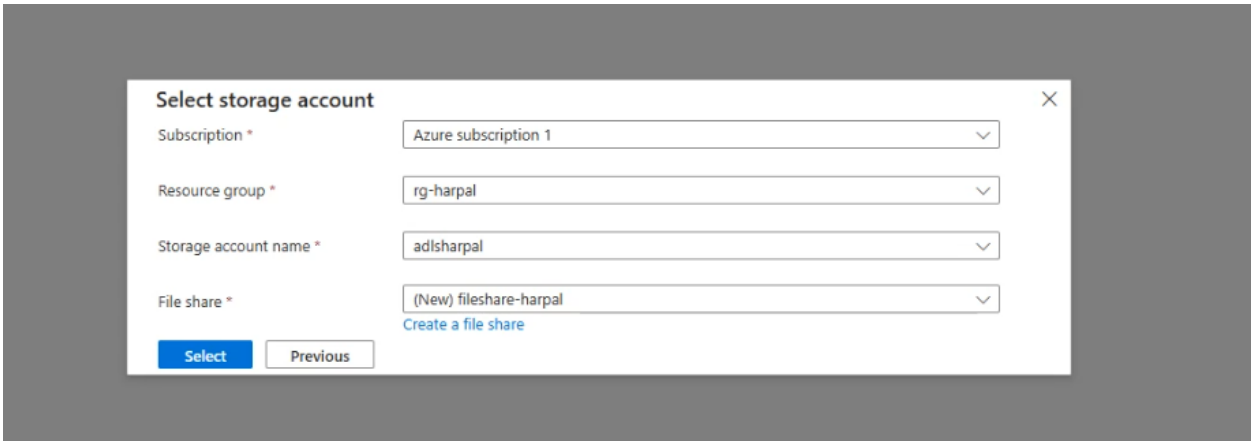


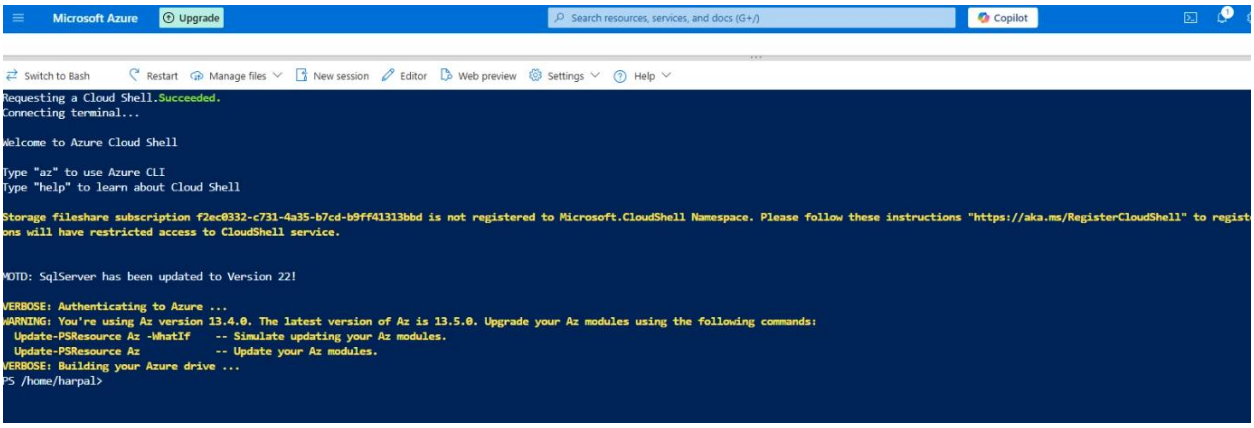
Select your Azure subscription



Create a new storage account or use an existing (e.g., adlsharpal)

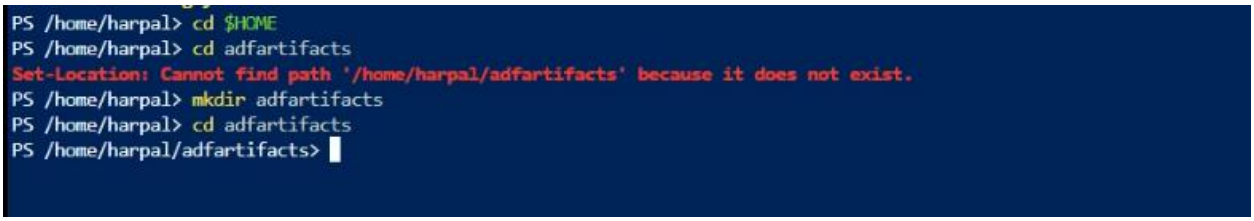
Create a file share (e.g., fileshare-harpal)





Step 1.3: Set Up Folder Structure in Cloud Shell

cd $HOME
mkdir adfartifacts
cd adfartifacts



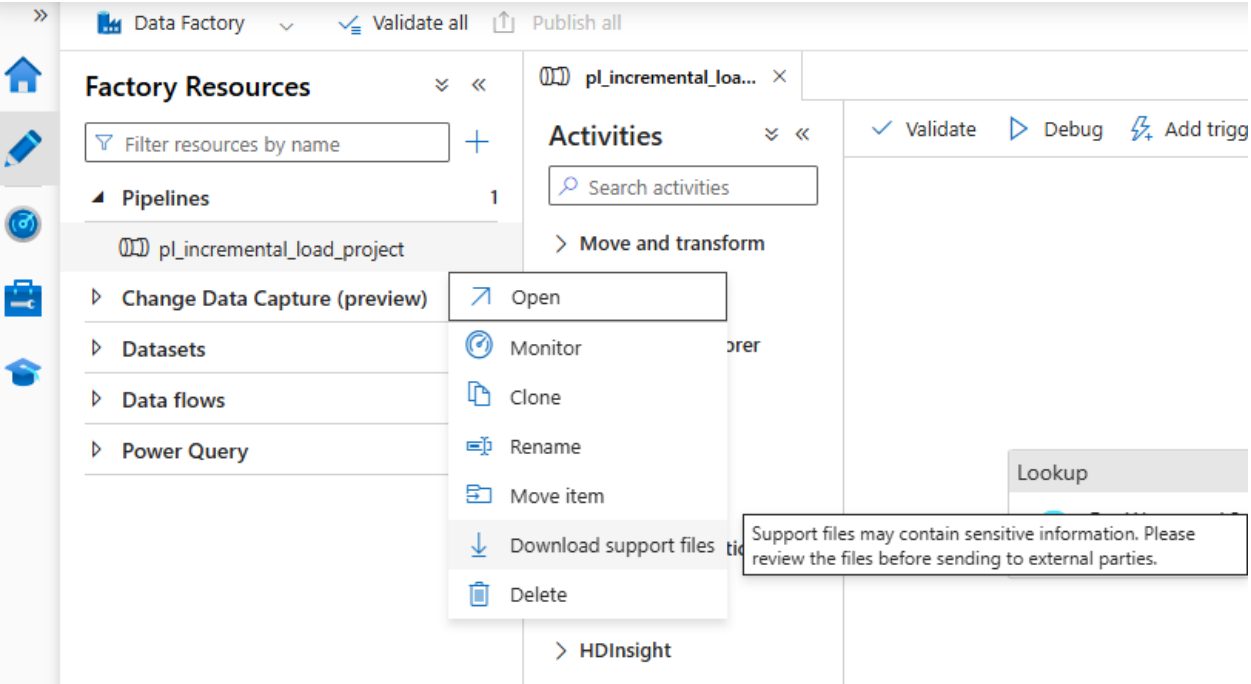
Step 2: Export ADF Pipeline

Step 2.1: Open ADF Studio

Go to your ADF resource > Author & Monitor

Locate pipeline pl_incremental_load_project

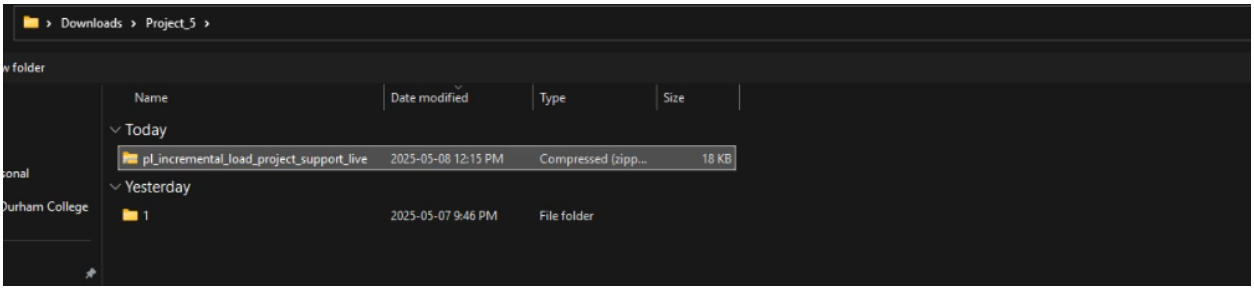
Right-click > Download support files

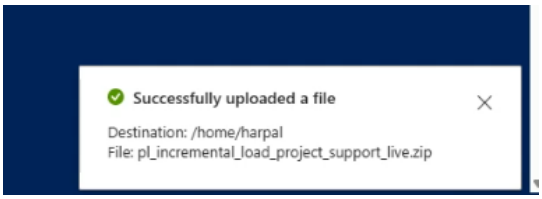


Step 2.2: Upload ZIP to Cloud Shell

Use Cloud Shell → Upload option

Upload the .zip file (e.g., pl_incremental_load_project_support_live.zip)





Moved that Zip file into newly created directory (adfartifacts)

```
PS /home/harpal/adfartifacts> cd ~
PS /home/harpal> ls
adfartifacts  clouddrive  Microsoft  pl_incremental_load_project_support_live.zip
PS /home/harpal> mv ./pl_incremental_load_project_support_live.zip ./adfartifacts/
PS /home/harpal> cd ./adfartifacts/
PS /home/harpal/adfartifacts> ls
pl_incremental_load_project_support_live.zip
PS /home/harpal/adfartifacts>
```

Step 2.3: Extract the ZIP

Expand-Archive "pl_incremental_load_project_support_live.zip" -DestinationPath "." -Force

ls

```
PS /home/harpal/adfartifacts> Expand-Archive "./pl_incremental_load_project_support_live.zip" -DestinationPath "." -Force
PS /home/harpal/adfartifacts> ls
dataset  diagnostic.json  info.txt  linkedService  pipeline  pl_incremental_load_project_support_live.zip
```

Step 3: Set PowerShell Environment

Step 3.1: Connect and Set Subscription

Connect-AzAccount
$context = Select-AzSubscription -SubscriptionName "Azure subscription 1"

```
PS /home/harpal/adfartifacts> $context = Get-AzContext
PS /home/harpal/adfartifacts> $context

   Tenant: b9b0d8ab-2a0a-4453-a4f5-34f2ed2382fd

SubscriptionName     SubscriptionId                       Account   Environment
----------------     --------------                       -------   -----------
Azure subscription 1 f2ec0332-c731-4a35-b7cd-b9ff41313bbd MSI@50342 AzureCloud
```

Step 3.2: Import Az.Synapse Module

Install-Module Az.Synapse -Force

```
PS /home/harpal/adfartifacts>
PS /home/harpal/adfartifacts>
PS /home/harpal/adfartifacts> Install-Module -Name Az.Synapse -Force -AllowClobber
```

```
PS /home/harpal/adfartifacts>
PS /home/harpal/adfartifacts>
Installing package 'Az.Synapse' [Installing dependent package 'Az.Accounts'
```

Import-Module Az.Synapse

```
PS /home/harpal/adfartifacts> Import-Module Az.Synapse
PS /home/harpal/adfartifacts> $workspace = "wsp-synapse-harpal"
```

Step 4: Import ADF Artifacts into Synapse

Set workspace name:

$workspace = "wsp-synapse-harpal"


```
PS /home/harpal/adfartifacts> Import-Module Az.Synapse
PS /home/harpal/adfartifacts> $workspace = "wsp-synapse-harpal"
```


Step 4.1: Import Linked Services

Set-AzSynapseLinkedService -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile "./linkedService/azuresqllookup.json" -DefaultProfile $context

Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_csvfiles -DefinitionFile "./linkedService/ls_csvfiles.json" -DefaultProfile $context

Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_key_vault -DefinitionFile "./linkedService/ls_key_vault.json" -DefaultProfile $context

Step 4.2: Import Datasets

Set-AzSynapseDataset -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile "./dataset/azuresqllookup.json" -DefaultProfile $context

Set-AzSynapseDataset -WorkspaceName $workspace -Name ds_csvfiles -DefinitionFile "./dataset/ds_csvfiles.json" -DefaultProfile $context

```
PS /home/harpal/adfartifacts> Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_csvfiles -DefinitionFile ".\linkedService\ls_csvfiles.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties    : Azure.Analytics.Synapse.Artifacts.Models.AzureBlobFSLinkedService
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/linkedservices/ls_csvfiles
Name          : ls_csvfiles
Type          : Microsoft.Synapse/workspaces/linkedservices
Etag          : 21027d0b-0000-0a00-0000-681cdde90000

PS /home/harpal/adfartifacts> Set-AzSynapseLinkedService -WorkspaceName $workspace -Name ls_key_vault -DefinitionFile ".\linkedService\ls_key_vault.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties    : Azure.Analytics.Synapse.Artifacts.Models.AzureKeyVaultLinkedService
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/linkedservices/ls_key_vault
Name          : ls_key_vault
Type          : Microsoft.Synapse/workspaces/linkedservices
Etag          : 21028e0c-0000-0a00-0000-681cddfb0000

PS /home/harpal/adfartifacts> Set-AzSynapseLinkedService -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile ".\linkedService\azuresqllookup.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties    : Azure.Analytics.Synapse.Artifacts.Models.AzureSqlDatabaseLinkedService
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/linkedservices/azuresqllookup
Name          : azuresqllookup
Type          : Microsoft.Synapse/workspaces/linkedservices
Etag          : 2102870d-0000-0a00-0000-681cde0c0000

PS /home/harpal/adfartifacts> Set-AzSynapseDataset -WorkspaceName $workspace -Name azuresqllookup -DefinitionFile ".\dataset\azuresqllookup.json" -DefaultProfile $context

WorkspaceName : wsp-synapse-harpal
Properties    : Azure.Analytics.Synapse.Artifacts.Models.AzureSqlTableDataset
Id            : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/datasets/azuresqllookup
Name          : azuresqllookup
Type          : Microsoft.Synapse/workspaces/datasets
Etag          : 2102e10e-0000-0a00-0000-681cde1b0000

PS /home/harpal/adfartifacts> Set-AzSynapseDataset -WorkspaceName $workspace -Name ds_csvfiles -DefinitionFile ".\dataset\ds_csvfiles.json" -DefaultProfile $context
```

Step 4.3: Import Pipeline

Set-AzSynapsePipeline -WorkspaceName $workspace -Name "pl_incremental_load_project" -DefinitionFile "./pipeline/pl_incremental_load_project.json" -DefaultProfile $context

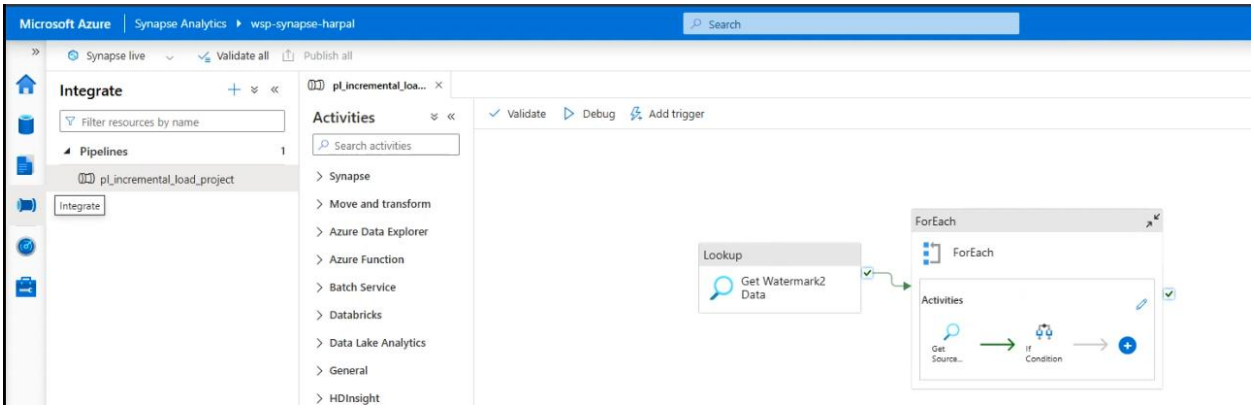
```
PS /home/harpal/adfartifacts> Set-AzSynapsePipeline -WorkspaceName $workspace -Name "pl_incremental_load_project" -DefinitionFile ".\pipeline\pl_incremental_load_project.json" -DefaultProfile $context

WorkspaceName        : wsp-synapse-harpal
Description          :
Activities           : {Get Watermark2 Data, ForEach}
Variables            : {}
Concurrency          :
Annotations          : {}
RunDimensions        : {}
Folder               : Microsoft.Azure.Commands.Synapse.Models.PSPipelineFolder
Parameters           : {}
AdditionalProperties : {}
Id                   : /subscriptions/f2ec0332-c731-4a35-b7cd-b9ff41313bbd/resourceGroups/rg-harpal/providers/Microsoft.Synapse/workspaces/wsp-synapse-harpal/pipelines/pl_incremental_load_project
Name                 : pl_incremental_load_project
Type                 : Microsoft.Synapse/workspaces/pipelines
Etag                 : 2102ef11-0000-0a00-0000-681cde370000
```

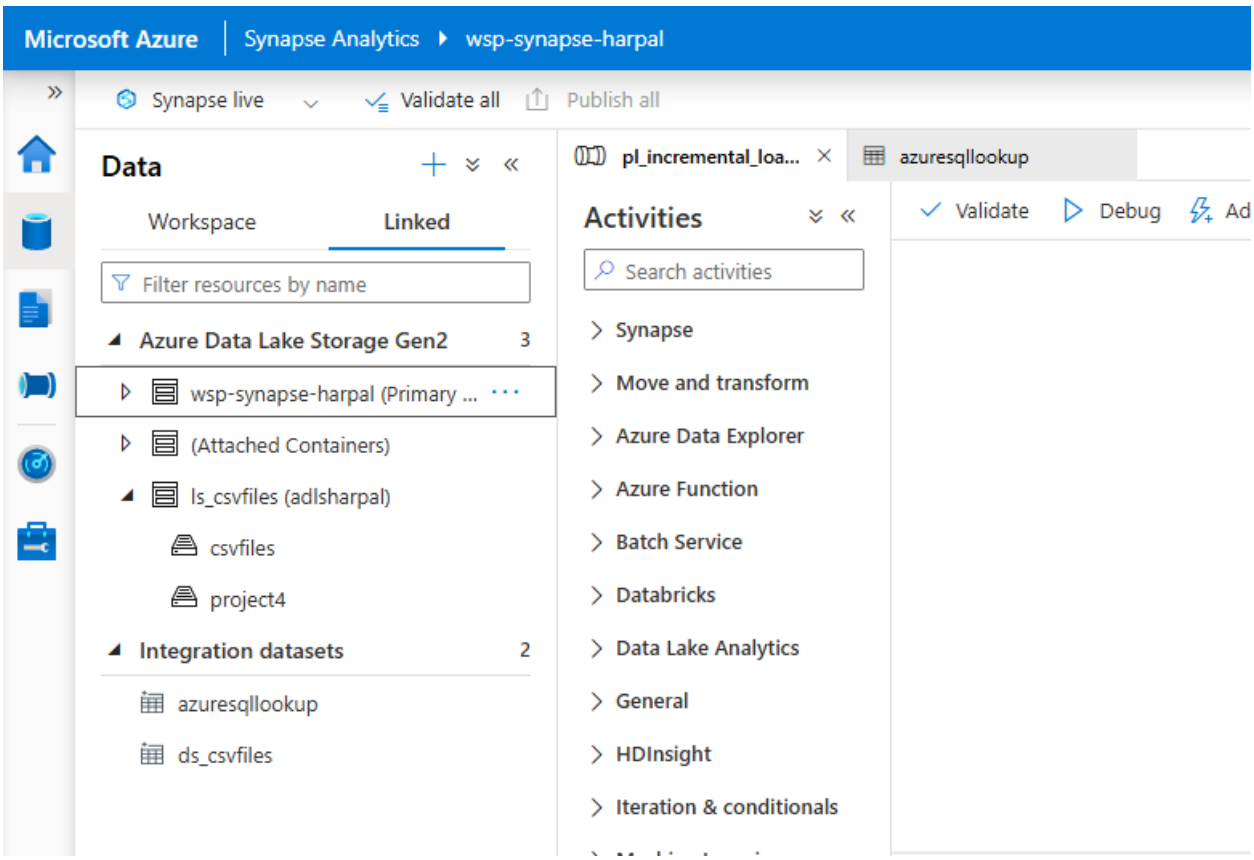
Step 5: Validate and Run Pipeline in Synapse

1. Open Synapse Studio
2. Go to Integrate > Verify pipeline pl_incremental_load_project exists
3. Click the pipeline > Click Validate (top bar)
4. Click Debug to test run the pipeline

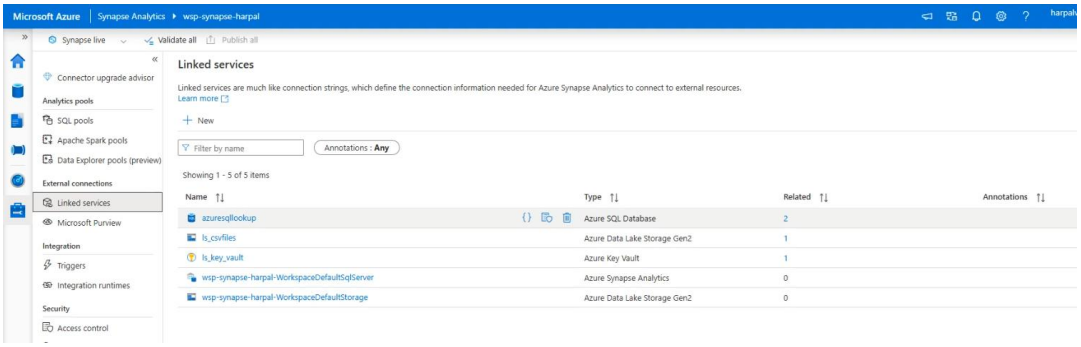
Integrate Tab:



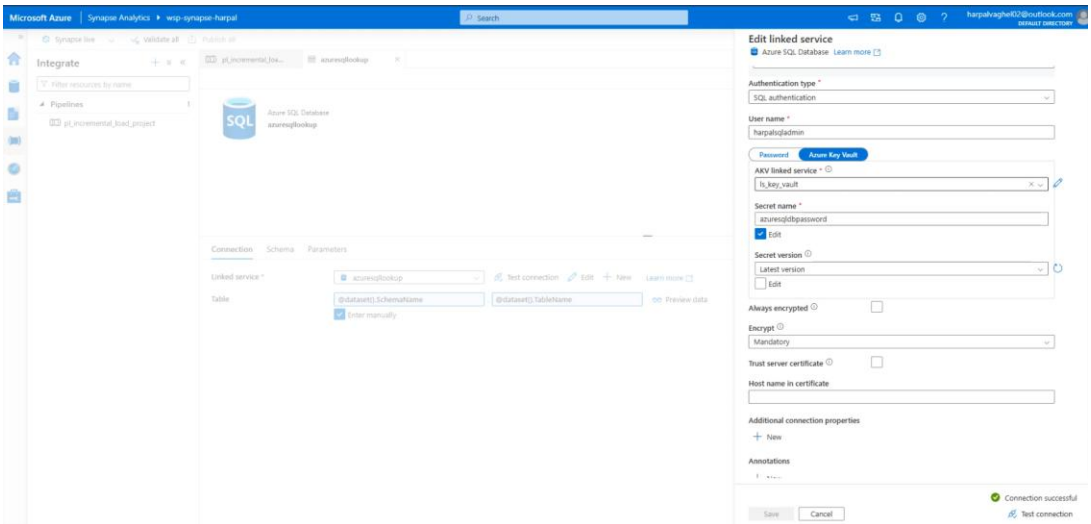
Check the Dataset in the Data Tab:



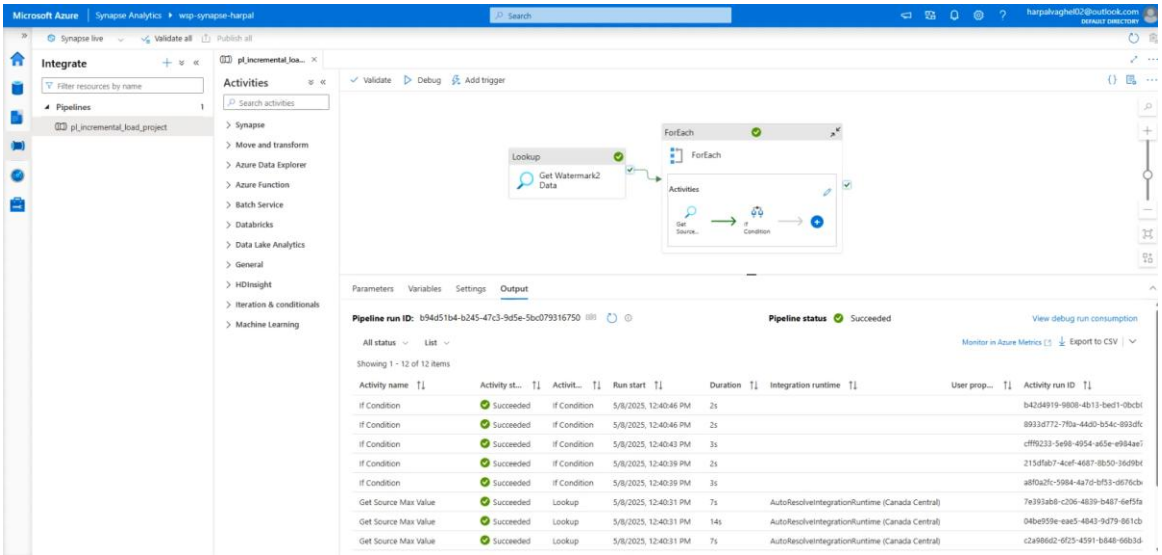
Manage Tab:



Key Vault Connection Checking in Linked Service in Synapse Workspace:



Debug the pipeline:



What to Do With /home/harpal/adfartifacts after the migration?

We can safely delete the contents after migration is 100% successful, because:

The artifacts (pipeline, datasets, linked services) have already been imported into Azure Synapse.

Synapse now stores its own copy, and Cloud Shell is no longer needed for them.

Run this in Cloud Shell to delete the folder:

rm -r ~/adfartifacts

Or remove the Zip file:

rm ~/pl_incremental_load_project_support_live.zip

```
PS /home/harpal/adfartifacts> rm -r ~/adfartifacts
PS /home/harpal/adfartifacts> ls
clouddrive  Microsoft
PS /home/harpal/adfartifacts> cd..
PS /home/harpal> ls
clouddrive  Microsoft
PS /home/harpal>
```

Points to remember:

- Make sure you are in the correct working directory when running import commands
- Ensure file names and paths match exactly
- Reimporting with an incorrect workspace name won't show errors, but nothing will appear
- Validate linked services post-import (especially Key Vault and Integration Runtime)

Conclusion:

This manual migration approach helps build a strong understanding of how ADF and Synapse Pipelines share the same underlying schema. Using scripting provides flexibility and is ideal for troubleshooting, learning, or precise CI/CD setups. The project was successfully executed from end-to-end with the pipeline running in the Synapse environment.

Method 2: Migration Using JSON Code (ADF to Synapse)

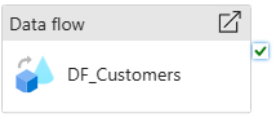
Objective

To migrate an existing pipeline and dataflow from Azure Data Factory (ADF) to Azure Synapse Analytics manually using the JSON definition of pipeline and dataflow artifacts.

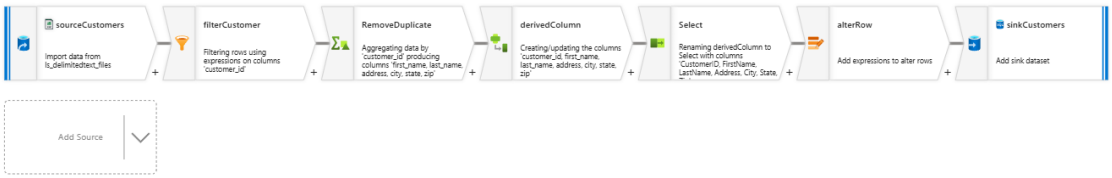
Step 1: Validate Pipeline in Azure Data Factory

We have this pipeline in ADF that cleans the customer's data from the ADLS Gen 2 storage account and loads it into the Azure SQL Database Table.

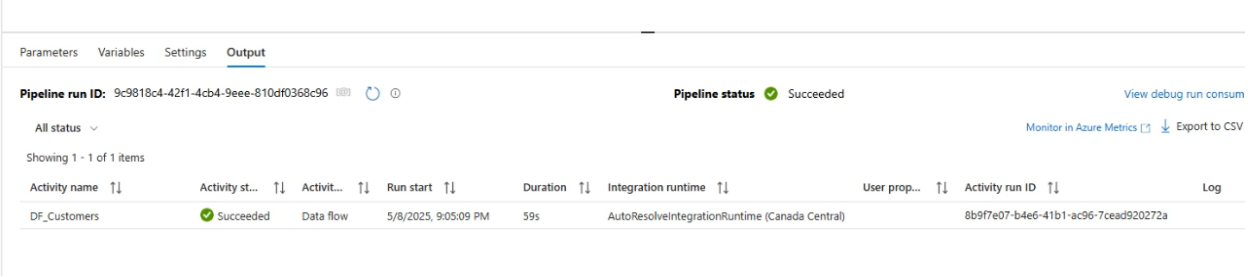
Open ADF workspace:



Dataflow Design:



Run the pipeline in Debug mode to ensure that it is working as expected



Step 2: Export JSON Definitions from ADF

Open the pipeline canvas in ADF
Click on the "Code" tab
Copy the entire JSON content of the pipeline design
Repeat the same process for the associated Data Flow

Dataflow Activity code at the pipeline level:

Pipeline name pl_adls_to_sql

Copy to clipboard

```
    "properties": {
        "activities": [
            {
                "name": "DF_Customers",
                "type": "ExecuteDataFlow",
                "dependsOn": [],
                "policy": {
                    "timeout": "0.12:00:00",
                    "retry": 0,
                    "retryIntervalInSeconds": 30,
                    "secureOutput": false,
                    "secureInput": false
                },
                "userProperties": [],
                "typeProperties": {
                    "dataflow": {
                        "referenceName": "df_customers_data",
                        "type": "DataFlowReference"
                    },
                    "compute": {
                        "coreCount": 8,
                        "computeType": "General"
                    },
                    "traceLevel": "None",
                    "cacheSinks": {
                        "firstRowOnly": true
                    }
                }
            }
        ],
        "annotations": [],
        "lastPublishTime": "2025-05-09T00:52:42Z"
    },
    "type": "Microsoft.DataFactory/factories/pipelines"
}
```

OK Cancel

Dataflow Design JSON Code:

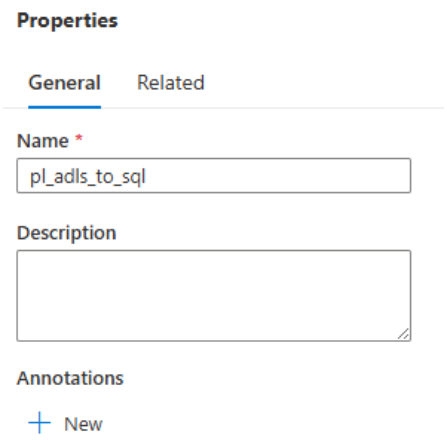
Copy to clipboard

```
{
    "name": "df_customers_data",
    "properties": {
        "type": "MappingDataFlow",
        "typeProperties": {
            "sources": [
                {
                    "linkedService": {
                        "referenceName": "ls_delimitedtext_files",
                        "type": "LinkedServiceReference"
                    },
                    "name": "sourceCustomers"
                }
            ],
            "sinks": [
                {
                    "linkedService": {
                        "referenceName": "ls_azure_sql_db",
                        "type": "LinkedServiceReference"
                    },
                    "name": "sinkCustomers"
                }
            ],
            "transformations": [
                {
                    "name": "filterCustomer"
                },
                {
                    "name": "RemoveDuplicate"
                },
                {
                    "name": "Select"
                },
                {
                    "name": "derivedColumn"
                },
```

Step 3: Open Azure Synapse Workspace

Launch Synapse Studio
Navigate to the Integrate hub
Click + Pipeline, then rename it exactly the same as the original ADF pipeline
Click on the "Code" tab and paste the copied JSON code from ADF

Confirm and save the pipeline.

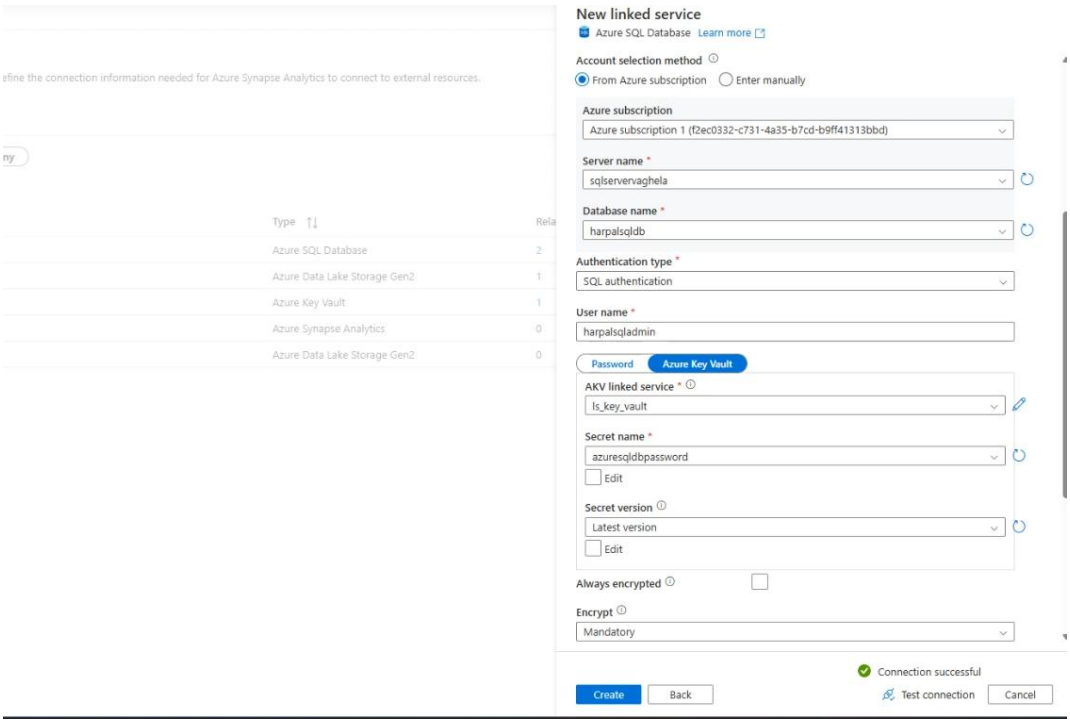


Step 4: Recreate Linked Services in Synapse

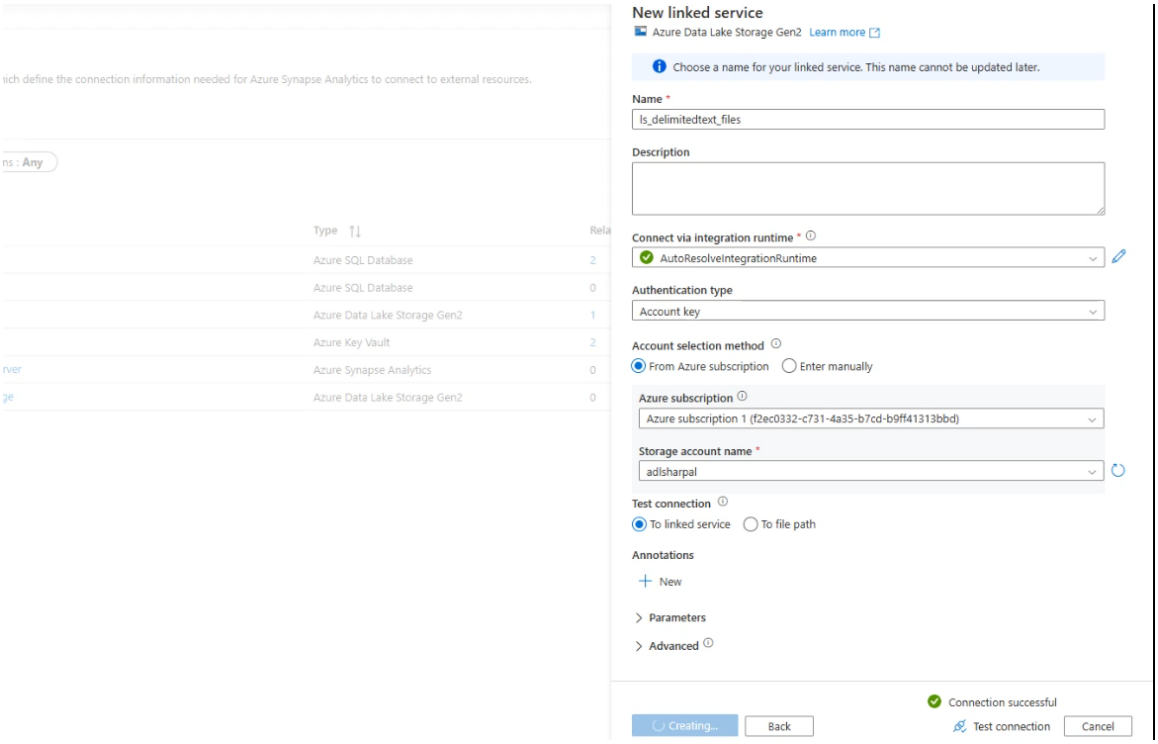
Go to the Manage tab in Synapse Studio
Manually create all Linked Services referenced in the original ADF pipeline
Azure SQL Database linked service
Azure Data Lake Gen2 linked service

Go to Synapse Workspace:

Azure SQL Database Linked Service:

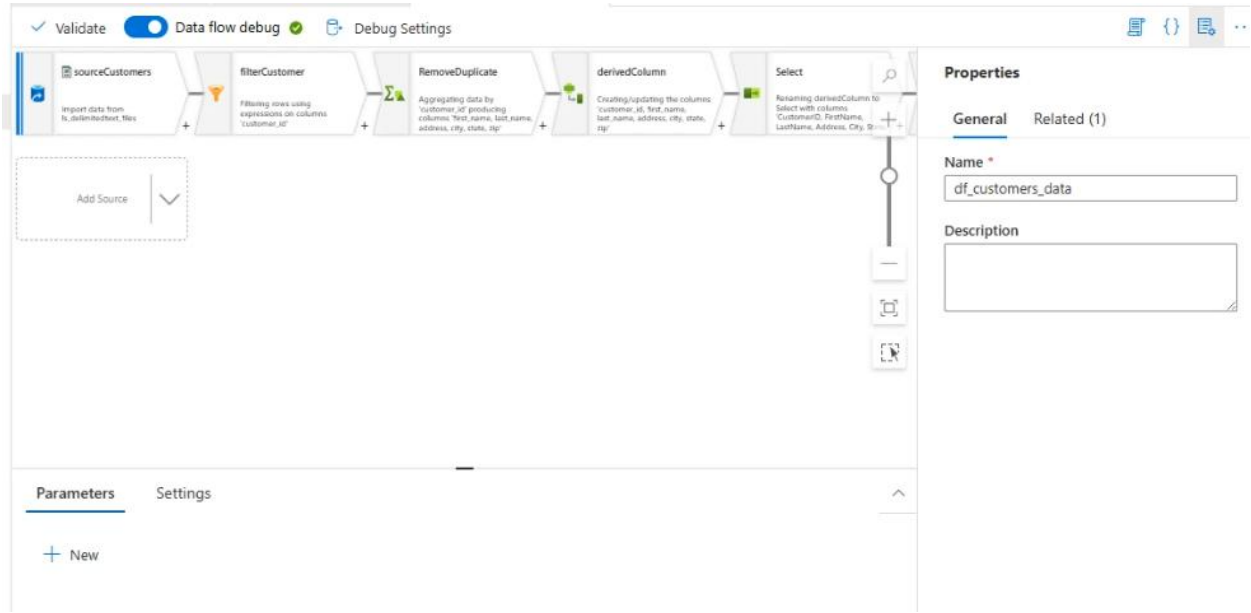


Azure Data Lake Gen 2 Linked Service:



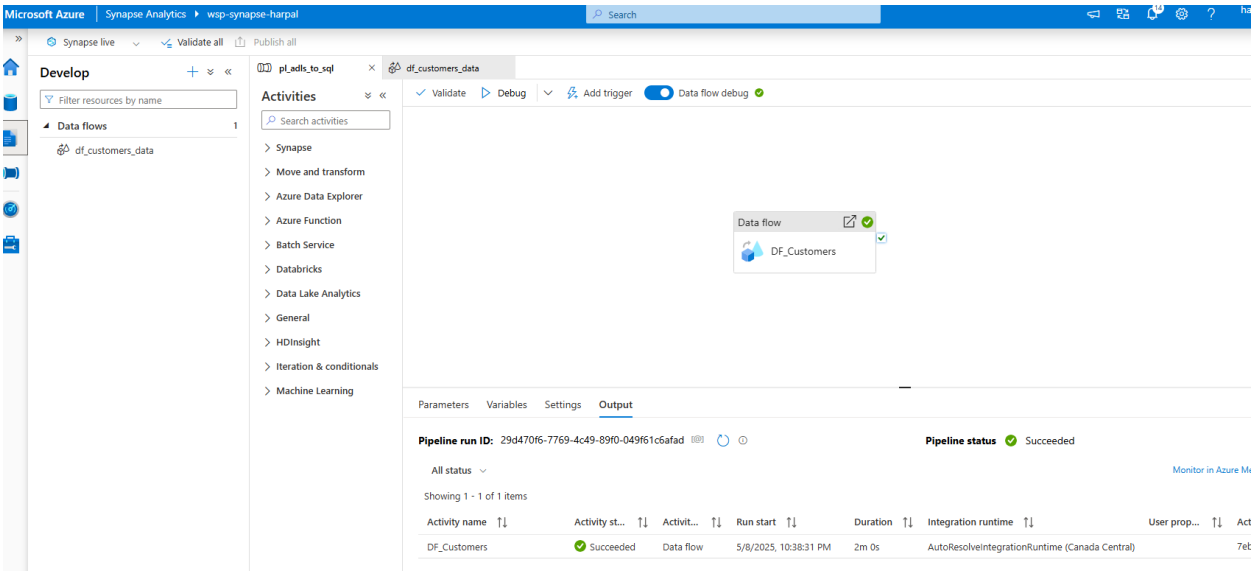
Step 5: Recreate Data Flow

In the Develop hub of Synapse
Create a new Data Flow with the same name as in ADF
Open the "Code" view and paste the JSON from the ADF Data Flow canvas
Save and validate the design



Step 6: Test the Migrated Pipeline in Synapse

Open the new pipeline in Synapse
Click Debug to test the execution
Ensure data loads from ADLS to Azure SQL as expected



Points to Remember:

- Ensure Linked Service names match exactly in Synapse or update references in JSON
- Validate the pipeline and dataflow in ADF before exporting
- Always test the pipeline in Debug mode post-migration
- Synapse supports most ADF components, but confirm compatibility for custom or preview features

Conclusion:

This manual JSON-based migration method successfully replicates an ADF pipeline and dataflow into Synapse Analytics. It avoids full Git-based export and gives granular control to migrate only what is needed.

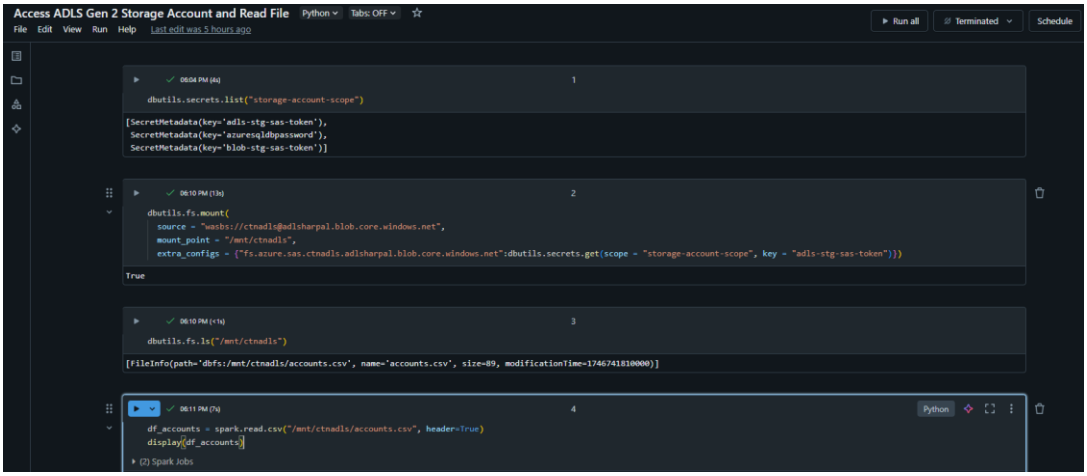
Migrate Databricks Notebook to Fabric Workspace

Objective

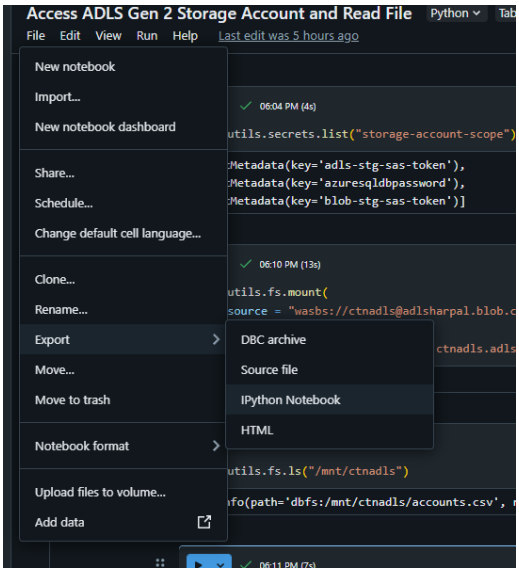
To successfully migrate a set of 4 Databricks notebooks into Microsoft Fabric Workspace by modifying unsupported features such as mount points, key vault access, and storage paths. This includes reading data from Azure Data Lake Gen2, applying transformations, and saving results into Fabric Lakehouse tables.

Notebook 1

Go to Databricks Workspace and open/create a notebook:

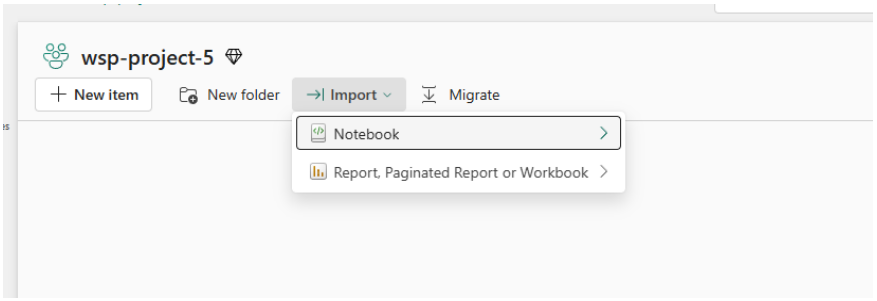


We will download this notebook in ipynb file format and migrate to the Fabric workspace.

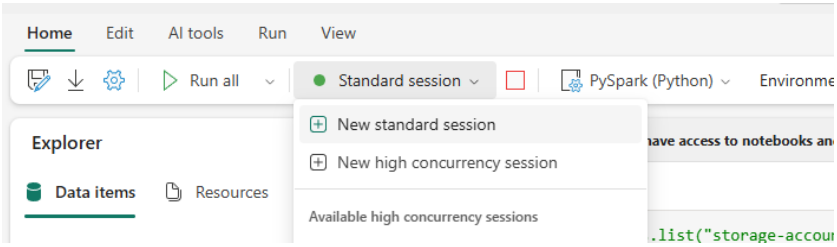


Turn on Fabric Capacity from the Azure Account

Now, open the Fabric Workspace and click on Import Notebook



Start the Spark Session



Make necessary changes for fabric notebook

Comment or remove code which is not needed

Add new code if we have used a mount point in Databricks, we have changed it here in the Fabric Notebook

```
sasToken = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "adls-stg-sas-token")
```

[1] ✓ - Command executed in 5 sec 811 ms by Fabric on 6:15:37 PM, 5/08/25

```
notebookutils.fs.mount(
    "abfss://ctnadls@adlsharpal.dfs.core.windows.net",
    "/test",
    {"sasToken":sasToken}
)
```

[2] ✓ - Command executed in 876 ms by Fabric on 6:16:28 PM, 5/08/25

Table | New chart

Table view

	localPath	mountPoint	scope	source	storageType
1	/synfs/notebo...	/test	job	abfss://ctna...	Data Lake Storag...

Display the csv file data from ADLS Gen 2:

```
containerPath = notebookutils.fs.getMountPath("/test")
```

[7] ✓ - Command executed in 297 ms by Fabric on 6:19:30 PM, 5/08/25

```
notebookutils.fs.ls(f"file://{containerPath}")
```

[6] ✓ - Command executed in 304 ms by Fabric on 6:18:39 PM, 5/08/25

```
[FileInfo(path=file:/synfs/notebook/b5ba821f-52a8-4c15-aa80-34bf435152b4/test/accounts.csv, name=accounts.csv, size=89)]
```

```
df_accounts = spark.read.option("header", True).csv(f"file://{containerPath}/accounts.csv")
display(df_accounts)
```

[14] ✓ - Command executed in 1 sec 624 ms by Fabric on 6:25:35 PM, 5/08/25

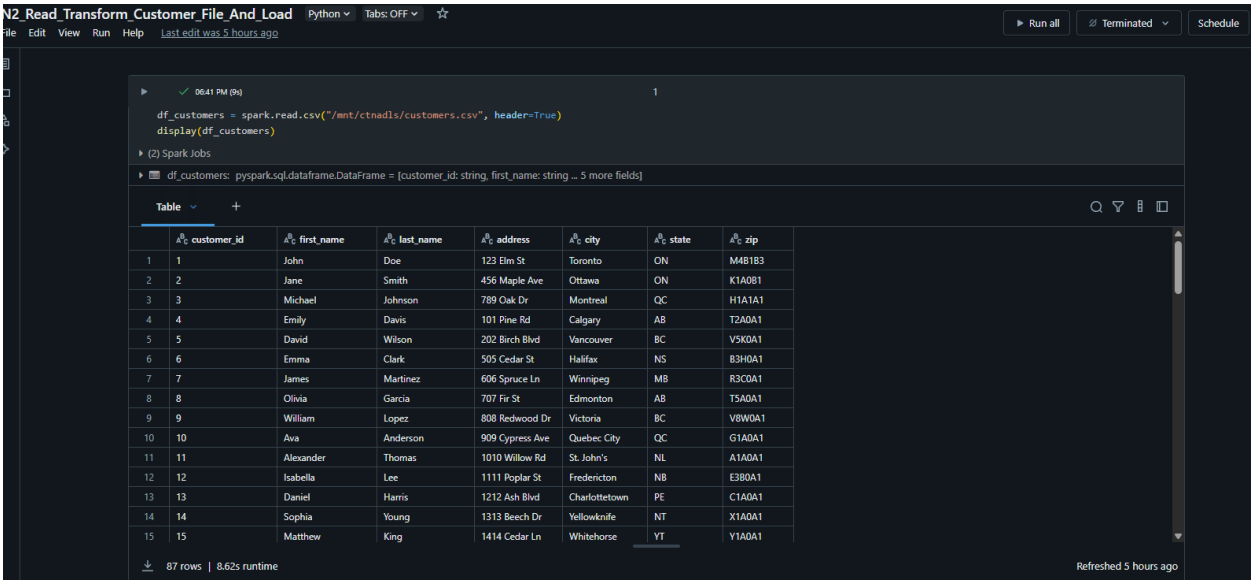
Table | New chart

Table view

	account_id	customer_id	account_type	balance
1	100	50	Checking	1111
2	101	51	Checking	1111

Notebook 2

Read the CSV file, do the transformation and load it into ADLS Gen 2

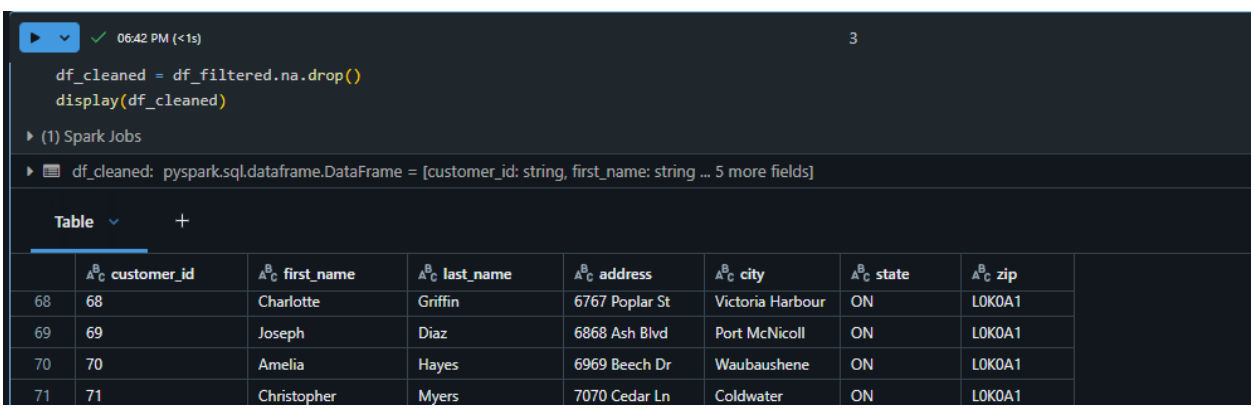


	customer_id	first_name	last_name	address	city	state	zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1
9	9	William	Lopez	808 Redwood Dr	Victoria	BC	V8W0A1
10	10	Ava	Anderson	909 Cypress Ave	Quebec City	QC	G1A0A1
11	11	Alexander	Thomas	1010 Willow Rd	St. John's	NL	A1A0A1
12	12	Isabella	Lee	1111 Poplar St	Fredericton	NB	E3B0A1
13	13	Daniel	Harris	1212 Ash Blvd	Charlottetown	PE	C1A0A1
14	14	Sophia	Young	1313 Beech Dr	Yellowknife	NT	X1A0A1
15	15	Matthew	King	1414 Cedar Ln	Whitehorse	YT	Y1A0A1

Filter out the null customer ids:

	customer_id	first_name	last_name	address	city	state	zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1

Drop the null records:



	customer_id	first_name	last_name	address	city	state	zip
68	68	Charlotte	Griffin	6767 Poplar St	Victoria Harbour	ON	L0K0A1
69	69	Joseph	Diaz	6868 Ash Blvd	Port McNicoll	ON	L0K0A1
70	70	Amelia	Hayes	6969 Beech Dr	Waubaushene	ON	L0K0A1
71	71	Christopher	Myers	7070 Cedar Ln	Coldwater	ON	L0K0A1

Rename the column name:

06:44 PM (1s) 4

```
df_renamed = df_cleaned.withColumnRenamed("customer_id", "CustomerID").withColumnRenamed("first_name", "FirstName").withColumnRenamed("last_name", "LastName").withColumnRenamed
("address", "Address").withColumnRenamed("city", "City").withColumnRenamed("state", "State").withColumnRenamed("zip", "Zip")

display(df_renamed)
```

(1) Spark Jobs

df_renamed: pyspark.sql.dataframe.DataFrame = [CustomerID: string, FirstName: string ... 5 more fields]

Table +

	CustomerID	FirstName	LastName	Address	City	State	Zip
72	72	Mia	Ford	7171 Elm St	Orillia	ON	L3V0A1
73	73	Andrew	Hamilton	7272 Maple Ave	Gravenhurst	ON	P1P0A1
74	74	Harper	Graham	7373 Oak Dr	Bala	ON	P0C0A1

Writing it into ADLS Gen 2 storage account after cleaning:

06:47 PM (2s) 5

```
df_renamed.write.mode("overwrite").option("header", "true").csv("/mnt/ctnadls/customers_cleaned.csv")
```

(1) Spark Jobs

+ Code + Text

06:47 PM (1s) 6

```
df_customers_cleaned = spark.read.csv("/mnt/ctnadls/customers_cleaned.csv", header=True)
display(df_customers_cleaned)
```

(2) Spark Jobs

df_customers_cleaned: pyspark.sql.dataframe.DataFrame = [CustomerID: string, FirstName: string ... 5 more fields]

Table +

Download this in ipynb file format

Now, open the Fabric Workspace and click on Import Notebook

Make necessary changes:

```
sasToken = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "adls-stg-sas-token")
notebookutils.fs.mount(
    "abfss://ctnadls@adlsharpal.dfs.core.windows.net",
    "/test",
    {"sasToken":sasToken}
)
containerPath = notebookutils.fs.getMountPath("/test")
```

✓ - Command executed in 880 ms by Fabric on 7:15:53 PM, 5/08/25

Table + New chart

Table view

	localPath	mountPoint	scope	source	storageType
1	/synfs/notebo...	/test	job	abfss://ctna...	Data Lake Storag...

```
df_customers = spark.read.csv(f"file://{containerPath}/customers.csv", header=True)
display(df_customers)
```

✓ - Command executed in 2 sec 494 ms by Fabric on 7:11:40 PM, 5/08/25

Diagnostics 1

Table | New chart | 7 col

Table view | Download

	customer_id	first_name	last_name	address	city	state	zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple A...	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1
8	8	Olivia	Garcia	707 Fir St	Edmonton	AB	T5A0A1

Filter Null Values:

```
from pyspark.sql.functions import col, when, lit
df_filtered = df_customers.filter((col("customer_id").isNotNull()))
display(df_filtered)
```

✓ - Command executed in 832 ms by Fabric on 7:15:58 PM, 5/08/25

Diagnostics 1

Table | New chart

Table view | Downloa

Drop Null records:

```
df_cleaned = df_filtered.na.drop()
display(df_cleaned)
```

✓ - Command executed in 821 ms by Fabric on 7:11:52 PM, 5/08/25

Diagnostics 1

Table | New chart

Table view

	customer_id	first_name	last_name	address	city	state	zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple A...	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1

Rename Column Name:

```
df_renamed = df_cleaned.withColumnRenamed("customer_id", "CustomerID").withColumnRenamed("first_name", "First

display(df_renamed)
```

✓ - Command executed in 882 ms by Fabric on 7:16:01 PM, 5/08/25

Diagnostics 1

Table | New chart

Table view

	CustomerID	FirstName	LastName	Address	City	State	Zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1
6	6	Emma	Clark	505 Cedar St	Halifax	NS	B3H0A1
7	7	James	Martinez	606 Spruce Ln	Winnipeg	MB	R3C0A1

Write cleaned data into the ADLS Gen 2 storage account:

```
#df_renamed.write.mode("overwrite").option("header", "true").csv("/mnt/ctnadls/customers_cleaned.csv")

df_renamed.write \
    .mode("overwrite") \
    .option("header", "true") \
    .csv(f"file:///{containerPath}/customers_cleaned_from_fabric.csv")
```

14] ✓ - Command executed in 4 sec 704 ms by Fabric on 7:16:55 PM, 5/08/25

Diagnostics 1

```
df_customers_cleaned = spark.read.csv(f"file:///{containerPath}/customers_cleaned_from_fabric.csv", header=True)
display(df_customers_cleaned)
```

15] ✓ - Command executed in 1 sec 490 ms by Fabric on 7:17:00 PM, 5/08/25

Table | New chart

Table view

	CustomerID	FirstName	LastName	Address	City	State	Zip
1	1	John	Doe	123 Elm St	Toronto	ON	M4B1B3
2	2	Jane	Smith	456 Maple Ave	Ottawa	ON	K1A0B1
3	3	Michael	Johnson	789 Oak Dr	Montreal	QC	H1A1A1
4	4	Emily	Davis	101 Pine Rd	Calgary	AB	T2A0A1
5	5	David	Wilson	202 Birch Blvd	Vancouver	BC	V5K0A1

Notebook 3

We will create a function to read tables and SQL queries in Databricks:

Weekend_Project_Data_Handling_with_SQL_Database_ADLS Gen2_and_Databricks

```python
def read_sql(table_name):
  df = (spark.read
  .format("jdbc")
  .option("url", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqldb-url'))
  .option("dbtable", table_name)
  .option("user", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqluser'))
  .option("password", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqlpassword'))
  .load()
  )
  return df
```

```python
def read_sql_query(query):
  df = (spark.read
  .format("jdbc")
  .option("url", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqldb-url'))
  .option("query", query)
  .option("user", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqluser'))
  .option("password", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqlpassword'))
  .load()
  )
  return df
```

```python
def write_data_sql(table_name, dataframe, mode):
    (dataframe.write
    .format("jdbc")
    .mode(mode)
    .option("url", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqldb-url'))
    .option("dbtable", table_name)
    .option("user", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqluser'))
    .option("password", dbutils.secrets.get(scope = 'azure_sqldb_scope', key = 'azuresqlpassword'))
    .save()
    )
```

Download it and upload it into the Fabric Workspace

Make necessary changes

```python
sqlpassword = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "azuresqldbpassword")
sqluser = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "azuresqluser")
url = notebookutils.credentials.getSecret("https://keyvault-harpal.vault.azure.net/", "azuresqldb-url")
```

- Command executed in 4 sec 70 ms by Fabric on 11:49:52 PM, 5/08/25 — PySpark (Python)

```python

def read_sql(table_name):
  df = (spark.read
  .format("jdbc")
  .option("url", url)
  .option("dbtable", table_name)
  .option("user", sqluser)
  .option("password", sqlpassword)
  .load()
  )
  return df
```

[0] - Waiting — PySpark (Python)

Notebook 4

Notebook 3 function we will use in this notebook

Databricks workspace:

```
1: Run Another notebook from this using magic command %run                                Python
%run "/Workspace/Harpal_Work_Files/Weekend_Project_Data_Handling_with_SQL_Database_ADLS_Gen2_and_Databricks"

2
df_table = read_sql(table_name="salesLt.Product")
display(df_table)

3: Remove Duplicate Record If Any
df_table_unique= df_table.dropDuplicates()
display(df_table_unique)

4: Remove Null Column
#Remove Null Column
df_table = df_table.drop("Weight","SellStartDate","SellEndDate", "ThumbNailPhoto", "DiscontinuedDate", "ThumbnailPhotoFileName", "rowguid", "ModifiedDate")

5
display(df_table)

6
df_table.printSchema()
```

```
#### ------------------------------------------------------------------------
from pyspark.sql.functions import col

fill_values = {
    'Color': 'Unknown',
    'Size': 'Not Specified'
}

df_table_replace = df_table.na.fill(fill_values)
display(df_table_replace)

8: Filter Data based on filter condition
df_table_replace = df_table_replace.filter((col("Size") > '45') & (col("Size").isin("M", "L", "XL")))
df_table_replace = df_table_replace.filter(~col("Color").isin("Unknown"))
display(df_table_replace)

9
df_table_col_renamed = df_table_replace.withColumnRenamed("Name","ProductName") #.withColumnRenamed("rowguid", "RowGUId")
display(df_table_col_renamed)

10
dbutils.fs.ls("/mnt/csvfiles")
```

11: Write Data to ADLS

```
df_table_col_renamed.write.format("delta").mode("overwrite").save("/mnt/csvfiles/ProductDetails")
```

12: Read Data from ADLS Delta Folder

```
df_productData = spark.read.format("delta").option("header","true").option("inferSchema", "true").load("/mnt/csvfiles/ProductDetails")
display(df_productData)
```

13: SCD Type 1 Logic start here:

```
%sql
--drop table hive_metastore.default.ProductDetailSCD1
```

14

```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.default.ProductDetailSCD1 (
    ProductID INT,
    ProductName STRING,
    ProductNumber STRING,
    Color STRING,
    StandardCost DECIMAL(10, 4),
    ListPrice DECIMAL(10, 4),
    Size STRING,
    ProductCategoryID INT,
    ProductModelID INT,
    HashKey BIGINT,
    CreatedDate TIMESTAMP,
    UpdatedDate TIMESTAMP,
    CreatedBy STRING,
    UpdatedBy STRING
)
USING DELTA
LOCATION '/mnt/csvfiles/scdtype1/gold/ProductDetailSCD1'
```

15

```
from pyspark.sql.functions import *
df_hash_derived = df_productData.withColumn("HashKey", crc32(concat(*df_productData.columns)))
display(df_hash_derived)
```

16

```
from delta.tables import DeltaTable
deltaTable = DeltaTable.forPath(spark, "/mnt/csvfiles/scdtype1/gold/ProductDetailSCD1")
deltaTable.toDF().show() #convert delta table to dataframe and display the data
```

17

```
df_src = (
    df_hash_derived.alias("src")
    .join(
        deltaTable.toDF().alias("tgt"),
        (col("src.ProductID") == col("tgt.ProductID")),  # Join only on id
        "left"
    )
    .filter( (col("tgt.ProductID").isNull()) | (col("src.HashKey") != col("tgt.HashKey")) )
    .select("src.*")
)
display(df_src)
```

```
18
deltaTable.alias("tgt").merge(
    df_src.alias("src"), "tgt.ProductID = src.ProductID"
).whenMatchedUpdate(
    set={
        "tgt.ProductID": "src.ProductID",
        "tgt.ProductName": "src.ProductName",
        "tgt.ProductNumber": "src.ProductNumber",
        "tgt.Color": "src.Color",
        "tgt.StandardCost": "src.StandardCost",
        "tgt.ListPrice": "src.ListPrice",
        "tgt.Size": "src.Size",
        "tgt.ProductCategoryID": "src.ProductCategoryID",
        "tgt.ProductModelID": "src.ProductModelID",
        "tgt.HashKey": "src.HashKey",
        "tgt.UpdatedBy": lit("Harpal-Updated"),
        "tgt.UpdatedDate": current_timestamp(),
    }
).whenNotMatchedInsert(
    values={
        "tgt.ProductID": "src.ProductID",
        "tgt.ProductName": "src.ProductName",
        "tgt.ProductNumber": "src.ProductNumber",
        "tgt.Color": "src.Color",
        "tgt.StandardCost": "src.StandardCost",
        "tgt.ListPrice": "src.ListPrice",
        "tgt.Size": "src.Size",
        "tgt.ProductCategoryID": "src.ProductCategoryID",
        "tgt.ProductModelID": "src.ProductModelID",
        "tgt.HashKey": "src.HashKey",
        "tgt.CreatedBy": lit("Harpal"),
        "tgt.CreatedDate": current_timestamp(),
        "tgt.UpdatedBy": lit("Harpal"),
        "tgt.UpdatedDate": current_timestamp(),
    }
).execute()
```

```
19
%sql
select * from hive_metastore.default.ProductDetailSCD1;
```

```
20
%sql
describe history hive_metastore.default.ProductDetailSCD1;
```

Fabric Workspace steps:

Modify the code according to the support for Fabric Notebook

```
[1]  %run Weekend_Project_Data_Handling_with_SQL_Database_ADLS Gen2_and_Databricks
✓ 14 sec - Command executed in 7 sec 34 ms by Fabric on 12:02:10 AM, 5/09/25
```

```
[2]  df_table = read_sql(table_name="salesLt.Product")
     display(df_table)
✓ 4 sec - Command executed in 4 sec 868 ms by Fabric on 12:02:15 AM, 5/09/25
> Spark jobs (1 of 1 succeeded)  Resources
```

Modified code:

Write data into Lakehouse Table

```
#df_table_col_renamed.write.format("delta").mode("overwrite").save("/mnt/csvfiles/ProductDetails")
df_table_col_renamed.write.mode("overwrite").saveAsTable("ProductDetails")
```

[11] ✓ 29 sec - Command executed in 29 sec 135 ms by Fabric on 12:02:53 AM, 5/09/25 PySpark (Python)

Spark jobs (3 of 3 succeeded) Resources

Read Data from Lakehouse

```
df_productdetails = spark.sql("SELECT * FROM lakehouse_hv.productdetails LIMIT 1000")
display(df_productdetails)
```

[14] ✓ 1 sec - Command executed in 1 sec 534 ms by Fabric on 12:03:58 AM, 5/09/25 PySpark (Python)

Spark jobs (1 of 1 succeeded) Resources

```
%%sql
CREATE TABLE IF NOT EXISTS ProductDetailSCD1 (
    ProductID INT,
    ProductName STRING,
    ProductNumber STRING,
    Color STRING,
    StandardCost DECIMAL(10, 4),
    ListPrice DECIMAL(10, 4),
    Size STRING,
    ProductCategoryID INT,
    ProductModelID INT,
    HashKey BIGINT,
    CreatedDate TIMESTAMP,
    UpdatedDate TIMESTAMP,
    CreatedBy STRING,
    UpdatedBy STRING
)
```

✓ 3 sec - Command executed in 3 sec 462 ms by Fabric on 12:05:38 AM, 5/09/25

Spark jobs (1 of 1 succeeded) Resources Log

```
from pyspark.sql.functions import *
df_hash_derived = df_productdetails.withColumn("HashKey", crc32(concat(*df_productdetails.columns)))
display(df_hash_derived)
```

✓ <1 sec - Command executed in 904 ms by Fabric on 12:06:05 AM, 5/09/25

```
from delta.tables import DeltaTable
#deltaTable = DeltaTable.forPath(spark, "lakehouse_hv/Tables/productdetailscd1")

deltaTable = DeltaTable.forName(spark, "productdetailscd1")
deltaTable.toDF().show() #convert delta table to dataframe and display the data
```

✓ 1 sec - Command executed in 1 sec 521 ms by Fabric on 12:10:09 AM, 5/09/25 PySpark (Pythor

```
+---------+-----------+-------------+-----+------------+---------+----+-----------------+---------------+-------+-----------+-----------+---------+---------+
|ProductID|ProductName|ProductNumber|Color|StandardCost|ListPrice|Size|ProductCategoryID|ProductModelID|HashKey|CreatedDate|UpdatedDate|CreatedBy|UpdatedBy|
+---------+-----------+-------------+-----+------------+---------+----+-----------------+---------------+-------+-----------+-----------+---------+---------+
+---------+-----------+-------------+-----+------------+---------+----+-----------------+---------------+-------+-----------+-----------+---------+---------+
```

```
%%sql
describe ProductDetailSCD1;
```

✓ 1 sec - Command executed in 1 sec 459 ms by Fabric on 12:10:52 AM, 5/09/25

Table | New chart

Table view

	col_name	data_type	comment
1	ProductID	int	NULL
2	ProductName	string	NULL
3	ProductNumber	string	NULL
4	Color	string	NULL
5	StandardCost	decimal(10,4)	NULL
6	ListPrice	decimal(10,4)	NULL
7	Size	string	NULL
8	ProductCateg...	int	NULL
9	ProductModelID	int	NULL
10	HashKey	bigint	NULL
11	CreatedDate	timestamp	NULL

Key Modifications Required

Feature	Databricks Code	Fabric Code
Mount Point	/mnt/...	Files/...
Key Vault	dbutils.secrets.get(...)	notebookutils.credentials.getSecret(...)
File Write	.write.csv("/mnt/...)`	.write.csv("Files/...)`
Table Read	.load("/mnt/table")	.read.table("tableName")

Conclusion

The migration from Databricks to Microsoft Fabric required rewriting some logic, removing mount dependencies, adjusting secret access methods, and ensuring compatibility with Fabric's OneLake file model. After applying these changes, all four notebooks were successfully executed in the Fabric Workspace using Delta tables and Lakehouse-backed CSV storage.