

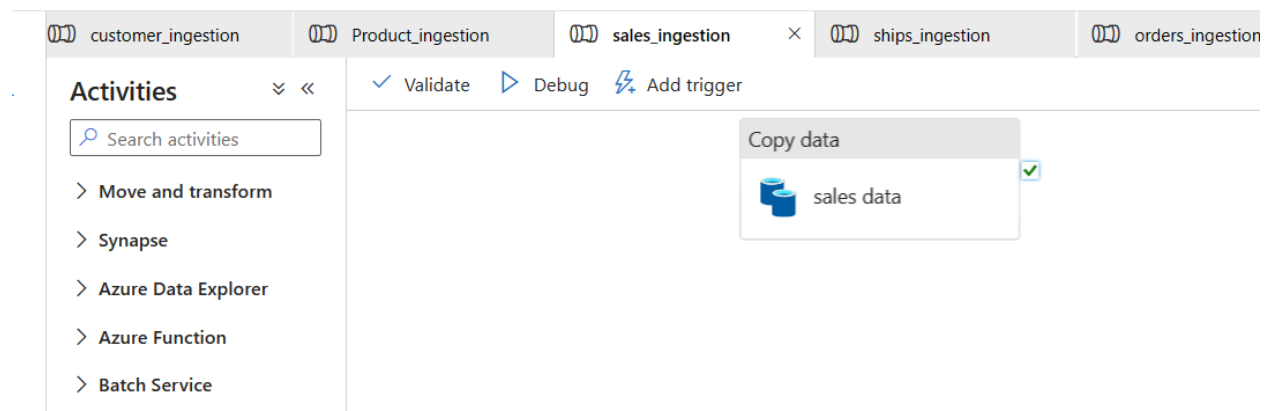
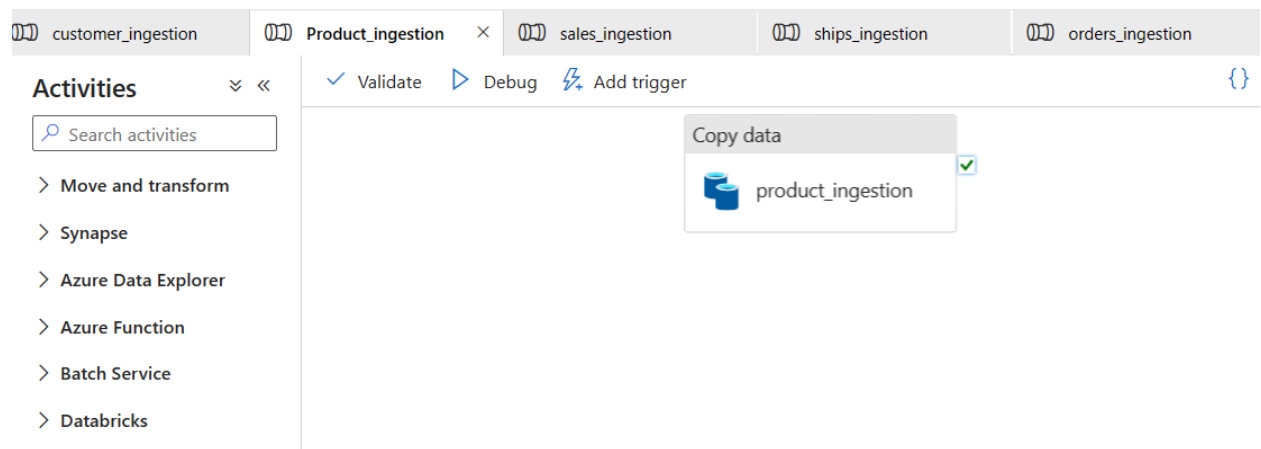
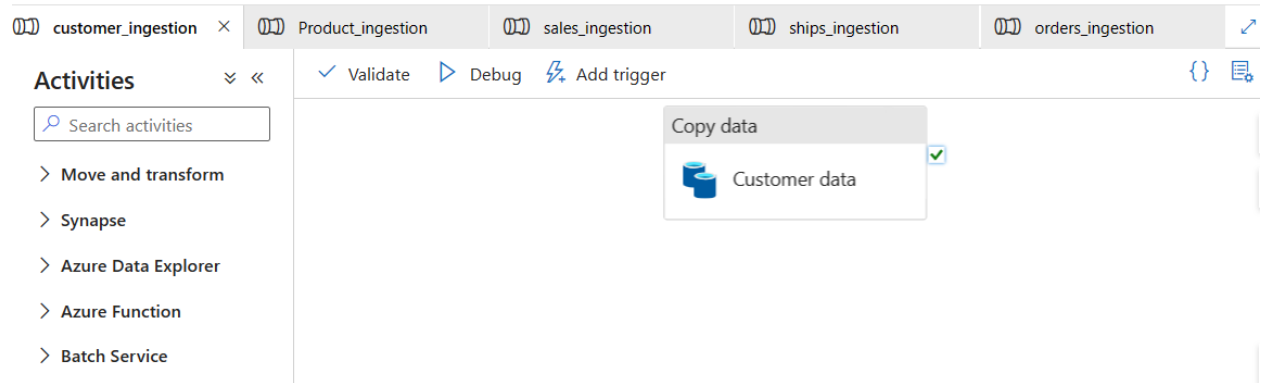
Store Data Analysis

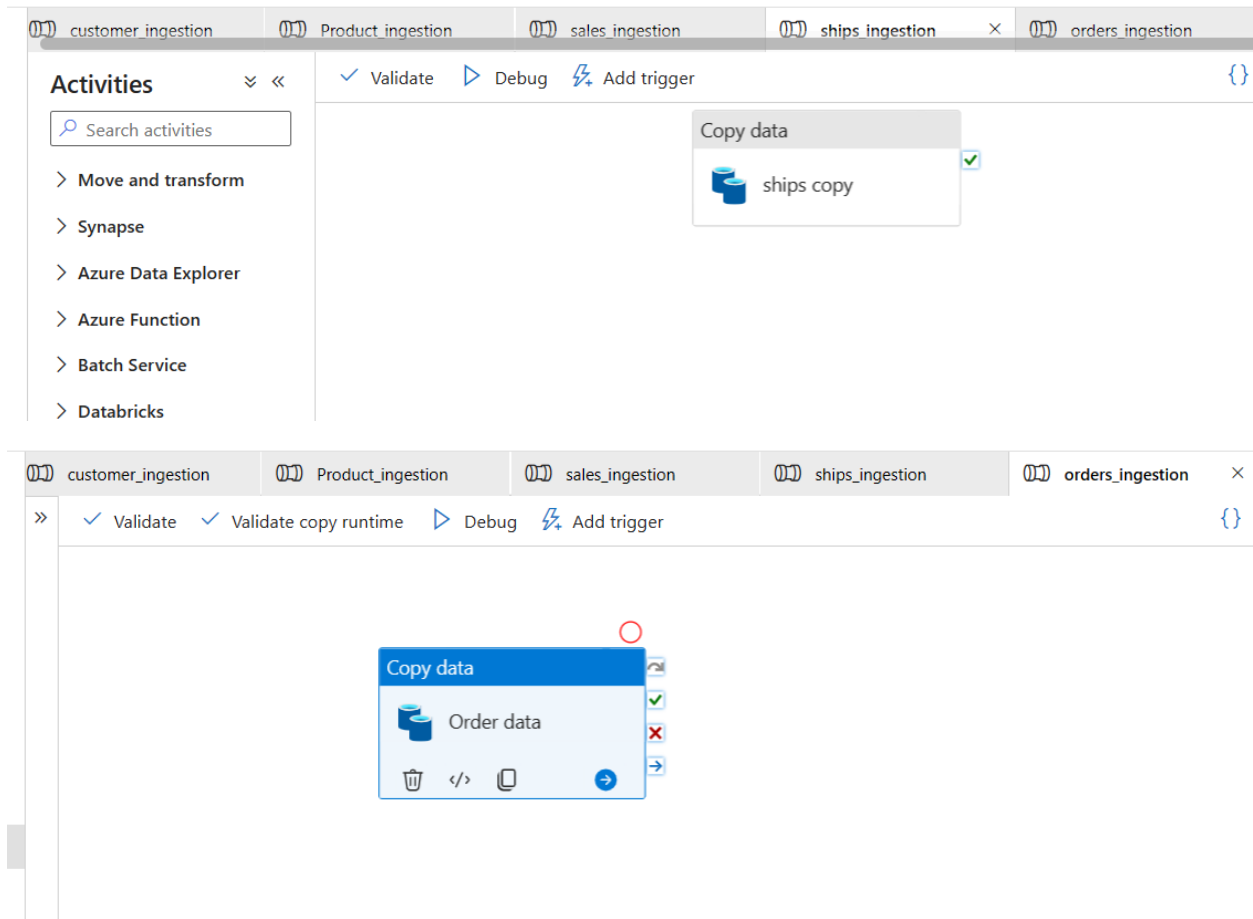
Objective: To perform an ETL process for order, ship, sale, product, and customer data using Azure Data Factory (ADF), Azure Databricks, and Azure SQL, as well as orchestrate the process with Apache Airflow.

1. Azure Data Factory (ADF) Ingestion Process:

Ingest data from CSV files into Blob Storage and convert it to Parquet format for the Raw layer. Here we create a separate Data Factory pipeline for each data source (order, ship, sale, product, customer).

Copy Data Activity: Use the "Copy Data" activity to move CSV files from a source location to or Blob Storage.





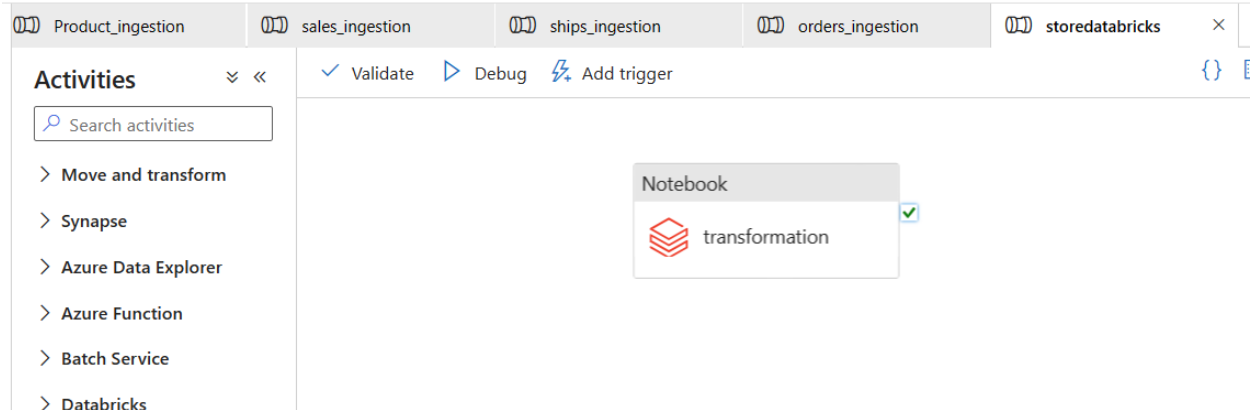
2. Azure Databricks Transformation Process:

Use Azure Databricks to normalize, clean unwanted data, drop duplicates, replace null values, perform analysis, and store the data in Azure SQL for the Transformation layer. Databricks notebooks for each data source contains the code for:

Read Parquet Data: Read the Parquet data from the Raw layer using Databricks.

Data Transformation: Perform data cleaning, denormalization, analysis, and any other necessary transformations in Databricks notebooks.

Write to Azure SQL: Store the transformed data in Azure SQL Database on the Transformation layer.

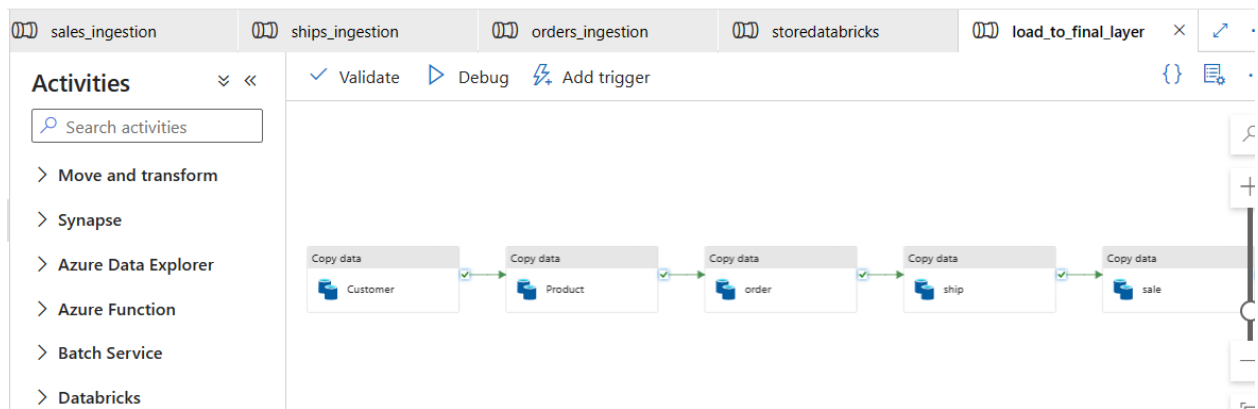


3. Azure SQL Database:

Create tables in Azure SQL Database to store the transformed data. Separate tables for each data source (order, ship, sale, product, customer) are created and data is stored using Azure Databricks.

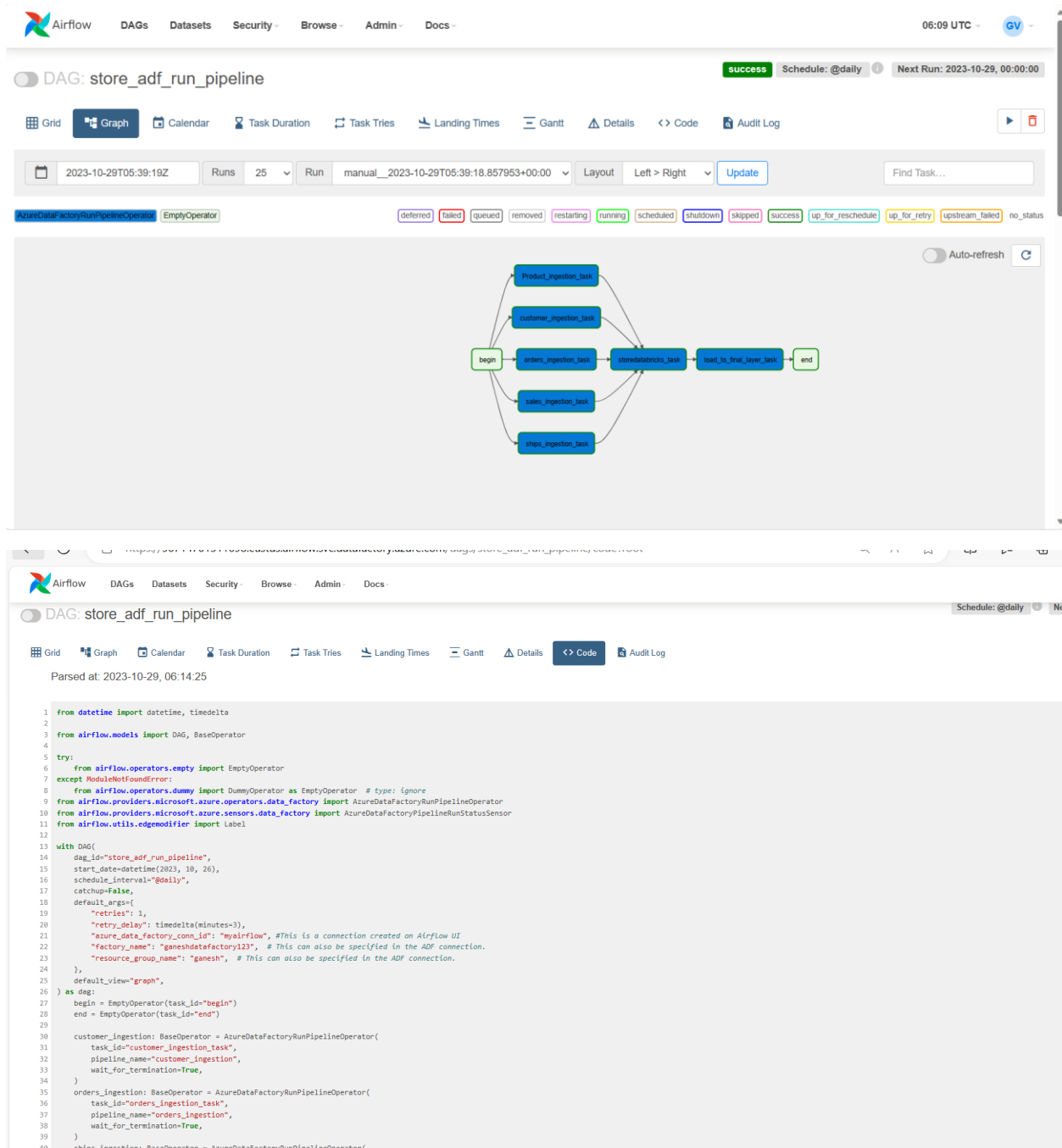
4. Final Layer Data Integration:

Create a final layer in Azure SQL Database and Azure Databricks where we join different tables and load the integrated data.



5. Apache Airflow Orchestration:

Orchestrate the entire ETL process using Apache Airflow. Create a DAG (Directed Acyclic Graph) that defines the order and dependencies of the tasks.



The screenshot displays the Apache Airflow web interface for a DAG named 'store_adf_run_pipeline'. The interface includes a top navigation bar with links for Airflow, DAGs, Datasets, Security, Browse, Admin, and Docs. The DAG is currently in a 'success' state, with a schedule of '@daily' and a next run time of 2023-10-29, 00:00:00. Below the navigation bar, there are tabs for Grid, Graph, Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, and Audit Log. The 'Graph' tab is selected, showing a DAG diagram with the following tasks: 'begin', 'customer_ingestion_task', 'orders_ingestion_task', 'sales_ingestion_task', 'ships_ingestion_task', 'store_adf_run_pipeline_operator', 'load_to_third_layer_task', and 'end'. The tasks are connected in a sequence, with 'begin' leading to the ingestion tasks, which then lead to the 'store_adf_run_pipeline_operator' task, followed by 'load_to_third_layer_task' and 'end'. The interface also includes a search bar, a 'Find Task...' input, and a 'Layout' dropdown set to 'Left > Right'. The 'Update' button is visible. Below the DAG diagram, there is a legend for task statuses: deferred, failed, queued, removed, restarting, running, scheduled, shutdown, skipped, success, up_for_reschedule, up_for_retry, upstream_failed, and no_status. The 'Auto-refresh' toggle is also present.

```
1 from datetime import datetime, timedelta
2
3 from airflow.models import DAG, BaseOperator
4
5 try:
6     from airflow.operators.empty import EmptyOperator
7 except ModuleNotFoundError:
8     from airflow.operators.dummy import DummyOperator as EmptyOperator # type: ignore
9
10 from airflow.providers.microsoft.azure.operators.data_factory import AzureDataFactoryRunPipelineOperator
11 from airflow.providers.microsoft.azure.sensors.data_factory import AzureDataFactoryPipelineRunStatusSensor
12 from airflow.utils.edgemodifier import Label
13
14 with DAG(
15     dag_id="store_adf_run_pipeline",
16     start_date=datetime(2023, 10, 26),
17     schedule_interval="@daily",
18     catchup=False,
19     default_args={
20         "retries": 1,
21         "retry_delay": timedelta(minutes=3),
22         "azure_data_factory_conn_id": "myairflow", # This is a connection created on Airflow UI
23         "factory_name": "ganeshtdatafactory123", # This can also be specified in the ADF connection.
24         "resource_group_name": "ganesht", # This can also be specified in the ADF connection.
25     },
26     default_view="graph",
27 ) as dag:
28     begin = EmptyOperator(task_id="begin")
29     end = EmptyOperator(task_id="end")
30
31     customer_ingestion = BaseOperator = AzureDataFactoryRunPipelineOperator(
32         task_id="customer_ingestion_task",
33         pipeline_name="customer_ingestion",
34         wait_for_termination=True,
35     )
36
37     orders_ingestion = BaseOperator = AzureDataFactoryRunPipelineOperator(
38         task_id="orders_ingestion_task",
39         pipeline_name="orders_ingestion",
40         wait_for_termination=True,
41     )
42
43     ships_ingestion = BaseOperator = AzureDataFactoryRunPipelineOperator(
44         task_id="ships_ingestion_task",
45         pipeline_name="ships_ingestion",
46         wait_for_termination=True,
47     )
48
49     store_adf_run_pipeline_operator = AzureDataFactoryRunPipelineOperator(
50         task_id="store_adf_run_pipeline_operator",
51         pipeline_name="store_adf_run_pipeline_operator",
52         wait_for_termination=True,
53     )
54
55     load_to_third_layer_task = BaseOperator = AzureDataFactoryRunPipelineOperator(
56         task_id="load_to_third_layer_task",
57         pipeline_name="load_to_third_layer_task",
58         wait_for_termination=True,
59     )
60
61     begin.set_downstream(customer_ingestion)
62     begin.set_downstream(orders_ingestion)
63     begin.set_downstream(ships_ingestion)
64     customer_ingestion.set_downstream(store_adf_run_pipeline_operator)
65     orders_ingestion.set_downstream(store_adf_run_pipeline_operator)
66     ships_ingestion.set_downstream(store_adf_run_pipeline_operator)
67     store_adf_run_pipeline_operator.set_downstream(load_to_third_layer_task)
68     load_to_third_layer_task.set_downstream(end)
```