# ****NYC Taxi Data Pipeline using Azure Data Factory & Databricks****

Personal Project Showcase

## ****1. Executive Summary****

This project demonstrates the design and implementation of an **end-to-end data engineering pipeline** for processing the NYC Taxi dataset using **Azure Data Factory (ADF), Databricks, and Azure Data Lake Storage (ADLS)**.

The pipeline ingests raw trip records, applies transformations for data quality and analytics readiness, and stores results in optimized **Parquet format**. Databricks was used for further analysis and generating insights such as vendor performance, payment trends, and trip distribution.

## ****2. Problem Statement****

The NYC Taxi dataset contains millions of trip records with diverse attributes such as vendor IDs, payment methods, fares, and trip distances.

-Raw data is often **inconsistent** and not ready for direct analytics.

-Manual data preparation leads to **scalability challenges**.

-Businesses require **automated pipelines** to ensure reliability, consistency, and efficiency.

-The objective was to build a solution that automates ingestion, cleaning, and transformation, while enabling analysis-ready datasets for reporting.

## ****3. Objectives****

-Ingest raw CSV trip data from external sources into **Azure Data Lake Storage (ADLS)**.

-Automate pipelines in **Azure Data Factory (ADF)** with parameterization for flexibility.

-Perform filtering, aggregation, and schema enforcement using **ADF Mapping Data Flows**.

-Store curated datasets in ****Parquet format**** for performance and scalability.

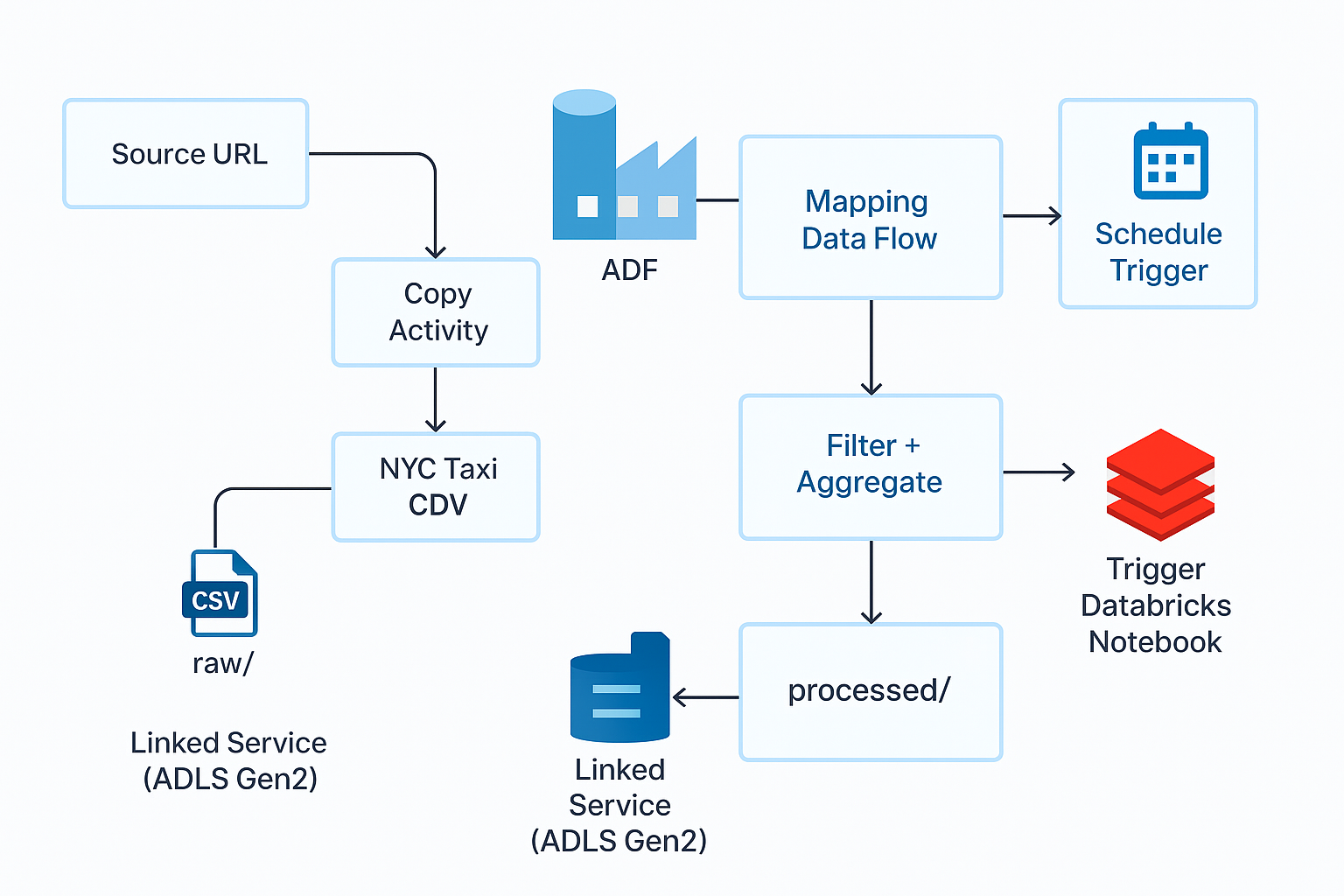
-Conduct analysis in **Databricks notebooks** to extract business insights.

-Schedule pipeline execution with **triggers** for automation.

## System Architecture

- Source: (NYC Taxi CSV endpoint) parameterized URL.  
- ADF: Pipeline with Copy Activity → landing raw folder; Mapping Data Flow for transformation/aggregation → processed Parquet; Databricks Notebook activity for analytics.  
- Storage: ADLS Gen2 container with `raw/` and `processed/` folders.  
- Compute: Azure Databricks for analysis.

**Visual Representation of the Architecture:**



## Azure Resources and Linked Services

Azure services used in the project:  
- Azure Data Factory (v2)  
- Azure Data Lake Storage Gen2 (Linked Service: LS\_ADLS\_NYCTaxi)  
- Azure Databricks (Linked Service: LS\_Databricks)  
- Integration Runtime: AutoResolveIntegrationRuntime

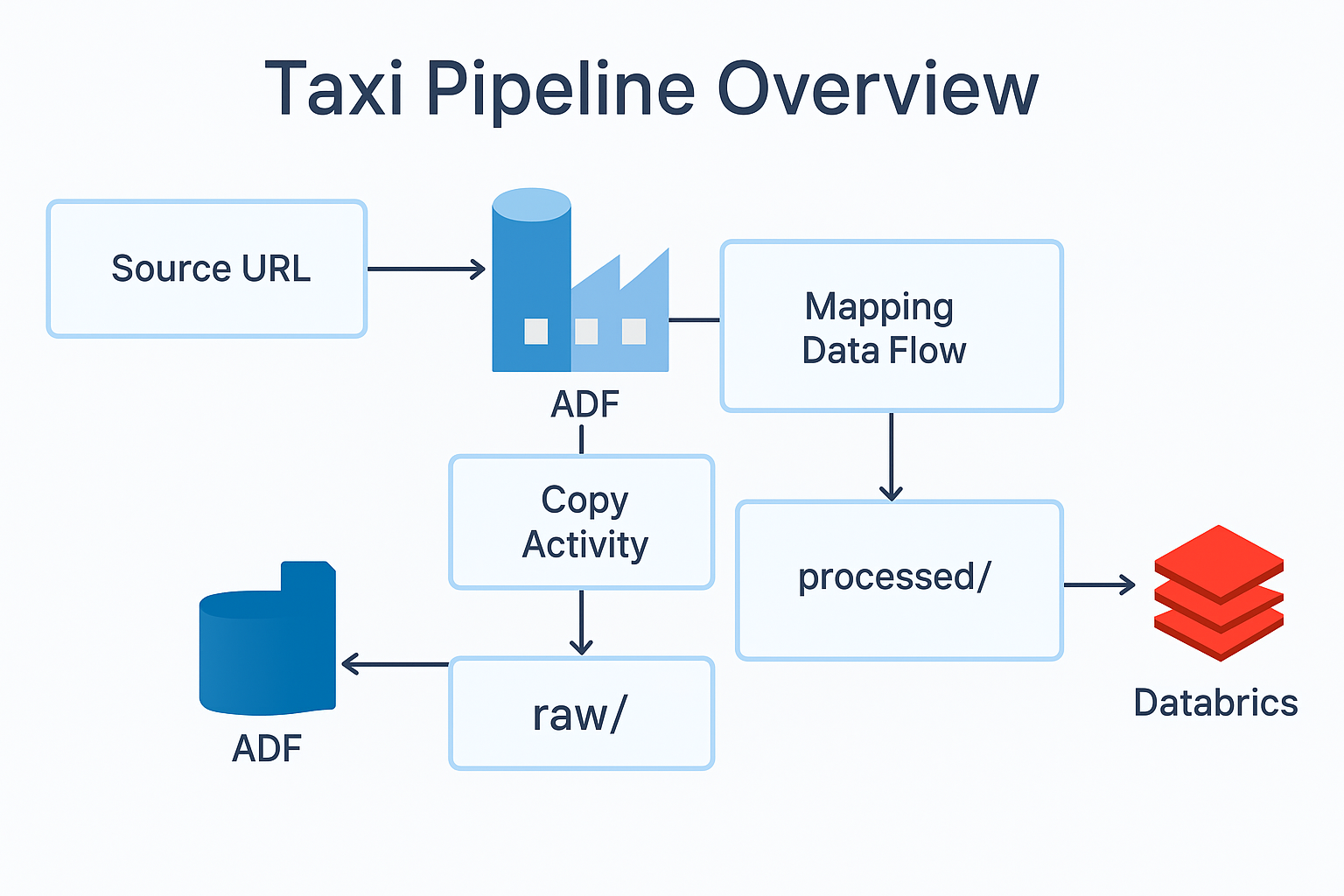
# **Datasets**

1. DS\_NYC\_Taxi\_CSV - linked service dataset.  
2. DS\_Raw\_ADLS — ADLS Gen2 linked dataset storing raw CSV.  
3. DS\_Processed\_Parquet — ADLS Gen2 linked dataset storing processed Parquet.

# 7. Pipeline Design

Pipeline Name: NYC\_Taxi\_Project\_Pipeline  
Parameters: Source URL, Filter Threshold   
Activities: Copy Activity (API → raw/), Mapping Data Flow (filter + aggregate), Databricks Notebook Activity (analyze Parquet)  
Trigger: Weekly schedule

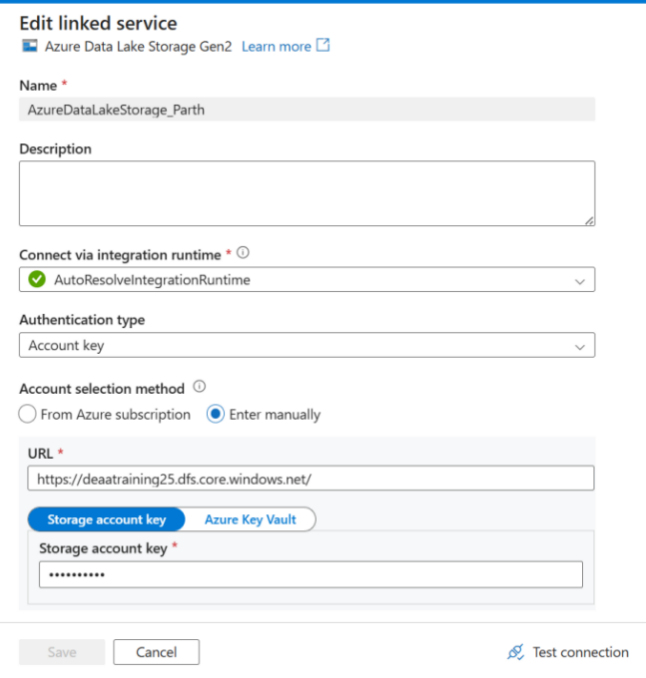
**Visual Representation of Pipeline Design:**

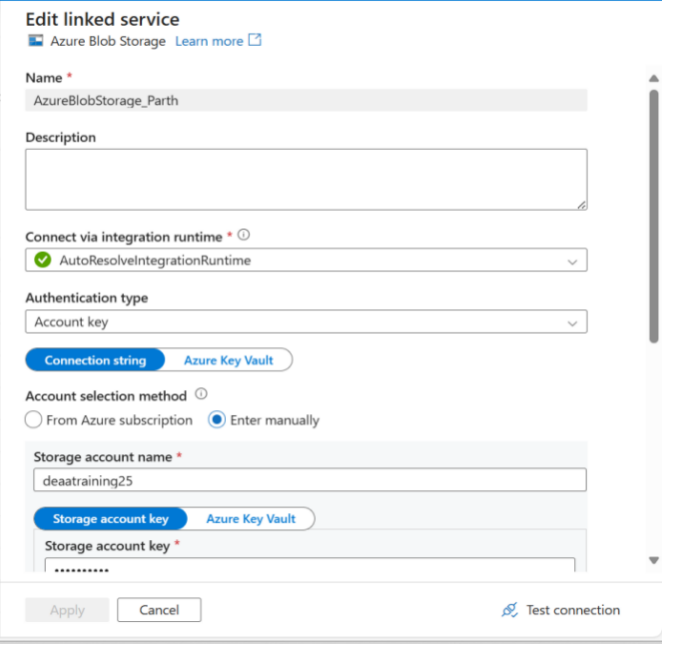


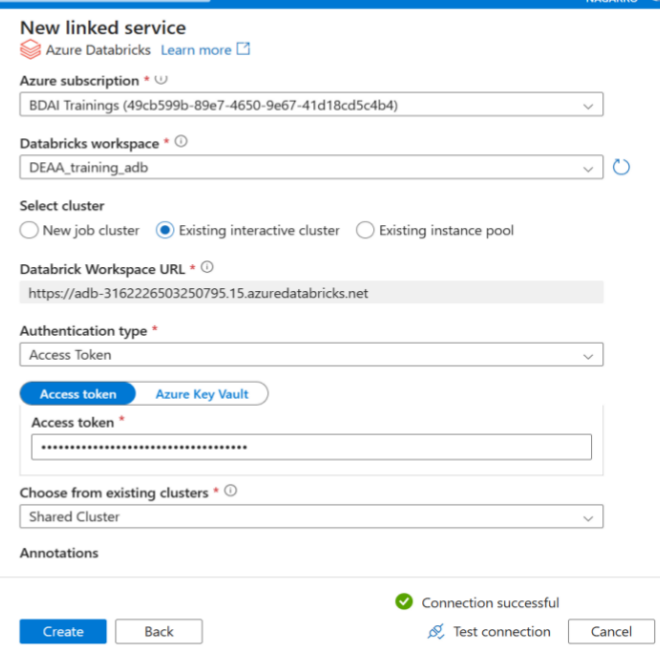
## **8**. Implementation Details****

**Step 1: Configured Linked Services**

Created secure connections for **ADLS, Databricks, and Blob Storage** using access keys/tokens.



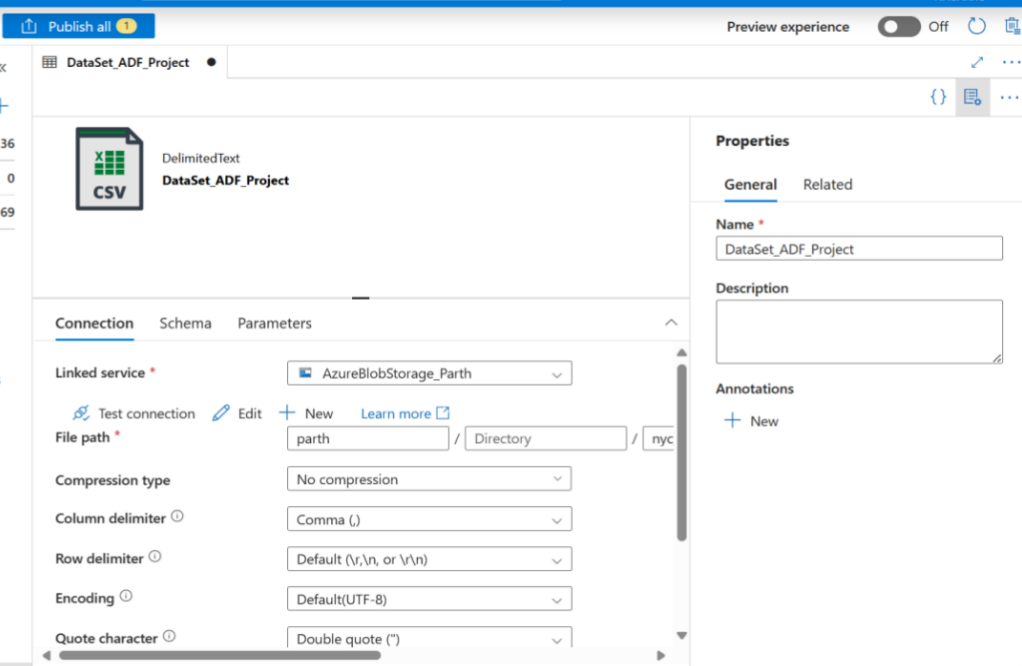


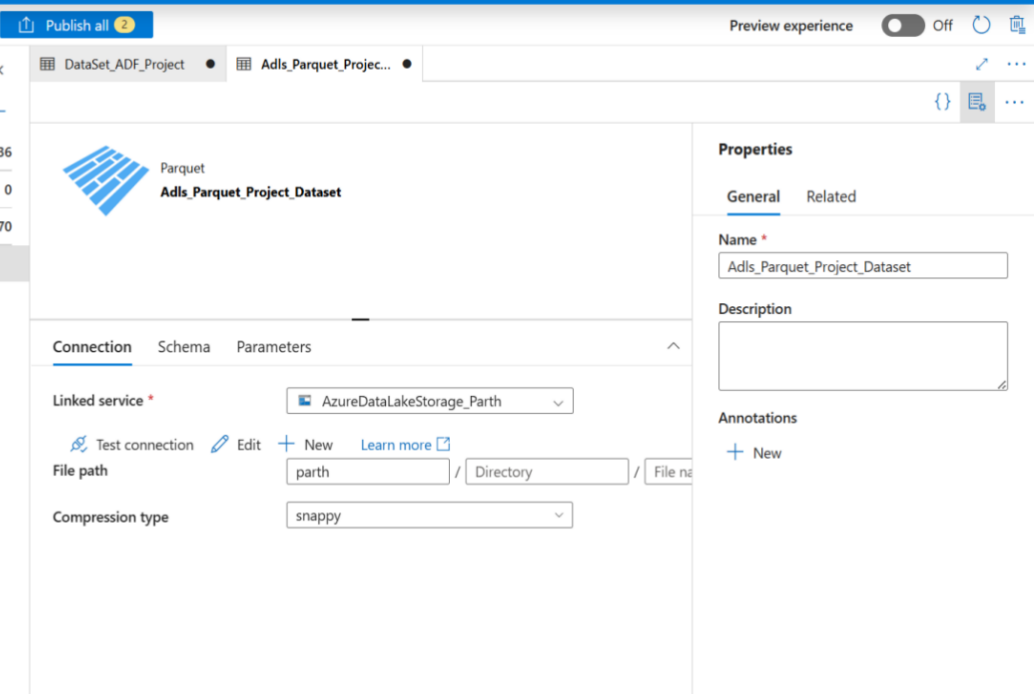


**Step 2: Defined Datasets**

Raw Dataset → pointed to raw/ container for CSV files.

Processed Dataset → pointed to processed/ container for storing Parquet outputs.

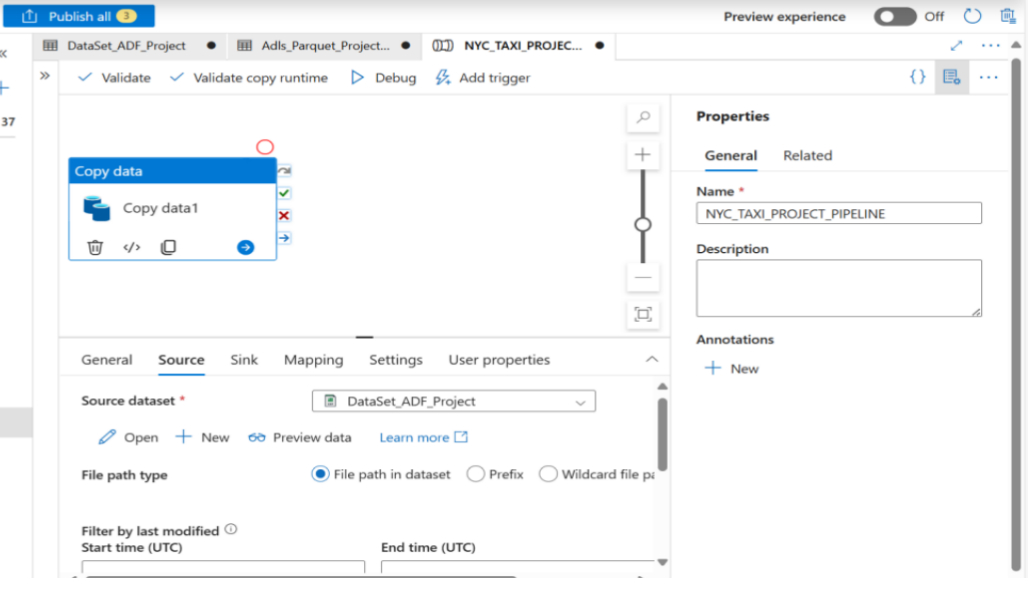


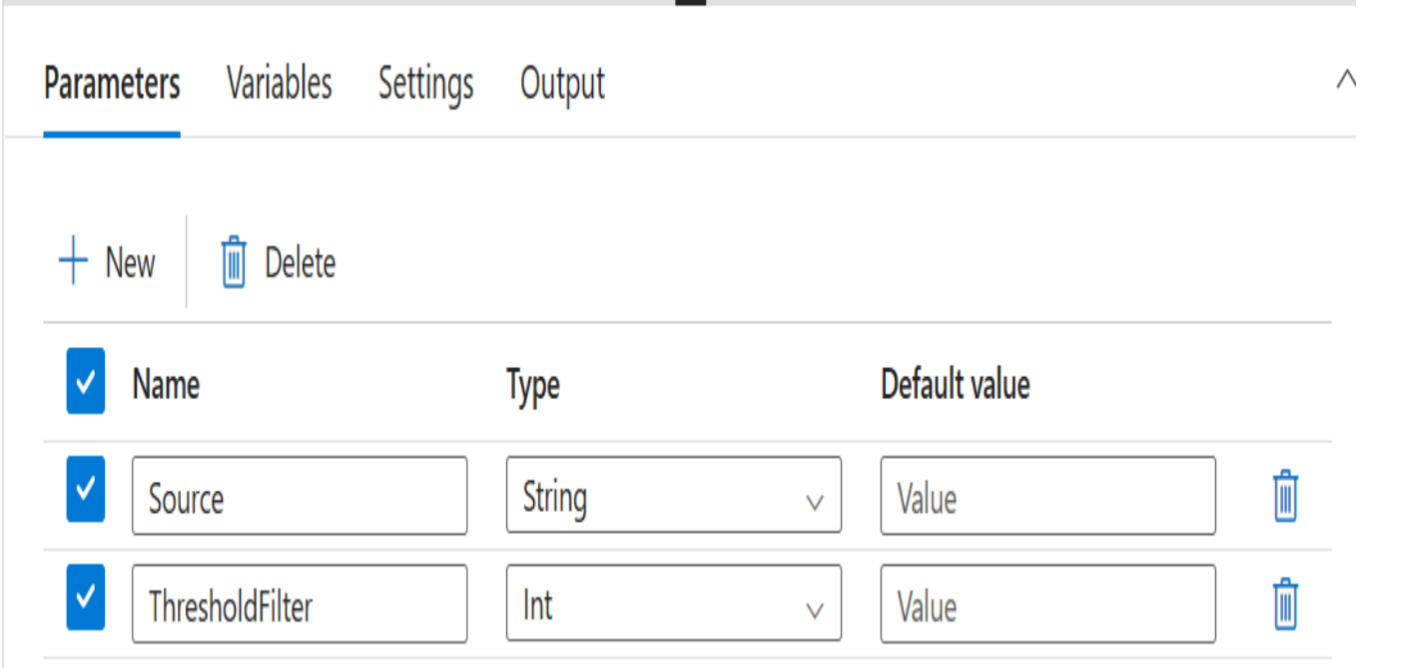


**Step 3: Copy Activity with Parameters**

-Ingested NYC Taxi data using a **Copy Data activity**.

-Used **pipeline parameters** for source URLs and file names, enabling reusability.



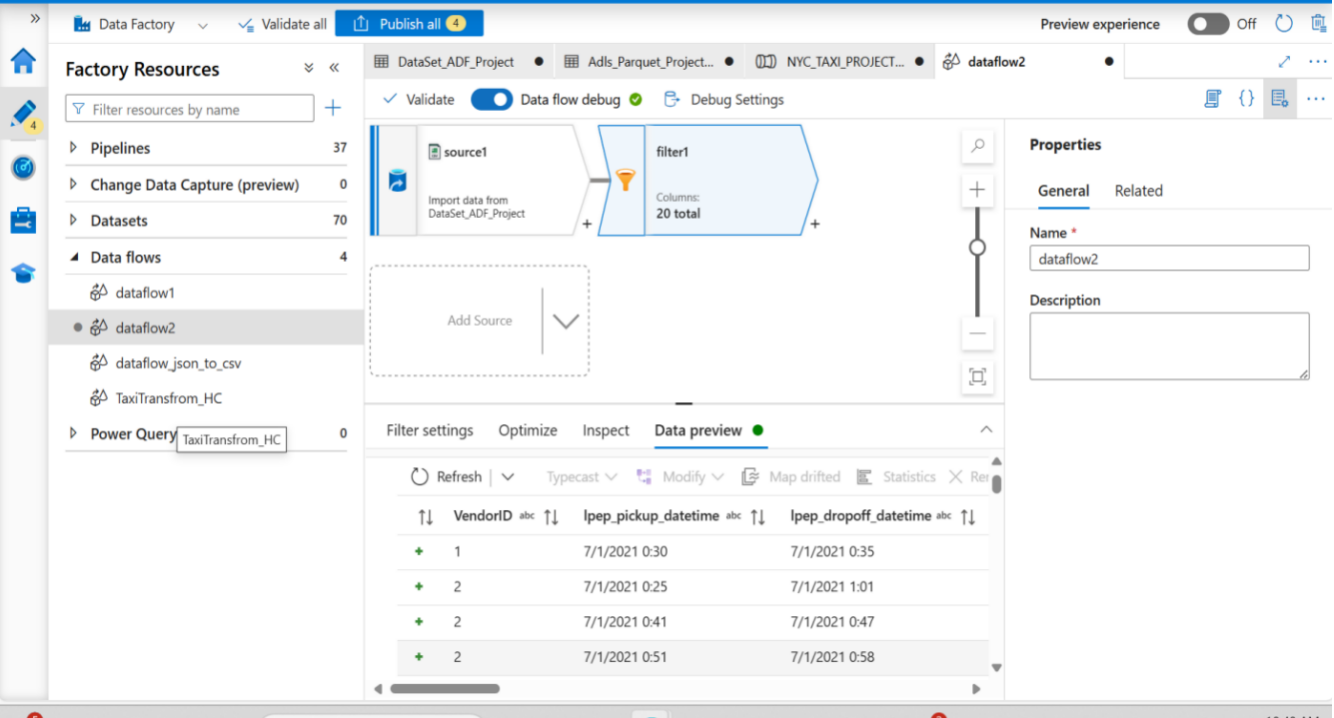


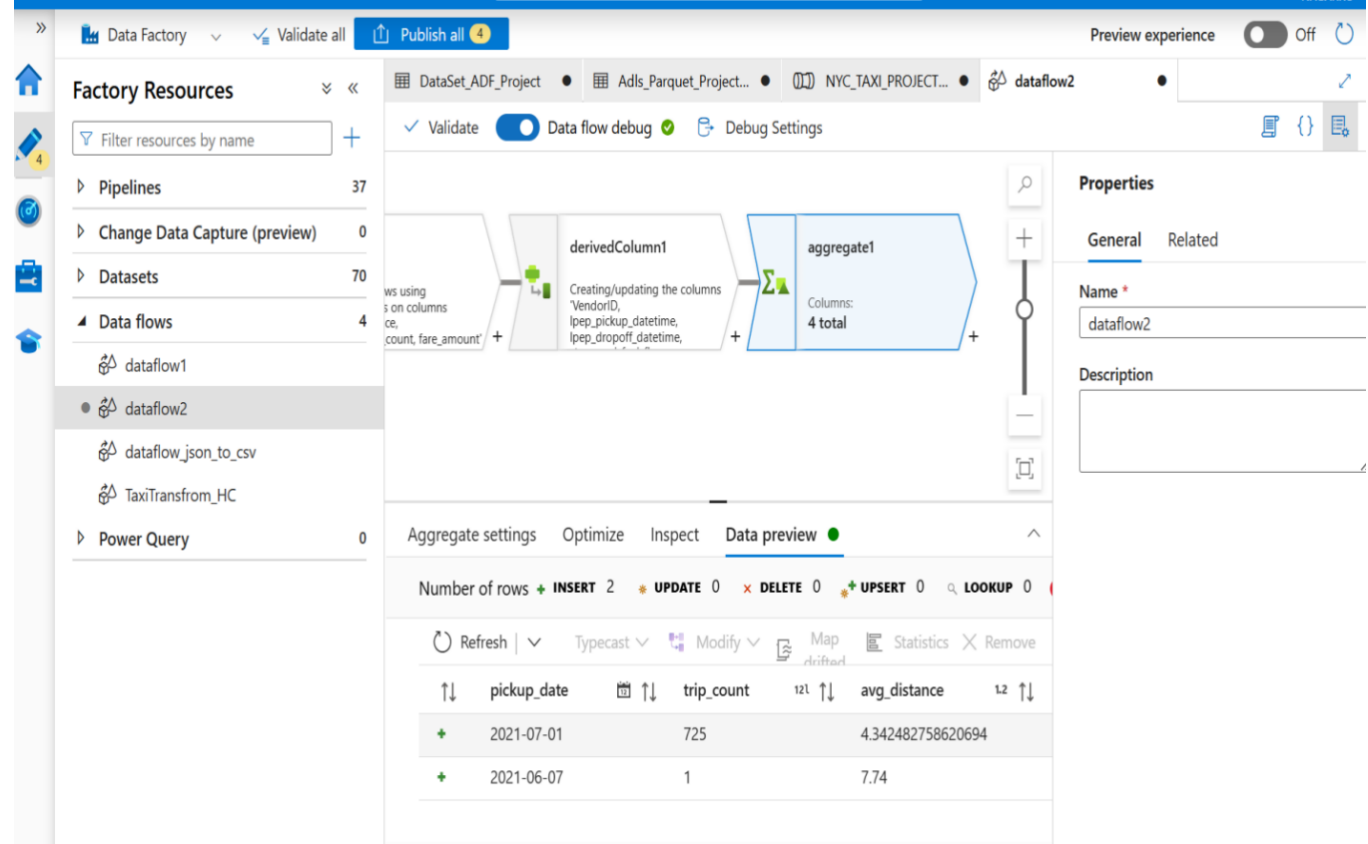
**Step 4: Mapping Data Flow Transformations**

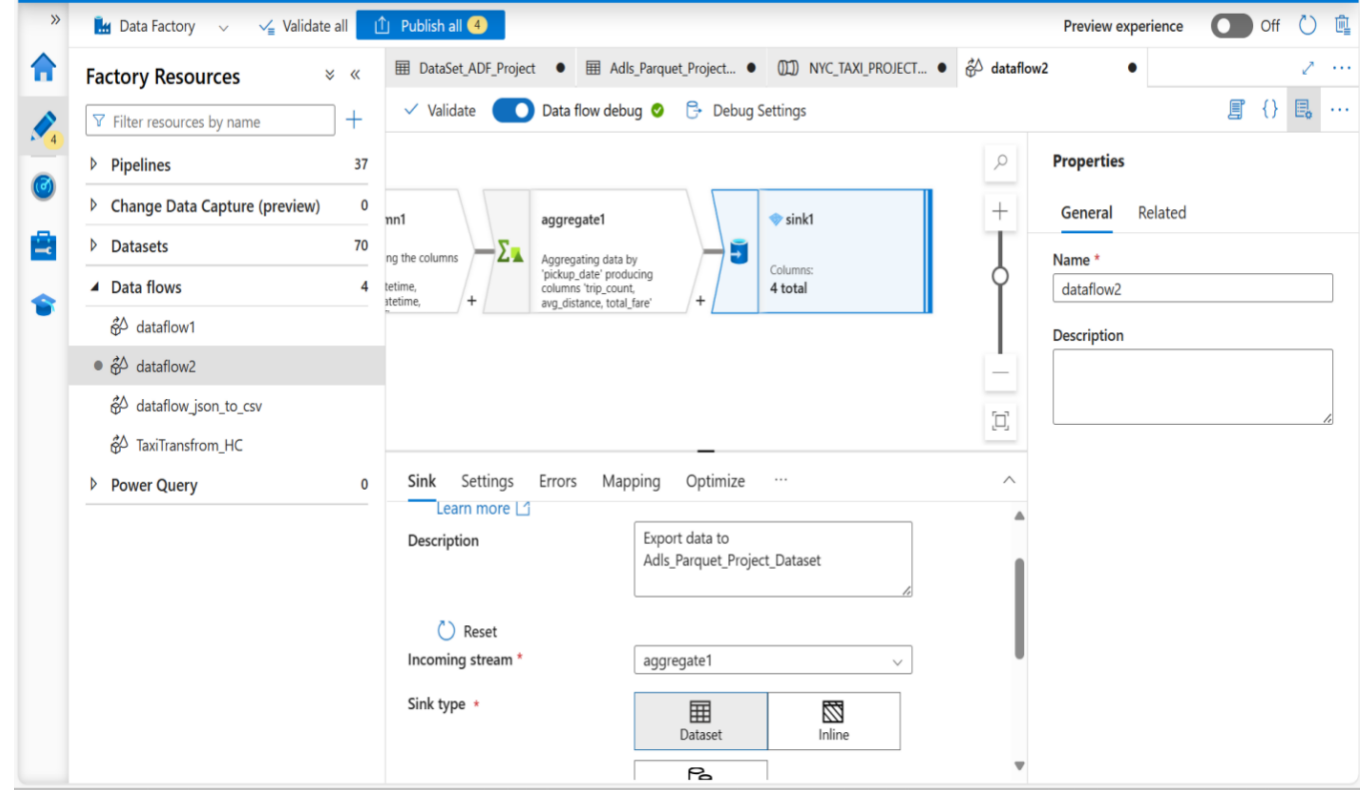
-Applied **Filter transformation** to exclude trips with very low fares or distance.

-Applied **Aggregate transformation** to group by vendor and payment type, calculating ride counts and average fares.

-Wrote transformed data into the **processed/Parquet** container.







**Step 5: Databricks Notebook for Analysis**

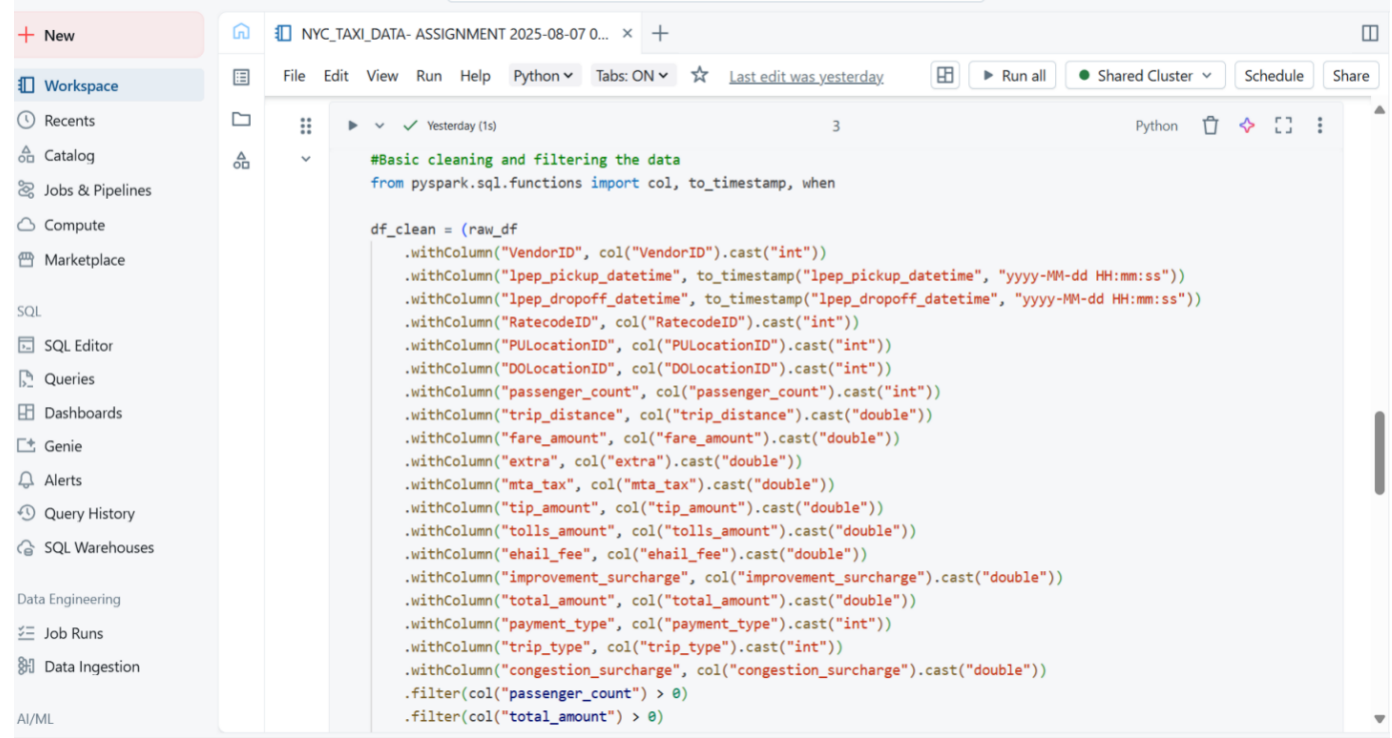
-Connected Databricks notebook to processed data.

-Generated insights:

-Vendor 2 handled the majority of rides.

-Card payments dominated higher fare trips, while cash dominated shorter rides.

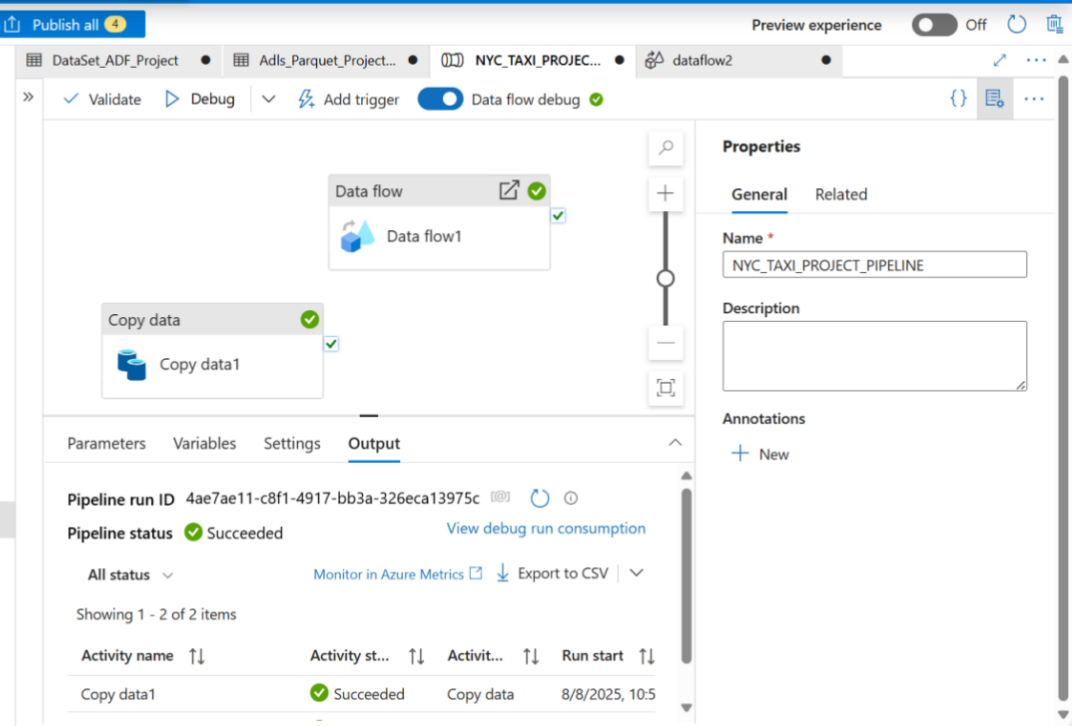
-Peak hours were identified around evening times.

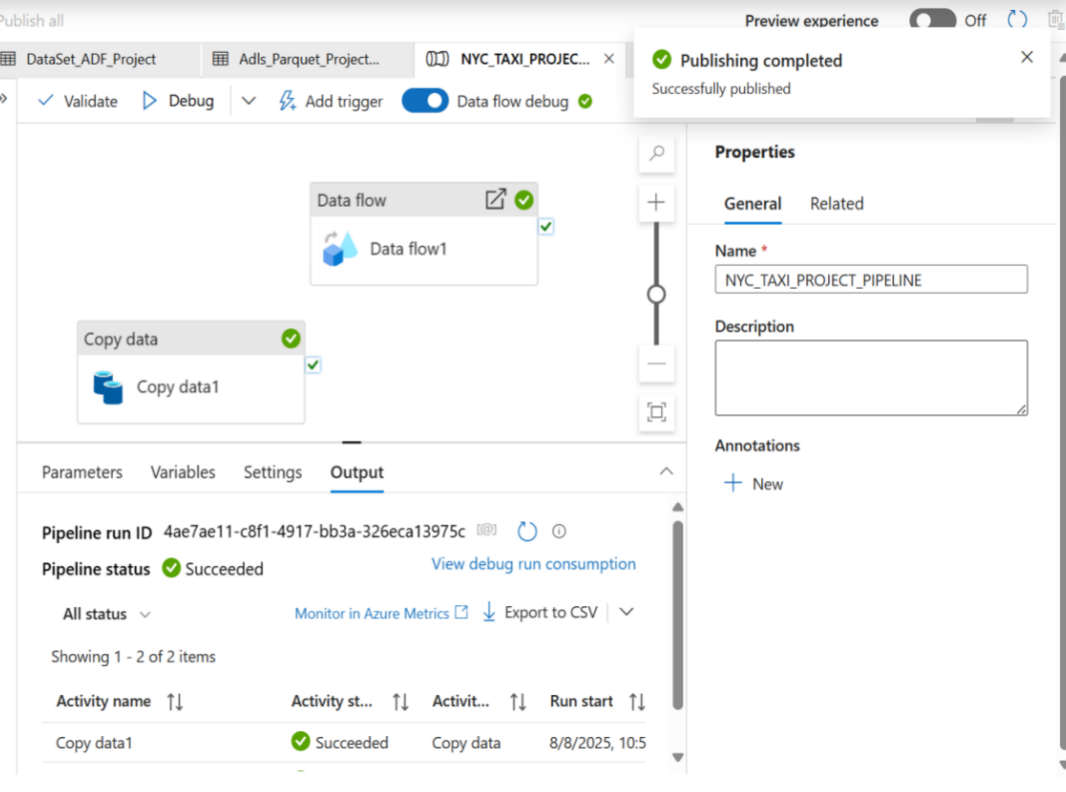


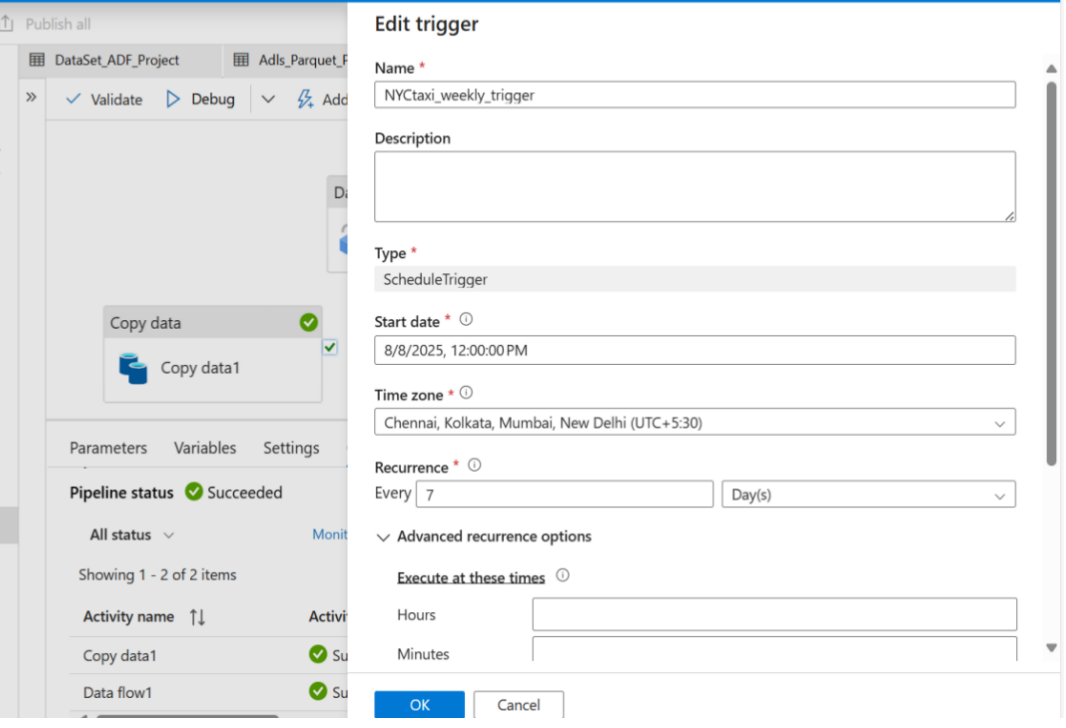
**Step 6: Automation & Scheduling**

-Debugged and published pipeline in ADF.

-Added a **weekly trigger** for automated execution.







## ****9. Results & Key Insights****

✅ Automated pipeline successfully ingests and processes millions of trip records.

✅ Processed data is available in **Parquet format**, ready for analytics and BI tools.

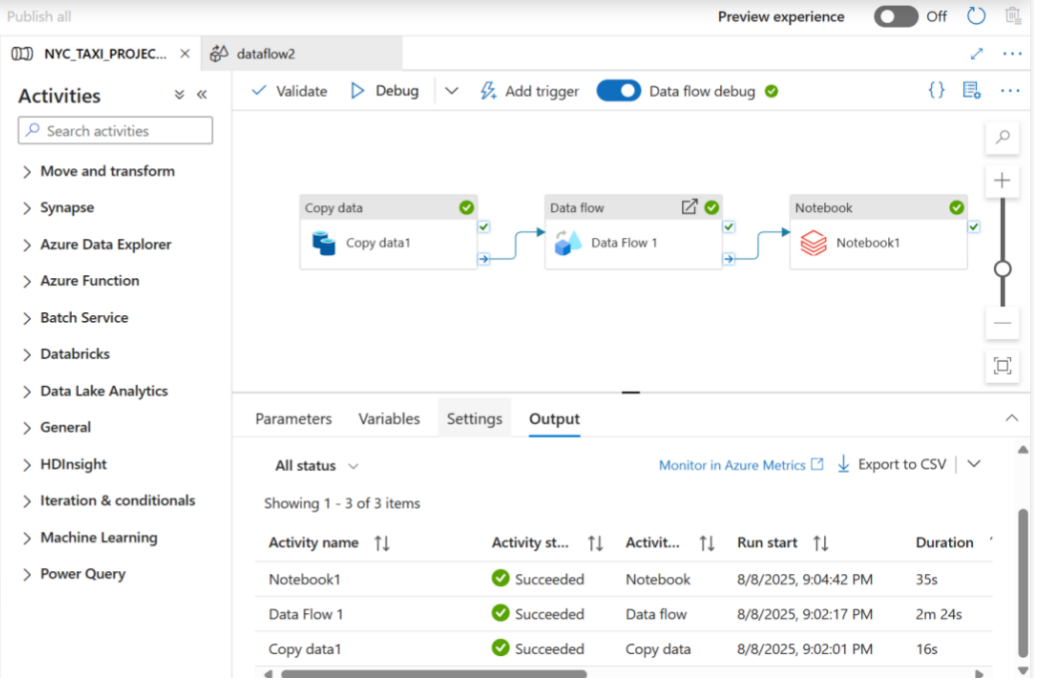
✅ Business insights generated:

-Vendor 2 contributes ~60% of rides.

-Credit card payments account for ~70% of revenue.

-Short trips (under 2 miles) are the most common category.

Complete visual representation of pipeline :



# 10. Cost & Use Case Comparison

Copy Activity:  
- Low-cost, optimized for ingestion and minimal transformation  
- Best for bulk data transfer  
  
Mapping Data Flow:  
- More expensive, suitable for complex transformations  
- Best for filtering, aggregation, joins, and schema drift handling  
  
Recommendation: Use Copy Activity for raw ingestion and Mapping Data Flow for transformations.

# Aws Equivalent

If we Perform the Same Task in AWS, these are all the things which we will use in AWS

platform:

- Storage: S3

- Ingestion: AWS Glue, Lambda, or Data Pipeline

- Transformation: Glue ETL or EMR Spark

- Analysis: Databricks on AWS or EMR

- Triggering: Event Bridge, Glue triggers, Step Function