**ETL Pipeline Implementation**

**1. Introduction**

This document outlines the step-by-step implementation of an ETL (Extract, Transform, Load) pipeline in Azure Data Factory (ADF) to transfer data from a CSV file stored in Azure Blob Storage to an Azure SQL Database. The pipeline automates data ingestion while applying necessary transformations.

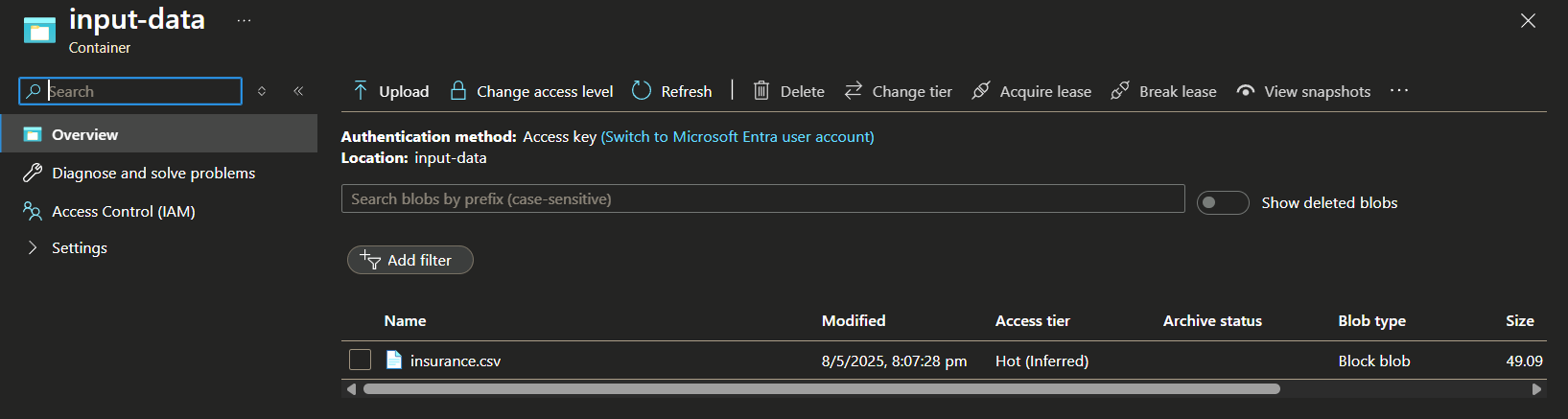
**2. Objectives**

* Extract data from a CSV file in Azure Blob Storage.
* Apply transformations .
* Load processed data into an Azure SQL Database table.
* Schedule the pipeline for automated daily execution.

**3. Implementation Steps**

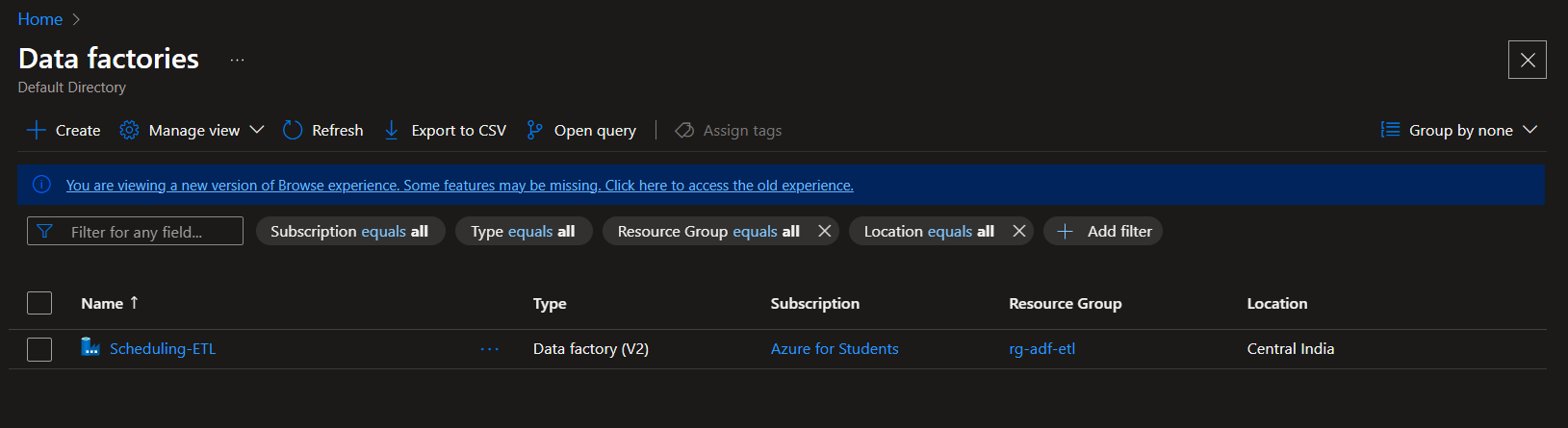
**3.1. Uploading CSV to Azure Blob Storage**

* Created a Storage Account in Azure.
* Uploaded the CSV file to a container named input-data.



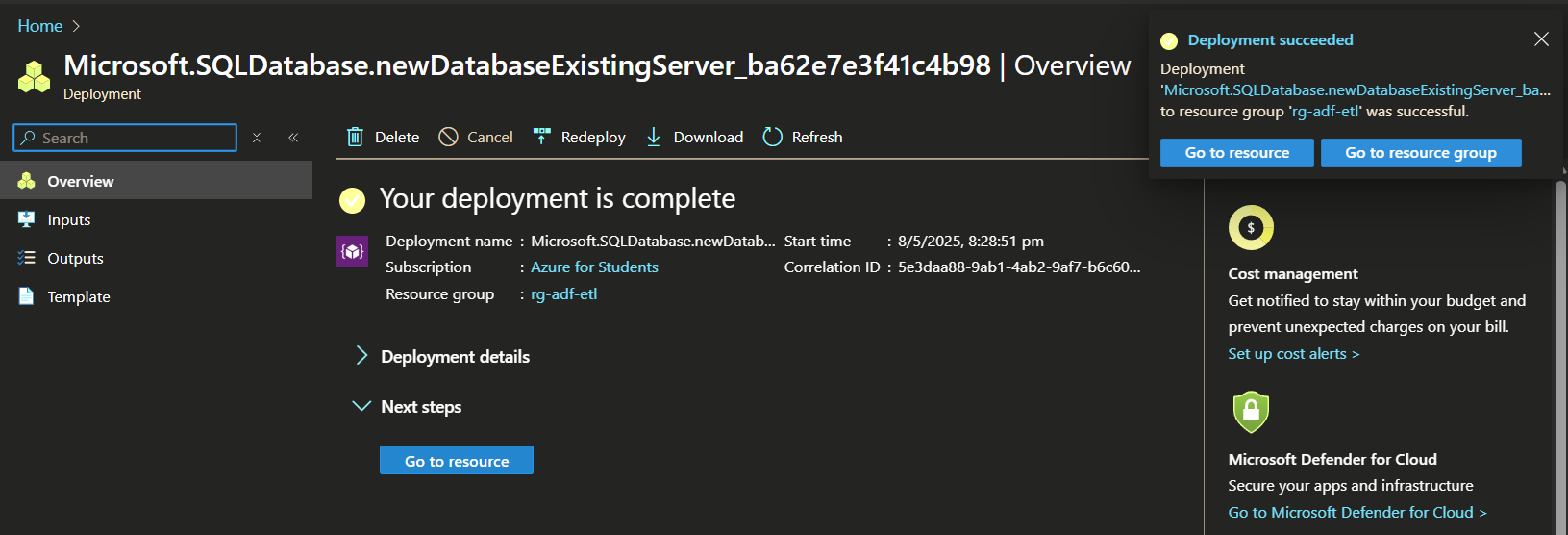
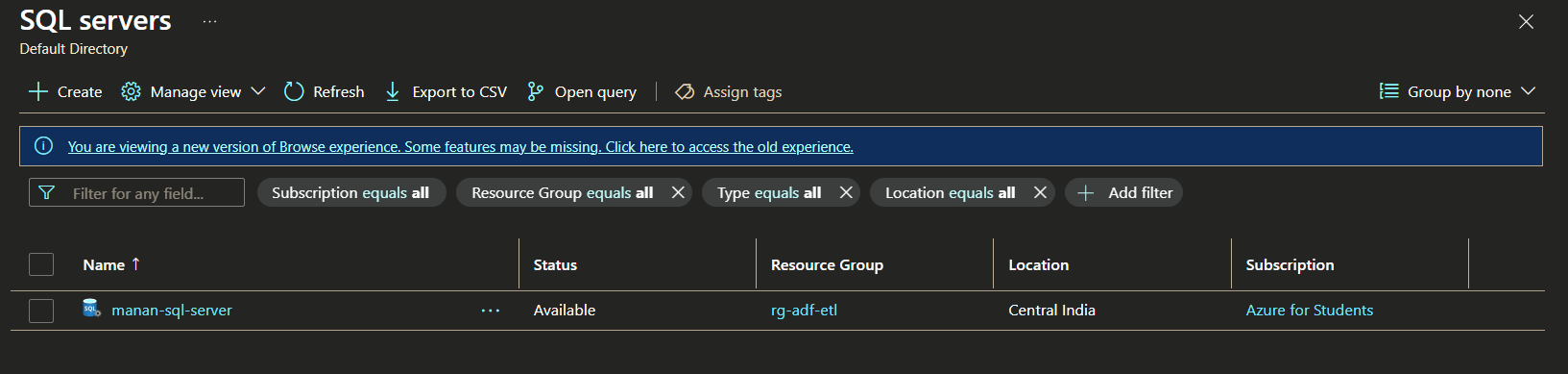
**3.2. Setting Up Azure Data Factory**

* Created an ADF instance (Scheduling\_ETL).
* Configured the region and resource group (rg-adf-etl).

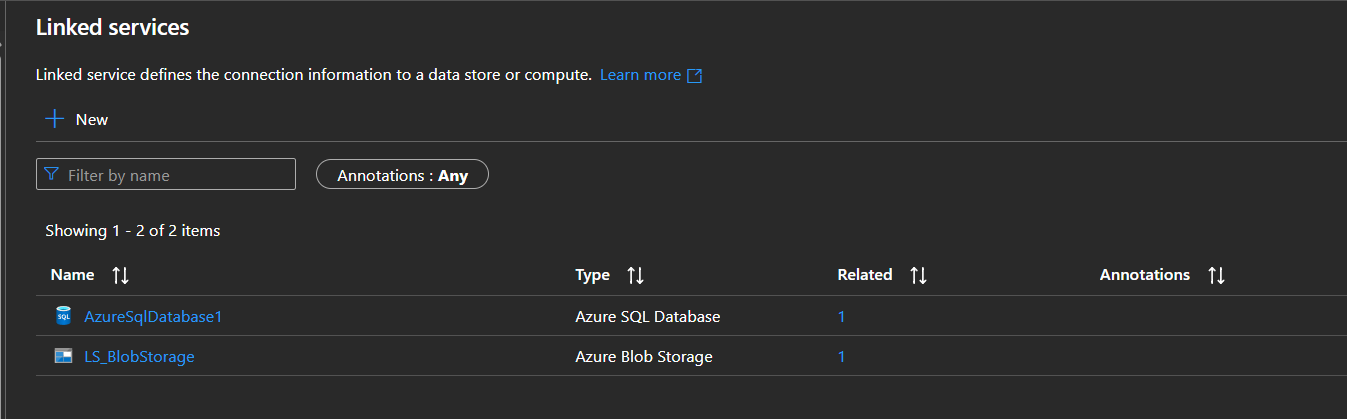


**3.3. Creating Linked Services**

* Blob Storage Linked Service (LS\_BlobStorage)
  + Connected to the Blob Storage containing the CSV file.
  + Authentication: Storage Account Key.
* SQL Database Linked Service (LS\_SQLDB)
  + Created a sql-server and a database to make this linked service.

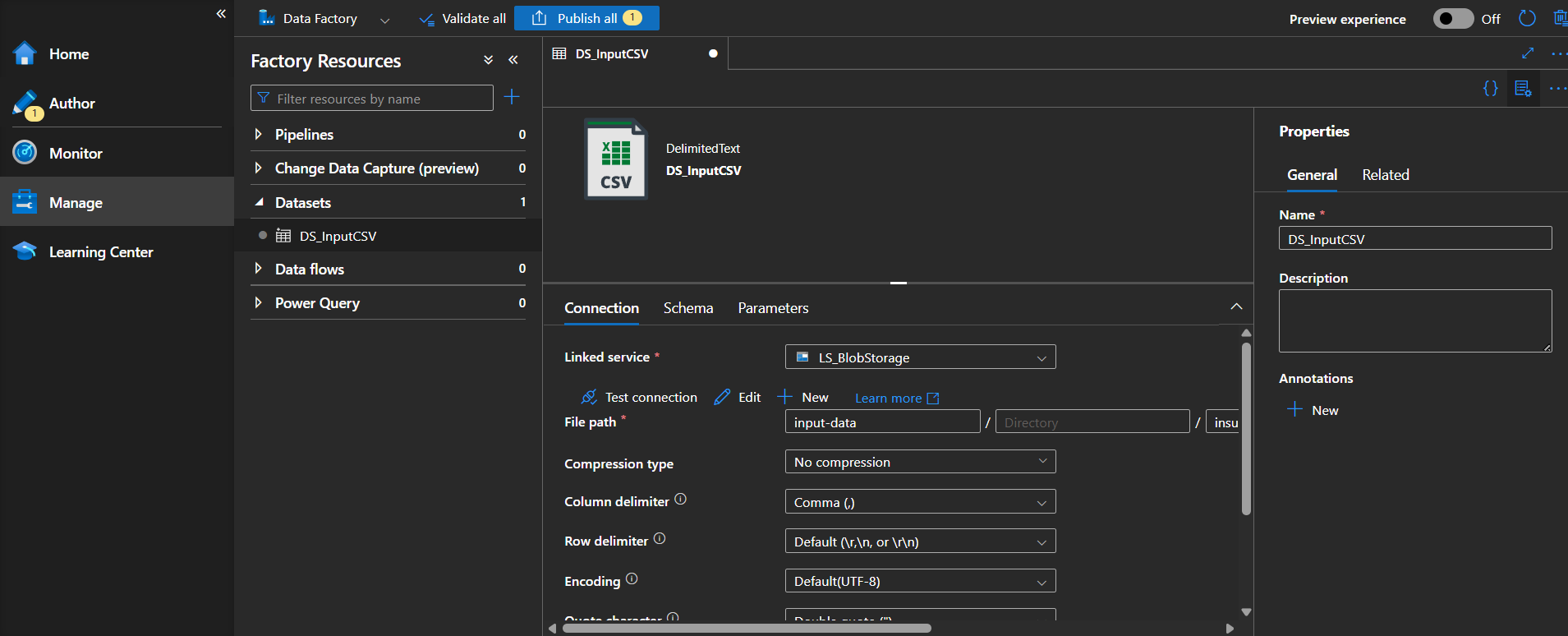


* + Configured connection to the destination SQL database.
  + Authentication: SQL credentials.

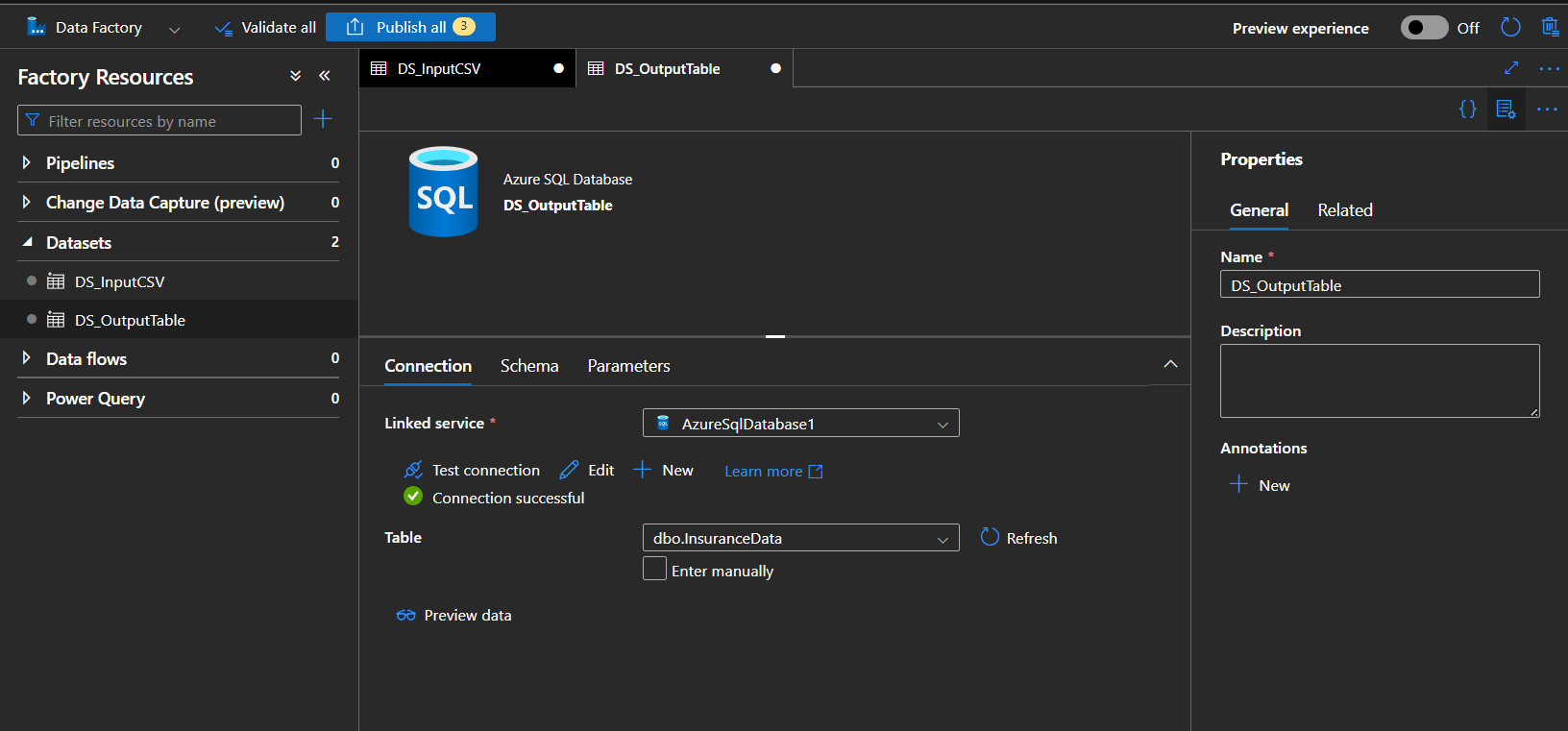
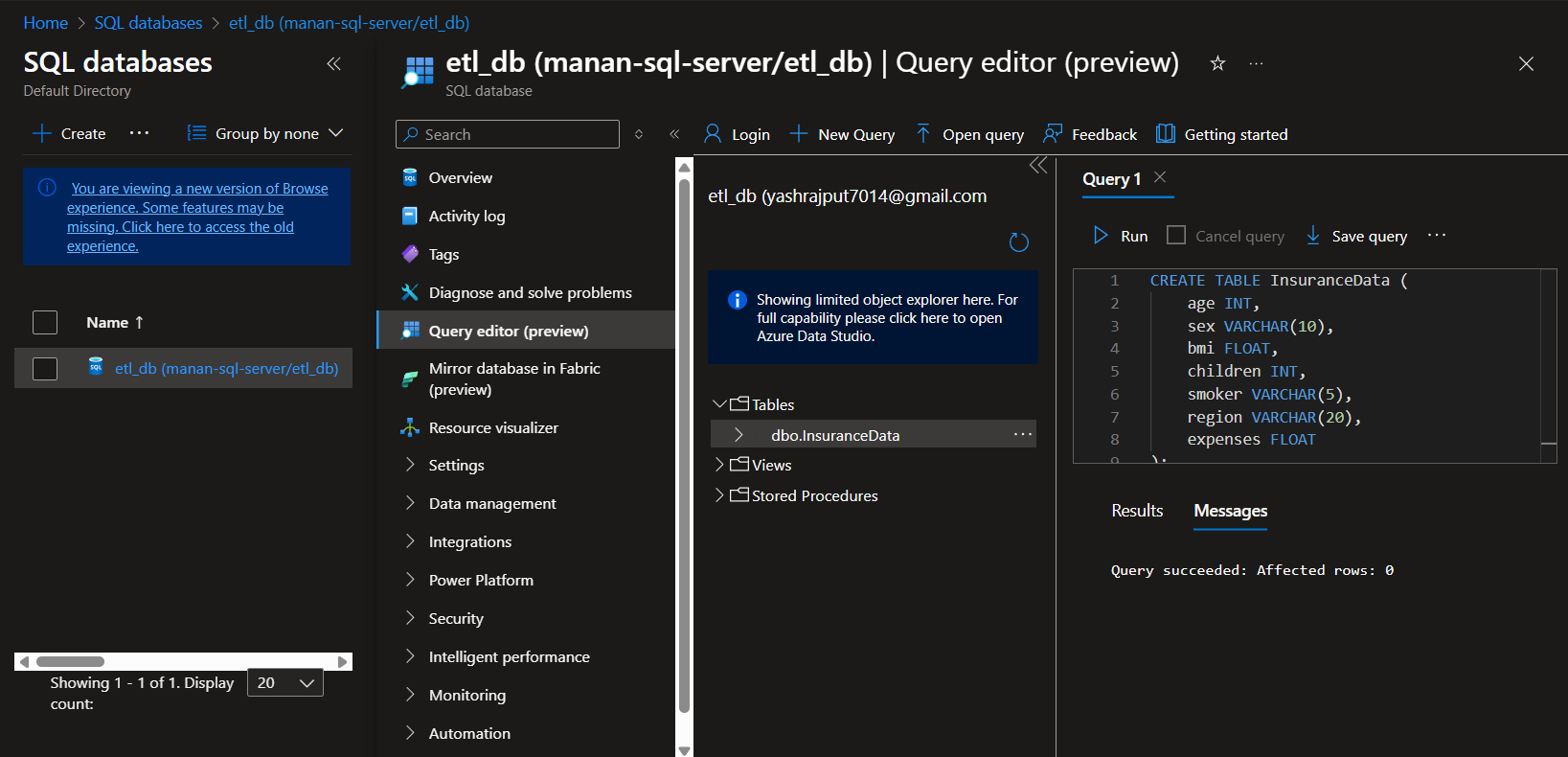


**3.4. Configuring Datasets**

* Source Dataset (DS\_InputCSV)
  + Format: CSV (First row as header enabled).
  + Schema imported automatically.

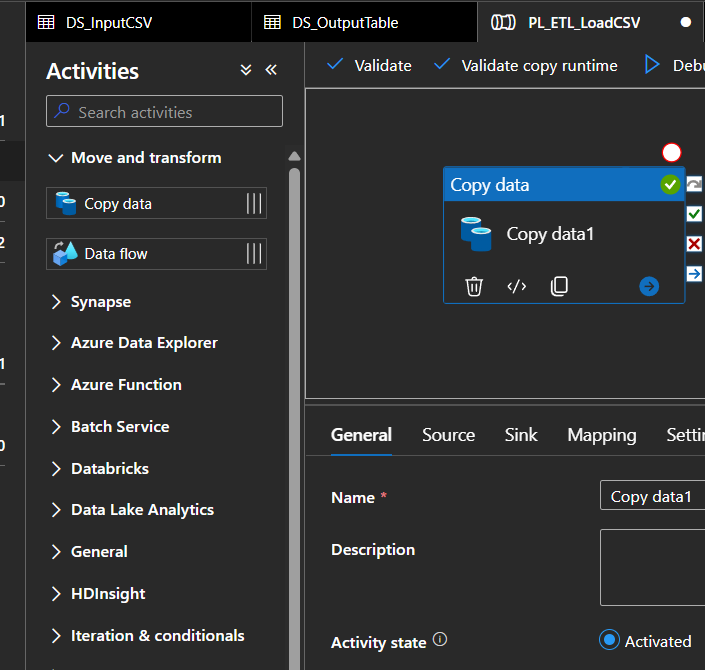


* Sink Dataset (DS\_OutputTable)
  + Target: Azure SQL Database table (dbo.InsuranceData).



**3.5. Building the Pipeline (**PL\_ETL\_LoadCSV**)**

* Used a Copy Data activity to transfer data.
* Source: DS\_InputCSV (Blob Storage).
* Sink: DS\_OutputTable (SQL Database).
* Mapping:
  + Auto-mapped columns where names matched.
  + Manually adjusted mismatched columns.

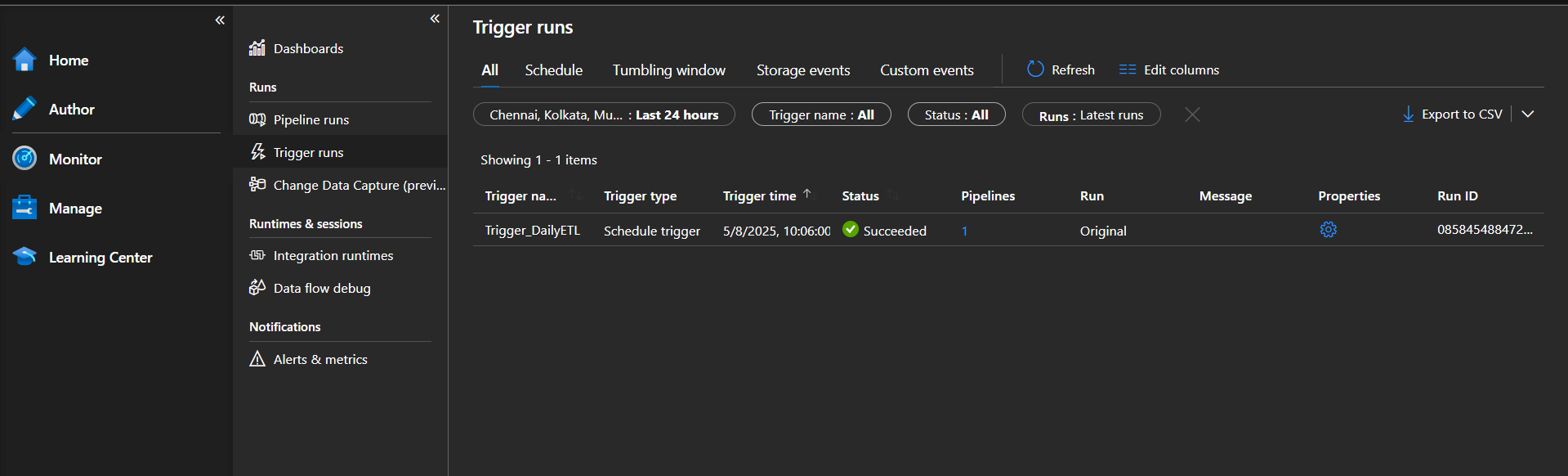
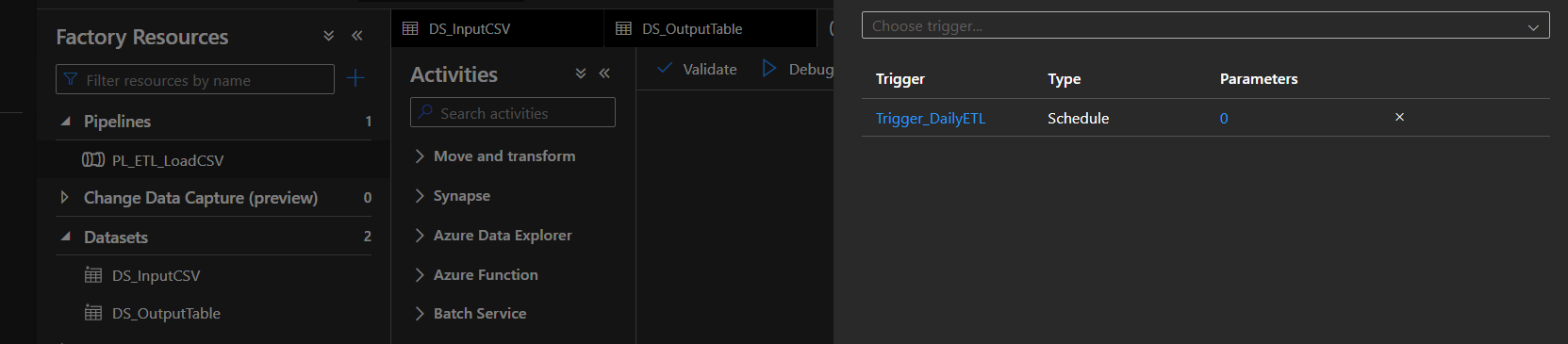


**3.6. Debugging & Publishing**

* Validated the pipeline to check for errors.
* Tested using Debug mode (verified successful data transfer).
* Published the pipeline for production.

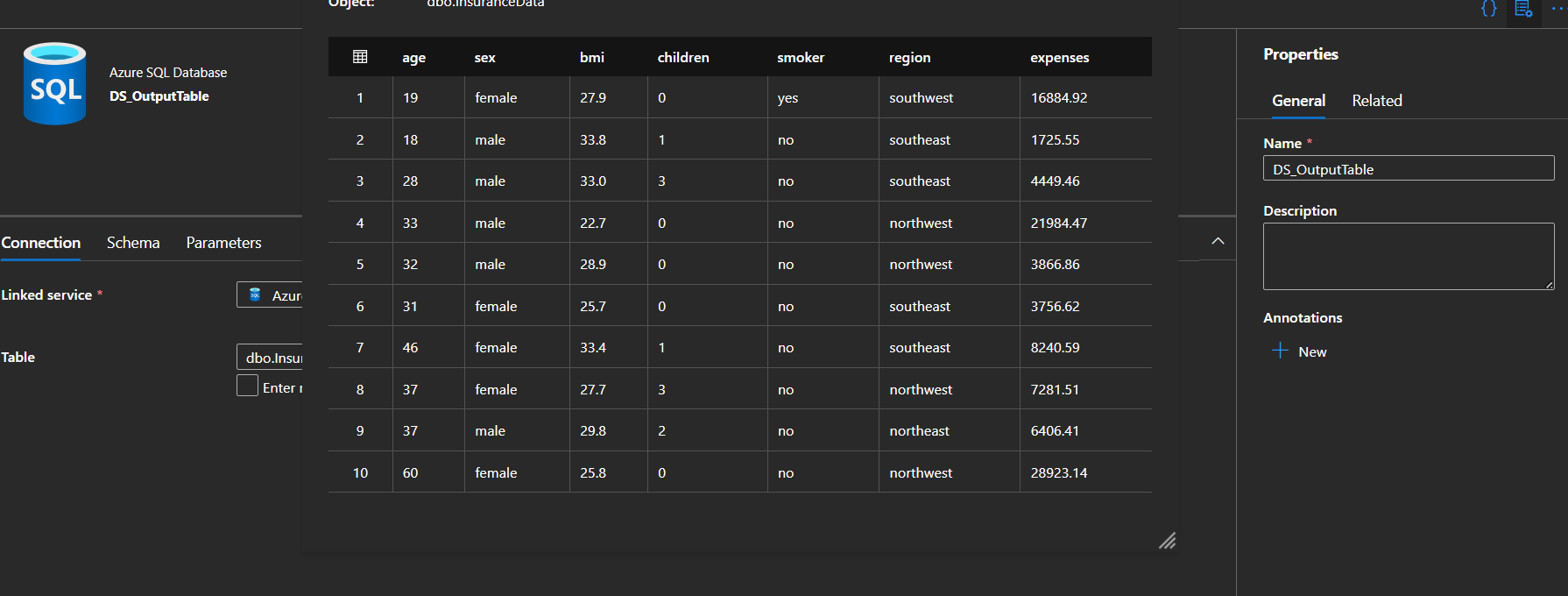
**3.7. Scheduling the Pipeline**

* Created a trigger (Trigger\_DailyETL) for daily execution.
* Set start time and recurrence (every 1 minute).
* Re-published to activate the schedule.



**4. Results & Validation**

* Data successfully transferred from Blob Storage to SQL Database.
* Transformations applied were verified.
* Scheduled job runs as expected without failures.



**5. Improvements**

* The data was not that complex so there were no great transformations so I could choose more complex data to practise more transformations.