Pranoy Chakraborty | +91-7996915434 | pcslg1998@gmail.com | LinkedIn | Portfolio

End-To-End Data Engineering Project using Microsoft Azure and Azure Databricks

Business Problem

Context:

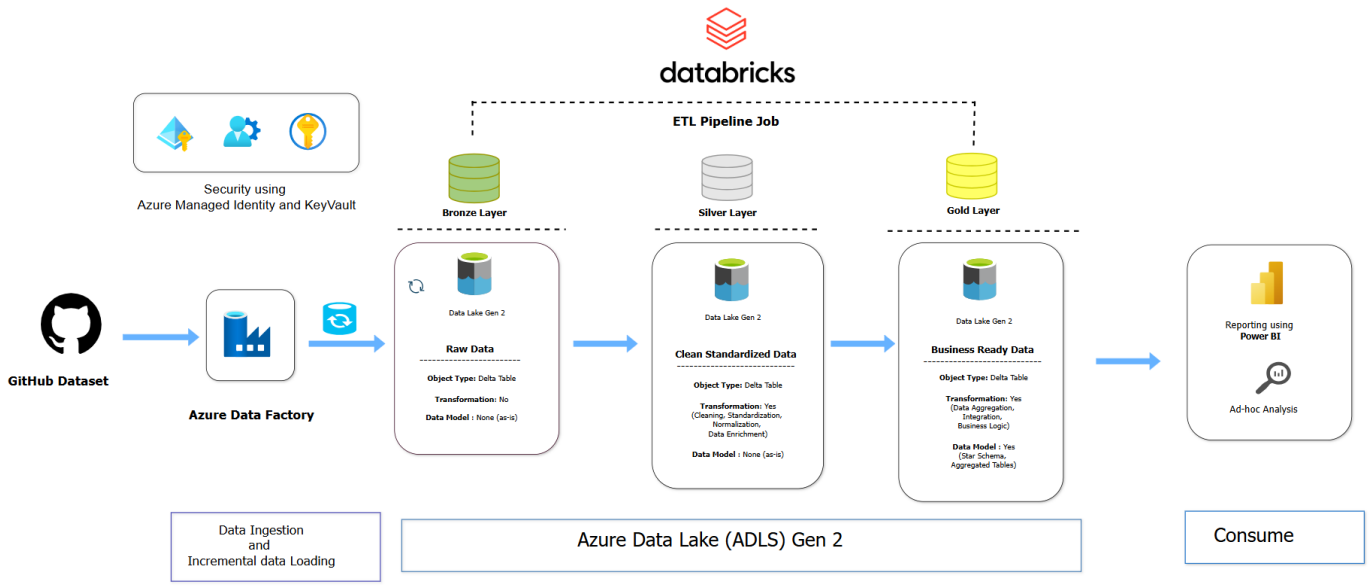
A retail company (or a similar enterprise in this case study) operates in multiple regions with diverse product lines and a network of resellers. Sales are driven by a distributed salesforce, and the company sets quarterly/annual sales targets at different levels (by product, region, or salesperson). However, the company faces challenges in tracking performance, identifying underperforming regions/resellers, and aligning sales strategy with business targets.

Key Business Problems:

- Lack of unified view of sales across regions, products, and resellers.
- Difficulty in tracking sales performance vs. targets.
- Inefficient allocation of resources to underperforming areas.
- Limited forecasting and strategy alignment due to poor data insights.

Proposed Data-Driven Solution:

- Ingest raw CSV datasets into **Azure Data Lake Gen2** using **Azure Data Factory**.
- Store raw, untransformed data in the **Bronze Layer** (Delta Tables) in Data Lake.
- Apply cleaning, standardization, and enrichment in the **Silver Layer**.
- Aggregate and apply business logic in the **Gold Layer** (Star Schema).
- Secure the pipeline with **Azure Managed Identity and Key Vault**.
- Deliver insights via **Power BI dashboard.**



GitHub Dataset

retail-analytics-pipeline-azure-databricks-project / data /

Add file

developersview updated files e92084c · 2 weeks ago History

Name	Last commit message	Last commit date
..		
Product.csv	updated files	2 weeks ago
Region.csv	updated data	2 weeks ago
Reseller.csv	updated files	2 weeks ago
Sales.csv	updated files	2 weeks ago
Salesperson.csv	updated data	2 weeks ago
SalespersonRegion.csv	updated data	2 weeks ago
Targets.csv	updated data	2 weeks ago

Resource Group

Resources used:

1. Azure Databricks
2. Azure Data Factory (ADF)
3. Azure Data Lake Storage (ADLS) Gen 2
4. Azure Databricks Access Connector

RG-DEProject
Resource group

How do I troubleshoot issues with this resource group? | How do I monitor this resource group? | +1

Search

Overview
Activity log
Access control (IAM)
Tags
Resource visualizer
Events
Settings
Cost Management
Monitoring
Automation
Help

Create | Manage view | Delete resource group | Refresh | Export to CSV | Open query | Assign tags

Essentials | JSON View

Resources | Recommendations (1)

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

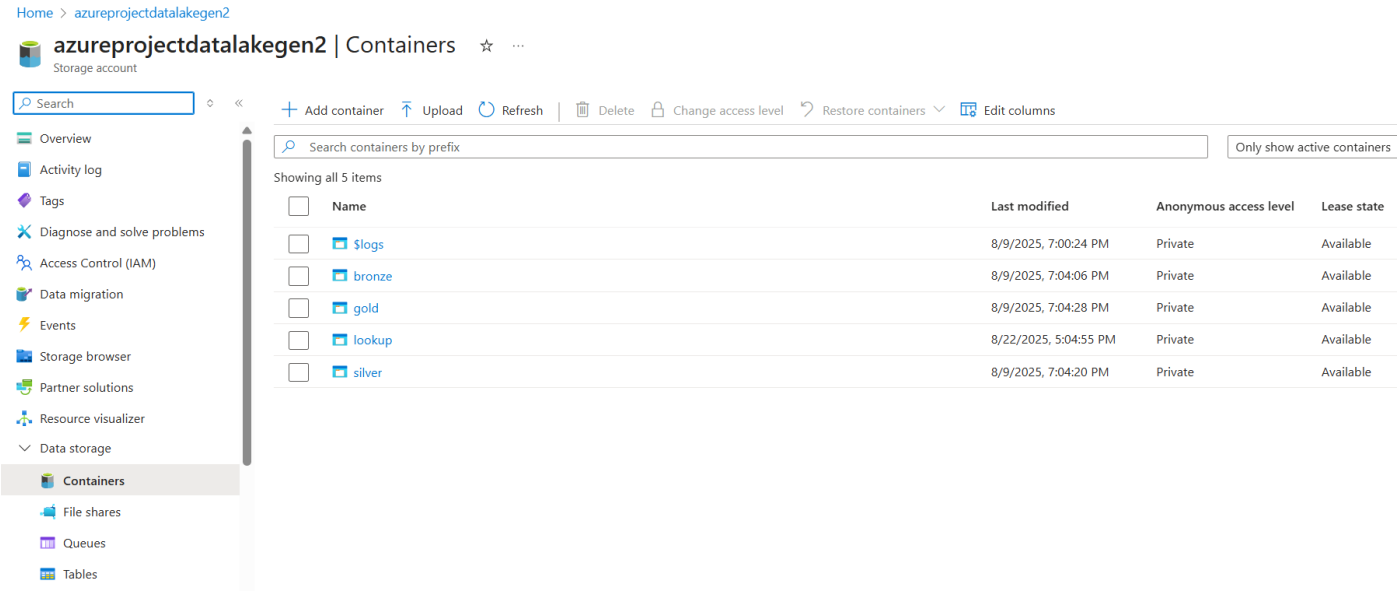
Showing 1 to 5 of 5 records. Show hidden types | No grouping | List view

Name	Type	Location
adb-retail-project-prd	Azure Databricks Service	West US 2
adf-retail-project-prd	Data factory (V2)	West US 2
azureprojectdatalakegen2	Storage account	West US
retail-project-access-connector	Access Connector for Azure Data...	West US 2
syn-retail-prd	Synapse workspace	UK South

< Previous | Page 1 of 1 | Next >

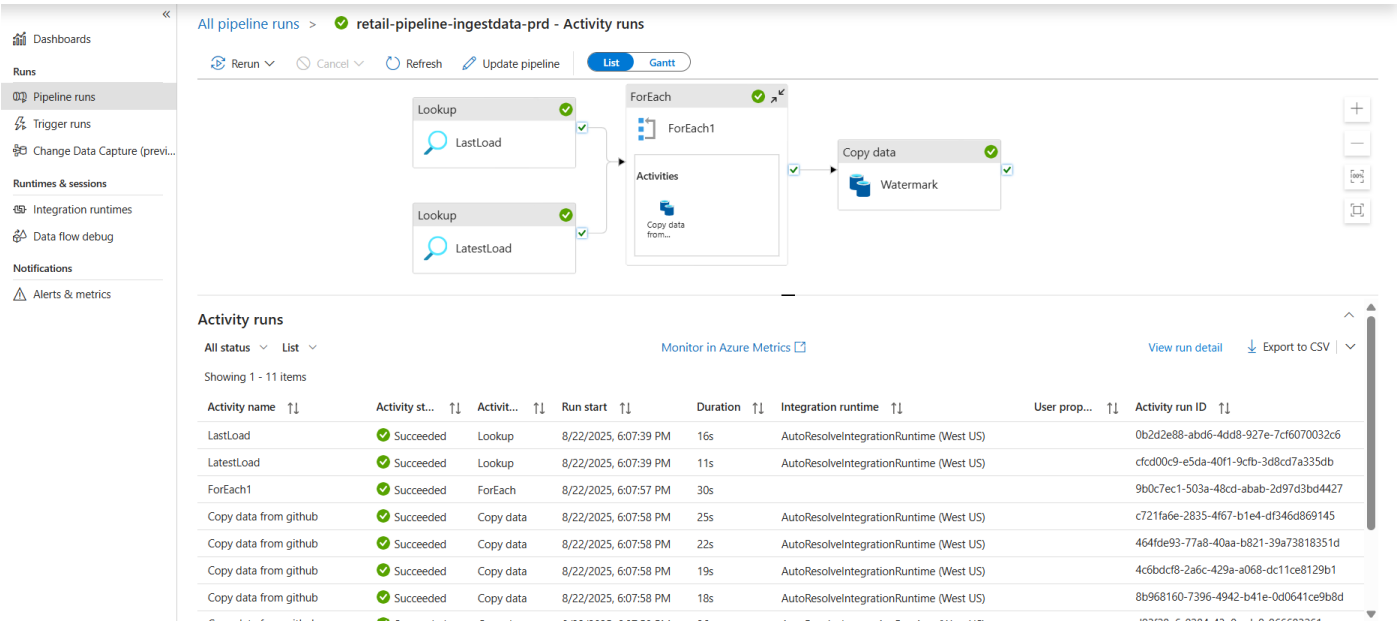
Give feedback

Azure Data Lake Storage (ADLS) Gen 2



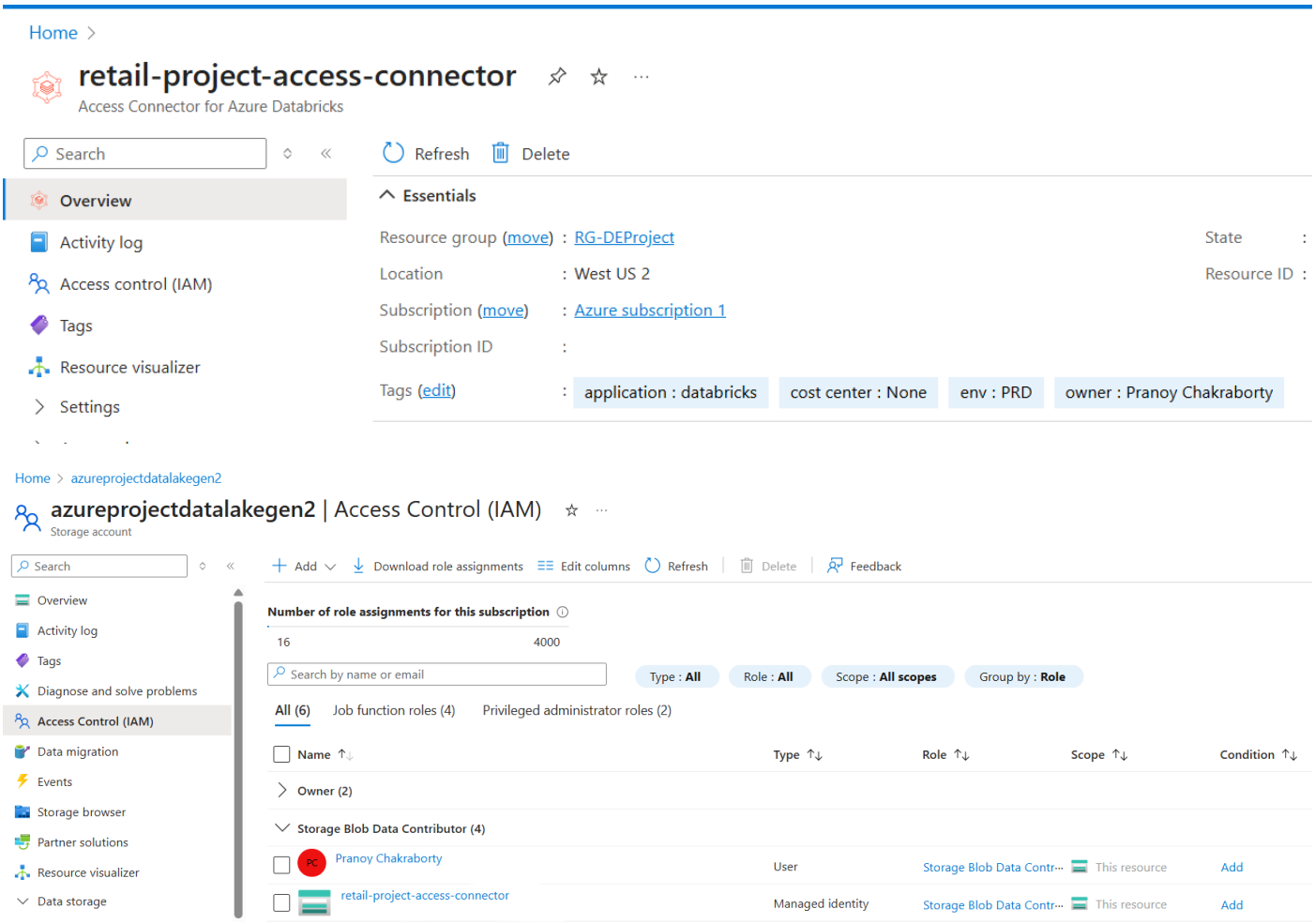
Azure Data Factory Pipeline – Data Ingestion + Incremental Data Loading

1. Use Lookup Activity to check the Last load value
2. Use another Lookup activity to check Latest Load value
3. Use Copy Activity nested in a ForEach Activity to copy data from GitHub or Any other source
4. Use Copy Activity to update the Last load value with Latest Load

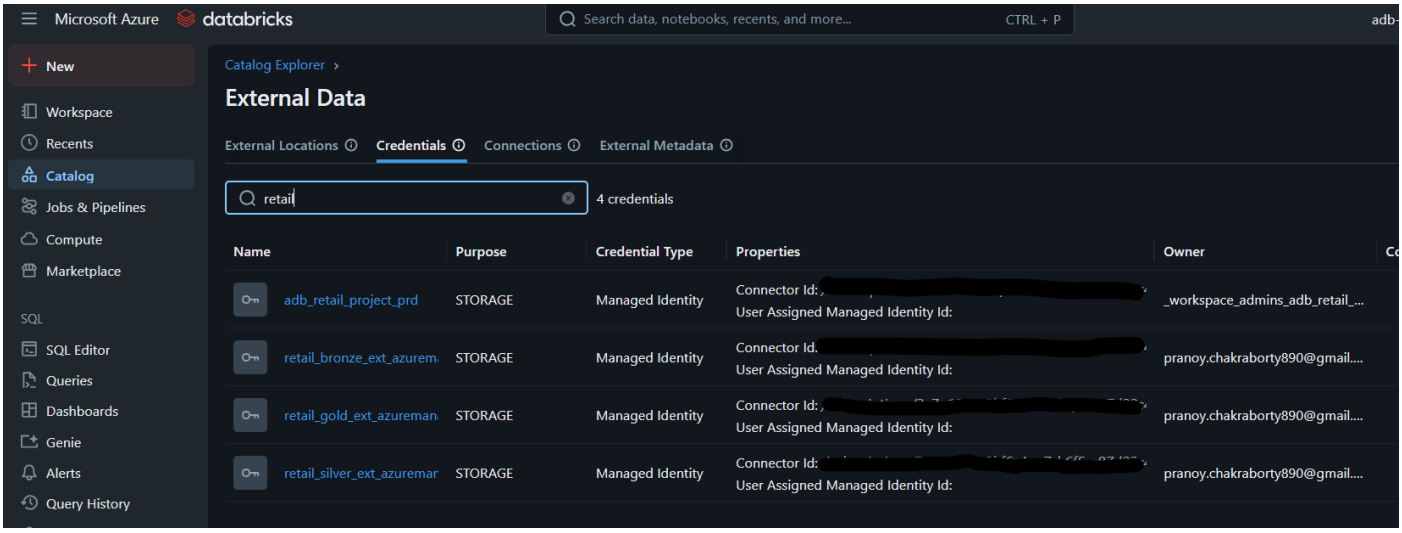


Azure Databricks Access Connector

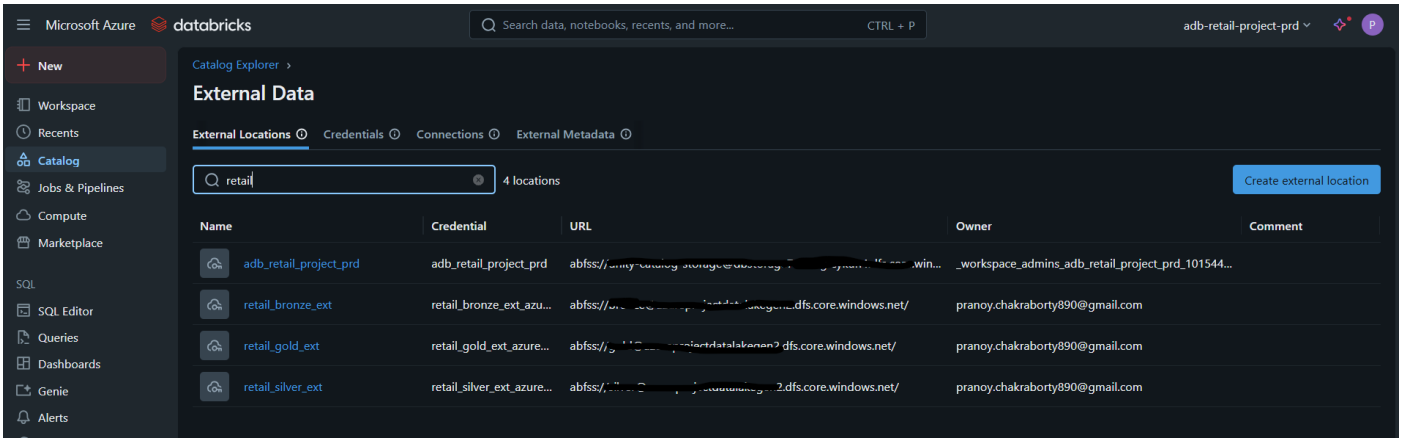
Azure Databricks Access Connector is required to access Azure Data Lake from Azure Databricks, without the access connector, Azure Databricks won't be able read / write data from / to Azure Databricks



Azure Databricks Credentials



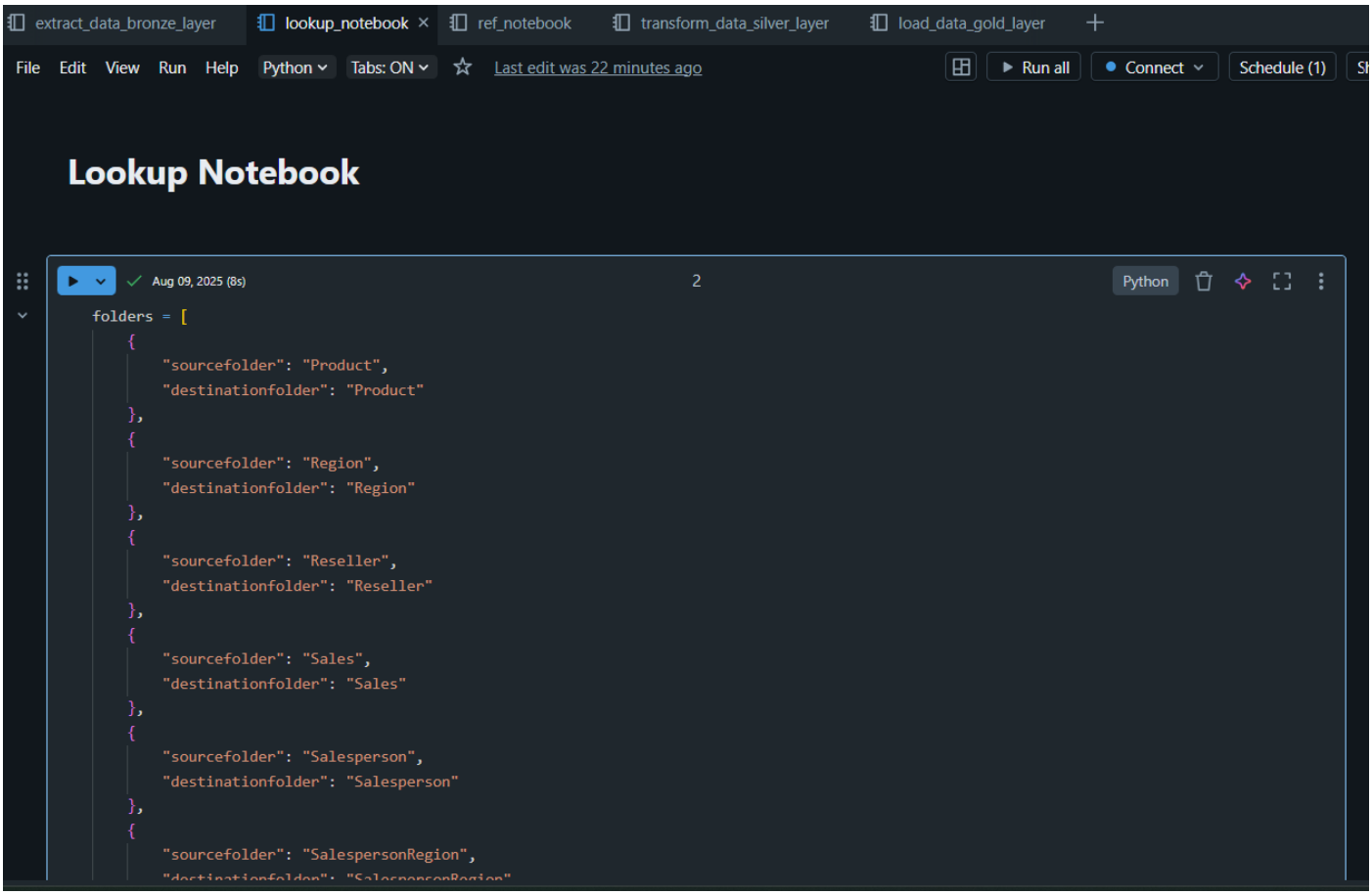
Azure Databricks External Location



ETL Pipeline in Azure Databricks

Lookup Notebook

The Lookup Notebook is used to set a list of files and folders. The output is used in Data Transformation from Bronze to Silver Layer.



Extract Data from Bronze Layer

Extract and Load data from Bronze layer to Silver layer:

extract_data_bronze_layer | lookup_notebook | ref_notebook | transform_data_silver_layer | load_data_gold_layer

File Edit View Run Help Python Tabs: ON Last edit was now Run all Connected Schedule (1)

destinationfolder: Product

sourcefolder: Product

Extract and load data from Bronze to Silver Layer

10:25 PM (5s) 2

```
print("Starting Extracting data from Bronze layer and ingest to silver layer ....")
```

```
Starting Extracting data from Bronze layer and ingest to silver layer ....
```

10:26 PM (<1s) 3

```
dbutils.widgets.text("sourcefolder", "Product")
dbutils.widgets.text("destinationfolder", "Product")
```

10:26 PM (<1s) 4

```
var_src_folder = dbutils.widgets.get("sourcefolder")
var_tgt_folder = dbutils.widgets.get("destinationfolder")
```

10:26 PM (17s) 5

Save to Silver layer as Delta format and save in Delta Table:

10:33 PM (4s) 7 Python

```
# df is your DataFrame
# Replace "/mnt/delta/fact_sales_table" with your desired path
df.write.format("delta").mode("overwrite").saveAsTable(f"netflix_data.netflix_schema.{var_tgt_folder}")
```

See performance (1) Optimize

Aug 09, 2025 (3s) 8

```
df.write.format("delta").mode("overwrite")\
    .option('overwriteSchema', 'true')\
    .option("path", f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/{var_tgt_folder}")\
    .save()
```

See performance (1) Optimize

Aug 09, 2025 (<1s) 9

```
print("Completed Extracting data from Bronze layer and ingest to silver layer ....")
```

```
Completed Extracting data from Bronze layer and ingest to silver layer ....
```

Data Transformation in Silver Layer

Product Data:

Product Data

Aug 10, 2025 (1s) 3

```
product_df = spark.read.format("delta")\
            .option("header", "true")\
            .option("inferSchema", "true")\
            .load(f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Product")
```

product_df: pyspark.sql.connect.dataframe.DataFrame = [ProductKey: integer, Product: string ... 4 more fields]

Aug 10, 2025 (2s) 4

```
product_df.printSchema()
product_df.display()
```

See performance (1) Optimize

```
root
 |-- ProductKey: integer (nullable = true)
 |-- Product: string (nullable = true)
```

Aug 10, 2025 (2s) 5

```
product_df = product_df.drop(col('BackgroundColorFormat'), col('FontColorFormat'))
product_df = product_df.withColumn('StandardCost', regexp_replace('StandardCost', "[$,]","").cast(DoubleType()))
product_df.printSchema()
product_df.display()
```

See performance (1) Optimize

product_df: pyspark.sql.connect.dataframe.DataFrame = [ProductKey: integer, Product: string ... 4 more fields]

```
root
 |-- ProductKey: integer (nullable = true)
 |-- Product: string (nullable = true)
 |-- StandardCost: double (nullable = true)
 |-- Color: string (nullable = true)
 |-- Subcategory: string (nullable = true)
 |-- Category: string (nullable = true)
```

Table +

	ProductKey	Product	StandardCost	Color	Subcategory	Category
1	210	HL Road Frame - Black, 58	868.63	Black	Road Frames	Components
2	215	Sport-100 Helmet, Black	12.03	Black	Helmets	Accessories
3	216	Sport-100 Helmet, Black	13.88	Black	Helmets	Accessories
4	217	Sport-100 Helmet, Black	13.09	Black	Helmets	Accessories
5	253	LL Road Frame - Black, 58	176.2	Black	Road Frames	Components
6	254	LL Road Frame - Black, 58	170.14	Black	Road Frames	Components

Region Data:

Region Data

Aug 10, 2025 (2s) 8

```
region_df = spark.read.format("delta")\
            .option("header", "true")\
            .option("inferSchema", "true")\
            .load(f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Region")
```

region_df: pyspark.sql.connect.dataframe.DataFrame = [SalesTerritoryKey: integer, Region: string ... 2 more fields]

Aug 10, 2025 (3s) 9

```
region_df.write.format("delta").mode("overwrite")\
    .option("path", f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Region")\
    .save()
```

See performance (1) Optimize

Reseller Data:

Reseller Data

Aug 10, 2025 (2s) 11

```
reseller_df = spark.read.format("delta")\
              .option("header", "true")\
              .option("inferSchema", "true")\
              .load(f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Reseller")
```

reseller_df: pyspark.sql.connect.dataframe.DataFrame = [ResellerKey: integer, BusinessType: string ... 4 more fields]

Aug 10, 2025 (3s) 12

```
reseller_df.write.format("delta").mode("overwrite")\
    .option("path", f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Reseller")\
    .save()
```

See performance (1) Optimize

Sales Data:

Sales Data

Aug 10, 2025 (1s) 14

```
sales_df = spark.read.format("delta")\
           .option("header", "true")\
           .option("inferSchema", "true")\
           .load(f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Sales")
```

sales_df: pyspark.sql.connect.dataframe.DataFrame = [SalesOrderNumber: string, OrderDate: date ... 8 more fields]

Aug 10, 2025 (2s) 15

```
sales_df.printSchema()
sales_df.display()
```

See performance (1) Optimize

```
root
 |-- SalesOrderNumber: string (nullable = true)
 |-- OrderDate: date (nullable = true)
 |-- ProductKey: integer (nullable = true)
 |-- ResellerKey: integer (nullable = true)
 |-- EmployeeKey: integer (nullable = true)
```

Aug 10, 2025 (2s) 16

```
sales_df = sales_df\
    .withColumn('UnitPrice', regexp_replace('UnitPrice', "[$,]",").cast(DoubleType()))\
    .withColumn('Sales', regexp_replace('Sales', "[$,]",").cast(DoubleType()))\
    .withColumn('Cost', regexp_replace('Cost', "[$,]",").cast(DoubleType()))\
    .withColumn('OrderDate', to_date(col('OrderDate'), 'dd/MM/yyyy'))

sales_df.printSchema()
sales_df.display()
```

See performance (1) Optimize

sales_df: pyspark.sql.connect.dataframe.DataFrame = [SalesOrderNumber: string, OrderDate: date ... 8 more fields]

```
root
 |-- SalesOrderNumber: string (nullable = true)
 |-- OrderDate: date (nullable = true)
 |-- ProductKey: integer (nullable = true)
 |-- ResellerKey: integer (nullable = true)
 |-- EmployeeKey: integer (nullable = true)
 |-- SalesTerritoryKey: integer (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- Sales: double (nullable = true)
 |-- Cost: double (nullable = true)
```

Salesperson Data:

Salesperson Data

Aug 10, 2025 (1s) 19

```
salesperson_df = spark.read.format("delta")\
            .option("header", "true")\
            .option("inferSchema", "true")\
            .load(f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Salesperson")
```

salesperson_df: pyspark.sql.connect.dataframe.DataFrame = [EmployeeKey: integer, EmployeeID: integer ... 3 more fields]

Aug 10, 2025 (3s) 20

```
salesperson_df.write.format("delta").mode("overwrite")\
    .option("path", f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Salesperson")\
    .save()
```

See performance (1) Optimize

Salesperson Region Data:

SalespersonRegion Data

Aug 10, 2025 (1s) 22

```
salespersonregion_df = spark.read.format("delta")\
            .option("header", "true")\
            .option("inferSchema", "true")\
            .load(f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/SalespersonRegion")
```

salespersonregion_df: pyspark.sql.connect.dataframe.DataFrame = [EmployeeKey: integer, SalesTerritoryKey: integer]

Aug 10, 2025 (3s) 23

```
salesperson_df.write.format("delta").mode("overwrite")\
    .option("path", f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Salesperson")\
    .save()
```

See performance (1) Optimize

Target Data:

Targets Data

Aug 10, 2025 (2s) 25

```
targets_df = spark.read.format("delta")\
            .option("header", "true")\
            .option("inferSchema", "true")\
            .load(f"abfss://silver@azureprojectdatalakegen2.dfs.core.windows.net/Targets")
```

targets_df: pyspark.sql.connect.dataframe.DataFrame = [EmployeeID: integer, Target: double ... 1 more field]

Aug 10, 2025 (3s) 26

```
targets_df.printSchema()
targets_df.display()
```

See performance (1) Optimize

```
root
 |-- EmployeeID: integer (nullable = true)
 |-- Target: double (nullable = true)
 |-- TargetDate: string (nullable = true)
```

Table +

Aug 10, 2025 (2s) 27

```
targets_df = targets_df.withColumnRenamed('TargetMonth', 'TargetDate')
targets_df = targets_df.withColumn('Target', regexp_replace('Target', "[$,]","").cast(DoubleType()))
targets_df.display()
```

See performance (1) Optimize

targets_df: pyspark.sql.connect.dataframe.DataFrame = [EmployeeID: integer, Target: double ... 1 more field]

Table +

	EmployeeID	Target	TargetDate
81	399771412	500000	01-11-2020
82	668991357	500000	01-02-2020
83	668991357	500000	01-03-2020
84	668991357	500000	01-12-2020
85	716374314	500000	01-12-2020
86	841560125	500000	01-03-2020
87	841560125	500000	01-07-2020
88	841560125	500000	01-08-2020
89	841560125	500000	01-09-2020
90	841560125	500000	01-10-2020
91	841560125	500000	01-11-2020
92	841560125	500000	01-12-2020
93	982310417	500000	01-01-2020
94	982310417	500000	01-02-2020
95	982310417	500000	01-03-2020

Data Aggregation and Serving in Gold Layer

Average cost and Total Cost based on Category and Subcategory:

1. Group by Category and subcategory
2. Use mean function for average and sum function for total
3. Order by Category and Subcategory

Aug 10, 2025 (1s) 6 Python

```python
product_df_aggregated = product_df.groupBy(["Category", "Subcategory"])\
    .agg(
        mean(col("StandardCost")).alias("AverageCost"),
        sum(col("StandardCost")).alias("TotalCost")
    )\
    .orderBy(["Category", "Subcategory"])
product_df_aggregated.display()
```

See performance (1) Optimize

product_df_aggregated: pyspark.sql.connect.dataframe.DataFrame = [Category: string, Subcategory: string ... 2 more fields]

Table +

	Category	Subcategory	AverageCost	TotalCost
1	Accessories	Bike Racks	44.88	44.88
2	Accessories	Bike Stands	59.47	59.47
3	Accessories	Bottles and Cages	2.99	8.97
4	Accessories	Cleaners	2.97	2.97
5	Accessories	Fenders	8.22	8.22
6	Accessories	Helmets	13	117
7	Accessories	Hydration Packs	20.57	20.57
8	Accessories	Lights	12.92	38.76
9	Accessories	Locks	10.31	10.31
10	Accessories	Panniers	51.56	51.56
11	Accessories	Pumps	9.280000000000001	18.5600000000000...
12	Accessories	Tires and Tubes	7.288181818181818	80.17

4. Save aggregated table as Delta Lake

Aug 10, 2025 (5s) 7

```python
product_df.write.format("delta").mode("overwrite")\
    .option("overwriteSchema", "true")\
    .option("path", f"abfss://gold@azureprojectdatalakegen2.dfs.core.windows.net/Product")\
    .save()

product_df_aggregated.write.format("delta").mode("overwrite")\
    .option("overwriteSchema", "true")\
    .option("path", f"abfss://gold@azureprojectdatalakegen2.dfs.core.windows.net/ProductAggregated")\
    .save()
```

See performance (2)

+ Code + Text Assistant

Top 5 Products as per category

- Use window function from PySpark.
- Define window spec using partition by and order by
- Use dense_rank() function for appropriate ranking and filter top 5 rows

Just now (1s) 7 Python

```python
window_spec = Window.partitionBy('Category').orderBy('TotalCost')

top5 = product_df_aggregated.withColumn("Rank", dense_rank().over(window_spec)).filter(col("Rank") <= 5)

top5.show()
```

See performance (1) Optimize

top5: pyspark.sql.connect.dataframe.DataFrame = [Category: string, Subcategory: string ... 3 more fields]

Category	Subcategory	AverageCost	TotalCost	Rank
Accessories	Cleaners	2.97	2.97	1
Accessories	Fenders	8.22	8.22	2
Accessories	Bottles and Cages	2.99	8.97	3
Accessories	Locks	10.31	10.31	4
Accessories	Pumps	9.280000000000001	18.560000000000002	5
Bikes	Touring Bikes	885.9327272727274	19490.520000000004	1
Bikes	Mountain Bikes	961.8855263157899	36551.650000000016	2
Bikes	Road Bikes	881.709692307693	57311.13000000004	3
Clothing	Socks	3.38	13.52	1
Clothing	Caps	5.953333333333333	17.86	2
Clothing	Vests	23.75	71.25	3
Clothing	Tights	30.929999999999996	92.78999999999999	4
Clothing	Gloves	11.513333333333334	103.62	5
Components	Chains	8.99	8.99	1
Components	Derailleurs	47.275	94.55	2
Components	Brakes	47.29	94.58	3
Components	Headsets	38.660000000000004	115.98	4
Components	Bottom Brackets	40.95333333333333	122.86	5

Count of cities per Country

Aug 10, 2025 (1s) 14

```python
reseller_df_aggregated = reseller_df.groupBy(col("CountryRegion"))\
    .agg(
        count(col("city")).alias("CityCount")
    )\
    .orderBy(col("CountryRegion"))
reseller_df_aggregated.display()
```

See performance (1) Optimize

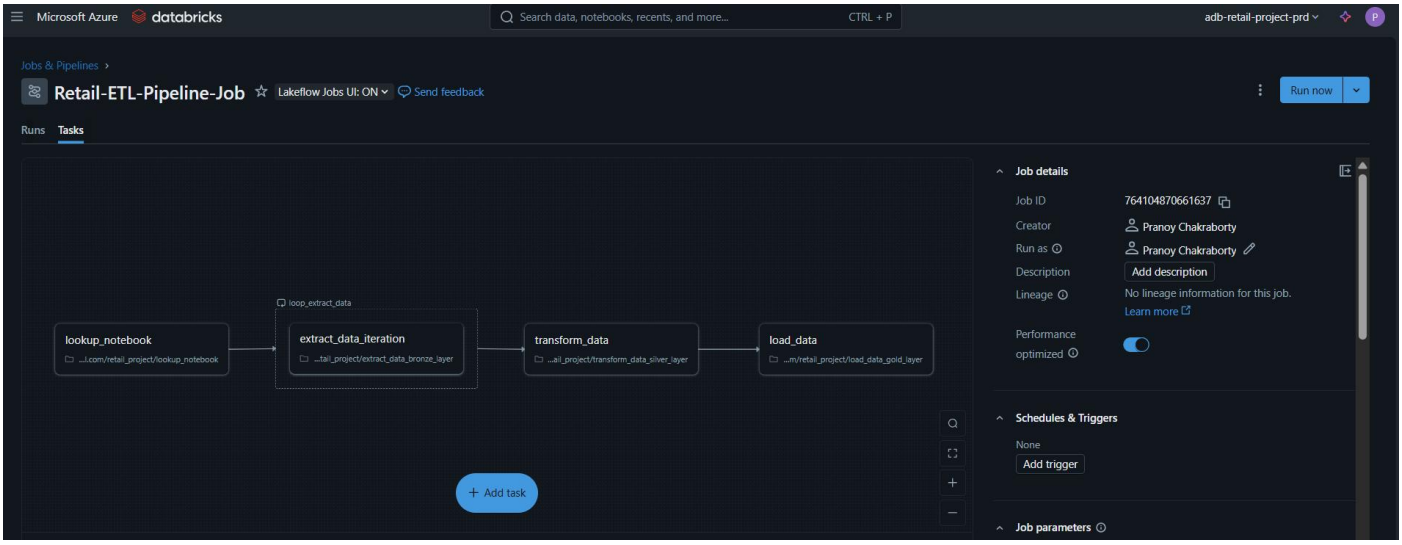
reseller_df_aggregated: pyspark.sql.connect.dataframe.DataFrame = [CountryRegion: string, CityCount: long]

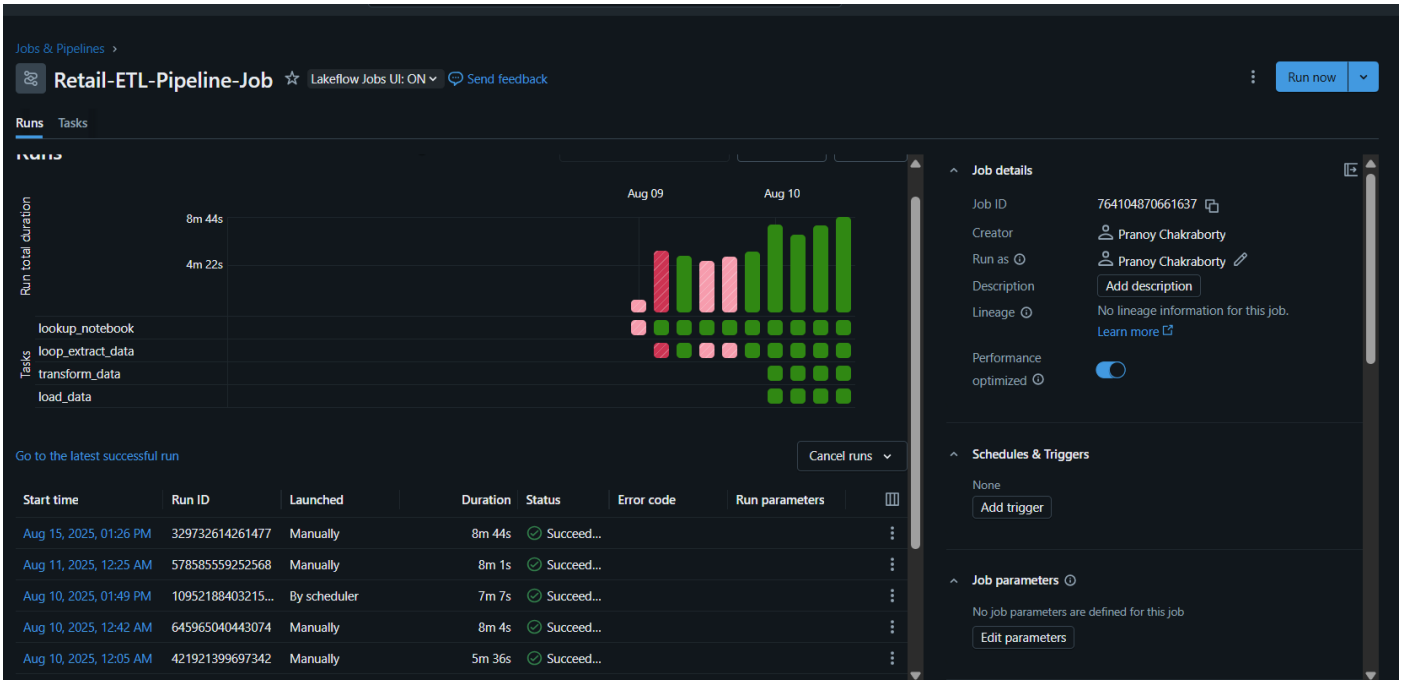
Table

	CountryRegion	CityCount
1	Australia	40
2	Canada	114
3	France	40
4	Germany	40
5	United Kingdom	40
6	United States	427

Retail-ETL-Pipeline-Job

Lakeflow Jobs UI





Job Started Notification:

[1015443569819756] Azure Databricks run 231903217517702 of Retail-ETL-Pipeline-Job has started Inbox ×

Microsoft Azure <azure-noreply@microsoft.com> 11:51 PM (1 minute ago)
to me

Microsoft Azure Azure Databricks

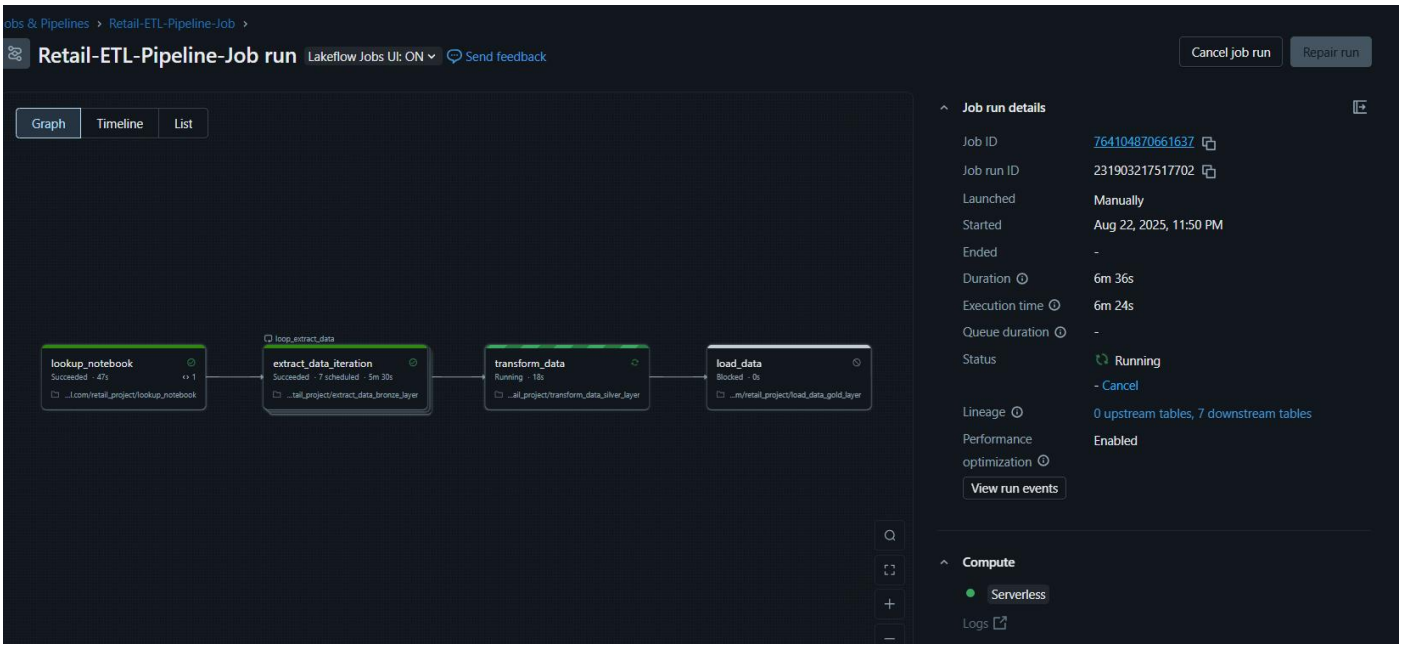
Your Azure Databricks job run has started

Run details

Workspace	adb-retail-project-prd (1015443569819756)
Job	Retail-ETL-Pipeline-Job (764104870661637)
Job run	231903217517702
Started at	2025-08-22 18:20:57 UTC
Launched	Manually

View run in Databricks >

Job is running:



Job Finished Notification:

[1015443569819756] Success—view run 231903217517702 of Retail-ETL-Pipeline-Job on Azure Inbox ×

Microsoft Azure <azure-noreply@microsoft.com>
to me

Fri, Aug 22, 11:59 PM (3 minutes ago)

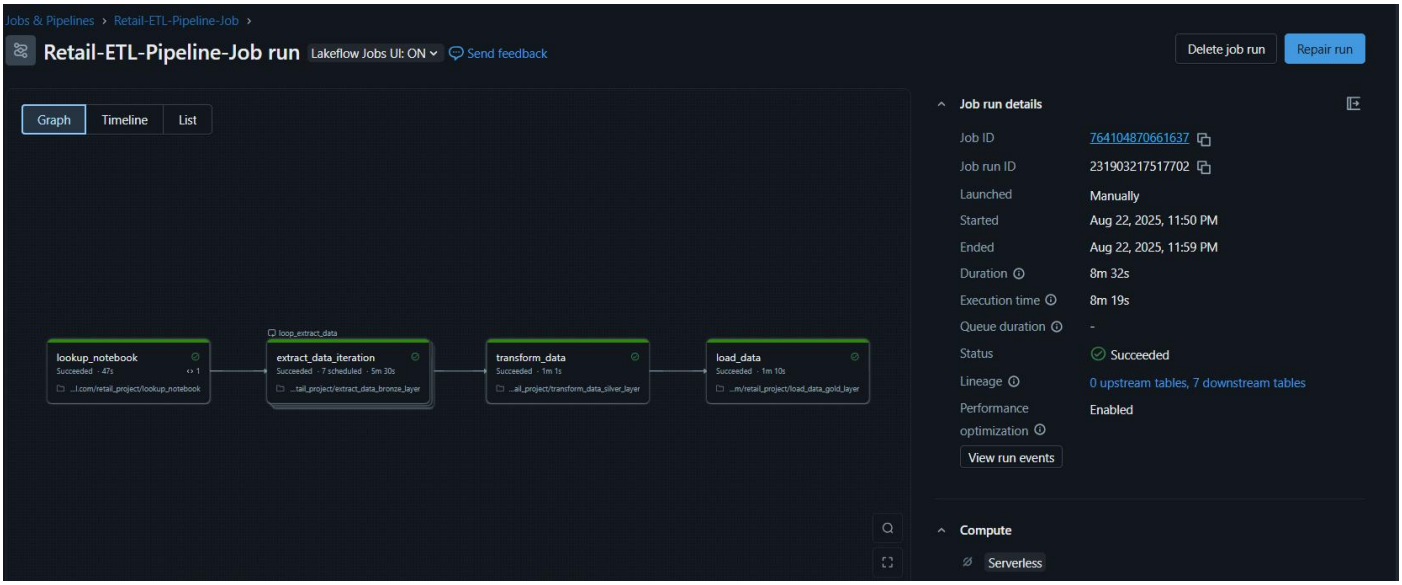
Microsoft Azure Azure Databricks

Your Azure Databricks job has finished its run

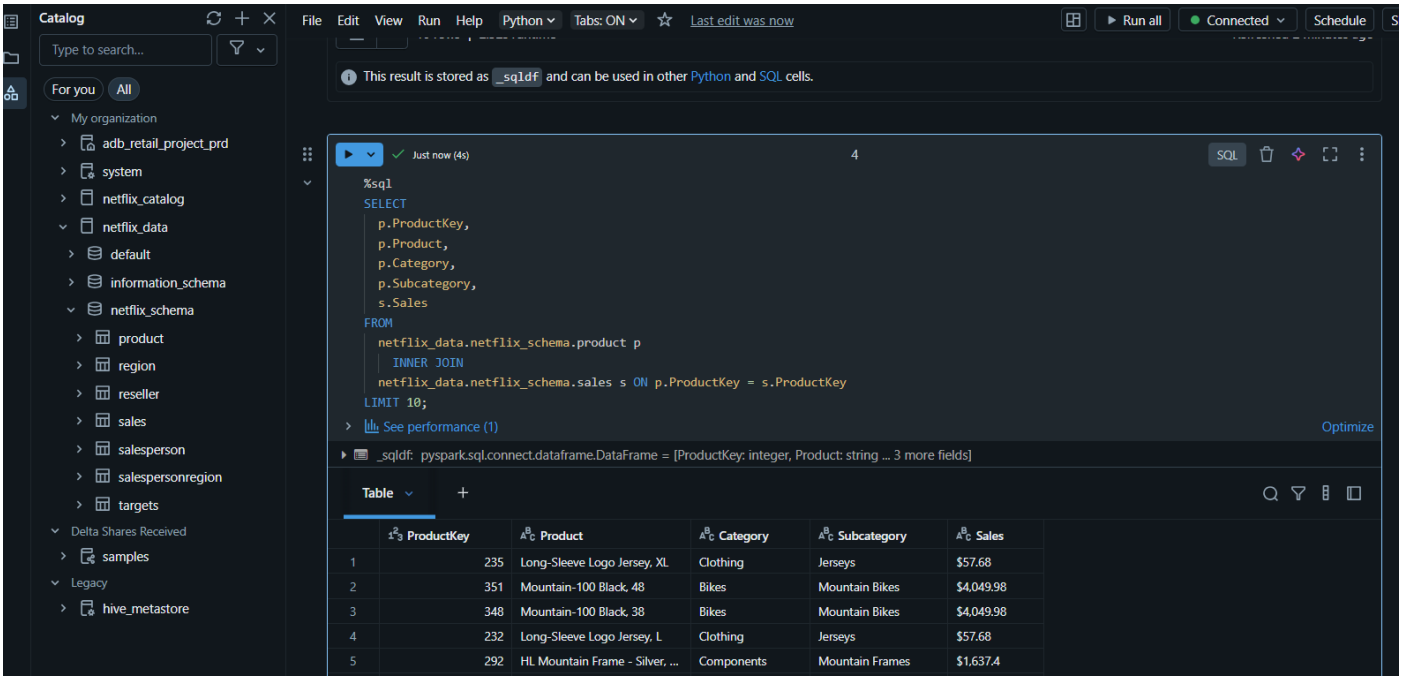
Run details

Workspace	adb-retail-project-prd (1015443569819756)
Job	Retail-ETL-Pipeline-Job (764104870661637)
Job run	231903217517702
Status	Success
Started at	2025-08-22 18:20:57 UTC
Duration	8m 31s
Launched	Manually

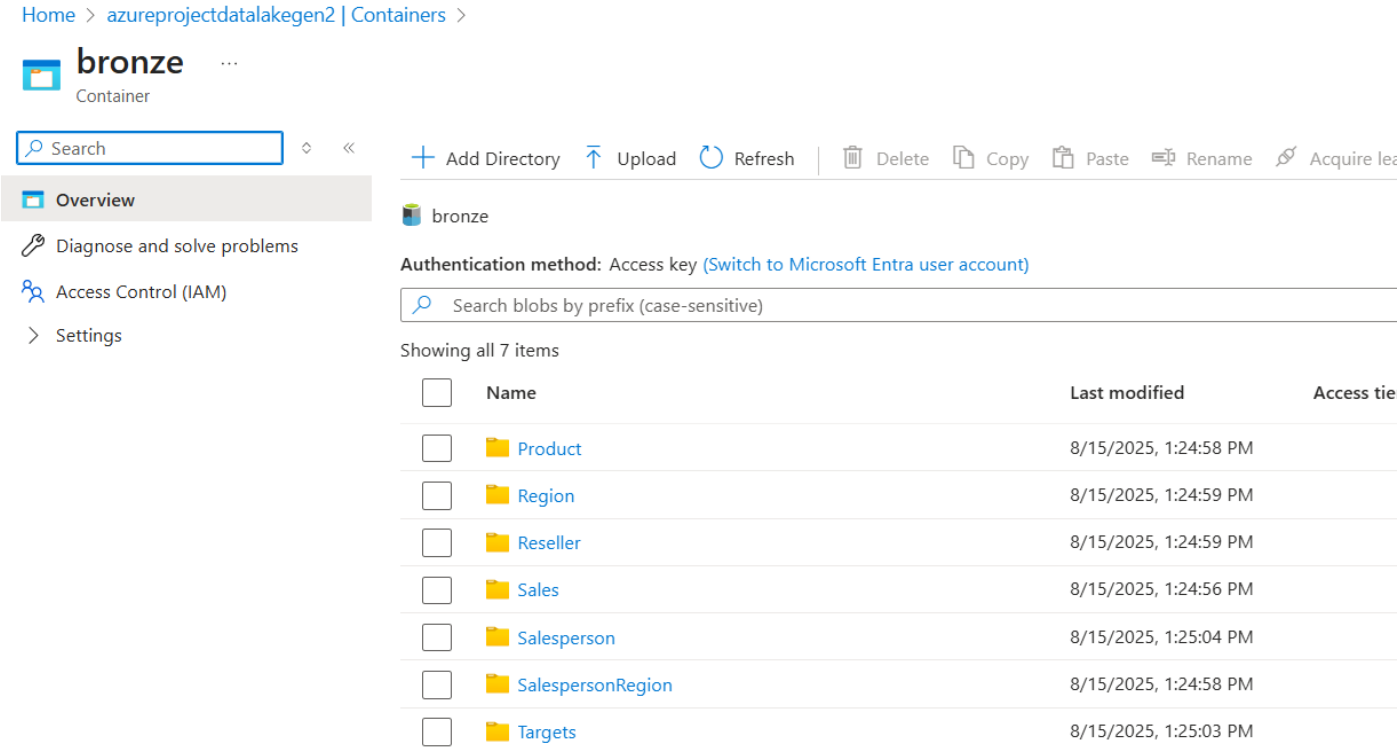
View run in Databricks >



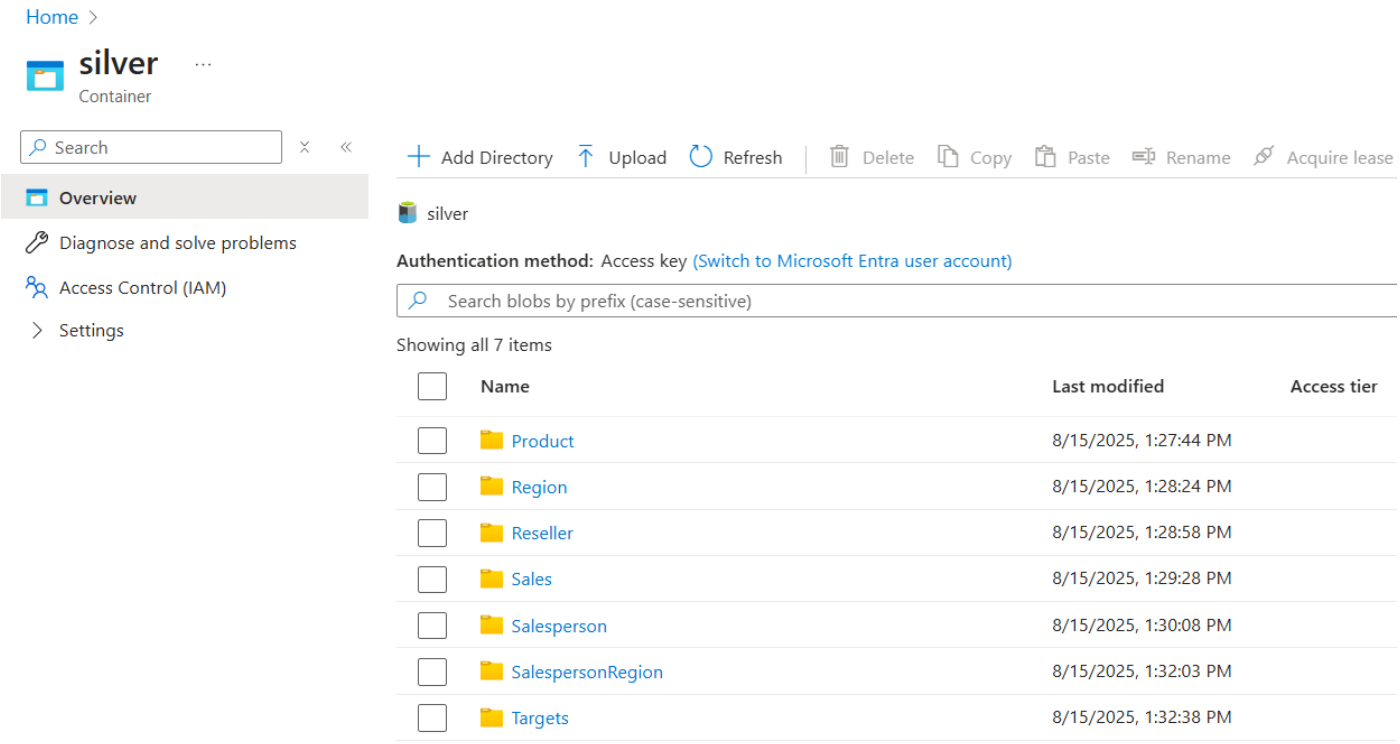
Delta Tables in Azure Databricks



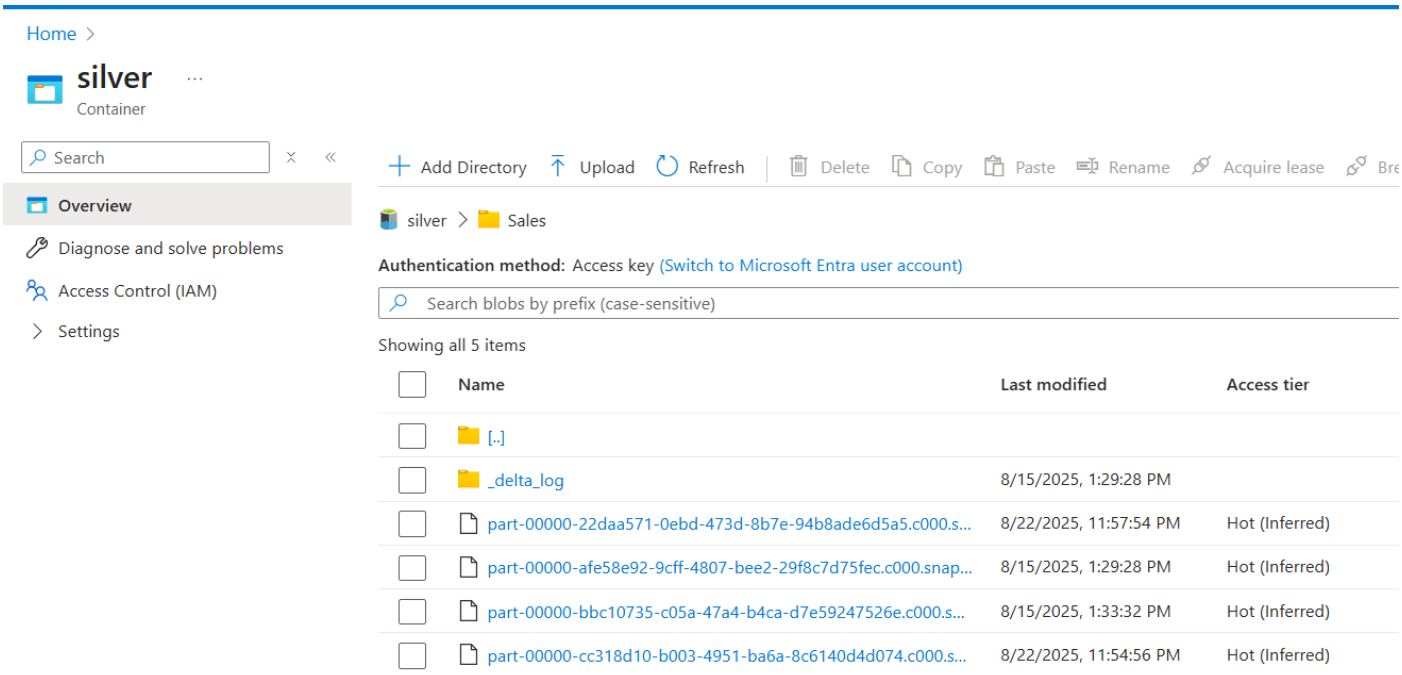
Raw data in Bronze Layer in Azure Data Lake



Cleaned, Enriched data stored in Silver Layer in Azure Data Lake



Data Stored in Delta Format



Aggregated, Business Ready data Stored in Gold Layer

Home > azureprojectdatalakegen2 | Containers >

gold
Container

Search

- Overview
- Diagnose and solve problems
- Access Control (IAM)
- Settings

Add Directory | Upload | Refresh | Delete | Copy | Paste | Rename | Acqui

gold

Authentication method: Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive)

Showing all 11 items

Name	Last modified	Access t
ProductAggregated	8/15/2025, 1:34:18 PM	
Region	8/15/2025, 1:34:21 PM	
Reseller	8/15/2025, 1:34:27 PM	
ResellerAggregated	8/15/2025, 1:34:29 PM	
Sales	8/15/2025, 1:34:37 PM	
SalesAggregated	8/15/2025, 1:34:39 PM	
Salesperson	8/15/2025, 1:34:42 PM	
SalespersonRegion	8/15/2025, 1:34:46 PM	
Targets	8/15/2025, 1:34:49 PM	
Top5	8/22/2025, 11:44:42 PM	

Add or remove favorites by pressing Ctrl+Shift+F

Home > azureprojectdatalakegen2 | Containers >

gold
Container

Search

- Overview
- Diagnose and solve problems
- Access Control (IAM)
- Settings

Add Directory | Upload | Refresh

gold > Top5

Authentication method: Access key (Switch to Micr

Search blobs by prefix (case-sensitive)

Showing all 3 items

Name
[..]
_delta_log
part-00000-0bbe3959-7e5b-4bbd-9e
part-00000-8e93df58-cf82-4ac7-971f

Add or remove favorites by pressing Ctrl+Shift+F

Top5/part-00000-0bbe3959-7e5b-4bbd-9e0e-e07c3f4e4ad0.c000.snappy.parquet
Blob

Save | Discard | Download | Refresh | Delete | Change tier | Acquire lease | Break lease | Give feedback

Overview | Versions | Edit | Generate SAS

Properties

URL	https://azureprojectdatal...
LAST MODIFIED	8/22/2025, 11:44:43 PM
CREATION TIME	8/22/2025, 11:44:43 PM
VERSION ID	-
TYPE	Block blob
SIZE	2.04 KiB
ACCESS TIER	Hot (Inferred)
ACCESS TIER LAST MODIFIED	N/A
ARCHIVE STATUS	-
REHYDRATE PRIORITY	-
SERVER ENCRYPTED	true
ETAG	0x8DDE1A7C4A32A68
VERSION-LEVEL IMMUTABILITY POLICY	Disabled
CACHE-CONTROL	
CONTENT-TYPE	application/octet-stream
CONTENT-MD5	
CONTENT-ENCODING	
CONTENT-LANGUAGE	
CONTENT-DISPOSITION	