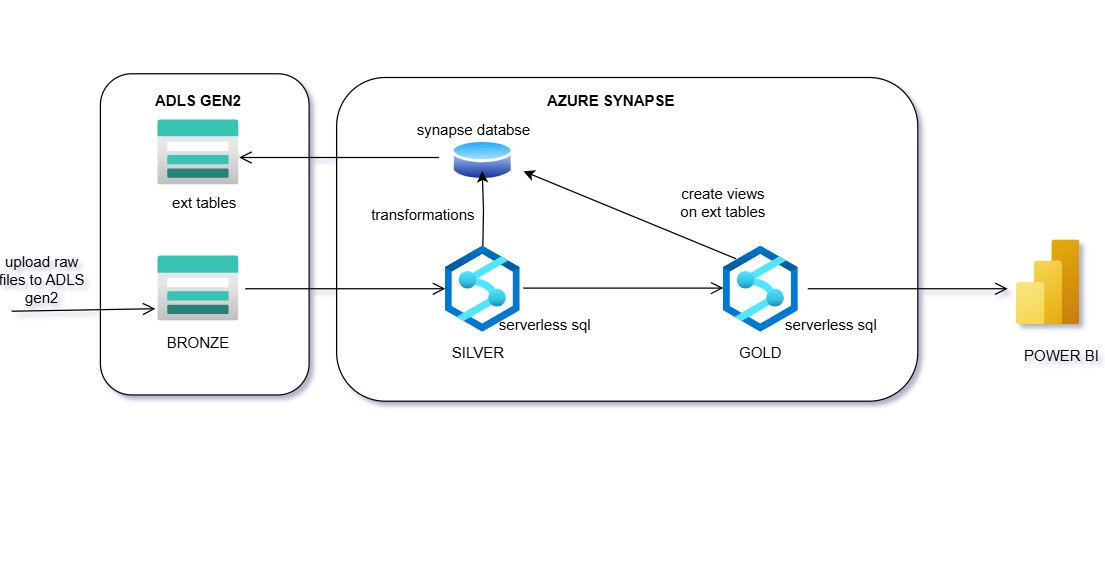
### **Project Overview:**

This project sets up a lakehouse architecture using ADLS Gen2 and Azure Synapse Serverless SQL Pool. Raw files are stored in the bronze layer, then cleaned and deduplicated in the silver layer using CETAS. The cleaned data is saved as external tables in a staging database. On top of this, four curated views form the gold layer. These views are used directly in Power BI for reporting and analysis.

### **Project Architecture:**



#### **Bronze Layer (Raw Ingestion)**

* Raw data files are uploaded directly to the **bronze container** in **ADLS Gen2**.
* No transformation is applied at this layer — data is stored as-is.

#### **Silver Layer (Cleansed Data)**

* Using **Serverless SQL Pool**, a new Synapse database named **staging** was created.
* Data from the bronze layer is read via external tables.
* Cleaning steps applied:
  + Removed rows with NULL values in critical fields.
  + Removed duplicates based on a primary key .
* Cleaned data is saved using **CETAS** (Create External Table As Select) into the **silver container** in ADLS Gen2.
* These external tables form the **silver layer** and are stored in the staging database.

#### **Gold Layer (Reporting Views)**

* Four curated **views** were created on top of the silver tables within the same staging database.
* These views represent business-ready datasets for reporting and KPIs.
* The gold layer is used directly in **Power BI** for dashboarding and analytics.

#### **Tools used:**

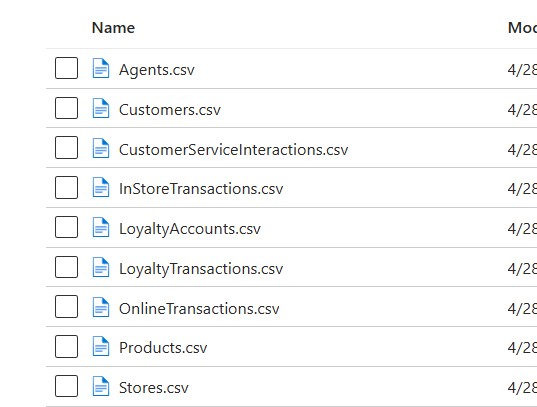
Azure Synapse Analytics

Azure Storage account (ADLS Gen2)

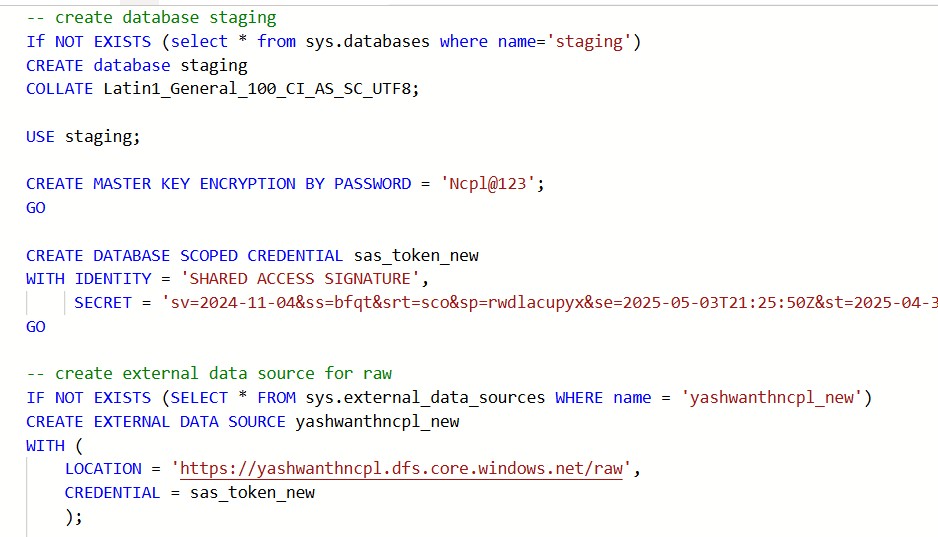
Power BI

## **Detailed Overview of the Project:**

Raw files are loaded manually into ADLS Gen2 bronze

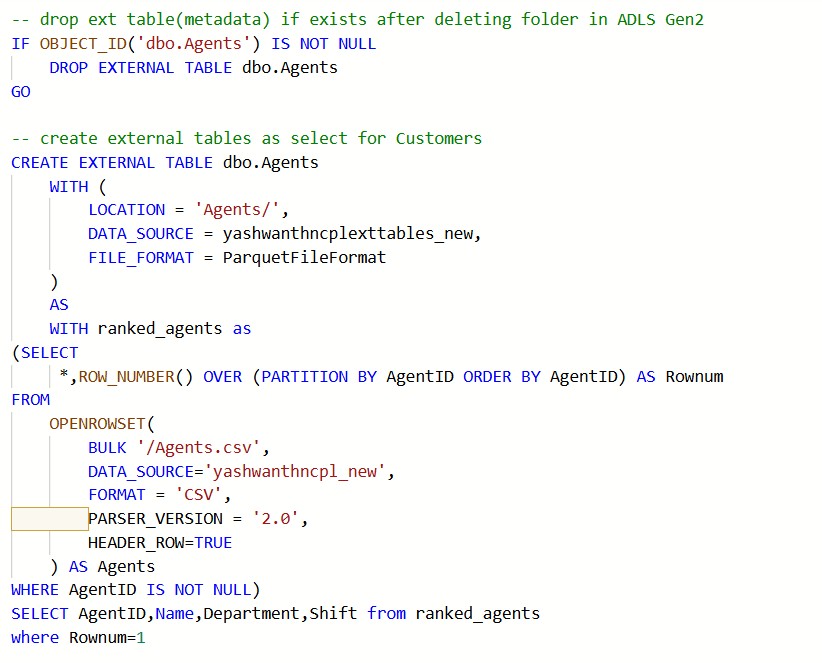


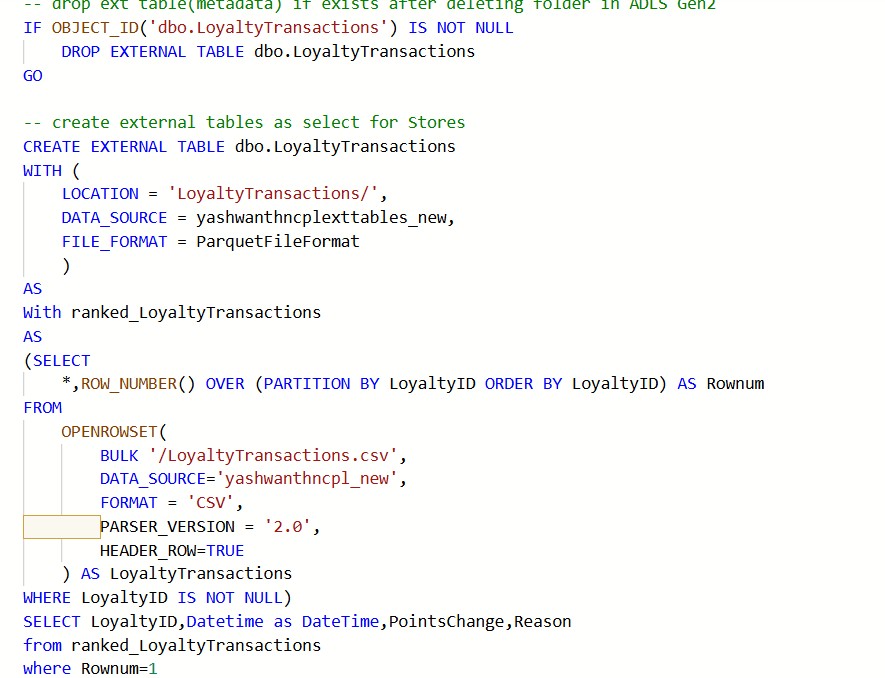
Create a database in serverless sql pool “staging”

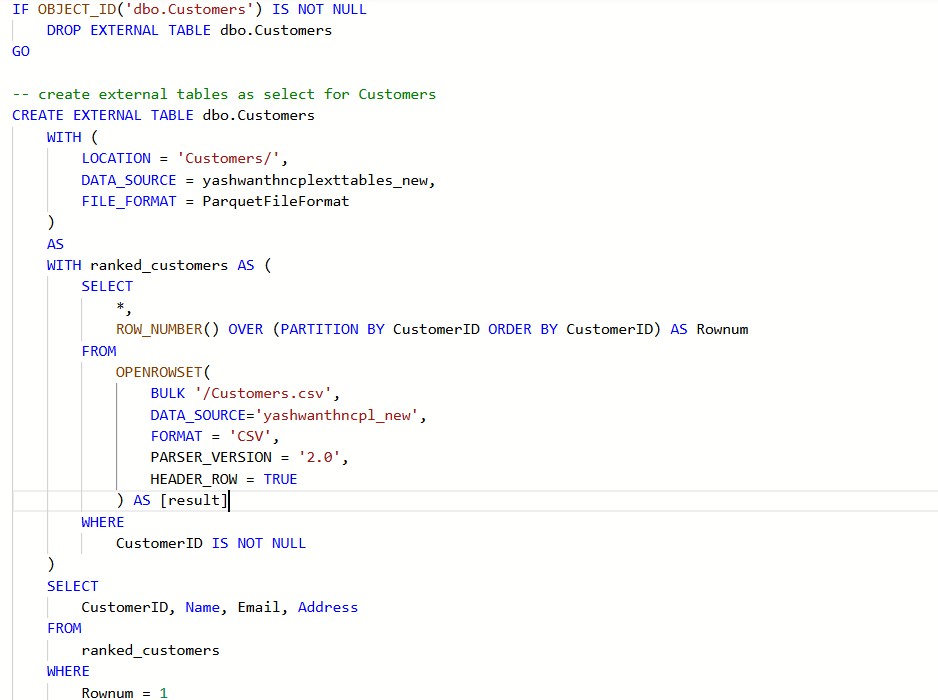




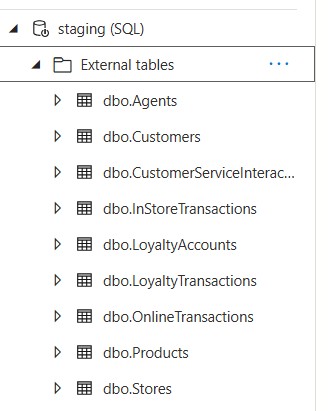
Transform raw files by removing null and duplicates of key column and then create external tables using CETAS for all the raw files

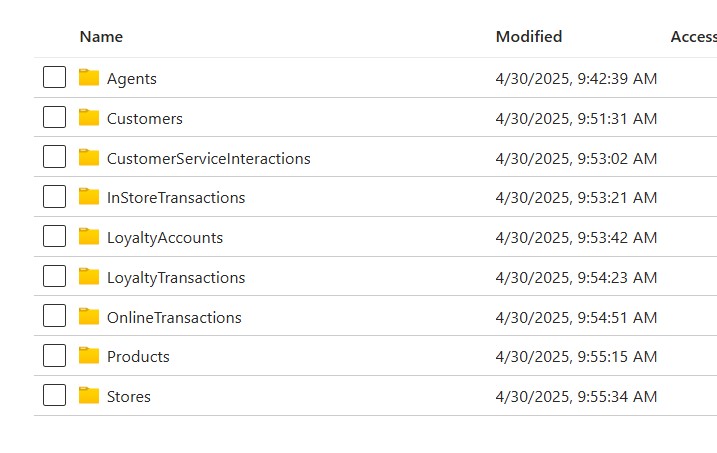






After running CETAS queries external tables will be created in staging database and files in ADLS





With the help of these external tables views are created in the analytics schema

View 1: Average Order value

-- create average order value

CREATE VIEW analyatics.view\_AOV AS

SELECT

p.ProductID,

p.Name AS ProductName,

p.Category,

s.StoreID,

s.Location,

SUM(ot.Amount) / COUNT(ot.OrderID) AS AverageOrderValue

FROM

OnlineTransactions ot

INNER JOIN dbo.Products p ON ot.ProductID = p.ProductID

LEFT JOIN dbo.Stores s ON s.StoreID IS NOT NULL

GROUP BY

p.ProductID, p.Name, p.Category, s.StoreID, s.Location;

View 2: Customer segementation

CREATE or ALTER view analyatics.view\_segementation

AS

with cte\_1 as(

select CustomerID,

OrderID,

Amount,

'Online' AS Channel from

dbo.OnlineTransactions

UNION ALL

select CustomerID,

TransactionID as OrderID,

Amount,

'Instore' AS Channel

from

dbo.InStoreTransactions)

select a.CustomerID,

SUM(a.Amount) as total\_spend,

Count(a.OrderID) as purchase\_frequency,

la.TierLevel,

CASE

WHEN SUM(a.Amount)>= (

SELECT PERCENTILE\_CONT(0.9)

WITHIN GROUP (ORDER BY SUM(a.Amount))

OVER ()

) THEN 'High-Value Customer'

WHEN COUNT(a.OrderID) = 1 THEN 'One-Time Buyer'

WHEN la.Tierlevel = 'Gold' OR la.TierLevel = 'Platinum' THEN 'Loyalty Champion'

ELSE 'Regular Customer'

END AS CustomerSegment

from cte\_1 a

left join dbo.LoyaltyAccounts la

on a.CustomerID=la.CustomerID

group by a.CustomerID,la.TierLevel

View 3: peak timings

CREATE OR ALTER view analyatics.view\_peak\_timings

as

select

DATENAME(WEEKDAY,DateTime) as day,

DATEPART(HOUR,DateTime) as time,

count(OrderID) as transactions,

'online' as transaction\_type

from dbo.[OnlineTransactions]

group by DATENAME(WEEKDAY,DateTime),DATEPART(HOUR,DateTime)

UNION ALL

select

DATENAME(WEEKDAY,DateTime) as day,

DATEPART(HOUR,DateTime) as time,

count(TransactionID) as transactions,

'instore' as transaction\_type

from [dbo].[InStoreTransactions]

group by DATENAME(WEEKDAY,DateTime),DATEPART(HOUR,DateTime)

View 4: Agents Performance

CREATE OR ALTER view analyatics.view\_agents\_performance

AS

With cte1 as

(

select cs.\*,

a.name

from [dbo].[CustomerServiceInteractions] cs

left join [dbo].[Agents] a on cs.AgentID=a.AgentID)

select name,

count(InteractionID) as no\_of\_interactions,

COUNT(CASE WHEN ResolutionStatus = 'Resolved' THEN 1 END) AS resolved\_interactions

from cte1

group by name;

## **Power BI:**

With the help of the views generated above in synapse we can generate visuals in Power Bi by connecting to serverless built in pool.

