EXECUTABLE INFO DOCUMENT

DATA WAREHOUSING INTEGRATION

1. INTRODUCTION

This document provides execution evidence and project results for the **Data Warehousing Integration** solution built using Azure Data Factory (ADF), Azure Databricks, Azure Data Lake Storage (ADLS), and Azure Synapse Analytics.

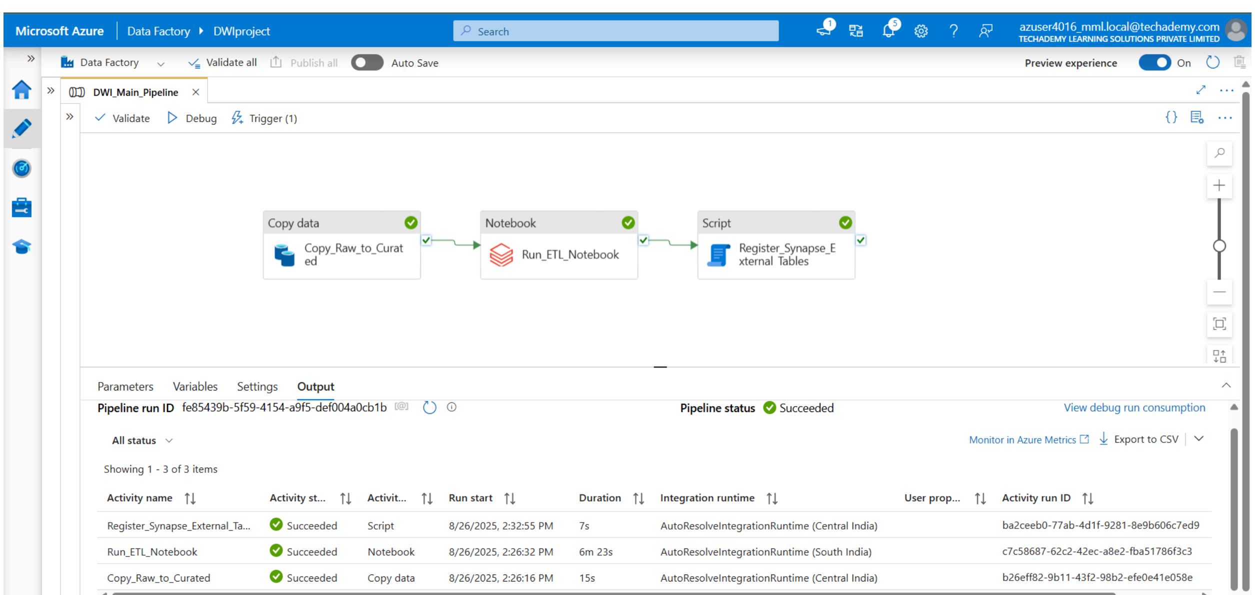
The purpose of this document is to:

- Demonstrate the successful end-to-end execution of the ETL pipeline.
- Showcase validation outputs and query results proving schema consistency and data correctness.
- Present analytical insights derived from curated warehouse tables.

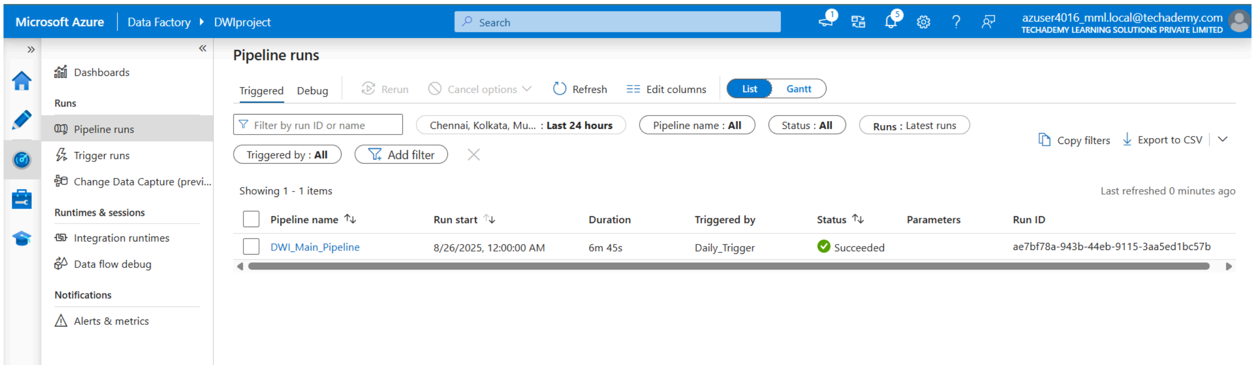
2. PIPELINE EXECUTION PROOF

2.1 AZURE DATA FACTORY (ADF)

- **Pipeline Design View**



- **Pipeline Monitoring**



2.2 AZURE DATABRICKS

- **Databricks Notebook Transformation Output**

```
Yesterday (<1s)                                                    8

# Clean column names (replace spaces with underscores)
df = df.toDF(*[c.replace(" ", "_") for c in df.columns])

df.printSchema()

df: pyspark.sql.dataframe.DataFrame = [Transaction_ID: string, Date: string ... 7 more fields]

root
 |-- Transaction_ID: string (nullable = true)
 |-- Date: string (nullable = true)
 |-- Customer_ID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: string (nullable = true)
 |-- Product_Category: string (nullable = true)
 |-- Quantity: string (nullable = true)
 |-- Price_per_Unit: string (nullable = true)
 |-- Total_Amount: string (nullable = true)
```

```
Yesterday (3s)                                                     10

print("Total rows:", df.count())

(2) Spark Jobs

Total rows: 1000
```

3. Transform – Create Dimension Tables

3.1 Customer Dimension

Yesterday (<1s) 13

```python
# Dim Customer
dim_customer = df.select(
    "Customer_ID", "Gender", "Age"
).dropDuplicates(["Customer_ID"])
```

dim_customer: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Gender: string ... 1 more field]

3.2 Product Dimension

Yesterday (<1s) 15

```python
# Dim Product
from pyspark.sql.functions import monotonically_increasing_id

# Proper Dim_Product with surrogate key
dim_product = df.select("Product_Category").dropDuplicates() \
                .withColumn("Product_ID", monotonically_increasing_id())
```

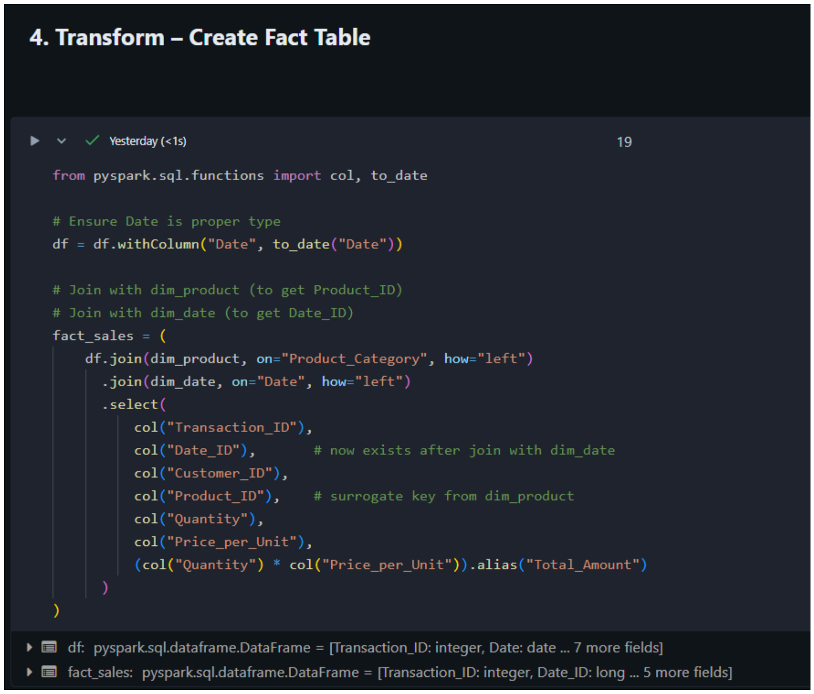
dim_product: pyspark.sql.dataframe.DataFrame = [Product_Category: string, Product_ID: long]

3.3 Other Dimensions (Date, Store, etc.)

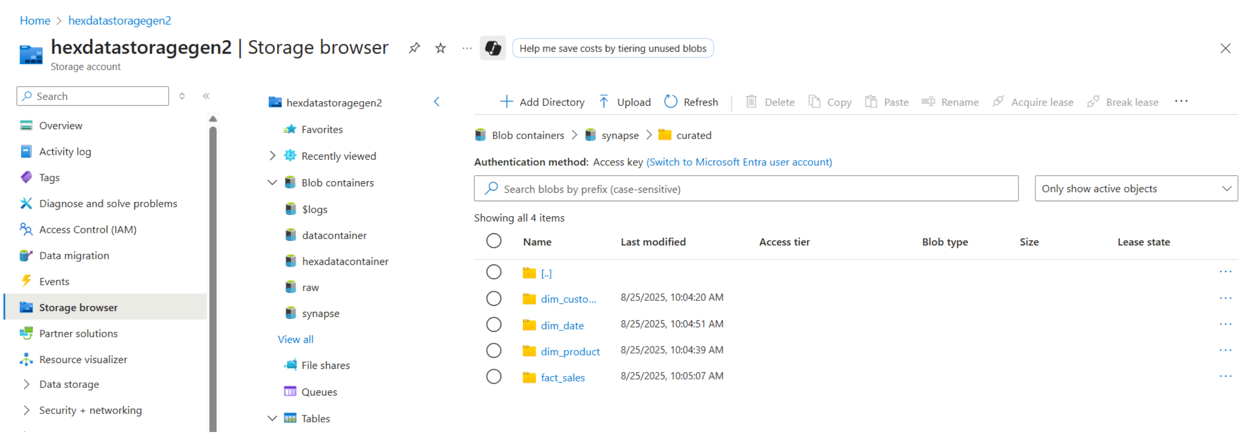
Yesterday (<1s) 17

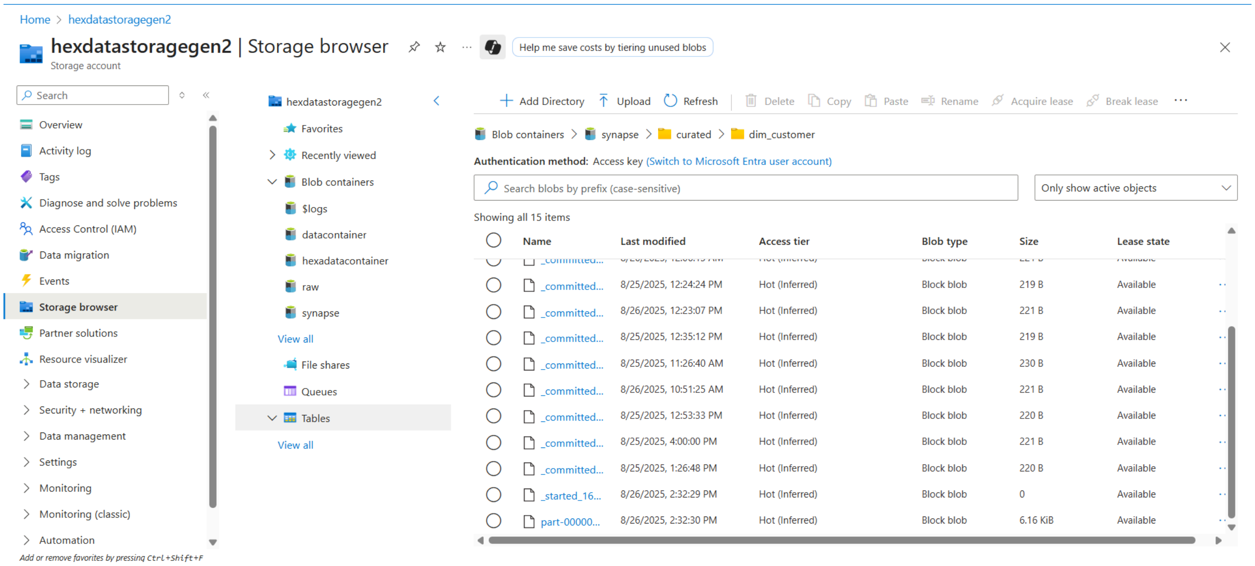
```python
from pyspark.sql.functions import col, to_date, year, month, dayofmonth,
weekofyear, quarter, date_format, monotonically_increasing_id

dim_date = df.withColumn("Date", to_date("Date")) \
             .select(
                 col("Date"),
                 year("Date").alias("Year"),
                 month("Date").alias("Month"),
                 dayofmonth("Date").alias("Day"),
                 weekofyear("Date").alias("Week"),
                 quarter("Date").alias("Quarter"),
                 date_format("Date", "EEEE").alias("DayName"),
                 date_format("Date", "MMMM").alias("MonthName")
             ) \
             .dropDuplicates() \
             .withColumn("Date_ID", monotonically_increasing_id())
```

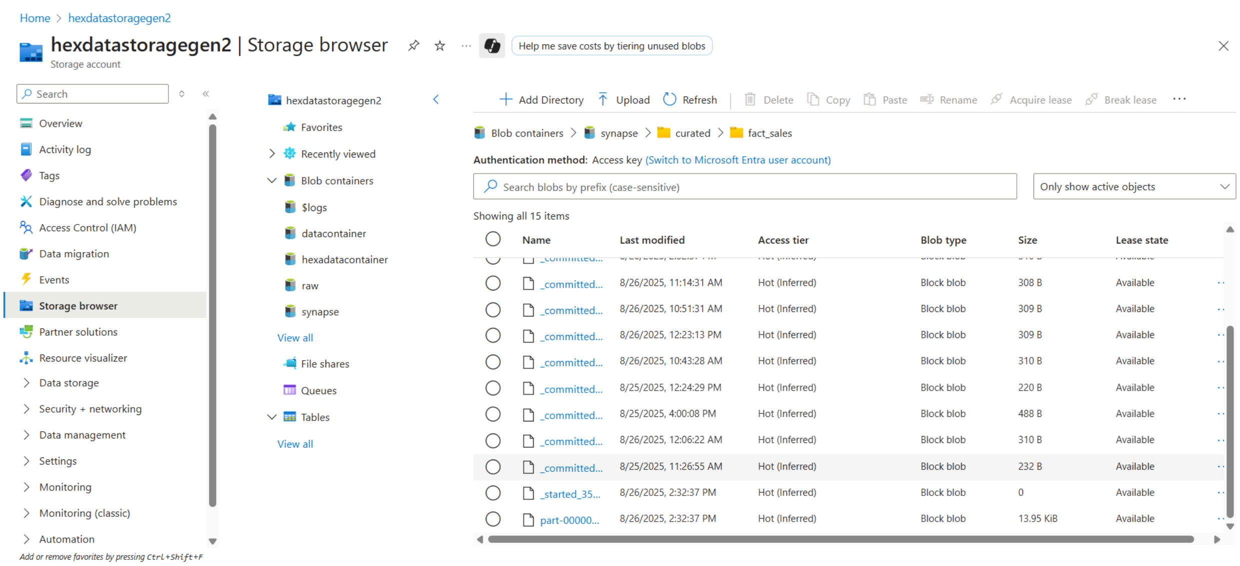
dim_date: pyspark.sql.dataframe.DataFrame = [Date: date, Year: integer ... 7 more fields]



- **Curated Data Stored in ADLS**

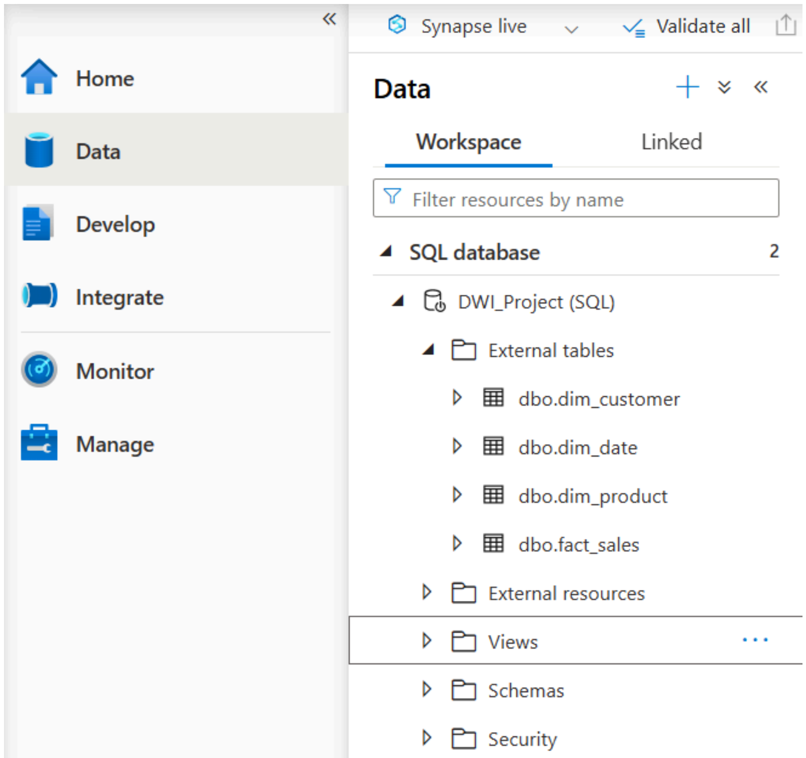



Home > hexdatastoragegen2
hexdatastoragegen2 | Storage browser
Storage account
Help me save costs by tiering unused blobs
Search
Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Partner solutions
Resource visualizer
Data storage
Security + networking
Data management
Settings
Monitoring
Monitoring (classic)
Automation
Add or remove favorites by pressing Ctrl+Shift+F
hexdatastoragegen2
Favorites
Recently viewed
Blob containers
$logs
datacontainer
hexadatacontainer
raw
synapse
View all
File shares
Queues
Tables
View all
Add Directory
Upload
Refresh
Delete
Copy
Paste
Rename
Acquire lease
Break lease
Blob containers > synapse > curated > dim_customer
Authentication method: Access key (Switch to Microsoft Entra user account)
Search blobs by prefix (case-sensitive)
Only show active objects
Showing all 15 items
Name
Last modified
Access tier
Blob type
Size
Lease state
_committed... 8/25/2025, 12:24:24 PM Hot (Inferred) Block blob 219 B Available
_committed... 8/26/2025, 12:23:07 PM Hot (Inferred) Block blob 221 B Available
_committed... 8/25/2025, 12:35:12 PM Hot (Inferred) Block blob 219 B Available
_committed... 8/25/2025, 11:26:40 AM Hot (Inferred) Block blob 230 B Available
_committed... 8/26/2025, 10:51:25 AM Hot (Inferred) Block blob 221 B Available
_committed... 8/25/2025, 12:53:33 PM Hot (Inferred) Block blob 220 B Available
_committed... 8/25/2025, 4:00:00 PM Hot (Inferred) Block blob 221 B Available
_committed... 8/25/2025, 1:26:48 PM Hot (Inferred) Block blob 220 B Available
_started_16... 8/26/2025, 2:32:29 PM Hot (Inferred) Block blob 0 Available
part-00000... 8/26/2025, 2:32:30 PM Hot (Inferred) Block blob 6.16 KiB Available


Home > hexdatastoragegen2
hexdatastoragegen2 | Storage browser
Storage account
Help me save costs by tiering unused blobs
Search
Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Partner solutions
Resource visualizer
Data storage
Security + networking
Data management
Settings
Monitoring
Monitoring (classic)
Automation
Add or remove favorites by pressing Ctrl+Shift+F
hexdatastoragegen2
Favorites
Recently viewed
Blob containers
$logs
datacontainer
hexadatacontainer
raw
synapse
View all
File shares
Queues
Tables
View all
Add Directory
Upload
Refresh
Delete
Copy
Paste
Rename
Acquire lease
Break lease
Blob containers > synapse > curated > fact_sales
Authentication method: Access key (Switch to Microsoft Entra user account)
Search blobs by prefix (case-sensitive)
Only show active objects
Showing all 15 items
Name
Last modified
Access tier
Blob type
Size
Lease state
_committed... 8/26/2025, 11:14:31 AM Hot (Inferred) Block blob 308 B Available
_committed... 8/26/2025, 10:51:31 AM Hot (Inferred) Block blob 309 B Available
_committed... 8/26/2025, 12:23:13 PM Hot (Inferred) Block blob 309 B Available
_committed... 8/26/2025, 10:43:28 AM Hot (Inferred) Block blob 310 B Available
_committed... 8/25/2025, 12:24:29 PM Hot (Inferred) Block blob 220 B Available
_committed... 8/25/2025, 4:00:08 PM Hot (Inferred) Block blob 488 B Available
_committed... 8/26/2025, 12:06:22 AM Hot (Inferred) Block blob 310 B Available
_committed... 8/25/2025, 11:26:55 AM Hot (Inferred) Block blob 232 B Available
_started_35... 8/26/2025, 2:32:37 PM Hot (Inferred) Block blob 0 Available
part-00000... 8/26/2025, 2:32:37 PM Hot (Inferred) Block blob 13.95 KiB Available

2.3 AZURE SYNAPSE ANALYTICS

- **External Tables Registered**



- **Data Source & Credential Setup**

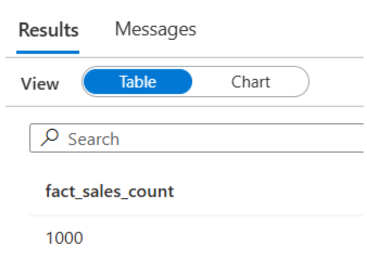
```
---------------------------------------------------
-- STEP 3: Create External Data Source
---------------------------------------------------
IF EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'dwi_curated')
    DROP EXTERNAL DATA SOURCE dwi_curated;

CREATE EXTERNAL DATA SOURCE dwi_curated
WITH (
    LOCATION = 'https://hexdatastoragegen2.blob.core.windows.net/synapse',
    CREDENTIAL = BlobSAS
);
```

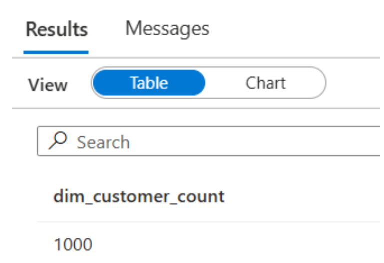
3. VALIDATION OUTPUTS

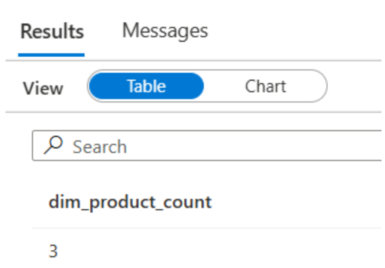
Validation was carried out in Synapse Studio using SQL queries to confirm row counts, schema correctness, and referential integrity.

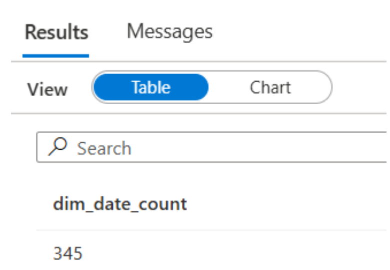
- **Row Count Validation**

```sql
-- 1. Row counts in each table
SELECT COUNT(*) AS fact_sales_count FROM dbo.fact_sales;
SELECT COUNT(*) AS dim_customer_count FROM dbo.dim_customer;
SELECT COUNT(*) AS dim_product_count FROM dbo.dim_product;
SELECT COUNT(*) AS dim_date_count FROM dbo.dim_date;
```



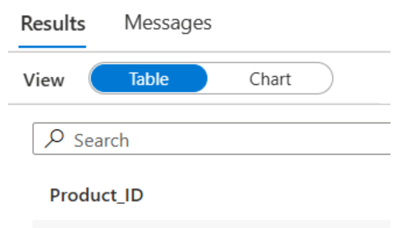




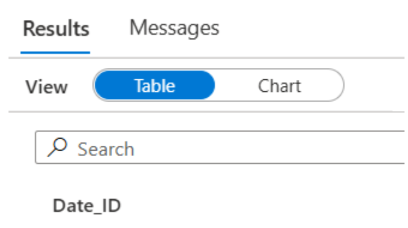


- **Referential Integrity Checks**

```sql
-- 2. Check for orphaned Product_IDs in fact
SELECT DISTINCT f.Product_ID
FROM dbo.fact_sales f
LEFT JOIN dbo.dim_product p ON f.Product_ID = p.Product_ID
WHERE p.Product_ID IS NULL;
```

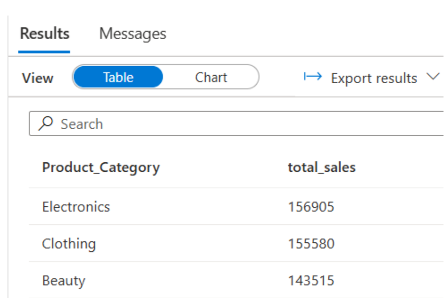



```sql
-- 3. Check for orphaned Date_IDs in fact
SELECT DISTINCT f.Date_ID
FROM dbo.fact_sales f
LEFT JOIN dbo.dim_date d ON f.Date_ID = d.Date_ID
WHERE d.Date_ID IS NULL;
```



- **Sales by Product Category**

```sql
-- 4. Sales by product category
SELECT p.Product_Category, SUM(f.Total_Amount) AS total_sales
FROM dbo.fact_sales f
JOIN dbo.dim_product p ON f.Product_ID = p.Product_ID
GROUP BY p.Product_Category
ORDER BY total_sales DESC;
```



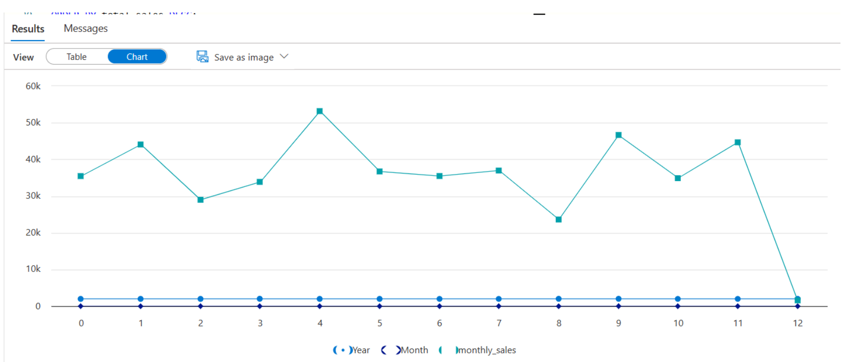
Product_Category	total_sales
Electronics	156905
Clothing	155580
Beauty	143515

- **Monthly Sales Trends**

```sql
-- 6. Sales trend by month (quick time sanity check)
SELECT d.Year, d.Month, SUM(f.Total_Amount) AS monthly_sales
FROM dbo.fact_sales f
JOIN dbo.dim_date d ON f.Date_ID = d.Date_ID
GROUP BY d.Year, d.Month
ORDER BY d.Year, d.Month;
```

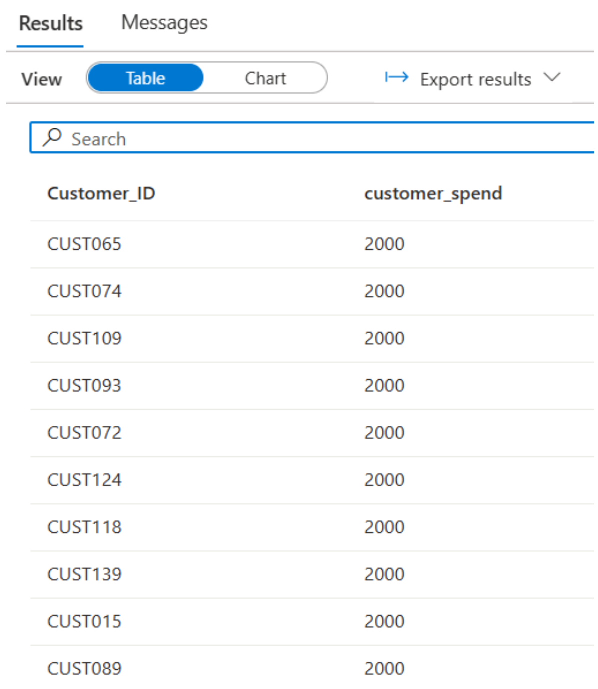
Search

Year	Month	monthly_sales
2023	1	35450
2023	2	44060
2023	3	28990
2023	4	33870
2023	5	53150



- **Top 10 Customers by Spend**

```sql
-- 8. Top 10 customers by spend
SELECT TOP 10 f.Customer_ID, SUM(f.Total_Amount) AS customer_spend
FROM dbo.fact_sales f
GROUP BY f.Customer_ID
ORDER BY customer_spend DESC;
```



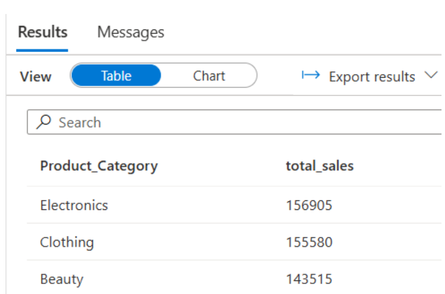
Customer_ID	customer_spend
CUST065	2000
CUST074	2000
CUST109	2000
CUST093	2000
CUST072	2000
CUST124	2000
CUST118	2000
CUST139	2000
CUST015	2000
CUST089	2000

4. ANALYSIS & INSIGHTS

The validated star schema enables rich analytical queries. Some business insights observed:

- **Product Category Contribution**: Certain categories dominate sales.

```sql
-- 4. Sales by product category
SELECT p.Product_Category, SUM(f.Total_Amount) AS total_sales
FROM dbo.fact_sales f
JOIN dbo.dim_product p ON f.Product_ID = p.Product_ID
GROUP BY p.Product_Category
ORDER BY total_sales DESC;
```



Product_Category	total_sales
Electronics	156905
Clothing	155580
Beauty	143515

- **Customer Segmentation**: Spending patterns vary across gender and age groups.

```sql
-- 5. Sales by gender
SELECT c.Gender, SUM(f.Total_Amount) AS total_sales
FROM dbo.fact_sales f
JOIN dbo.dim_customer c ON f.Customer_ID = c.Customer_ID
GROUP BY c.Gender;
```

Results Messages

View Table Chart Export results

Search

Gender	total_sales
Female	232840
Male	223160

```sql
-- 7. Age group sales contribution
SELECT
    CASE
        WHEN c.Age BETWEEN 18 AND 25 THEN '18-25'
        WHEN c.Age BETWEEN 26 AND 35 THEN '26-35'
        WHEN c.Age BETWEEN 36 AND 50 THEN '36-50'
        ELSE '50+'
    END AS Age_Group,
    SUM(f.Total_Amount) AS total_sales
FROM dbo.fact_sales f
JOIN dbo.dim_customer c ON f.Customer_ID = c.Customer_ID
GROUP BY CASE
        WHEN c.Age BETWEEN 18 AND 25 THEN '18-25'
        WHEN c.Age BETWEEN 26 AND 35 THEN '26-35'
        WHEN c.Age BETWEEN 36 AND 50 THEN '36-50'
        ELSE '50+'
    END
ORDER BY total_sales DESC;
```

Results Messages

View Table Chart Export results

Search

Age_Group	total_sales
36-50	139660
50+	133310
26-35	98480
18-25	84550

- **Revenue Trends Over Time**: Sales volume shows monthly variations, enabling time-series analysis.

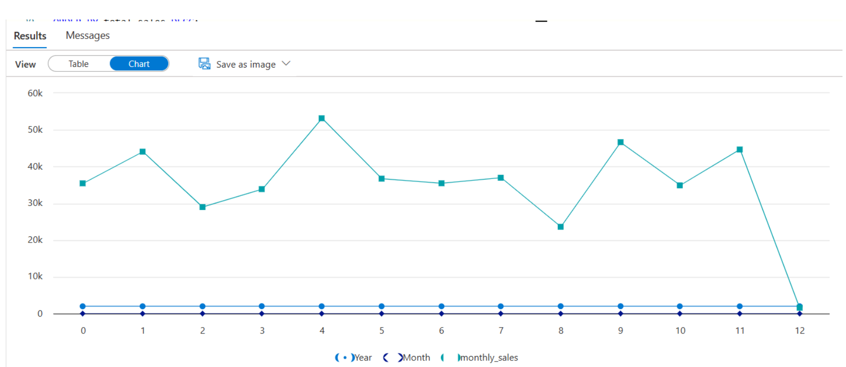
```sql
-- 6. Sales trend by month (quick time sanity check)
SELECT d.Year, d.Month, SUM(f.Total_Amount) AS monthly_sales
FROM dbo.fact_sales f
JOIN dbo.dim_date d ON f.Date_ID = d.Date_ID
GROUP BY d.Year, d.Month
ORDER BY d.Year, d.Month;
```

Results Messages

View Table Chart Export results

Search

Year	Month	monthly_sales
2023	1	35450
2023	2	44060
2023	3	28990
2023	4	33870
2023	5	53150



5. CONCLUSION

The project successfully delivered:

- Automated ETL pipelines (ADF orchestrated).
- Scalable data transformations (Databricks PySpark).
- Optimized loading into a **Synapse star schema** (fact + dimensions).
- Validation queries confirming data integrity and usability for analytics.

This system is now **analytics-ready**, with potential for Power BI integration to deliver dashboards and business reporting.
