

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

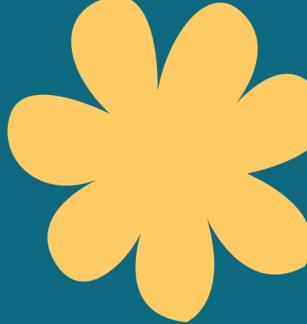
PROJECT OBJECTIVES

- 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

DATA SET 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 DATABRICKS



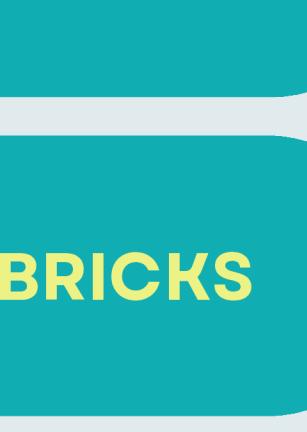
NoSQL DB (MongoDB)



AZURE DATA LAKE GEN2



AZURE SYNAPSE ANALYTICS



SQL DB (MySQL)

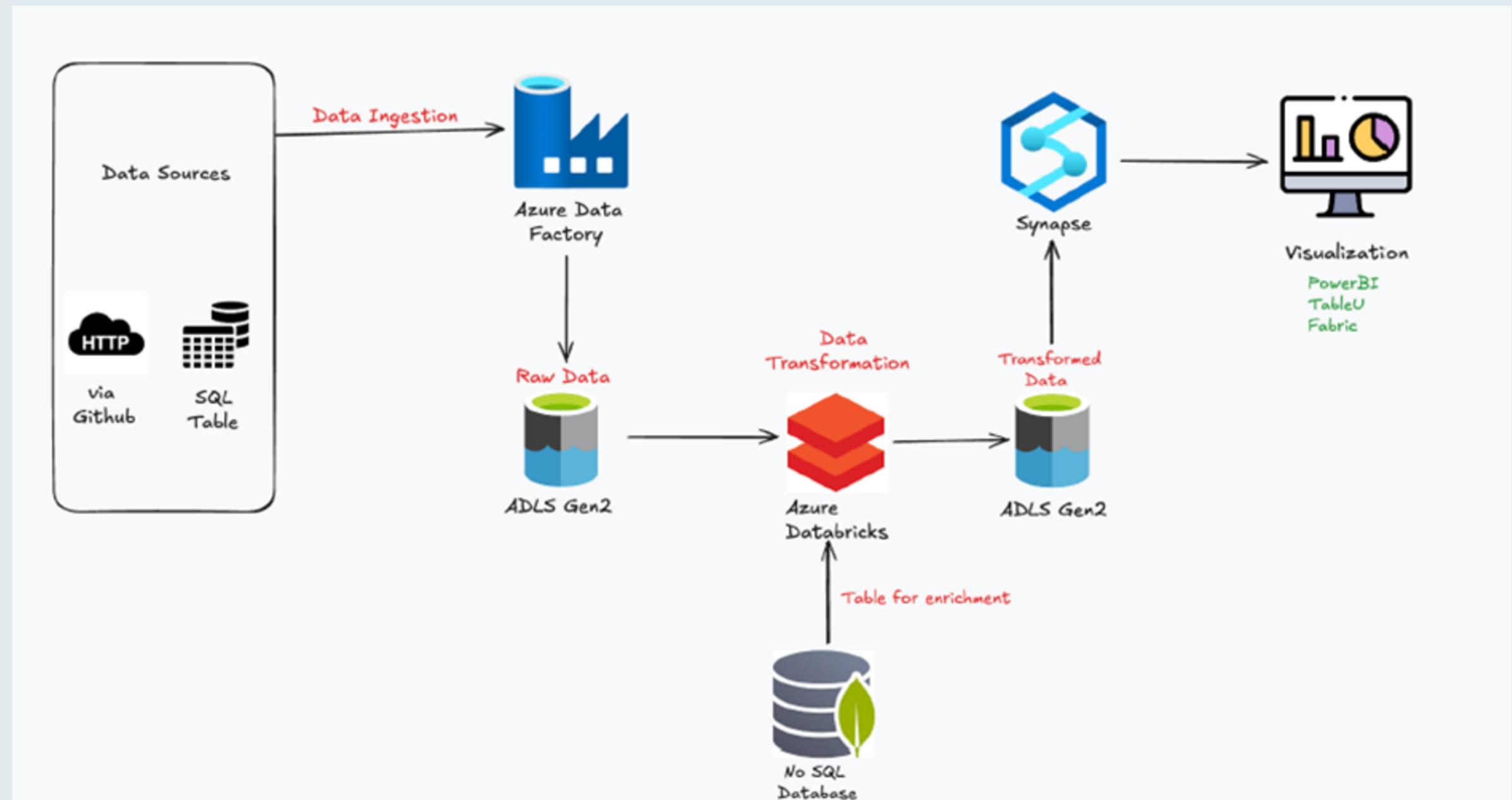


PySpark

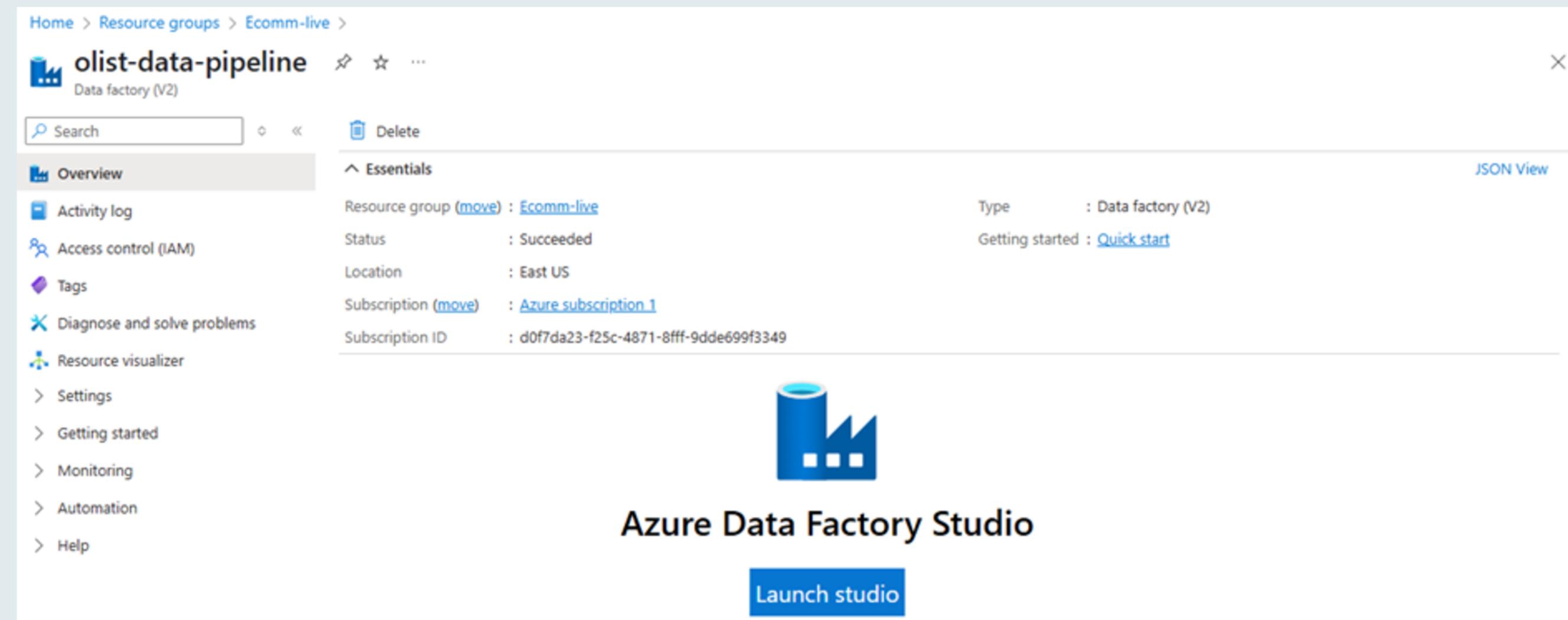


Visualization Tools: Tableau

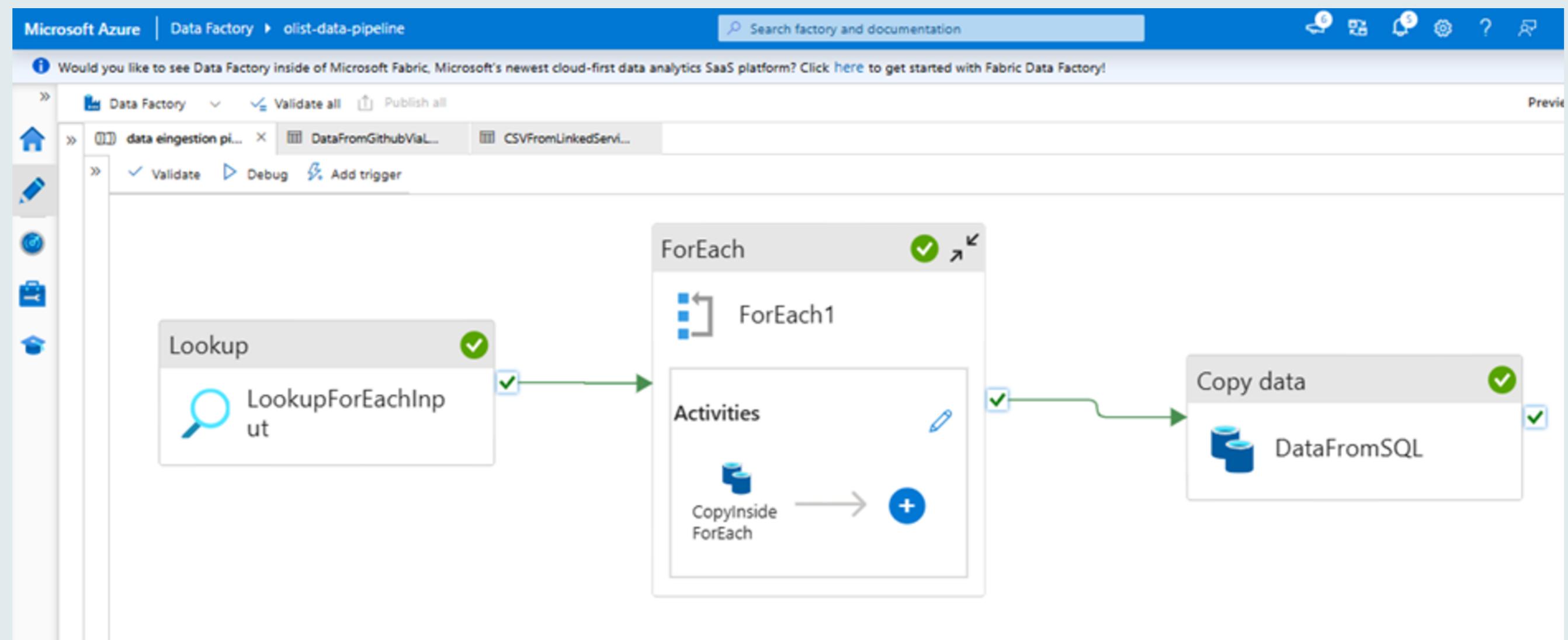
ARCHITECTURE OVERVIEW



DATA INGESTION WITH AZURE DATA FACTORY



DATA INGESTION WITH AZURE DATA FACTORY



RAW DATA STORAGE - ADLS GEN2

The screenshot shows the Azure Storage Accounts interface for the storage account "olistdatastoragemayuresh". The left sidebar navigation includes Home, Storage accounts, olistdatastoragemayuresh, Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, Partner solutions, Resource visualizer, Data storage (Containers selected), and File shares. The main content area displays the "Containers" page with two containers listed: "\$logs" and "olistdatamayur". The table columns are Name, Last modified, Anonymous access level, and Lease state.

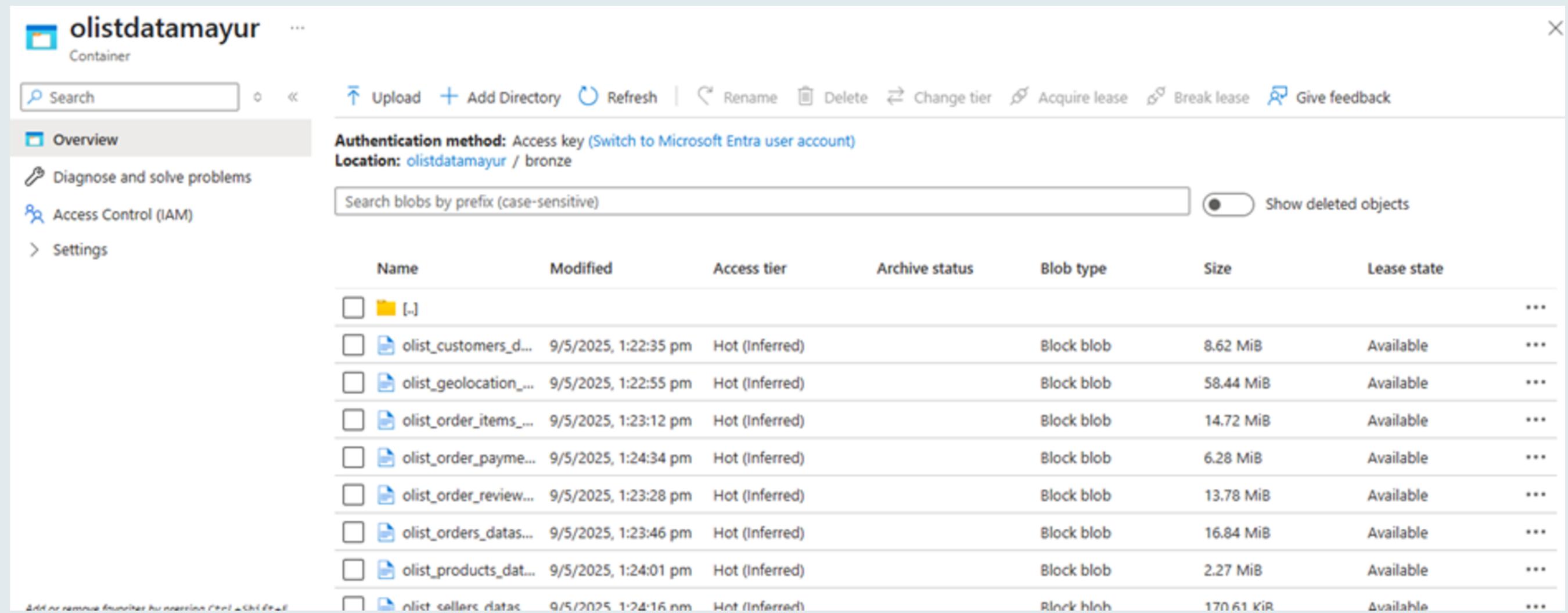
Name	Last modified	Anonymous access level	Lease state
\$logs	9/5/2025, 11:38:15 am	Private	Available
olistdatamayur	9/5/2025, 11:42:15 am	Private	Available

RAW DATA STORAGE - ADLS GEN2

The screenshot shows the Azure Storage Explorer interface for the 'olistdatamayur' container. The left sidebar includes options for Overview, Diagnose and solve problems, Access Control (IAM), and Settings. The main area displays the container's authentication method (Access key) and location (olistdatamayur). A search bar allows filtering by blob prefix. The table lists the following data:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
bronze	9/5/2025, 11:42:57 am				-	...
gold	9/5/2025, 11:43:13 am				-	...
silver	9/5/2025, 11:43:03 am				-	...

RAW DATA STORAGE - ADLS GEN2



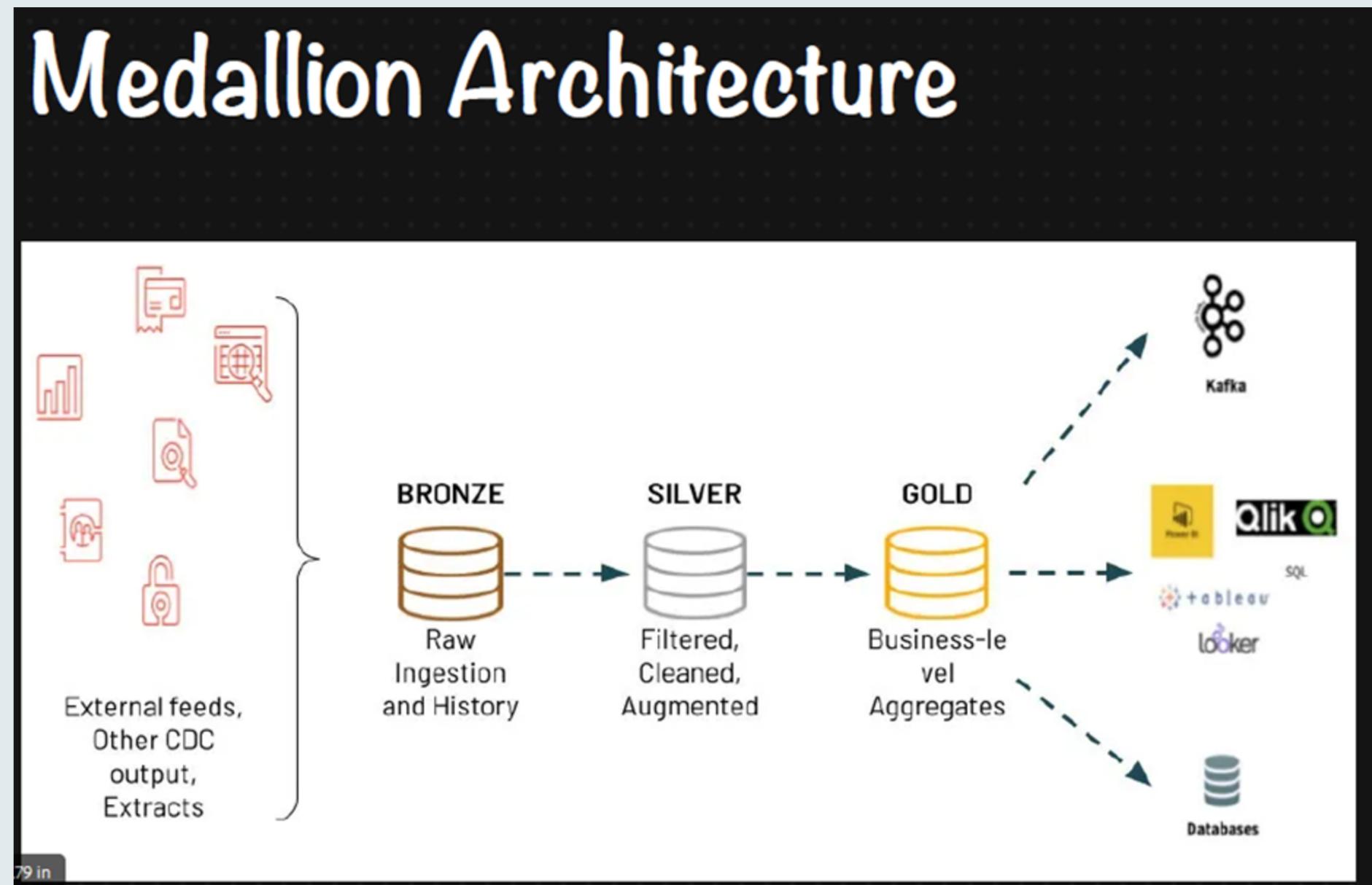
The screenshot shows the Azure Storage Explorer interface for the 'olistdatamayur' container. The container is a 'Container' type with an 'Access key' authentication method and is located in the 'bronze' storage account tier. The interface includes a search bar, navigation buttons, and a toolbar with options like Upload, Add Directory, Refresh, Rename, Delete, Change tier, Acquire lease, Break lease, and Give feedback.

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

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[...]						
olist_customers_d...	9/5/2025, 1:22:35 pm	Hot (Inferred)		Block blob	8.62 MiB	Available
olist_geolocation_...	9/5/2025, 1:22:55 pm	Hot (Inferred)		Block blob	58.44 MiB	Available
olist_order_items_...	9/5/2025, 1:23:12 pm	Hot (Inferred)		Block blob	14.72 MiB	Available
olist_order_payme...	9/5/2025, 1:24:34 pm	Hot (Inferred)		Block blob	6.28 MiB	Available
olist_order_review...	9/5/2025, 1:23:28 pm	Hot (Inferred)		Block blob	13.78 MiB	Available
olist_orders_datas...	9/5/2025, 1:23:46 pm	Hot (Inferred)		Block blob	16.84 MiB	Available
olist_products_dat...	9/5/2025, 1:24:01 pm	Hot (Inferred)		Block blob	2.27 MiB	Available
olist_sales_data...	9/5/2025, 1:24:16 pm	Hot (Inferred)		Block blob	170.61 KiB	Available

RAW DATA STORAGE - ADLS GEN2



APP REGISTRATIONS

The screenshot shows the 'App registrations' page in the Azure portal. The top navigation bar includes links for 'Home', 'App registrations', 'Endpoints', 'Troubleshoot', 'Refresh', 'Download', 'Preview features', and 'Got feedback?'. A message banner at the top states: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph.' Below the banner, there are tabs for 'All applications' (which is selected), 'Owned applications', 'Deleted applications', and 'Applications from personal account'. A search bar allows filtering by 'Display name or application (client) ID'. The main table displays one application entry:

Display name	Application (client) ID	Created on	Certificates & secrets
olist-app-registration-from-db-to-adls	6f9881f4-7379-4656-8ffc-07e0ed559c79	9/5/2025	<input checked="" type="checkbox"/> Current

APP REGISTRATION

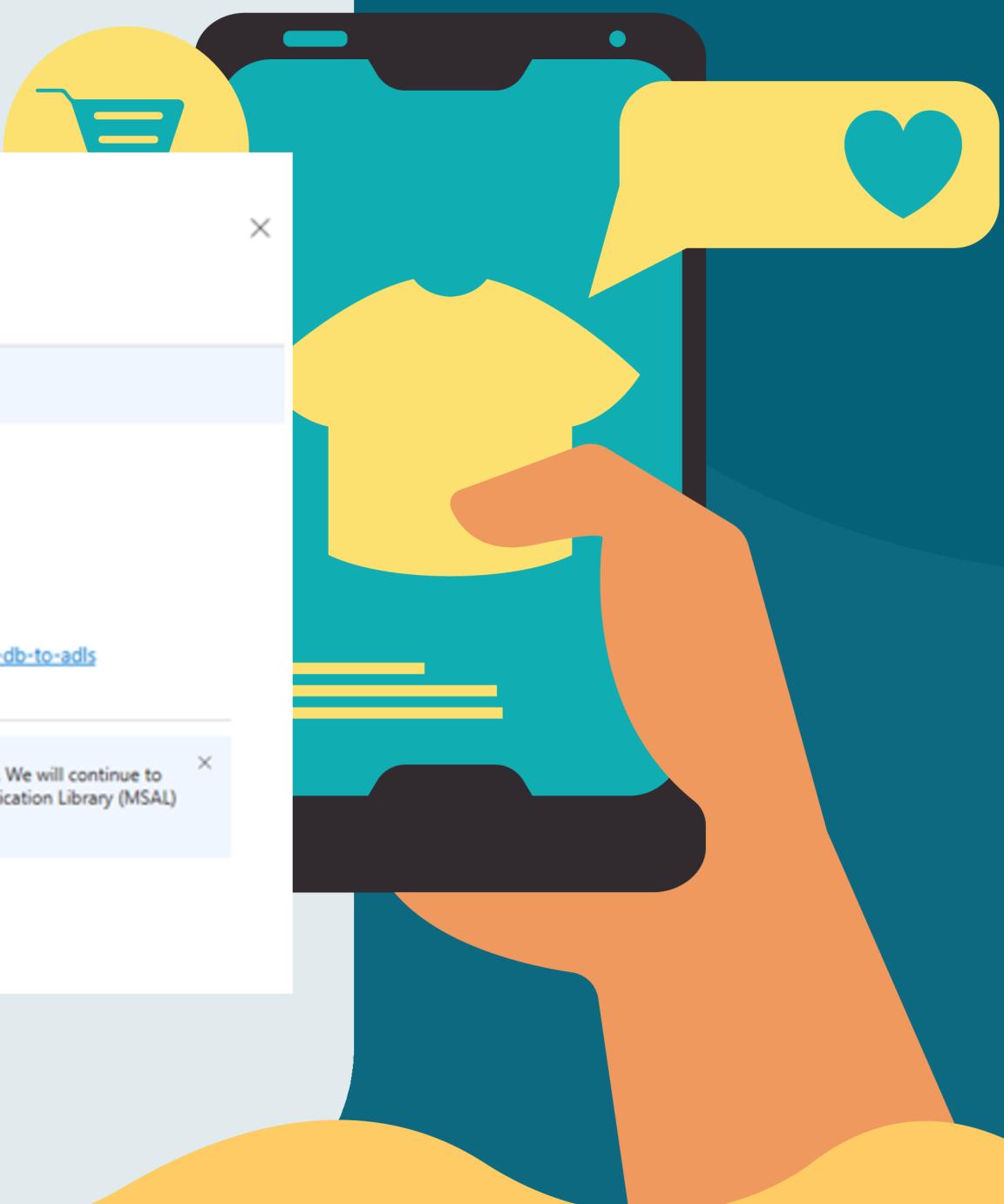
The screenshot shows the Azure App Registrations overview page for an application named "olist-app-registration-from-db-to-adls". The left sidebar includes links for Overview, Quickstart, Integration assistant, Diagnose and solve problems, Manage (Branding & properties, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners), Get Started, and Documentation.

Essentials

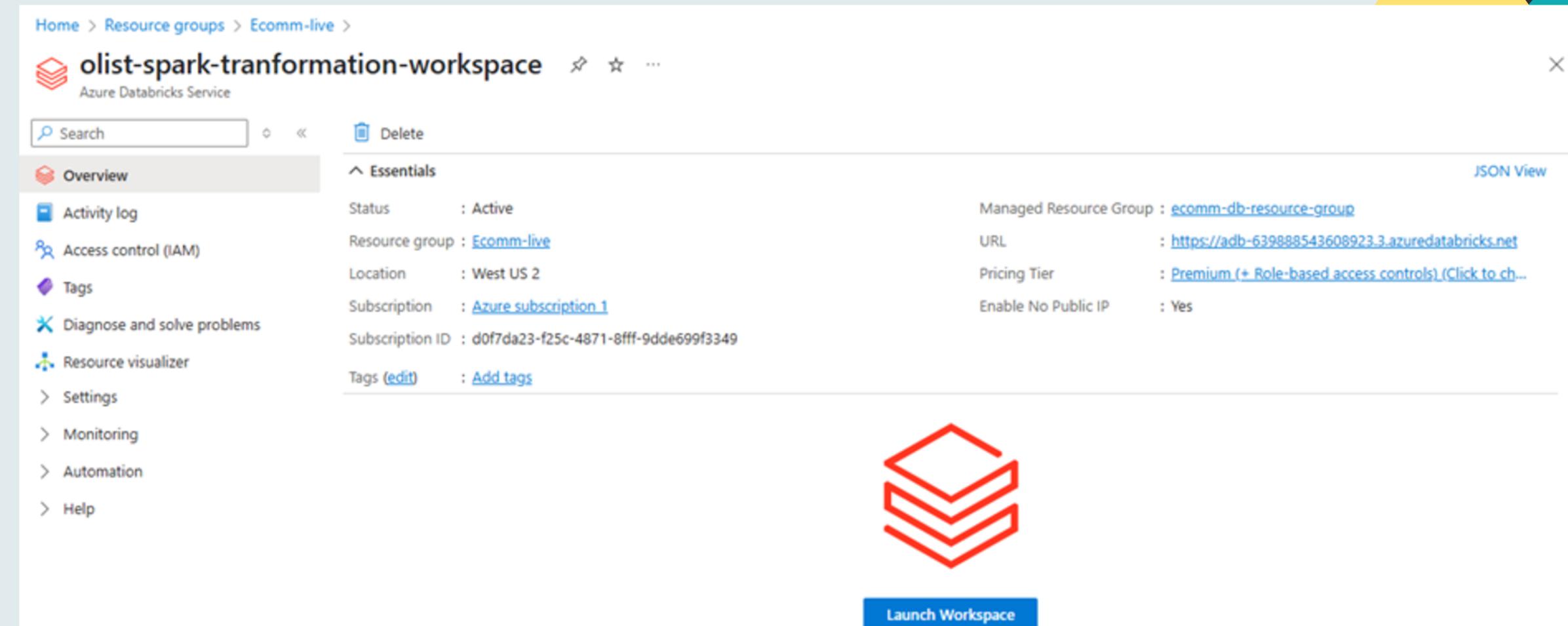
Display name	: olist-app-registration-from-db-to-adls	Client credentials	: 0 certificate, 1 secret
Application (client) ID	: 6f9881f4-7379-4656-8ffcc-07e0ed559c79	Redirect URIs	: Add a Redirect URI
Object ID	: 132748a7-3579-4a76-b73f-d7bb07d614bd	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	: 392a457d-9318-460e-a390-0f2d953aa39c	Managed application in ...	: olist-app-registration-from-db-to-adls

Supported account types : [My organization only](#)

Important Message: Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)



WHY AZURE DATABRICKS?



WHY AZURE DATABRICKS?

Mayuresh's Cluster

Policy: Unrestricted

Access mode: Single user or group access (Dedicated (formerly: Single user))

Rushabh Runwal

Performance

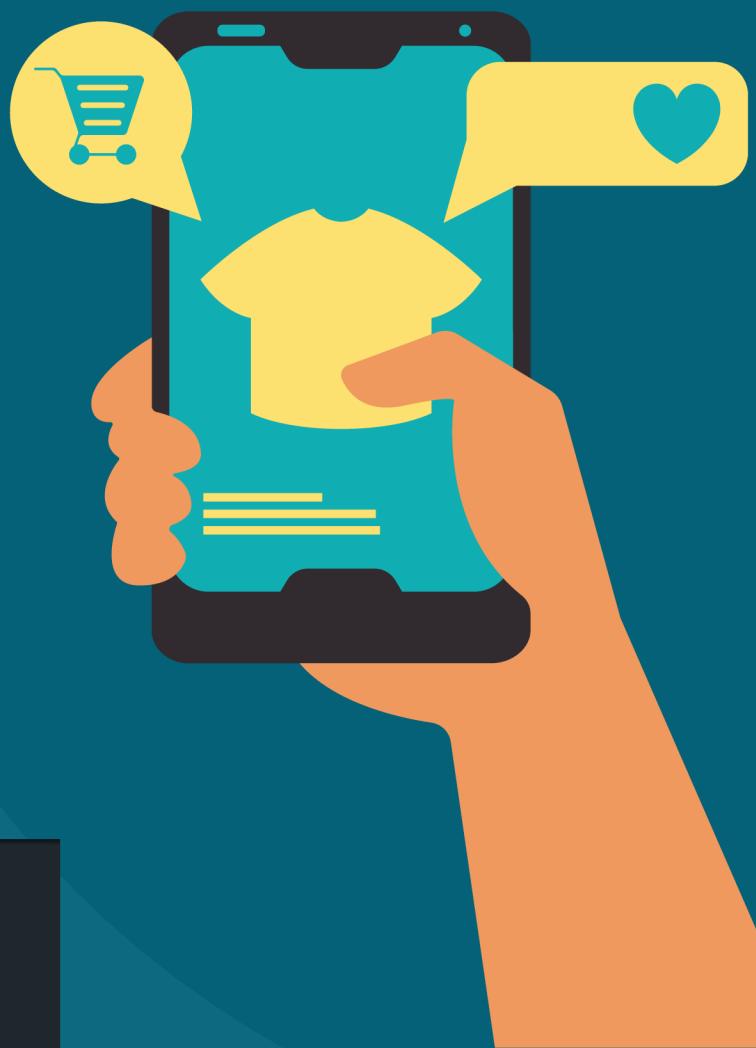
Databricks runtime version: Runtime: 14.3 LTS (Scala 2.12, Spark 3.5.0)

Use Photon Acceleration: checked

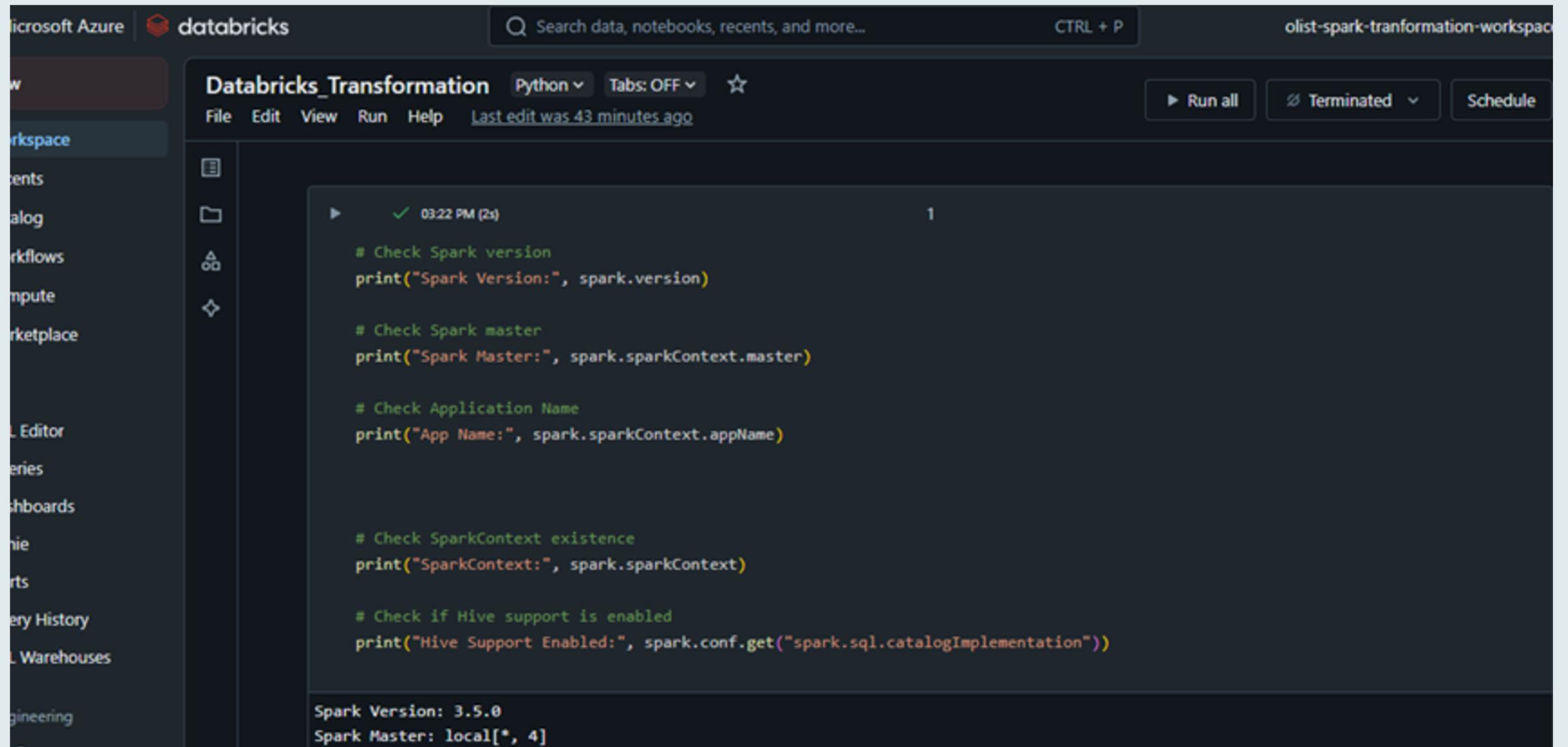
Node type: Standard_D4ds_v5 (16 GB Memory, 4 Cores)

Summary

1 Driver	16 GB Memory, 4 Cores		
Runtime	14.3.x-scala2.12		
Unity Catalog	Photon	Standard_D4ds_v5	2 DBU/h



WHY AZURE DATABRICKS?

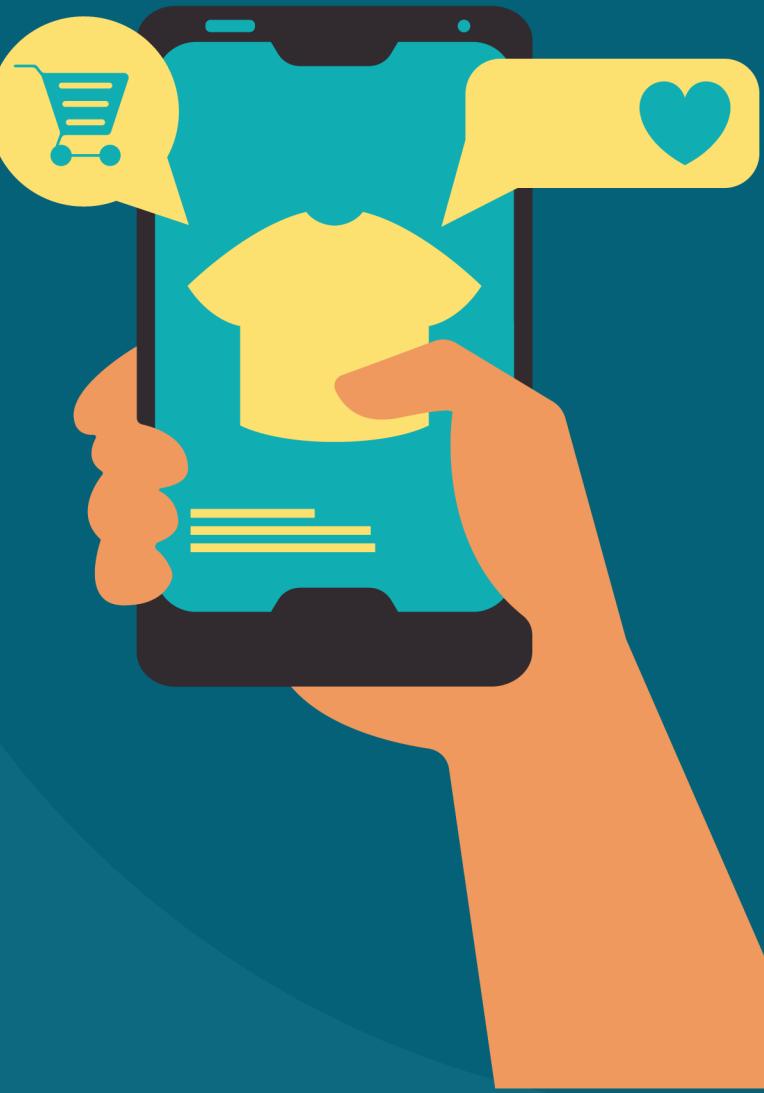


The screenshot shows the Azure Databricks workspace interface. On the left, there's a sidebar with various navigation options like Workspaces, Datasets, Catalog, Workflows, Compute, Marketplace, Editor, Databases, Dashboards, Pipeline, Scripts, Activity History, Warehouses, and Engineering. The main area is a notebook titled "Databricks_Transformation" in Python. The code in the notebook is as follows:

```
# Check Spark version  
print("Spark Version:", spark.version)  
  
# Check Spark master  
print("Spark Master:", spark.sparkContext.master)  
  
# Check Application Name  
print("App Name:", spark.sparkContext.appName)  
  
# Check SparkContext existence  
print("SparkContext:", spark.sparkContext)  
  
# Check if Hive support is enabled  
print("Hive Support Enabled:", spark.conf.get("spark.sql.catalogImplementation"))
```

The output of the notebook shows the results of the printed statements:

```
Spark Version: 3.5.0  
Spark Master: local[* , 4]
```



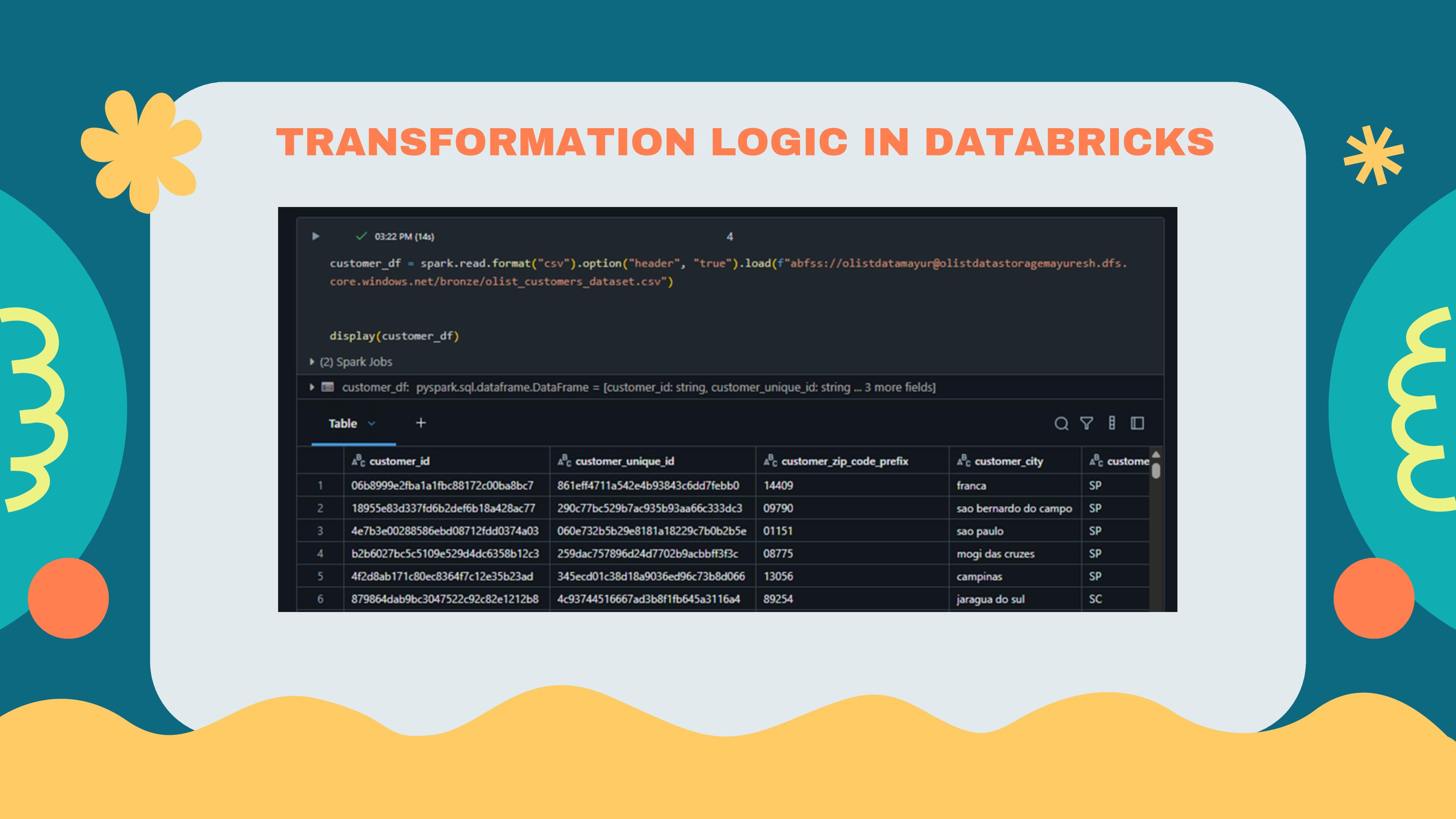
TRANSFORMATION LOGIC IN DATABRICKS

```
▶  ✓ 03:22 PM (<1s) 2

storage_account = "olistdatastoragemayuresh"
application_id = "6f9881f4-7379-4656-8ffc-07e0ed559c79"
directory_id = "392a457d-9318-460e-a390-0f2d953aa39c"

spark.conf.set(f"fs.azure.account.auth.type.{storage_account}.dfs.core.windows.net", "OAuth")
spark.conf.set(f"fs.azure.account.oauth.provider.type.{storage_account}.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.
oauth2.ClientCredsTokenProvider")
spark.conf.set(f"fs.azure.account.oauth2.client.id.{storage_account}.dfs.core.windows.net", application_id)
spark.conf.set(f"fs.azure.account.oauth2.client.secret.{storage_account}.dfs.core.windows.net",
"lWC8Qw0K6097ckYNzLZwLfVBwIEQv0KRk02Iaci")
spark.conf.set(f"fs.azure.account.oauth2.client.endpoint.{storage_account}.dfs.core.windows.net", f"https://login.
microsoftonline.com/{directory_id}/oauth2/token")
```

TRANSFORMATION LOGIC IN DATABRICKS



```
▶ ✓ 03:22 PM (14s) 4
customer_df = spark.read.format("csv").option("header", "true").load("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 + Q Y E □
+---+-----+-----+-----+-----+-----+
| 1 | 06b8999e2fba1a1fb88172c00ba8bc7 | 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

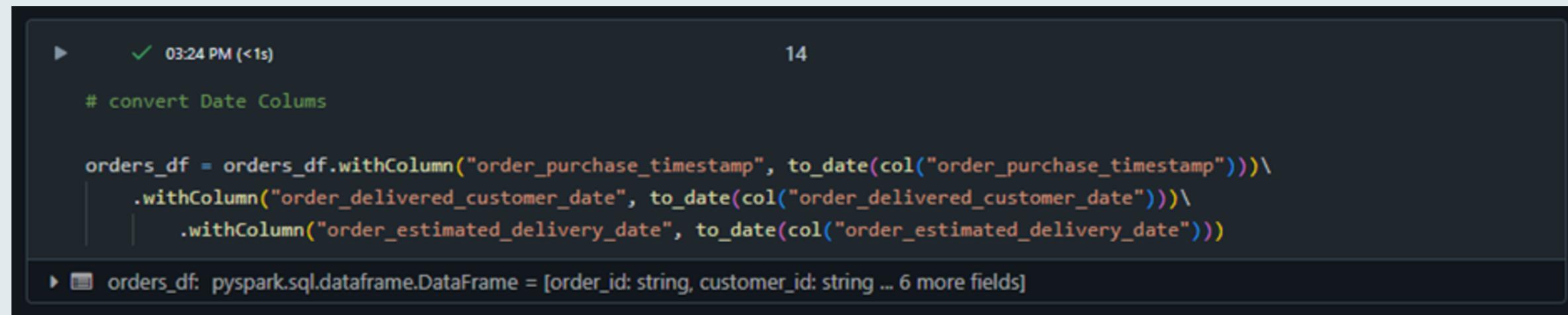
▶   ✓ 03:24 PM (2s) 13
def clean_dataframe(df,name):
    print("Cleaning "+name)
    return df.dropDuplicates().na.drop('all')

orders_df = clean_dataframe(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 + Q Y E □
+-----+-----+-----+-----+-----+
| AB order_id | AB customer_id | AB order_status | AB order_purchase_timestamp | AB order_ap |
+-----+-----+-----+-----+-----+
1 | 3923e3ade70348985bd2ca389905cf19 | 6454e6cba392b35aa21527063026fc92 | delivered | 2018-03-07 23:00:33 | 2018-03-09 0
```

TRANSFORMATION LOGIC IN DATABRICKS



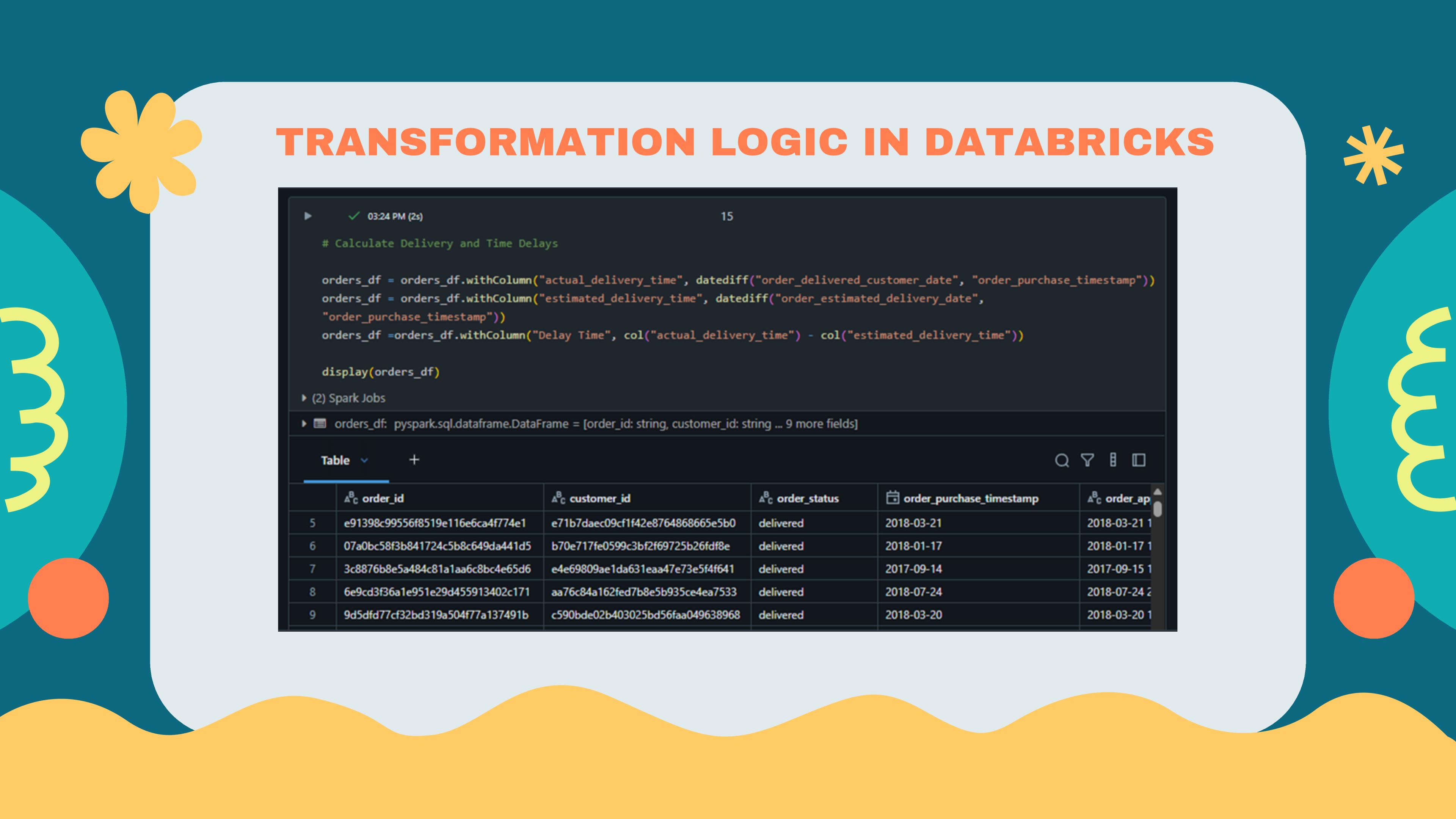
The image shows a dark-themed Jupyter Notebook cell. At the top left is a play button icon followed by a green checkmark and the text "03:24 PM (<1s)". On the right is the number "14". The code in the cell is as follows:

```
# convert Date Columns

orders_df = orders_df.withColumn("order_purchase_timestamp", to_date(col("order_purchase_timestamp")))\n    .withColumn("order_delivered_customer_date", to_date(col("order_delivered_customer_date")))\n    .withColumn("order_estimated_delivery_date", to_date(col("order_estimated_delivery_date")))
```

Below the code, there is a small icon of a table followed by the text "orders_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 6 more fields]".

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 + Q Y E □

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

TRANSFORMATION LOGIC IN DATABRICKS

```
▶   ✓ 03:32 PM (<1s) 17
orders_cutomers_df = orders_df.join(customers_df, orders_df.customer_id == customers_df.customer_id,"left")

orders_payments_df = orders_cutomers_df.join(payments_df, orders_cutomers_df.order_id == payments_df.order_id,"left")

orders_items_df = orders_payments_df.join(items_df,"order_id","left")

orders_items_products_df = orders_items_df.join(products_df, orders_items_df.product_id == products_df.product_id,"left")

final_df = orders_items_products_df.join(sellers_df, orders_items_products_df.seller_id == sellers_df.seller_id,"left")
▶ final_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 38 more fields]
▶ orders_cutomers_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 14 more fields]
▶ orders_items_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 25 more fields]
▶ orders_items_products_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 34 more fields]
▶ orders_payments_df: pyspark.sql.dataframe.DataFrame = [order_id: string, customer_id: string ... 19 more fields]
```

TRANSFORMATION LOGIC IN DATABRICKS

03:32 PM (5s)

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	3be5a877ceeb ea2954404c6ef8e70be2	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	e4e69809ae1da631eaaa47e73e5f4f641	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

```
▶  ✓ 03:23 PM (5s)          6
!pip install pymongo

Note: you may need to restart the kernel using dbutils.library.restartPython() to use updated packages.
Collecting pymongo
  Downloading pymongo-4.12.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
  ━━━━━━━━━━━━━━━━ 1.2/1.2 MB 13.4 MB/s eta 0:00:00
Collecting dnsPYTHON<3.0.0,>=1.16.0
  Downloading dnsPYTHON-2.7.0-py3-none-any.whl (313 kB)
  ━━━━━━━━━━━━━━ 313.6/313.6 kB 14.5 MB/s eta 0:00:00
Installing collected packages: dnsPYTHON, pymongo
Successfully installed dnsPYTHON-2.7.0 pymongo-4.12.1
Note: you may need to restart the kernel using dbutils.library.restartPython() to use updated packages.

💡
```

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

The screenshot shows a Jupyter Notebook interface with two code cells and a resulting DataFrame.

Code Cell 1:

```
▶ ✓ 03:23 PM (1s) 9
import pandas as pd
collection = mydatabase['product_categories']

mongo_data = pd.DataFrame(list(collection.find()))
```

Code Cell 2:

```
▶ ✓ 03:24 PM (1s) 10
display(products_df)
▶ (1) Spark Jobs
```

Resulting DataFrame:

Table	+ product_id	product_category_name	product_name_length	product_description_length	product...

DATA ENRICHMENT USING NOSQL

```
▶ ✓ 03:23 PM (1s) 9
import pandas as pd
collection = mydatabase['product_categories']

mongo_data = pd.DataFrame(list(collection.find()))

▶ ✓ 03:24 PM (1s) 10
display(products_df)
▶ (1) Spark Jobs
Table + Q ▾ □
ABC product_id ABC product_category_name ABC product_name_length ABC product_description_length ABC pro
```

DATA ENRICHMENT USING NOSQL

The screenshot shows a Jupyter Notebook cell with the title "mongo_data". The cell displays a pandas DataFrame with 71 rows and 3 columns. The columns are labeled "_id", "product_category_name", and "product_category_name_english". The data represents product categories mapped to their English equivalents. The first few rows are:

	_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 + Q Y E □
```

	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]

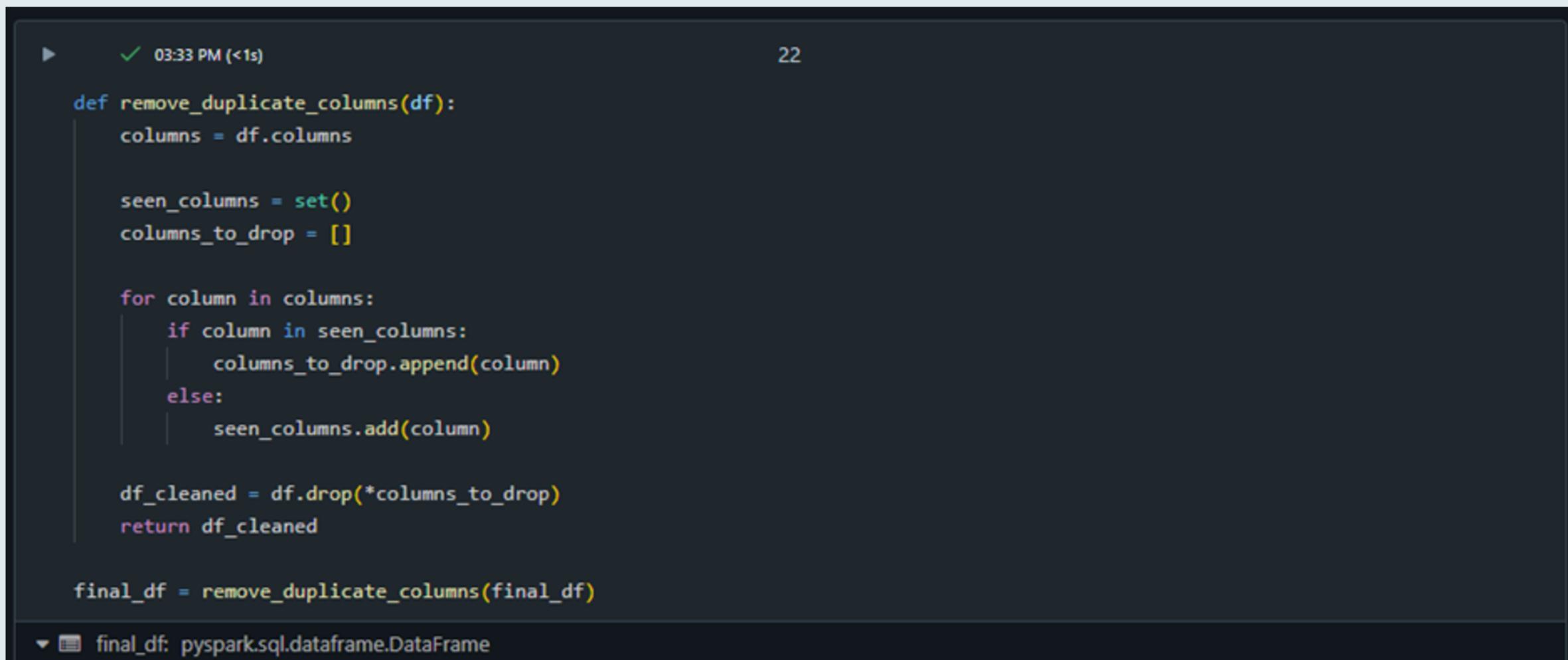
▶ ✓ 03:32 PM (4s) 21
display(final_df)

▶ (8) Spark Jobs

Table + Q F E □

+-----+-----+-----+-----+-----+
| ABC product_category_name | ABC order_id | ABC customer_id | ABC order_status | ABC 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 | 3be5a877ceebca2954404c6ef8e70be2 | 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



```
▶  ✓ 03:33 PM (<1s) 22

def remove_duplicate_columns(df):
    columns = df.columns

    seen_columns = set()
    columns_to_drop = []

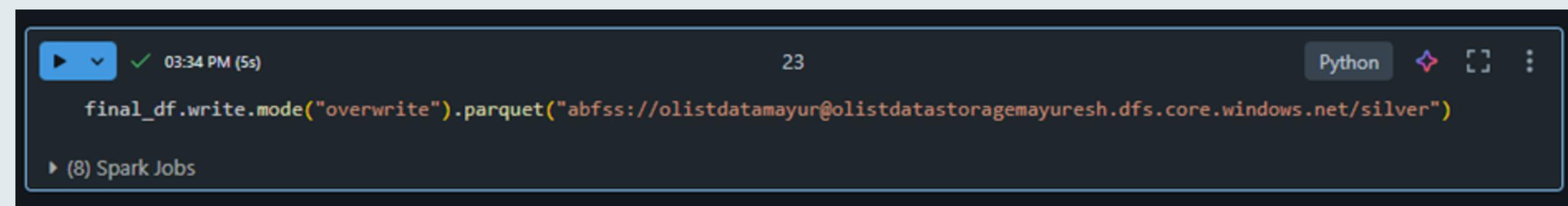
    for column in columns:
        if column in seen_columns:
            columns_to_drop.append(column)
        else:
            seen_columns.add(column)

    df_cleaned = df.drop(*columns_to_drop)
    return df_cleaned

final_df = remove_duplicate_columns(final_df)

▼ final_df: pyspark.sql.dataframe.DataFrame
```

TRANSFORMED DATA STORAGE

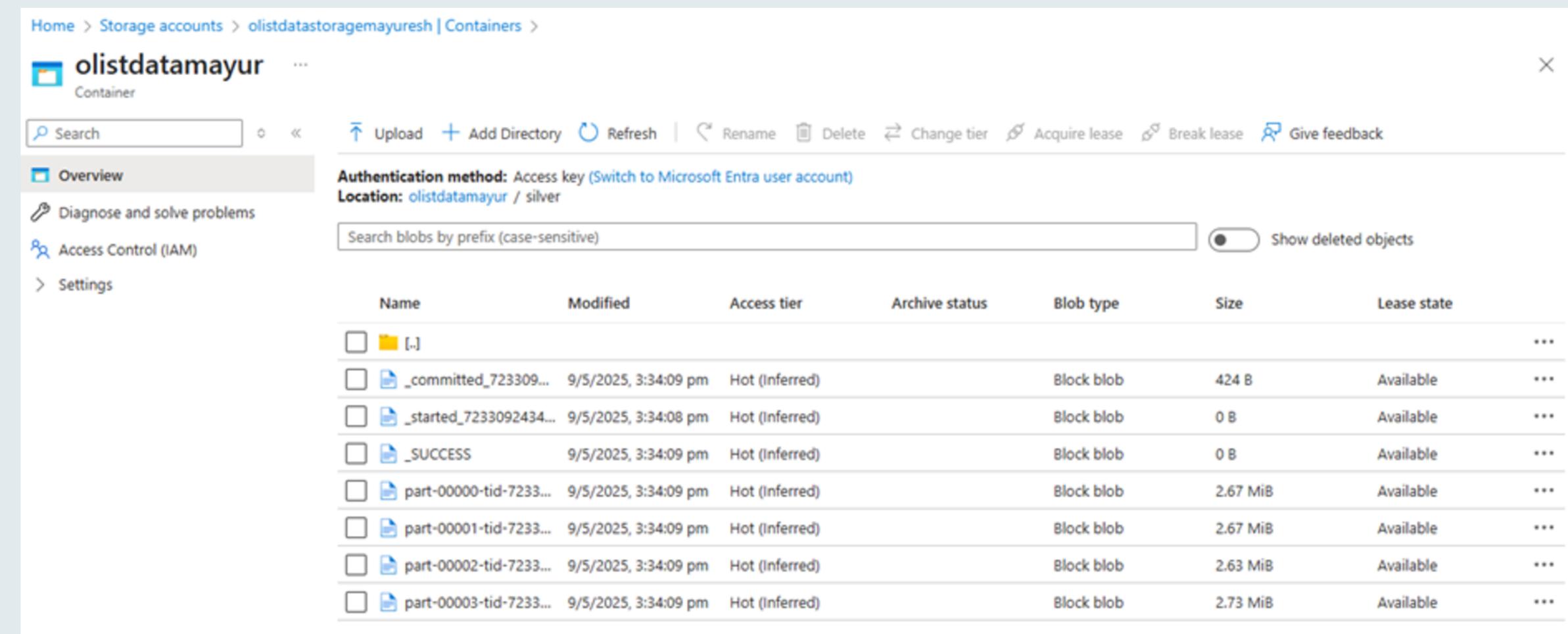


A screenshot of a Jupyter Notebook cell. The cell contains the following Python code:

```
final_df.write.mode("overwrite").parquet("abfss://olistdatamayur@olistdatastoragemayuresh.dfs.core.windows.net/silver")
```

The cell has a status bar at the top indicating "03:34 PM (5s)" and "23". On the right side, there are buttons for "Python", "Run", "Cell", and "More". Below the code, a message "(8) Spark Jobs" is visible.

TRANSFORMED DATA STORAGE



The screenshot shows the Azure Storage Explorer interface for a container named "olistdatamayur". The container is a "Container" type with the following details:

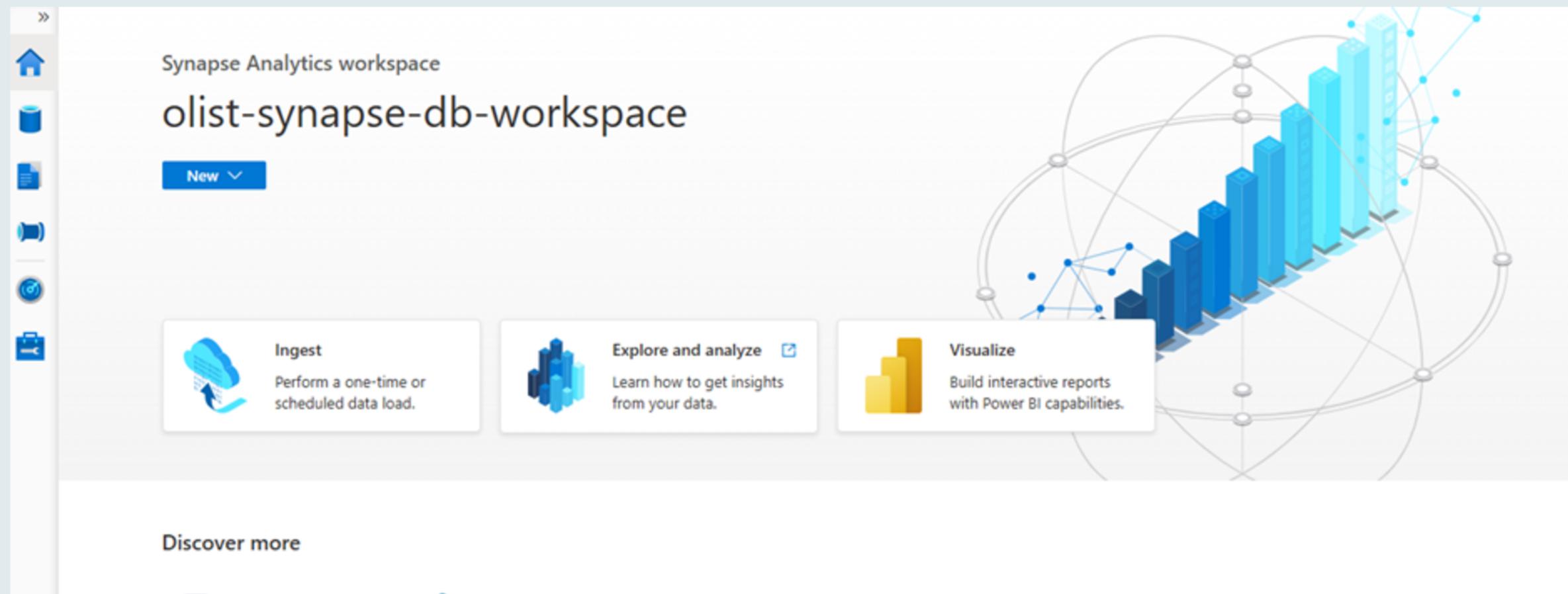
- Authentication method:** Access key (Switch to Microsoft Entra user account)
- Location:** olistdatamayur / silver

A search bar at the top allows searching for blobs by prefix (case-sensitive). A toggle switch is present to "Show deleted objects".

The main table lists the following blob objects:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]						...
_committed_723309...	9/5/2025, 3:34:09 pm	Hot (Inferred)		Block blob	424 B	Available
_started_7233092434...	9/5/2025, 3:34:08 pm	Hot (Inferred)		Block blob	0 B	Available
_SUCCESS	9/5/2025, 3:34:09 pm	Hot (Inferred)		Block blob	0 B	Available
part-00000-tid-7233...	9/5/2025, 3:34:09 pm	Hot (Inferred)		Block blob	2.67 MiB	Available
part-00001-tid-7233...	9/5/2025, 3:34:09 pm	Hot (Inferred)		Block blob	2.67 MiB	Available
part-00002-tid-7233...	9/5/2025, 3:34:09 pm	Hot (Inferred)		Block blob	2.63 MiB	Available
part-00003-tid-7233...	9/5/2025, 3:34:09 pm	Hot (Inferred)		Block blob	2.73 MiB	Available

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'
);

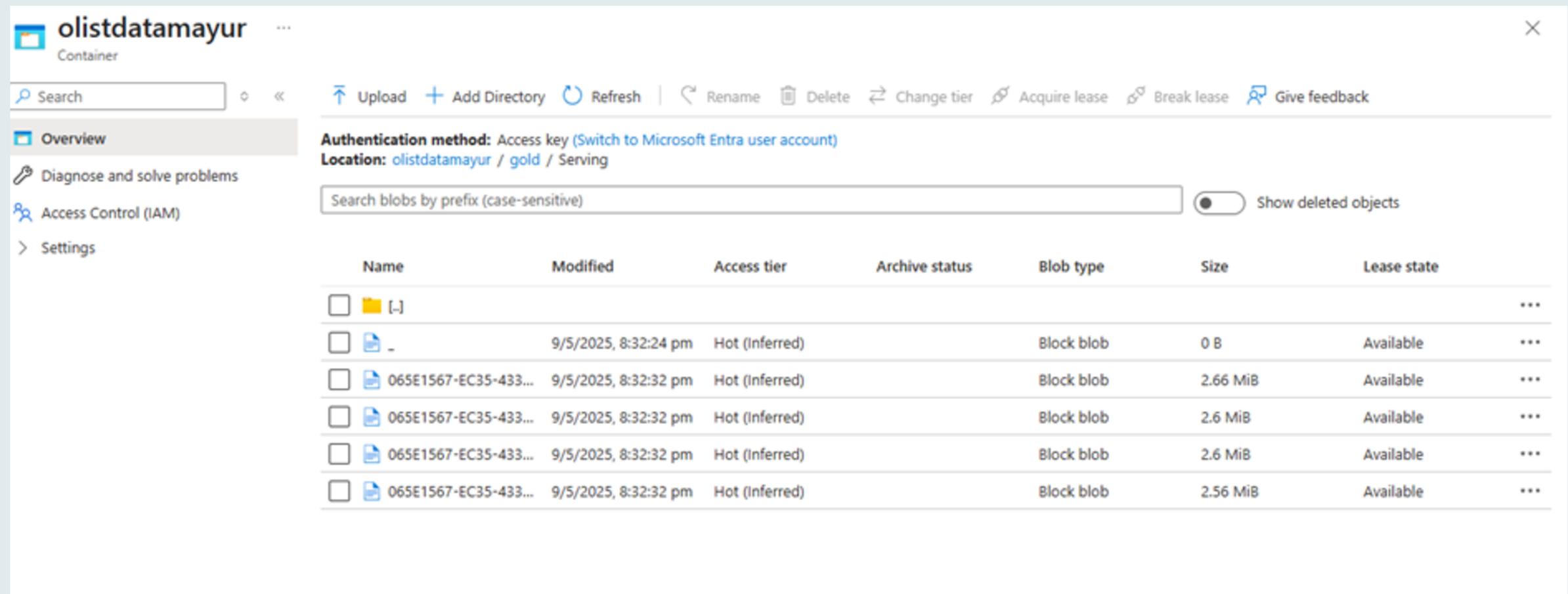
CREATE EXTERNAL DATA SOURCE goldlayer WITH (
    LOCATION = 'https://olistdatastorageaccount.dfs.core.windows.net/olistdata/gold/',
    CREDENTIAL = mayankadmin
);

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

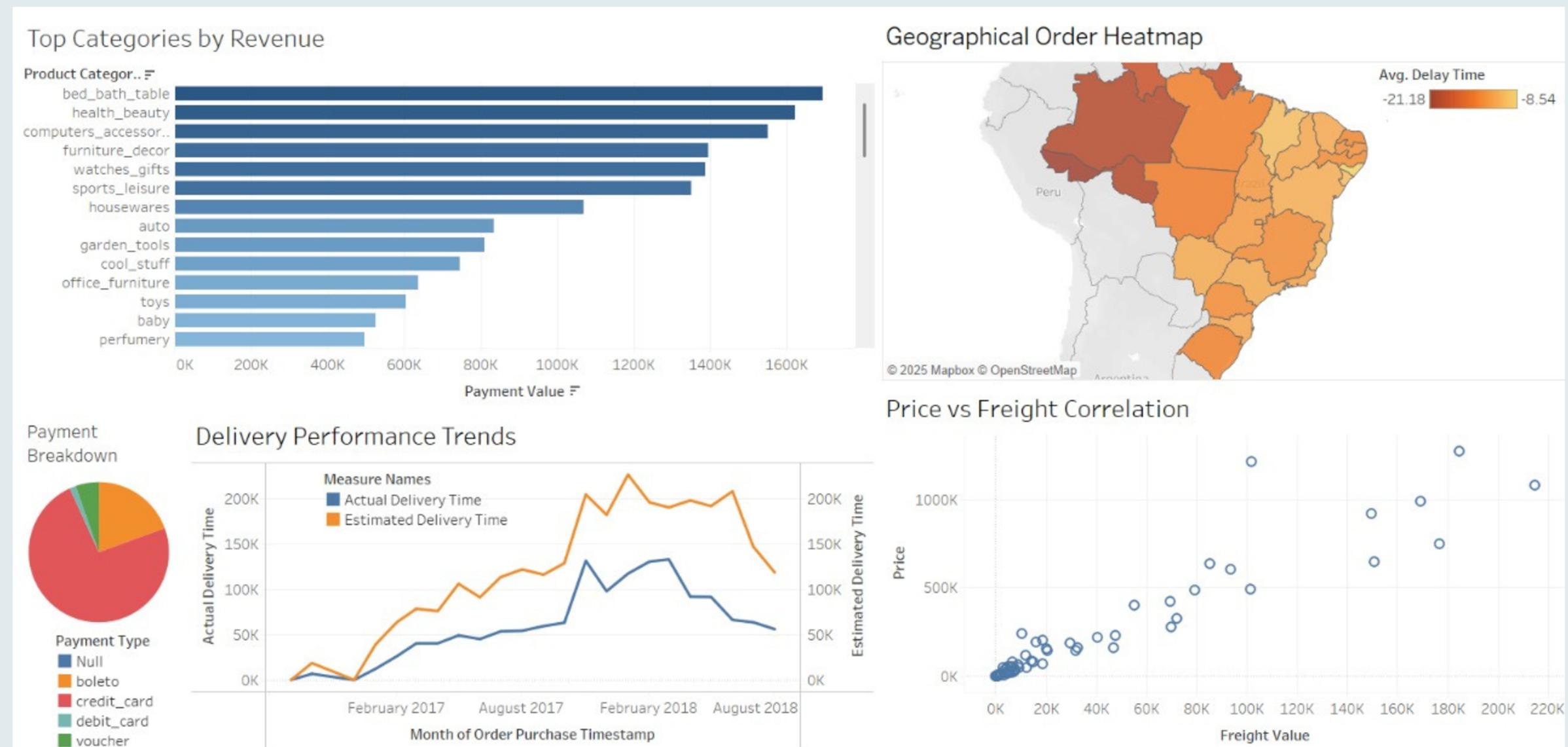
select * from gold.finaltable

select * from gold.final2
```

SYNAPSE ANALYTICS INTEGRATION



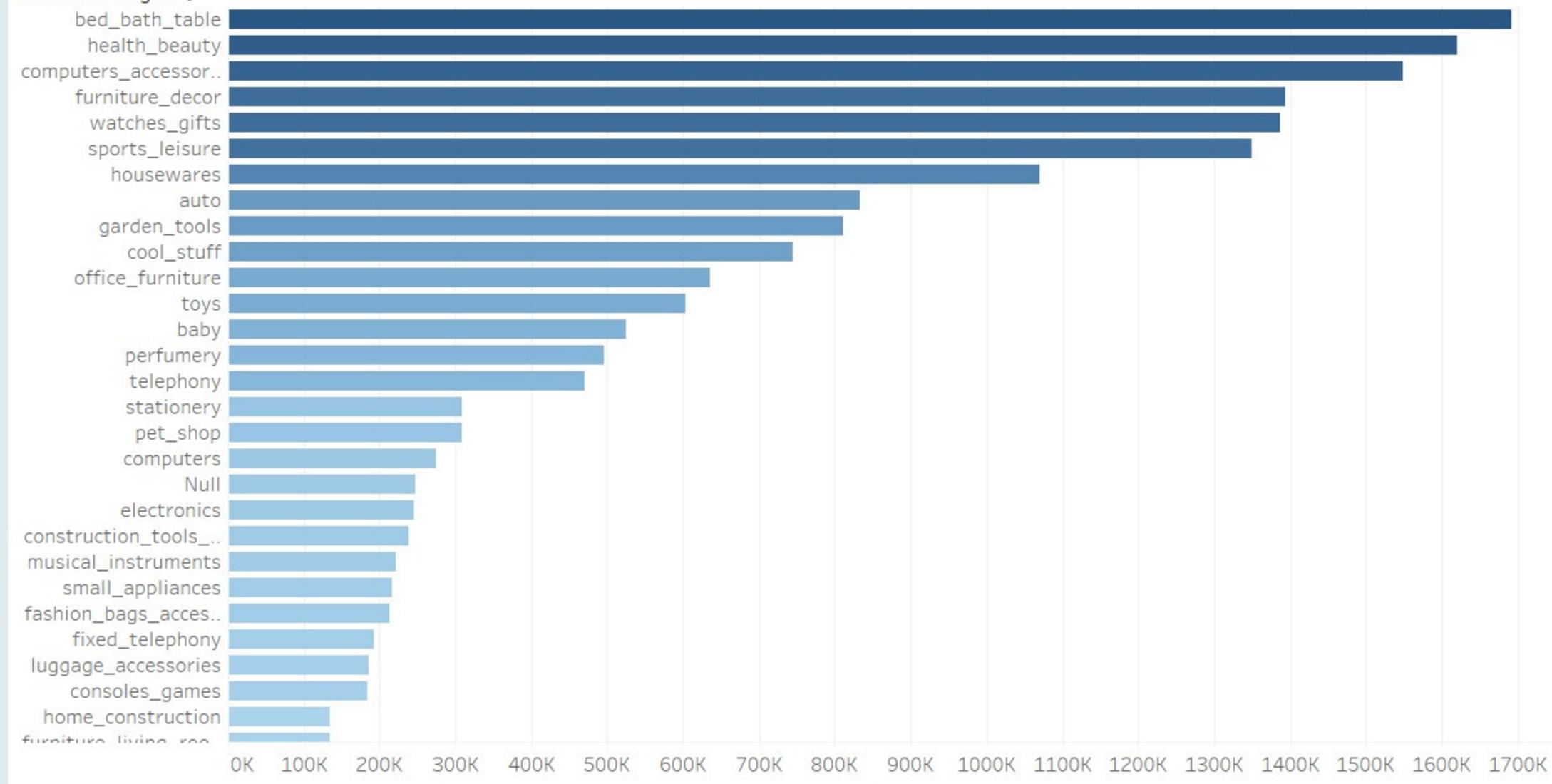
VISUALIZATION LAYER



DASHBOARD INSIGHTS

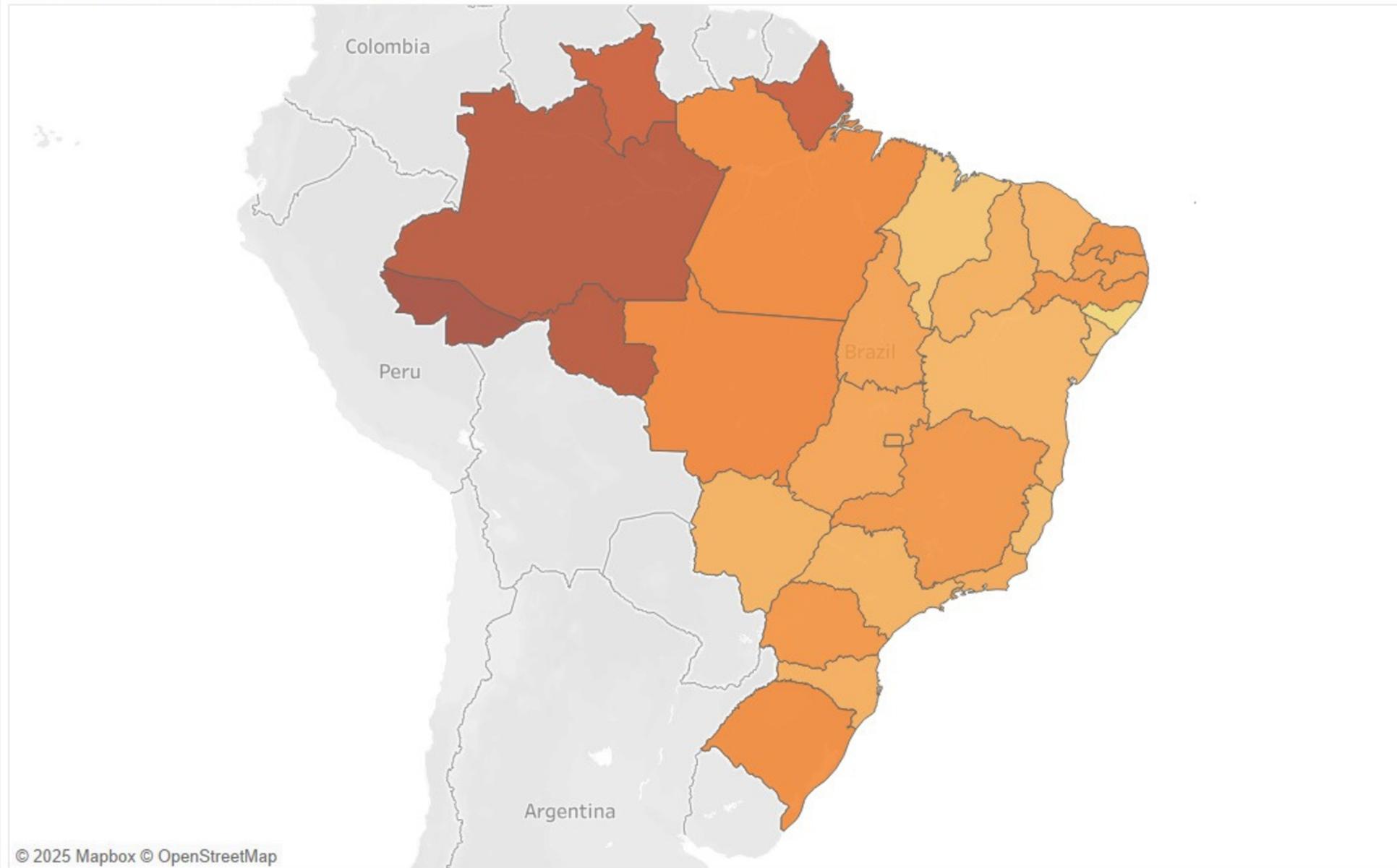
Top Categories by Revenue

Product Categor...

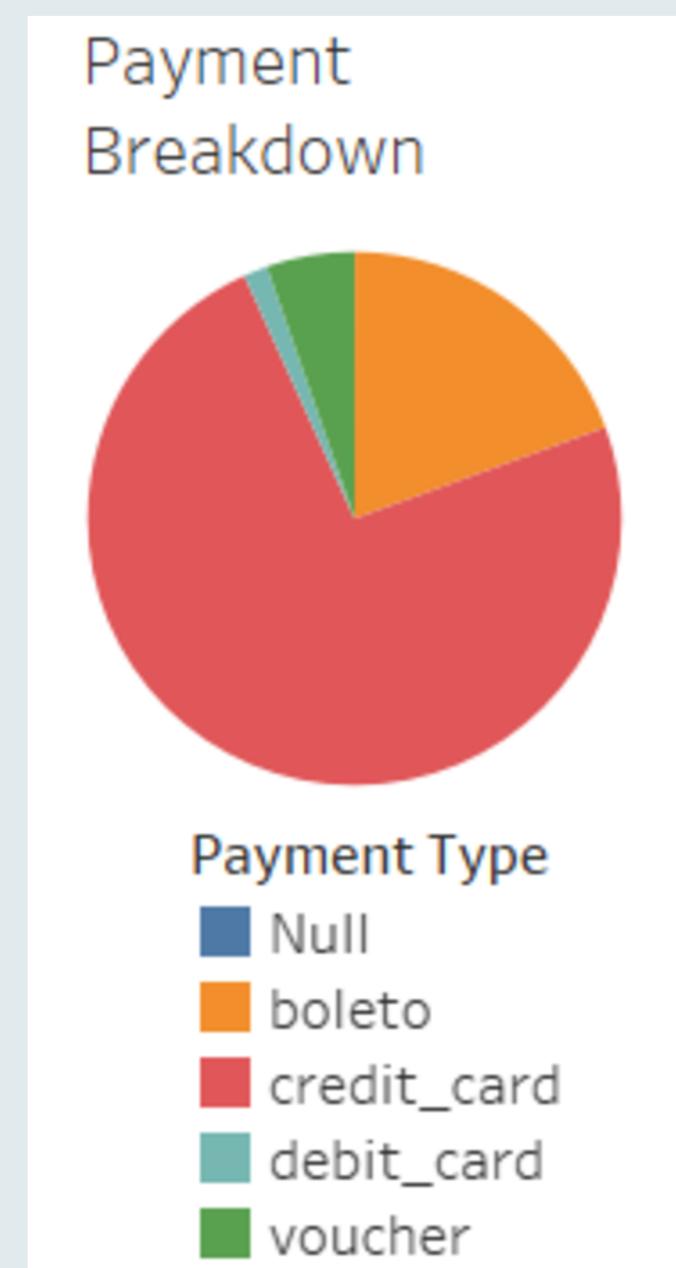


DASHBOARD INSIGHTS

Geographical Order Heatmap



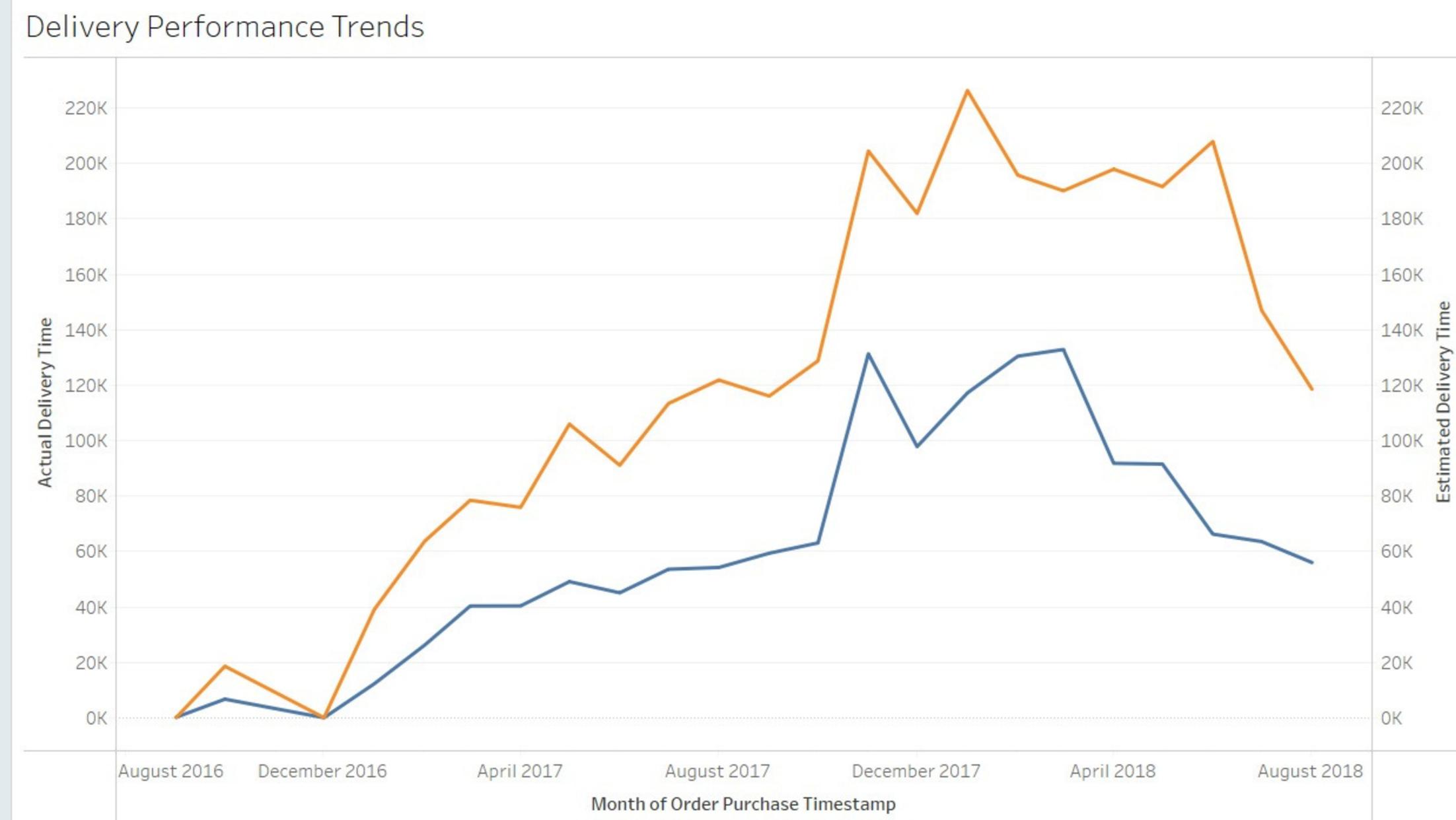
DASHBOARD INSIGHTS



DASHBOARD INSIGHTS

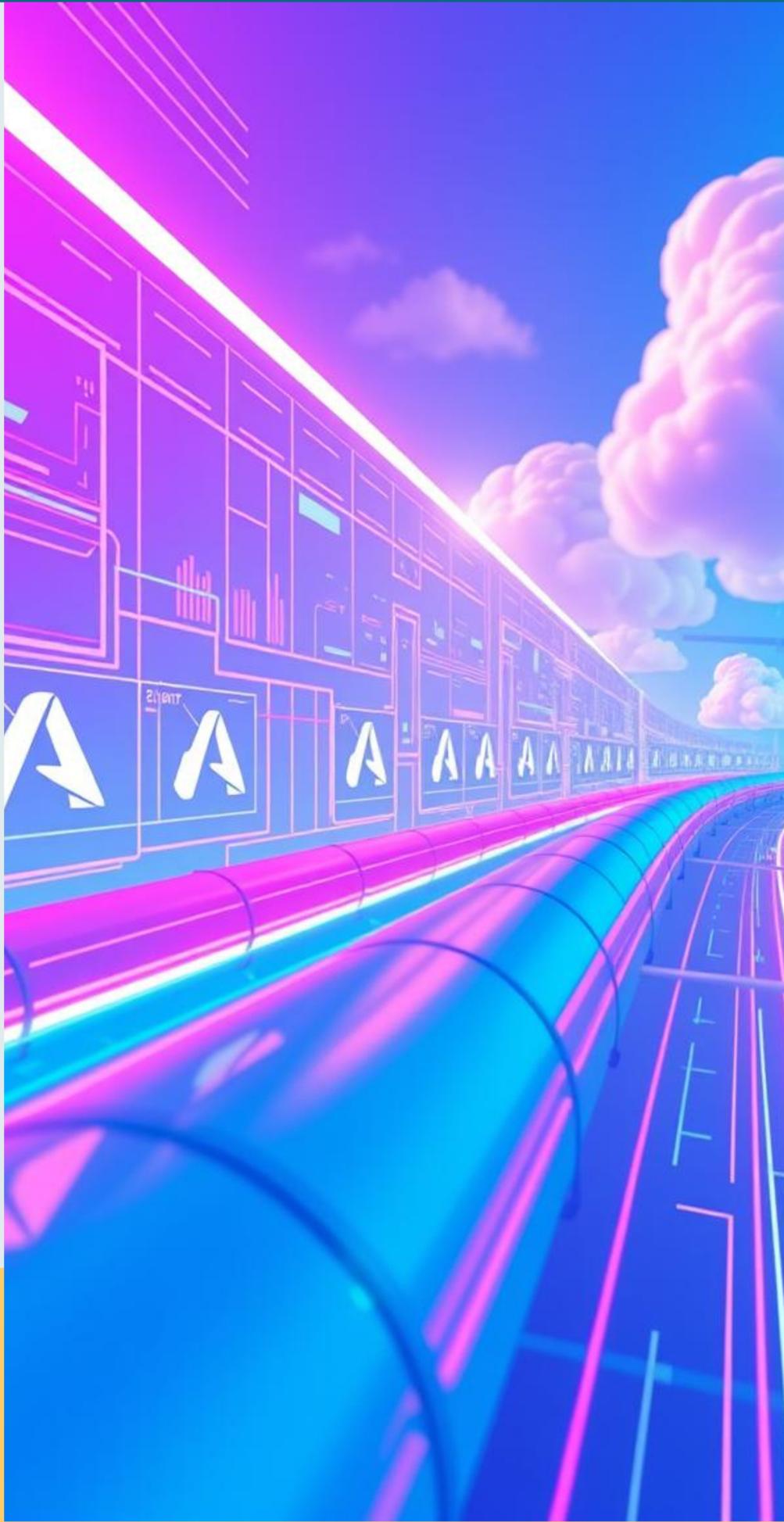


DASHBOARD INSIGHTS



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?

