To complete this project, I used the data that I collected from my midterm project. This included the classic models database in the mysqlsampledatabase.sql file, and the API data from the public API FFXIV Collect. Full documentation for that process can be found in my midterm project submission.

Then, for this project, I first created an Azure MySQL database and transferred my old database into that new database. Similarly, I had to create a new database in MongoDB Atlas to store my data, noting both of these databases' names.

In order to accommodate the requirements, I exported my fact orders table and split it into three separate JSON files. Using this, I was able to simulate a data stream once I had completed my code. Similarly, I also exported several other dimension tables into JSON files, as they would be necessary to move onto MongoDB Atlas. I placed the files within DBFS on Databricks, and ensured my paths referenced the right files (the fact order files within the stream folder, and the dim_ tables within the batch folder). From there, I began coding.

From here, to create the ETL pipeline, I first pulled the date dimension data from Azure MySQL and stored it as a delta table in my new database. I then used MongoDB Atlas to store and pull data from the customers, employees, and mounts dimension tables. I also stored these tables into new delta tables in my new database.

After completing the cold path data and storing it all in delta tables, I started creating the autoloader. This involved creating three different levels of Bronze, Silver, and Gold. In the Bronze table, I began to read the data stream from the fact order table (that was split into three files). Then, I added metadata onto this table, and also created a temporary view, from which I created a new delta table. Similarly, I created the Silver table and associated temporary view, which merged the fact order data with the dimension tables, and then wrote it to a new delta table. Lastly, I created the Gold table using the CTAS approach, and aggregated the data, and returning a list of customers based on how many units they ordered and the total price.