Scope of Work (SoW) for Transportation & Logistics Data Processing & Analysis Project Documentation

Aftab Tamboli (INT-06)

Biz-Metric India Pvt Ltd

2025

Author(s)
Aftab Tamboli

Degree
Master's Degree of Statistics

Report/Thesis Title
Scope of Work (SoW) for Transportation & Logistics Data Processing & Analysis

Number of pages and appendix pages
29 + 2

With the growing demand for data-driven decision-making in the transportation and logistics sector, organizations are turning toward modern, scalable data solutions. This project focuses on the development of an end-to-end data platform using the Medallion Architecture (Bronze, Silver, and Gold layers) for a logistics company, aiming to streamline operations such as route optimization, fleet performance monitoring, and delivery tracking. The goal is to automate the data pipeline using Azure Data Factory, Azure Data Lake Gen2, and Databricks for efficient data ingestion, transformation, and storage, while using MySQL and Power BI to deliver actionable insights to business stakeholders. The initial sections provide theoretical foundations and introduce core architecture and tools. The implementation section details the data journey from raw source files to refined insights, highlighting processing steps, governance practices, and the use of Apache Spark and Delta Lake. The project concludes with a reflection on the results, evaluating the objectives achieved, and discussing both the strengths and potential improvements of the implemented solution.

Key words
Medallion Architecture, Data Engineering, Data Transformation, Business Intelligence, Microsoft Azure

Table of Contents

1. Introduction

This project focuses on processing and analyzing transportation and logistics data using PySpark, MySQL, and Power BI, following the Medallion Architecture. The goal is to ingest, clean, transform, and aggregate data related to transportation routes, vehicle performance, delivery times, and shipment statuses to generate meaningful business insights.

The final output is a Power BI dashboard displaying key metrics like route optimization, delivery efficiency, and fleet performance. The entire project was implemented on the Microsoft Azure platform using Medallion Architecture for data processing and automating the complete data pipeline from raw data ingestion to the final Gold layer.

A key feature of this implementation is complete automation - a customer or data engineer only needs to add data into the raw files. Once the raw file is updated, the pipeline is triggered automatically, and insights are displayed on the Power BI dashboard without any manual intervention.

2. Software & Tools

The project utilizes Microsoft Azure to set up the following tools:

- **Azure Blob Storage (Storage Account)**: Storing raw data
- **Azure Data Lake Storage Gen2 (Storage Account)**: Containing medallion architecture
- **PySpark**: Data processing & transformation
- **Azure Databricks Service (Jupyter Notebook)**: Running PySpark scripts interactively
- **Azure Database for MySQL Server**: Storing transformed data & final aggregated results
- **Azure Data Factory**: Automating the entire pipeline
- **Power BI**: Data visualization & reporting

3. Medallion Architecture Overview

The project follows the Medallion Architecture, which consists of three structured layers:

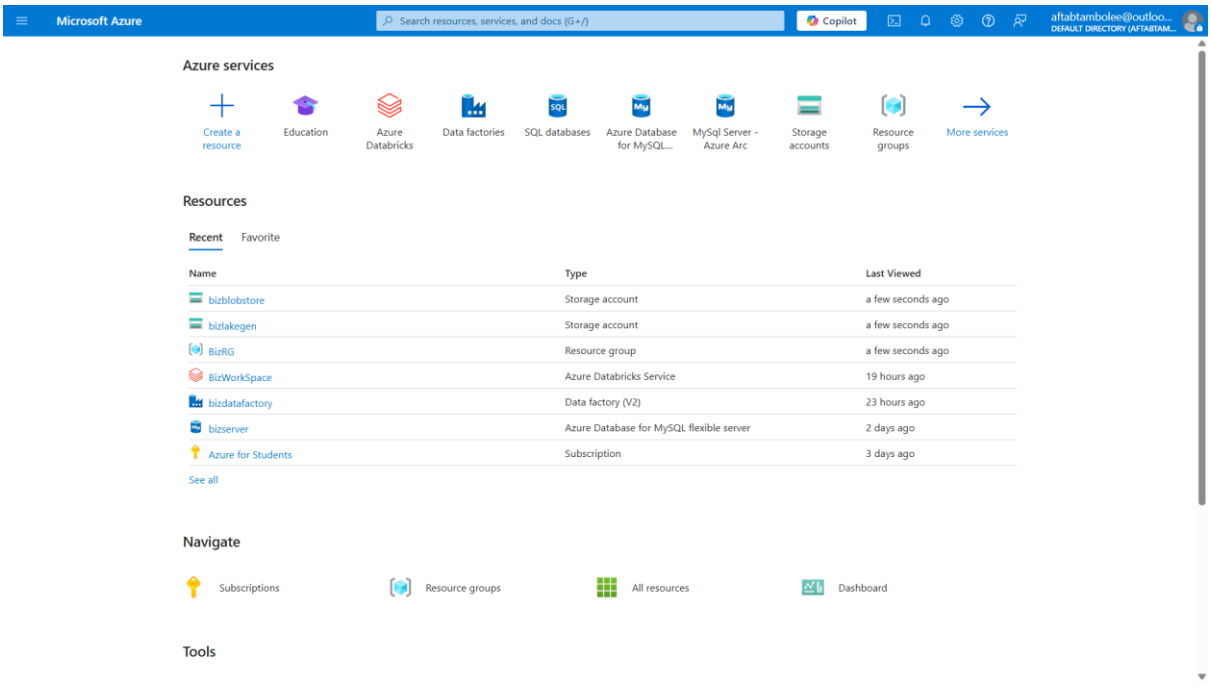
- **Bronze Layer**: Stores raw data with minimal processing (Parquet files)
- **Silver Layer**: Cleansed, transformed, and enriched data (Parquet & MySQL tables)
- **Gold Layer**: Aggregated data for reporting & dashboards (MySQL tables)

4. Project Implementation

4.1 Microsoft Azure Setup

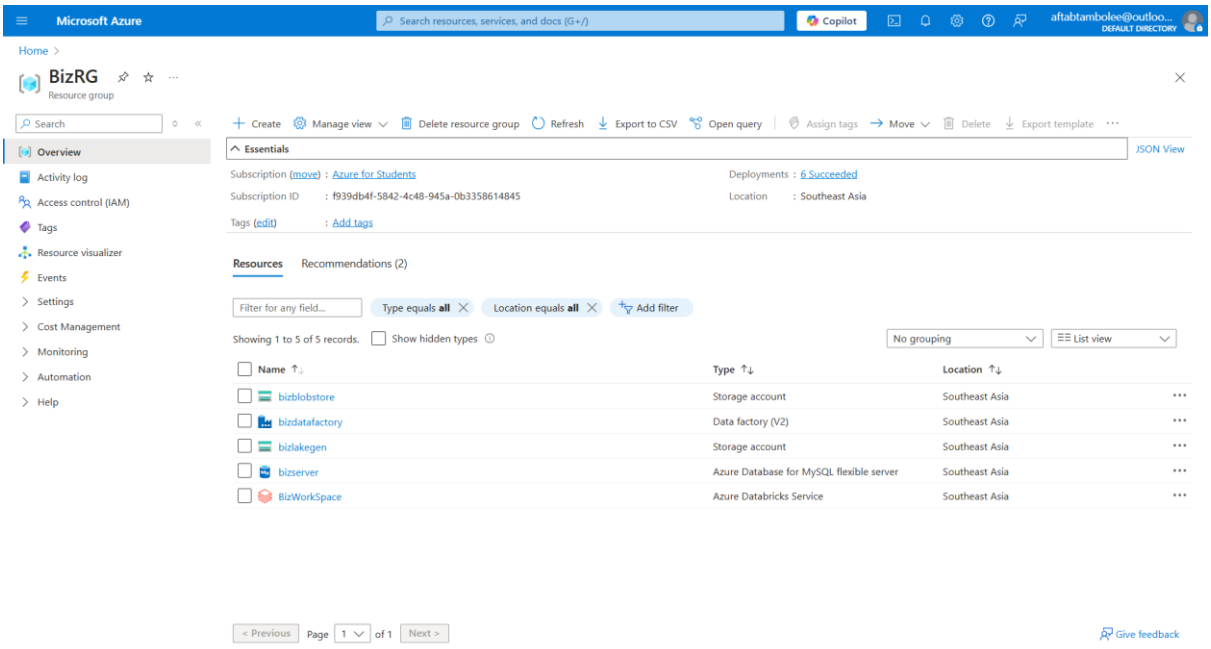
Microsoft Azure is a cloud computing platform used for building, deploying, and managing applications. It offers services like data storage, analytics, AI, networking, and security. Azure is flexible, scalable, and supports hybrid solutions, making it ideal for businesses and developers across various industries.

Here's how the home page of Microsoft Azure looks:



A resource group named "BizRG" contains all storage accounts, Data Factory, Database for MySQL Server, and Databricks Services. From here, you can directly access any tools, service, or software utilized for the project.

This is the dashboard for resource group - BizRG:

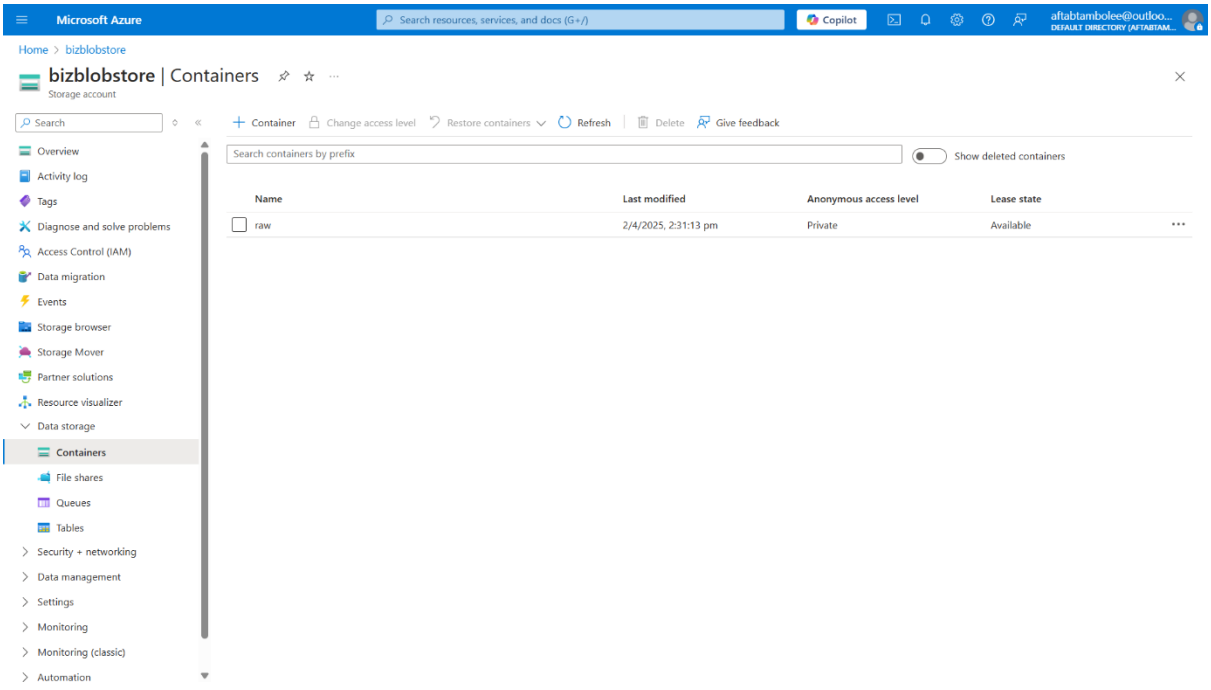


4.2 Storage Setup

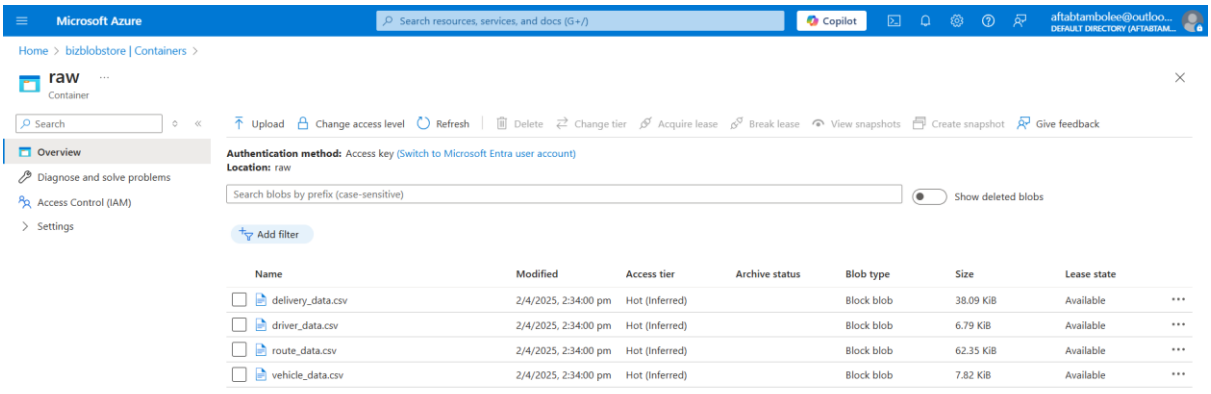
4.2.1 Azure Blob Storage

A storage account called "bizblobstore" was created as Azure Blob Storage. Within bizblobstore, a new container named "raw" was created to store the initial four raw data files provided by the company in CSV format. These raw files will be copied later and converted into parquet format for storage in the bronze layer.

This is the Azure blob storage – bizblobstore:

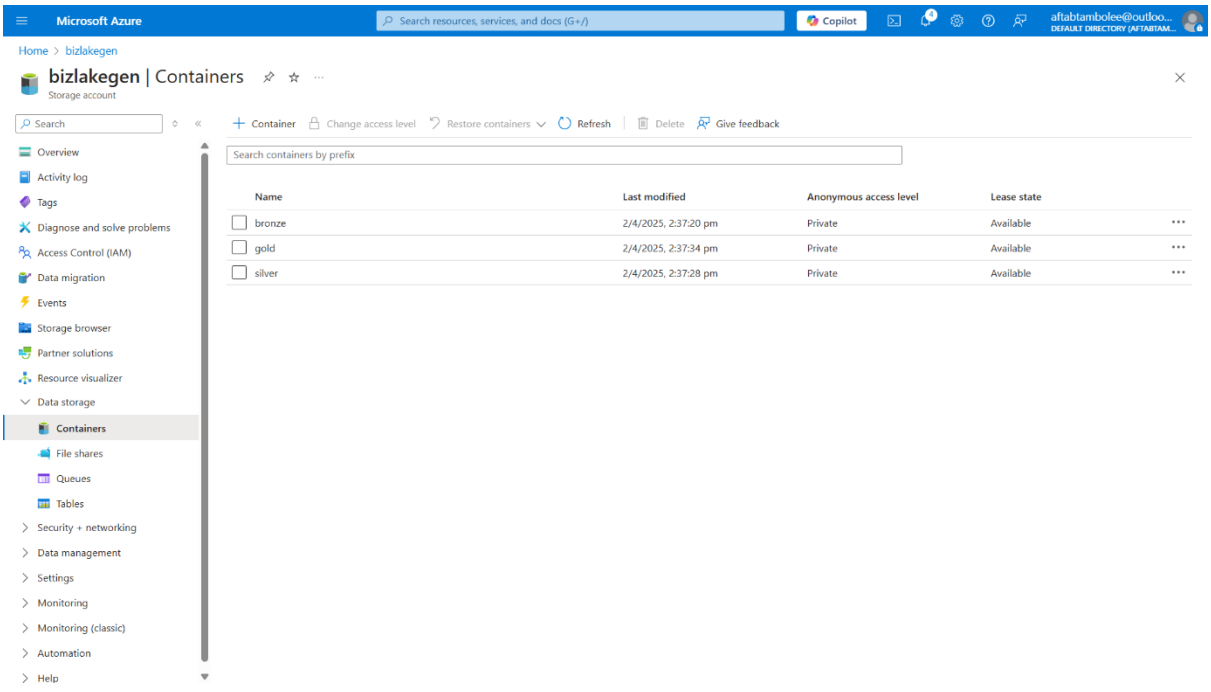


The raw files in the container are:

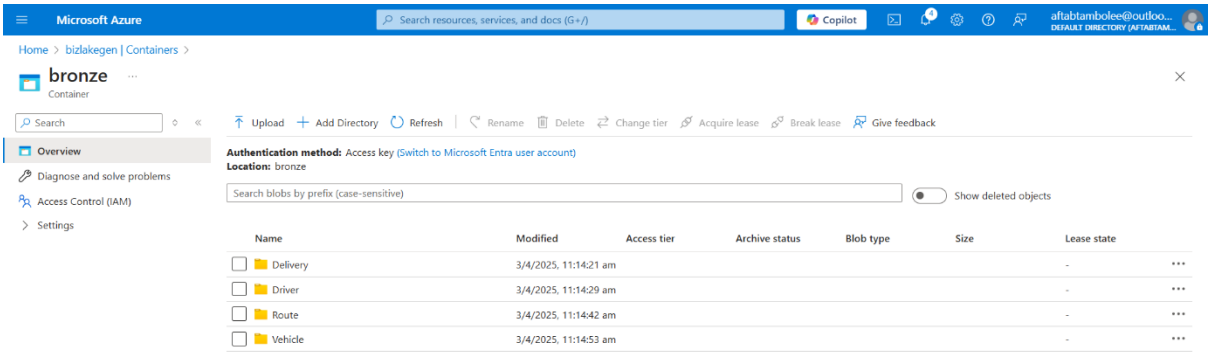


4.2.2 Azure Data Lake Storage Gen2

Another storage account named "bizlakegen" was created as Azure Data Lake Storage Gen2 for implementing the medallion architecture (bronze layer, silver layer, gold layer). As we progress, bizlakegen will be used to establish connections with the Azure Data Factory. In bizlakegen, three containers were created: bronze, silver, and gold.



In the bronze container, four directories were created to store the parquet format of each file and the generated logs:

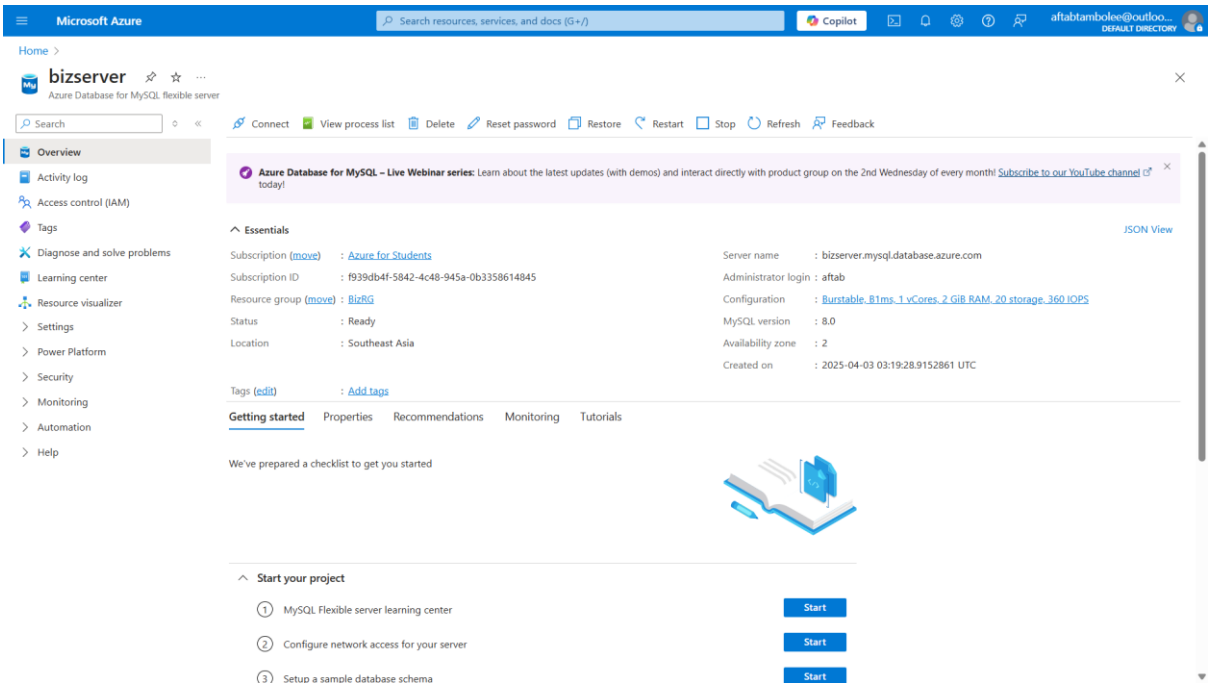


https://portal.azure.com/#

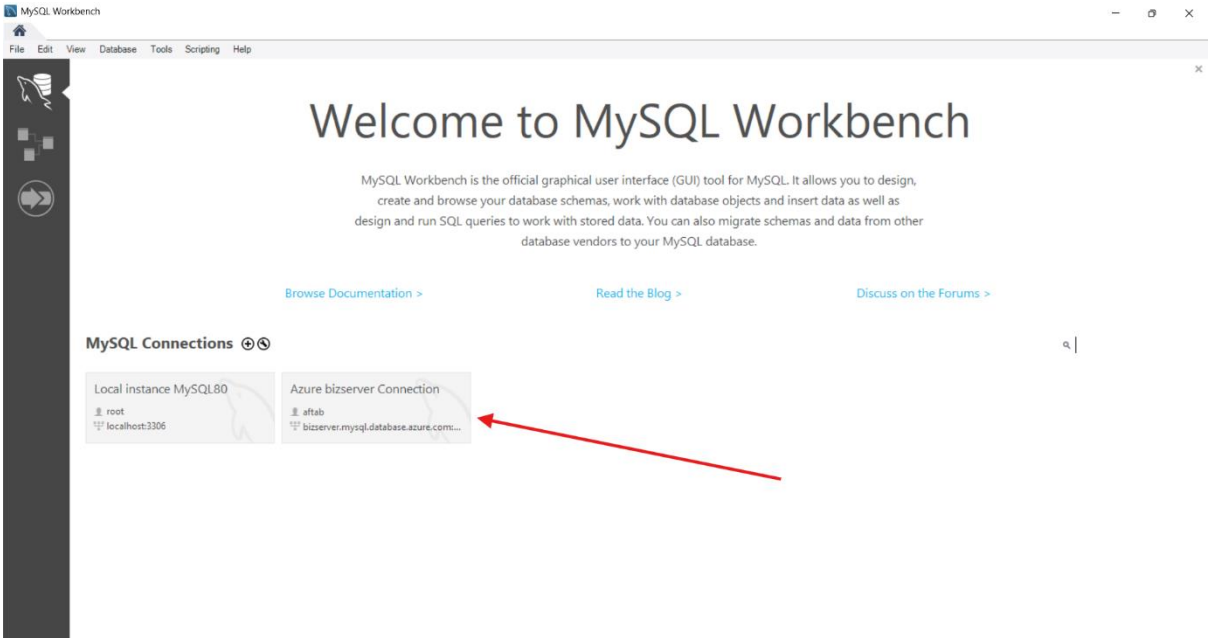
The silver and gold containers are initially empty as they will contain the processed data once the pipeline is triggered.

4.3 Database Setup

An Azure Database for MySQL Flexible Server was created for storing transformed data and final aggregated results:

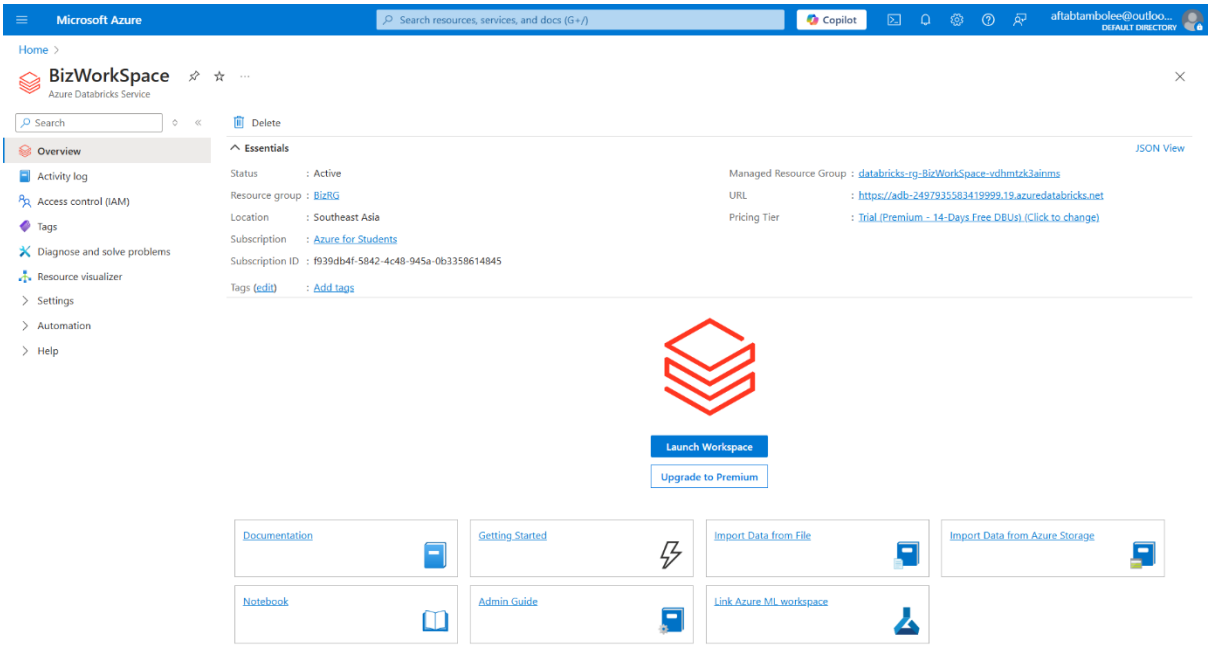


This was connected to MySQL Workbench (already installed on the local system) using server name, hostname, password, etc. This connection enables running SQL queries for the project.



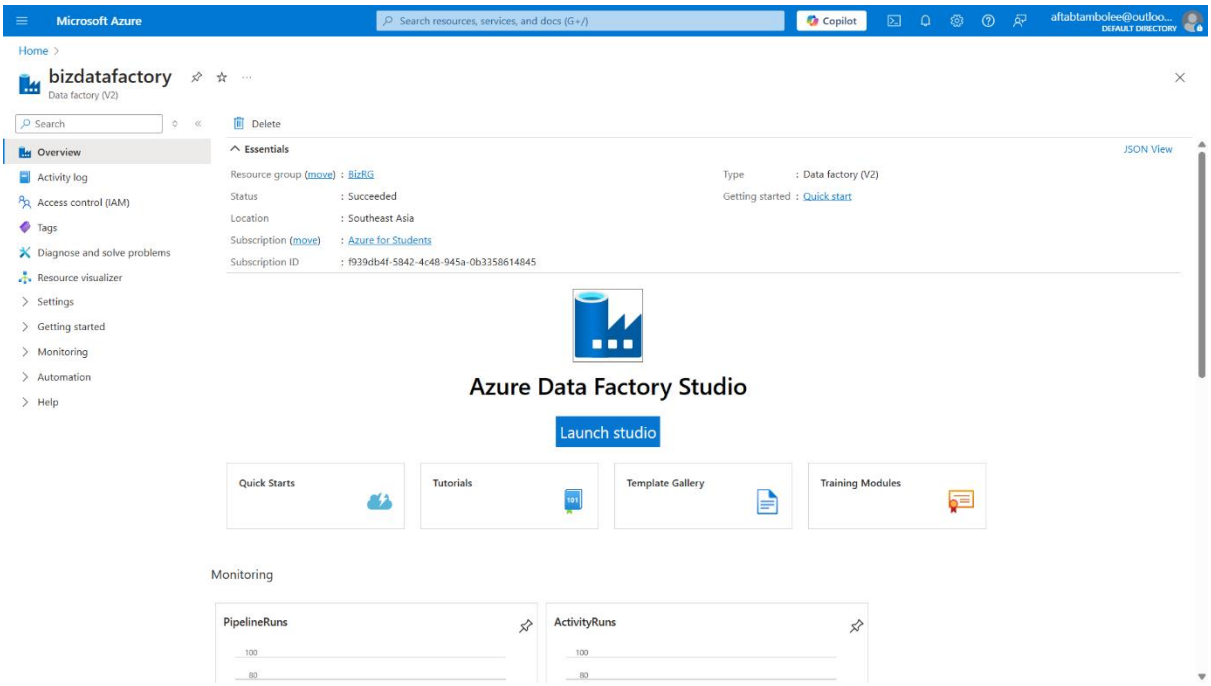
4.4 Azure Databricks Service

Azure Databricks Service named "BizWorkSpace" was set up to write PySpark commands/code in databricks notebooks for transformations, operations, or aggregations. Later, this will be connected to Azure Data Factory to automate the databricks notebook in the data pipeline.



4.5 Azure Data Factory

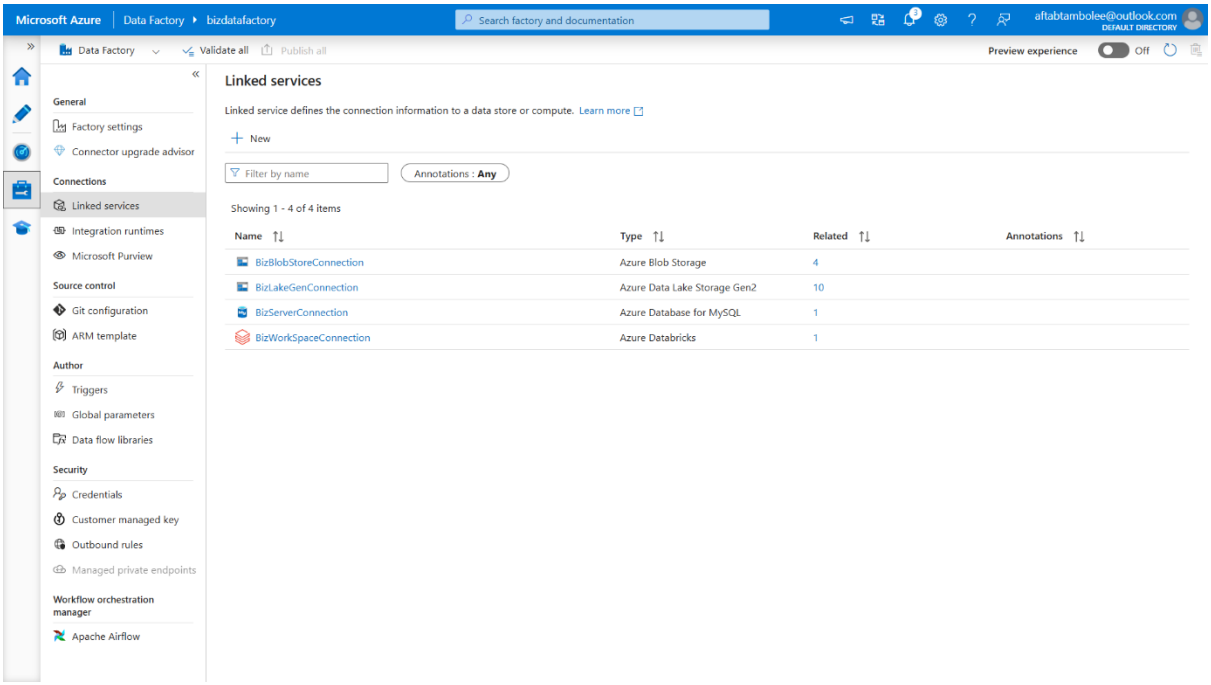
Finally, Azure Data Factory (ADF) named "bizdatafactory" was created to orchestrate and automate the movement and transformation of data across different layers of the project. This is where everything happens, from ingestion of raw data to preparing the final gold layer.



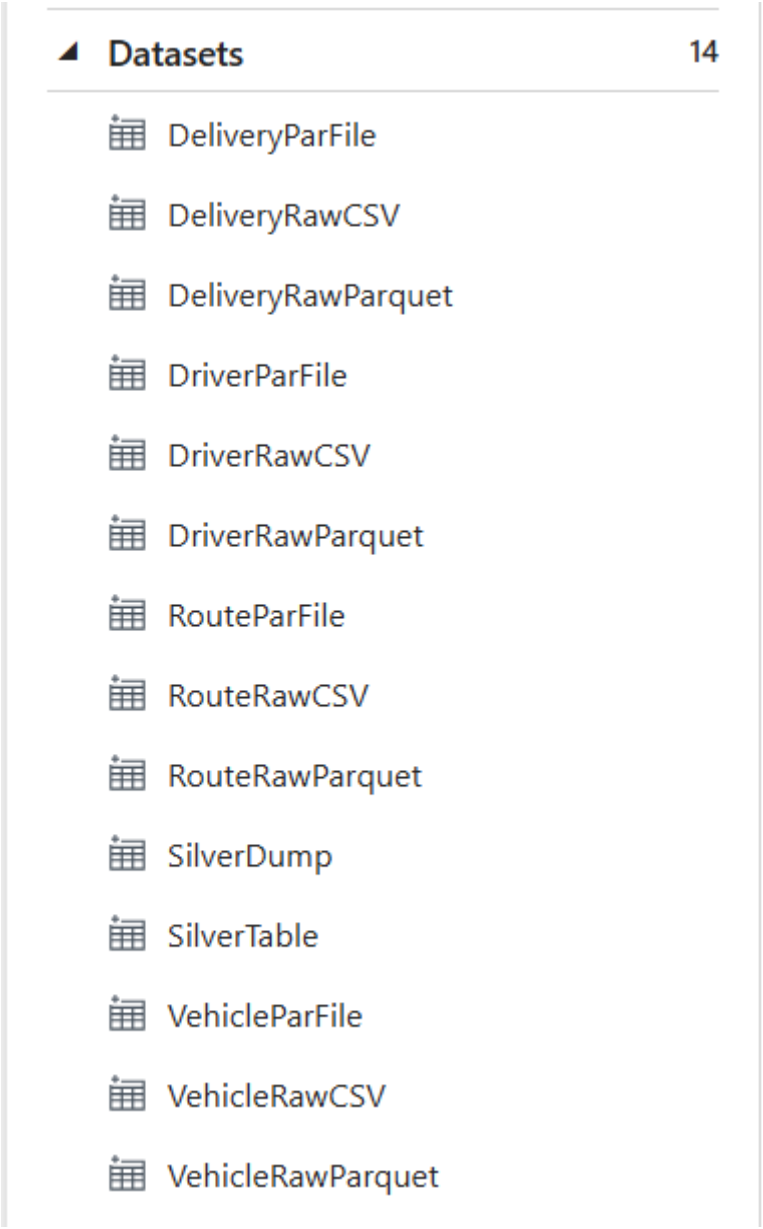
5. Pipeline Implementation

5.1 Connections and Datasets

Here are the connections made with Azure Data Factory:



Fourteen datasets were created in the data factory to establish connections with particular activities in the data pipeline:

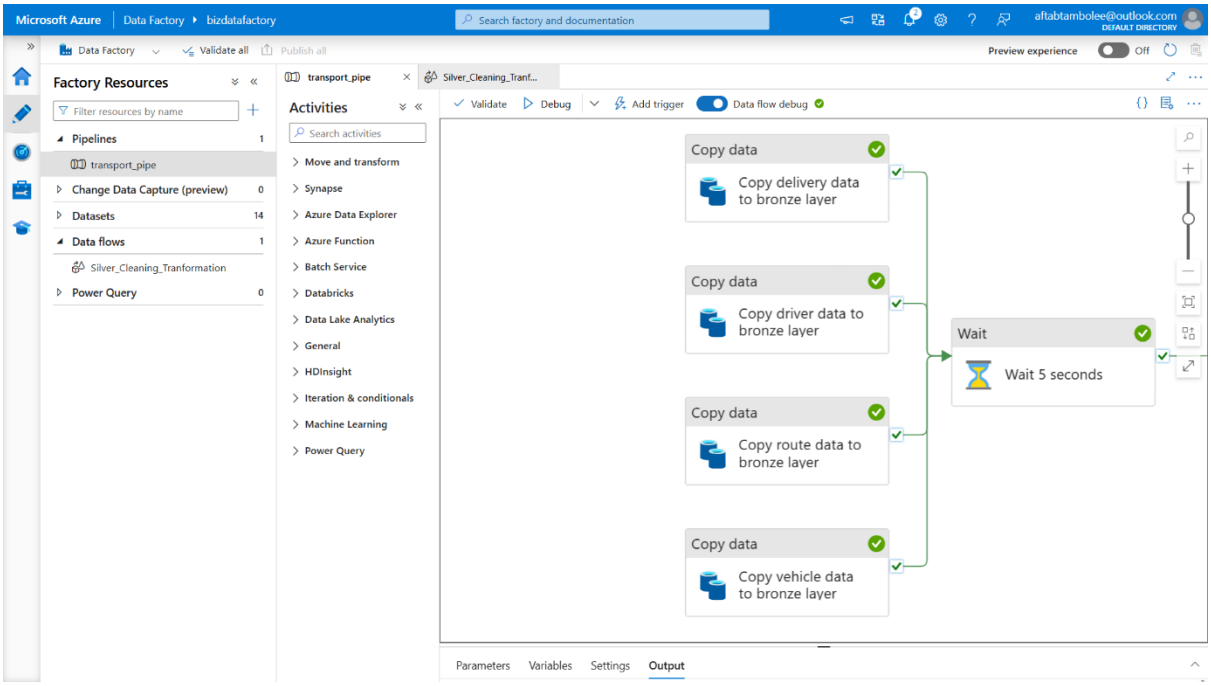


5.2 Bronze Layer Implementation

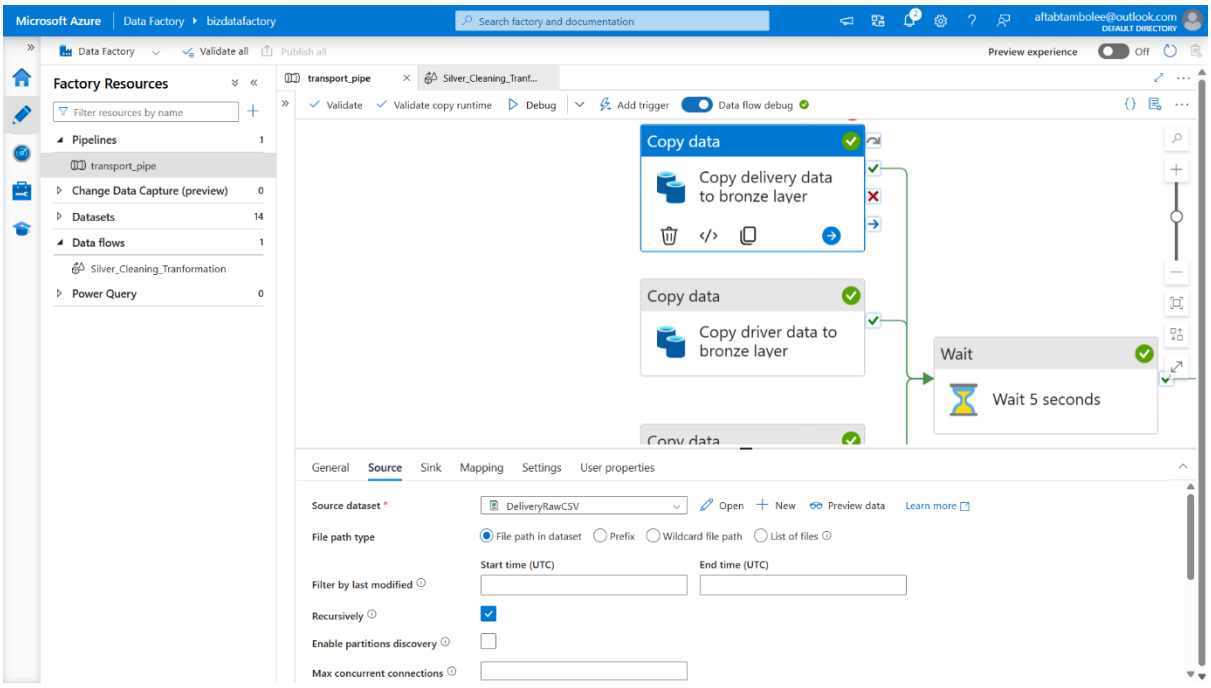
A new pipeline was created in bizdatafactory to construct the complete data flow. The pipeline begins with the ingestion of raw data and proceeds to preparing the gold layer.

First, copy activity was used in the pipeline to copy the raw CSV data files from the raw container in bizblobstore to the bronze container in bizlakegen. The data remains unchanged but is converted to parquet format.

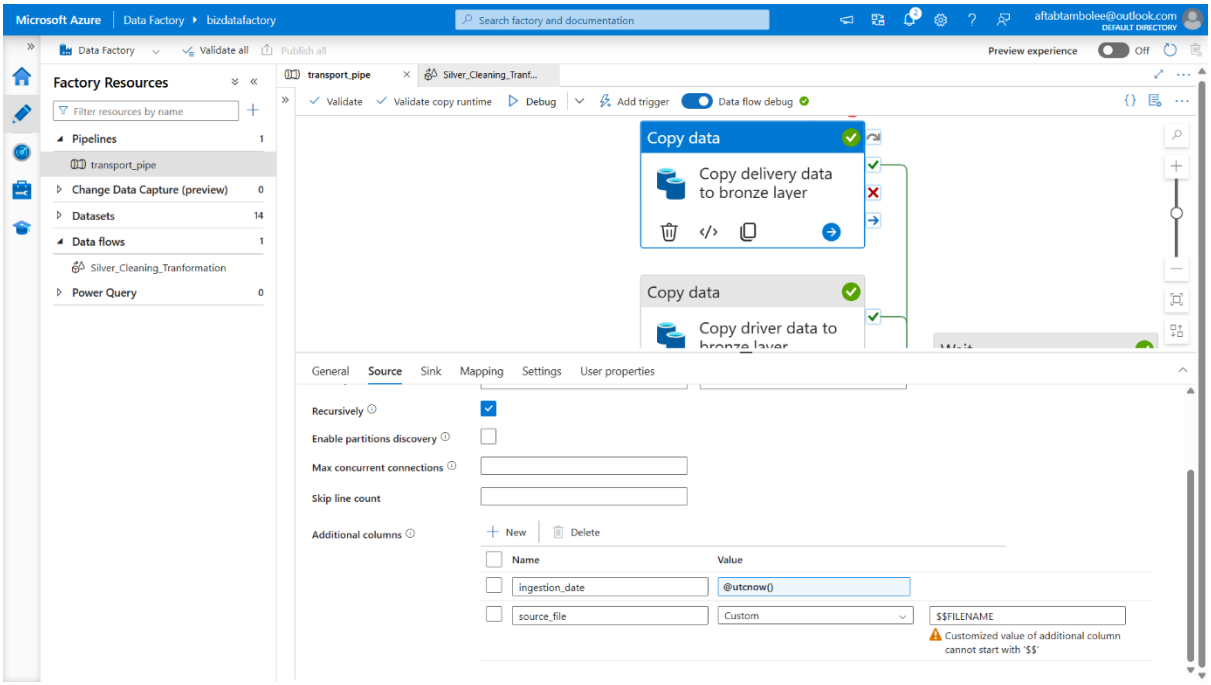
Since there are 4 files, 4 copy activities were used:



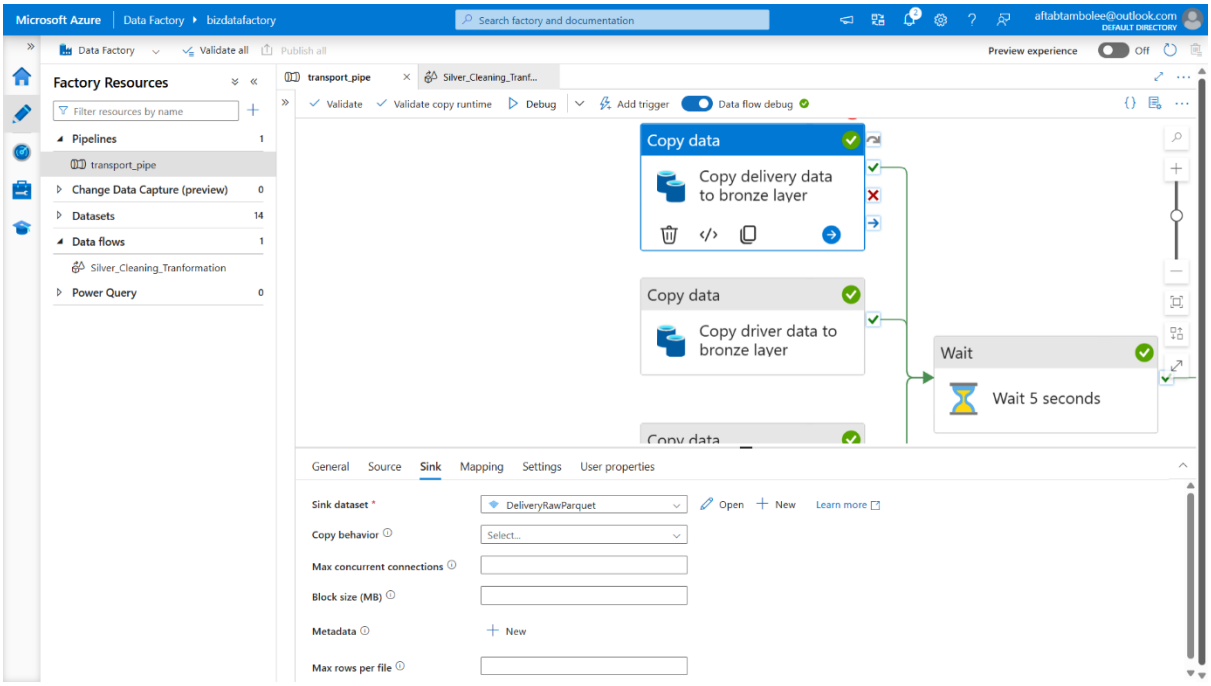
For each copy activity, the source was set as the raw CSV file from bizblobstore:



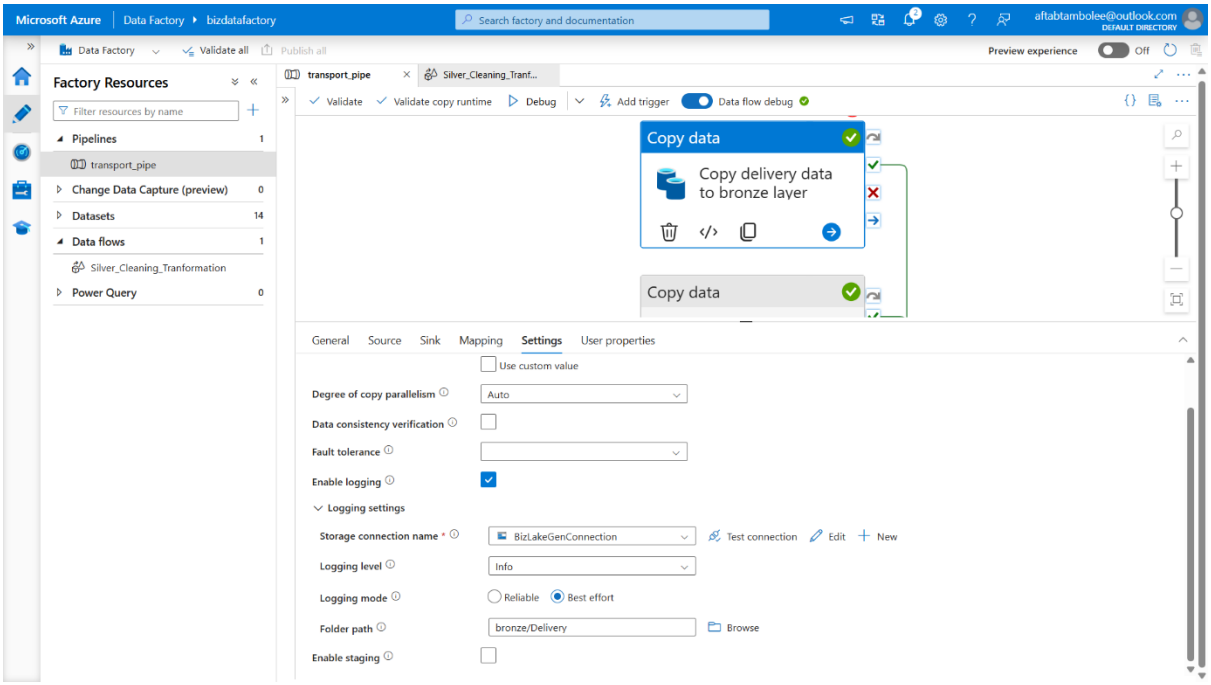
Two audit columns (ingestion_date, source_file) were added to each bronze table to track data lineage:



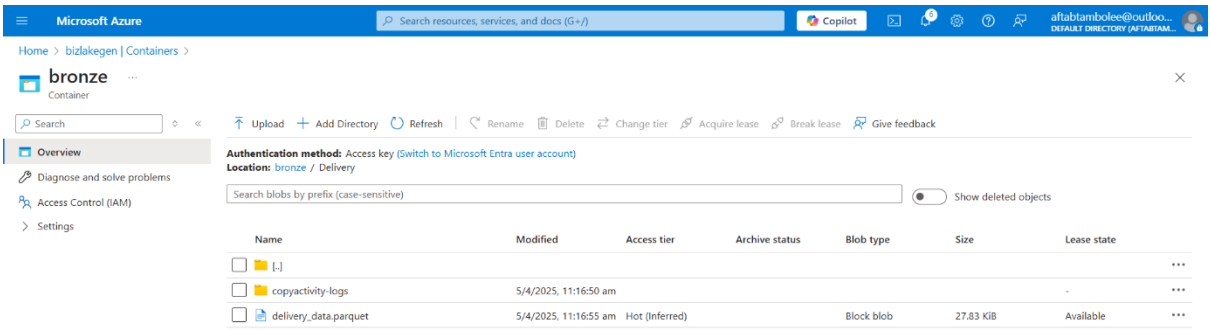
The sink was set to the appropriate directory in the bronze container to store each raw parquet file and their generated logs:



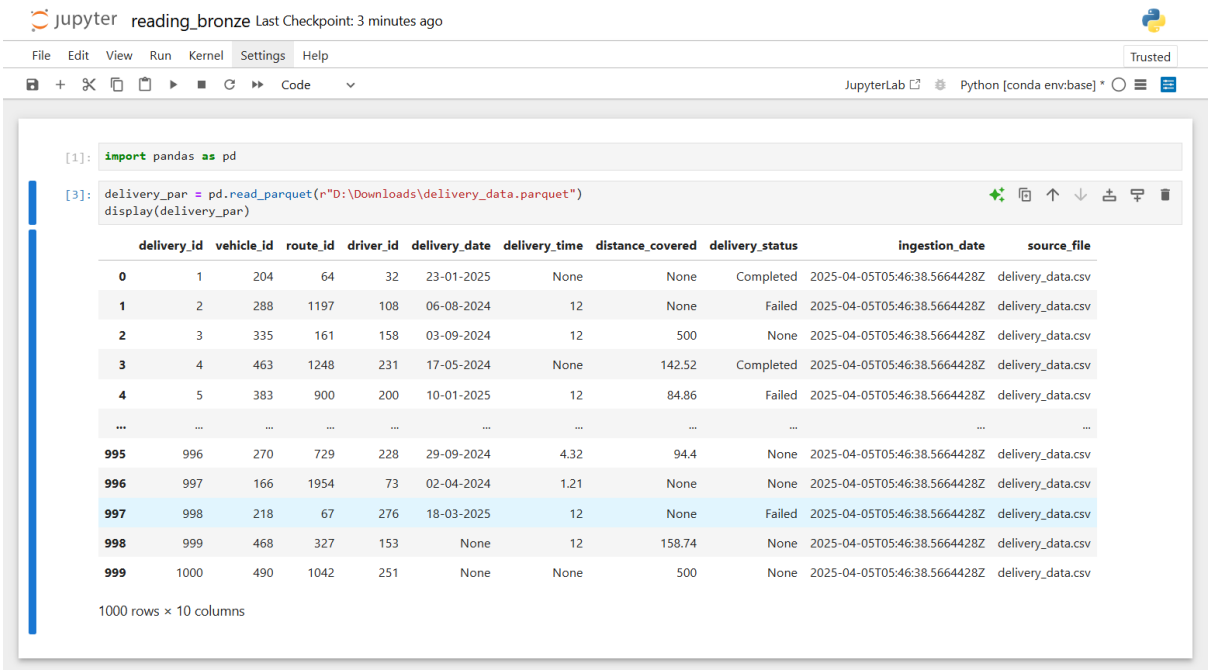
Logging was enabled at the INFO level in each copy activity, with the log folder path set to the respective file directory in the bronze container:



The parquet files were successfully generated in their respective directories along with their logs:



The raw data contains some null/empty values. The total raw data with 2 audit columns is shown below (example for delivery data; similar structure applies to driver, route, and vehicle data):



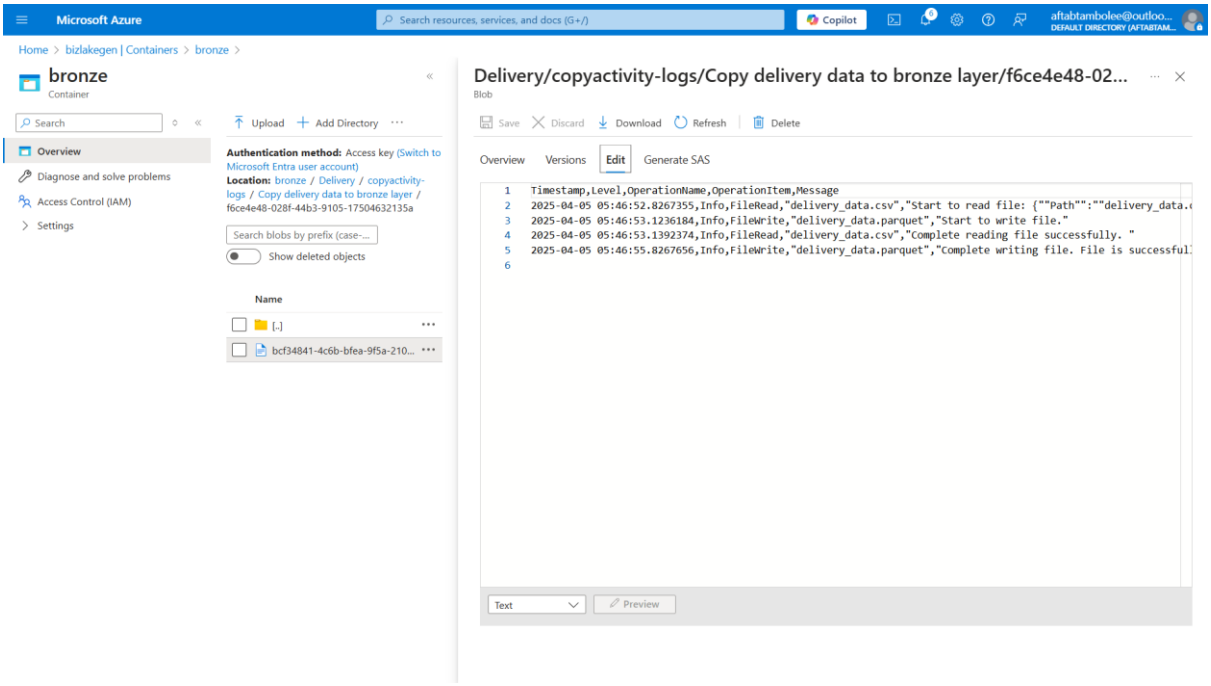
```
[1]: import pandas as pd

[3]: delivery_par = pd.read_parquet(r"D:\Downloads\delivery_data.parquet")
     display(delivery_par)
```

	delivery_id	vehicle_id	route_id	driver_id	delivery_date	delivery_time	distance_covered	delivery_status	ingestion_date	source_file
0	1	204	64	32	23-01-2025	None	None	Completed	2025-04-05T05:46:38.5664428Z	delivery_data.csv
1	2	288	1197	108	06-08-2024	12	None	Failed	2025-04-05T05:46:38.5664428Z	delivery_data.csv
2	3	335	161	158	03-09-2024	12	500	None	2025-04-05T05:46:38.5664428Z	delivery_data.csv
3	4	463	1248	231	17-05-2024	None	142.52	Completed	2025-04-05T05:46:38.5664428Z	delivery_data.csv
4	5	383	900	200	10-01-2025	12	84.86	Failed	2025-04-05T05:46:38.5664428Z	delivery_data.csv
...	...	...	...	...	...	...	...	...	...	...
995	996	270	729	228	29-09-2024	4.32	94.4	None	2025-04-05T05:46:38.5664428Z	delivery_data.csv
996	997	166	1954	73	02-04-2024	1.21	None	None	2025-04-05T05:46:38.5664428Z	delivery_data.csv
997	998	218	67	276	18-03-2025	12	None	Failed	2025-04-05T05:46:38.5664428Z	delivery_data.csv
998	999	468	327	153	None	12	158.74	None	2025-04-05T05:46:38.5664428Z	delivery_data.csv
999	1000	490	1042	251	None	None	500	None	2025-04-05T05:46:38.5664428Z	delivery_data.csv

1000 rows × 10 columns

Generated logs example:



```
Timestamp,Level,OperationName,OperationItem,Message
2025-04-05 05:46:52.8267355,Info,FileRead,"delivery_data.csv","Start to read file: {""Path"":""delivery_data.
2025-04-05 05:46:53.1236184,Info,FileWrite,"delivery_data.parquet","Start to write file."
2025-04-05 05:46:53.1392374,Info,FileRead,"delivery_data.csv","Complete reading file successfully. "
2025-04-05 05:46:55.8267656,Info,FileWrite,"delivery_data.parquet","Complete writing file. File is successful
```

At this point, the bronze layer is successfully completed with each parquet file and its logs:

- delivery_data_bronze.csv → delivery_data.parquet
- vehicle_data_bronze.csv → vehicle_data.parquet
- route_data_bronze.csv → route_data.parquet
- driver_data_bronze.csv → driver_data.parquet

All four copy activities were merged with a wait activity applying a 5-second delay to ensure all files have enough time to be transferred to their destinations before the pipeline automatically performs the silver layer transformation activities.

5.3 Silver Layer Implementation

For the silver layer preparation, the data was cleaned, structured, and enriched for analysis by performing:

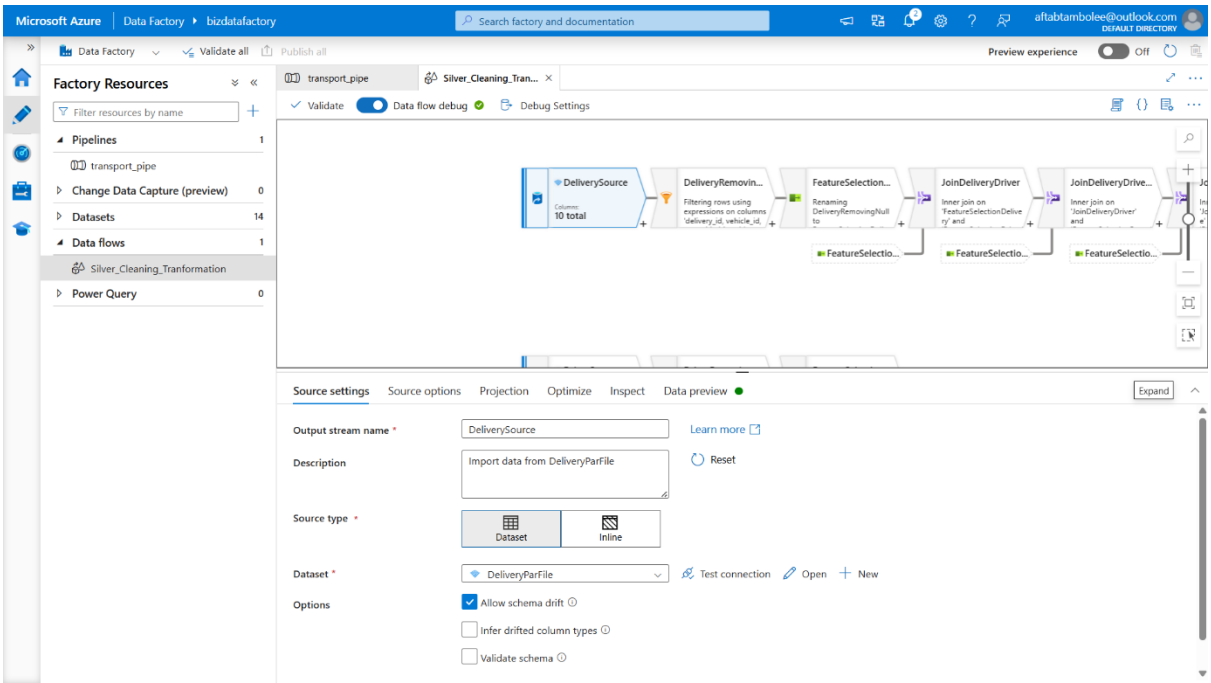
- **Data Cleaning**: Removing records with null values in critical columns
- **Joining Related Tables**: Adding vehicle details, driver details, and route descriptions
- **Computing Additional Fields**: e.g., fuel_consumed = distance_covered / fuel_efficiency
- **Storing Data** in both Parquet and MySQL for further aggregation

Dataflow in Azure Data Factory was used for this purpose, for several reasons:

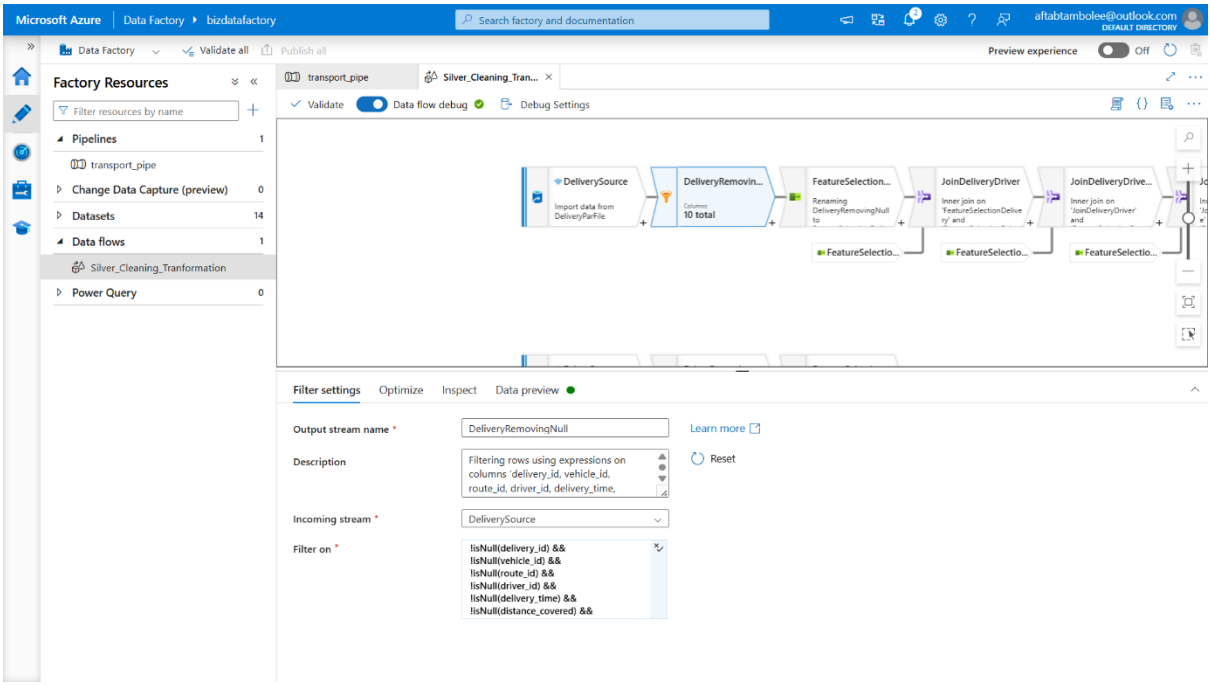
1. It provides freedom to generate parquet files with desired naming conventions, which is not possible in Databricks PySpark notebook
2. It offers easier transformation (cleaning, joining, computing) with graphical representation of the flow
3. It avoids the need to make connections with ADLS from Databricks

Dataflow implementation:

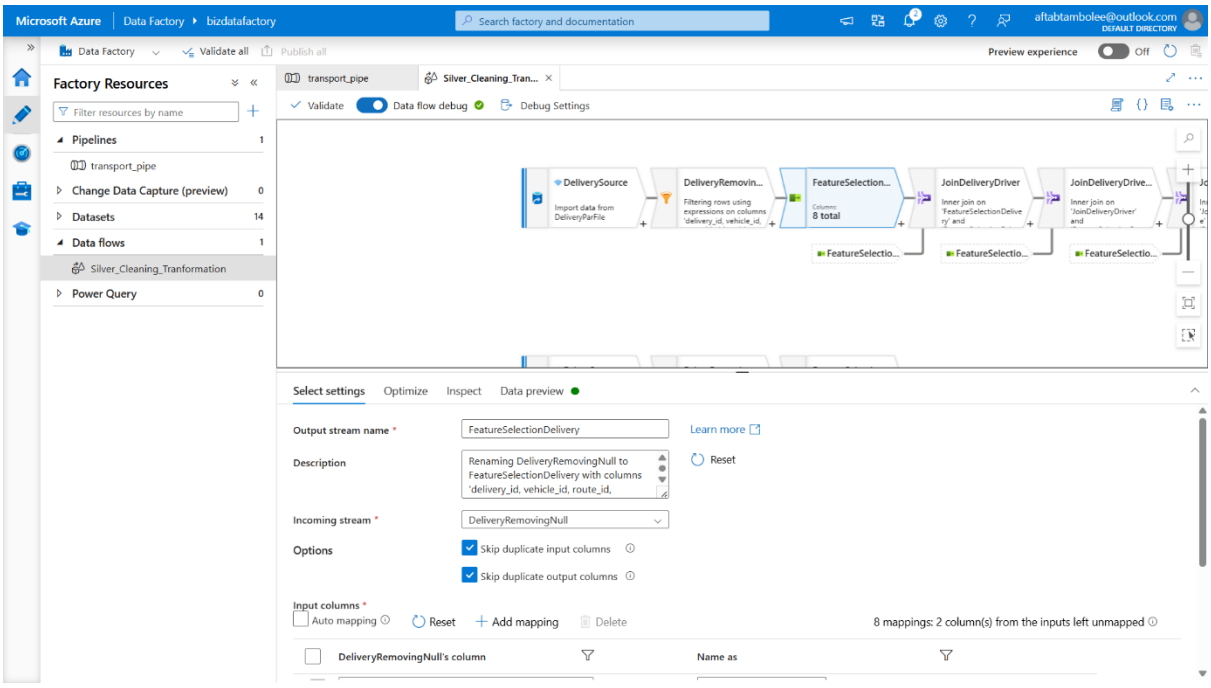
1. Four source activities were used to take the parquet datasets for each bronze file/table:



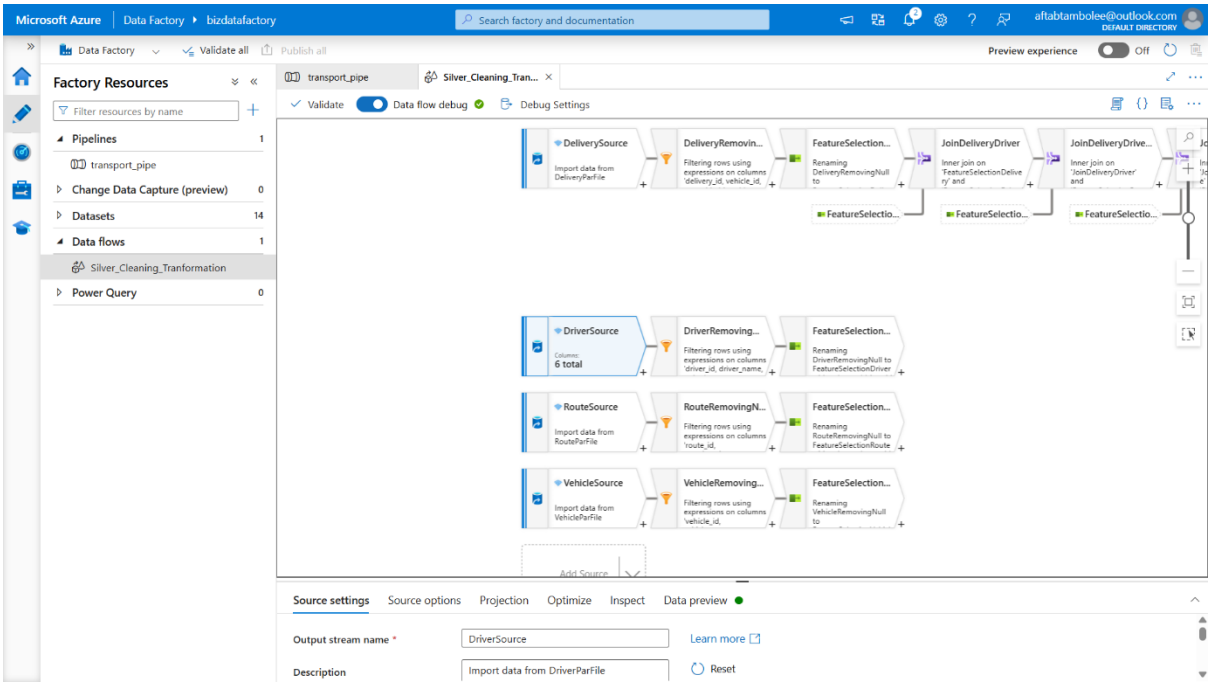
2. Filter activity was used to remove null values only from critical columns for each parquet dataset:



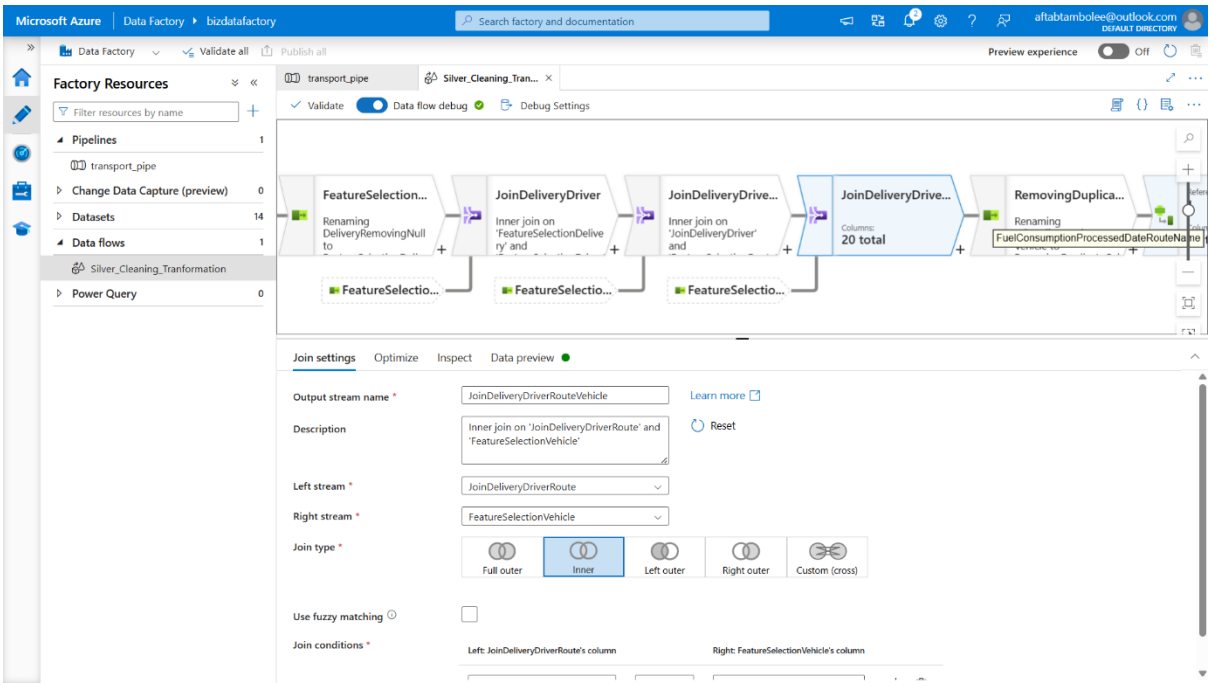
3. Select activity was used to remove unwanted columns (feature selection):



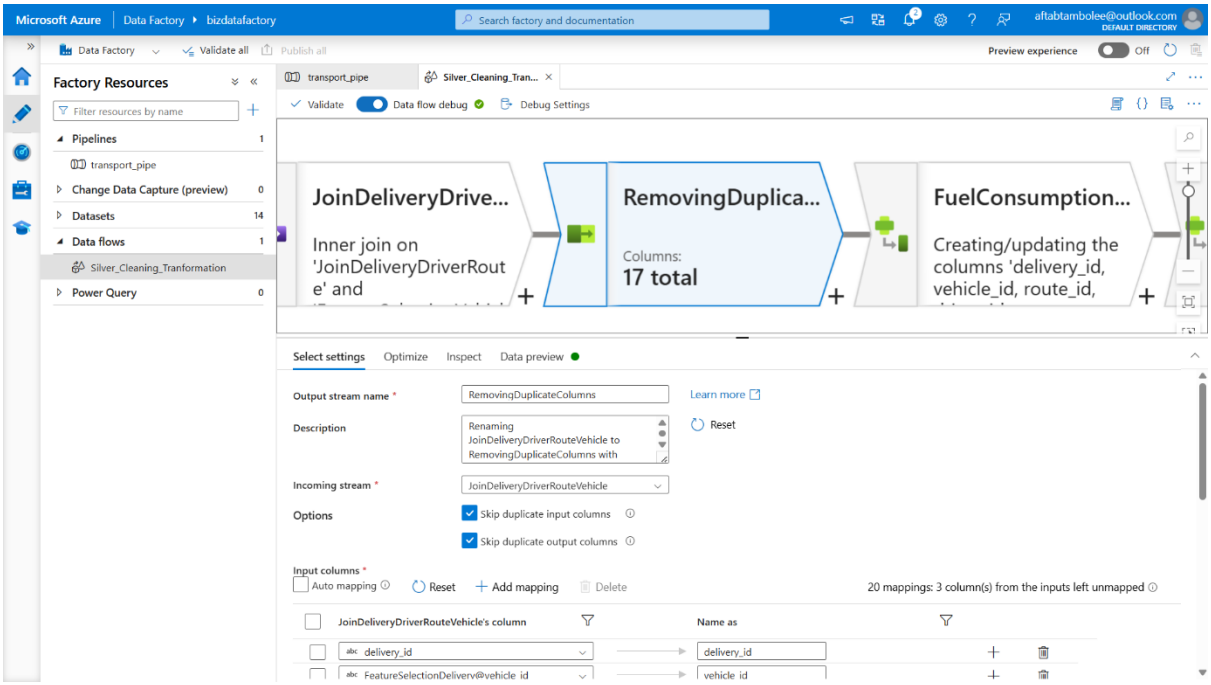
4. These three steps were performed for each of the bronze tables/files/datasets (delivery, driver, route, and vehicle):



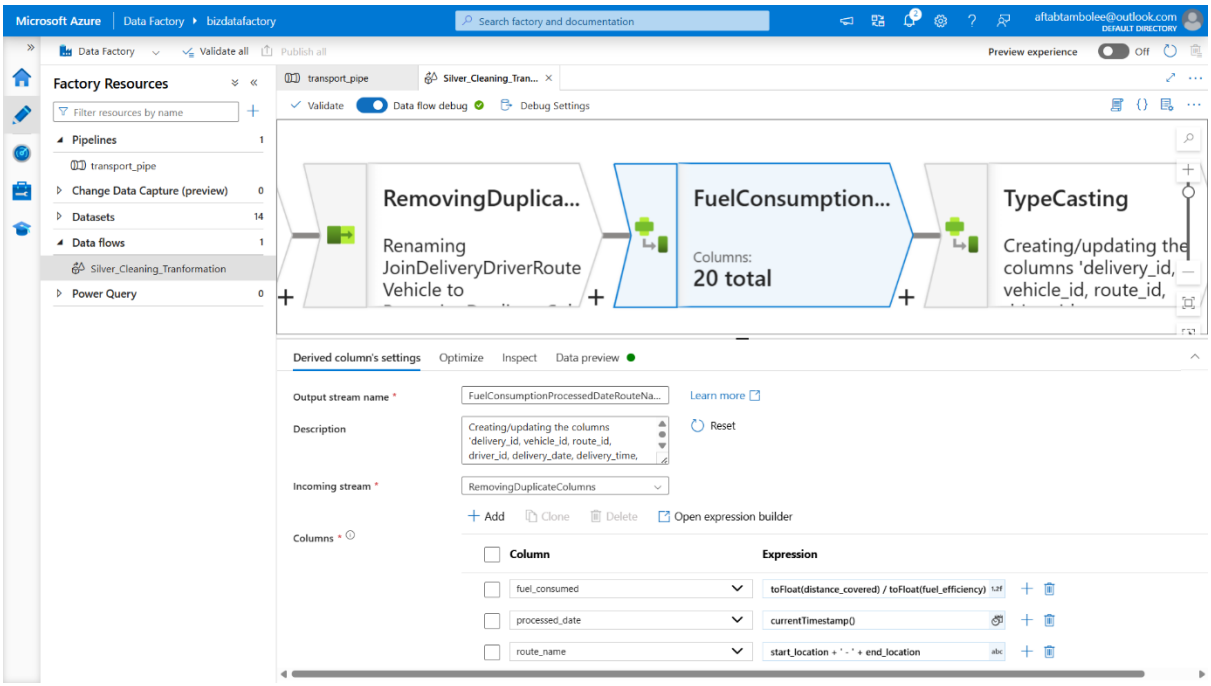
5. Join activity was applied to combine the four tables using primary keys and foreign keys with inner join:



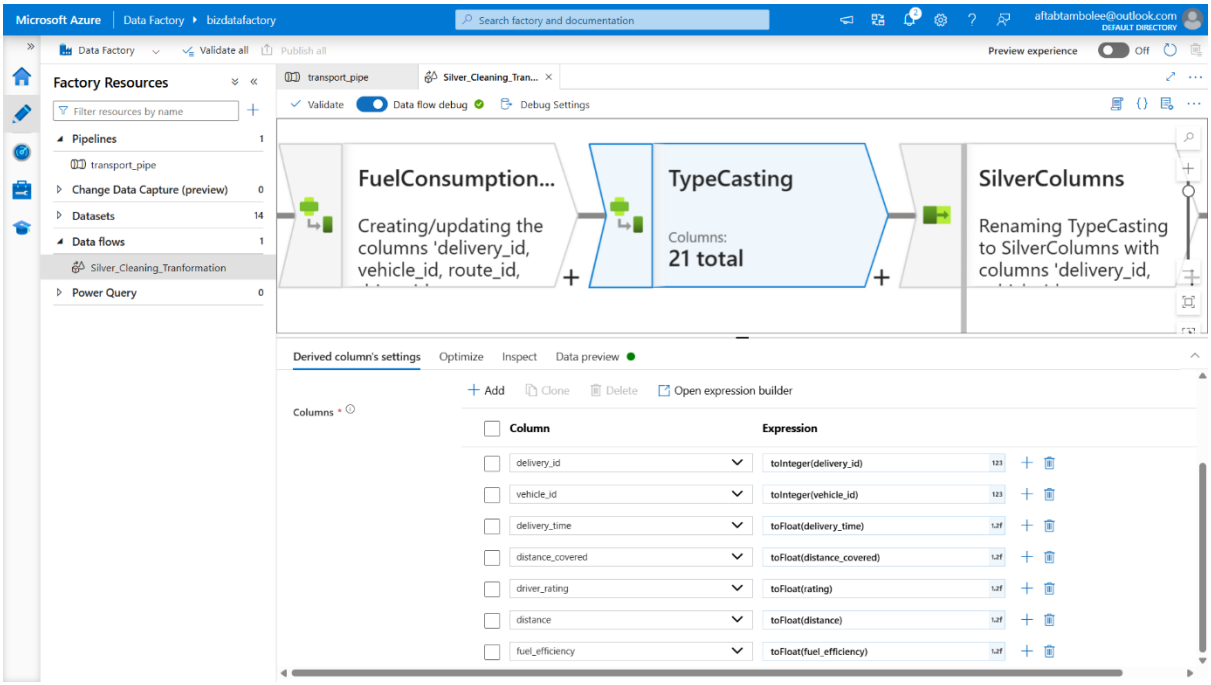
6. During joining, duplicate columns were removed using select activity:



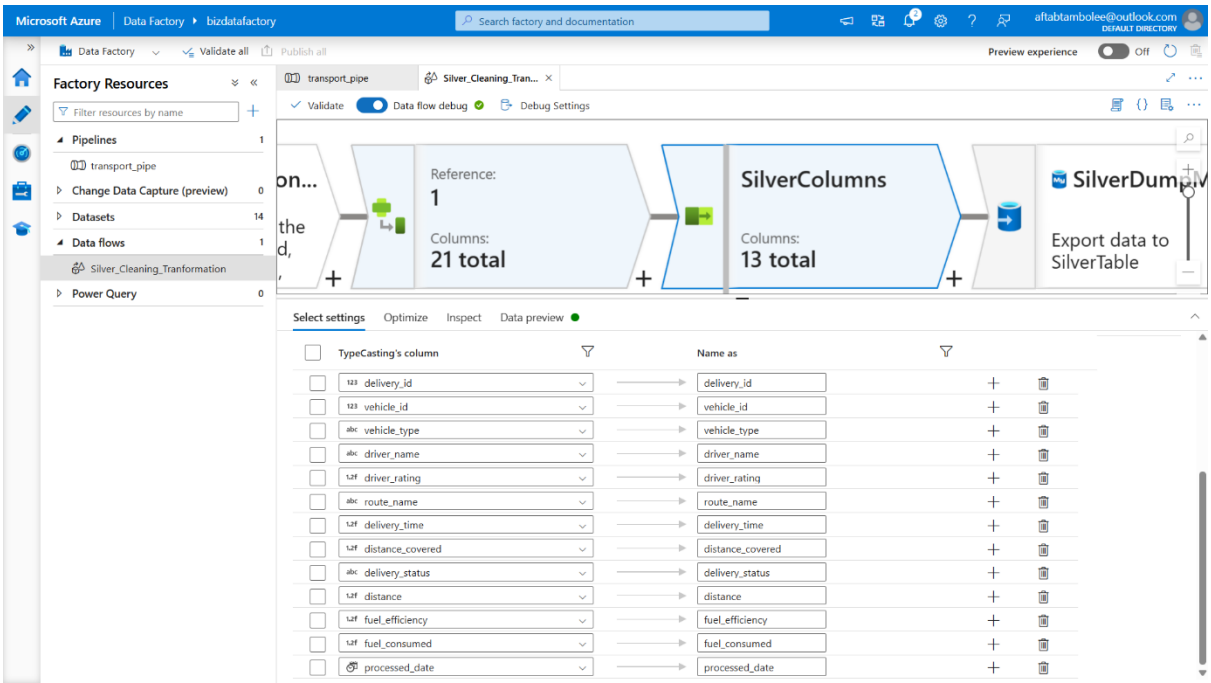
7. Additional columns were added for computing fuel_consumed, processed date, and route_name:



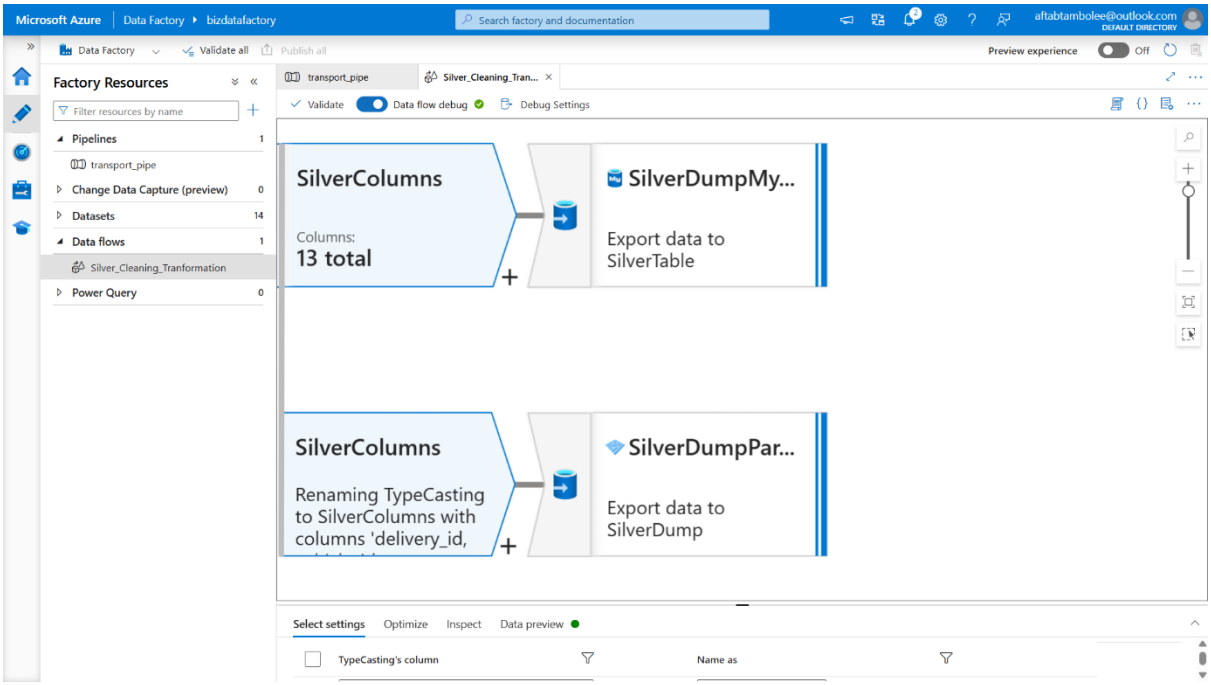
8. Type casting was performed on relevant columns to ensure correct data types for aggregation operations in the gold layer:



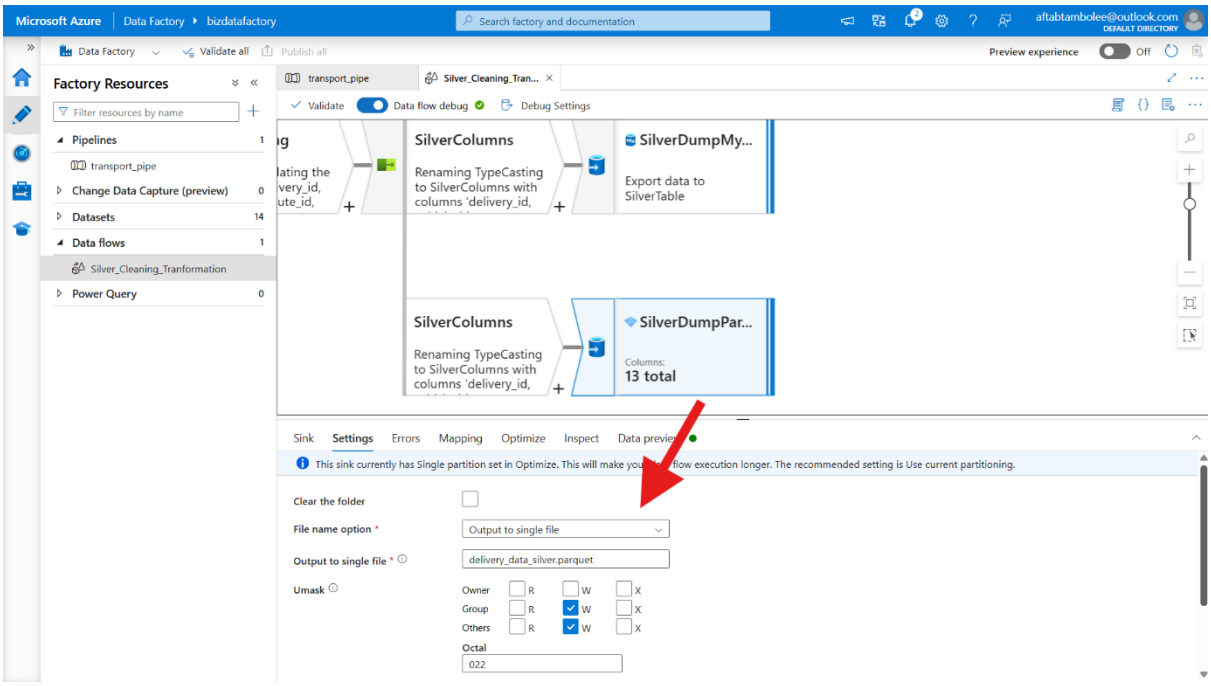
9. Select activity was used again to retain only important columns for the silver table:



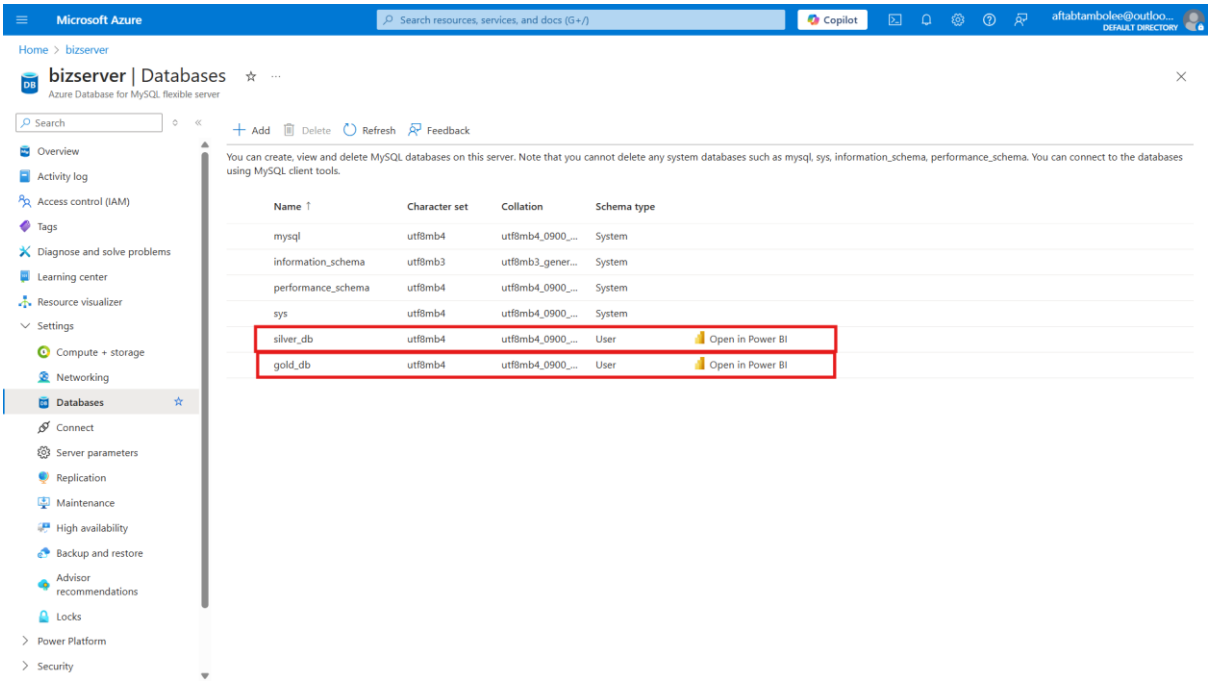
10. Two sink activities were used to store the prepared silver table in MySQL database (bizserver) and Data Lake Gen 2 (bizgenlake)'s silver container:



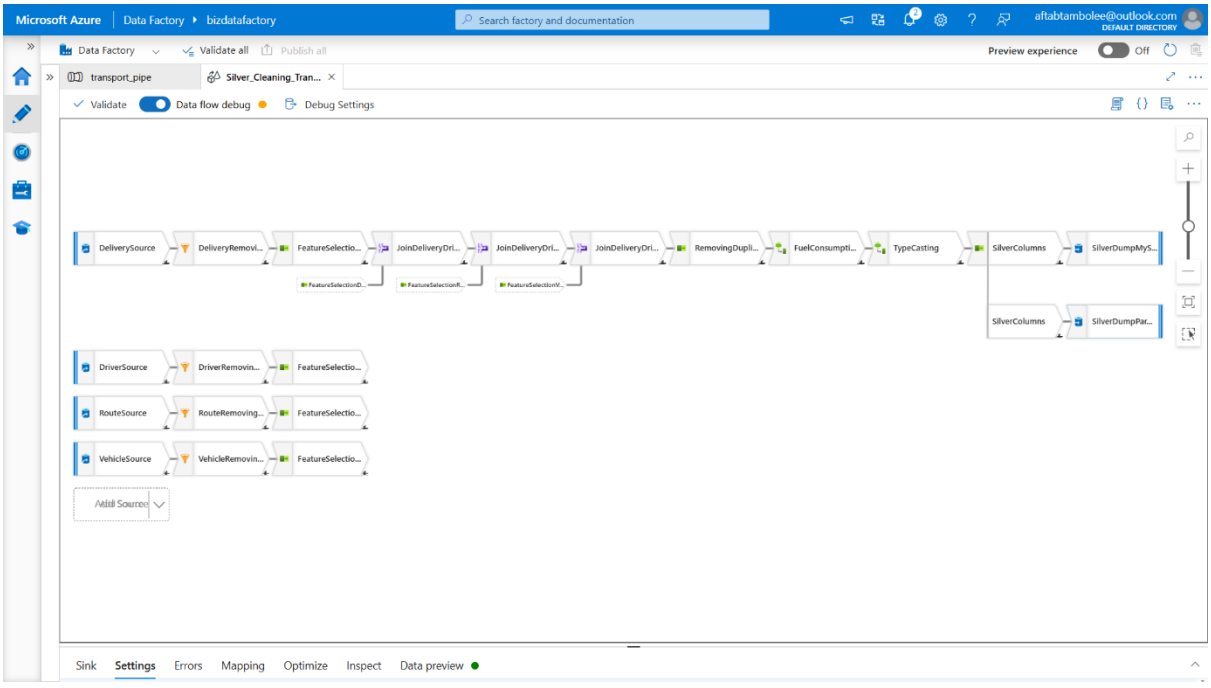
Custom naming for the parquet file is shown here:



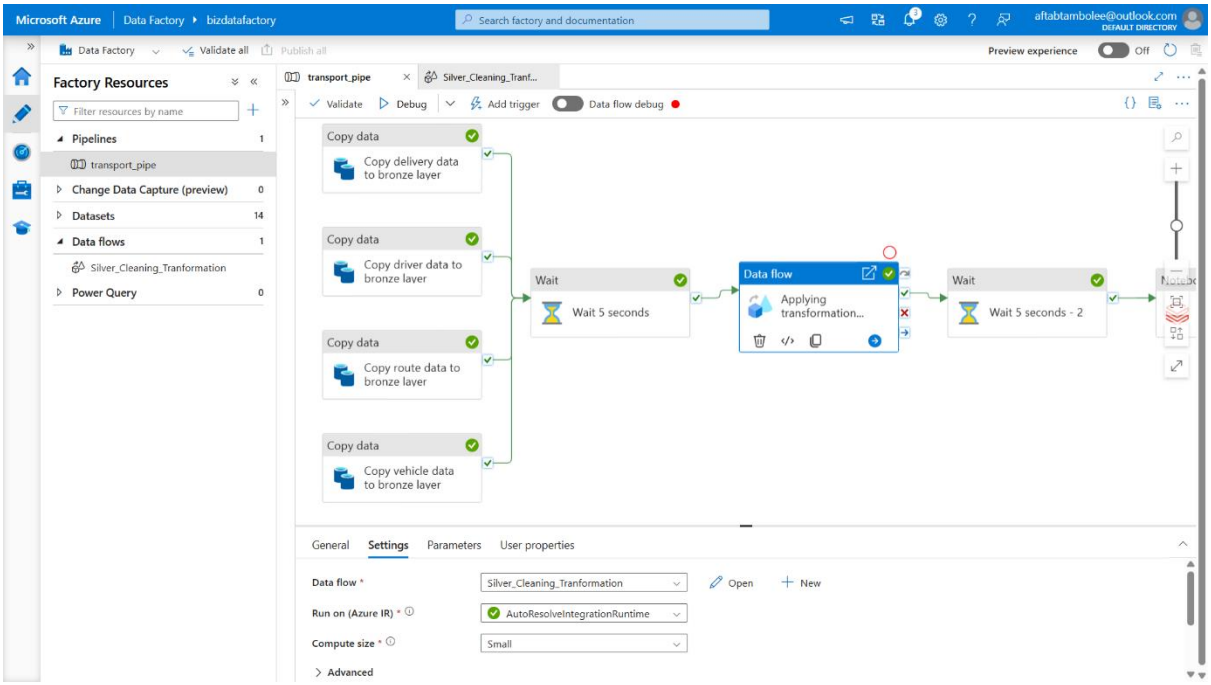
The MySQL connection was established between Azure Database for MySQL server and MySQL Workbench. Two databases were created for storing silver table and gold table, with the silver_db database/schema used to sink the dataflow's silver table:



The completed dataflow:

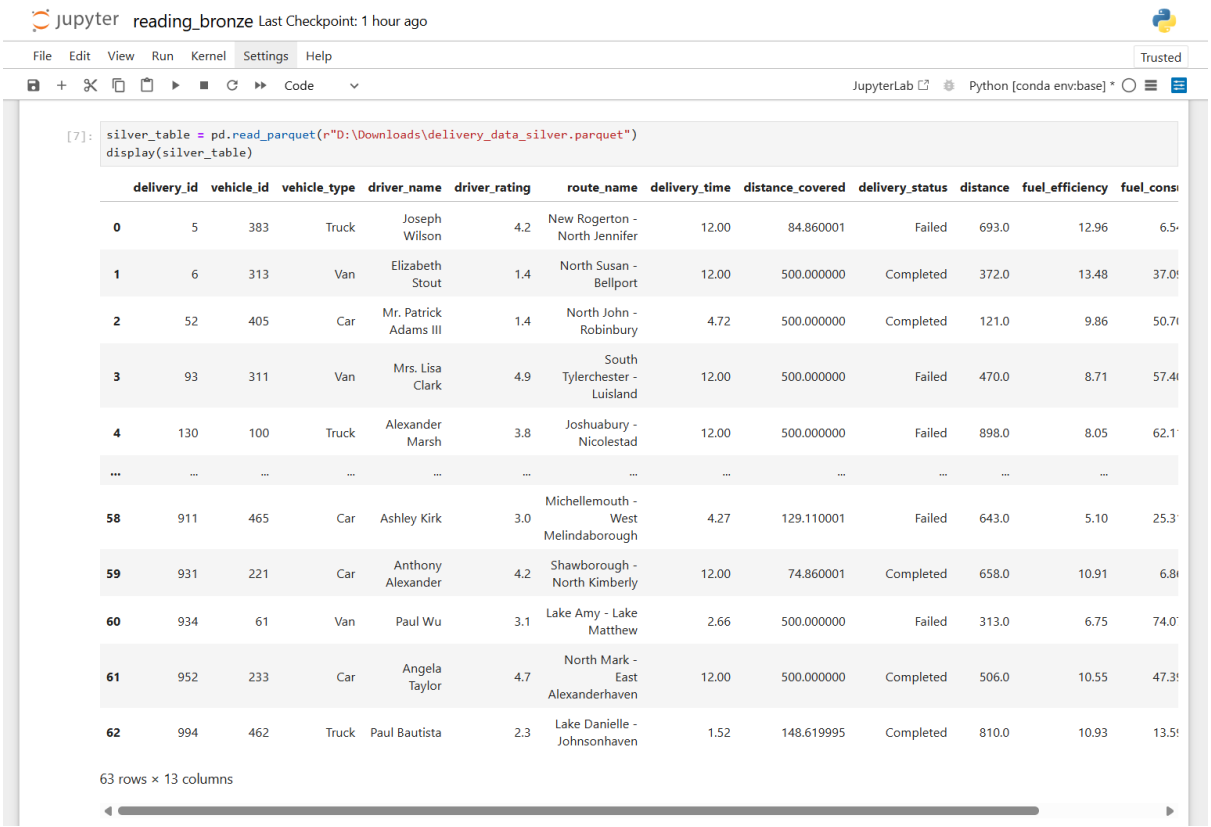


The dataflow was deployed in the data pipeline:



The final silver layer dataset/silver table:

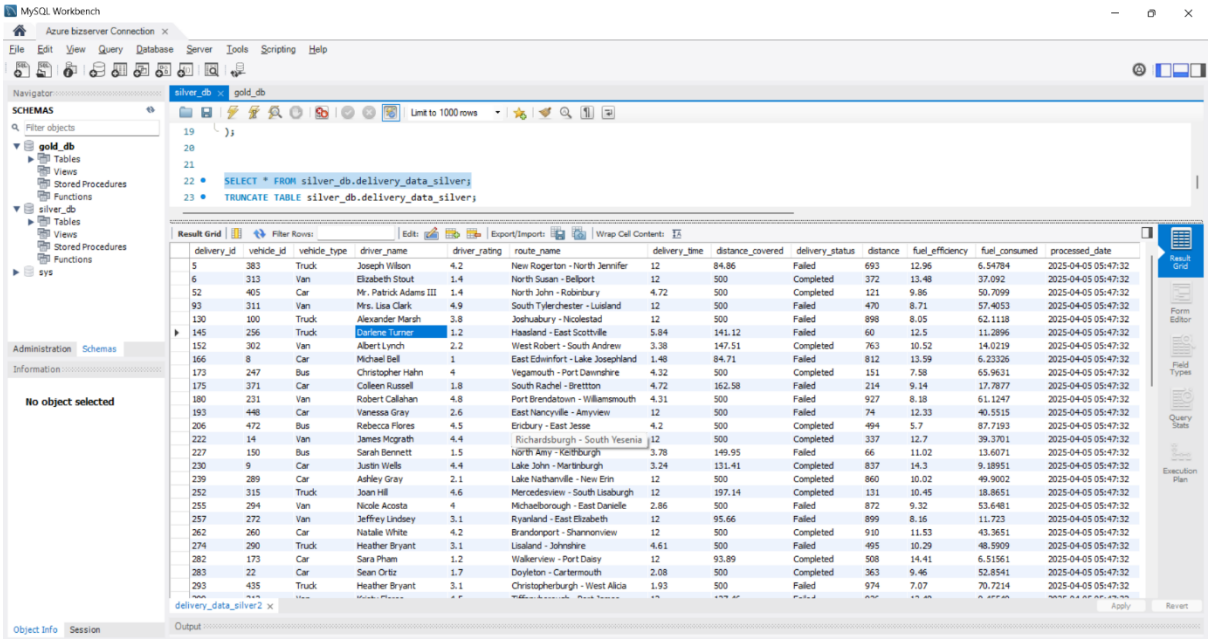
Silver container (parquet):



	delivery_id	vehicle_id	vehicle_type	driver_name	driver_rating	route_name	delivery_time	distance_covered	delivery_status	distance	fuel_efficiency	fuel_consu
0	5	383	Truck	Joseph Wilson	4.2	New Rogerton - North Jennifer	12.00	84.860001	Failed	693.0	12.96	6.5
1	6	313	Van	Elizabeth Stout	1.4	North Susan - Bellport	12.00	500.000000	Completed	372.0	13.48	37.0
2	52	405	Car	Mr. Patrick Adams III	1.4	North John - Robinbury	4.72	500.000000	Completed	121.0	9.86	50.7
3	93	311	Van	Mrs. Lisa Clark	4.9	South Tylerchester - Luisland	12.00	500.000000	Failed	470.0	8.71	57.4
4	130	100	Truck	Alexander Marsh	3.8	Joshuabury - Nicolestad	12.00	500.000000	Failed	898.0	8.05	62.1
...	...	...	...	...	...	...	...	...	...	...	...	...
58	911	465	Car	Ashley Kirk	3.0	Michellemouth - West Melindaborough	4.27	129.110001	Failed	643.0	5.10	25.3
59	931	221	Car	Anthony Alexander	4.2	Shawborough - North Kimberly	12.00	74.860001	Completed	658.0	10.91	6.8
60	934	61	Van	Paul Wu	3.1	Lake Amy - Lake Matthew	2.66	500.000000	Failed	313.0	6.75	74.0
61	952	233	Car	Angela Taylor	4.7	North Mark - East Alexanderhaven	12.00	500.000000	Completed	506.0	10.55	47.3
62	994	462	Truck	Paul Bautista	2.3	Lake Danielle - Johnsonhaven	1.52	148.619995	Completed	810.0	10.93	13.5

63 rows × 13 columns

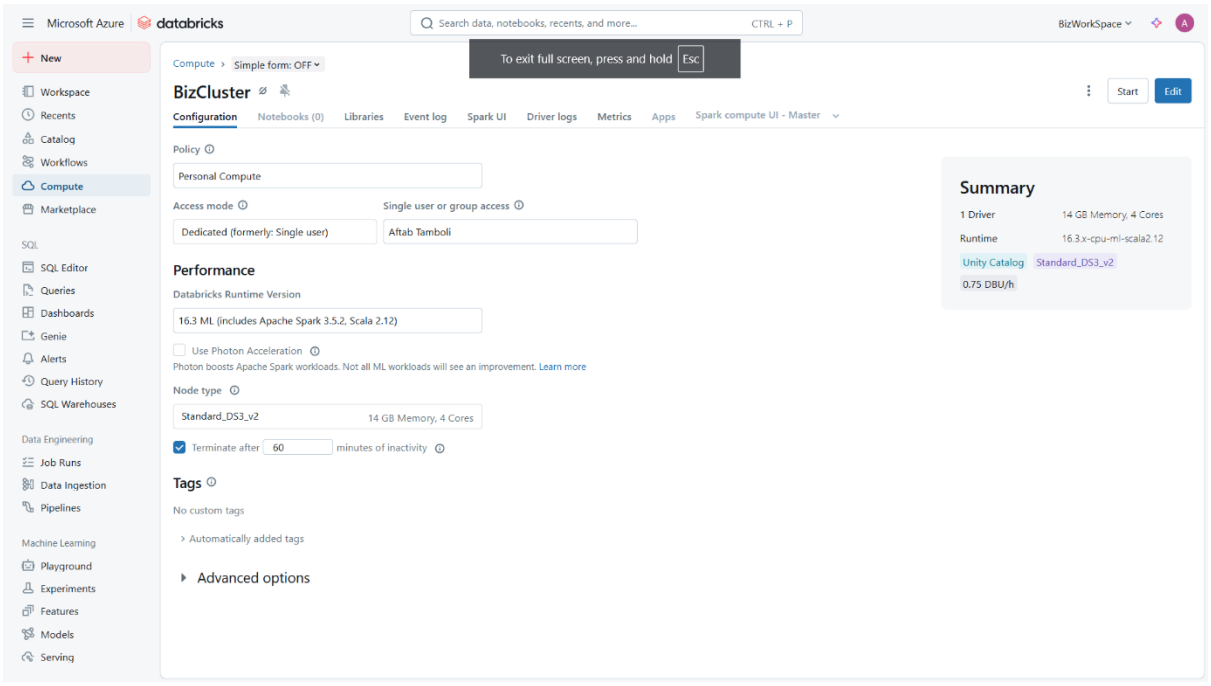
MySQL table:



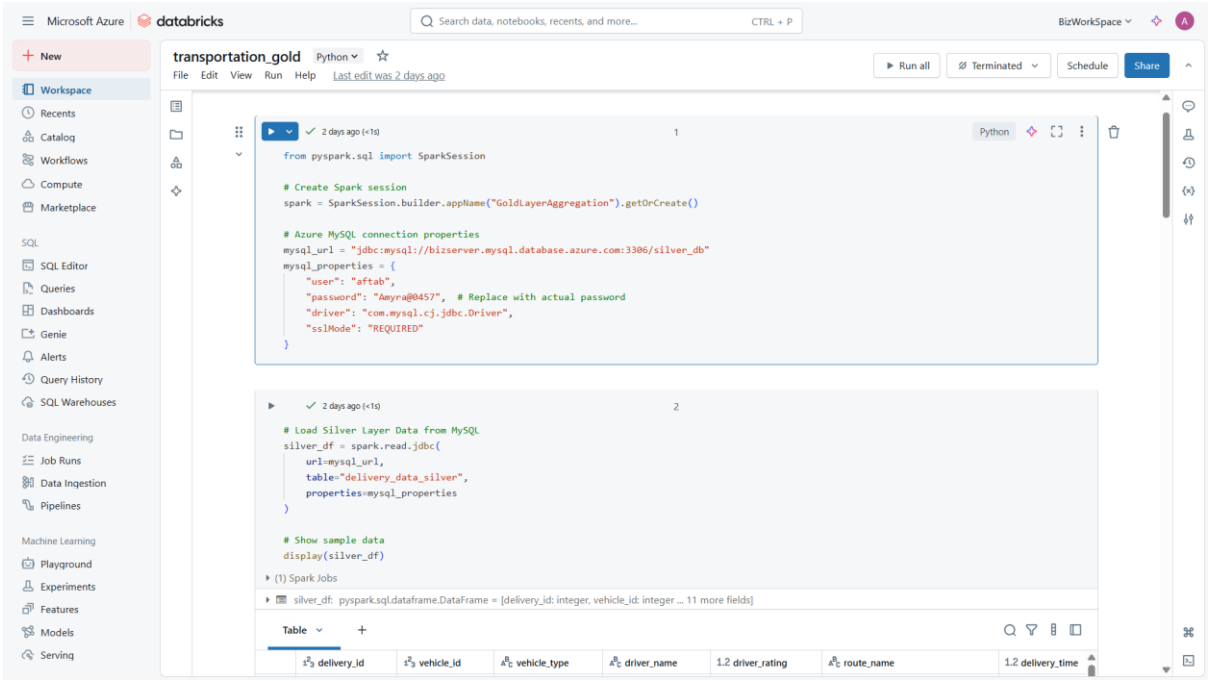
5.4 Gold Layer Implementation

For the gold layer, the transformed silver table was used to apply aggregations and create a new gold table in both parquet format and MySQL. Azure Databricks service was utilized for this purpose.

A new cluster named "BizCluster" was created in Databricks:



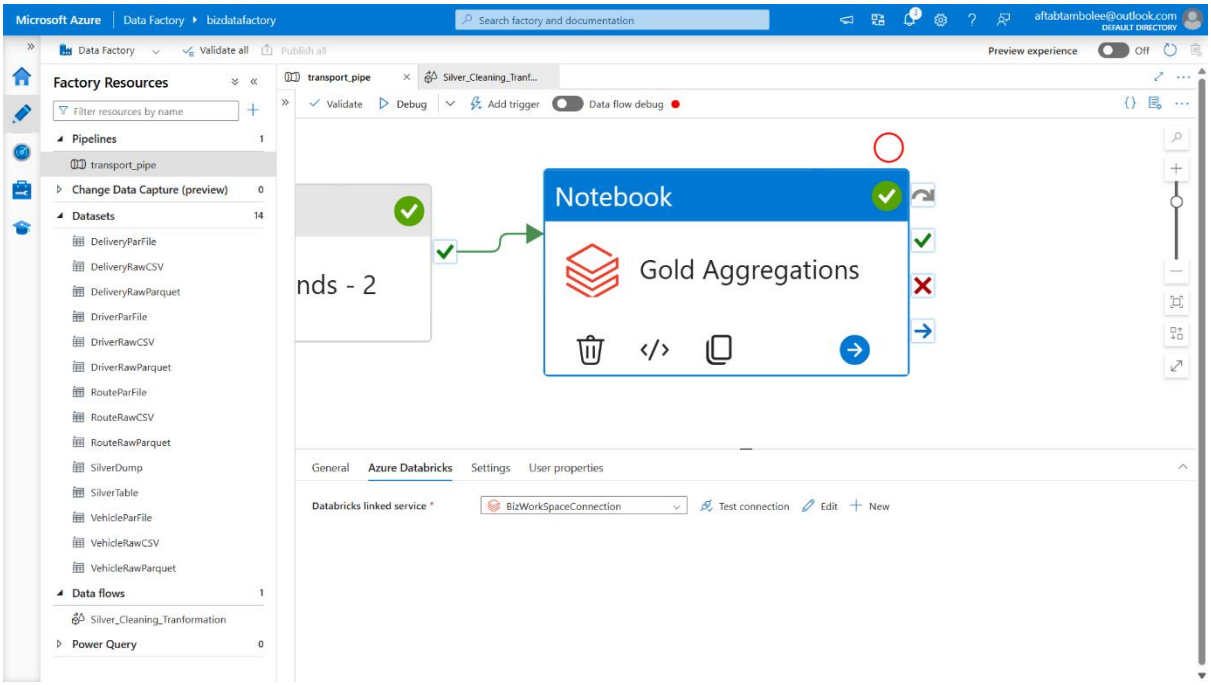
This cluster was attached to the PySpark notebook:



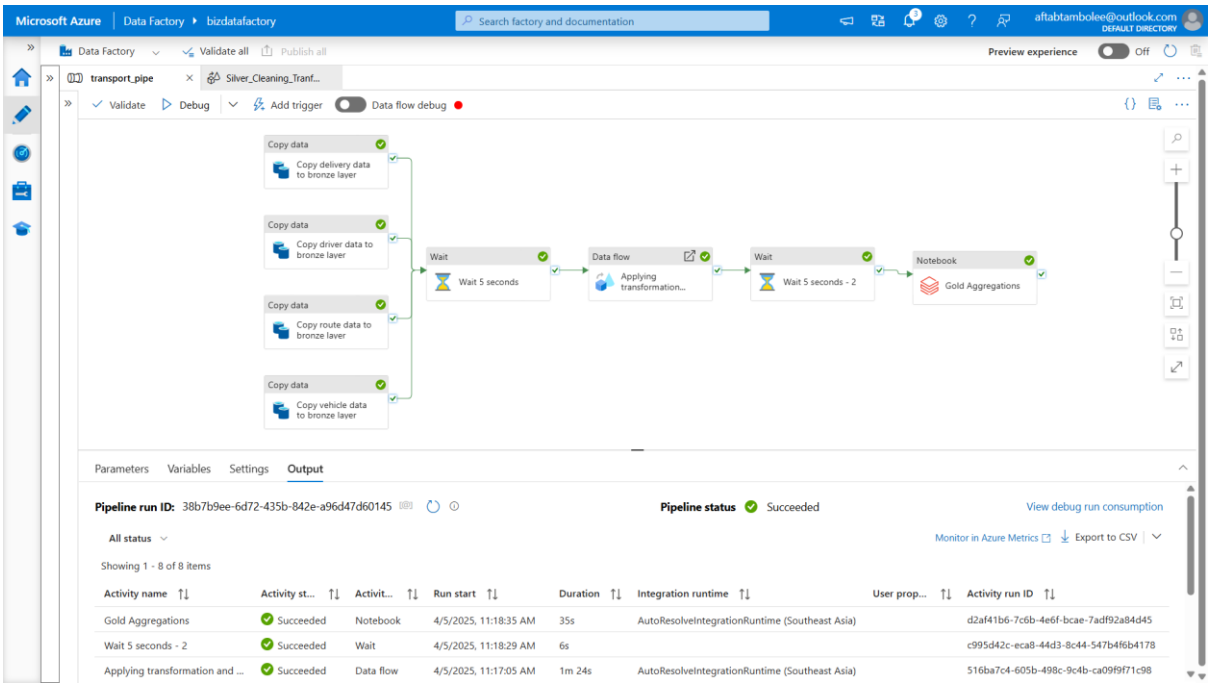
The notebook implementation included:

1. Creating a Spark session named "GoldLayerAggregation"
2. Establishing an Azure MySQL connection to import the silver table
3. Loading the silver table
4. Performing three aggregated operations to calculate:
    - Route optimization Analysis
    - Fleet performance
    - Driver performance
5. Joining the above three tables to create a single gold table
6. Arranging columns in the required sequential format
7. Connecting to MySQL's gold_db database to transfer the gold table
8. Connecting to Azure Lake Gen 2's bizlakegen storage to transfer the table to gold container in parquet format along with the logging file

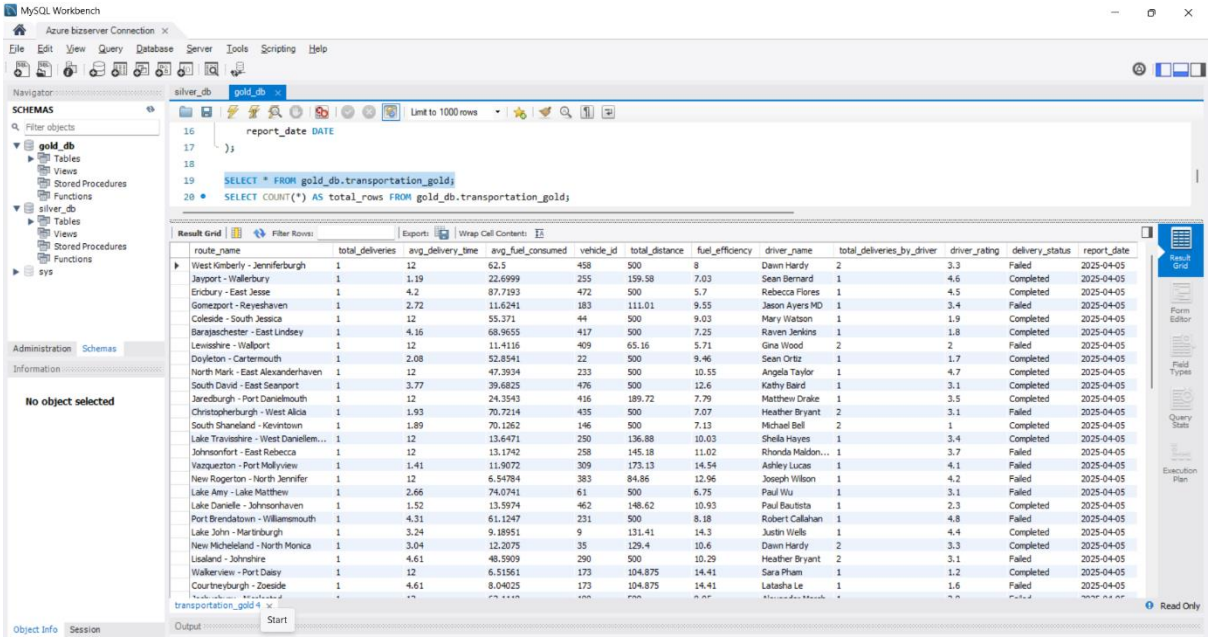
The notebook was deployed in the data pipeline:



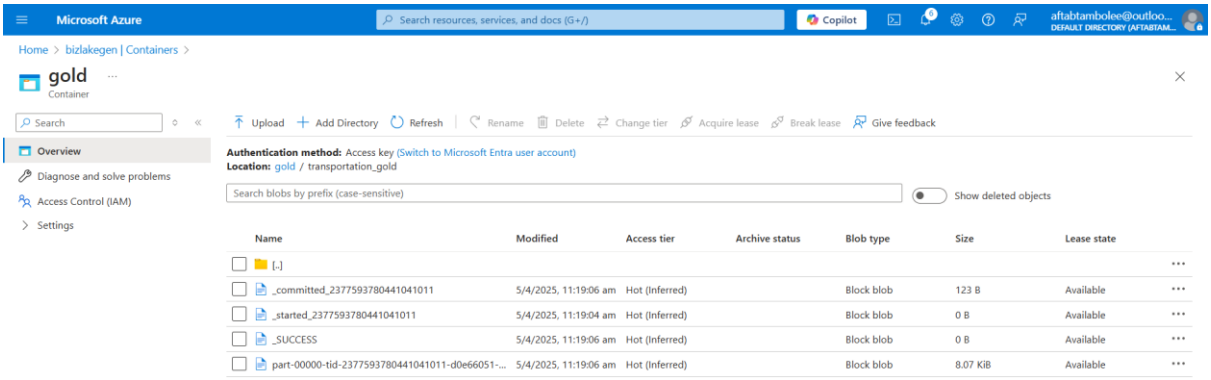
The complete data pipeline in Azure Data Factory:



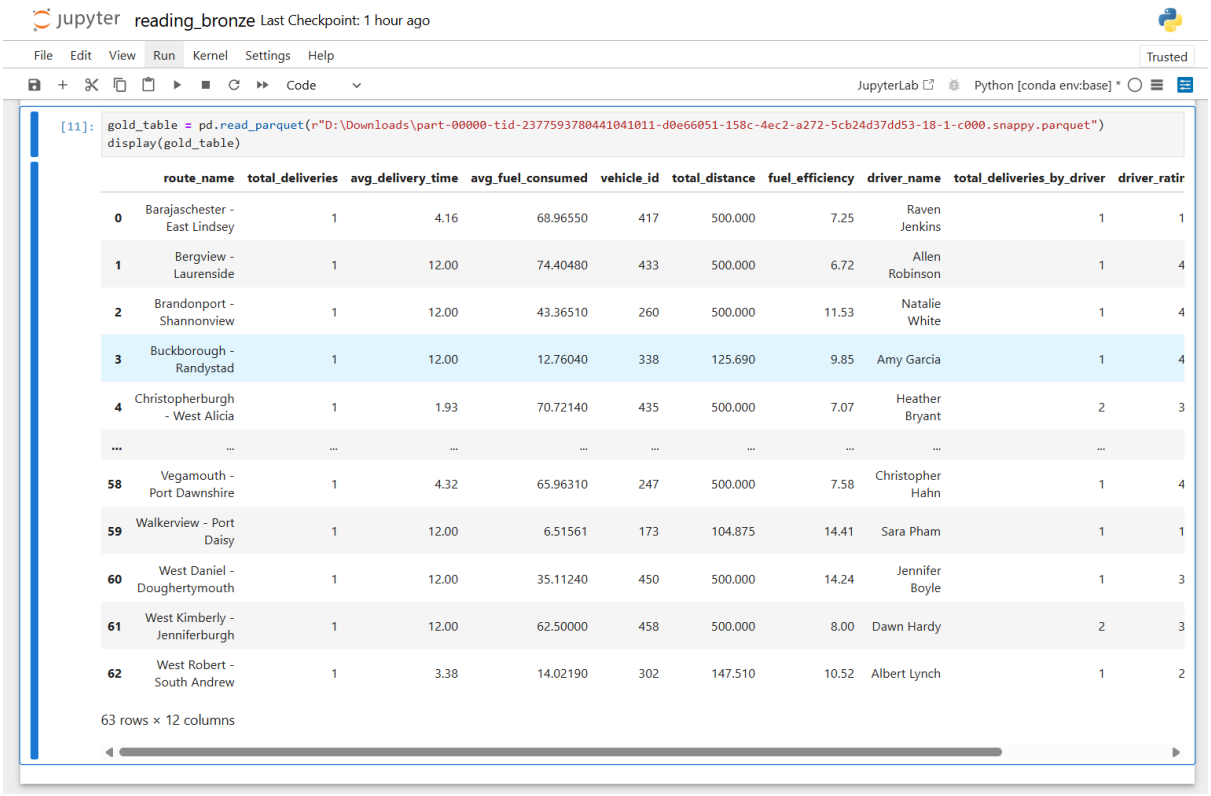
The gold table in MySQL (gold_db.transportation_gold):



The gold container in parquet format with logging file:



The gold container's parquet file:



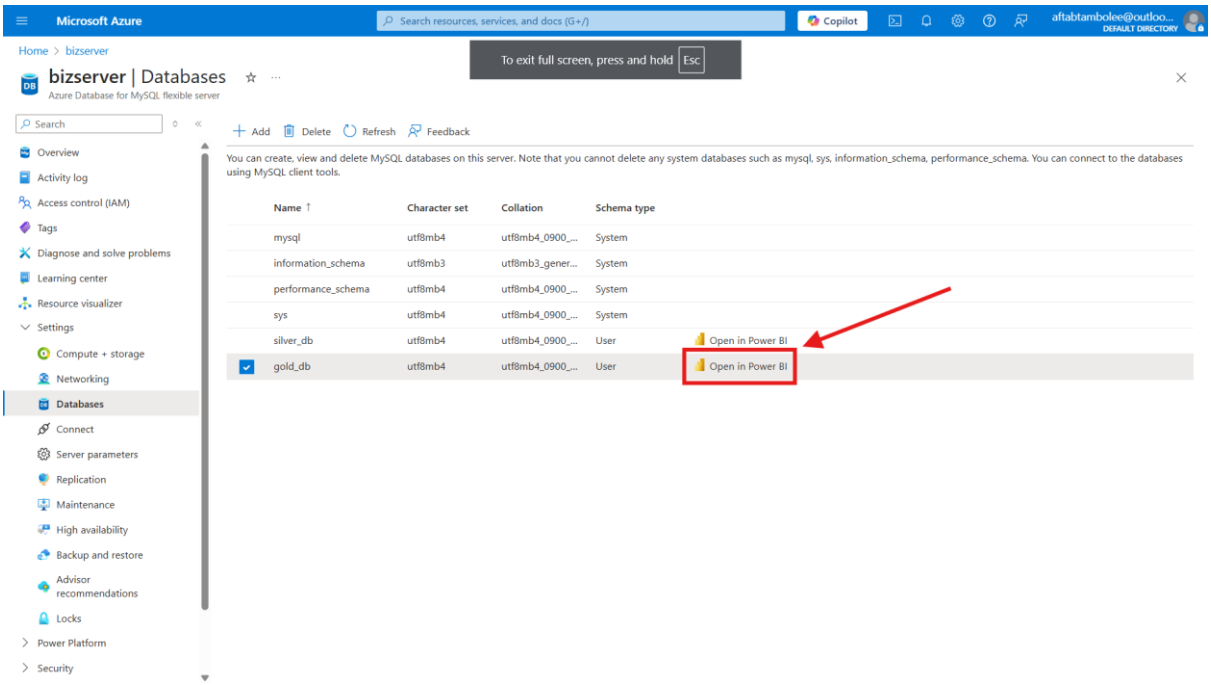
	route_name	total_deliveries	avg_delivery_time	avg_fuel_consumed	vehicle_id	total_distance	fuel_efficiency	driver_name	total_deliveries_by_driver	driver_ratir
0	Barajaschester - East Lindsey	1	4.16	68.96550	417	500.000	7.25	Raven Jenkins	1	1
1	Bergview - Laurenside	1	12.00	74.40480	433	500.000	6.72	Allen Robinson	1	4
2	Brandonport - Shannonview	1	12.00	43.36510	260	500.000	11.53	Natalie White	1	4
3	Buckborough - Randystad	1	12.00	12.76040	338	125.690	9.85	Amy Garcia	1	4
4	Christopherburgh - West Alicia	1	1.93	70.72140	435	500.000	7.07	Heather Bryant	2	3
...	...	...	...	...	...	...	...	...	...	
58	Vegamouth - Port Dawnshire	1	4.32	65.96310	247	500.000	7.58	Christopher Hahn	1	4
59	Walkerview - Port Daisy	1	12.00	6.51561	173	104.875	14.41	Sara Pham	1	1
60	West Daniel - Doughertymouth	1	12.00	35.11240	450	500.000	14.24	Jennifer Boyle	1	3
61	West Kimberly - Jenniferburgh	1	12.00	62.50000	458	500.000	8.00	Dawn Hardy	2	3
62	West Robert - South Andrew	1	3.38	14.02190	302	147.510	10.52	Albert Lynch	1	2

63 rows × 12 columns

At this point, the medallion architecture implementation is complete, with all steps performed: Raw data → Bronze layer → Silver layer → Gold layer

6. Power BI Dashboard

The gold table was then used for reporting and creating a Power BI dashboard. A connection file was downloaded from the Azure Database for MySQL server (bizserver) that automatically creates the connection between MySQL gold_db table and Power BI:



6.1 Key Performance Indicators

Five KPIs were created as per documentation:

1. **Total Deliveries**: Sum of total deliveries per route/vehicle/driver
2. **Average Delivery Time**: Average delivery time per route/vehicle/driver
3. **Fuel Efficiency**: Average fuel efficiency per vehicle
4. **Driver Performance**: Average driver rating
5. **Delivery Status**: Completed vs Failed

6.2 Visualizations

The dashboard was created with the following visualizations:

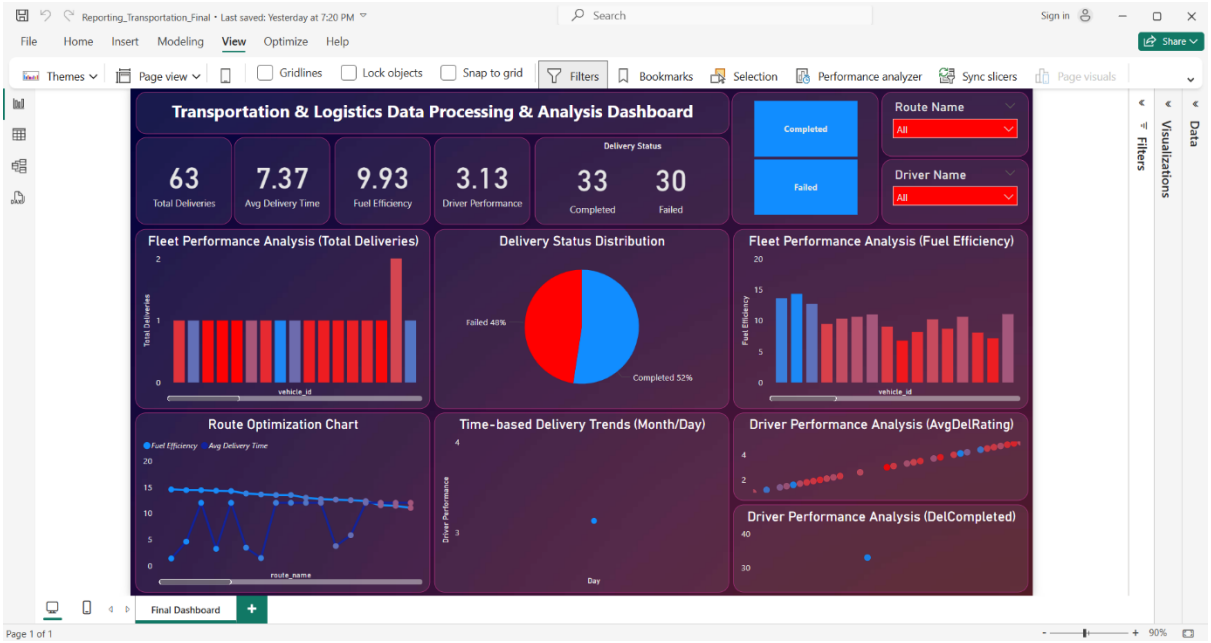
1. **Route Optimization Chart**: Line chart showing average delivery times and fuel consumption per route
2. **Fleet Performance Analysis**: Bar chart visualizing total deliveries per vehicle and fuel efficiency
3. **Driver Performance Analysis**: Scatter plot showing driver performance based on deliveries completed and average ratings
4. **Delivery Status Distribution**: Pie chart showing the distribution of completed and failed deliveries across routes
5. **Time-based Delivery Trends**: Line chart showing monthly/weekly delivery performance trends

The dashboard includes slicers and filters for:

- Delivery status (completed/failed)

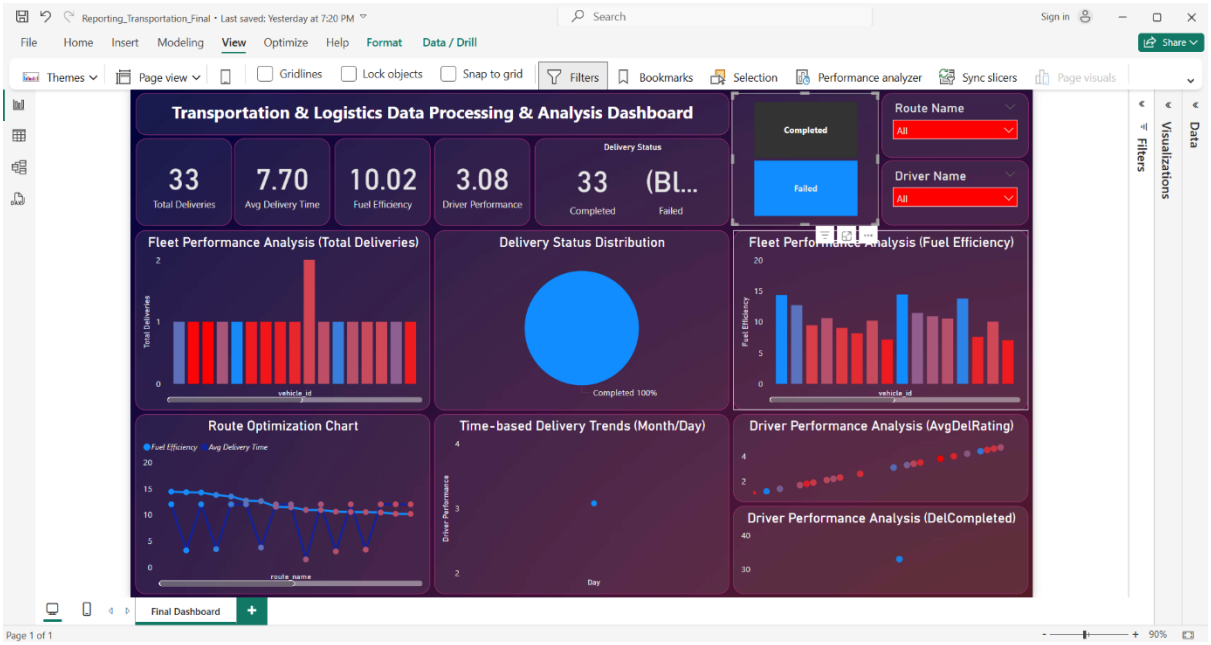
- Route name
- Driver name

The final dashboard:

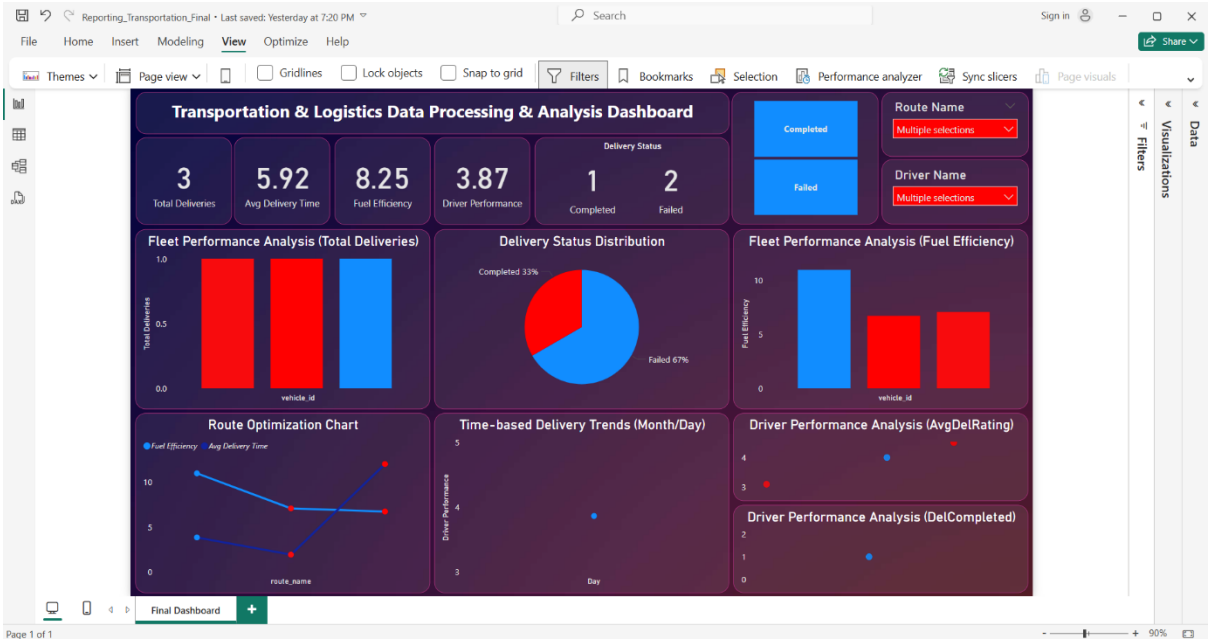


6.3 Examples

Example 1: Dashboard filtered by deliveries completed:



Example 2: Dashboard filtered by selected route name and driver name:



7. Conclusion

This project successfully implements a complete data pipeline following the medallion architecture on Microsoft Azure. The pipeline automatically processes transportation and logistics data from raw format to insightful visualizations, requiring minimal manual intervention. The solution demonstrates the effective use of various Azure services to create a scalable, automated, and reliable data processing system that provides valuable business insights through Power BI dashboards.