END-TO-END E-COMMERCE DATA ENGINEERING PIPELINE USING AZURE

SCALABLE DATA INGESTION, TRANSFORMATION & VISUALIZATION WITH DATABRICKS AND SYNAPSE

Presented by: TEAM 10

Mayuresh Pramod Pandey
Aayush Anil Patil
Devanshi Vyas
Nikhil Deepak Swami
Suyog Sanjeevan Jadhav
Keerthika Loganathan

Date: May 10,2025
Course: DATA228-Big Data Technologies and Applications
San Jose State University

PROJECT INTRODUCTION

- "This project demonstrates how to build a scalable and efficient data pipeline using Azure cloud components to ingest, process, and visualize real-world e-commerce data from the Olist dataset."
- Enabling businesses to gain insights into customer behavior, logistics efficiency, and product performance.

PROBLEM STATEMENT

- Large-scale e-commerce platforms generate complex data from various sources: reviews, transactions, shipping, inventory.
- Manual or semi-automated ETL processes are error-prone and slow.

Need for :

CENTRALIZED DATA STORAGE

AUTOMATED AND SCALABLE TRANSFORMATION

REAL-TIME ANALYTICS CAPABILITY

CLEAN VISUALIZATION LAYER FOR BUSINESS DECISIONS

PROJECT OBJECTIVES

- Build a cloud-native pipeline with modular components
- Automate ingestion from public and SQL sources using Azure Data Factory
- Clean and enrich data with Apache Spark on Azure Databricks
- Store and query transformed data using Synapse Analytics
- Visualize metrics with Power BI, Tableau, or Microsoft Fabric

DATASET OVERVIEW

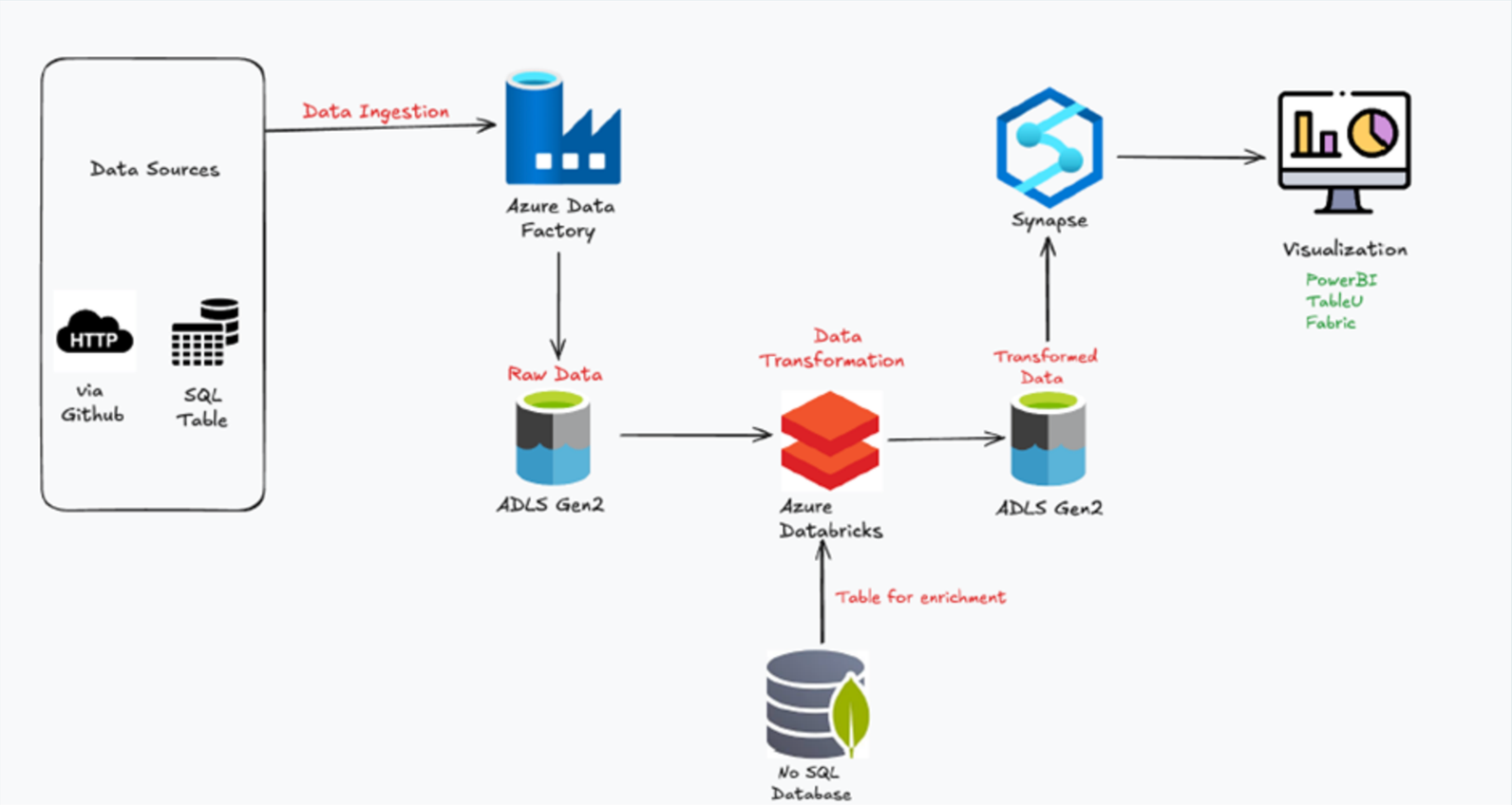
- Based on the Brazilian E-commerce Public Dataset by Olist
- Data files used:
  - Orders,Customers,Reviews,Payments,Products, Sellers, Geolocation,etc.
- Features:
  - 13M+ records of transactional data
  - Multi-table relationships (foreign keys)
  - Encompasses logistics, customer feedback, and sales metrics

TOOLS & TECHNOLOGIES

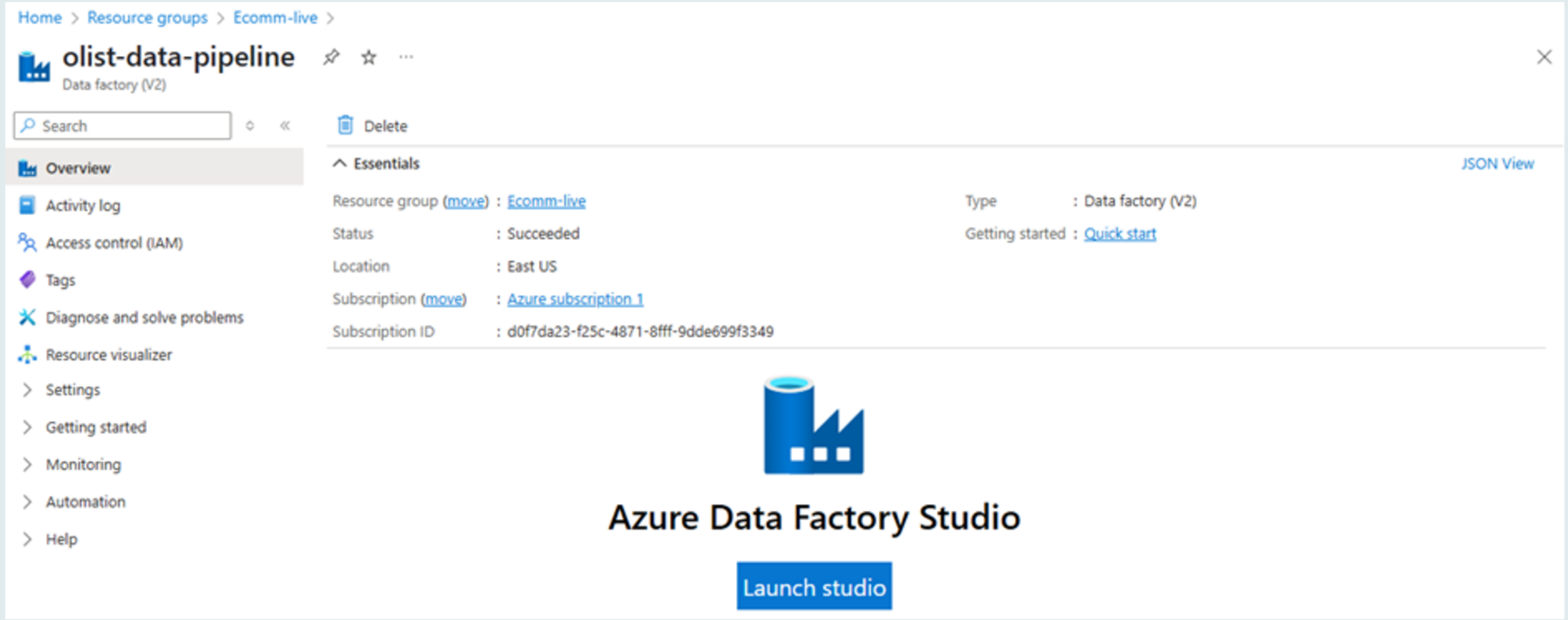
- AZURE DATA FACTORY (ADF)
- AZURE DATA LAKE GEN2
- AZURE DATABRICKS
- AZURE SYNAPSE ANALYTICS
- NoSQL DB (MongoDB)
- SQL DB (MySQL)
- PySpark

Visualization Tools: Tableau

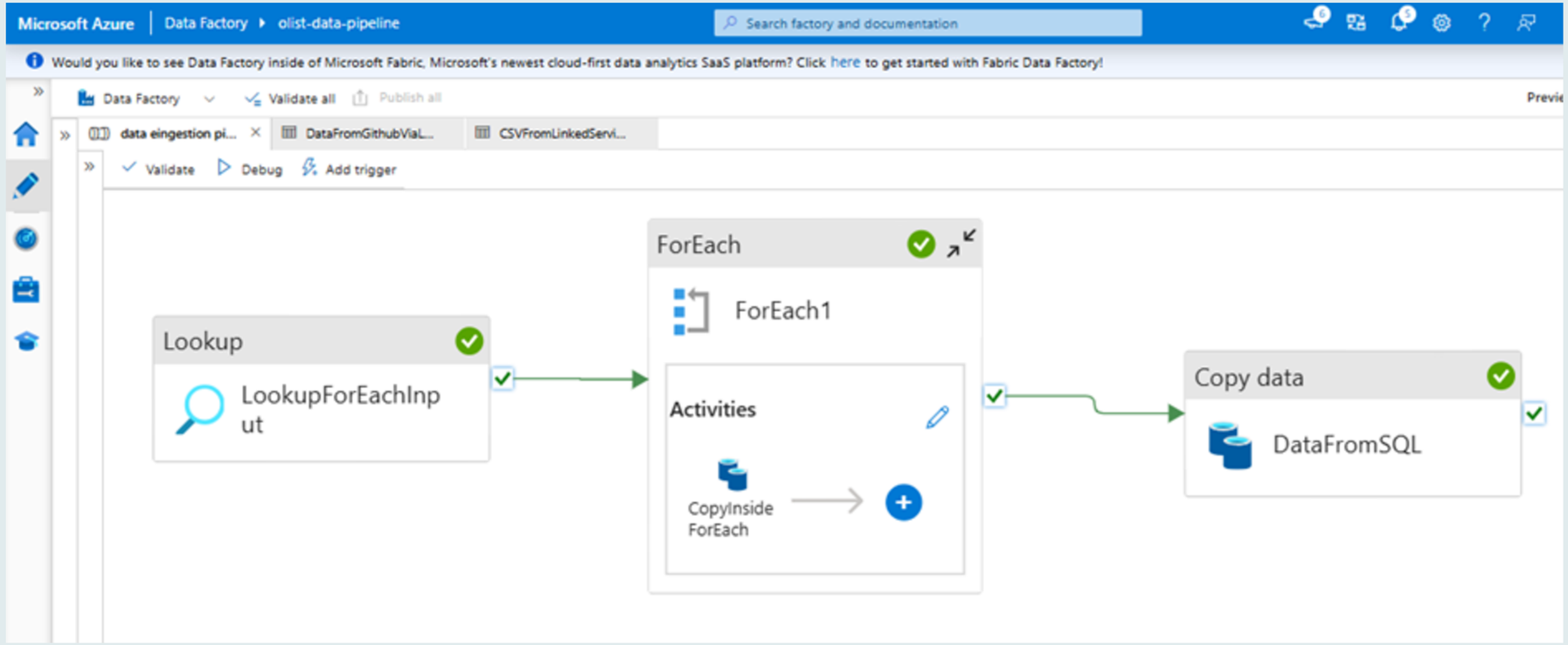
ARCHITECTURE OVERVIEW



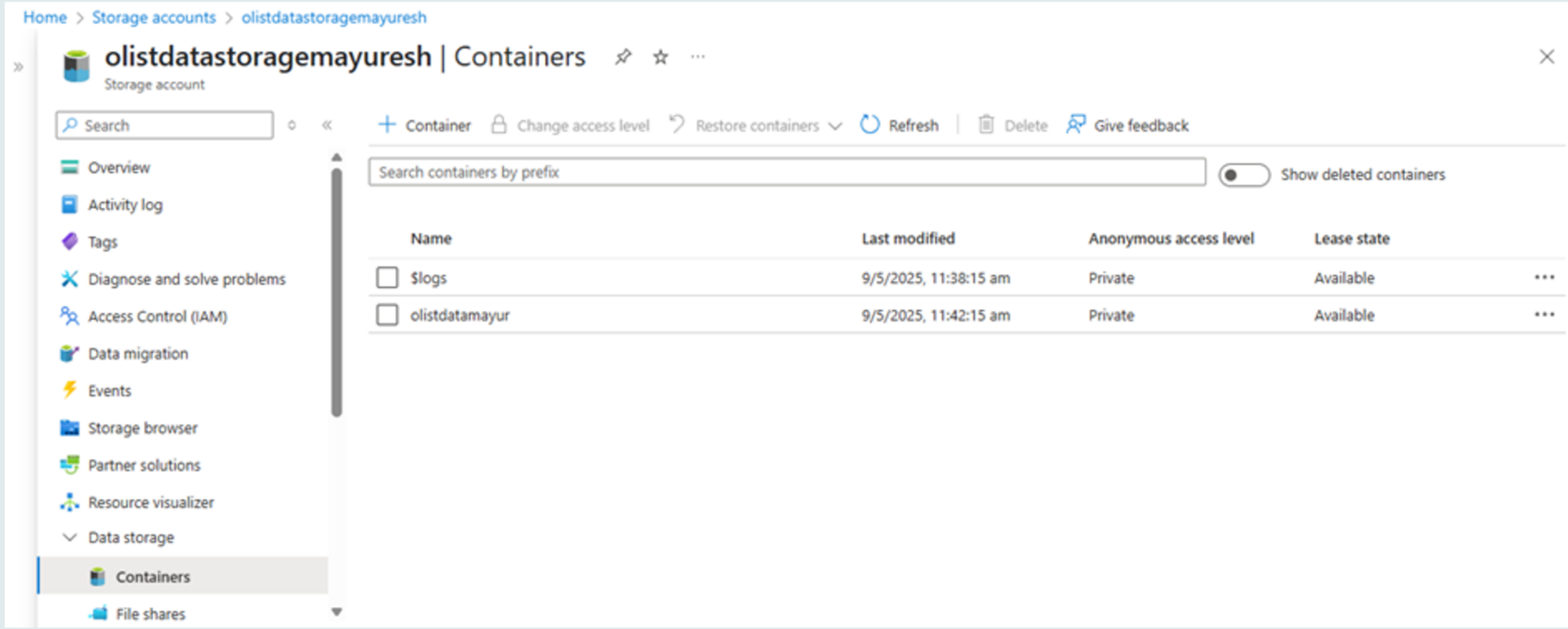
DATA INGESTION WITH AZURE DATA FACTORY



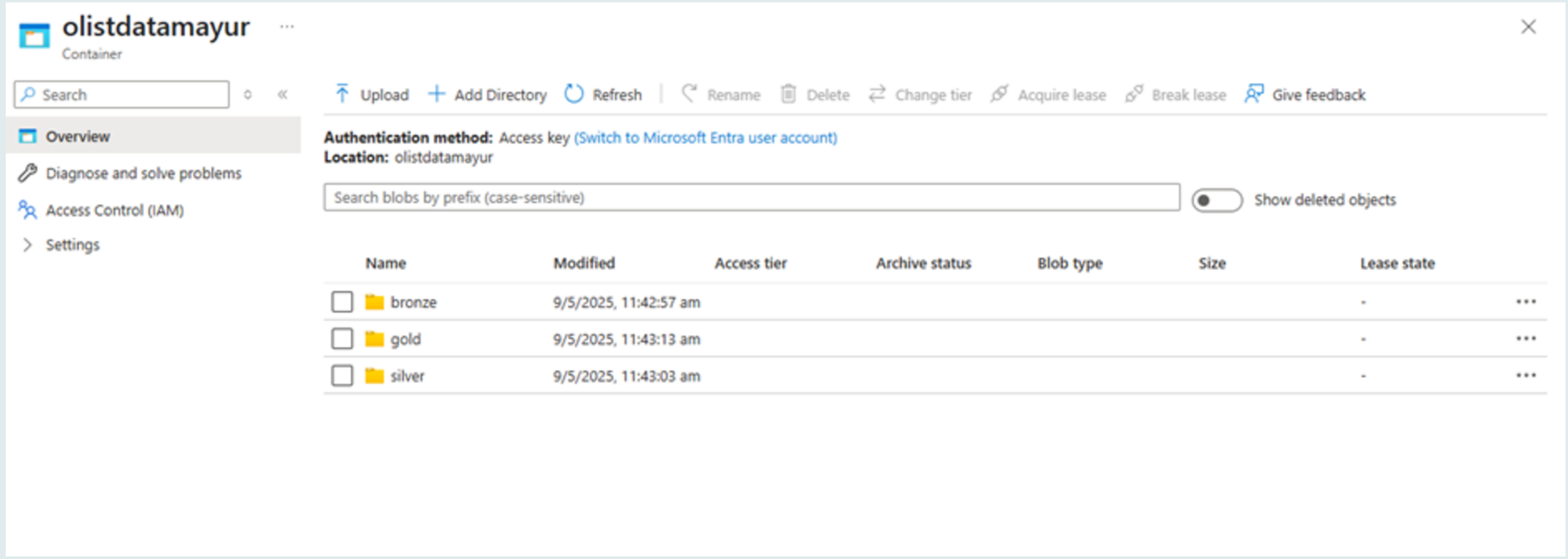
DATA INGESTION WITH AZURE DATA FACTORY



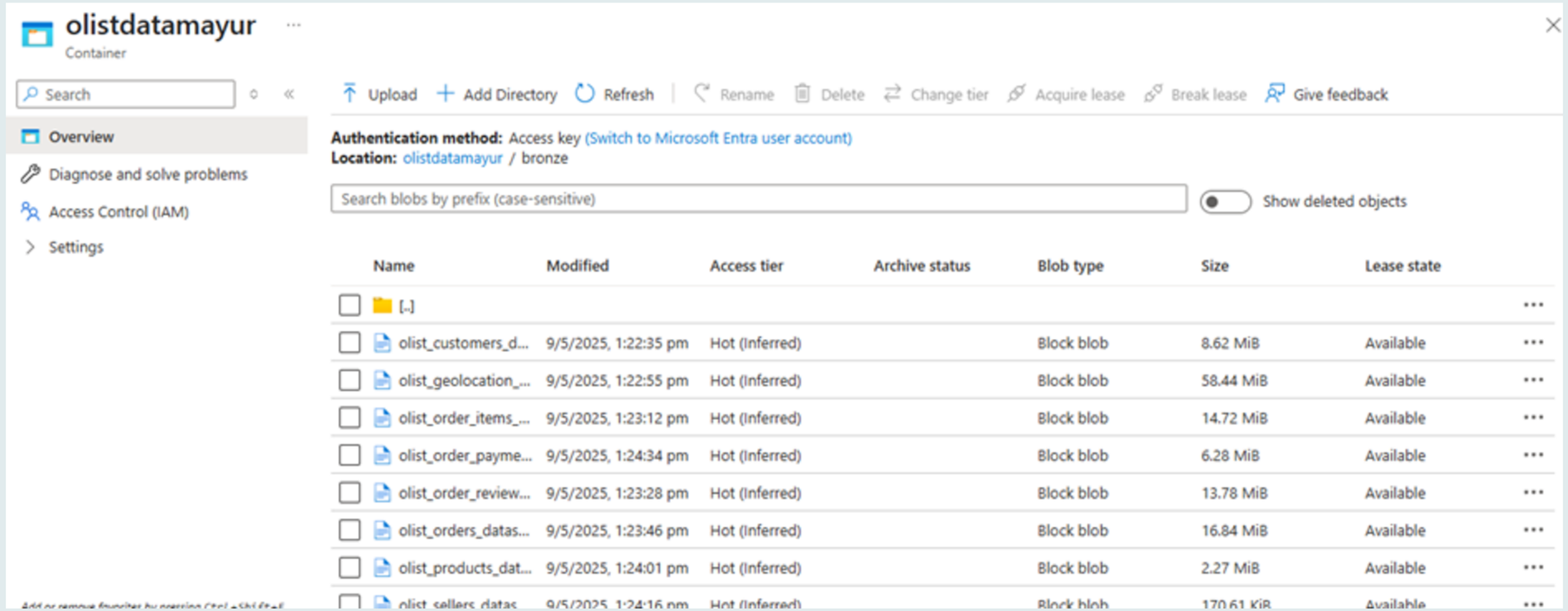
RAW DATA STORAGE - ADLS GEN2



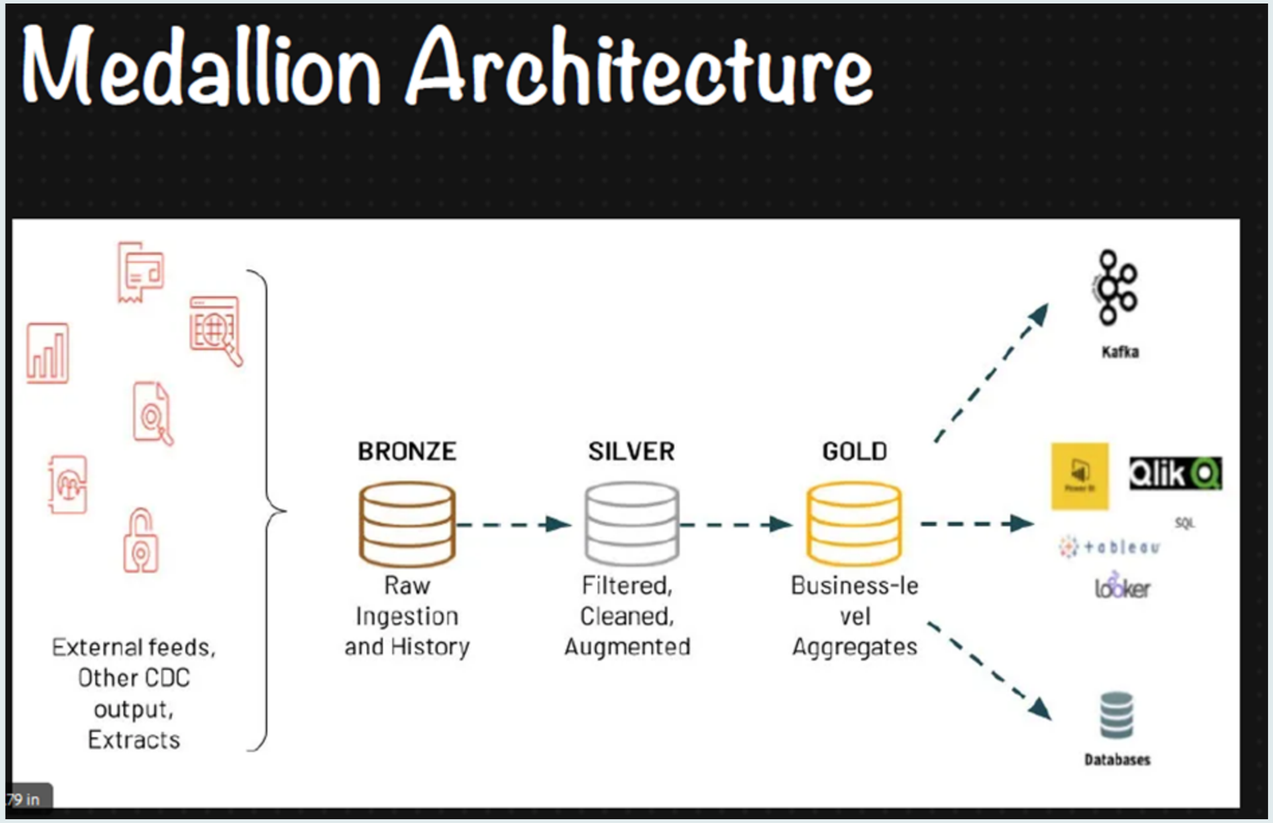
RAW DATA STORAGE - ADLS GEN2



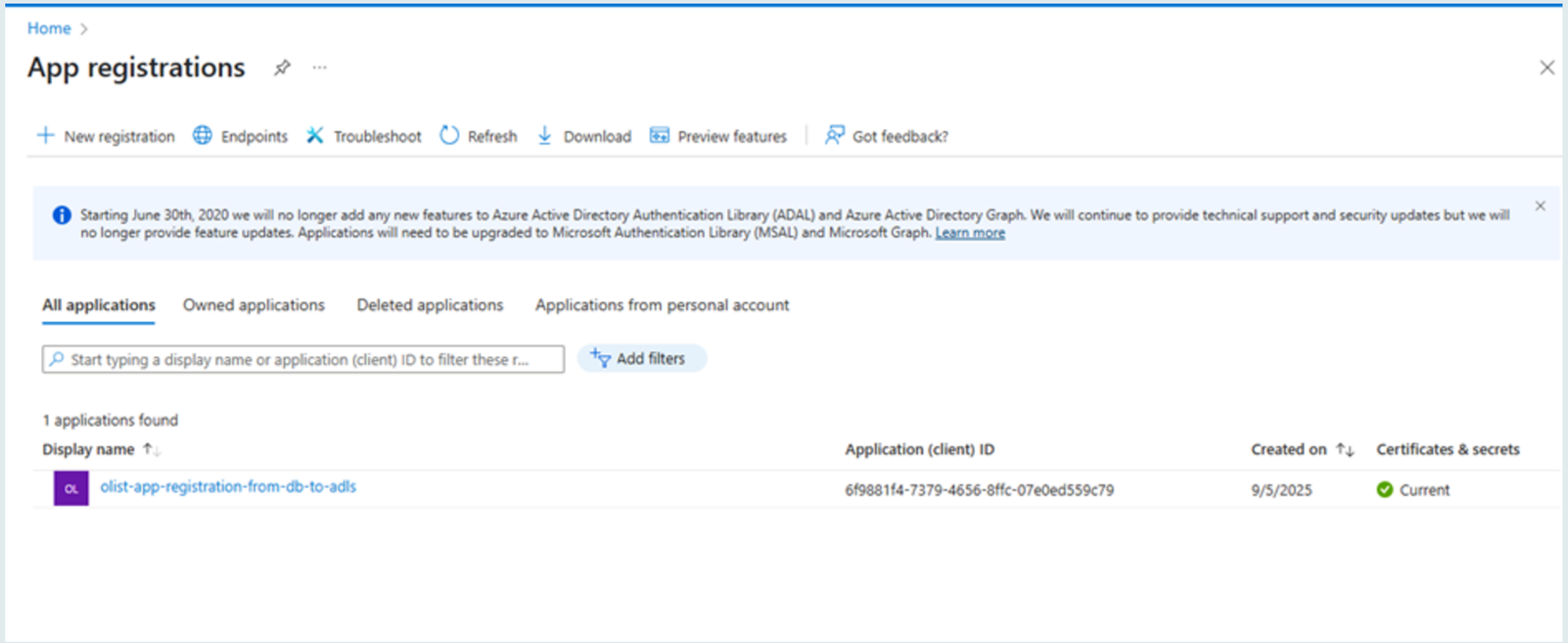
RAW DATA STORAGE - ADLS GEN2



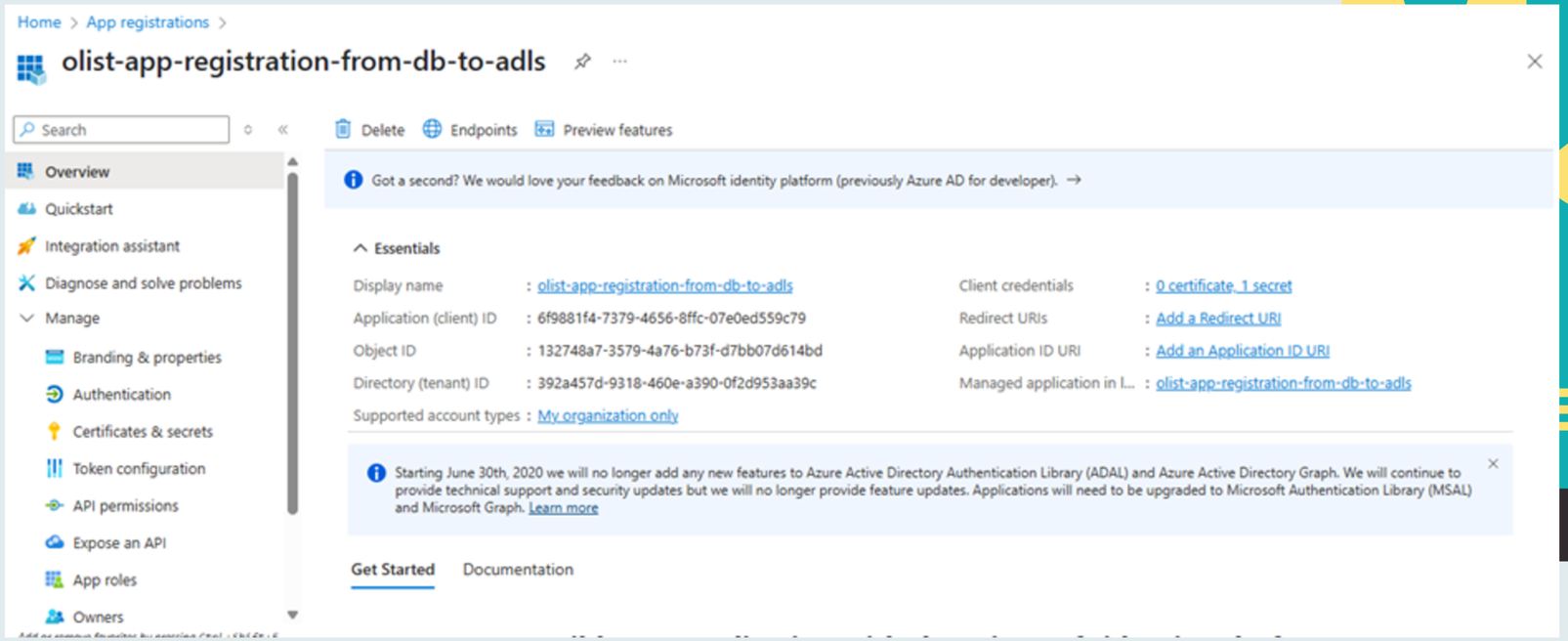
RAW DATA STORAGE - ADLS GEN2



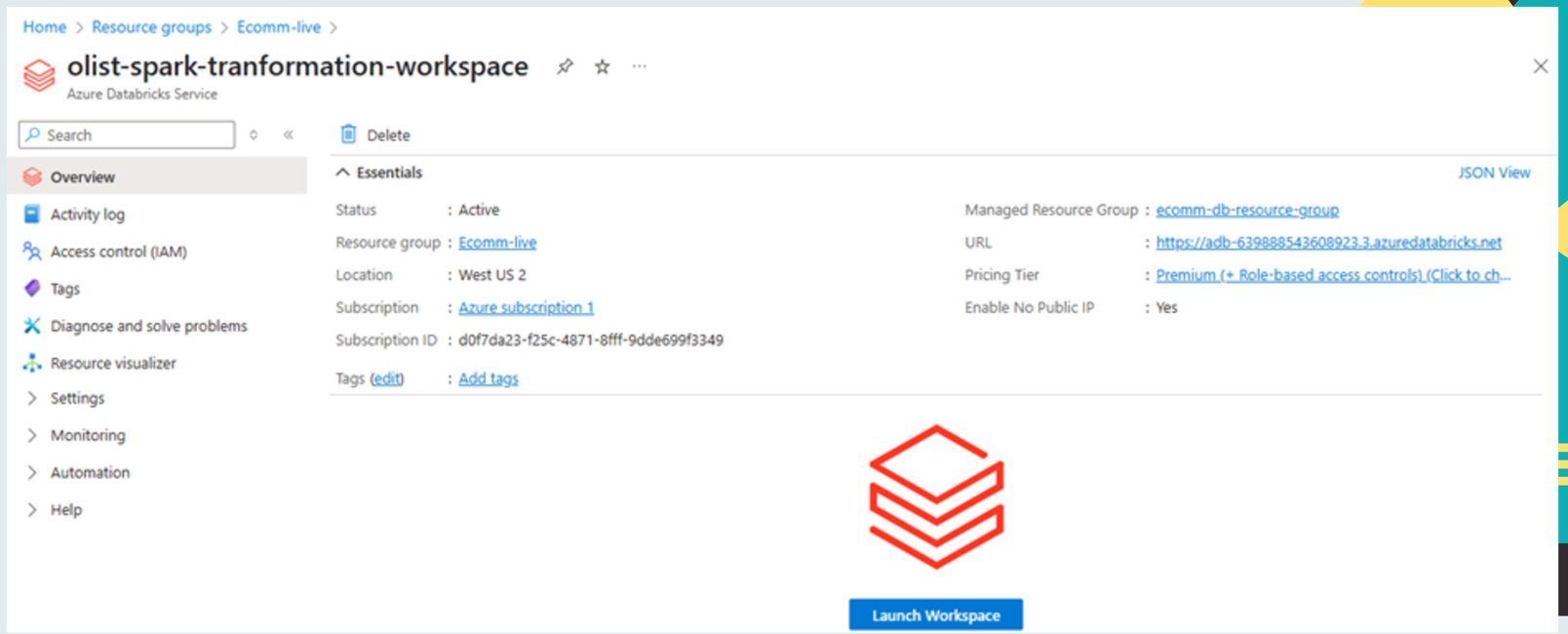
APP REGISTRATIONS



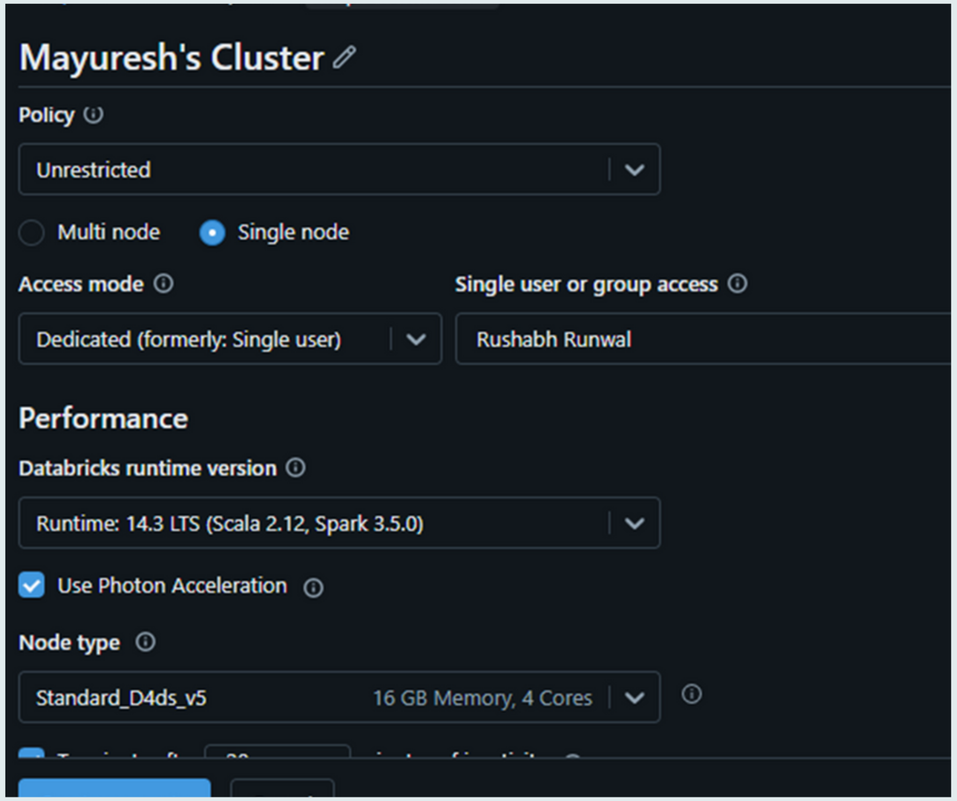
APP REGISTRATION



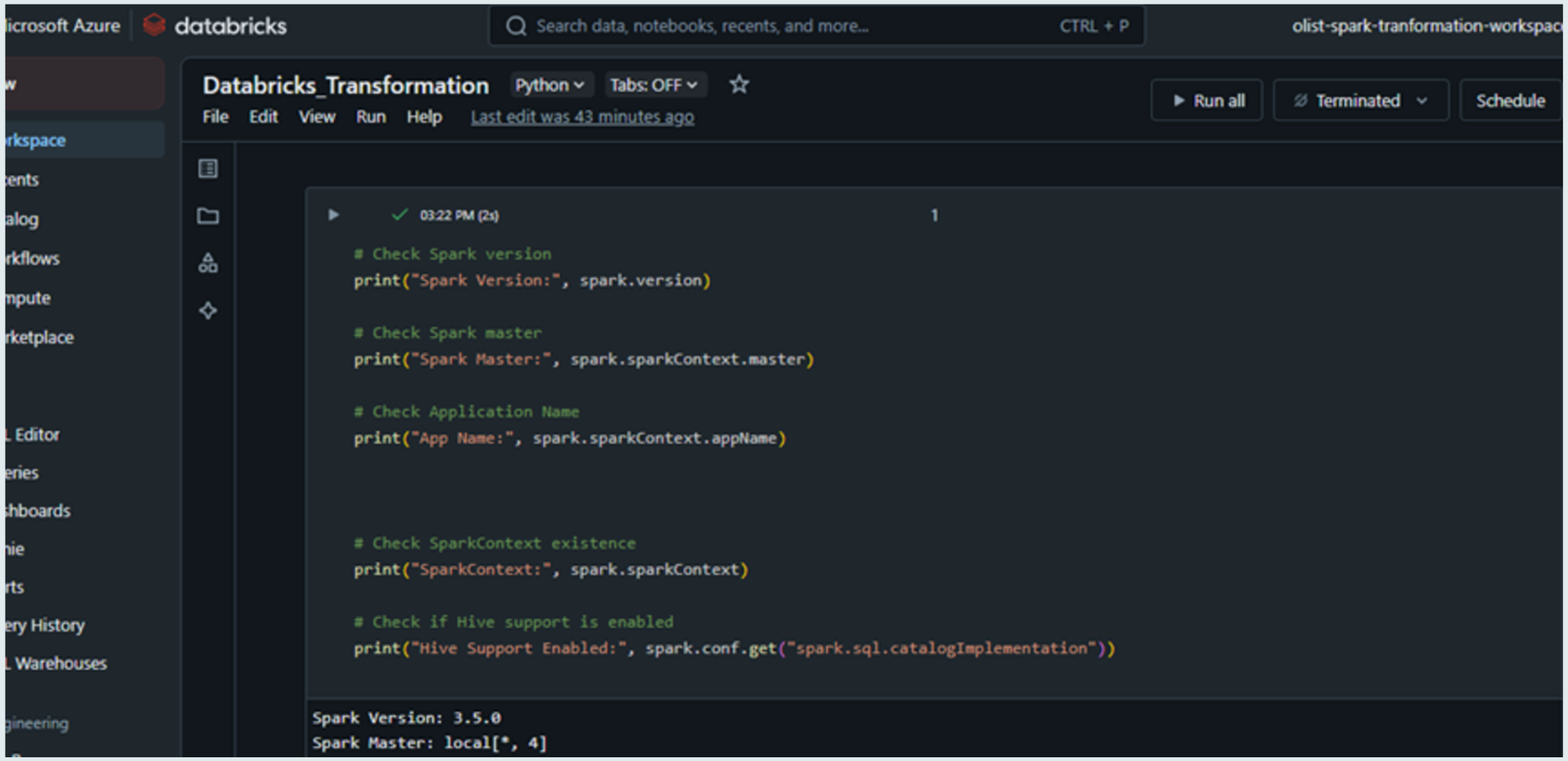
WHY AZURE DATABRICKS?



WHY AZURE DATABRICKS?

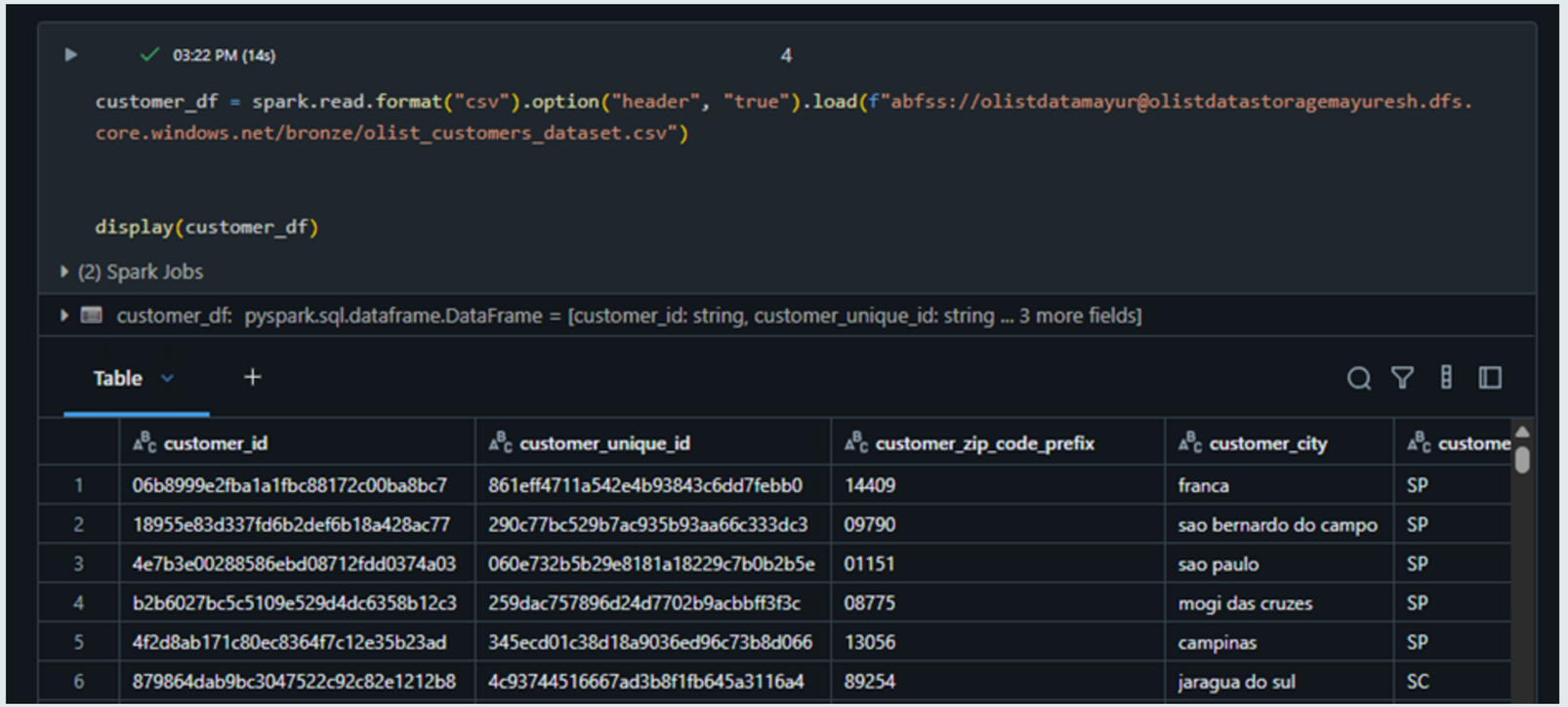


WHY AZURE DATABRICKS?



TRANSFORMATION LOGIC IN DATABRICKS

TRANSFORMATION LOGIC IN DATABRICKS



03:22 PM (14s) 4

```
customer_df = spark.read.format("csv").option("header", "true").load(f"abfss://olistdatamayur@olistdatastoragemayuresh.dfs.
core.windows.net/bronze/olist_customers_dataset.csv")


display(customer_df)
```

(2) Spark Jobs

customer_df: pyspark.sql.dataframe.DataFrame = [customer_id: string, customer_unique_id: string ... 3 more fields]

Table +

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	custome
1	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
2	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	09790	sao bernardo do campo	SP
3	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	01151	sao paulo	SP
4	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	08775	mogi das cruzes	SP
5	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP
6	879864dab9bc3047522c92c82e1212b8	4c93744516667ad3b8f1fb645a3116a4	89254	jaragua do sul	SC

TRANSFORMATION LOGIC IN DATABRICKS

```
✓ 03:22 PM (5s)                                                    5

base_path = "abfss://olistdatamayur@olistdatastoragemayuresh.dfs.core.windows.net/bronze/"
orders_path = base_path + "olist_orders_dataset.csv"
payments_path = base_path + "olist_order_payments_dataset.csv"
reviews_path = base_path + "olist_order_reviews_dataset.csv"
items_path = base_path + "olist_order_items_dataset.csv"
customers_path = base_path + "olist_customers_dataset.csv"
sellers_path = base_path + "olist_sellers_dataset.csv"
geolocation_path = base_path + "olist_geolocation_dataset.csv"
products_path = base_path + "olist_products_dataset.csv"


orders_df = spark.read.format("csv").option("header", "true").load(orders_path)
payments_df = spark.read.format("csv").option("header", "true").load(payments_path)
reviews_df = spark.read.format("csv").option("header", "true").load(reviews_path)
items_df = spark.read.format("csv").option("header", "true").load(items_path)
customers_df = spark.read.format("csv").option("header", "true").load(customers_path)
sellers_df = spark.read.format("csv").option("header", "true").load(sellers_path)
geolocation_df = spark.read.format("csv").option("header", "true").load(geolocation_path)
products_df = spark.read.format("csv").option("header", "true").load(products_path)

▸ (8) Spark Jobs
```

TRANSFORMATION LOGIC IN DATABRICKS

03:24 PM (<1s) 12

```
from pyspark.sql.functions import col,to_date,datediff,current_date,when
```

03:24 PM (2s) 13

```
def clean_datafram(df,name):
    print("Cleaning "+name)
    return df.dropDuplicates().na.drop('all')

orders_df = clean_datafram(orders_df,"Orders")
display(orders_df)
```

(2) Spark Jobs

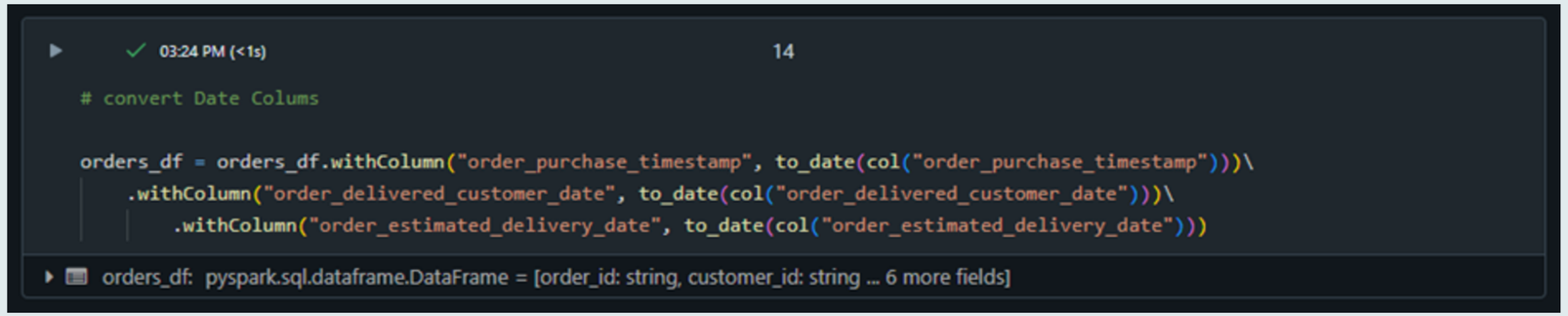
orders_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 6 more fields]

```
Cleaning Orders
```

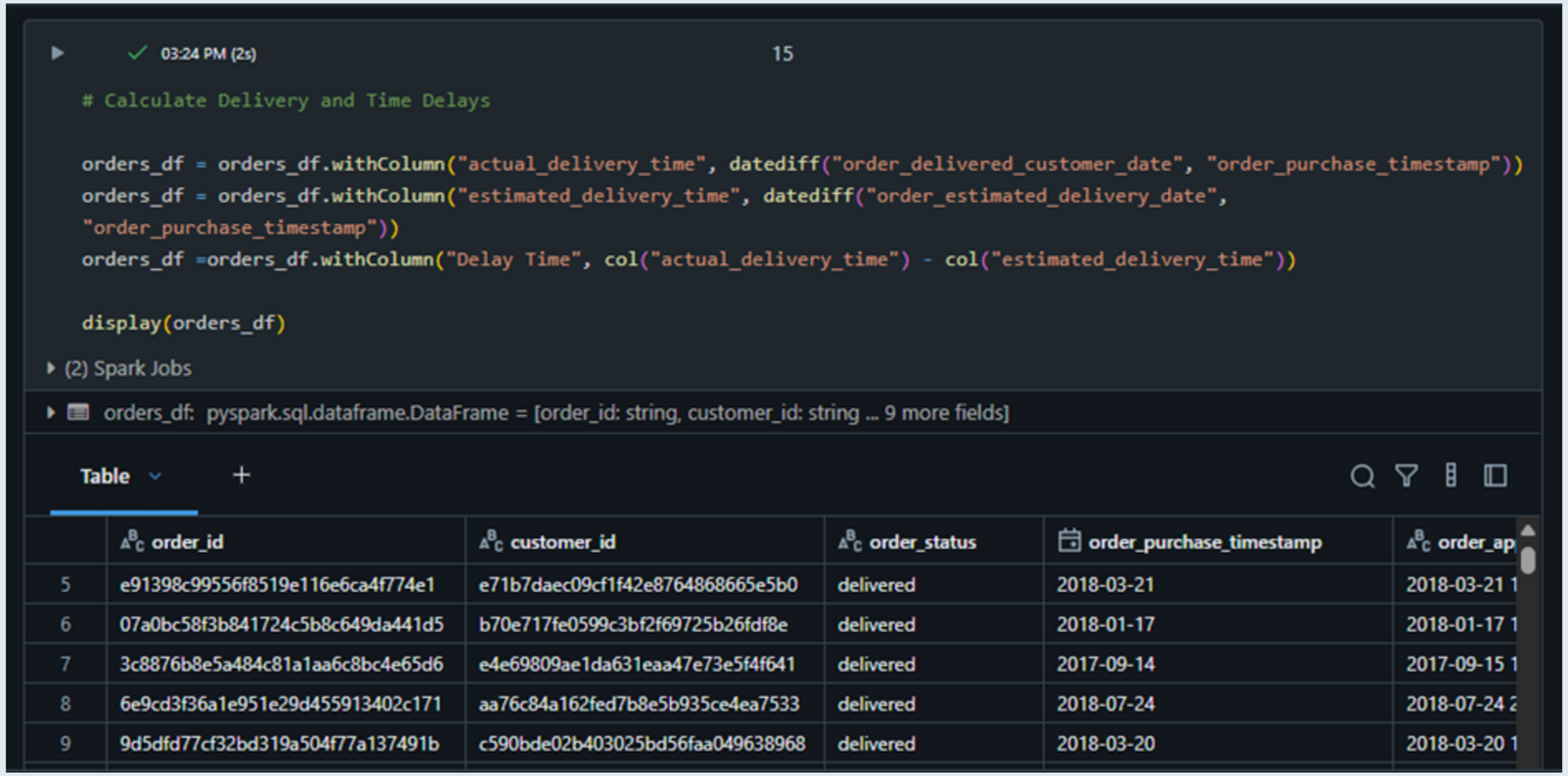
Table +

	order_id	customer_id	order_status	order_purchase_timestamp	order_ap
1	3923e3ade70348985bd2ca389905cf19	6454e6cba392b35aa21527063026fc92	delivered	2018-03-07 23:00:33	2018-03-09 0

TRANSFORMATION LOGIC IN DATABRICKS



TRANSFORMATION LOGIC IN DATABRICKS



03:24 PM (2s) 15

```
# Calculate Delivery and Time Delays

orders_df = orders_df.withColumn("actual_delivery_time", datediff("order_delivered_customer_date", "order_purchase_timestamp"))
orders_df = orders_df.withColumn("estimated_delivery_time", datediff("order_estimated_delivery_date",
"order_purchase_timestamp"))
orders_df =orders_df.withColumn("Delay Time", col("actual_delivery_time") - col("estimated_delivery_time"))

display(orders_df)
```

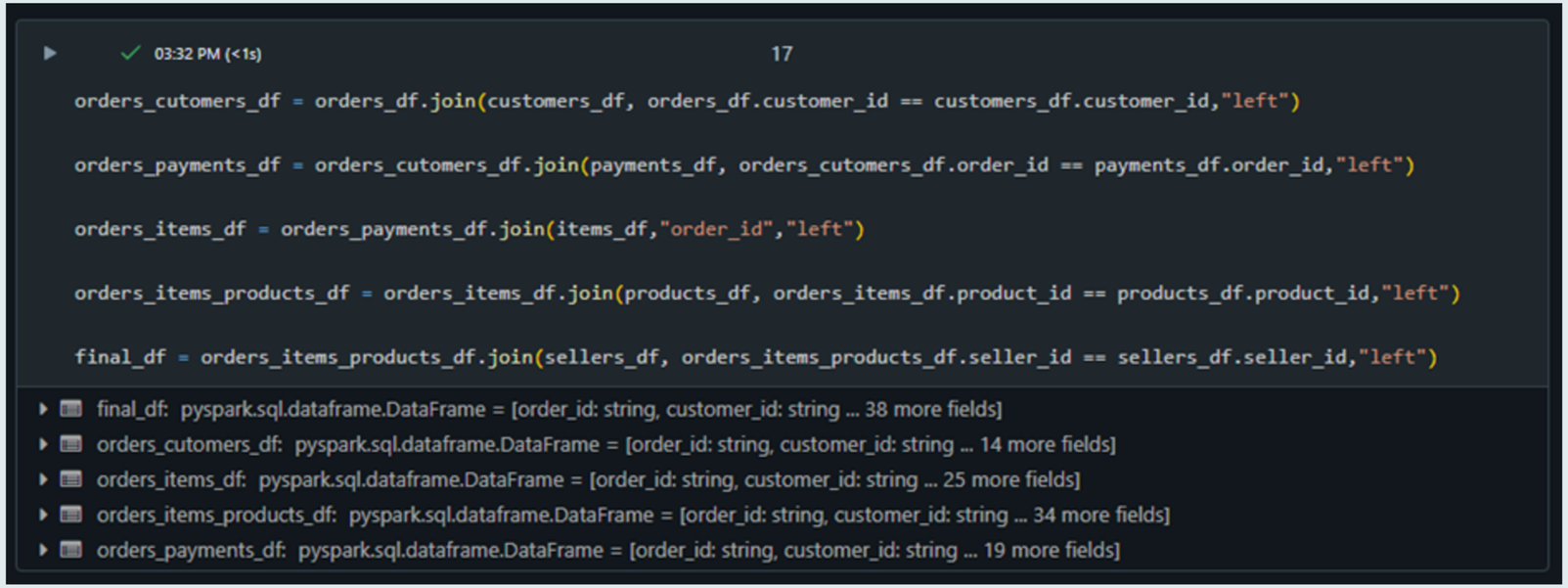
(2) Spark Jobs

orders_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 9 more fields]

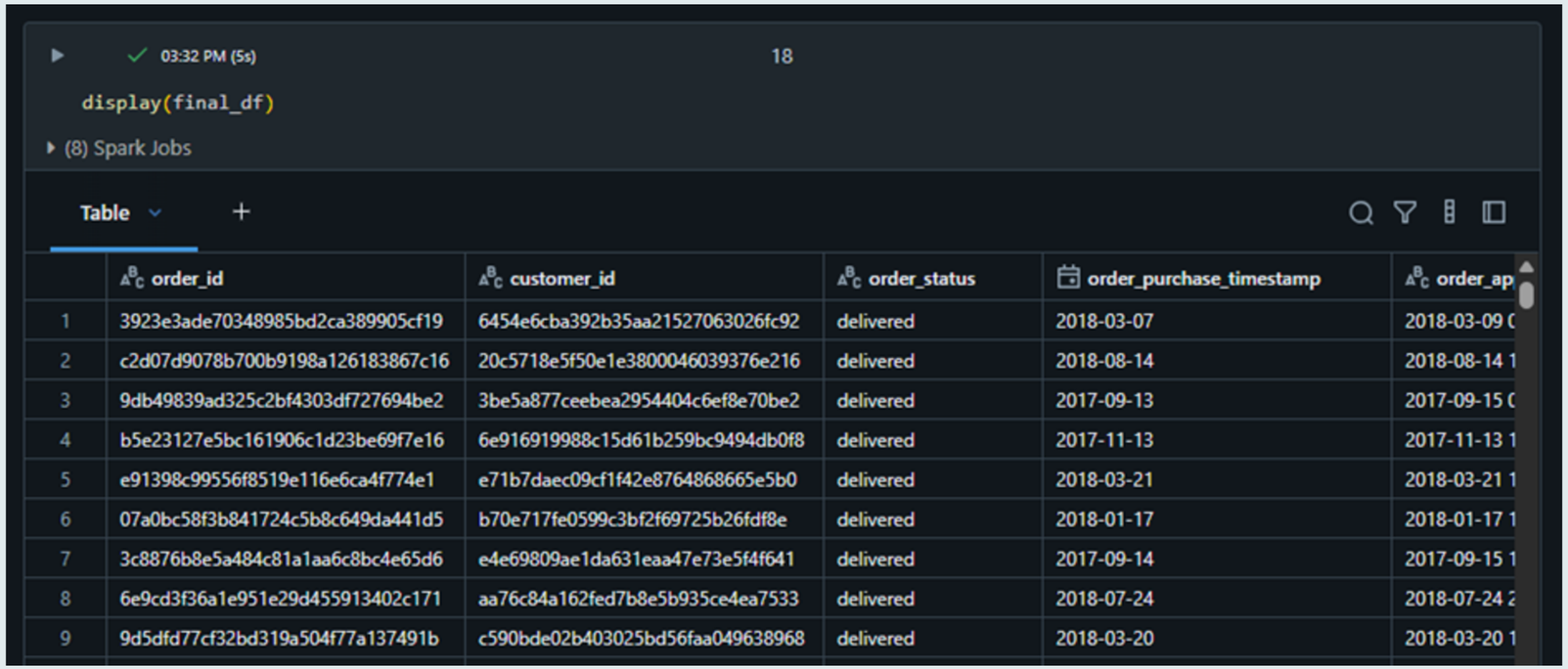
Table +

	order_id	customer_id	order_status	order_purchase_timestamp	order_ap
5	e91398c99556f8519e116e6ca4f774e1	e71b7daec09cf1f42e8764868665e5b0	delivered	2018-03-21	2018-03-21 1
6	07a0bc58f3b841724c5b8c649da441d5	b70e717fe0599c3bf2f69725b26fdf8e	delivered	2018-01-17	2018-01-17 1
7	3c8876b8e5a484c81a1aa6c8bc4e65d6	e4e69809ae1da631eaa47e73e5f4f641	delivered	2017-09-14	2017-09-15 1
8	6e9cd3f36a1e951e29d455913402c171	aa76c84a162fed7b8e5b935ce4ea7533	delivered	2018-07-24	2018-07-24 2
9	9d5dfd77cf32bd319a504f77a137491b	c590bde02b403025bd56faa049638968	delivered	2018-03-20	2018-03-20 1

TRANSFORMATION LOGIC IN DATABRICKS



TRANSFORMATION LOGIC IN DATABRICKS



03:32 PM (5s) 18

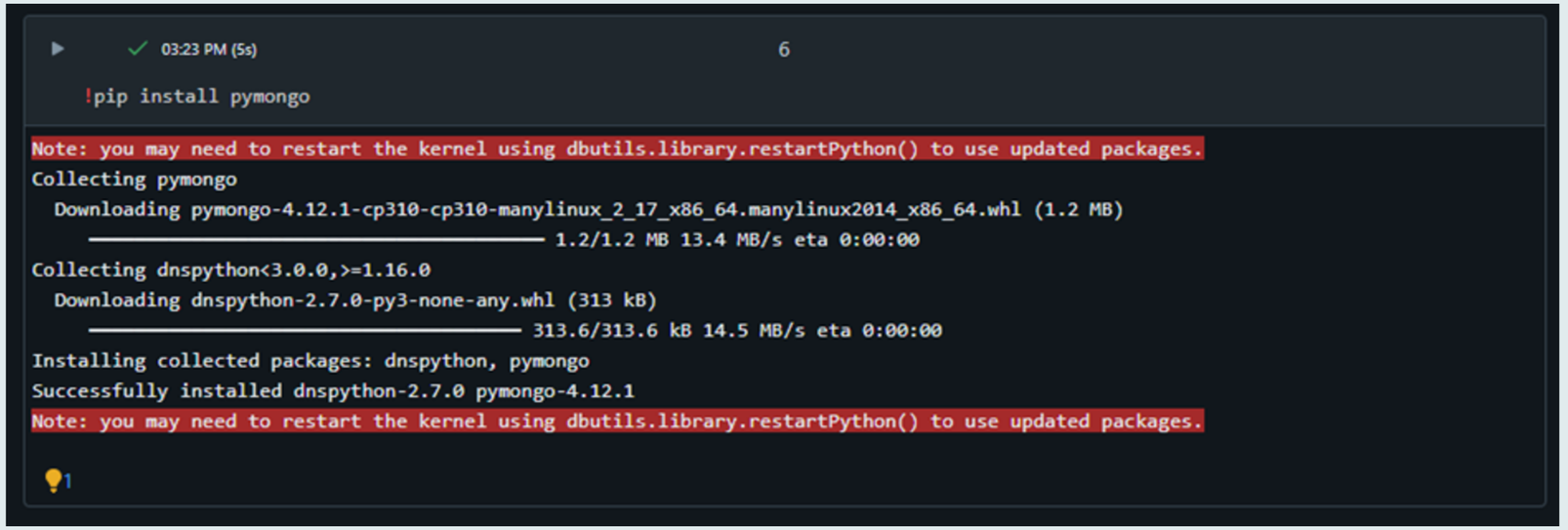
```
display(final_df)
```

(8) Spark Jobs

Table +

	order_id	customer_id	order_status	order_purchase_timestamp	order_ap
1	3923e3ade70348985bd2ca389905cf19	6454e6cba392b35aa21527063026fc92	delivered	2018-03-07	2018-03-09 0
2	c2d07d9078b700b9198a126183867c16	20c5718e5f50e1e3800046039376e216	delivered	2018-08-14	2018-08-14 1
3	9db49839ad325c2bf4303df727694be2	3be5a877ceebea2954404c6ef8e70be2	delivered	2017-09-13	2017-09-15 0
4	b5e23127e5bc161906c1d23be69f7e16	6e916919988c15d61b259bc9494db0f8	delivered	2017-11-13	2017-11-13 1
5	e91398c99556f8519e116e6ca4f774e1	e71b7daec09cf1f42e8764868665e5b0	delivered	2018-03-21	2018-03-21 1
6	07a0bc58f3b841724c5b8c649da441d5	b70e717fe0599c3bf2f69725b26fdf8e	delivered	2018-01-17	2018-01-17 1
7	3c8876b8e5a484c81a1aa6c8bc4e65d6	e4e69809ae1da631eaa47e73e5f4f641	delivered	2017-09-14	2017-09-15 1
8	6e9cd3f36a1e951e29d455913402c171	aa76c84a162fed7b8e5b935ce4ea7533	delivered	2018-07-24	2018-07-24 2
9	9d5dfd77cf32bd319a504f77a137491b	c590bde02b403025bd56faa049638968	delivered	2018-03-20	2018-03-20 1

DATA ENRICHMENT USING NOSQL



DATA ENRICHMENT USING NOSQL

03:23 PM (<1s) 7

```
from pymongo import MongoClient
```

03:23 PM (<1s) 8

```
# importing module
from pymongo import MongoClient
import pandas as pd

hostname = "k-0r6.h.filess.io"
database = "olistDataNoSQL_certainoff"
port = "61004"
username = "olistDataNoSQL_certainoff"
password = "5a3018893ecdea2b54eebbd03ebd9d9e6e19c82e"

uri = "mongodb://" + username + ":" + password + "@" + hostname + ":" + port + "/" + database

# Connect with the portnumber and host
client = MongoClient(uri)

# Access database
mydatabase = client[database]
mydatabase
```

```
Database(MongoClient(host=['k-0r6.h.filess.io:61004'], document_class=dict, tz_aware=False, connect=True), 'olistDataNoSQL_certainof
```

DATA ENRICHMENT USING NOSQL

DATA ENRICHMENT USING NOSQL

DATA ENRICHMENT USING NOSQL

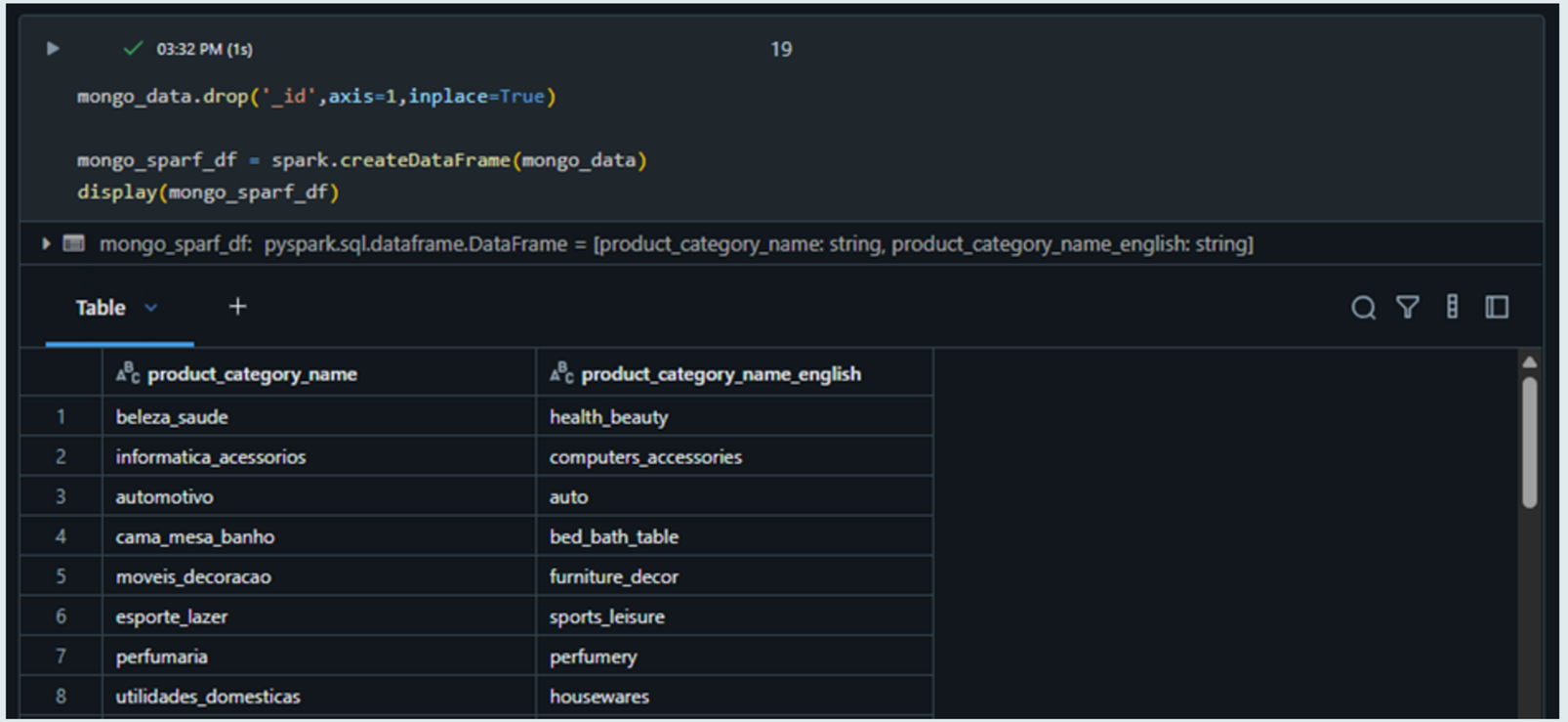
03:24 PM (<1s) 11

mongo_data

	_id	product_category_name	product_category_name_english
0	681e6aa7566250adaaa34634	beleza_saude	health_beauty
1	681e6aa7566250adaaa34635	informatica_acessorios	computers_accessories
2	681e6aa7566250adaaa34636	automotivo	auto
3	681e6aa7566250adaaa34637	cama_mesa_banho	bed_bath_table
4	681e6aa7566250adaaa34638	moveis_decoracao	furniture_decor
...	...	...	...
66	681e6aa7566250adaaa34676	flores	flowers
67	681e6aa7566250adaaa34677	artes_e_artesanato	arts_and_craftmanship
68	681e6aa7566250adaaa34678	fraldas_higiene	diapers_and_hygiene
69	681e6aa7566250adaaa34679	fashion_roupa_infanto_juvenil	fashion_childrens_clothes
70	681e6aa7566250adaaa3467a	seguros_e_servicos	security_and_services

71 rows × 3 columns

DATA ENRICHMENT USING NOSQL



03:32 PM (1s) 19

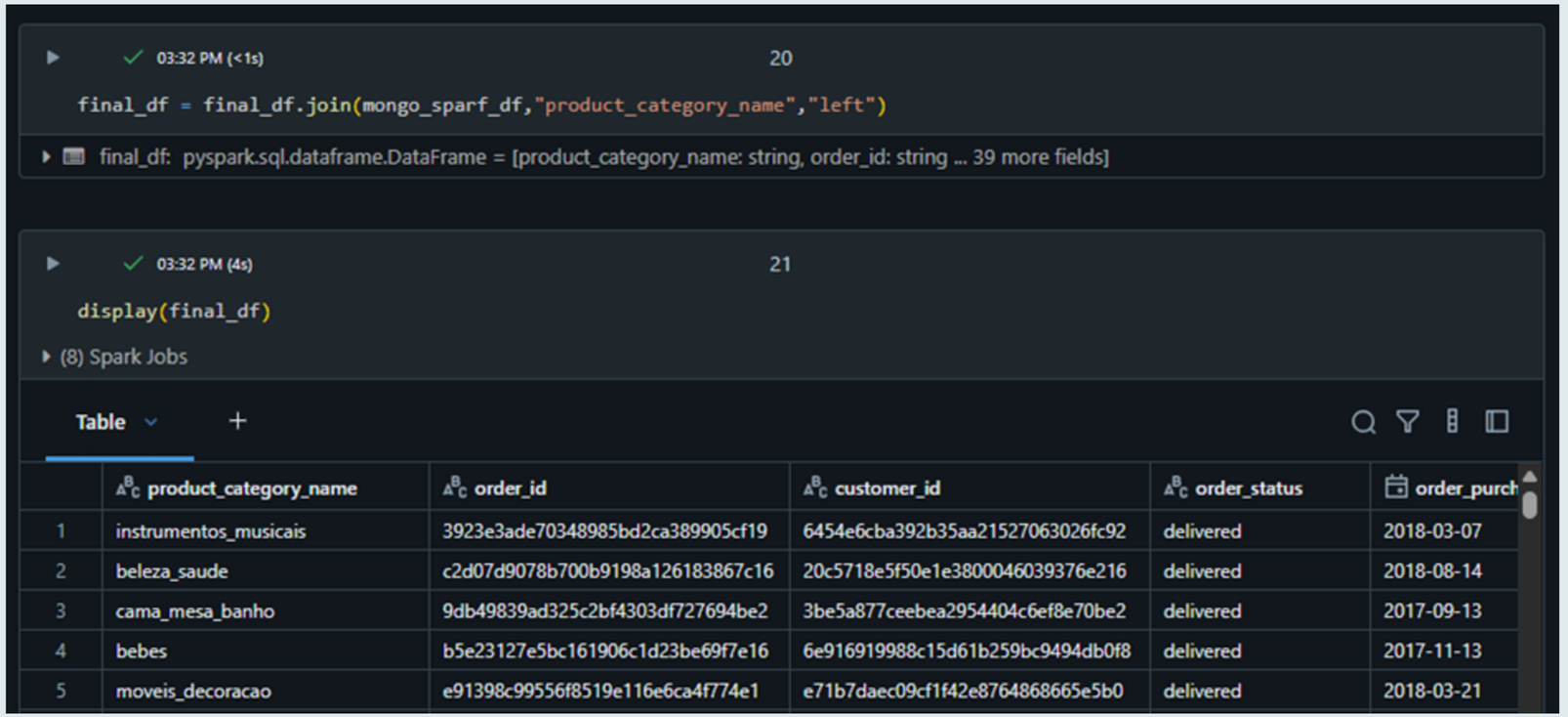
```
mongo_data.drop('_id',axis=1,inplace=True)

mongo_sparf_df = spark.createDataFrame(mongo_data)
display(mongo_sparf_df)
```

mongo_sparf_df: pyspark.sql.dataframe.DataFrame = [product_category_name: string, product_category_name_english: string]

Table +

	product_category_name	product_category_name_english
1	beleza_saude	health_beauty
2	informatica_acessorios	computers_accessories
3	automotivo	auto
4	cama_mesa_banho	bed_bath_table
5	moveis_decoracao	furniture_decor
6	esporte_lazer	sports_leisure
7	perfumaria	perfumery
8	utilidades_domesticas	housewares

DATA ENRICHMENT USING NOSQL



03:32 PM (<1s) 20

```
final_df = final_df.join(mongo_sparf_df,"product_category_name","left")
```

final_df: pyspark.sql.dataframe.DataFrame = [product_category_name: string, order_id: string ... 39 more fields]

03:32 PM (4s) 21

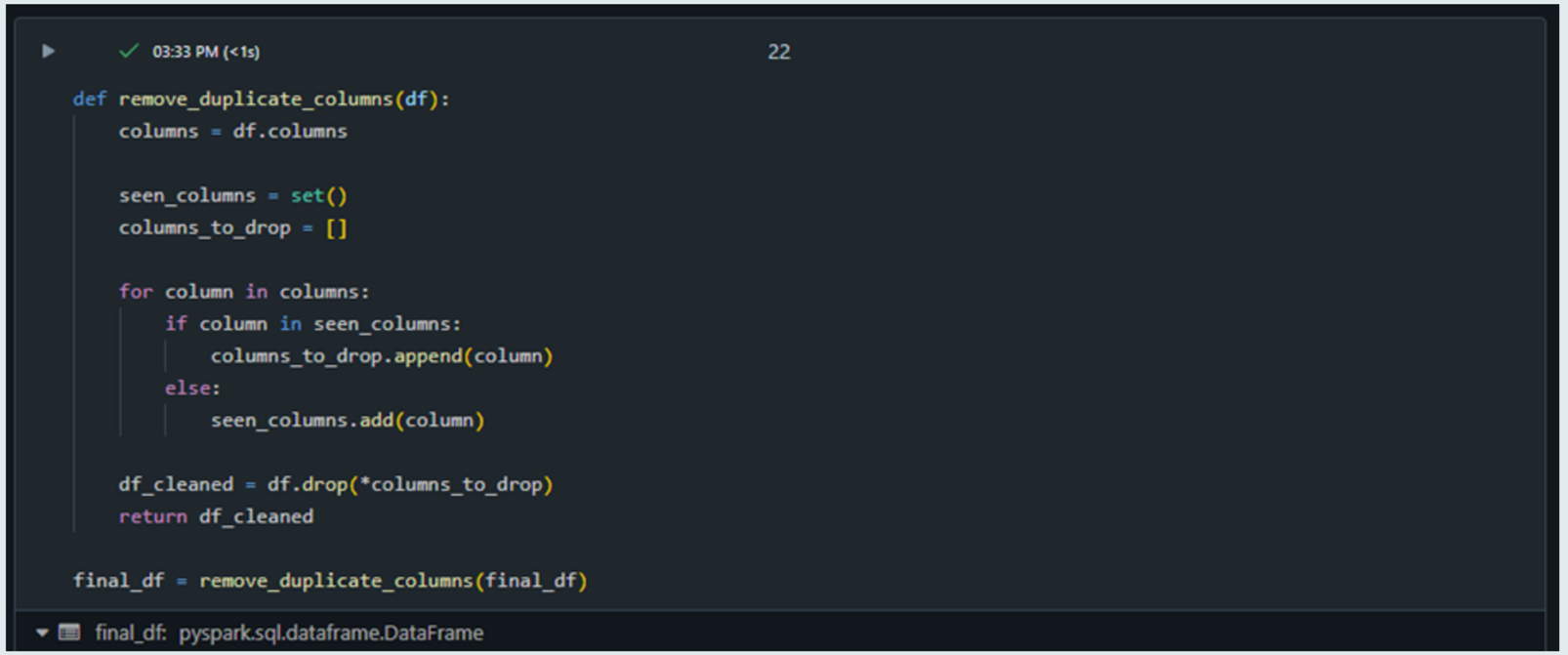
```
display(final_df)
```

(8) Spark Jobs

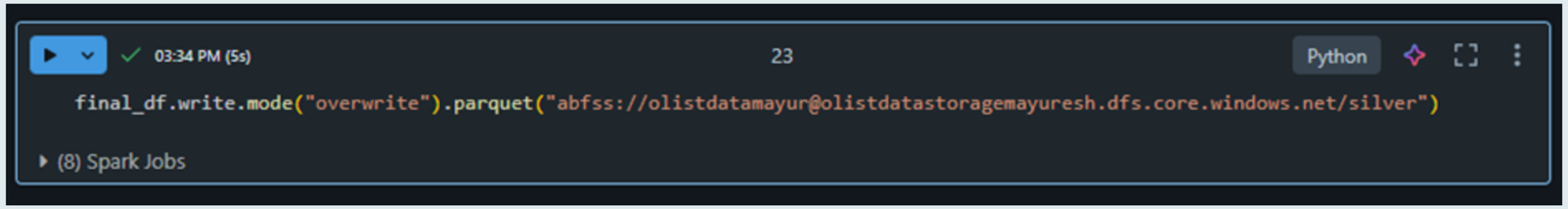
Table +

	product_category_name	order_id	customer_id	order_status	order_purch
1	instrumentos_musicais	3923e3ade70348985bd2ca389905cf19	6454e6cba392b35aa21527063026fc92	delivered	2018-03-07
2	beleza_saude	c2d07d9078b700b9198a126183867c16	20c5718e5f50e1e3800046039376e216	delivered	2018-08-14
3	cama_mesa_banho	9db49839ad325c2bf4303df727694be2	3be5a877ceebea2954404c6ef8e70be2	delivered	2017-09-13
4	bebes	b5e23127e5bc161906c1d23be69f7e16	6e916919988c15d61b259bc9494db0f8	delivered	2017-11-13
5	moveis_decoracao	e91398c99556f8519e116e6ca4f774e1	e71b7daec09cf1f42e8764868665e5b0	delivered	2018-03-21

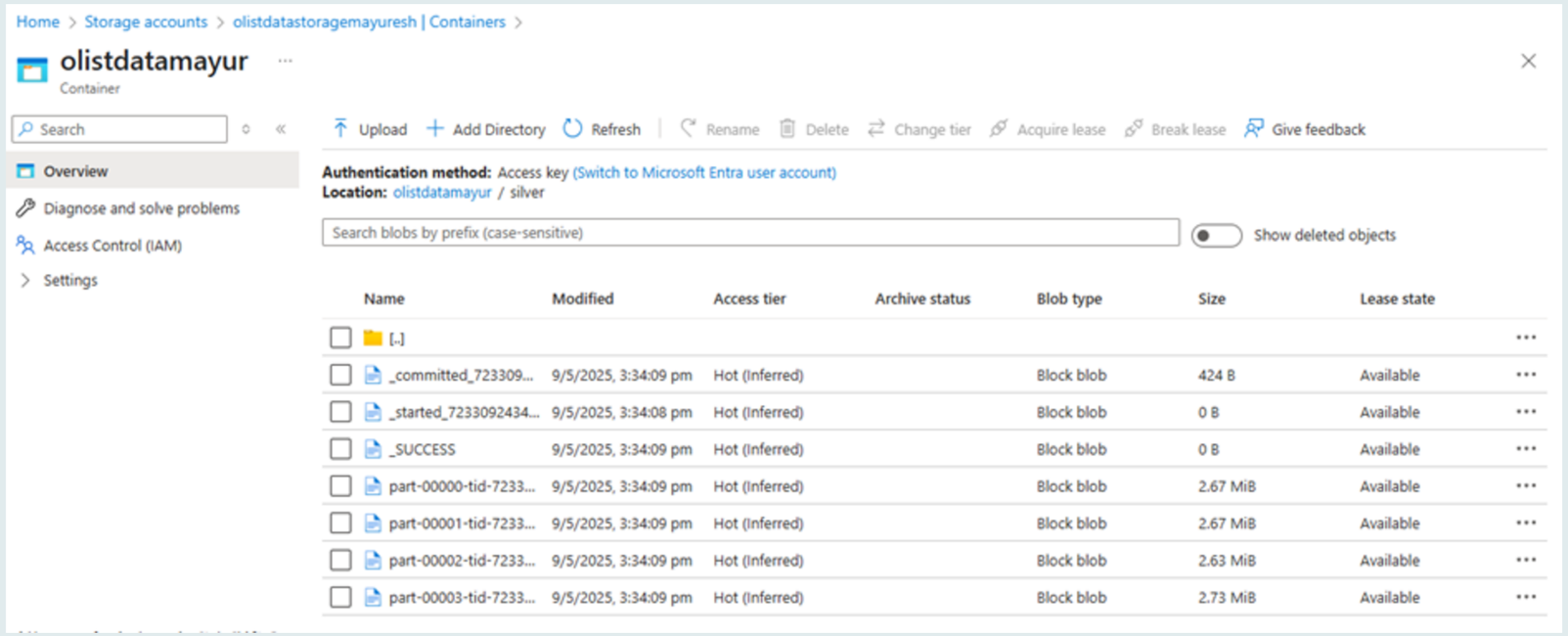
DATA ENRICHMENT USING NOSQL



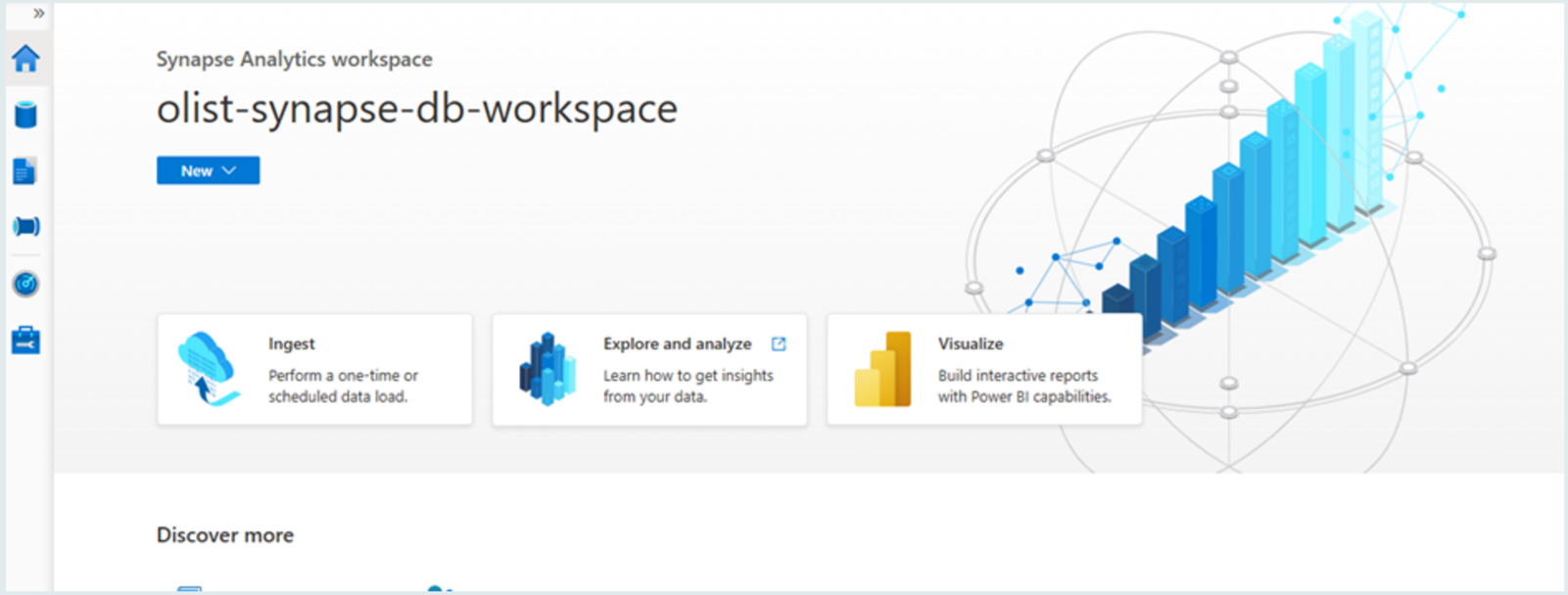
TRANSFORMED DATA STORAGE



TRANSFORMED DATA STORAGE



SYNAPSE ANALYTICS INTEGRATION



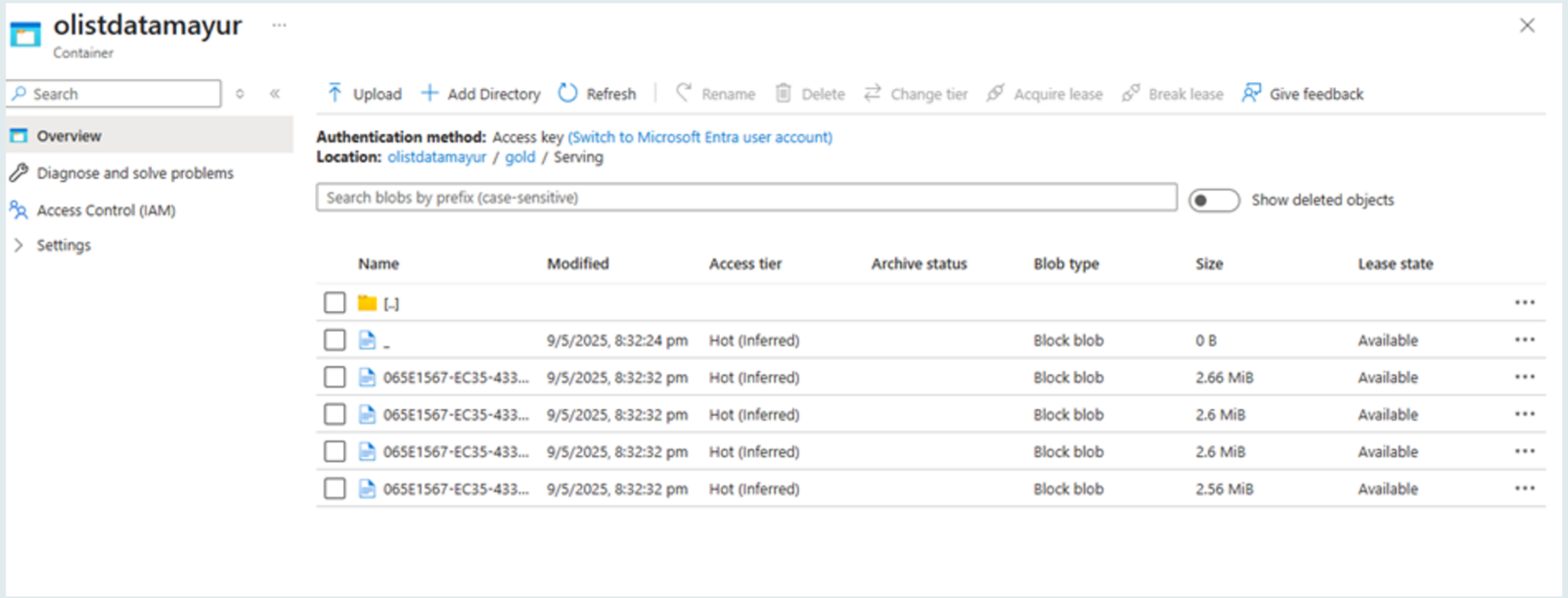
SYNAPSE ANALYTICS INTEGRATION

```
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://synapseolistdefault.dfs.core.windows.net/synapseolistfs/customers.csv',
        FORMAT = 'CSV',
        PARSER_VERSION = '2.0'
    ) AS [result]



SELECT
    *
FROM
    OPENROWSET(
        BULK 'https://olistdatastorageaccount.dfs.core.windows.net/olistdata/silver/',
        FORMAT = 'PARQUET'
    ) AS result1

--- https://olistdatastorageaccount.blob.core.windows.net/olistdata/silver



create schema gold

create view gold.final
```

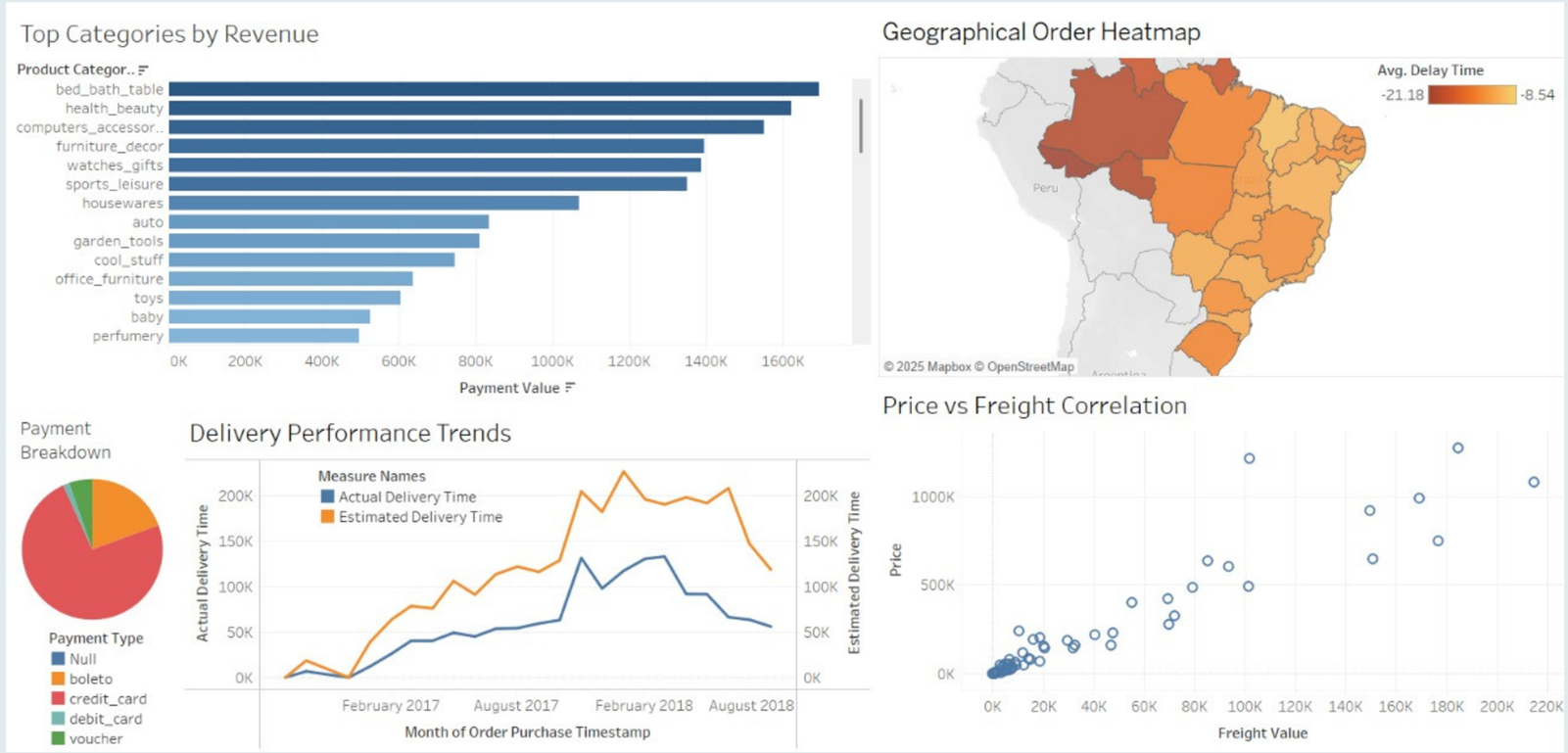
SYNAPSE ANALYTICS INTEGRATION

```
-- CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'd8TySvmWr4rPBUk';
-- CREATE DATABASE SCOPED CREDENTIAL mayankadmin WITH IDENTITY = 'Managed Identity';

-- select * from sys.database_credentials

CREATE EXTERNAL FILE FORMAT extfileformat WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
);

CREATE EXTERNAL DATA SOURCE goldlayer WITH (
    LOCATION = 'https://olistdatastorageaccount.dfs.core.windows.net/olistdata/gold/',
    CREDENTIAL = mayankadmin
);


CREATE EXTERNAL TABLE gold.finaltable WITH (
        LOCATION = 'Serving',
        DATA_SOURCE = goldlayer,
        FILE_FORMAT = extfileformat
) AS
SELECT * FROM gold.final2;



select * from gold.finaltable

select * from gold.final2
```

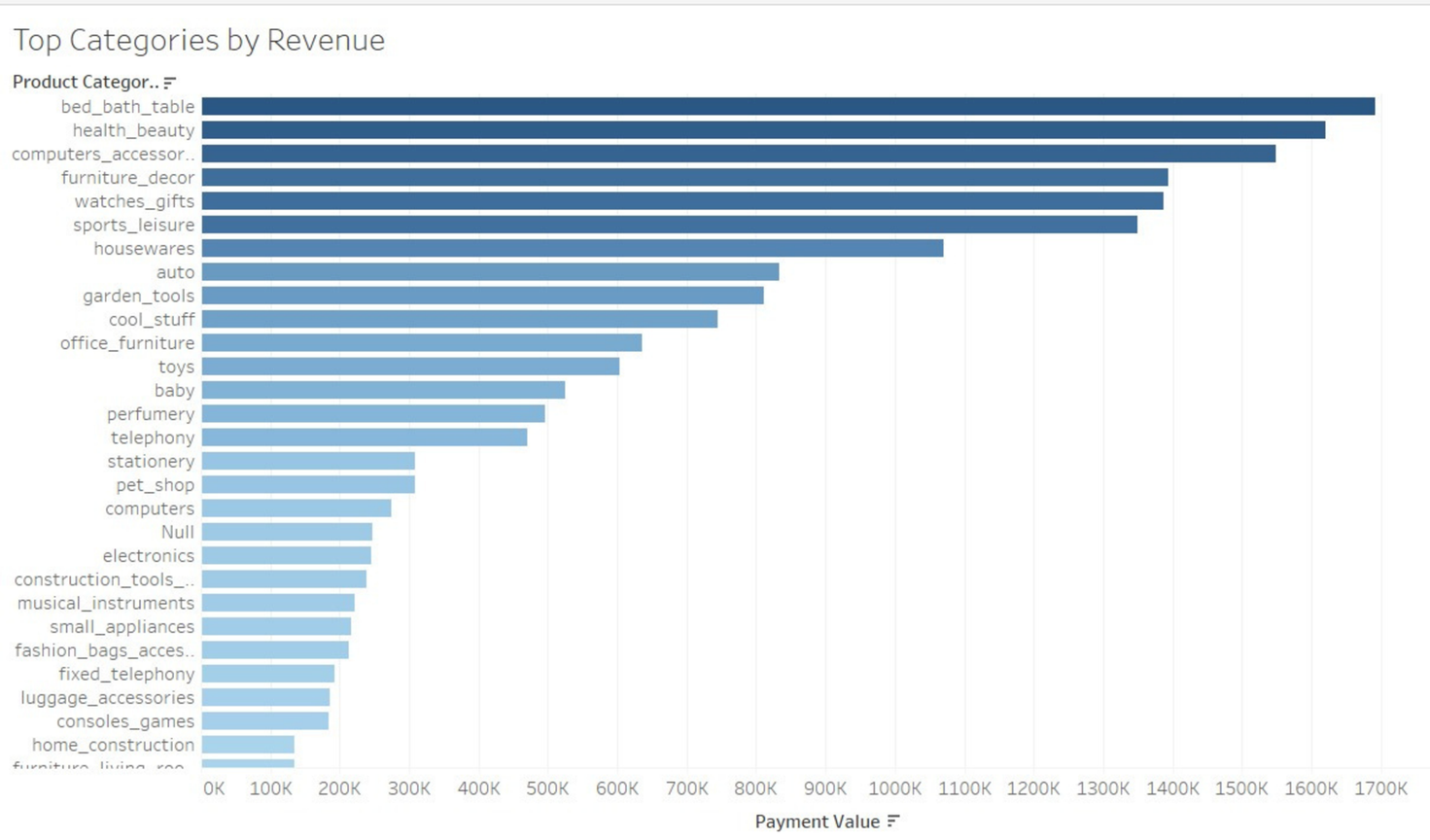
SYNAPSE ANALYTICS INTEGRATION



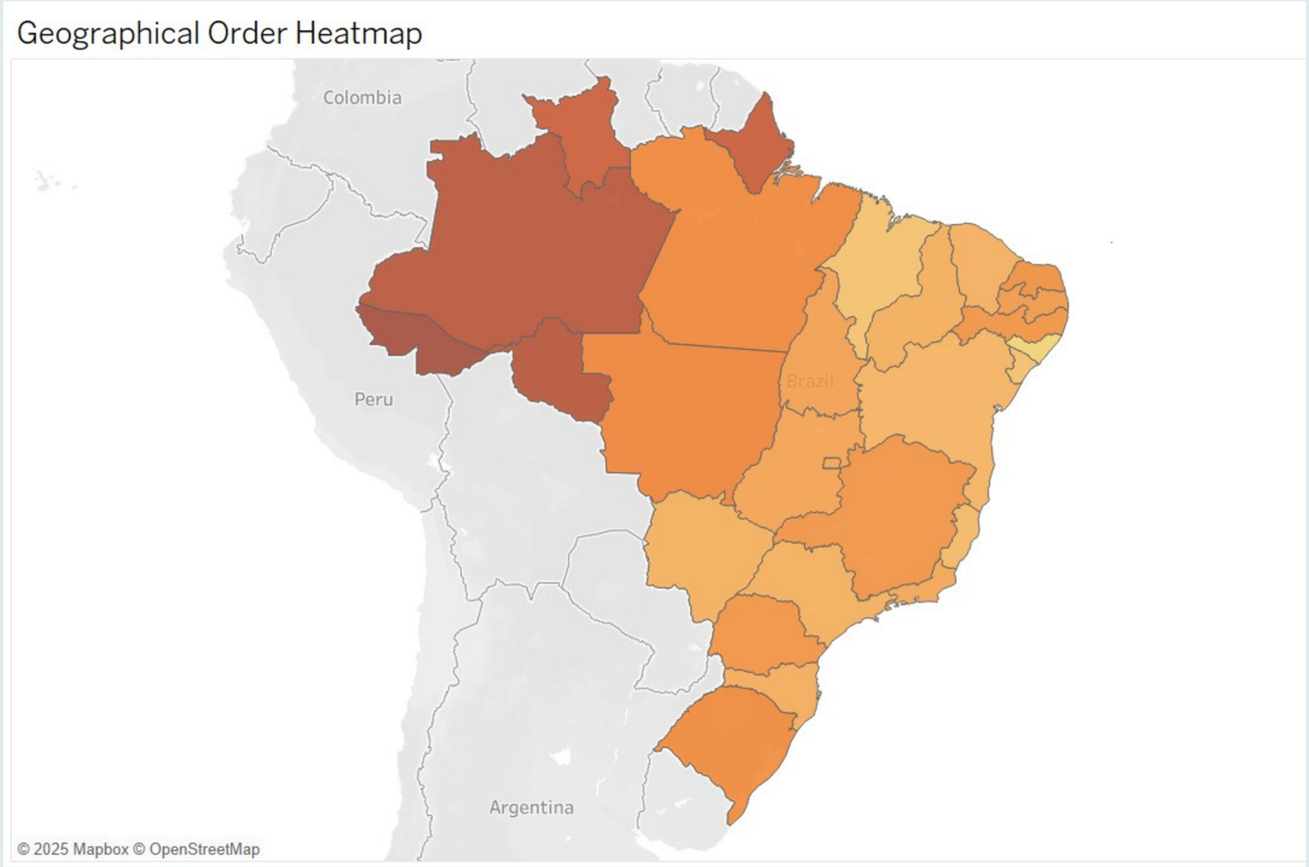
VISUALIZATION LAYER



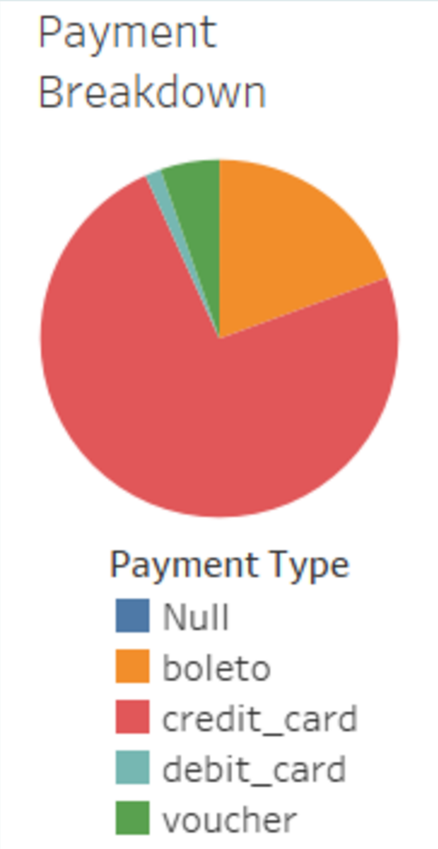
DASHBOARD INSIGHTS



DASHBOARD INSIGHTS

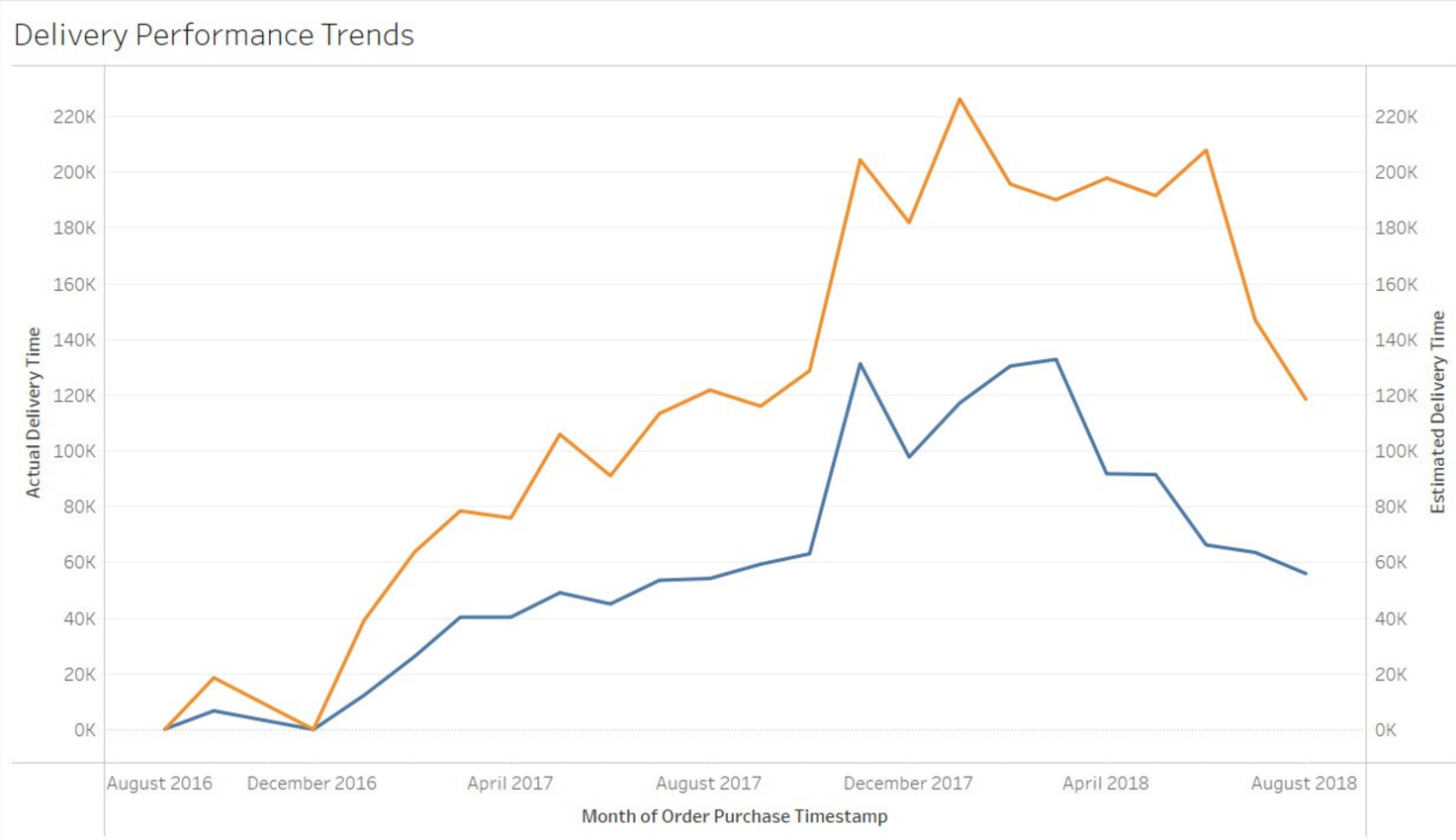


DASHBOARD INSIGHTS



DASHBOARD INSIGHTS

DASHBOARD INSIGHTS



PERFORMANCE & COST OPTIMIZATION

- Spark caching, partitioning, and broadcast joins in Databricks
- Parquet compression in ADLS
- Minimized Synapse compute usage with materialized views
- Scalable and cost-effective setup

CHALLENGES ENCOUNTERED

- Inconsistent or missing data across Olist tables
- Setting up secure ADF connections to GitHub
- Initial slow joins in Databricks due to large shuffle
- Integrating NoSQL into Spark transformation pipeline

FUTURE ENHANCEMENTS

- Add real-time data streaming using Azure Event Hub or Kafka.
- Integrate machine learning models for predictive analytics.
- Set up alerting and monitoring with Azure Monitor and Log Analytics.

THANK YOU!

ANY QUESTIONS?