

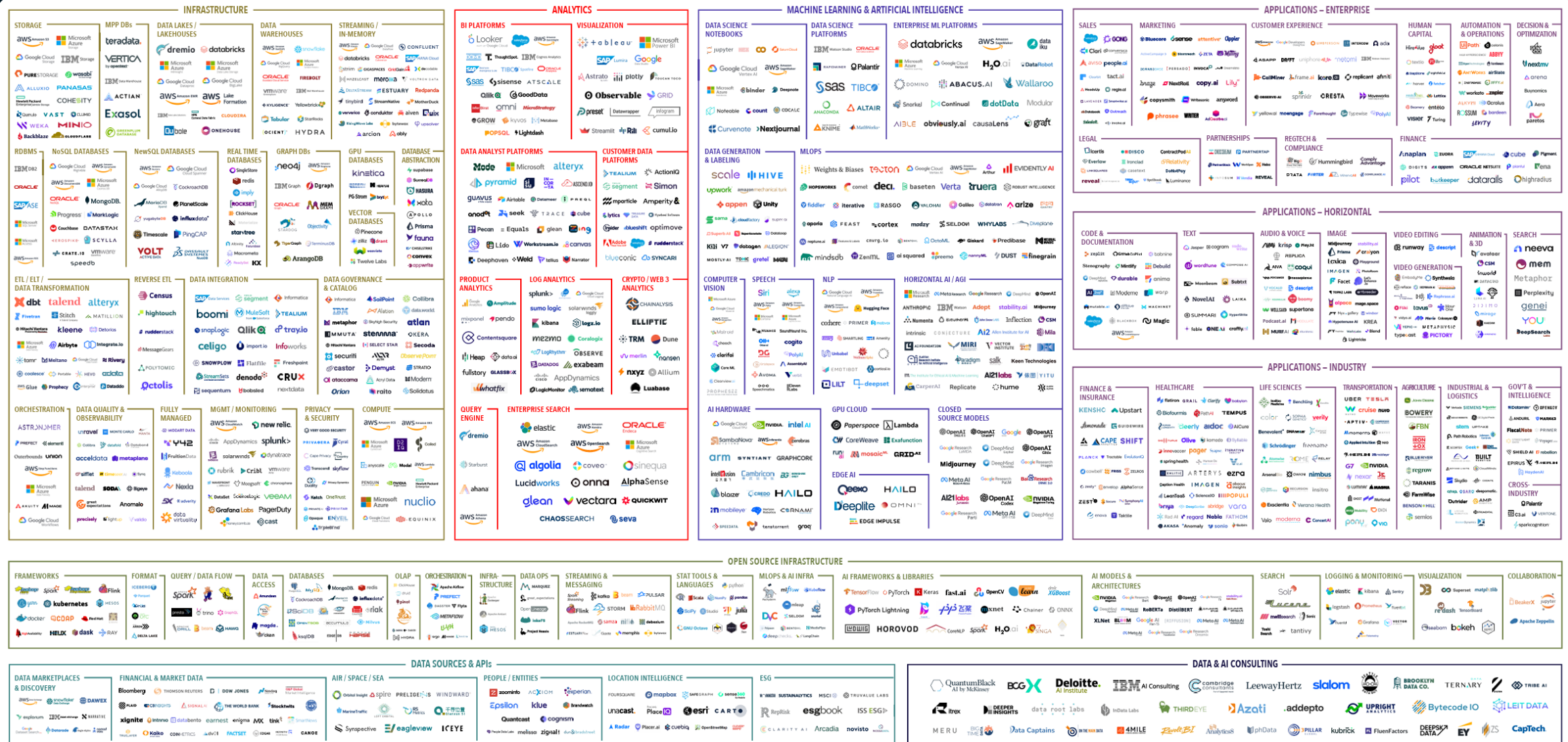


# Introducing Microsoft Fabric

The All-in-One Analytics Solution



# The 2024 ML, AI, and Data Landscape





# Scalable analytics are complex and fragmented

Every analytics project  
has many subsystems

---

Every subsystem need a  
different class of product

---

Products often comes  
from multiple vendors

---

Integration at scale across  
products is complex,  
fragile and expensive



//

**Simplify,**

I am the Chief Data Officer  
and don't want to be the  
Chief Integration Officer."

Every CDO, Every Enterprise

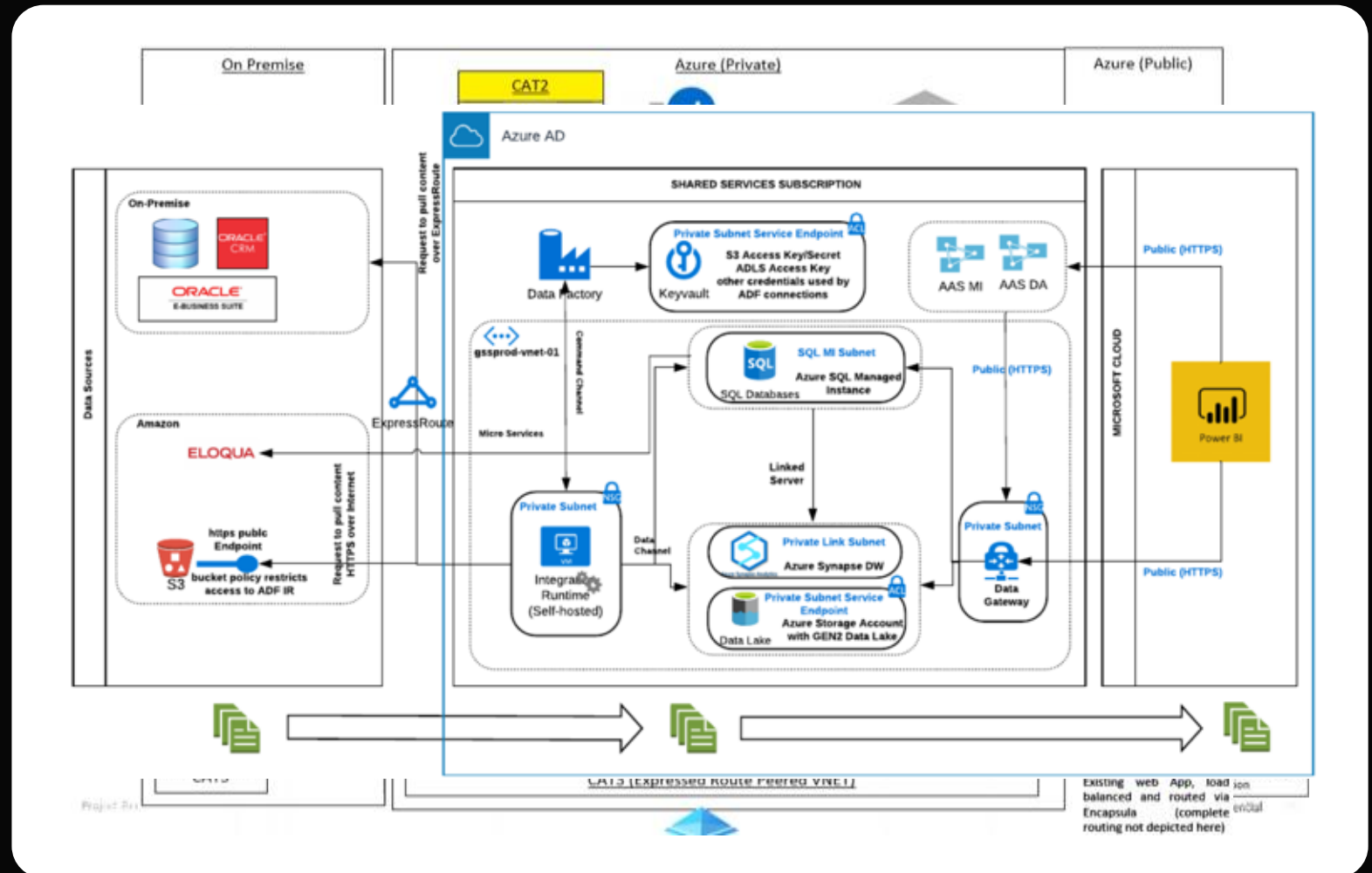
# Analytics is complex and fragmented

Every project has many subsystems

Every subsystem need a different class of product

Products often comes from multiple vendors

Integration is complex, fragile and expensive



# A silver lining?

Analytics have very predictable patterns

---

**Microsoft has all the products with the right scale needed to build a complete analytics system**



# Still far too complex

Many Products

Different Experiences

Proprietary and Open

Dedicated and Serverless

PaaS and SaaS

Different Business Models

Steep Learning Curves

Deep Expertise Needed

High Integration Effort



Purview



Power BI



Kusto



Data Factory



Azure AI



Synapse DW



Synapse Spark

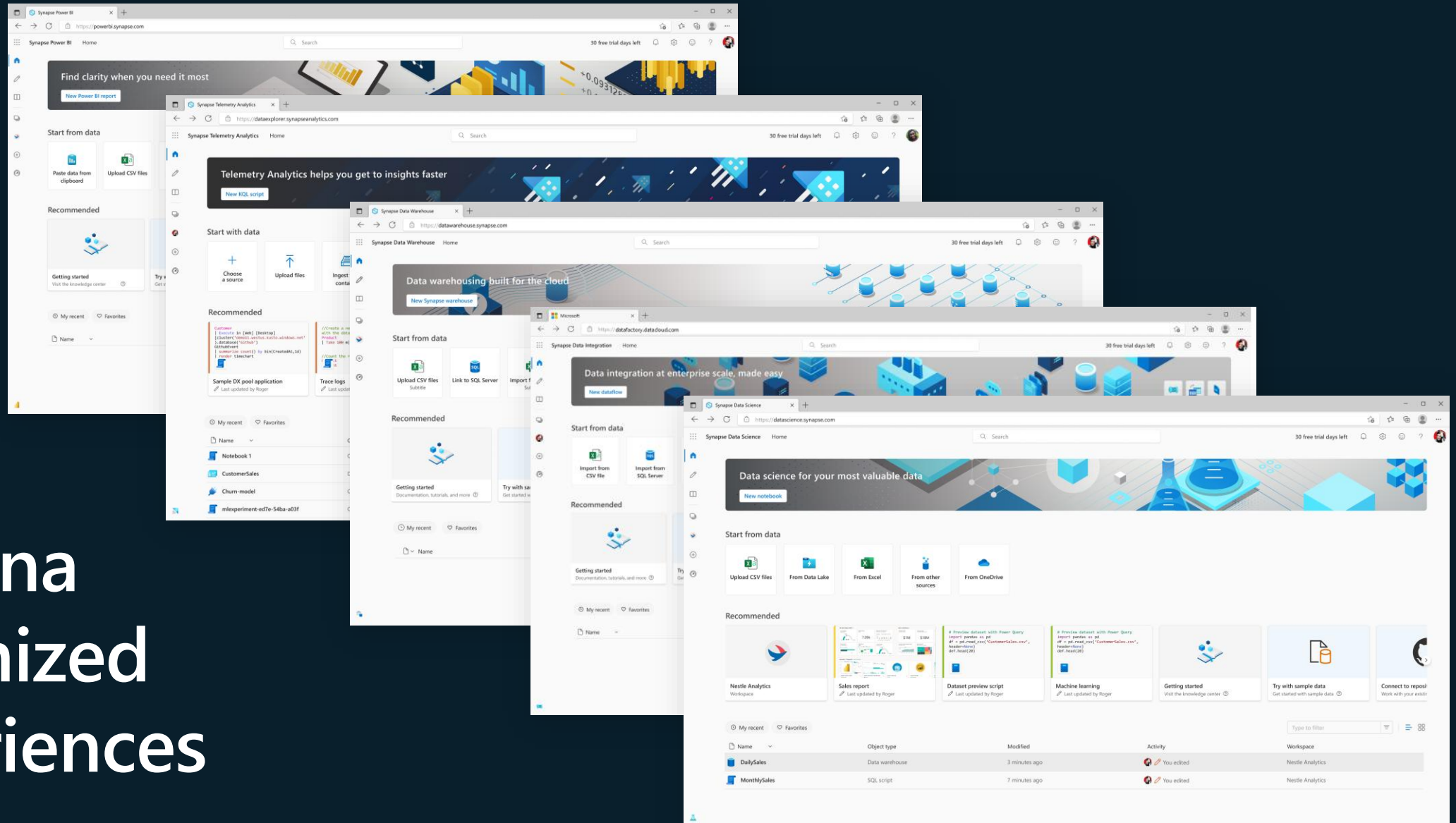
Introducing



**Microsoft Fabric**

The data platform for the era of AI

# Persona optimized experiences







# Microsoft Fabric

## The data platform for the era of AI

### Complete Analytics Platform

Everything, unified

---

SaaS-ified

---

Secured and governed

### Lake centric and open

OneLake

---

One Copy

---

Open at every tier

### Empower Every Business User

Familiar and intuitive

---

Built into Microsoft 365

---

Insight to action

### AI Powered

Copilot accelerated

---

ChatGPT on your data

---

AI driven insights

# SaaS

## "It just works"

### 5x5

Frictionless onboarding

Instant Provisioning

Quick results w/ Intuitive UX

### Success by default

Minimal knobs

Auto optimized

Auto Integrated

### Centralized administration

Tenant-wide governance

Centralized  
security management

Compliance built-in



# Microsoft Fabric

The unified data platform for the era of AI



Data  
Factory



Analytics



Databases



Real-Time  
Intelligence



Power BI



AI



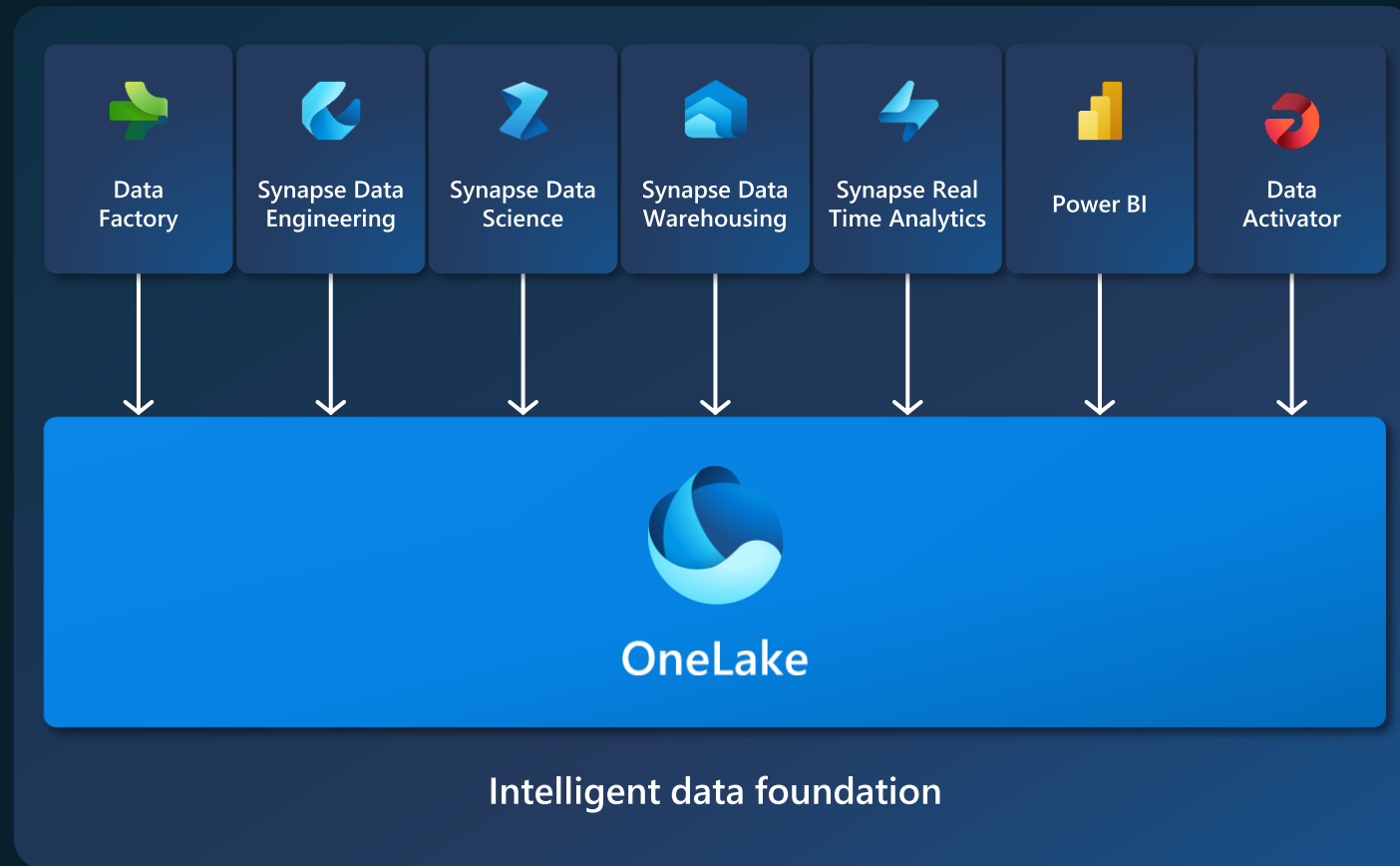
OneLake



Purview

# OneLake for all Data

## "The OneDrive for Data"



A single SaaS lake for the whole organization

Provisioned automatically with the tenant

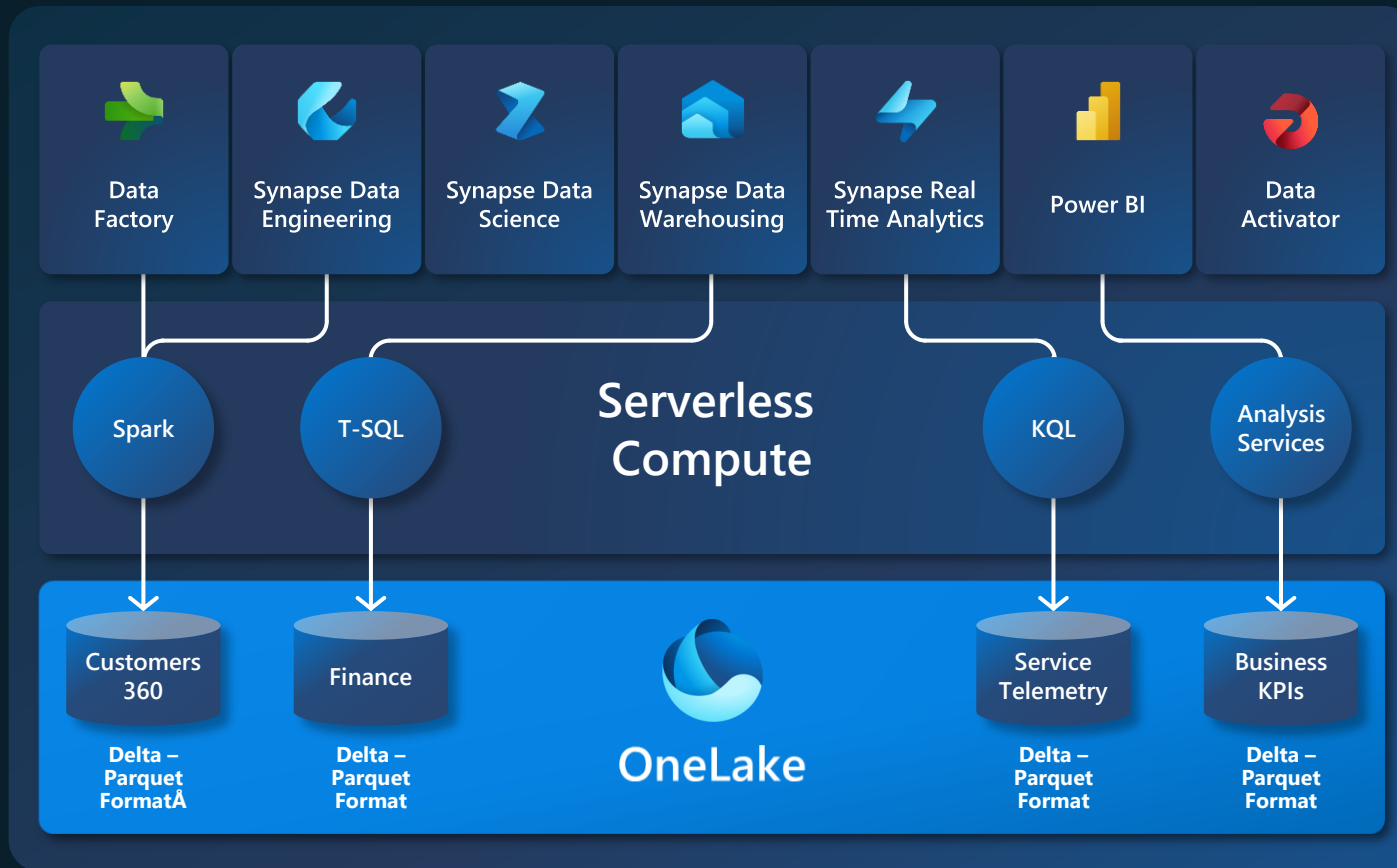
All workloads automatically store their data in the OneLake workspace folders

All the data is organized in an intuitive hierarchical namespace

The data in OneLake is automatically indexed for discovery, MIP labels, lineage, PII scans, sharing, governance and compliance

# One Copy for all computes

## Real separation of compute and storage



All the compute engines store their data automatically in OneLake

The data is stored in a single common format

**Delta – Parquet**, an open standards format, is the storage format for all tabular data in Analytics vNext

Once data is stored in the lake, it is directly accessible by all the engines without needing any import/export

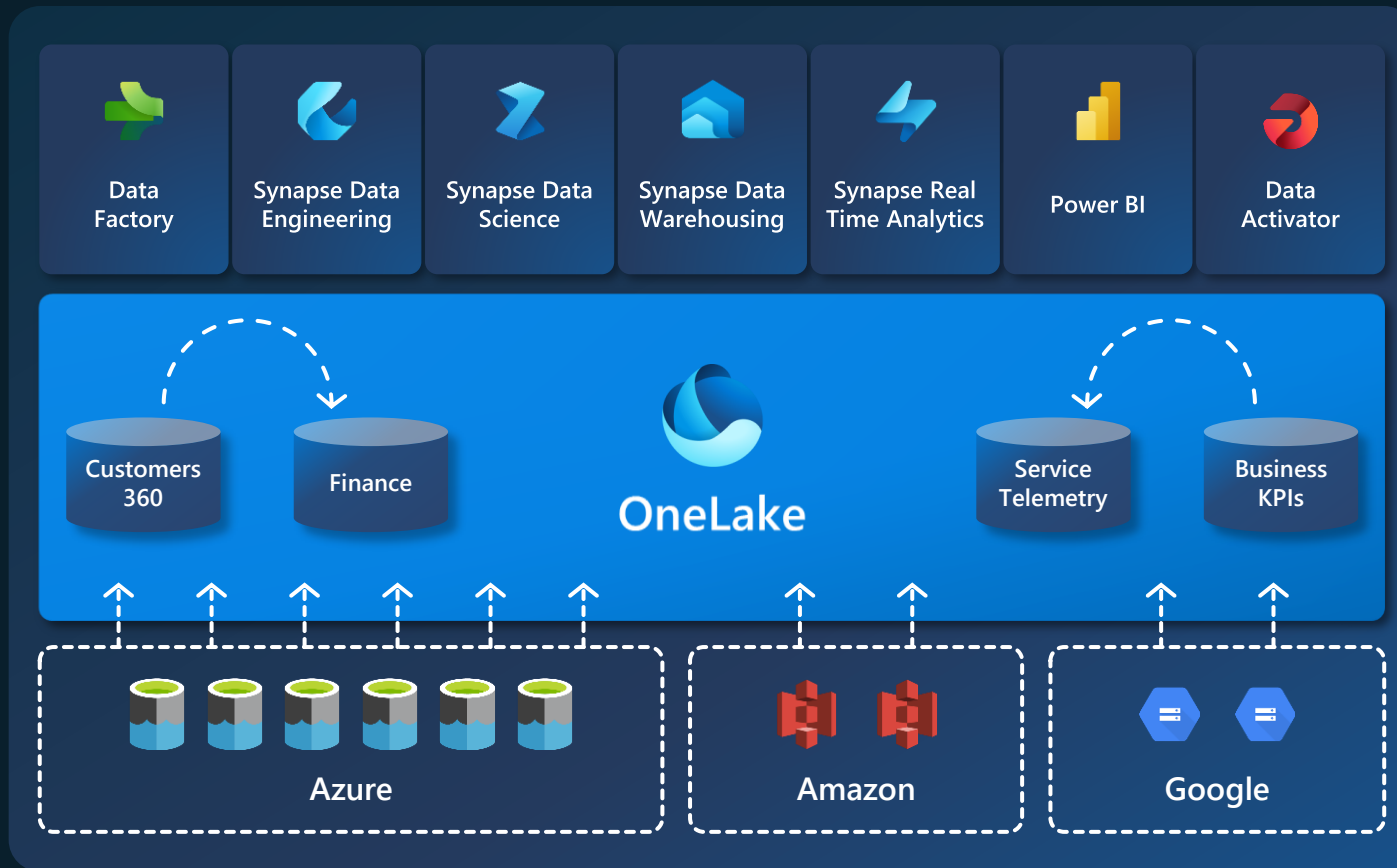
All the compute engines have been fully optimized to work with Delta Parquet as their native format

Shared universal security model is enforced across all the engines



# Taking One Copy to the Next Level

## Shortcuts



Sharing data in OneLake is as easy as sharing files in OneDrive, removing the needs for data duplication

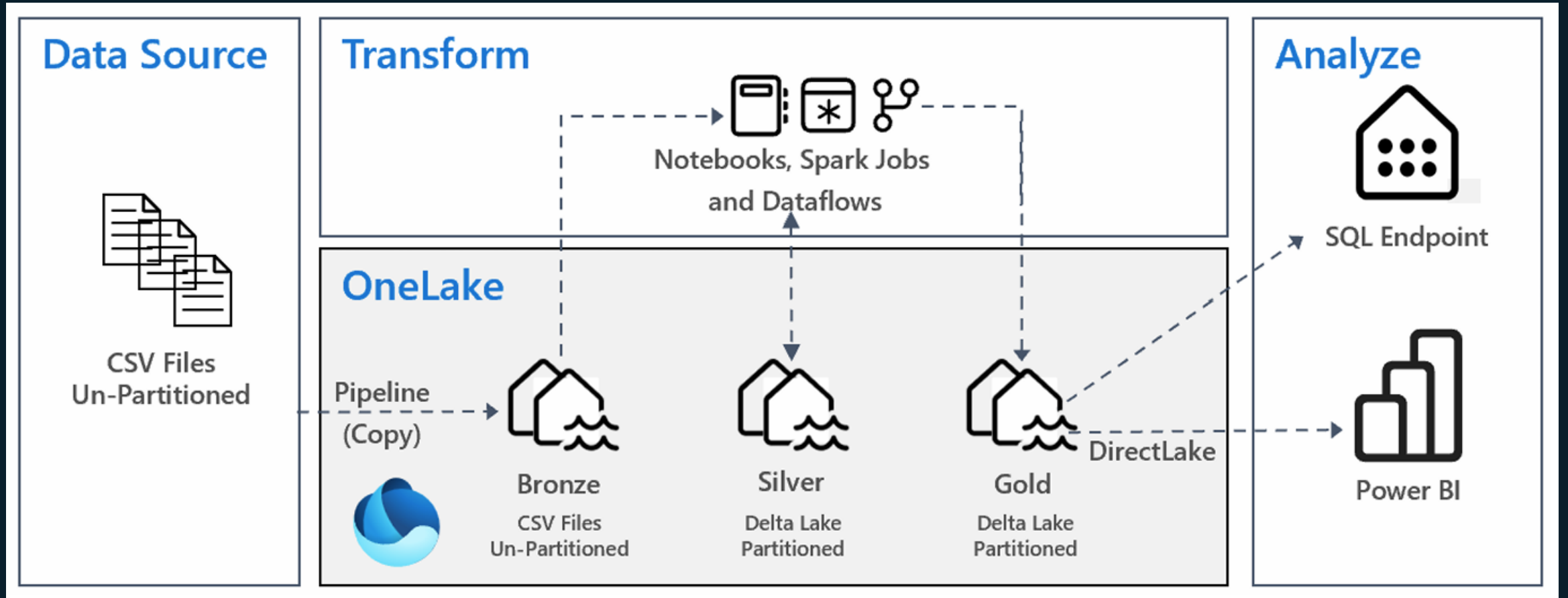
With **shortcuts**, data throughout OneLake can be composed together without any data movement

Shortcuts also allow instant linking of data already existing in Azure and in other clouds, without any data duplication and movement, making **OneLake the first multi-cloud data lake**

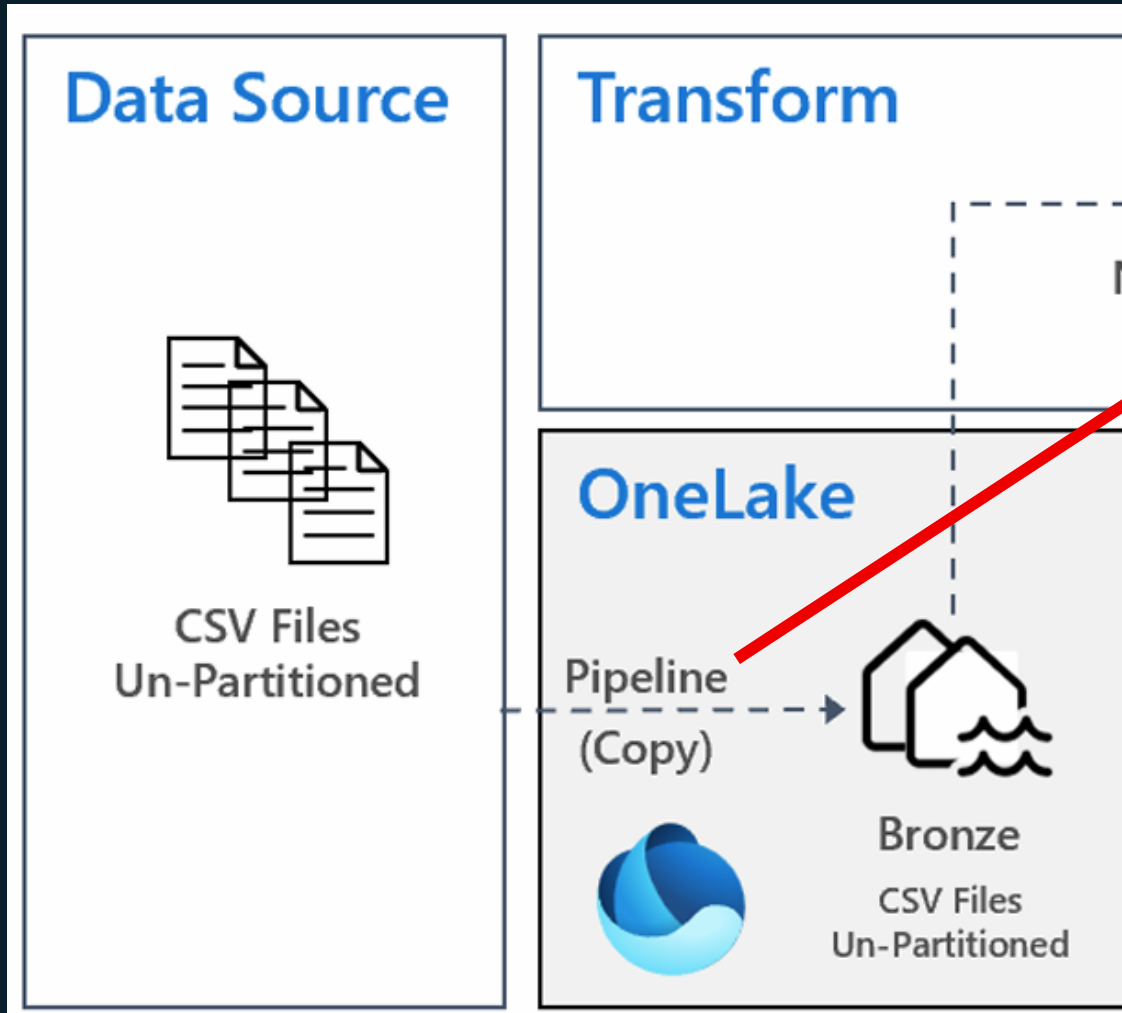
With support for industry standard APIs, OneLake data can be directly accessed by any application or service

**Demo - Lakehouse**

# Architecture, data & transformation flow



# Demo – Ingestion (Bronze)



The screenshot shows the **데이터 복사** (Copy Data) configuration window. The window has a green header bar with the title **데이터 복사**. Below the header, there is a section titled **Copy Data to Bronze** with a trash icon, a code icon, a document icon, and a green arrow icon.

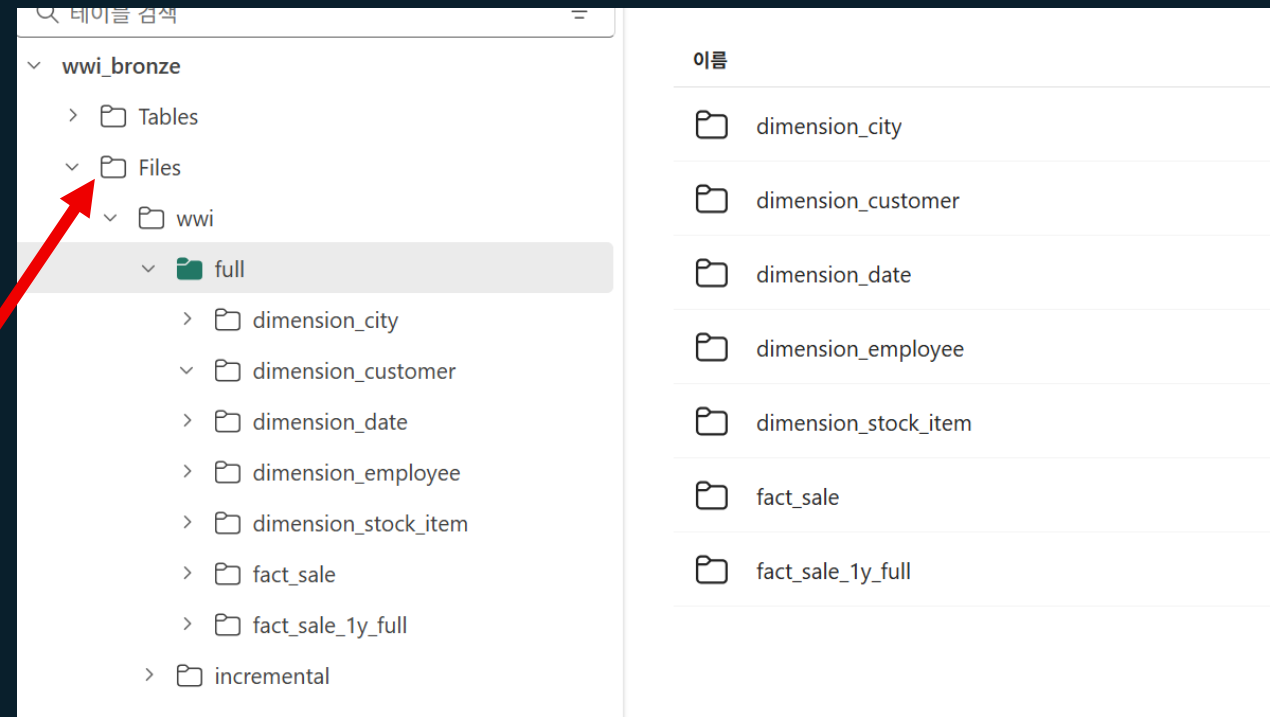
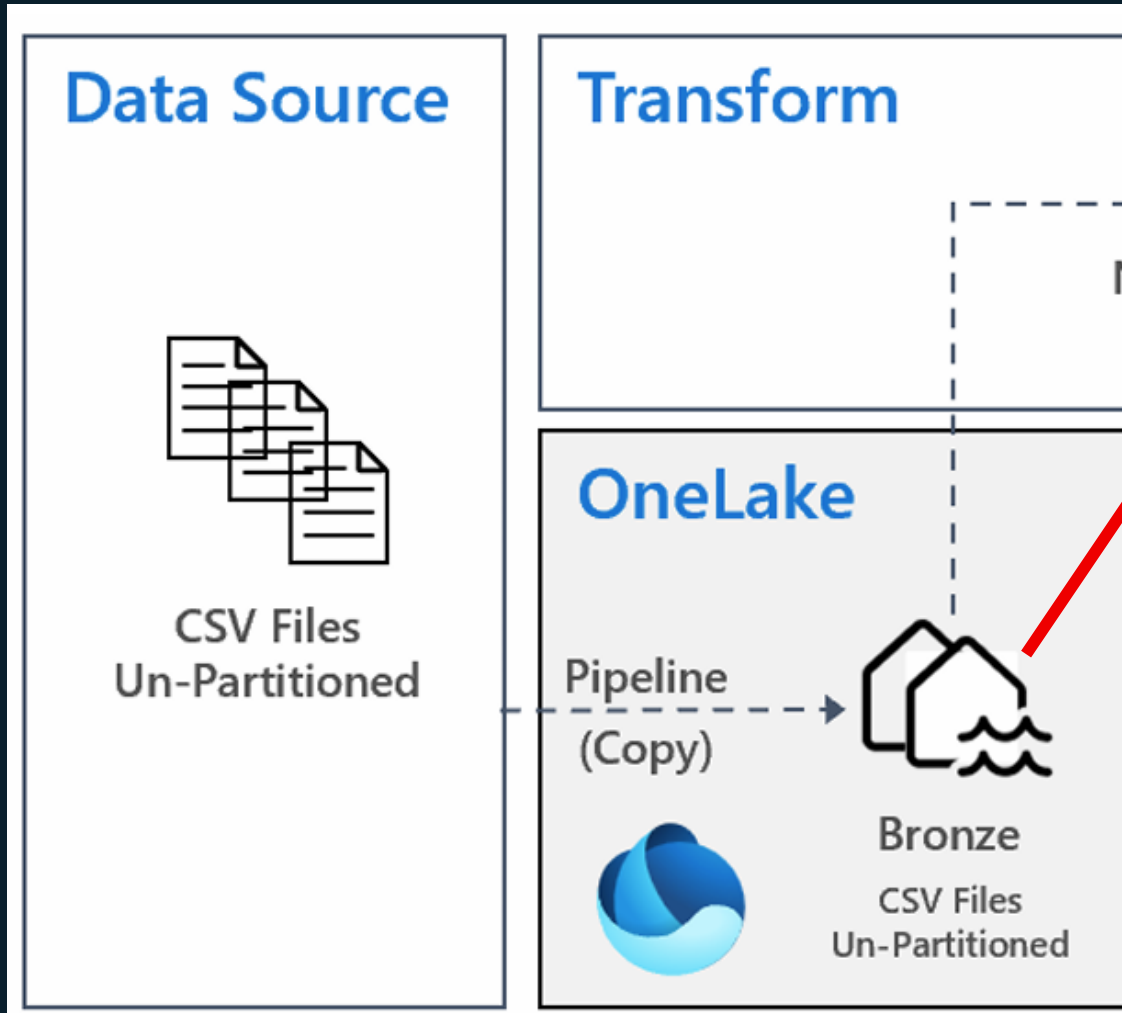
The window is divided into tabs: **일반** (General), **원본** (Source), **대상** (Target), **매핑** (Mapping), and **설정** (Settings). The **일반** tab is selected.

The **일반** tab contains the following fields:

- 이름 \*** (Name):  [자세한 정보](#)
- 설명** (Description):
- 활동 상태** (Activity Status): ☒ 활성화됨 ☐ 비활성화됨
- 시간 초과** (Timeout):
- 다시 시도** (Retries):

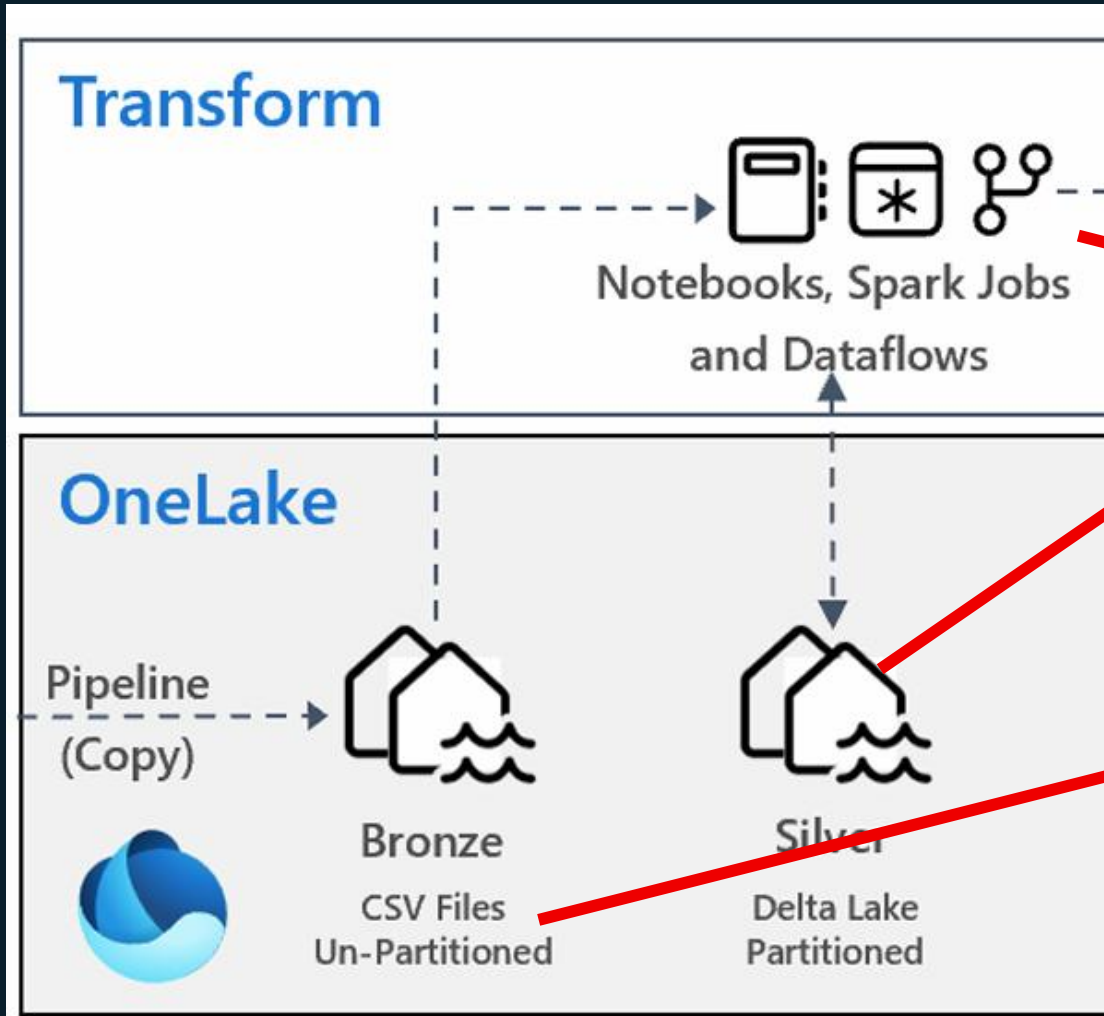
At the bottom, there is a **> 고급** (Advanced) link.

# Demo – Ingestion (Bronze)





# Demo – Transformation (Silver)

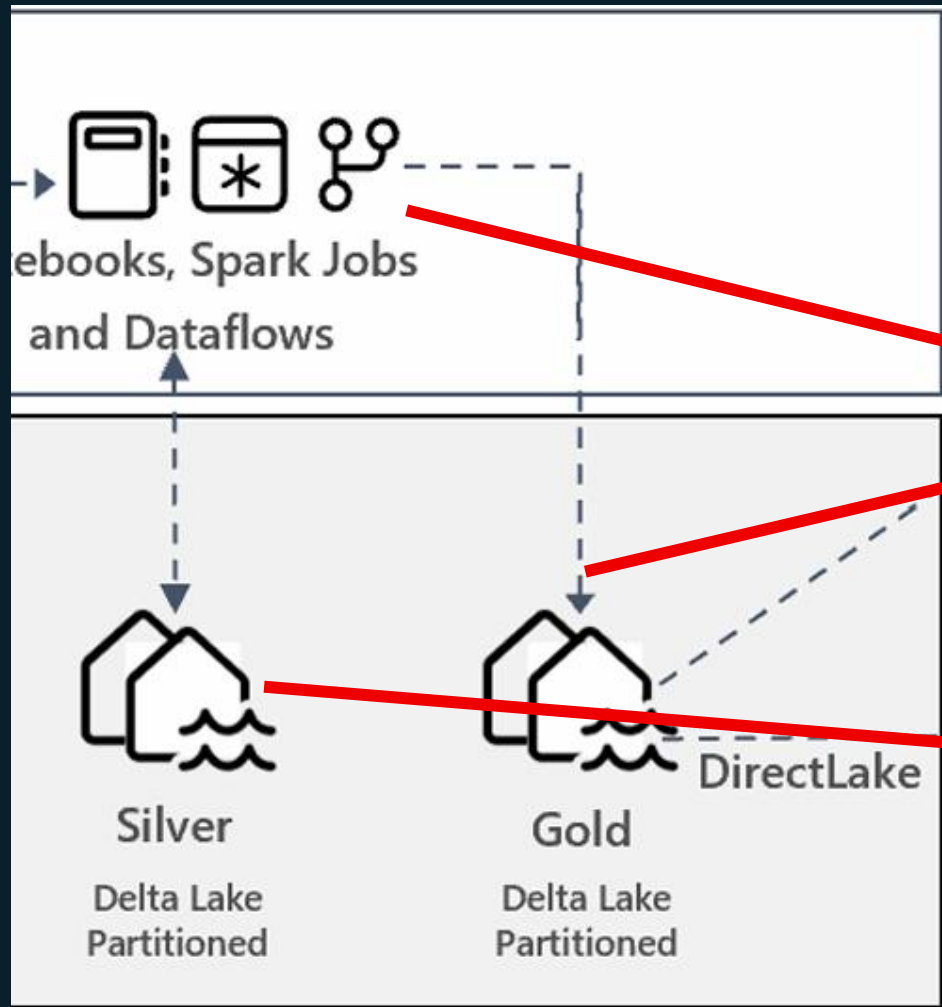


The screenshot shows the Databricks Explorer interface. The left pane displays the **wwi\_silver** database structure, including a list of tables and files. The right pane shows the **Dimension - City** table, which is a Delta table. The table's schema is defined by a PySpark code cell.

```
1 from pyspark.sql.types import *
2
3 table_name = 'dimension_city'
4
5 dimension_city_schema = StructType([
6     StructField('CityKey', IntegerType(), True),
7     StructField('WWICityID', IntegerType(), True),
8     StructField('City', StringType(), True),
9     StructField('StateProvince', StringType(), True),
10    StructField('Country', StringType(), True),
11    StructField('Continent', StringType(), True),
12    StructField('SalesTerritory', StringType(), True),
13    StructField('Region', StringType(), True),
14    StructField('Subregion', StringType(), True),
15    StructField('Location', StringType(), True),
16    StructField('LatestRecordedPopulation', LongType(), True),
17    StructField('ValidFrom', TimestampType(), True),
18    StructField('ValidTo', TimestampType(), True),
19    StructField('LineageKey', IntegerType(), True)]])
20
21 df = spark.read.format("csv").schema(dimension_city_schema).option("header",
22 df.write.mode("overwrite").format("delta").save("Tables/" + table_name)
```

The code cell is executed, and the output shows the table's schema and the number of rows (18 rows, 193 ms).

# Demo – Aggregation(Gold)



Explorer

데이터 항목 리소스

+ 데이터 항목 추가

Items

- wwi\_gold
  - Tables
    - aggregate\_sale\_by\_date\_city
    - aggregate\_sale\_by\_date\_employee
    - dimension\_city
    - dimension\_customer
    - dimension\_date
    - dimension\_employee
    - dimension\_stock\_item
    - fact\_sale
  - Files

Approach #1 - sale\_by\_date\_city

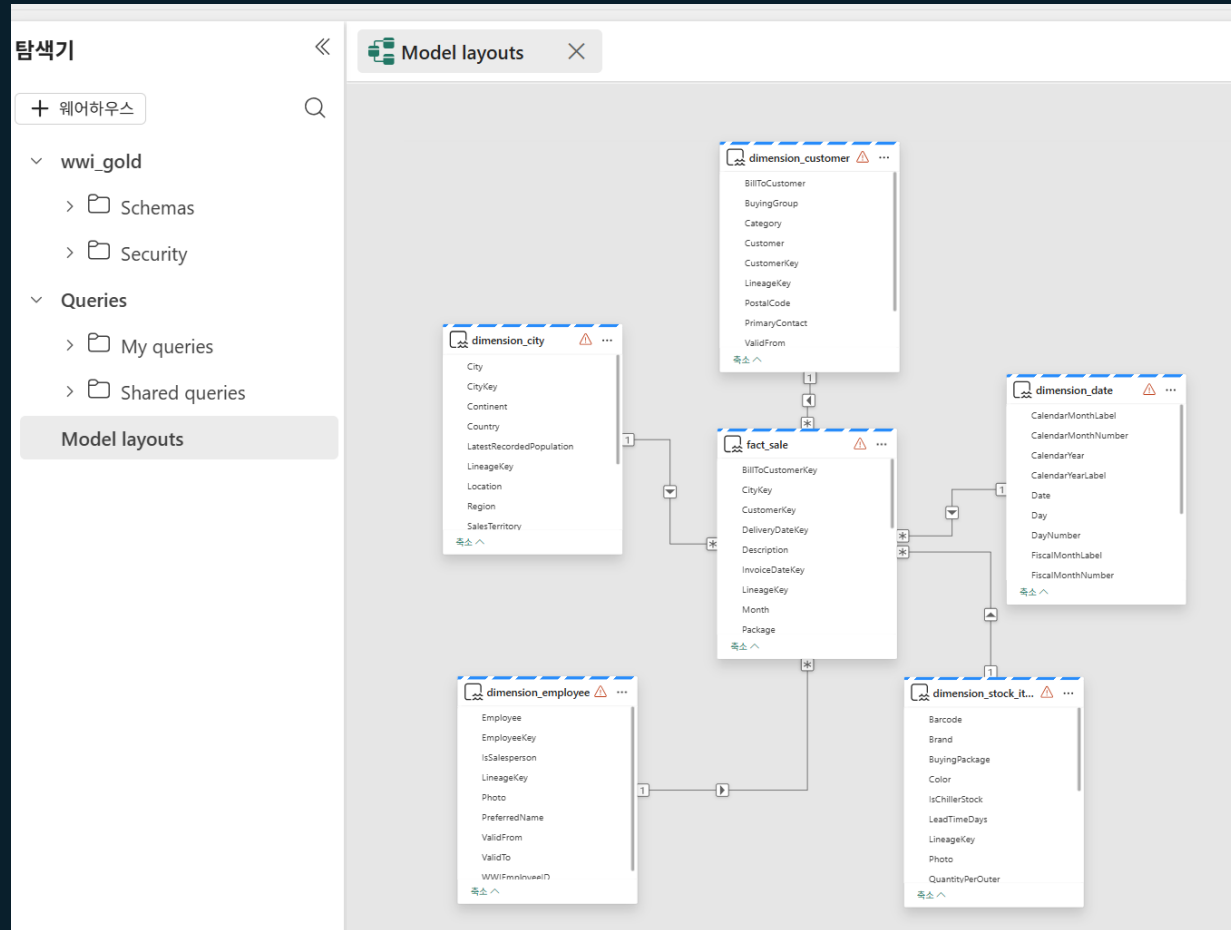
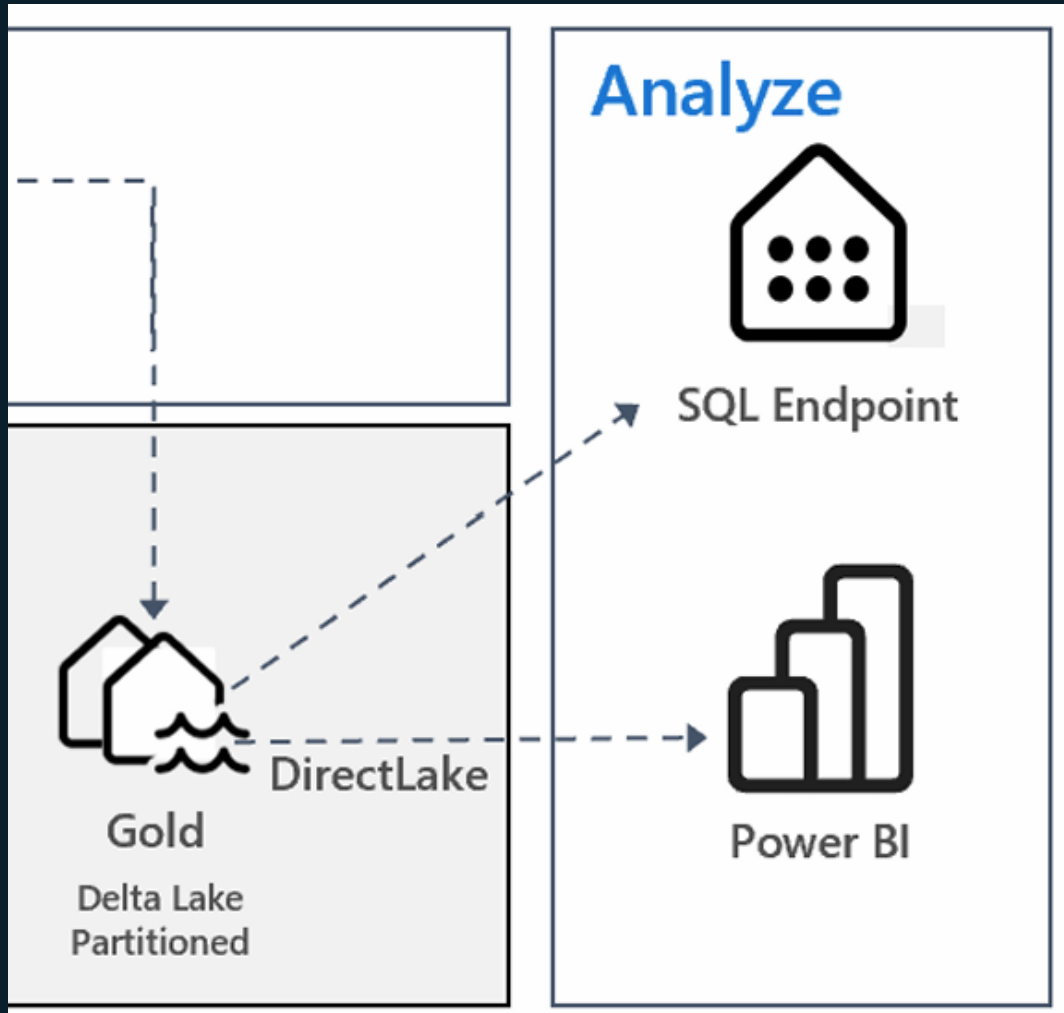
```
1 df_fact_sale = spark.read.table("wwi_gold.fact_sale")
2 df_dimension_date = spark.read.table("wwi_gold.dimension_date")
3 df_dimension_city = spark.read.table("wwi_gold.dimension_city")
```

[1] ✓ - 15 초 566 ms에 세션이 준비되었습니다. Command executed in 45 초 991 ms by System Administrator of

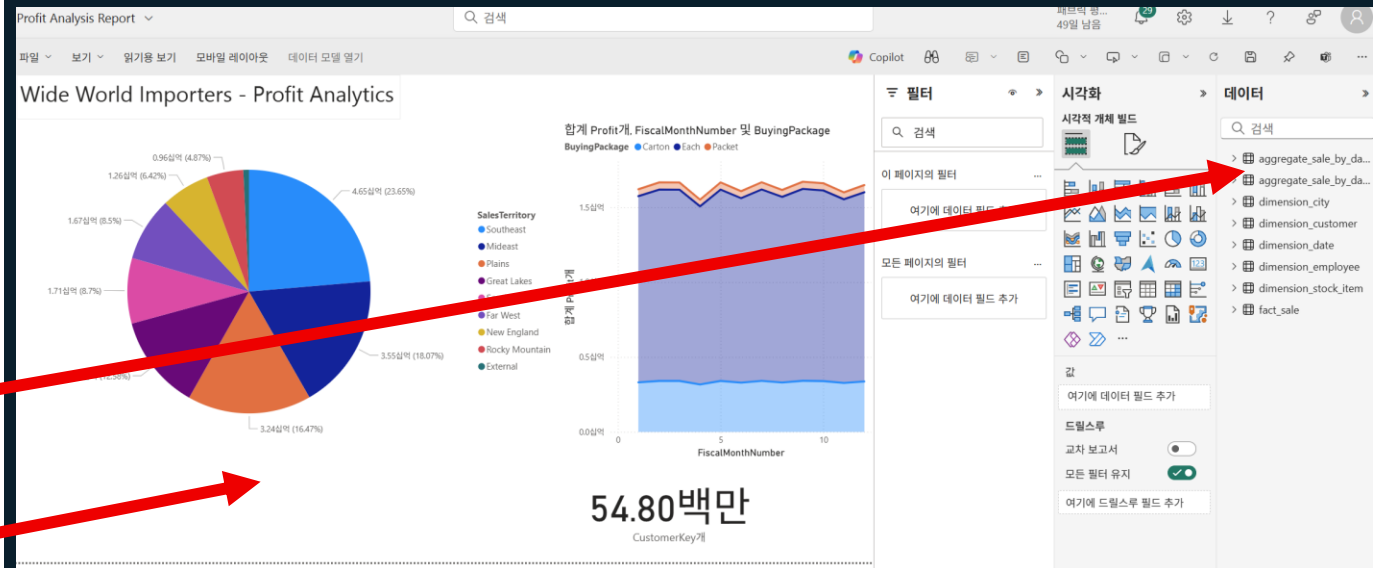
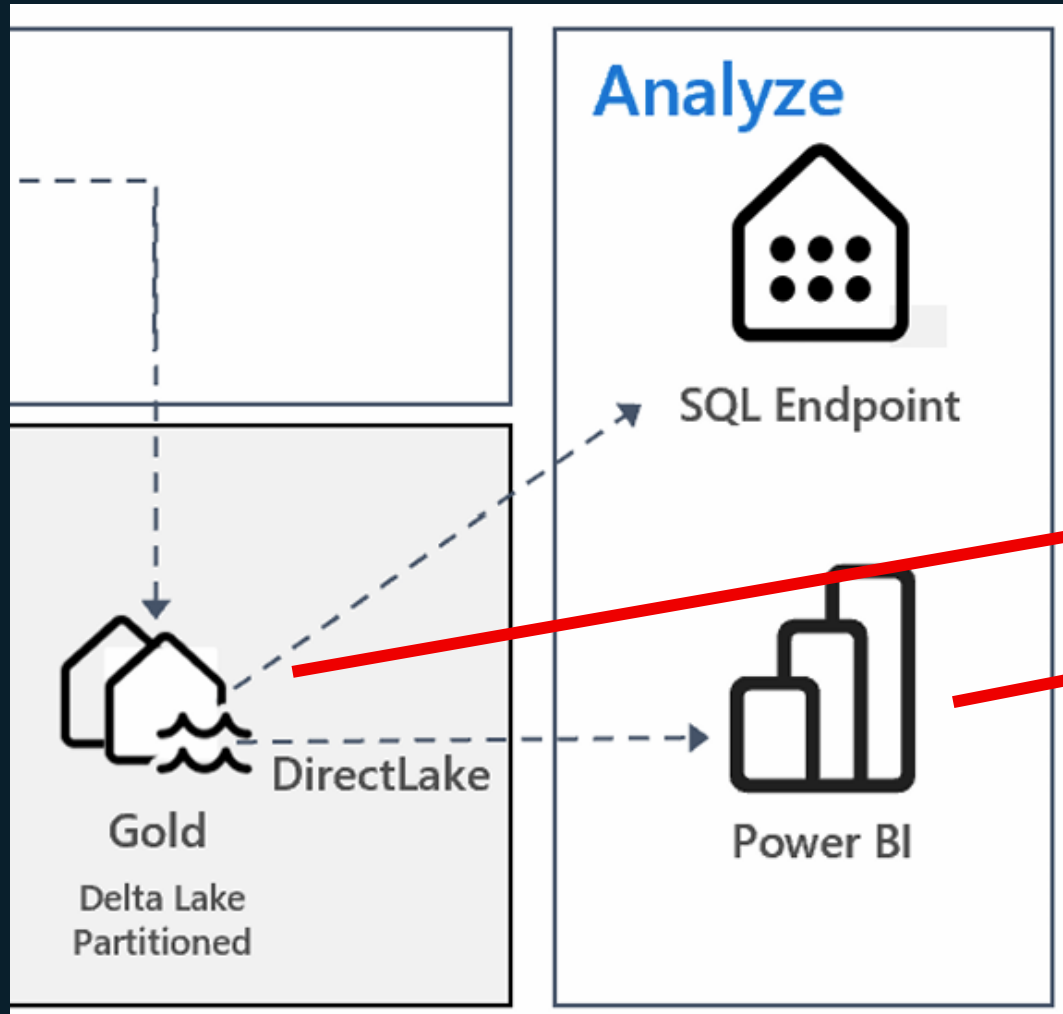
```
1 sale_by_date_city = df_fact_sale.alias("sale") \
2 .join(df_dimension_date.alias("date"), df_fact_sale.InvoiceDateKey == df_dimension_date.DateKey) \
3 .join(df_dimension_city.alias("city"), df_fact_sale.CityKey == df_dimension_city.CityKey) \
4 .select("date.Date", "date.CalendarMonthLabel", "date.Day", "date.StateProvince", "city.City") \
5 .groupBy("date.Date", "date.CalendarMonthLabel", "date.Day", "date.StateProvince", "city.City") \
6 .sum("sale.TotalExcludingTax", "sale.TaxAmount", "sale.TotalIncludingTax", "sale.Profit") \
7 .withColumnRenamed("sum(TotalExcludingTax)", "SumOfTotalExcludingTax") \
8 .withColumnRenamed("sum(TaxAmount)", "SumOfTaxAmount") \
9 .withColumnRenamed("sum(TotalIncludingTax)", "SumOfTotalIncludingTax") \
10 .withColumnRenamed("sum(Profit)", "SumOfProfit") \
11 .orderBy("date.Date", "city.StateProvince", "city.City")
12
13 sale_by_date_city.write.mode("overwrite").format("delta").option("overwriteMode", "error").saveAsTable("wwi_gold.aggregate_sale_by_date_city")
```

[2] ✓ Command executed in 10 초 380 ms에 세션이 준비되었습니다. Command executed in 10 초 548 ms by System Administrator of

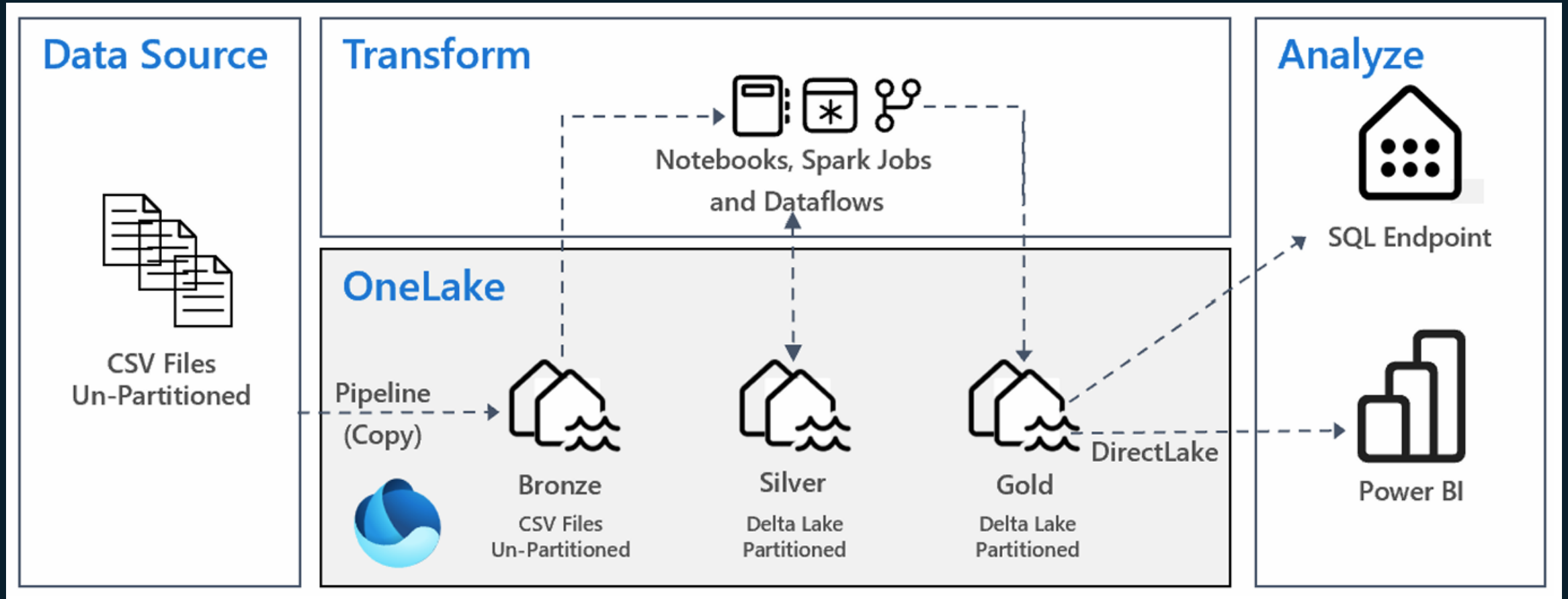
# Demo – Analyze(PBI)



# Demo – Analyze(PBI)



# Architecture, data & transformation flow







Questions?



감사합니다

