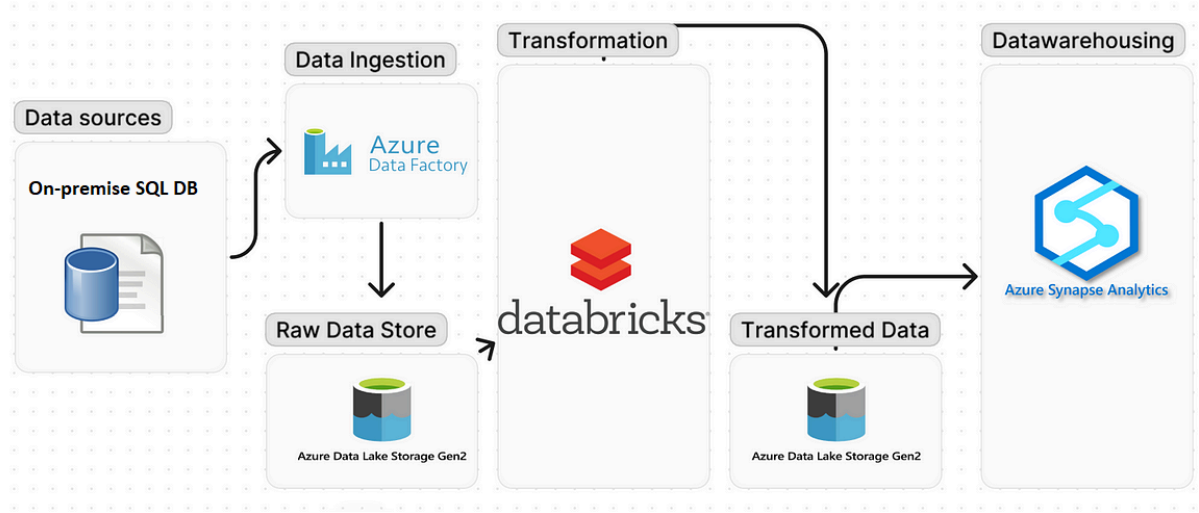
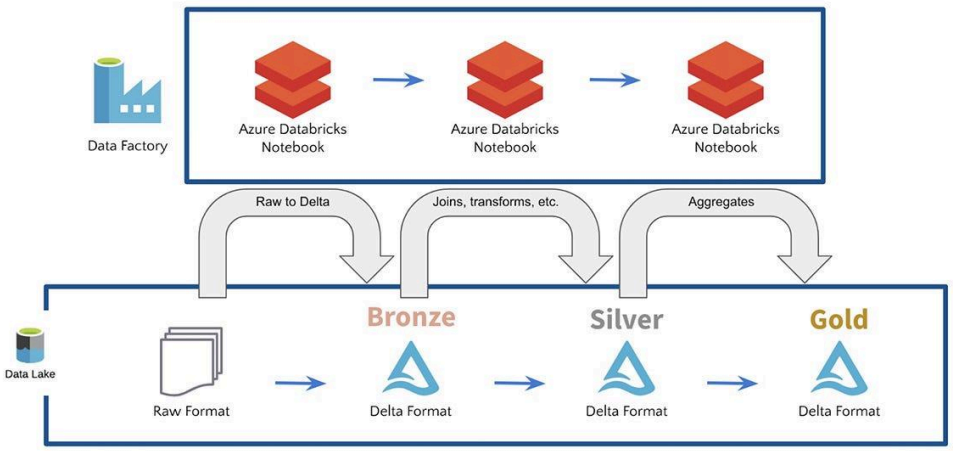
Project 2

Project Architecture:

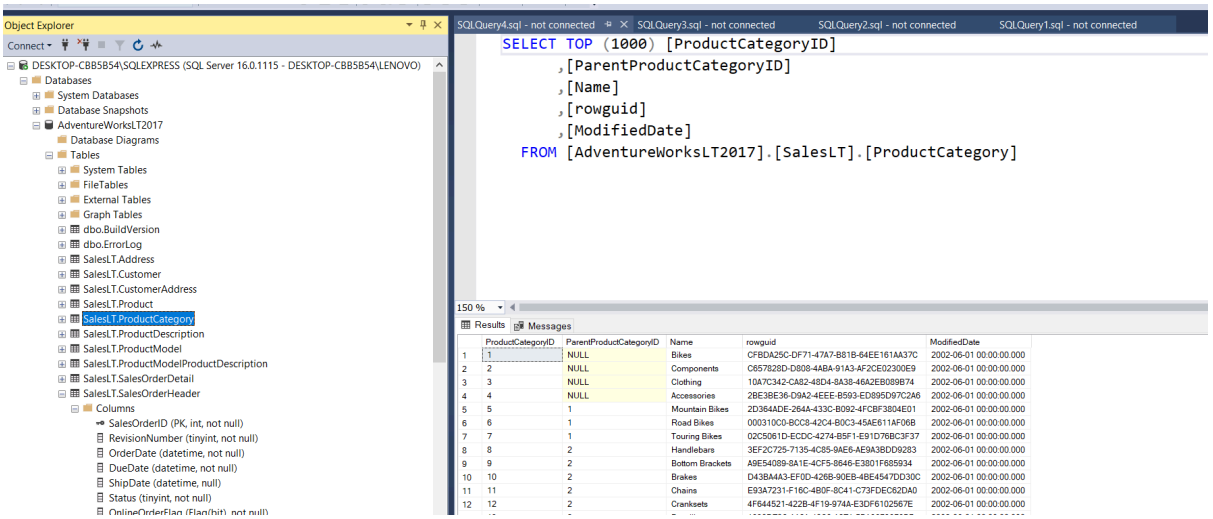




Part 1 - Dataset Overview:

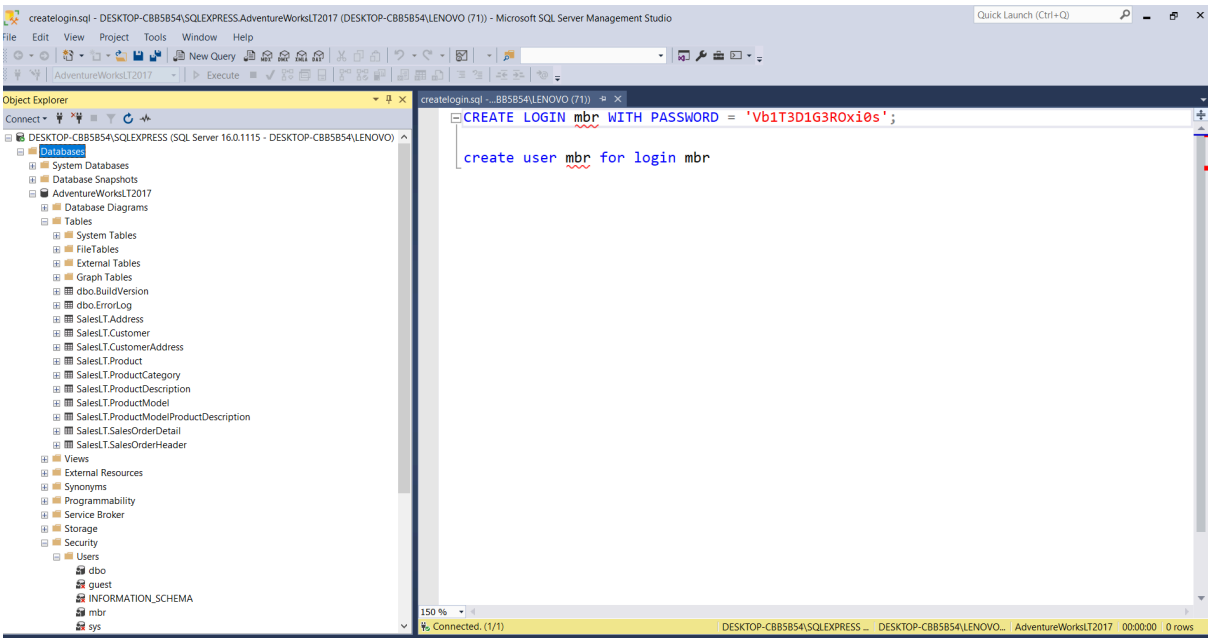
AdventureWorks database supports standard online transaction processing scenarios for a fictitious bicycle manufacturer - Adventure Works Cycles. Scenarios include Manufacturing, Sales, Purchasing, Product Management, Contact Management, and Human Resources.

Mahesh Raut - Project - 31 May 2024



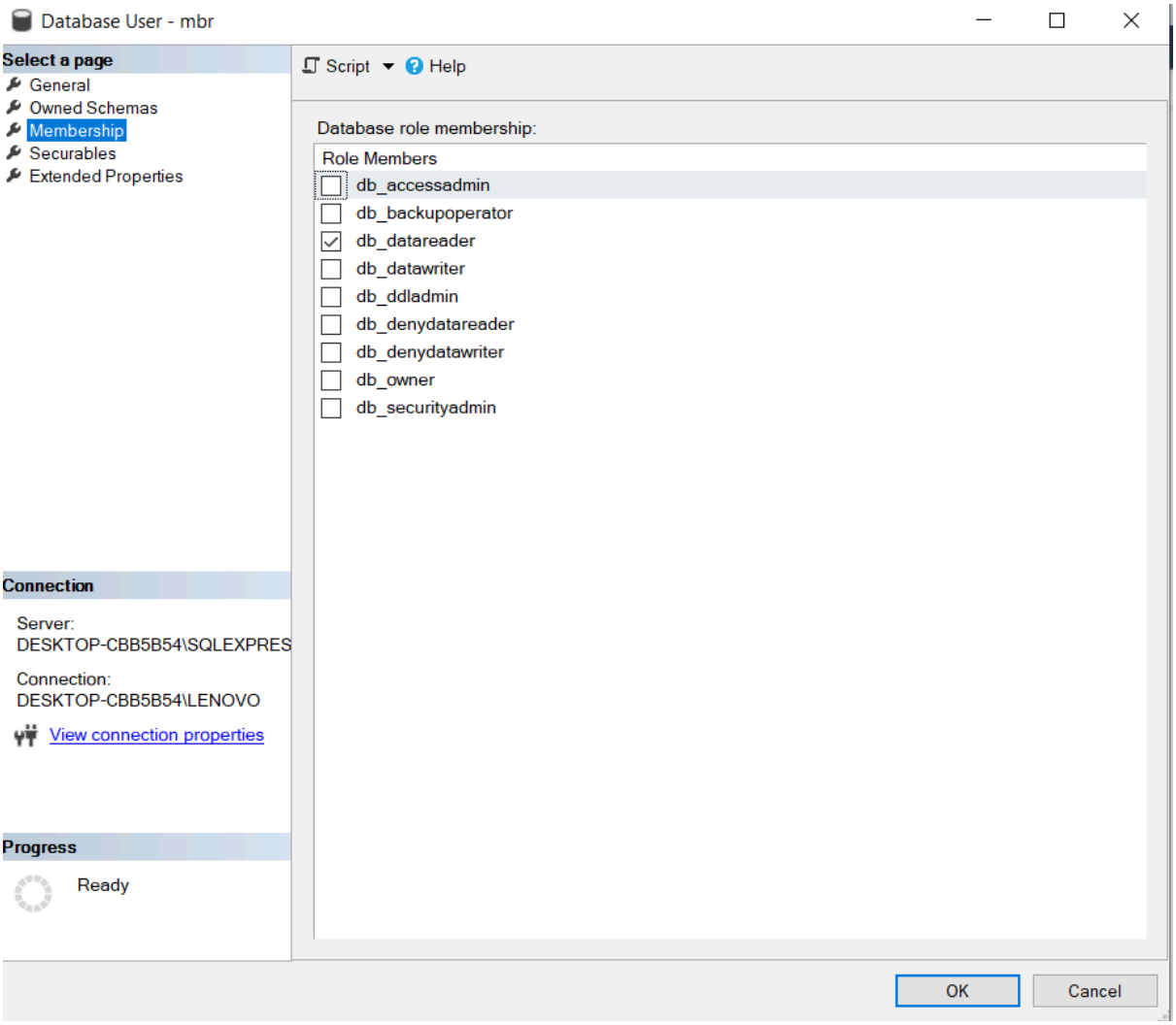
Create Database login

```
CREATE LOGIN mbr WITH PASSWORD = 'Vb1T3D1G3ROxi0s';
create user mbr for login mbr
```

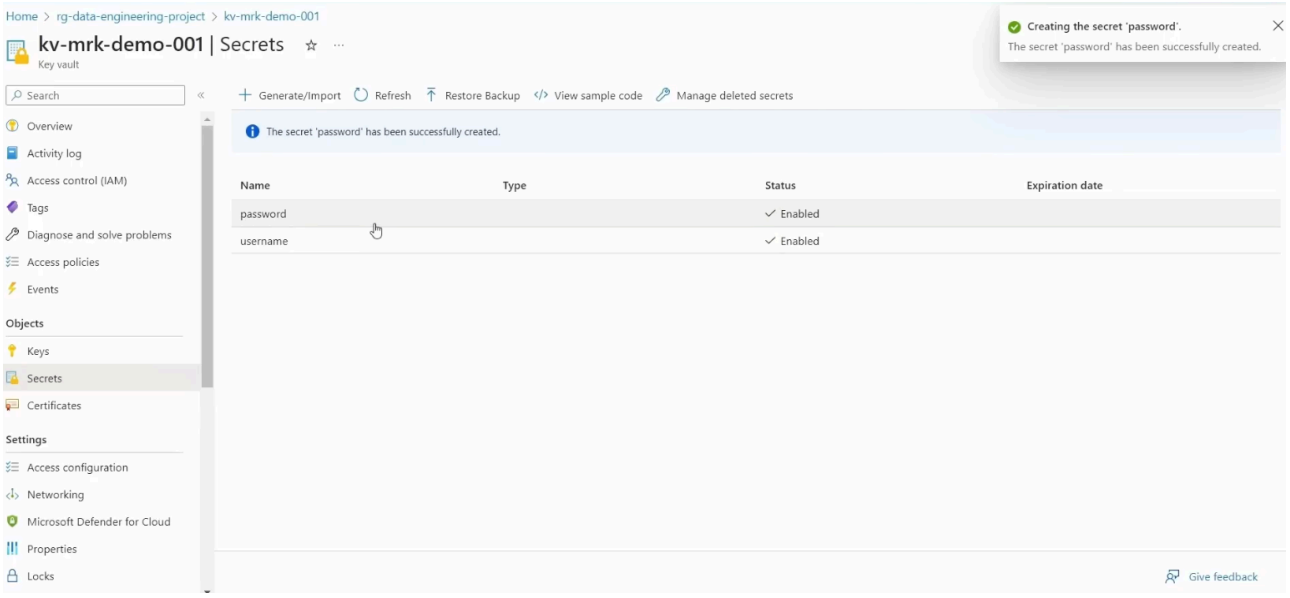


Mahesh Raut - Project - 31 May 2024

Give User access:

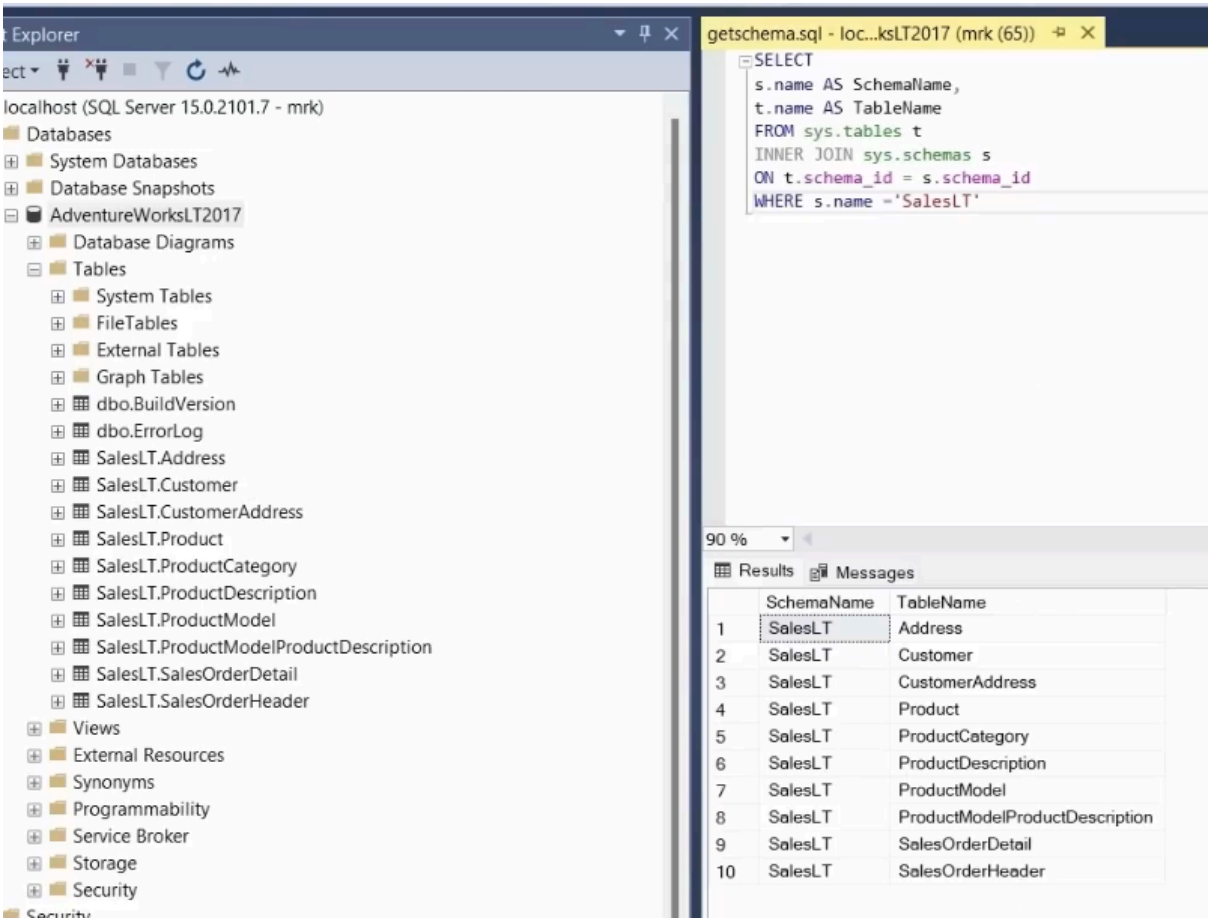


Store secrets in Azure Key Vault:



Part 2 - Data Ingestion: From On-premise to Azure Data Lake Storage (Bronze container) (SQL DB to Parquet Format)

getschema.sql

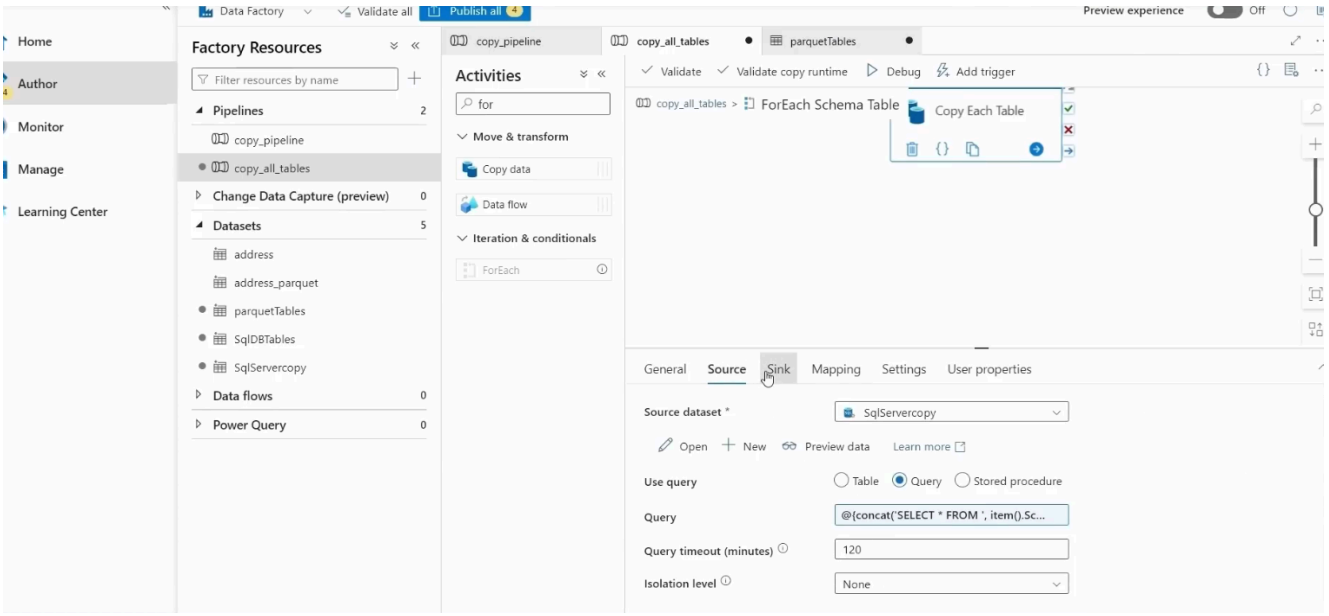


Folder Structure in Data Lake:

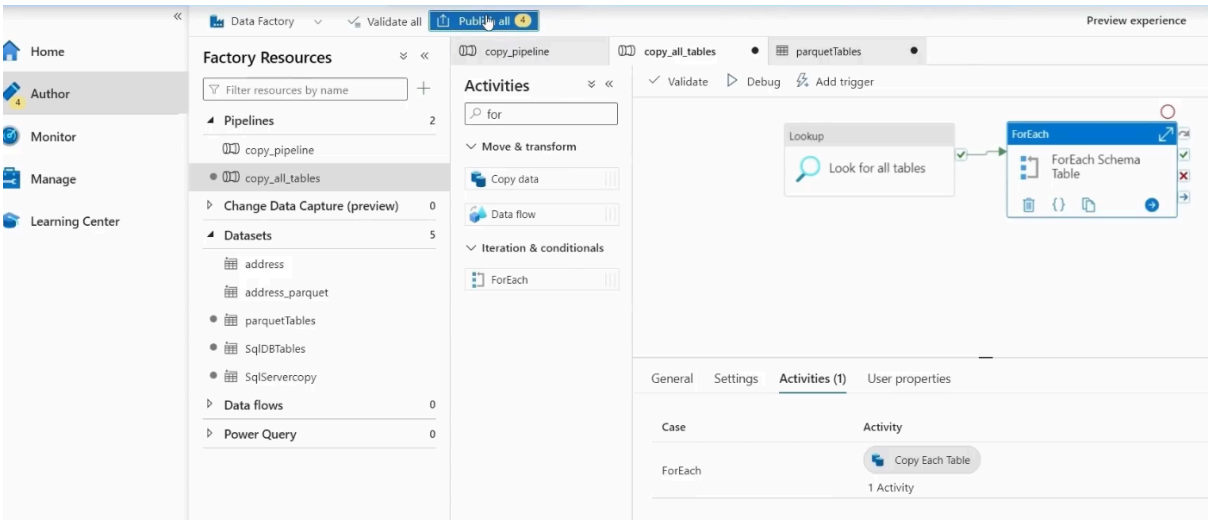
```
bronze/Schema/Tablename/Tablename.parquet
Ex. bronze/SalesLT/Address/Address.parquet
```

Mahesh Raut - Project - 31 May 2024

Source dataset - sql db

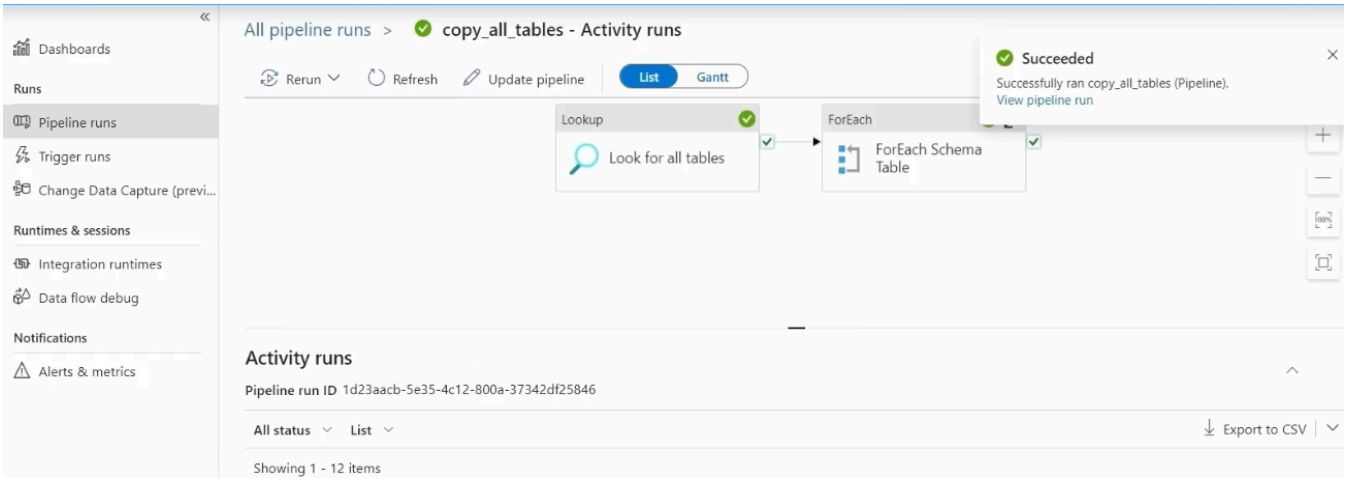


Pipeline build for looping each table in sql db to copy all tables in bronze container

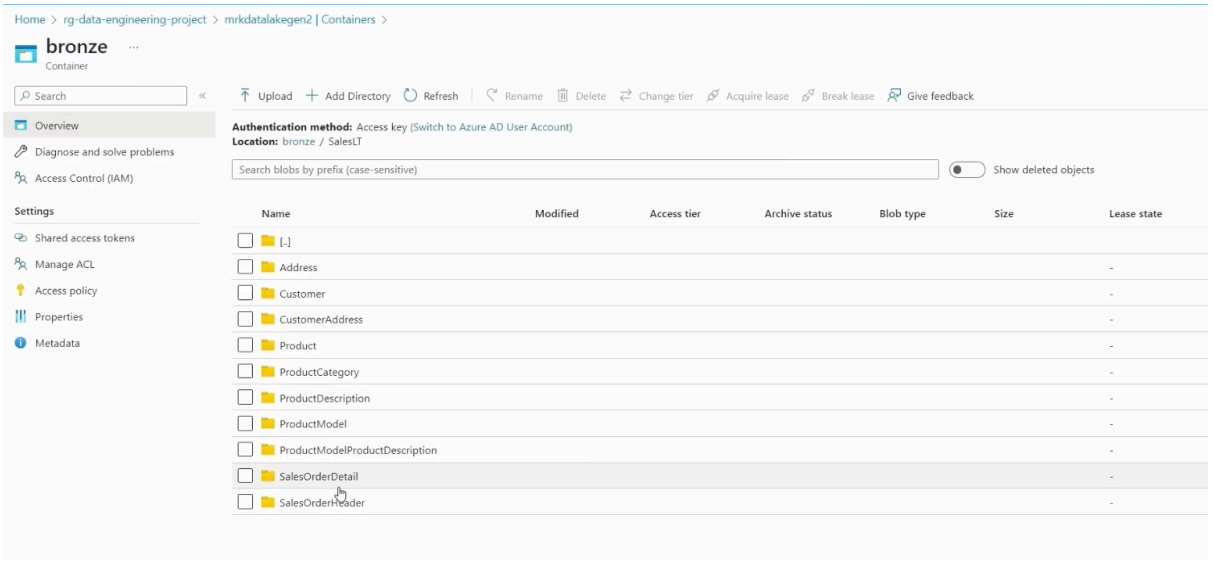


Mahesh Raut - Project - 31 May 2024

Trigger pipeline once - copy all tables

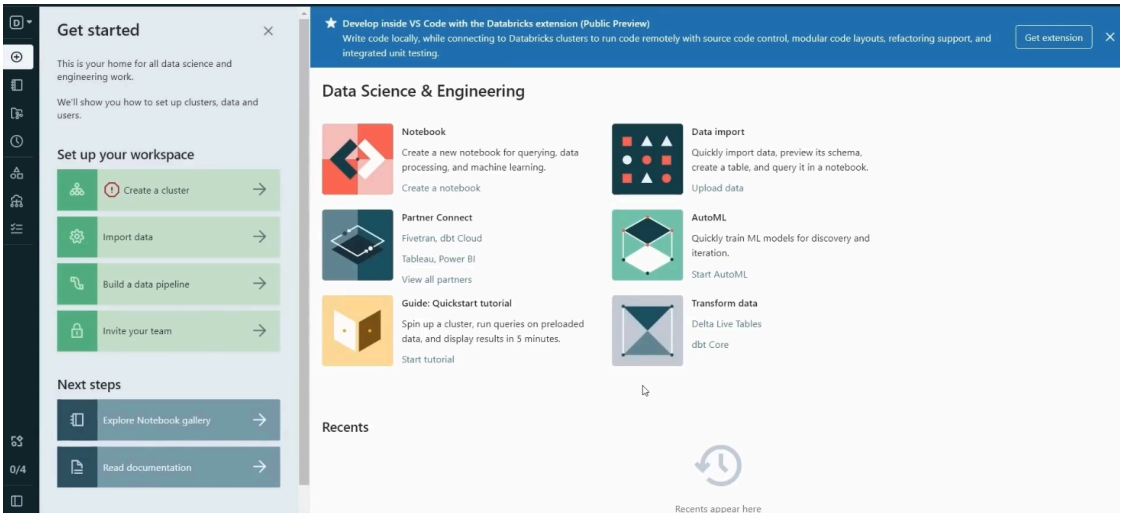


Check bronze container



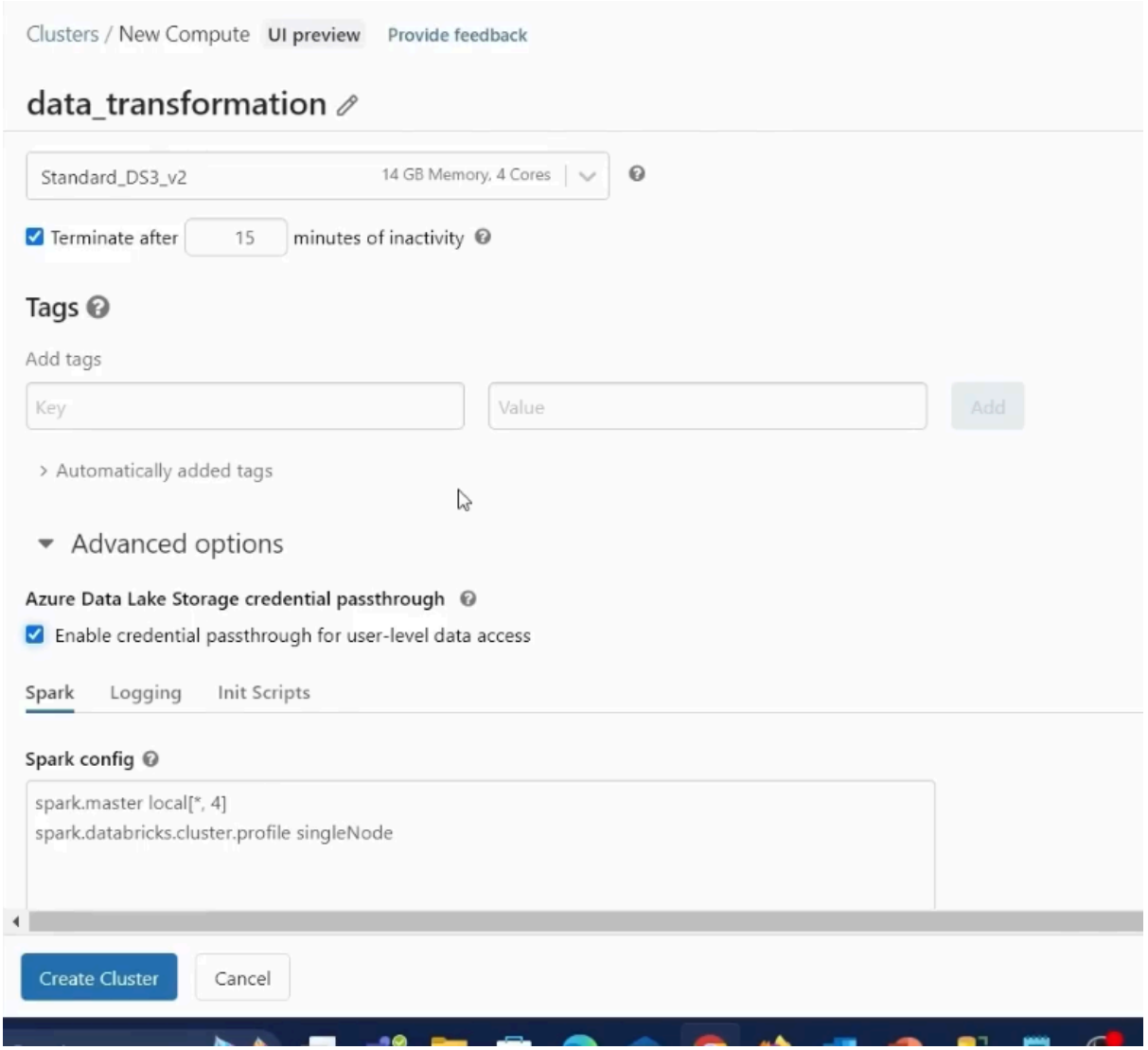
Part 3 - Data Transformation

Azure Databricks Overview

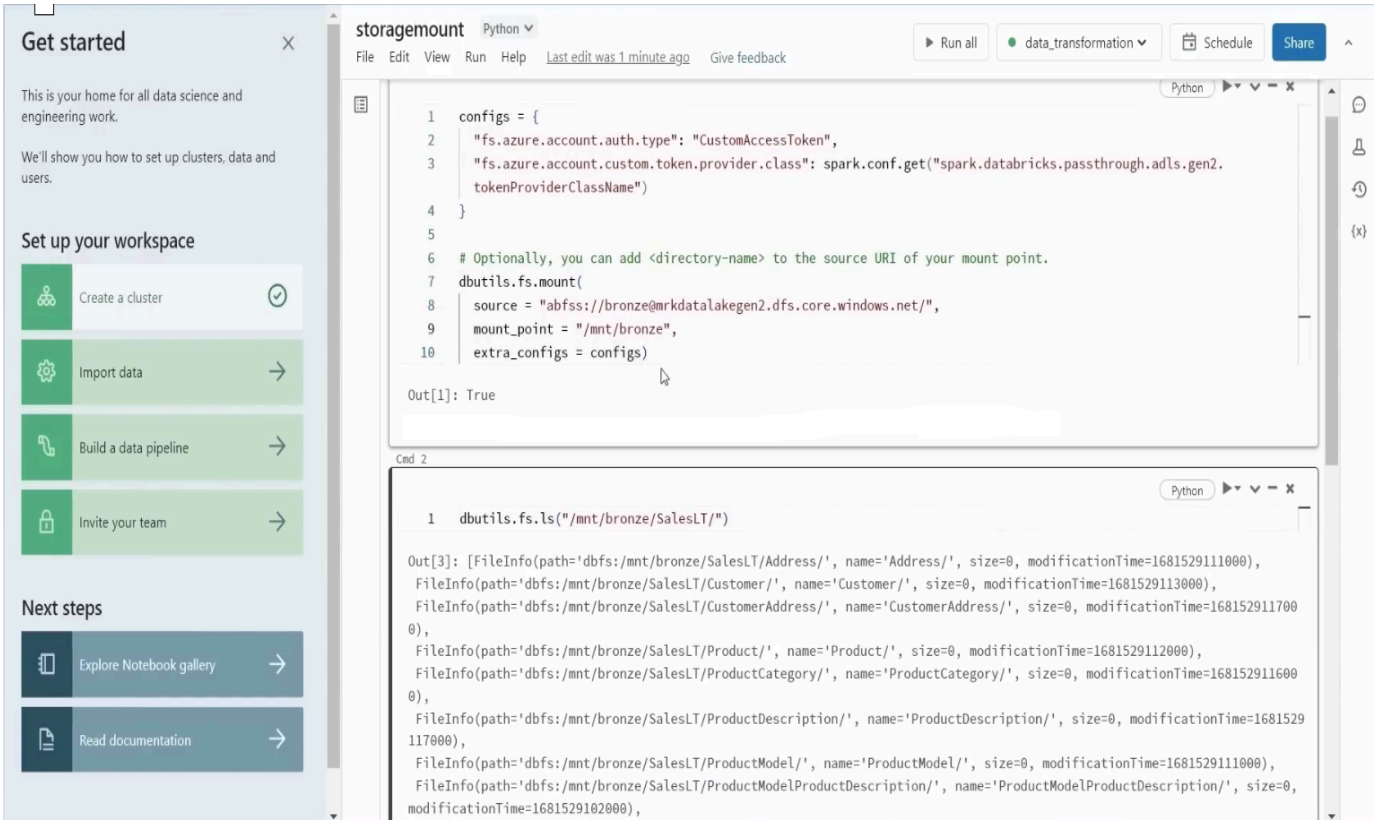


Mahesh Raut - Project - 31 May 2024

Create Compute Cluster

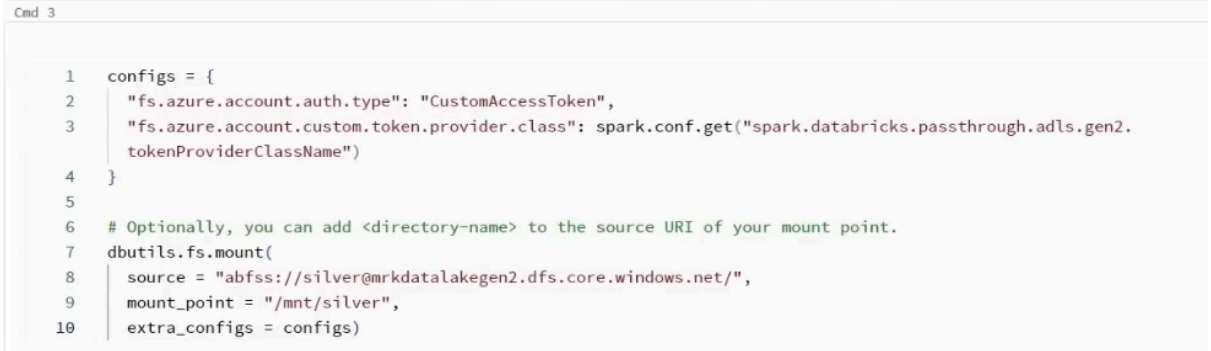


Storagemount notebook - bronze, silver, gold




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

dbutils.fs.mount(
    source = "abfss://bronze@mrkdatakagen2.dfs.core.windows.net/",
    mount_point = "/mnt/bronze",
    extra_configs = configs
)

dbutils.fs.ls("/mnt/bronze/SalesLT/")
```

Silvermount



Goldmount

```
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@mrkdatalakegen2.dfs.core.windows.net/",
  mount_point = "/mnt/gold",
  extra_configs = configs)
```

```
Out[5]: True
```

Bronze to silver notebook

Convert ModifiedDate column with Date Column

ModifiedDate
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000
2002-06-01 00:00:00.000

---------- to ----------

ModifiedDate
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01
2002-06-01

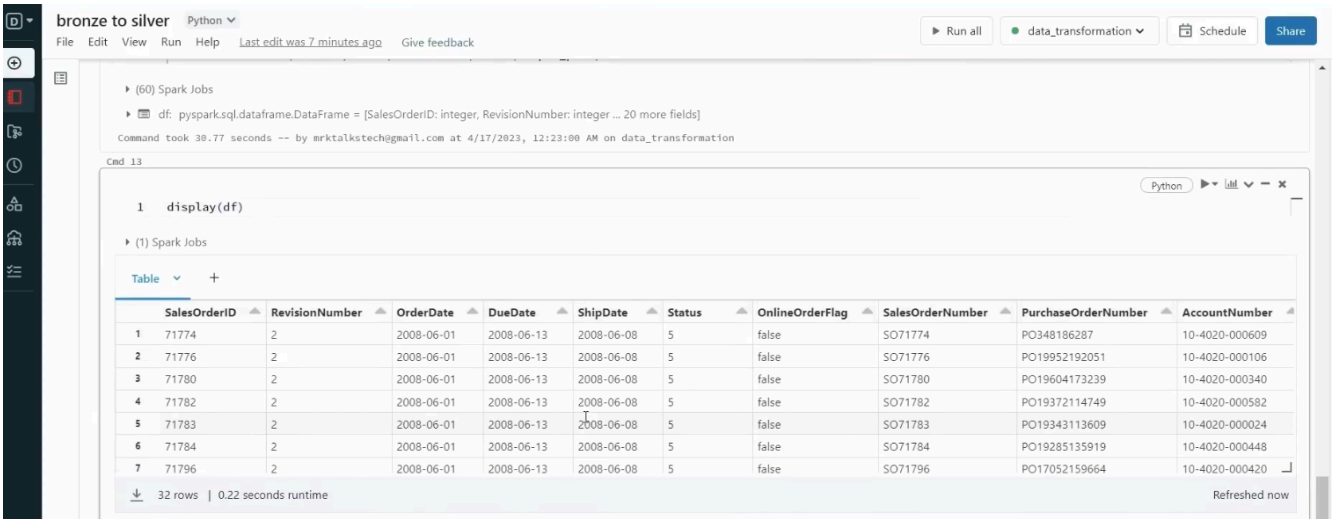
```
dbutils.fs.ls('mnt/bronze/SalesLT/')

dbutils.fs.ls('mnt/silver/')

input_path = '/mnt/bronze/SalesLT/Address/Address.parquet'

df = spark.read.format('parquet').load(input_path)

display(df)

from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

df = df.withColumn("ModifiedDate",
date_format(from_utc_timestamp(df["ModifiedDate"].cast(TimestampType()),
"UTC"), "yyyy-MM-dd"))
```

Doing transformation for all tables

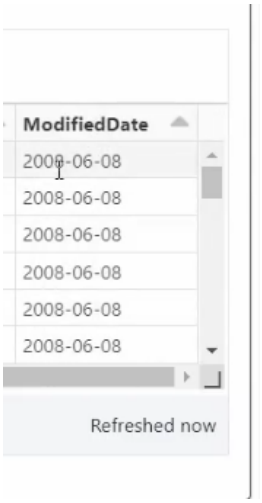
```
table_name = []
for i in dbutils.fs.ls('mnt/bronze/SalesLT/'):
        table_name.append(i.name.split('/')[0])
```

```
from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

# Loop through table names
for i in table_name:
  # Construct path to the Parquet file
  path = '/mnt/bronze/SalesLT/' + i + '/' + i + '.parquet'

  # Read data from Parquet file
  df = spark.read.format('parquet').load(path)

  # Get list of columns in the DataFrame
  column = df.columns

  # Loop through each column
  for col in column:
    # Check if the column name contains "Date" or "date"
    if "Date" in col or "date" in col:
      # Convert the column to timestamp format (assuming it's not
already a timestamp)
      df = df.withColumn(col,
date_format(from_utc_timestamp(df[col].cast(TimestampType()), "UTC"),
"yyyy-MM-dd"))

  # Construct path to write the transformed data
  output_path = '/mnt/silver/SalesLT/' + i + '/'

  # Write the transformed DataFrame to Delta format in overwrite mode
  df.write.format('delta').mode("overwrite").save(output_path)

display(df)
```

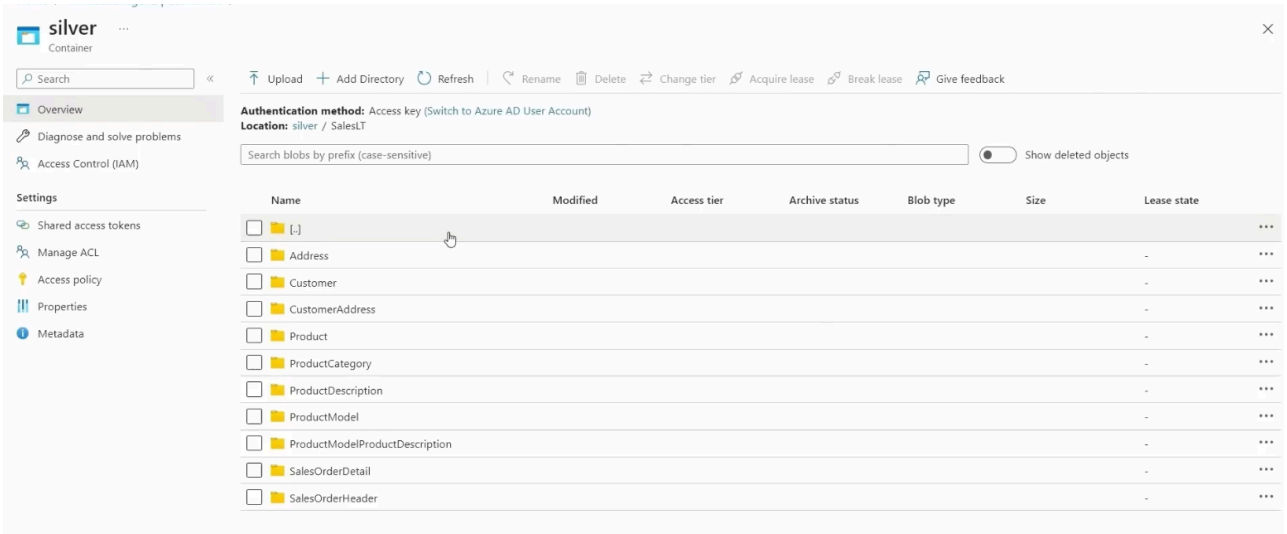
Mahesh Raut - Project - 31 May 2024



Updated column:

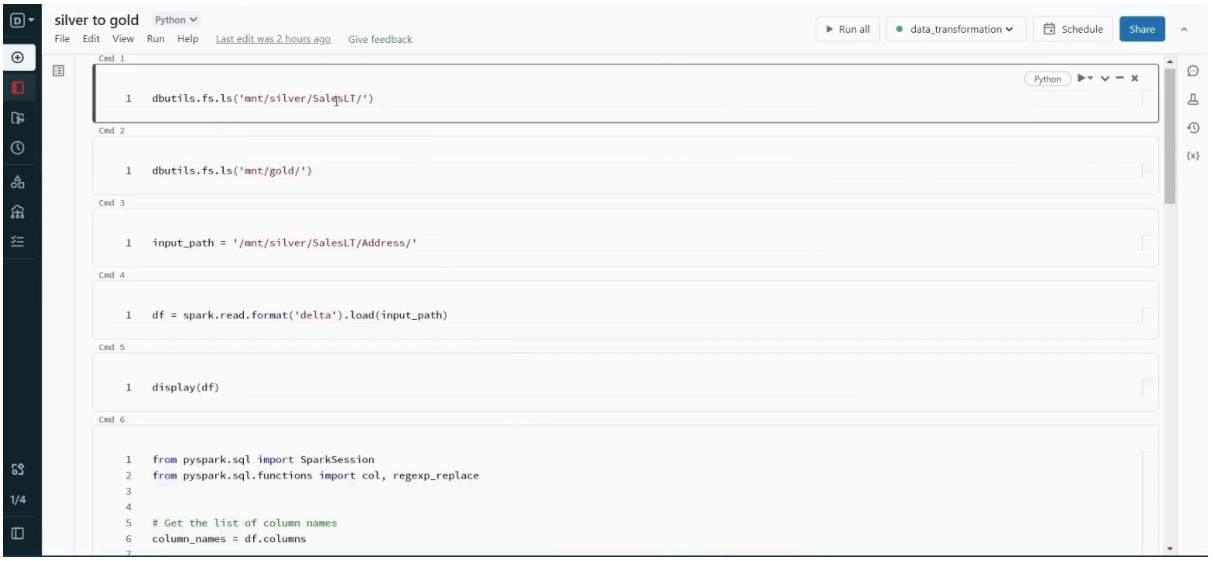


Silver container



Silver to gold notebook

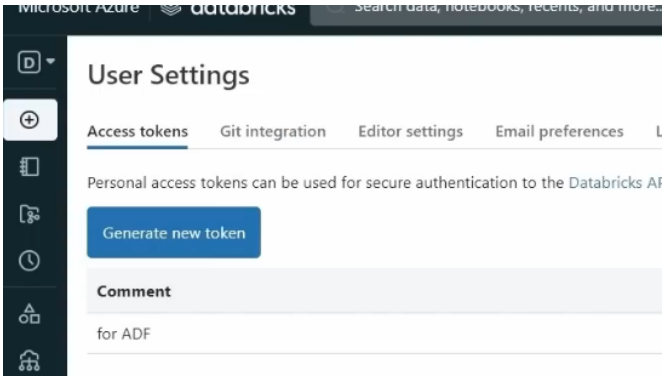
Transform the column names for all tables (ex. ProductId to Product_Id)

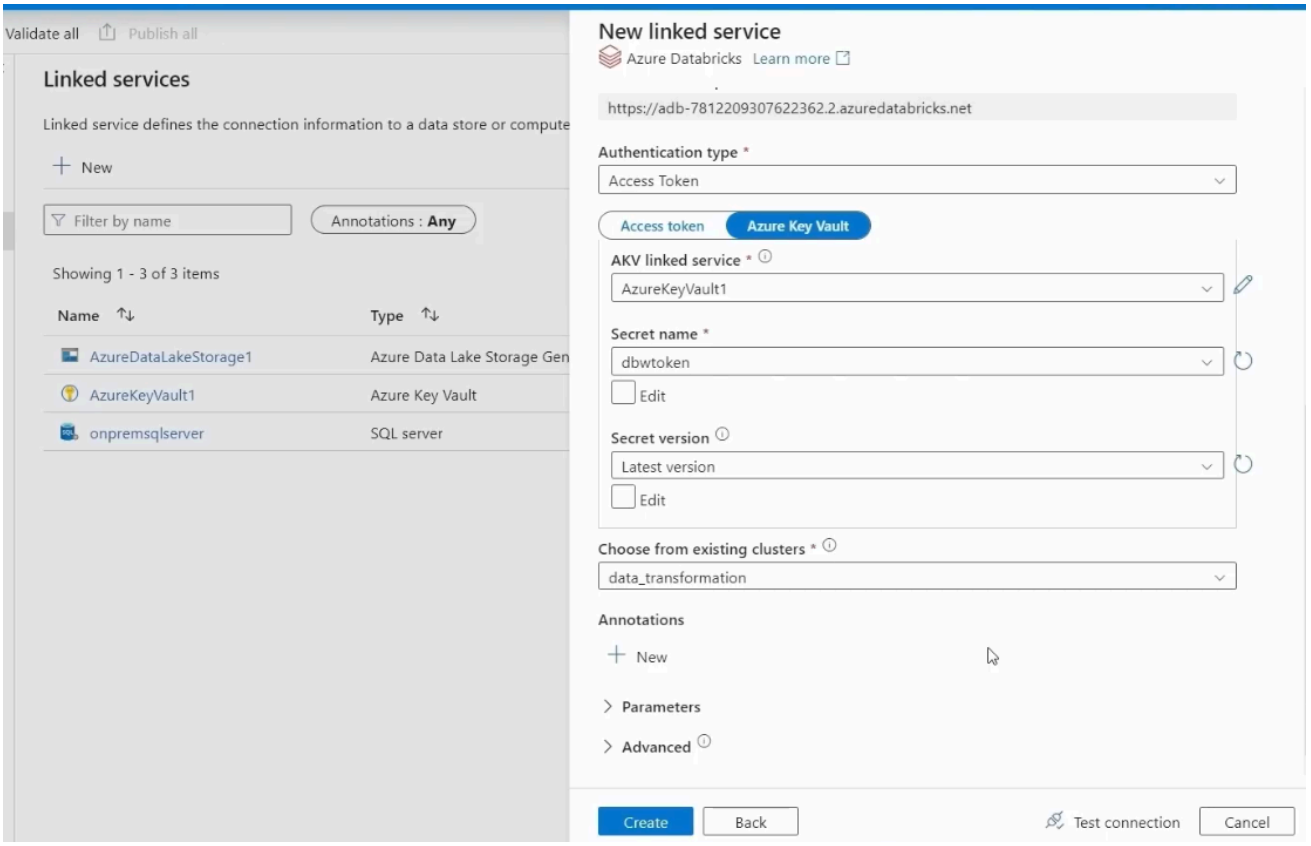


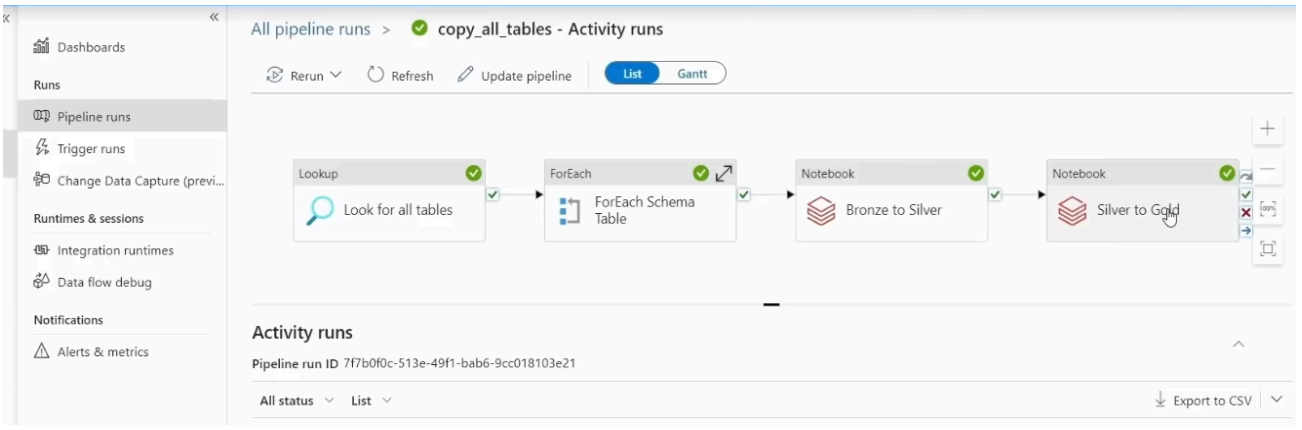

```
table_name = []
for i in dbutils.fs.ls('mnt/silver/SalesLT/'):
  table_name.append(i.name.split('/')[0])
# Get list of column names in column_names
column_names = df.columns

# Loop through each column name
for old_col_name in column_names:
  # Convert column name from CamelCase to snake_case format
  new_col_name = "".join([word.lower() if word.isupper() and not i == 0
else word for i, word in enumerate(old_col_name)]).strip("_")

  # Change the column name using withColumnRenamed
  df = df.withColumnRenamed(old_col_name, new_col_name)

# Print transformed DataFrame
print(df)
```

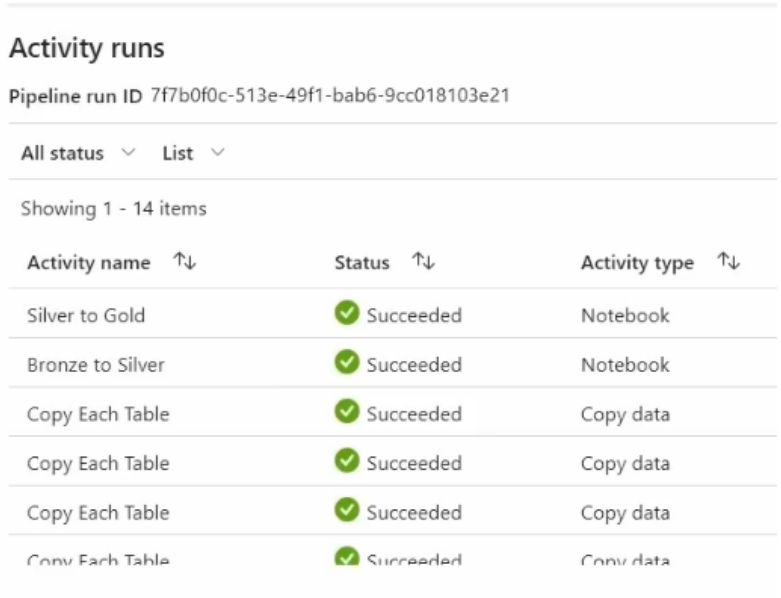
Mahesh Raut - Project - 31 May 2024

Make connection with cluster in ADF









Activity runs

Pipeline run ID 7f7b0f0c-513e-49f1-bab6-9cc018103e21

All status | List

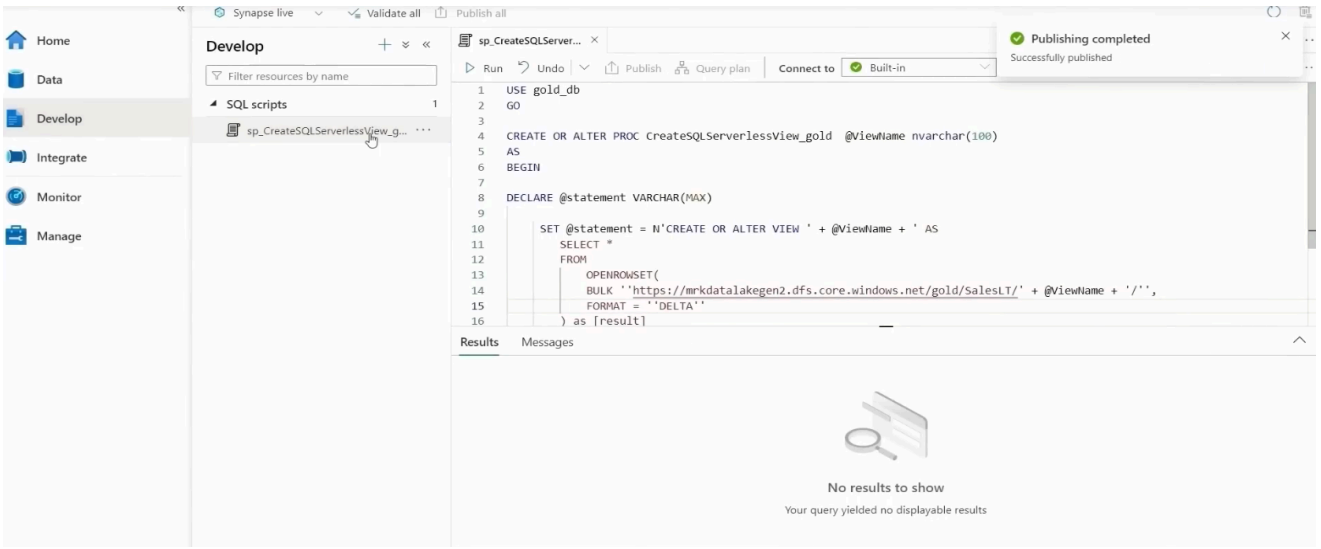
Showing 1 - 14 items

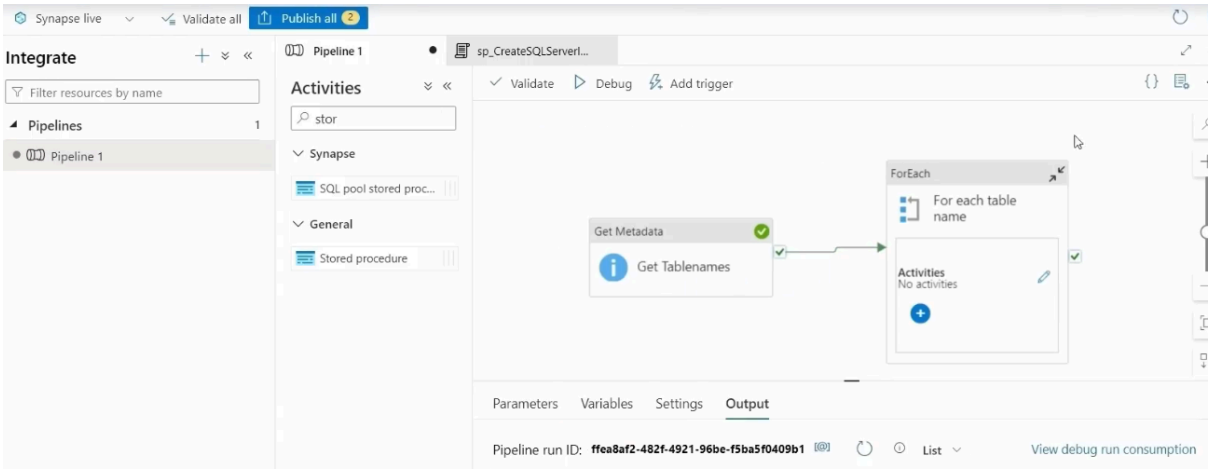
Activity name	Status	Activity type
Silver to Gold	Succeeded	Notebook
Bronze to Silver	Succeeded	Notebook
Copy Each Table	Succeeded	Copy data
Copy Each Table	Succeeded	Copy data
Copy Each Table	Succeeded	Copy data
Copy Each Table	Succeeded	Copy data

Part 4 - Data Loading - Gold container to Azure Synapse Serverless SQL Pool

```
USE gold_db
GO
CREATE OR ALTER PROC CreateSQLServerlessView_gold @ViewName nvarchar(100)
AS
BEGIN
DECLARE @statement VARCHAR (MAX)
SET @statement = N'CREATE OR ALTER VIEW ' + @ViewName + N' AS
                                SELECT *
                                FROM                            OPENROWSET (
                                                                        BULK
''https://mrkdatalakegen2.dfs.core.windows.net/gold/SalesLT/' + @ViewName + '/'',
                                                                        FORMAT =
''DELTA"
                                                                ) as [result]'

EXEC (@statement)
END
GO
```

Mahesh Raut - Project - 31 May 2024





After trigger: