DATA WAREHOUSING INTEGRATION

[*Date of Submission: 29th August 2025*]

TABLE OF CONTENTS

---

1. INTRODUCTION

1.1 PURPOSE

The purpose of this project is to design and implement a data warehousing integration solution using Microsoft Azure services. The solution automates data ingestion, transformation, and loading (ETL) from raw sources into a centralized warehouse, enabling scalable and reliable analytics for business users.

1.2 BACKGROUND

In the retail and e-commerce industry, large amounts of data are generated every day from transactions, customers, and product interactions. Without a consolidated data warehouse, this data remains fragmented, making it difficult to answer key business questions such as sales performance, customer behavior, and product profitability. By leveraging Azure services, this project addresses these gaps with a modern cloud-based data warehouse.

1.3 SCOPE

- **In-Scope:** Ingesting raw CSV files, transforming datasets into curated formats, creating star schema models in Synapse, and validating data through SQL queries.

- **Out-of-Scope:** Real-time streaming pipelines, advanced machine learning, and external API integrations.

2. PROBLEM STATEMENT

Organizations often struggle to consolidate fragmented datasets across multiple systems, leading to reporting delays, inconsistent metrics, and poor decision-making. This project addresses the problem by building an automated ETL pipeline that ensures clean, structured, and analytics-ready data is always available in Azure Synapse.

3. OBJECTIVES

- Automate data ingestion from source to data lake.

- Transform data into structured fact and dimension tables.

- Load curated datasets into Synapse for analytics.

- Validate results to ensure data quality and consistency.

- Establish a scalable pipeline framework for future BI reporting.

4. TOOLS & TECHNOLOGIES USED

4.1 AZURE DATA FACTORY (ADF)

Used for orchestrating data pipelines, managing linked services, and scheduling ETL processes.

4.2 AZURE DATA LAKE STORAGE (ADLS)

Serves as the centralized data repository with structured zones: *raw* for ingestion and *curated* for transformed outputs.

4.3 AZURE DATABRICKS (PYSPARK)

Provides scalable data transformation capabilities, enabling cleaning, enrichment, and preparation of datasets for Synapse.

4.4 AZURE SYNAPSE ANALYTICS

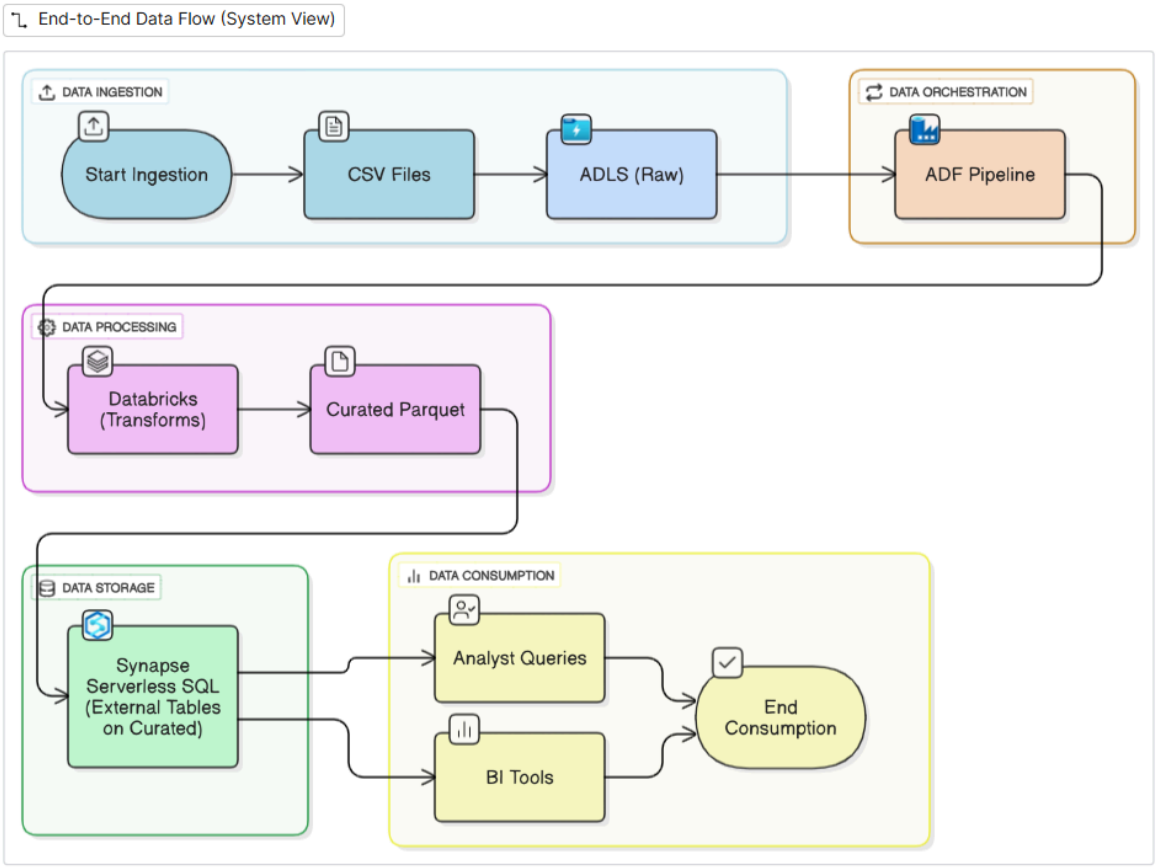
Acts as the data warehouse, hosting fact and dimension tables through external table mappings to curated data.

5. SOLUTION ARCHITECTURE

5.1 HIGH-LEVEL DESIGN

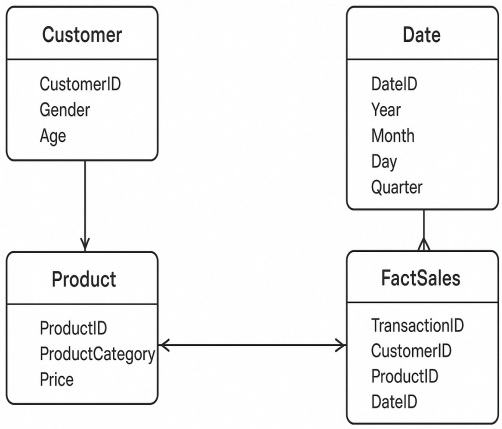
The architecture follows a modular ETL approach: data is ingested by ADF, stored in ADLS, transformed in Databricks, and loaded into Synapse for reporting.

5.2 DATA FLOW DIAGRAM



5.3 ENTITY RELATIONSHIP DIAGRAM (ERD)

- **DimCustomer** (CustomerKey, CustomerID, Name, Gender, Age).
- **DimProduct** (ProductKey, ProductID, ProductName, Category, Price).
- **DimDate** (DateKey, Date, Year, Month, Day).
- **FactSales** (SalesKey, CustomerKey, ProductKey, DateKey, Quantity, TotalAmount).



6. EXECUTION OVERVIEW

6.1 ENVIRONMENT SETUP

Provisioned ADLS, ADF, Databricks, and Synapse workspaces. Created storage containers (`raw/`, `curated/`). Configured linked services and secured access with SAS tokens.

6.2 DATA INGESTION

Implemented ADF pipelines to copy raw CSVs into ADLS staging. Handled credential issues by resetting admin details and correcting Synapse server configuration.

6.3 DATA TRANSFORMATION

Developed PySpark scripts in Databricks to clean nulls, remove duplicates, calculate sales metrics, and create fact/dimension tables. Outputs were stored in Parquet format under `/curated/`.

6.4 DATA LOADING

Created scoped credentials and external data sources in Synapse. Registered external tables (`dim_customer`, `dim_product`, `dim_date`, `fact_sales`) mapped to curated Parquet files.

6.5 VALIDATION & TESTING

Ran SQL queries to validate row counts, integrity between fact and dimension tables, and perform aggregations (e.g., sales by product, customer, gender, and month).

6.6 ORCHESTRATION & MONITORING

Configured end-to-end orchestration in ADF. Pipelines executed in sequence (Ingest → Transform → Load → Validate). Monitoring dashboards confirmed successful runs.

7. TASKS PERFORMED

7.1 ENVIRONMENT SETUP TASKS

- Created ADLS with hierarchical namespace.

- Provisioned Databricks workspace and linked it to ADLS.
- Set up Synapse workspace and SQL pools.

7.2 DATA INGESTION TASKS

- Configured ADF linked services for ADLS and Synapse.
- Built pipelines to move raw CSVs into staging.

7.3 TRANSFORMATION TASKS

- Implemented PySpark jobs for data cleaning and enrichment.
- Generated curated fact and dimension tables.

7.4 DATA LOADING TASKS

- Registered external data sources and credentials in Synapse.
- Created external tables referencing curated Parquet data.

7.5 VALIDATION QUERIES & RESULTS

- Validated row counts in Synapse.
- Confirmed fact-dimension relationships.
- Verified aggregations matched curated outputs.

7.6 PIPELINE ORCHESTRATION TASKS

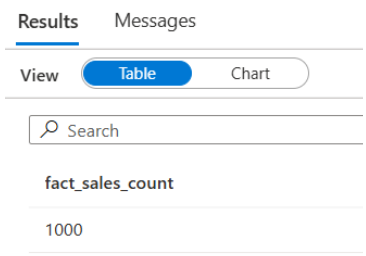
- Designed ADF workflows with retry logic.
- Debugged errors in linked services and external tables.

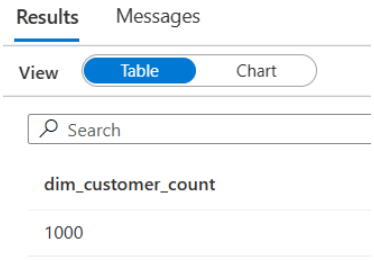
8. RESULTS & INSIGHTS

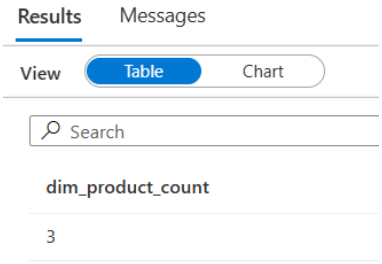
8.1 ROW COUNTS

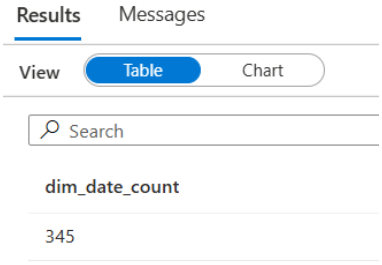
Validated successful ingestion with expected row counts across fact and dimension tables.

```
-- 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;
```





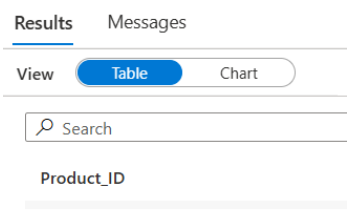




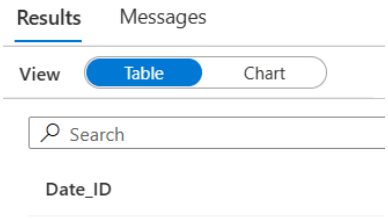
8.2 REFERENTIAL INTEGRITY CHECKS

Confirmed no orphaned records existed between fact and dimension tables.

```
-- 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;
```



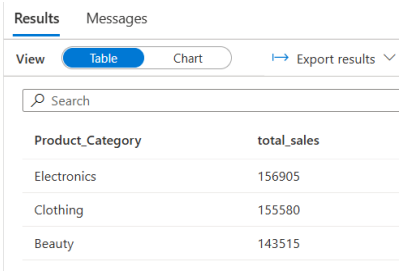
```sql
-- 3. Check for orphaned Date_IDs in fact
SELECT DISTINCT f.Date_ID
FROM dbo.fact_sales f
LEFT JOIN dbo.dim_date d ON f.Date_ID = d.Date_ID
WHERE d.Date_ID IS NULL;
```



8.3 AGGREGATED INSIGHTS

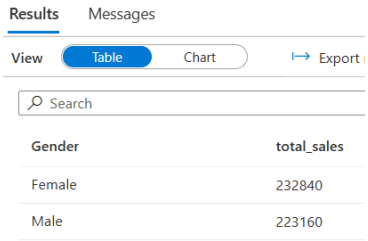
- Top-selling products identified.

```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;
```

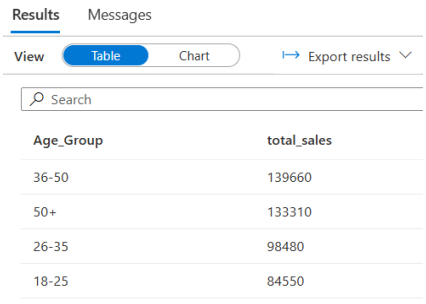


- Sales trends by gender and age groups.

```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;
```

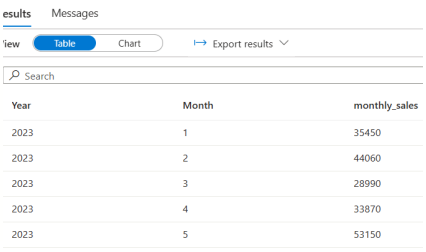



```sql
-- 7. Age group sales contribution
SELECT
    CASE
        WHEN c.Age BETWEEN 18 AND 25 THEN '18-25'
        WHEN c.Age BETWEEN 26 AND 35 THEN '26-35'
        WHEN c.Age BETWEEN 36 AND 50 THEN '36-50'
        ELSE '50+'
    END AS Age_Group,
    SUM(f.Total_Amount) AS total_sales
FROM dbo.fact_sales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



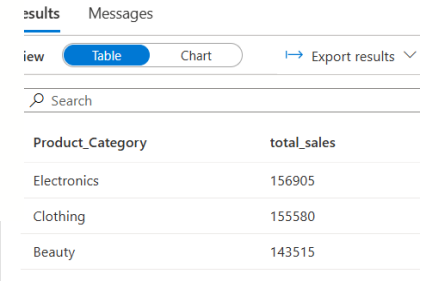
- Revenue patterns by month and product category.

```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;
```



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

```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

9. KEY LEARNINGS

- SAS token scoping must be carefully managed.
- Iterative debugging improves reliability of external tables.
- Modular ETL design improves scalability and maintainability.

10. CHALLENGES & SOLUTIONS

10.1 SAS TOKEN ISSUES

Resolved by regenerating Blob-only SAS tokens with proper permissions.

10.2 EXTERNAL TABLE ACCESSIBILITY ERRORS

Fixed by remapping credentials and data sources in Synapse.

10.3 CREDENTIAL & DATA SOURCE DEPENDENCIES

Addressed by removing existing dependencies before recreating credentials.

10.4 SCHEMA MISMATCHES

Resolved by aligning data types and ensuring schema consistency across transformations.

11. DELIVERABLES

- Requirement Document
- Project Overview Document
- Execution Overview Document
- Results & Insights Document
- Final Project Document
- PowerPoint Presentation

12. FUTURE SCOPE

- Enable real-time ingestion pipelines.
- Integrate Power BI dashboards for business reporting.
- Extend the warehouse to support machine learning workloads.

13. CONCLUSION

The project successfully implemented an automated data warehousing pipeline integrating ADF, Databricks, ADLS, and Synapse. With curated datasets structured into a star schema, the solution provides reliable, scalable, and secure foundations for analytics. This project not only met its objectives but also laid the groundwork for future BI and advanced analytics initiatives.
