

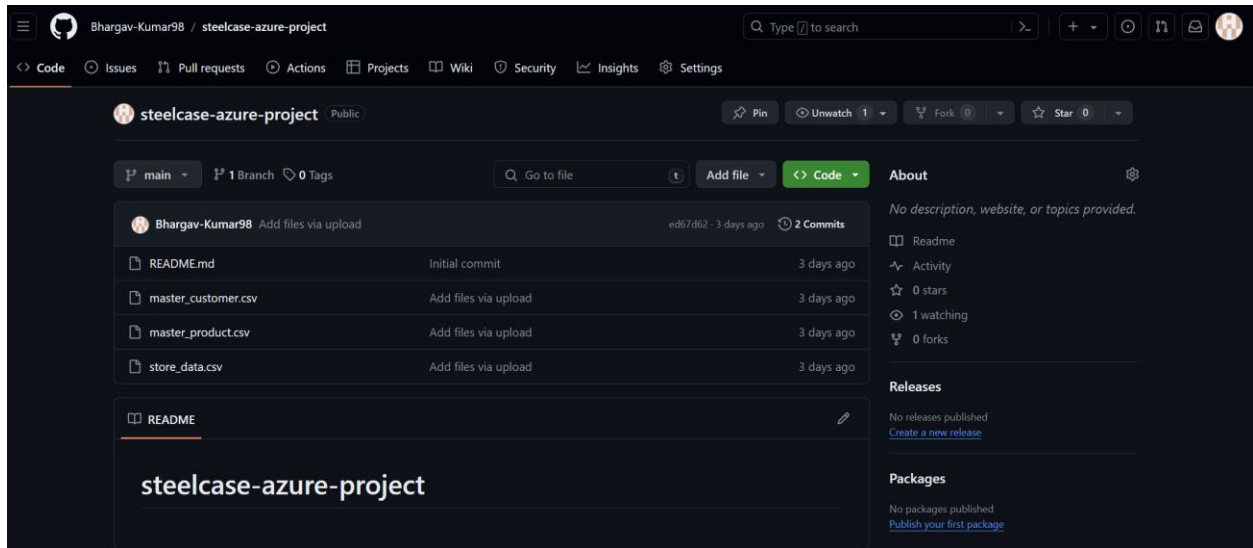
# Datawarehouse Final Project

## Stage 1: Data Ingestion

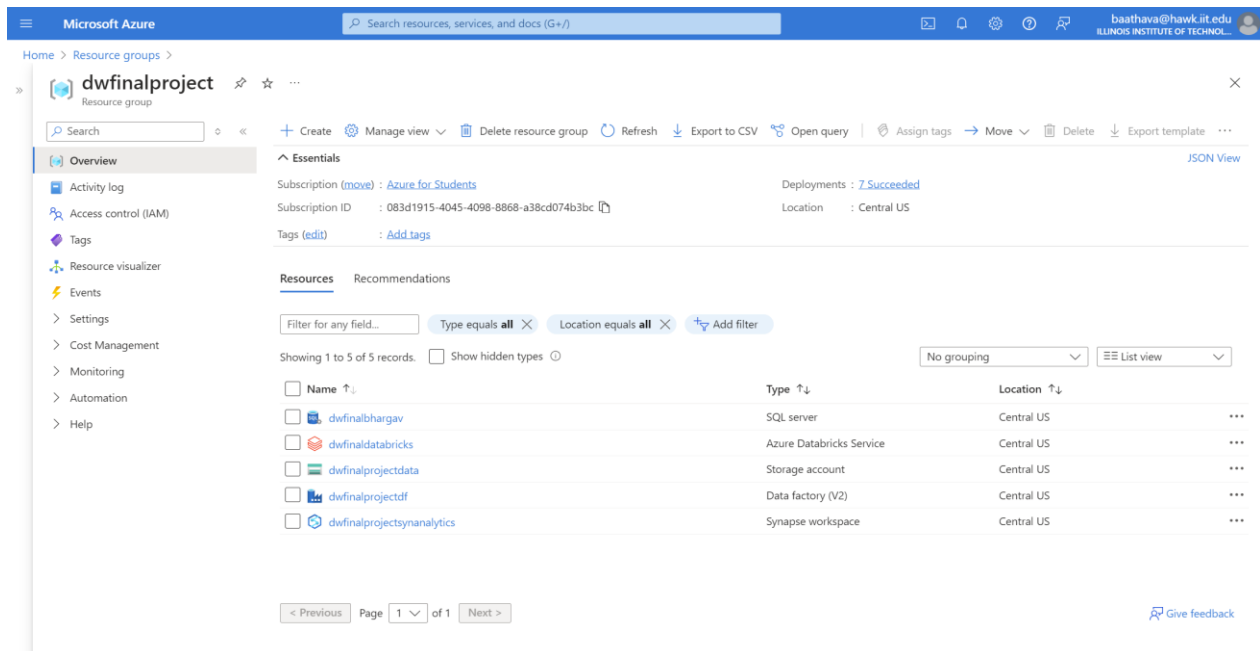
Dataset used for this project:

<https://www.kaggle.com/datasets/tforysyt/4-year-historical-sales-data/data>

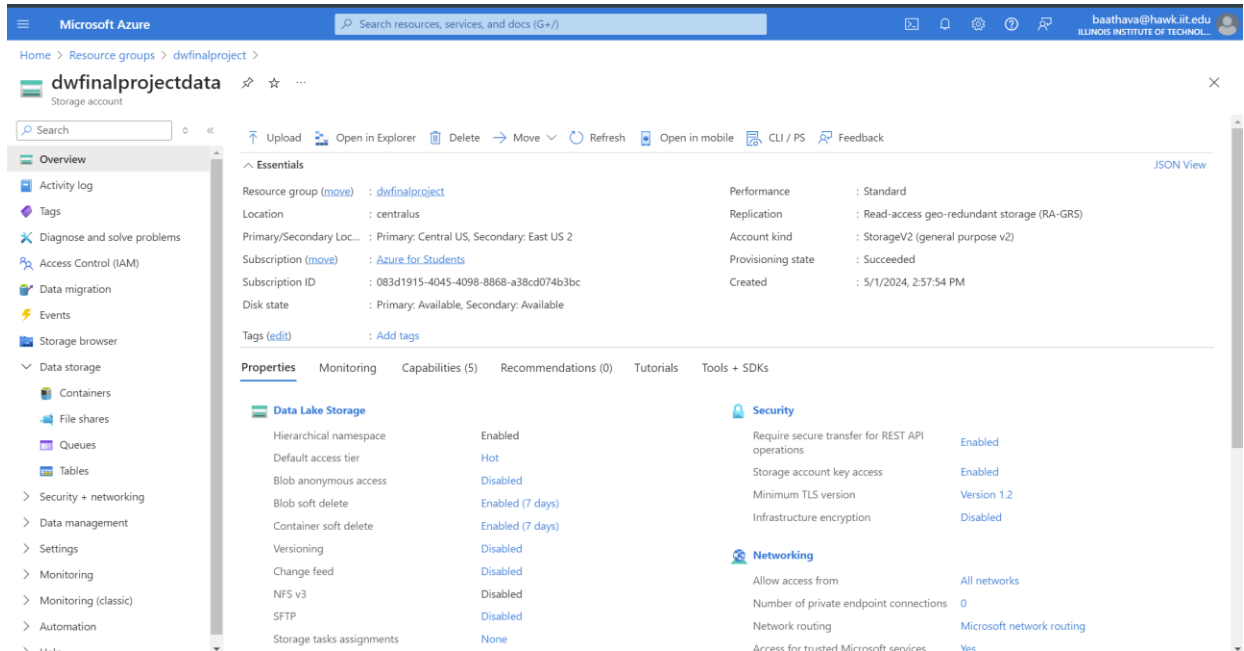
Initially, the dataset is uploaded to GitHub:



Created a resource group to centralize all resources, allowing for quick access and efficient administration. This resource group help me easily arrange and manage multiple resources, improving monitoring and access control for enhanced efficiency:

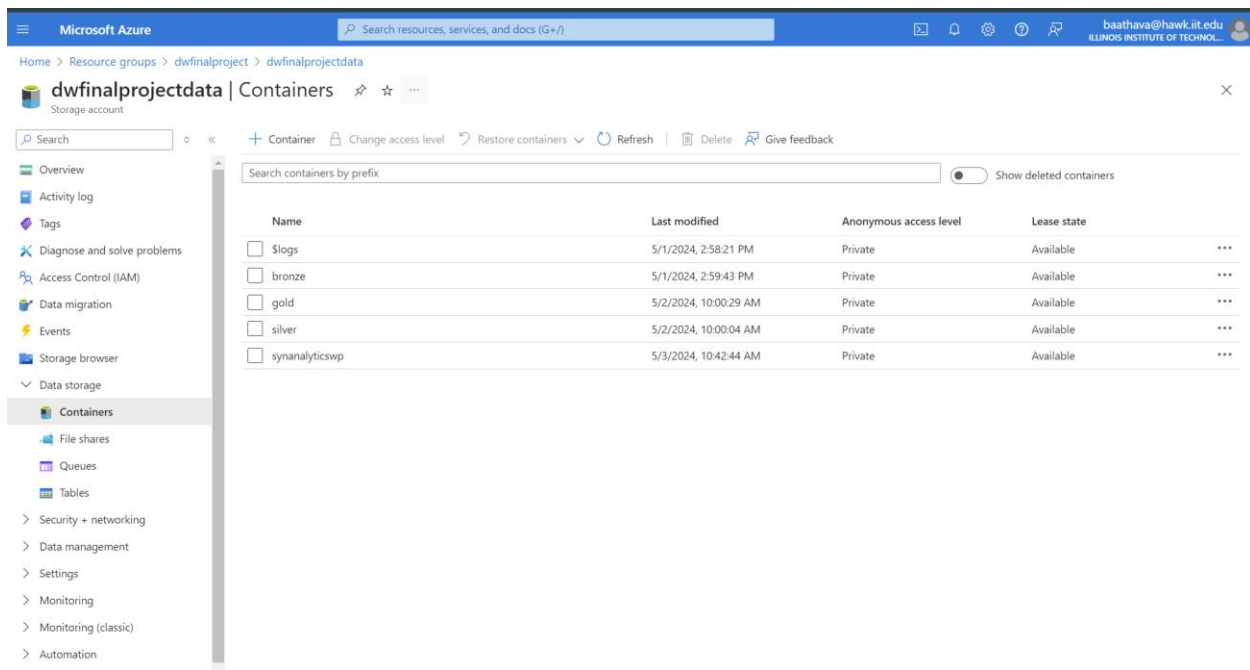


Initially Created an Azure storage account to store the data imported from GitHub. It provides scalable, durable, and highly accessible storage solutions, assuring the dependability and accessibility of stored data. This also enables easy integration of data pipelines and rapid data processing and analysis inside the Azure environment:

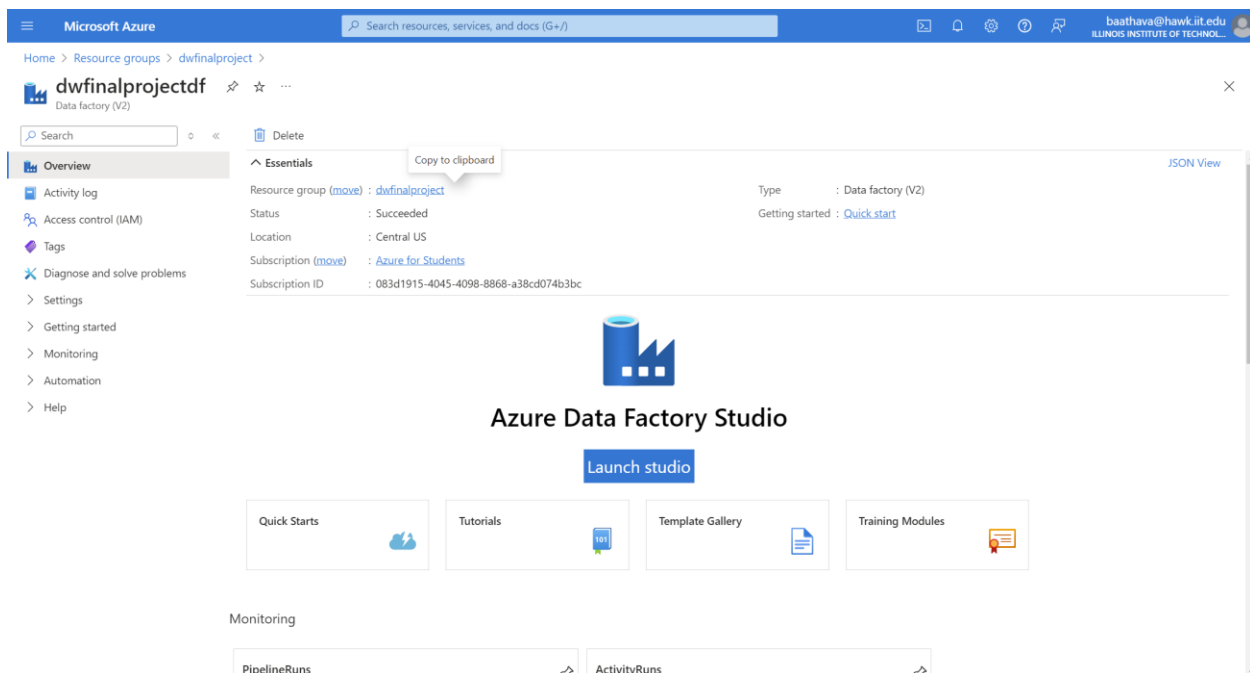


Created Bronze, Silver, Gold Containers Inside Storage account:

- Bronze – To store raw data to keep its original form and provide flexibility for future analysis.
- Silver – To assure data quality through validation and basic transformations and preparing it for downstream usage.
- Gold – To store finalized, optimized data, ready for use by end users, with improved performance for efficient analysis and decision-making.



Setting up a data factory, which is a fully managed data integration service that orchestrates and automates data transfer and transformation:



Created a pipeline in Azure Data Factory to seamlessly import data from GitHub and store it in the data lake created in the previous step. This pipeline orchestrates data transfer, making data ingestion operations more automated and efficient.

Microsoft Azure | Data Factory | dwfinalprojectdf

Search factory and documentation

baathava@hawk.iit.edu  
ILLINOIS INSTITUTE OF TECHNOLOGY

Validate all Publish all

Preview experience Off

Factory Resources

Filter resources by name

Pipelines 1

data-ingestion

Change Data Capture (preview) 0

Datasets 6

customers

customersink

products

productsink

store

storesink

Data flows 0

Power Query 0

data-ingestion

Validate Validate copy runtime Debug Add trigger

Copy data

Copy customer

Copy data

Copy product

Copy data

Copy store

General Source Sink Mapping Settings User properties

Source dataset \*

customers

Request method \*

GET

Additional headers

Request body

Request timeout

Files created in the bronze container after running the pipeline created in previous step:

Microsoft Azure | Data Factory | dwfinalprojectdf

Search factory and documentation

baathava@hawk.iit.edu  
ILLINOIS INSTITUTE OF TECHNOLOGY

Validate all Publish all

Preview experience Off

Factory Resources

Filter resources by name

Pipelines 1

data-ingestion

Change Data Capture (preview) 0

Datasets 6

customers

customersink

products

productsink

store

storesink

Data flows 0

Power Query 0

data-ingestion

Validate Debug Add trigger

Copy data

Copy customer

Copy data

Copy product

Copy data

Copy store

Parameters Variables Settings Output

Pipeline run ID: 5708e3a2-4995-4f5f-b472-60b0a6e42f6

Pipeline status Succeeded

View debug run consumption

All status

Showing 1 - 3 of 3 items

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Ac
Copy store	Succeeded	Copy data	5/1/2024, 10:23:06 PM	12s	AutoResolveIntegration		2f
Copy product	Succeeded	Copy data	5/1/2024, 10:22:51 PM	15s	AutoResolveIntegration		42
Copy customer	Succeeded	Copy data	5/1/2024, 10:22:39 PM	12s	AutoResolveIntegration		d3

bronze

Container

Search

o

<

- Upload
- Add Directory
- Refresh
- Rename
- Delete
- Change tier
- Acquire lease
- Break lease
- Give feedback

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

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

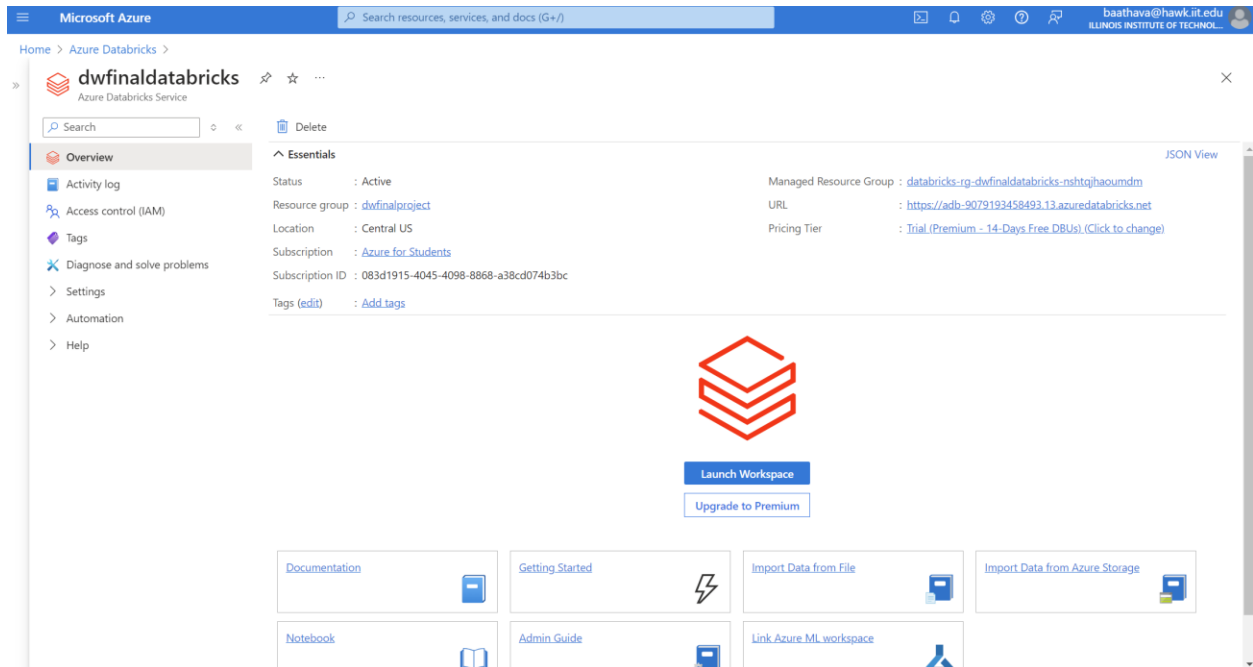
Search blobs by prefix (case-sensitive)

Show deleted objects

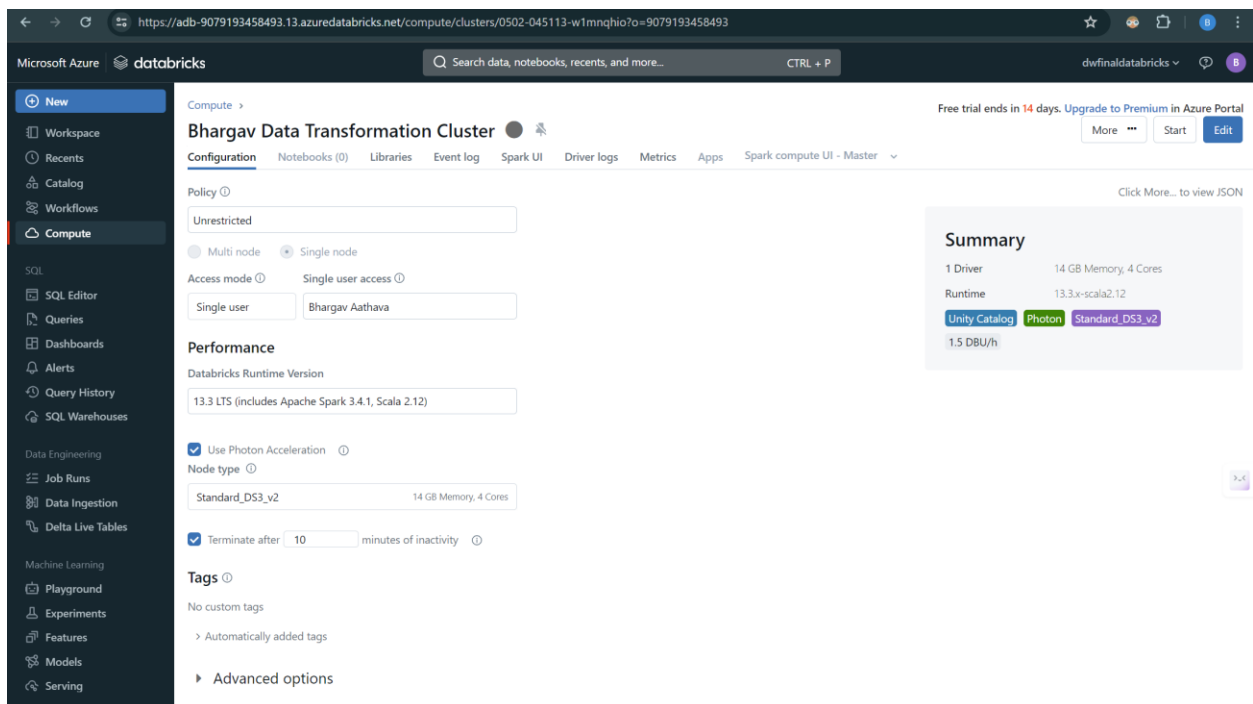
	Name	Modified	Access tier	Archive status	Blob type	Size	Lease state	
<input type="checkbox"/>	customer.csv	5/3/2024, 11:41:25 AM	Hot (Inferred)		Block blob	64.04 KiB	Available	...
<input type="checkbox"/>	products.csv	5/3/2024, 11:41:37 AM	Hot (Inferred)		Block blob	141.15 KiB	Available	...
<input type="checkbox"/>	stores.csv	5/3/2024, 11:41:57 AM	Hot (Inferred)		Block blob	883.44 KiB	Available	...

## Stage 2: Data Ingestion

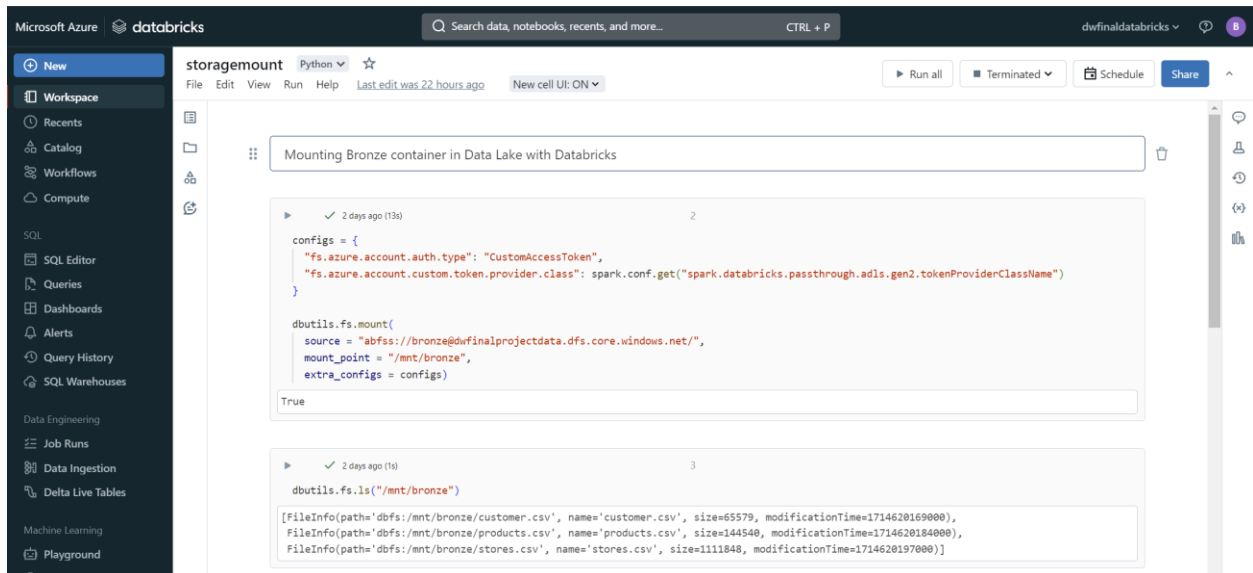
Created an Azure Databricks service to help with the transformation process, ensuring that raw data in the bronze layer is cleaned and ready for further analysis. Using Azure Databricks for transformation because it offers faster data transformation operations:



Created the Cluster in data bricks to utilize computational resources to run the notebooks:



Mounting Databricks with all the containers we created in Azure's data lake to enable seamless data access and integration:



The screenshot shows a Databricks workspace titled 'storagemount' with a Python notebook. The notebook is titled 'Mounting Bronze container in Data Lake with Databricks'. It contains two code cells. The first cell defines configurations for mounting a bronze container and then mounts it. The second cell lists the files in the mounted container.

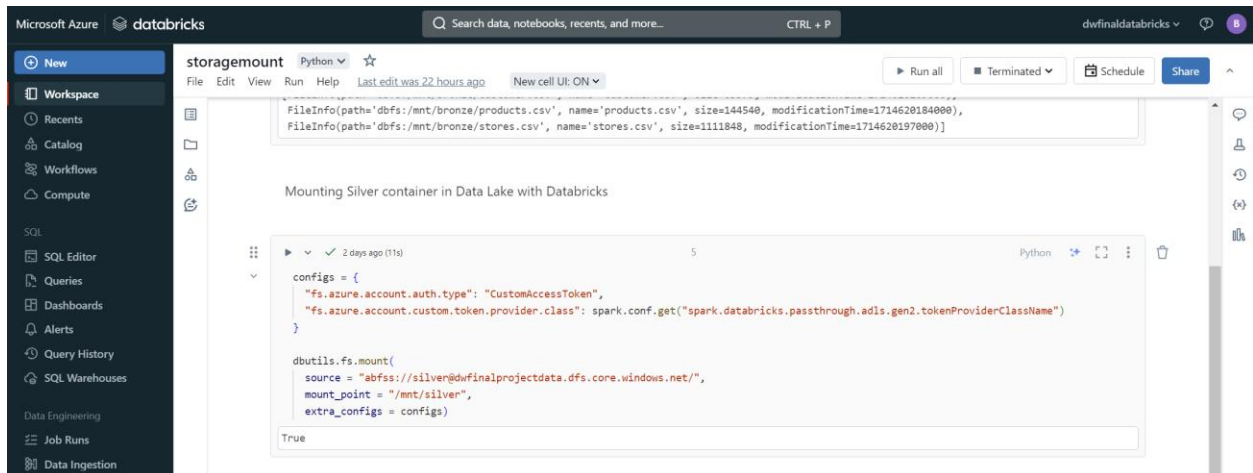
```
configs = {
    "fs.azure.account.auth.type": "CustomAccessToken",
    "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

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

True
```

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

[FileInfo(path='dbfs:/mnt/bronze/customer.csv', name='customer.csv', size=65579, modificationTime=1714620169000),
 FileInfo(path='dbfs:/mnt/bronze/products.csv', name='products.csv', size=144540, modificationTime=1714620184000),
 FileInfo(path='dbfs:/mnt/bronze/stores.csv', name='stores.csv', size=1111848, modificationTime=1714620197000)]
```



The screenshot shows a Databricks workspace titled 'storagemount' with a Python notebook. The notebook is titled 'Mounting Silver container in Data Lake with Databricks'. It contains one code cell that defines configurations for mounting a silver container and then mounts it.

```
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://silver@dwfinalprojectdata.dfs.core.windows.net/",
    mount_point = "/mnt/silver",
    extra_configs = configs)

True
```

Microsoft Azure databricks

Search data, notebooks, recents, and more... CTRL + P dwfinaldatabricks

New Workspace Recents Catalog Workflows Compute SQL SQL Editor Queries Dashboards Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Delta Live Tables Machine Learning Playground Experiments Features Models Serving

storageamount Python 2 days ago (11s) 5 Run all Terminated Schedule Share

```
source = "abfss://silver@dwfinalprojectdata.dfs.core.windows.net/",
mount_point = "/mnt/silver",
extra_configs = configs)

True
```

Mounting Gold container in Data Lake with Databricks

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

True
```

[Shift+Enter] to run and move to next cell  
[Esc H] to see all keyboard shortcuts

Created a notebook 'Bronze to Silver' to perform data transformations on the data in 'bronze' container and then store the transformed data in the 'silver' container:

Microsoft Azure databricks

Search data, notebooks, recents, and more... CTRL + P dwfinaldatabricks

New Workspace Recents Catalog Workflows Compute SQL SQL Editor Queries Dashboards Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Delta Live Tables Machine Learning Playground Experiments Features Models Serving

bronze to silver Python 2 days ago (7s) 43 Run all Terminated Schedule Share

5	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO/20335	OFF-ST-10000760	335520
6	CA-2015-115812	2015-06-09	2015-06-14	Standard Class	BH/11710	FUR-FU-10001487	732900
7	CA-2015-115812	2015-06-09	2015-06-14	Standard Class	BH/11710	OFF-AR-10002833	109200
8	CA-2015-115812	2015-06-09	2015-06-14	Standard Class	BH/11710	TEC-PH-10002275	13607280
9	CA-2015-115812	2015-06-09	2015-06-14	Standard Class	BH/11710	OFF-BI-10003910	277560
10	CA-2015-115812	2015-06-09	2015-06-14	Standard Class	BH/11710	OFF-AP-10002892	1723500
11	CA-2015-115812	2015-06-09	2015-06-14	Standard Class	BH/11710	FUR-TA-10001539	25592760
12	CA-2015-115812	2015-06-09	2015-06-14	Standard Class	BH/11710	TEC-PH-10002033	13671360
13	CA-2018-114412	2018-04-15	2018-04-20	Standard Class	AA/10480	OFF-PA-10002365	233280
14	CA-2017-161389	2017-12-05	2017-12-10	Standard Class	IM/15070	OFF-BI-10003656	6119640

9,770 rows | 0.59 seconds runtime Refreshed 2 days ago

Saving cleaned data in the silver container

```
dfs = [df_customer, df_product, df_store]
df_names = ["df_customer", "df_product", "df_store"]

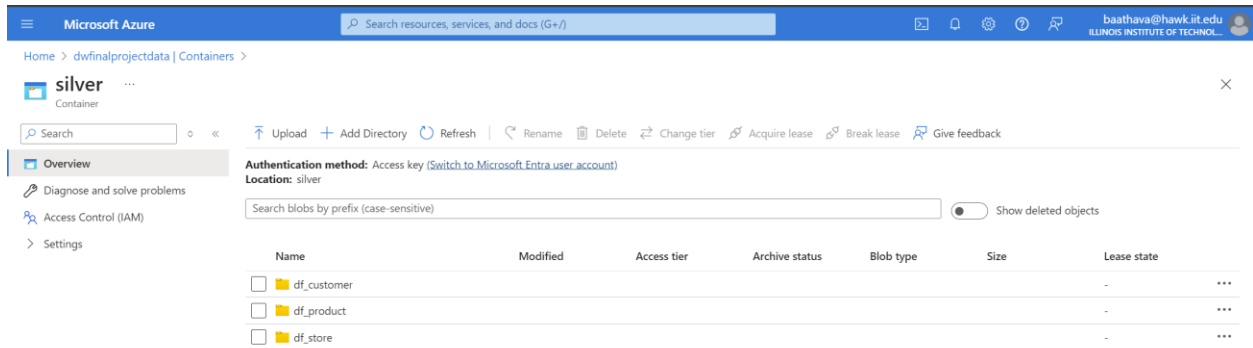
for i, df in enumerate(dfs, start=1):
    output_path = '/mnt/silver/' + df_names[i-1] + '/'
    df.write.format('delta').mode("overwrite").save(output_path)
```

(12) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [Order\_ID: string, Order\_Date: date ... 6 more fields]



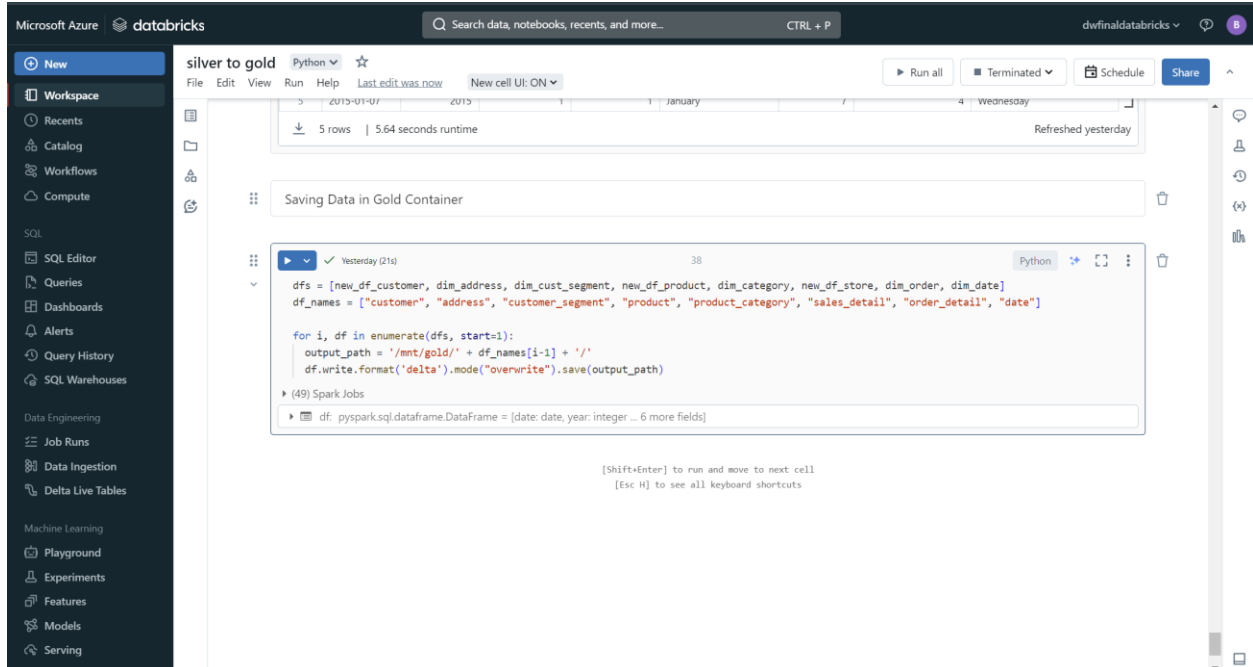
## Result after running Bronze to Silver notebook:



The screenshot shows the Microsoft Azure portal interface for a container named 'silver'. The container is located in the 'dwtinalprojectdata' resource group. The authentication method is 'Access key' and the location is 'silver'. The container contains three blobs: 'df\_customer', 'df\_product', and 'df\_store'. The table below shows the details of these blobs.

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
df_customer						-
df_product						-
df_store						-

Created another notebook 'Silver to Gold' to create facts and dimensions to design data warehouse and then store the transformed datasets in the 'Gold' container:



The screenshot shows the Databricks notebook interface for a notebook named 'silver to gold'. The notebook is written in Python and contains a code cell that saves data to the 'Gold' container. The code is as follows:

```
dfs = [new_df_customer, dim_address, dim_cust_segment, new_df_product, dim_category, new_df_store, dim_order, dim_date]
df_names = ["customer", "address", "customer_segment", "product", "product_category", "sales_detail", "order_detail", "date"]

for i, df in enumerate(dfs, start=1):
    output_path = '/mnt/gold/' + df_names[i-1] + '/'
    df.write.format('delta').mode("overwrite").save(output_path)
```

The code cell is titled 'Saving Data in Gold Container' and has a status of 'Yesterday (21s)'. The output of the code cell is a Spark job that has completed successfully.

## Result after running Silver to Gold notebook:

Microsoft Azure

Search resources, services, and docs (G+)

Home > dwfinalprojectdata | Containers >

gold

Container

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

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

Location: gold

Search blobs by prefix (case-sensitive)

Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
address						-
customer						-
customer_segment						-
date						-
order_detail						-
product						-
product_category						-
sales_detail						-

Integrating data bricks Notebooks into the pipeline we previously developed in Data Factory allows us to conduct sophisticated data transformations smoothly within the current pipeline, streamlining data processing.

Microsoft Azure

Data Factory > dwfinalprojectdf

Search factory and documentation

Validate all Publish all

Preview experience Off

Factory Resources

Filter resources by name

Pipelines

data-ingestion

Change Data Capture (preview)

Datasets

customers

customersink

products

productsink

store

storesink

Data flows

Power Query

data-ingestion

Validate Debug Add trigger

Trigger test runs of the current pipeline without publishing your changes to the service

Copy data

Copy customer

Copy data

Copy product

Copy data

Copy store

Notebook

Bronze To Silver

Notebook

Silver To Gold

General Azure Databricks Settings User properties

Databricks linked service

AzureDatabricks1

Test connection Edit New

## Stage 3: Data Loading

Finally, I created a serverless SQL database in Azure Synapse Analytics to store views generated from final tables saved in gold containers. The serverless design of Azure Synapse Analytics enables efficient querying and analysis of stored data when we are using the data for reporting purposes.

The screenshot shows the Microsoft Azure portal interface for a Synapse workspace. The top navigation bar includes the Microsoft Azure logo, a search bar, and user information. The main content area displays the workspace overview, including essential details like Resource group, Status, Location, Subscription, and various endpoints. A 'Getting started' section offers links to Open Synapse Studio and Read documentation. Below this, there's a section for Analytics pools with a search filter and a table header for Name, Type, and Size.

Name	Type	Size
SQL pools		

Created a storage procedure to create the views:

The screenshot shows the Synapse Studio interface with a SQL script editor open. The script is titled 'serverlessview\_sp' and is designed to create or alter a serverless view. It uses a dynamic SQL approach, constructing a CREATE OR ALTER VIEW statement based on the view name provided as a parameter. The script includes a DECLARE statement for the dynamic SQL text, a SET statement to build the query, and an EXEC statement to run the query. The script is set to run against the 'final\_db' database.

```
1 USE final_db
2 GO
3
4 CREATE OR ALTER PROC CreateSQLServerlessView @ViewName nvarchar(100)
5 AS
6 BEGIN
7
8 DECLARE @Statement VARCHAR(MAX)
9 SET @Statement = N'CREATE OR ALTER VIEW ' + @ViewName + ' As
10 SELECT
11     *
12 FROM
13     OPENROWSET(
14         BULK ''https://dwfinalprojectdata.dfs.core.windows.net/gold/' + @ViewName + '/',
15         FORMAT = ''DELTA''
16     ) AS [result]
17
18 EXEC (@Statement)
19
20 END
21 GO
```

Created new pipeline in Synapse analytics to create view:

Microsoft Azure | Synapse Analytics | dwfinalprojectsynanalytics

We use optional cookies to provide a better experience. [Learn more](#)

Accept Reject More options

Analytics pools

- SQL pools
- Apache Spark pools
- Data Explorer pools (preview)

Activities

- SQL requests
- KQL requests
- Apache Spark applications
- Data flow debug

Integration

- Pipeline runs
- Trigger runs
- Integration runtimes
- Link connections

Rerun Cancel Refresh Update pipeline List Gantt

Get Metadata (Get TableNames) → ForEach (ForEachTableName) → Activities (Stored procedure...)

Activity	Status	Type	Start Time	Duration	Integration	Run ID
Get TableNames	Succeeded	Get Metadata	5/3/2024, 11:49:54 AM	2s	AutoResolveIntegration	0229eb03-3bd
ForEachTableName	Succeeded	ForEach	5/3/2024, 11:49:57 AM	20s	AutoResolveIntegration	511dcb01-52c
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	4s	AutoResolveIntegration	3f4a9d8a-f66f-
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	14s	AutoResolveIntegration	92ffac86-c50f-
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	3s	AutoResolveIntegration	c8c0904f-157f-
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	3s	AutoResolveIntegration	5db418a3-5b6
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	3s	AutoResolveIntegration	03011185-c44f-
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	4s	AutoResolveIntegration	66c74e87-87c2
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	4s	AutoResolveIntegration	2f9dd5f7-e050
Stored procedure1	Succeeded	Stored procedure	5/3/2024, 11:49:58 AM	17s	AutoResolveIntegration	57fbc60-a0fd-

Result after loading data in SQL database created on Synapse Analytics:

Microsoft Azure | Synapse Analytics | dwfinalprojectsynanalytics

We use optional cookies to provide a better experience. [Learn more](#)

Accept Reject More options

Synapse live Validate all Publish all

Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.

SQL script 1

Run Undo Publish Query plan Connect to Built-in Use database final\_db

```
1 SELECT TOP (100) [Address_ID]
2 , [Country]
3 , [Region]
4 , [State]
5 , [City]
6 , [Postal_Code]
7 FROM [dbo].[address]
```

Results Messages

View Table Chart Export results

Address_ID	Country	Region	State	City	Postal_Code
1	United States	West	Arizona	Gilbert	85234
2	United States	West	California	Los Angeles	90045
3	United States	South	Virginia	Charlottesville	22901

00:00:08 Query executed successfully.

Properties

General Related (0)

Name \* SQL script 1

Description

Type .sql script

Size 104 bytes

Results settings per query

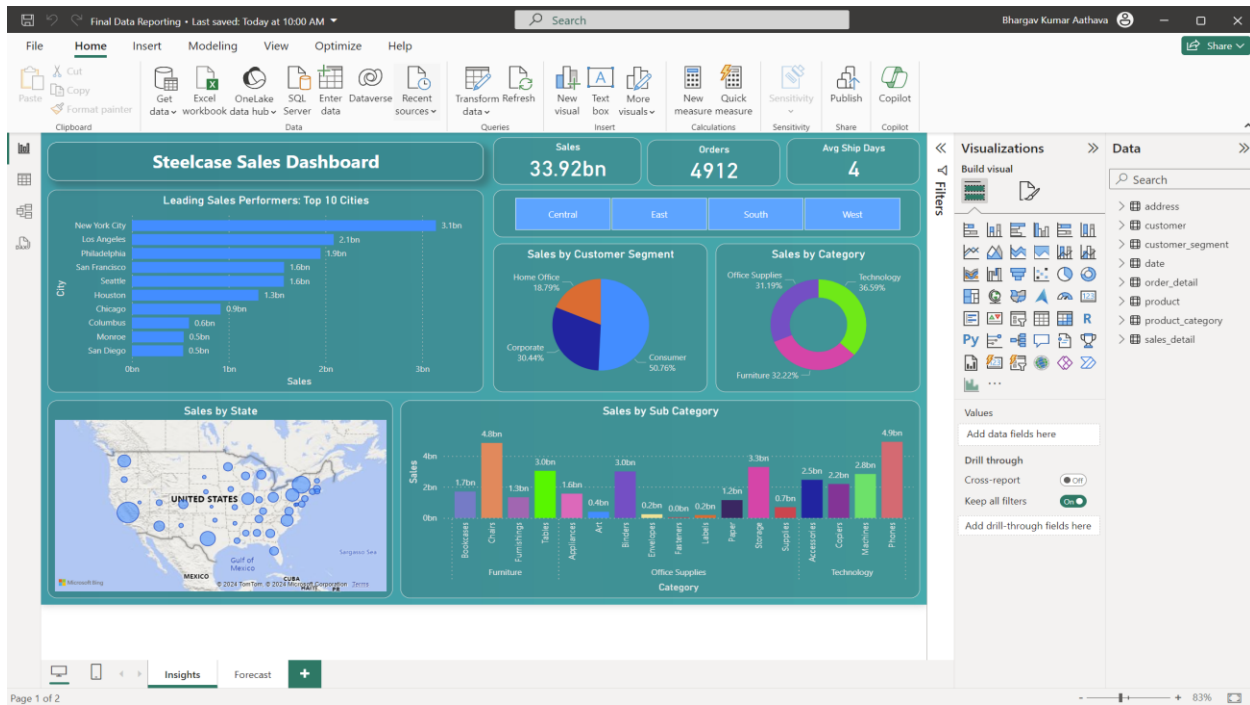
First 5000 rows (default)

All rows

## Stage 4: Data Reporting Using Power BI

Imported data from azure synapse analytics SQL database on to PowerBI to create Reports & Dashboards. Below are the final dashboards.

Insights:



Forecast:

