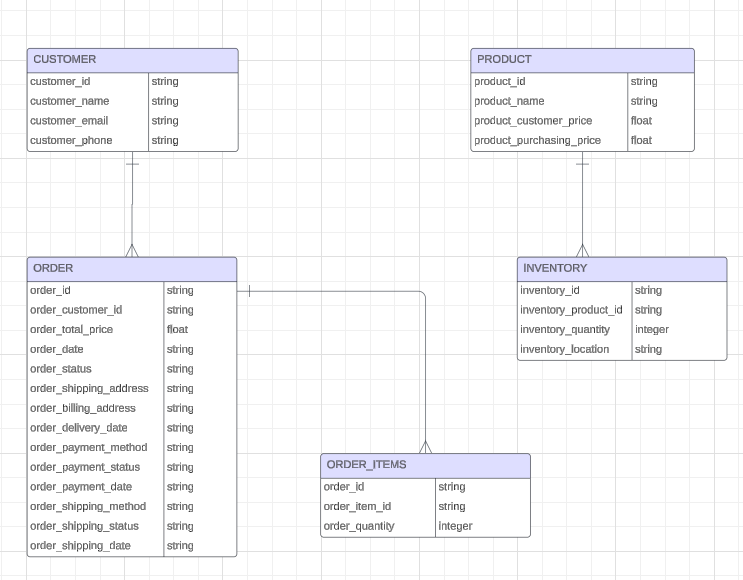
CODING ASSIGNMENT – ECOMMERCE SHOP DATA WAREHOUSE

For this assignment below are the process details followed to achieve the end goal. The technologies used to complete the assignment are, Azure BLOB Storage for file storage, Azure Synapse to develop ETL pipeline to load data from files to tables and Microsoft SQL Server to store data into tables and use for further analysis.

**STEP 1: Creating a Data Model**

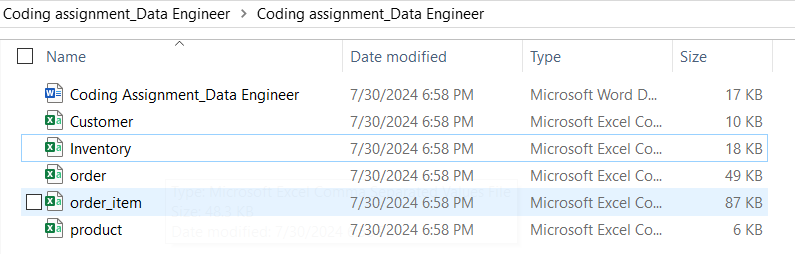
The data shared was downloaded and analyzed, based on which the data model was created. Below is the relationship and association between the tables:

* CUSTOMER 🡪 ORDER: There is one-to-many relationship between these two tables. These tables can be joined using the customer\_id to order\_customer\_id.
* ORDER 🡪 ORDER\_ITEMS: There is one-to-many relationship between these two tables. These tables can be joined using the order\_id to order\_id.
* PRODUCT 🡪 INVENTORY: There is many-to-one relationship between these two tables. These tables can be joined using the product\_id to inventory\_product\_id.

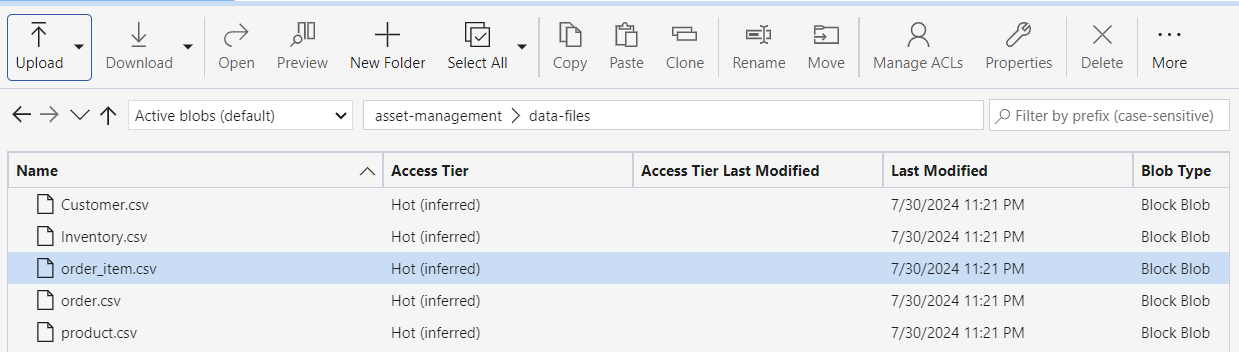


**STEP 2: Download the zip file**

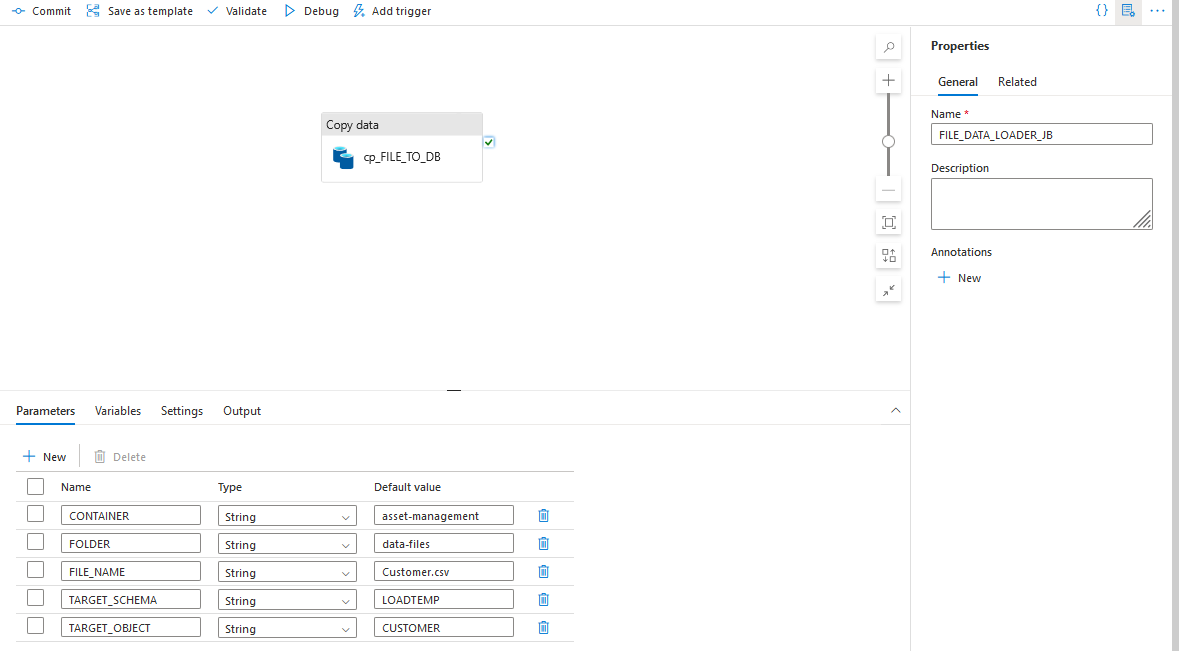
* The zip file shared for the assignment was downloaded locally and contents extracted into the respective folder



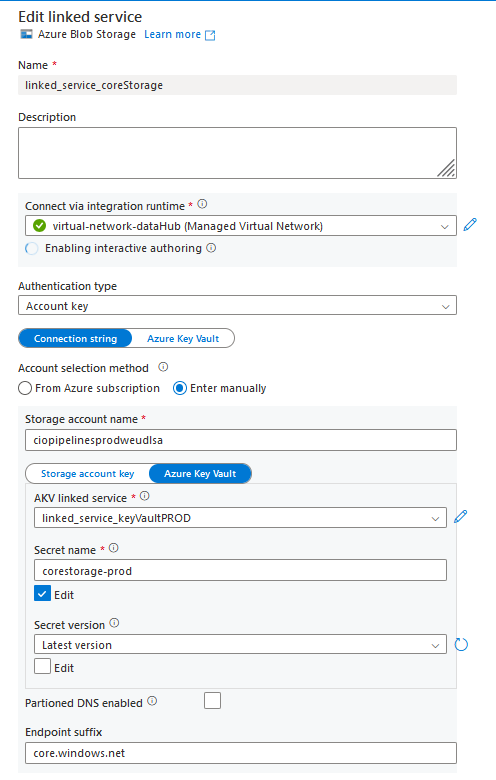
* These files were then uploaded to the BLOB storage

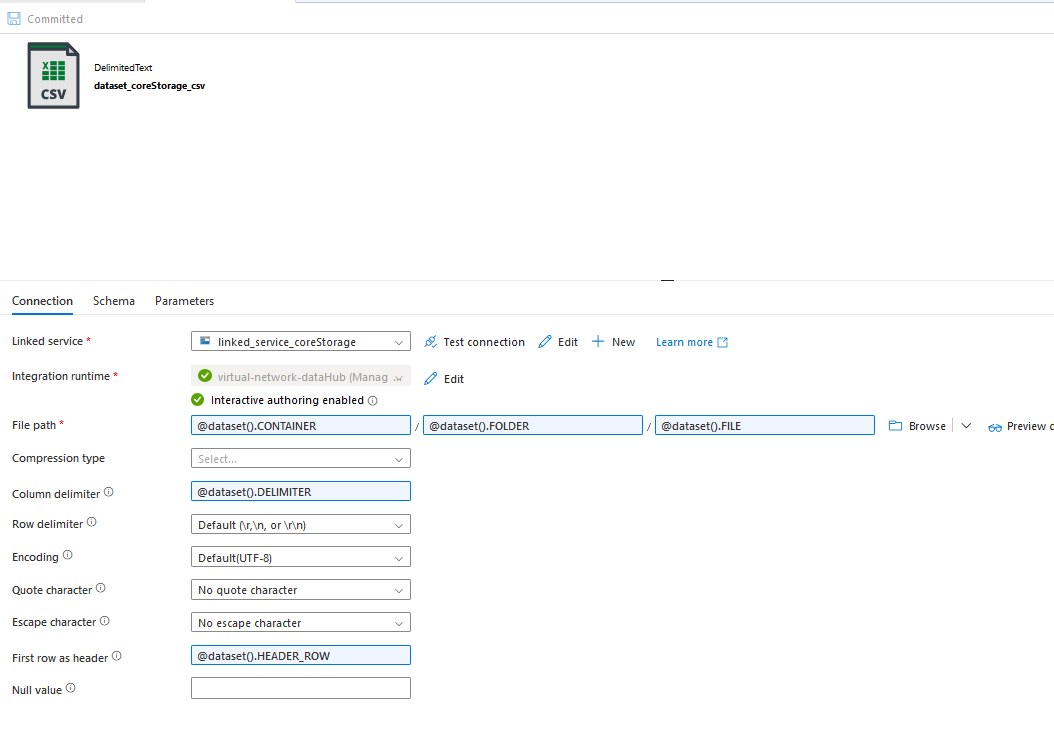


* The ETL pipeline ***FILE\_DATA\_LOADER\_JB*** was created in Azure Synapse using the COPY stage.

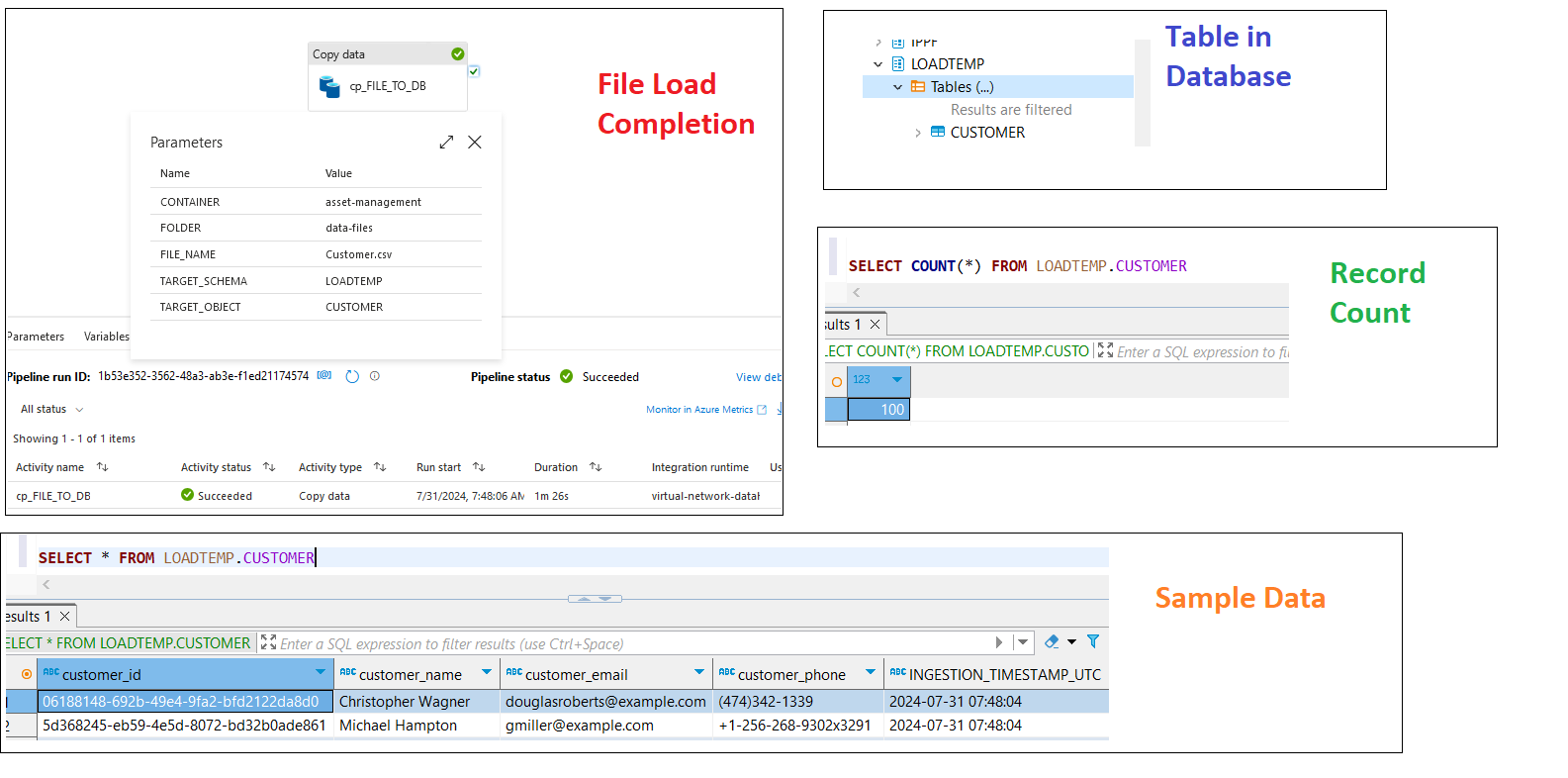


* The COPY stage involves SOURCE and SINK. In our case **SOURCE** was BLOB storage, for which **linked service** was developed to be able to connect to BLOB and associated **Integration dataset** was created to read the CSV file. Below are details of these.

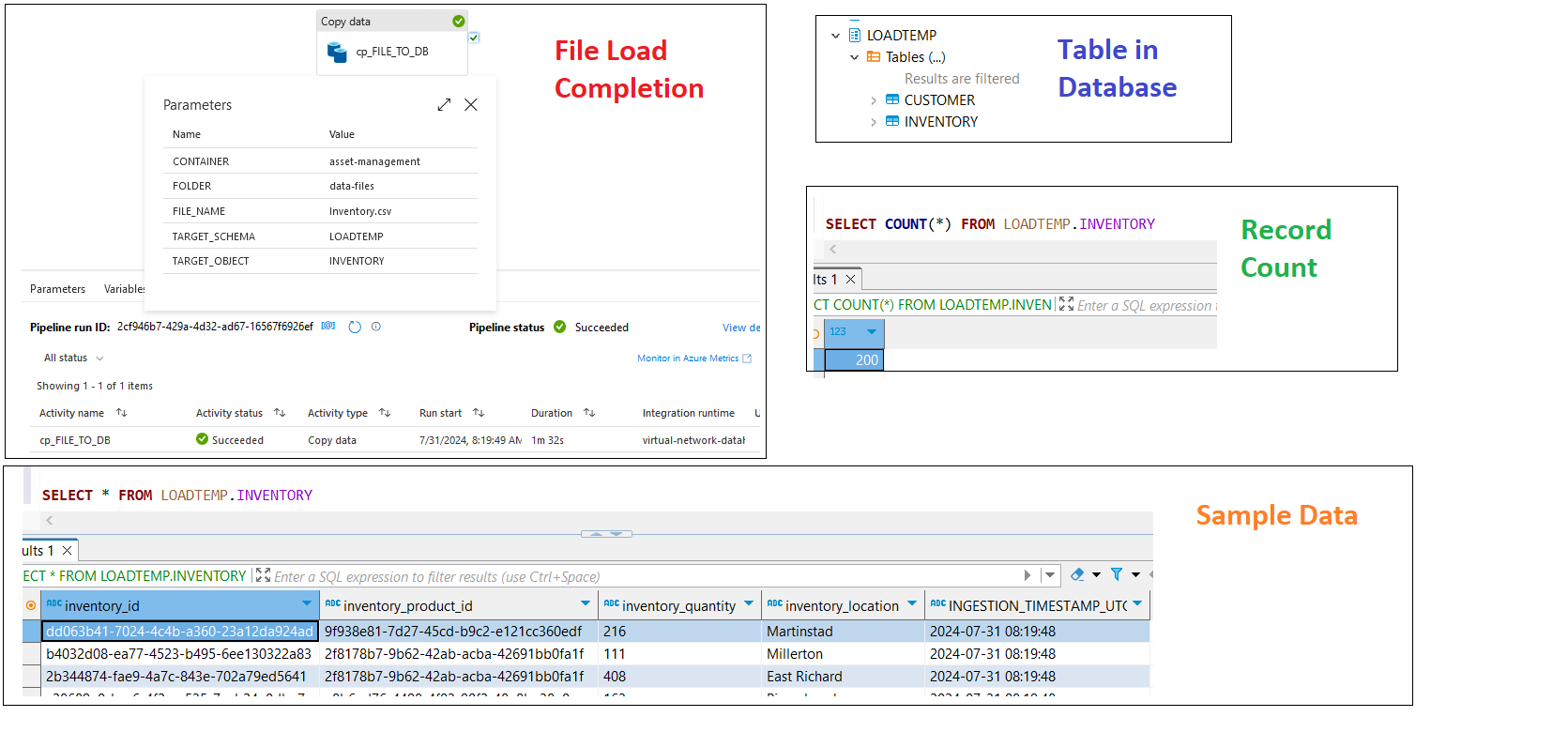




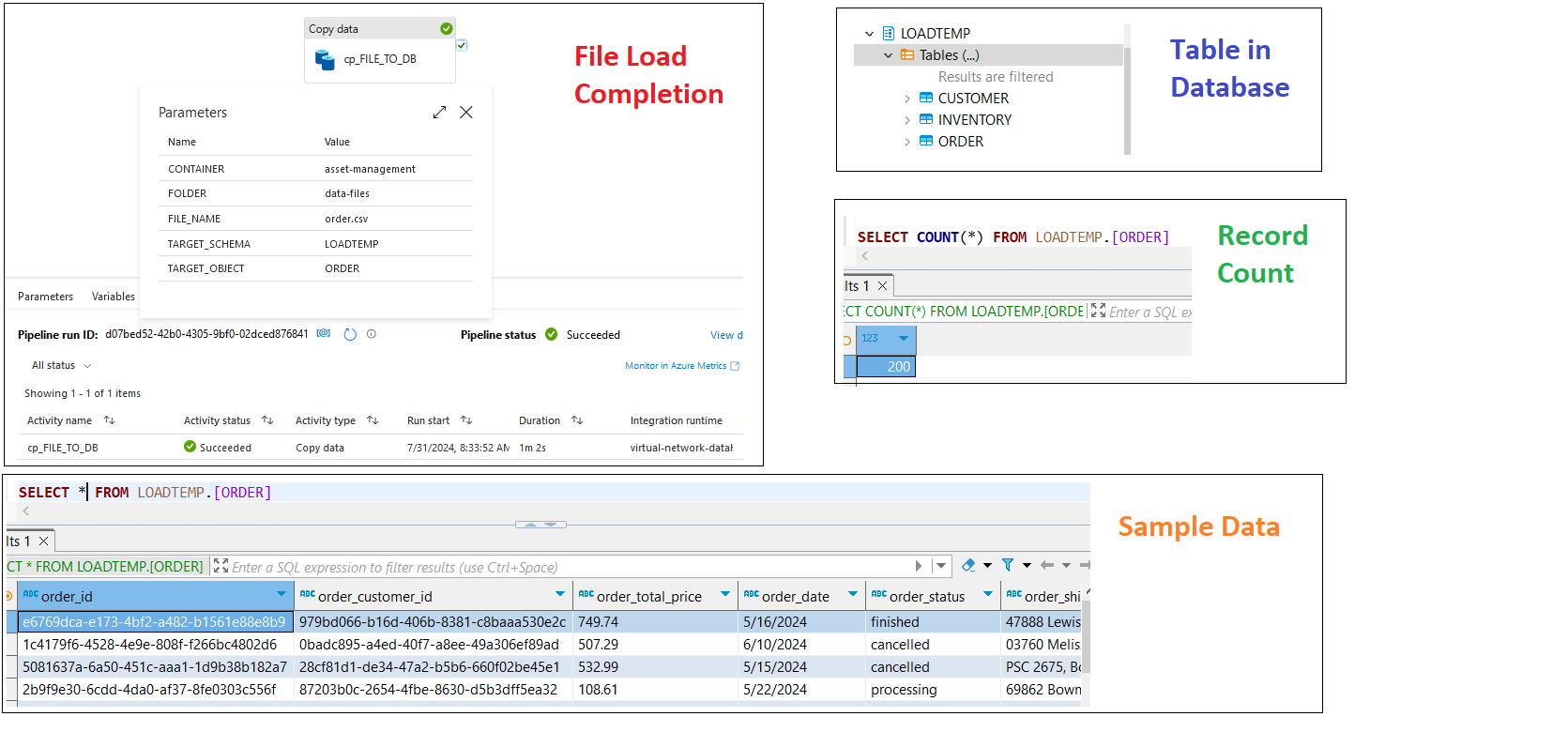
* Loading data for **CUSTOMER**. The data for CUSTOMER was successfully loaded using the ETL pipeline. The data was ingested to **LOADTEMP** schema on SQL Server. A total of 100 records were fetched with sample records displayed.



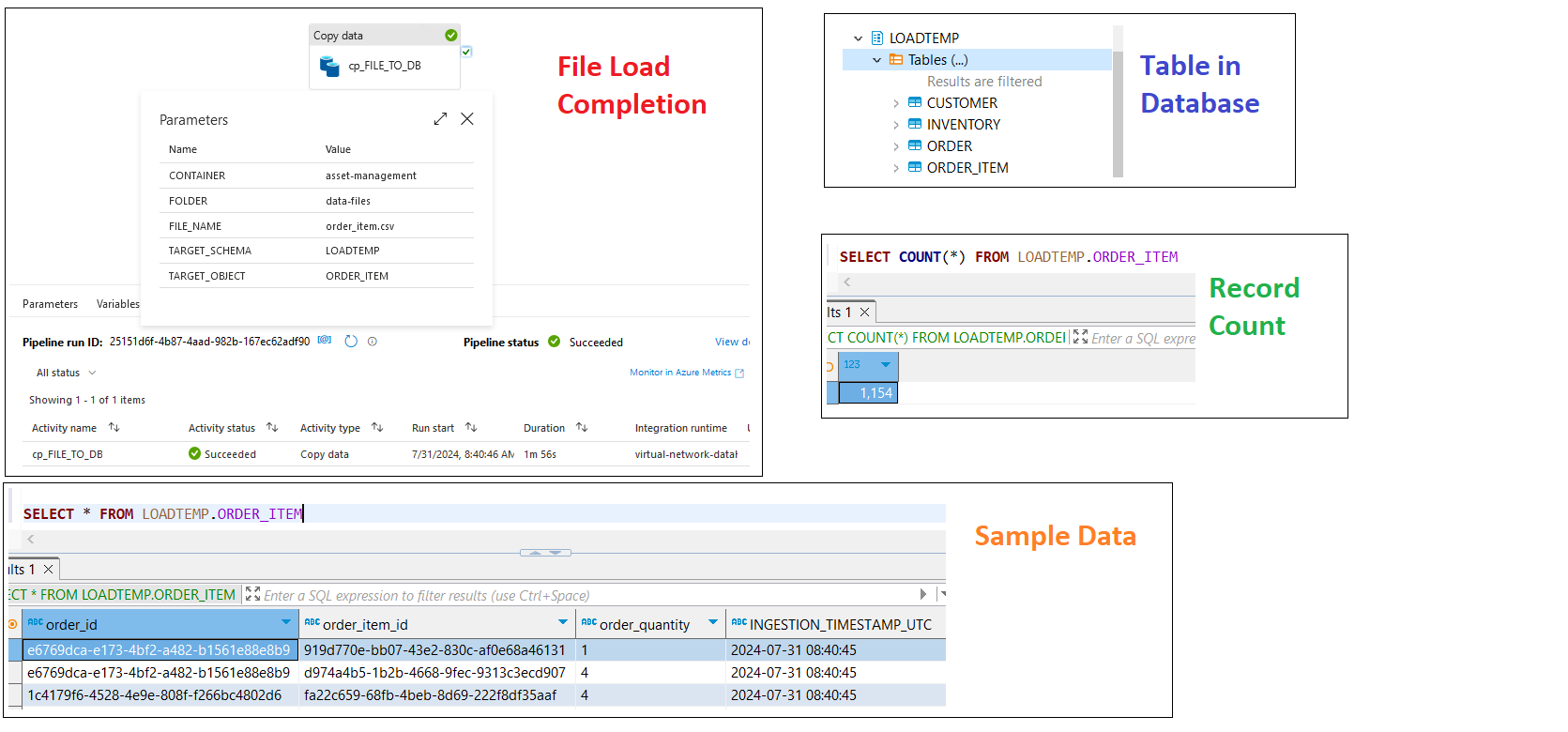
* Loading data for **INVENTORY**. The data for INVENTORY was successfully loaded using the ETL pipeline. The data was ingested to **LOADTEMP** schema on SQL Server. A total of 200 records were fetched with sample records displayed.



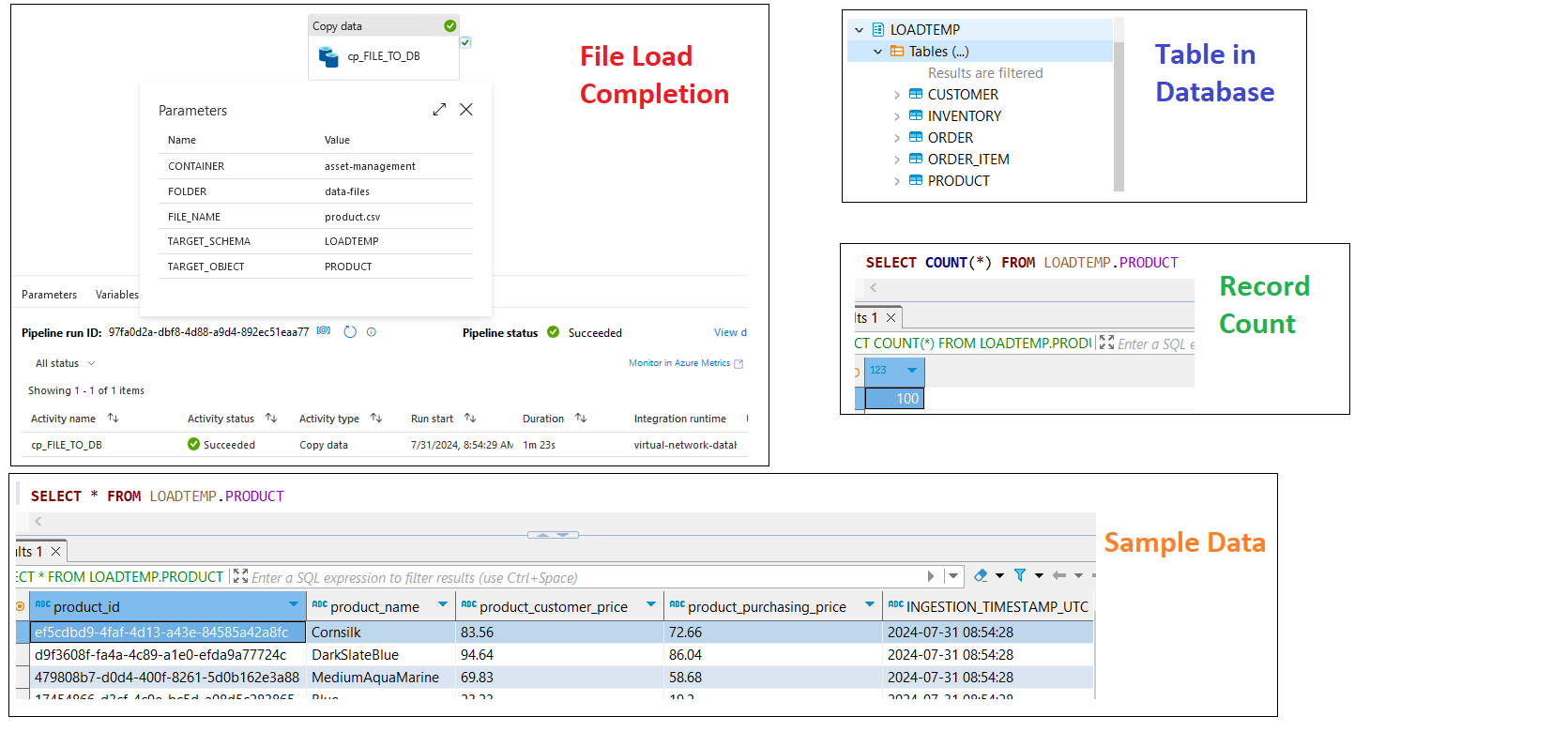
* Loading data for **ORDER**. The data for ORDER was successfully loaded using the ETL pipeline. The data was ingested to **LOADTEMP** schema on SQL Server. A total of 200 records fetched with sample records displayed.



* Loading data for **ORDER\_ITEM**. The data for ORDER\_ITEM was successfully loaded using the ETL pipeline. The data was ingested to **LOADTEMP** schema on SQL Server. A total of 1154 records were fetched with sample records displayed.



* Loading data for **PRODUCT**. The data for PRODUCT was successfully loaded using the ETL pipeline. The data was ingested to **LOADTEMP** schema on SQL Server. A total of 100 records fetched with sample records displayed.



**STEP 3: Revenue Per Customer**

Below is the SQL script to compute revenue per customer. The output is as displayed.

**SELECT** *CUS*.customer\_id, *CUS*.customer\_name, *CUS*.customer\_email, *CUS*.customer\_phone,

SUM(CAST(*ORI*.order\_quantity AS INT) \* CAST(*ORD*.order\_total\_price AS DECIMAL(10,2))) AS *total\_revenue*

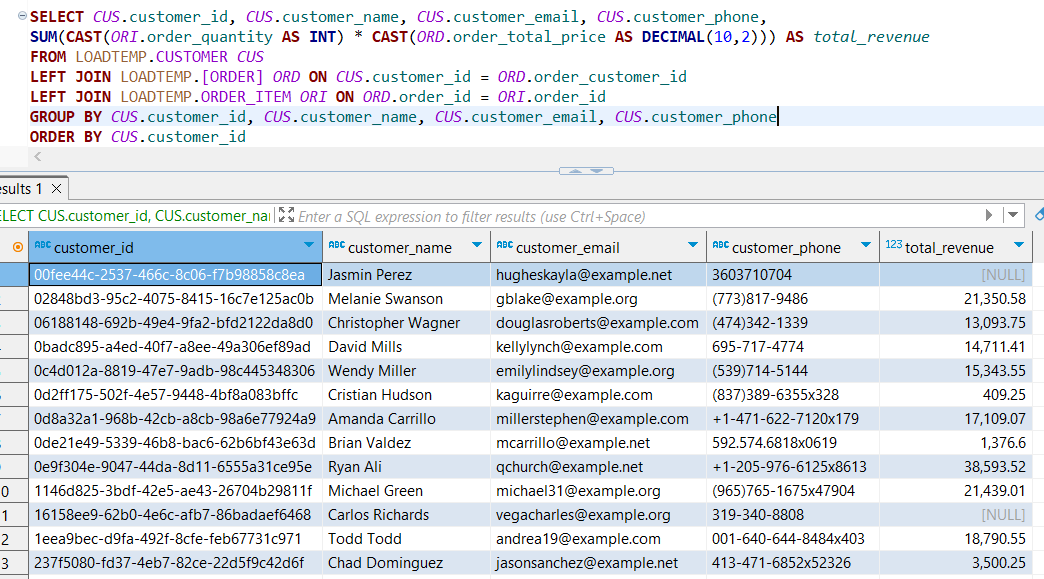
**FROM** LOADTEMP.CUSTOMER *CUS*

**LEFT** JOIN LOADTEMP.[ORDER] *ORD* **ON** *CUS*.customer\_id = *ORD*.order\_customer\_id

**LEFT JOIN** LOADTEMP.ORDER\_ITEM *ORI* **ON** *ORD*.order\_id = *ORI*.order\_id

**GROUP BY** *CUS*.customer\_id, *CUS*.customer\_name, *CUS*.customer\_email, *CUS*.customer\_phone

**ORDER BY** *CUS*.customer\_id



**STEP 4: Inventory Value Per Location**

Below is the SQL script to compute goods value per inventory location. The output is as displayed.

**SELECT** *INV*.inventory\_location,

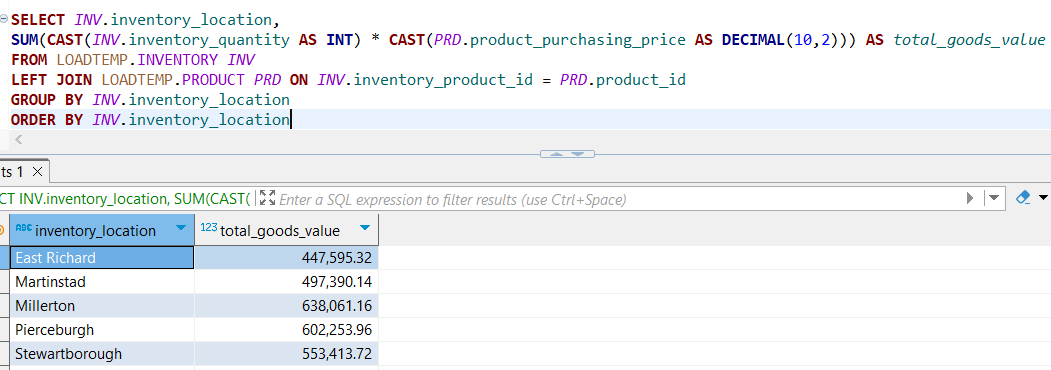
**SUM**(**CAST**(*INV*.inventory\_quantity **AS** **INT**) \* **CAST**(*PRD*.product\_purchasing\_price **AS** **DECIMAL**(10,2))) **AS** *total\_goods\_value*

**FROM** LOADTEMP.INVENTORY *INV*

**LEFT** **JOIN** LOADTEMP.PRODUCT *PRD* **ON** *INV*.inventory\_product\_id = *PRD*.product\_id

**GROUP** **BY** *INV*.inventory\_location

**ORDER** **BY** *INV*.inventory\_location



**STEP 5: Average Order Value Per Customer**

Below is the SQL script to compute average order value per customer. The output is as displayed.

**SELECT** **CAST**(**AVG**(**CASE** **WHEN** *total\_revenue* **IS** **NULL** **THEN** 0 **ELSE** *total\_revenue* **END**) **AS** **DECIMAL**(10,2)) **AS** *avg\_revenue\_per\_customer*

**FROM** (

**SELECT** *CUS*.customer\_id, *CUS*.customer\_name, *CUS*.customer\_email, *CUS*.customer\_phone,

**SUM**(**CAST**(*ORI*.order\_quantity **AS** **INT**) \* **CAST**(*ORD*.order\_total\_price **AS** **DECIMAL**(10,2))) **AS** *total\_revenue*

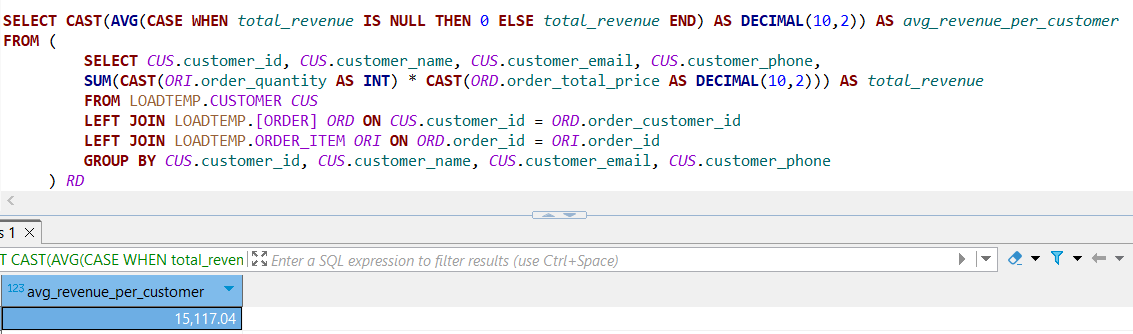
**FROM** LOADTEMP.CUSTOMER *CUS*

**LEFT** **JOIN** LOADTEMP.[ORDER] *ORD* **ON** *CUS*.customer\_id = *ORD*.order\_customer\_id

**LEFT** **JOIN** LOADTEMP.ORDER\_ITEM *ORI* **ON** *ORD*.order\_id = *ORI*.order\_id

**GROUP** **BY** *CUS*.customer\_id, *CUS*.customer\_name, *CUS*.customer\_email, *CUS*.customer\_phone

) *RD*



**STEP 6: Thoughts to Improve Datawarehouse Usage**

Below are few of the aspects to consider:

* **Security & Administration:** Limit database access to only required users. This will ease the administration of access management, improve data security and improve database performance. This can be achieved by implementing strategies as:
  + Restricting the access to technical users vs. individual users in scenario’s where data is to be consumed by analytical tools, reporting purpose or inputs to dashboards.
  + Creating and granting access based on project / project teams.
  + Giving access to the right environment DEV/TEST/PROD basis the requirement, avoiding unnecessary traffic.
  + Avoiding full-control access where not required.
* **Data Historization:** Data has become more important than ever and each day it adds value. In today’s world data availability is less of an issue than the amount of data available. This brings us to the challenges where table sizes are growing exponentially with time, making it difficult to access and analyze data. This demands a solution that would better manage data accessibility.This can be achieved by implementing strategies as:
  + Creating historization tables and moving the data out of master tables at required intervals. This will help maintain the size of the master table and in turn better accessibility. The data management strategies can vary depending on end user data consumption requirement. The history tables can also be used for further consumption if required for machine learning projects or such others.
  + Archiving data sources no longer active. In scenarios where data is loaded from a source which is sunset, it is better to archive the entire database schema, achieving increased database space availability and optimizing database usage.
* **Data Quality & Data Transformation:**  Developing efficient ETL pipelines to improve data quality. This can be achieved with:
  + Optimizing data volumes by restricting the data load to required fields.
  + Optimizing data volumes by using filters to restrict data load to vital data.
  + Optimizing data volumes by using distinct to avoid data duplication.
  + Using data transformation functions to convert the source data as required.
* **Delta Data Load:** Developing efficient ETL pipelines to optimize the data load. This can be achieved with:
  + Restricting the data load to incremental data where possible, reducing the overall ETL cycle time. This will improve access availability and query processing.
  + Using UPSERT in possible situations to improve the overall data processing.
* **Data Aggregation:** Developing views and/or physical tables with data aggregation where the data volume is huge. This will help:
  + Improve performance of data accessibility.
  + Optimize query execution time.
  + Overcome timeout errors.
  + Efficient performance of associated tools.
* **Data Monitoring:** Developing a robust monitoring system is the key to maintaining the health of the database. This can be achieved with:
  + Consuming process logs if available from the tool and creating efficient dashboards for monitoring the status of the end-to-end data flow.
  + Creating custom monitoring logs where required and developing analytics to derive trends.
  + Developing efficient notification system to alert the responsible person/team in event of failures.
* **Automation:** Developing solutions that will help automate manual process and optimize the overall process. This can be accomplished with:
  + Implementing solutions for auto-recovery in situations of failures.
  + Scheduling activities.
  + Developing functions vs. using complex computational scripts when repetitive usage.
  + Developing stored procedures for efficient performance where applicable.
  + Parallel vs. Sequential processing where applicable.
  + Efficient utilization of CPU, Memory, Processing nodes.
  + Data externalization.