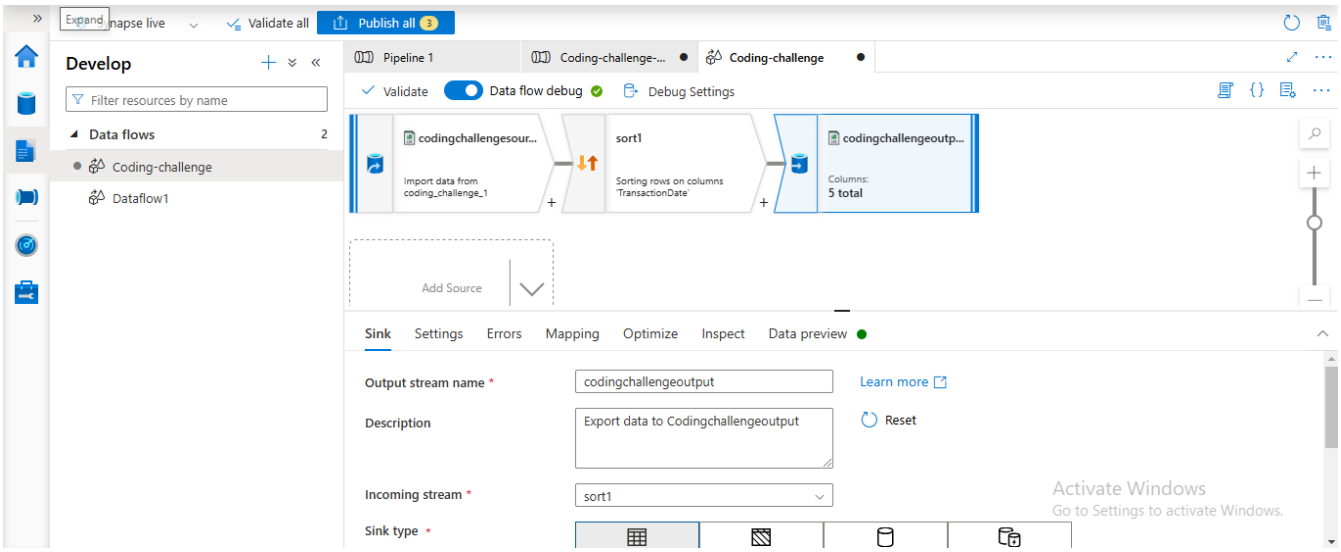
# DEVOPS CODING CHALLENGE

## HARSHINI V

**Build an etl pipeline with azure synapse with dataflow running on it**

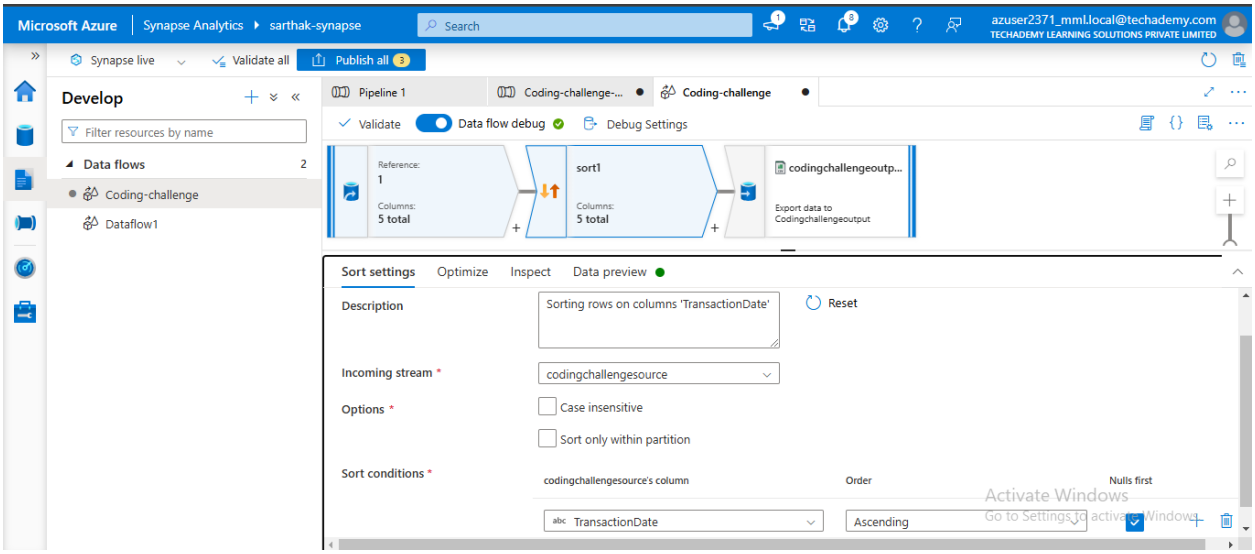
### Setup Dataflow

To begin building the ETL pipeline, I set up the Dataflow in Azure Synapse Studio. I navigated to the Integrate section and created a new Dataflow to start defining the transformations. I added a source dataset, which was configured to pull data from Azure Data Lake. The dataflow allowed me to process the raw data efficiently by applying various transformations like filtering, aggregating, and joining multiple datasets. This step laid the foundation for transforming raw data into a clean, usable format ready for analysis.



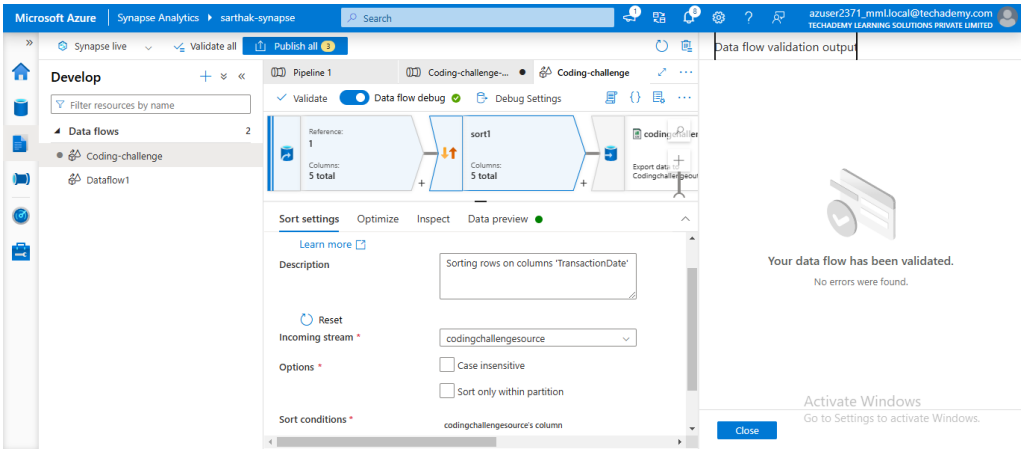

### Configure Sort Settings

Next, I configured the sort settings to ensure the data is organized properly within the dataflow. Sorting was essential to arrange the records in a logical sequence, particularly for aggregation and join operations. I set up sorting based on specific columns, ensuring that data was ordered before applying further transformations. The sorting settings also helped optimize performance and ensure that downstream activities, such as data loading, occurred in the correct order for accurate results.




### Validate the whole dataflow

After building the dataflow, I validated the entire structure to ensure that all transformations and settings were applied correctly. This step involved checking each transformation and ensuring that the source, transformations, and sink were all properly linked. I ran sample data through the dataflow to confirm that the flow was working as expected, checking for any errors or issues that could prevent the ETL process from running smoothly. This validation helped identify potential bottlenecks or misconfigurations early on.

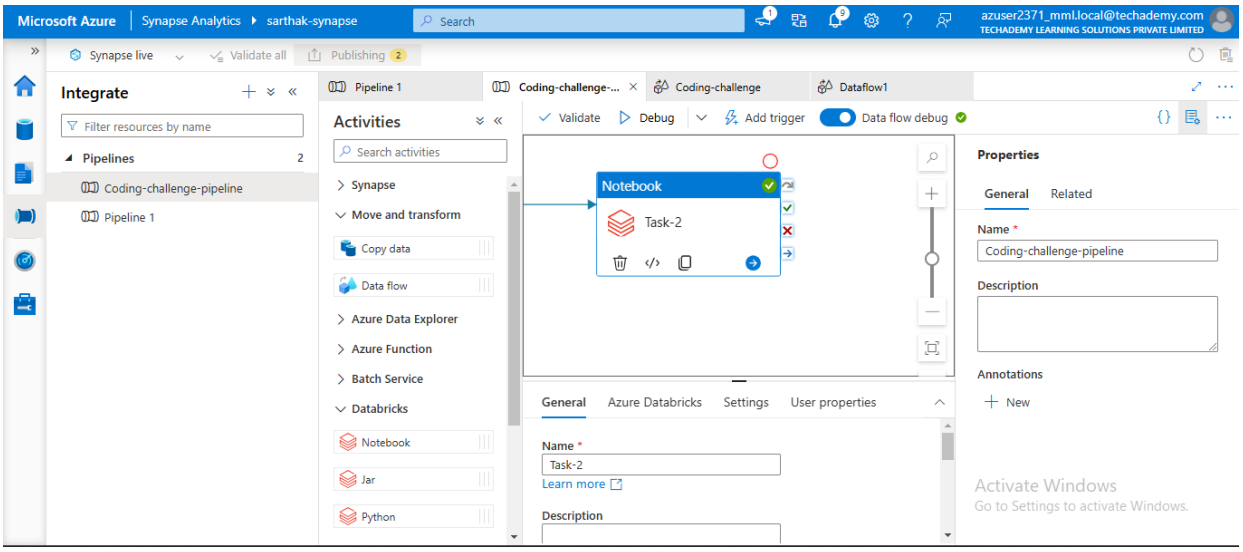



## PIPELINE

### 1. Insert Databricks Notebook

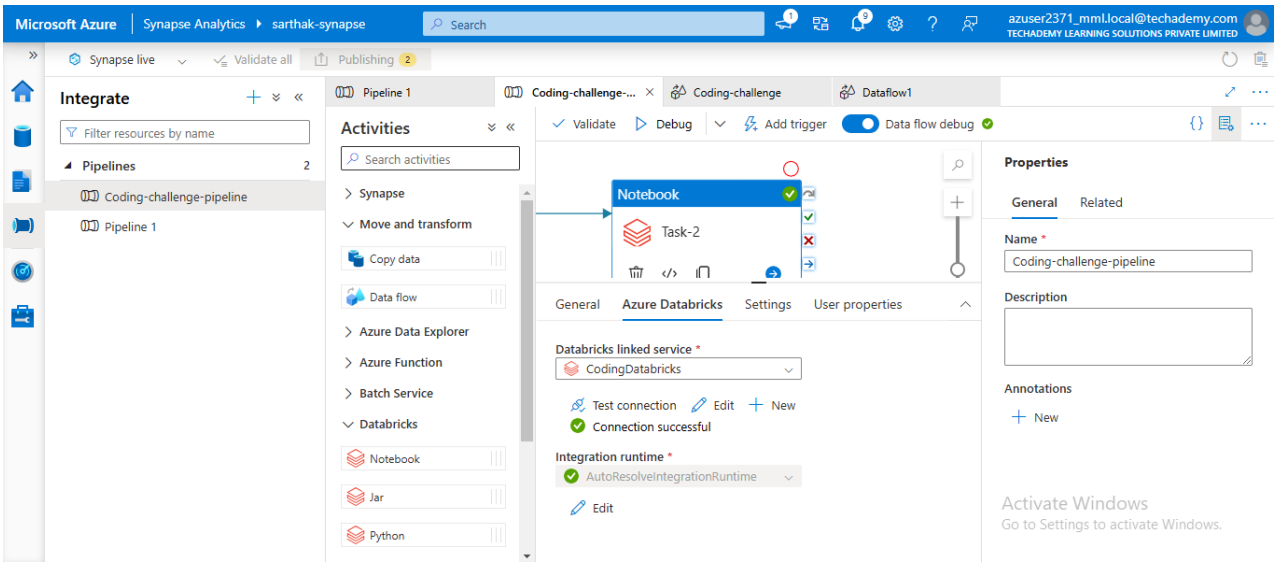
In the next task, I inserted a Databricks notebook into the pipeline to execute complex transformations on the data. This notebook acted as a bridge to process data with PySpark, utilizing the advanced computational power of Azure Databricks. I created a new notebook within Databricks and embedded it into the pipeline.

This integration allowed me to handle large datasets and complex transformations that were difficult to manage within Dataflow alone.



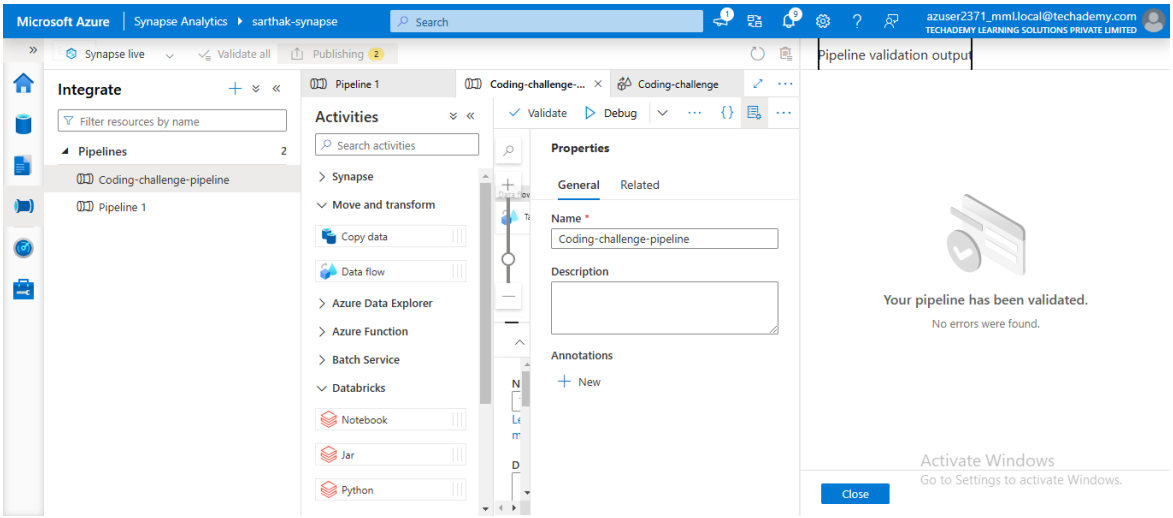

## 2. Connect the Azure Databricks notebook

I connected the Azure Databricks notebook to the pipeline by configuring the notebook activity in Synapse. I set the connection parameters to ensure that the pipeline could trigger the notebook and pass necessary input data. This connection was crucial for linking the pipeline to the Databricks cluster, enabling the execution of Spark-based transformations. I ensured the notebook's output could be properly passed back into the pipeline for further processing or storage.



### 3. Then validate the notebook

Once the notebook was connected, I validated its execution to confirm that it was processing data correctly. I ran the notebook in isolation to ensure the logic was working as intended, and then tested it within the pipeline context to ensure smooth integration. This validation step was crucial to ensure that no errors would occur when the pipeline was executed in its entirety. I reviewed the notebook's output to check for data correctness and performance issues.




## DEPLOYING PIPELINE

### 1. Validate the whole Pipeline

Before deployment, I validated the entire pipeline to ensure that all activities, including the Dataflow and Databricks notebook, were correctly linked and configured. This involved checking every step, from data ingestion to transformations and output storage. I used the validation feature in Azure Synapse Studio to catch any potential errors or misconfigurations. This helped ensure that the ETL pipeline would run without issues when deployed in the production environment.