Transfer data from Azure Blob Storage (Gen2) to SQL database

Data Engineering course - ZoomCamp - DataTalks

The tutorial I will explain how to transfer data from Azure Blob Storage (Gen2) to a SQL database using Data Factory. It involves creating data sets for the CSV file and SQL table, configuring the copy activity, and verifying the successful transfer. Debugging the schema mapping ensures the file matches the table.

Initially, the schema and table are created in Vscode, followed by creating a new folder and pipeline in the Data Factory. New tables are created for the source (CSV file) and the sink (SQL database table).

For the CSV file, the linked service is set as the **Blob Storage**, specifying the file path and delimiter. The Data Factory automatically reads columns due to the file having a header, serving as instructions for reading the data.

The database table is defined as a dataset in Data Factory, with the linked service set as the **SQL Database** and the table name provided. Data Factory reads columns from the actual table, used as instructions for writing the data.

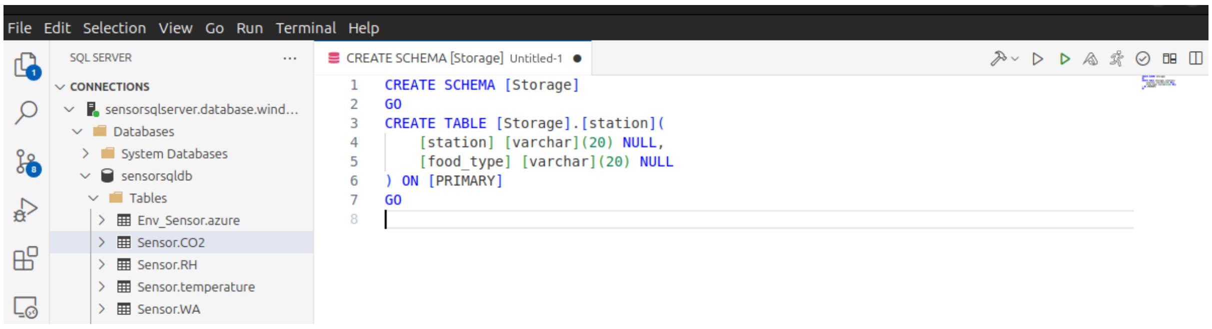
The copy activity is then configured in the pipeline, selecting the source (CSV file) and destination (SQL table). The **Mapping** tab visualises how the source data will be transformed and loaded into the destination database. Data Factory automatically matches columns from the CSV file to the corresponding columns in the database table, however, if the schema does not match you can manually edit the JSON file for schema matching.

After configuration, the pipeline can be validated and debugged to copy the CSV file to the SQL database. The amount of data transferred can be checked by comparing the rows read and written. Finally, the data from the SQL database can be utilised.

Maria Fisher

Steps:

First, we need to create the schema and the tables. To execute this you can connect to Azure Server in Vscode and remotely create a database.

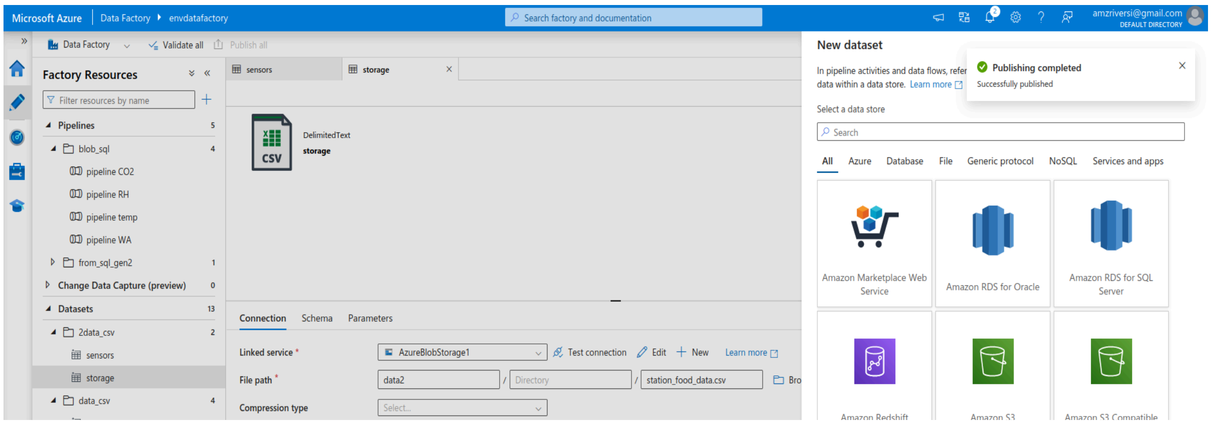


```sql
CREATE SCHEMA [Storage]
GO
CREATE TABLE [Storage].[station](
    [station] [varchar](20) NULL,
    [food_type] [varchar](20) NULL
) ON [PRIMARY]
GO
```

In Data Factory

Generate a fresh directory and subsequently establish a novel pipeline within said directory. Subsequently, generate novel directories, one for the tables that will serve as your source (csv) and another for the tables that will serve as the destination (sink).

Begin by specifying the service linked to the CSV file, which in this case is Blob Storage. Provide the file path and designate the comma as the delimiter. Since the CSV file includes a header, Data Factory can automatically extract the columns. This dataset will serve as a reference for Data Factory to understand the data.

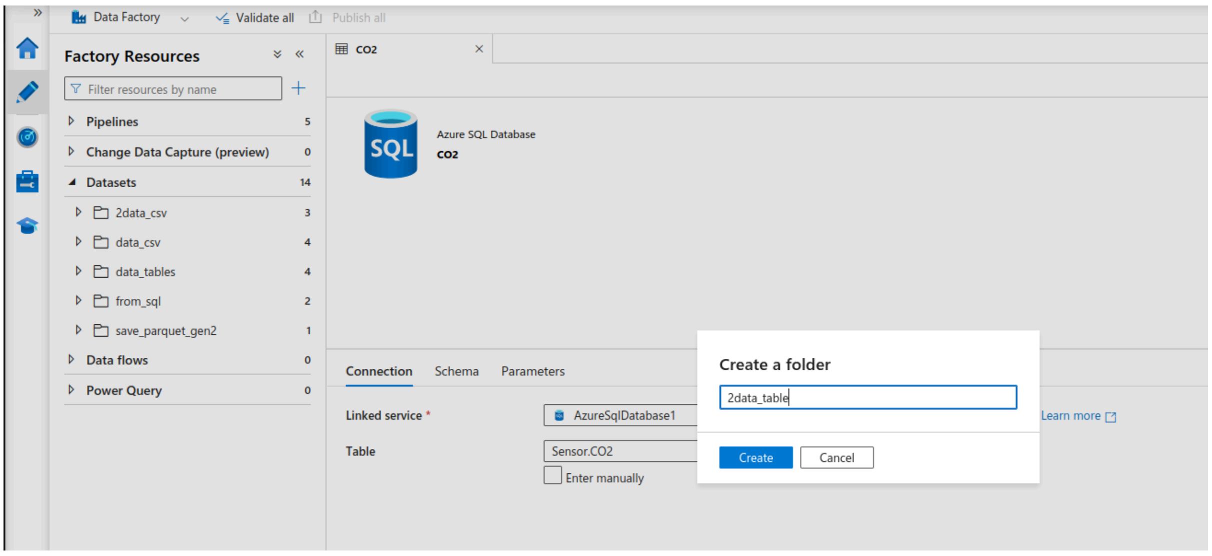


Linked service * AzureBlobStorage1
File path * data2 / Directory / station_food_data.csv
Compression type Select...

In pipeline activities and data flows, refer data within a data store. Learn more

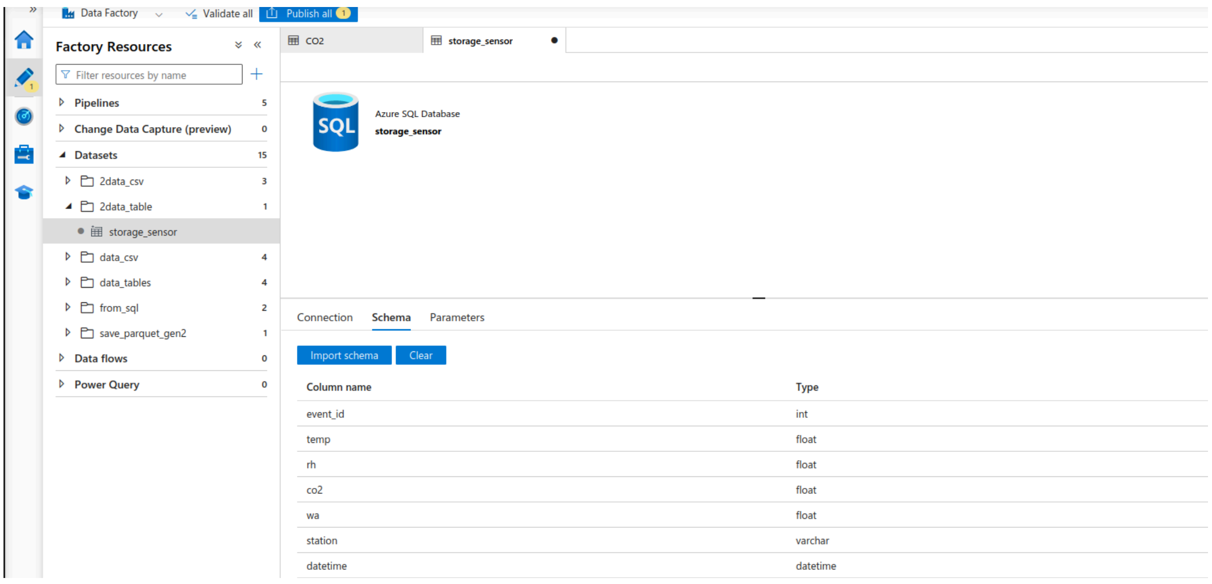
Publishing completed
Successfully published

Next, establish the database table as a dataset within Data Factory. Indicate the linked service as the SQL database and input the table name.



Linked service *	AzureSqlDatabase1
Table	Sensor.CO2
	Enter manually
Create a folder	2data_table

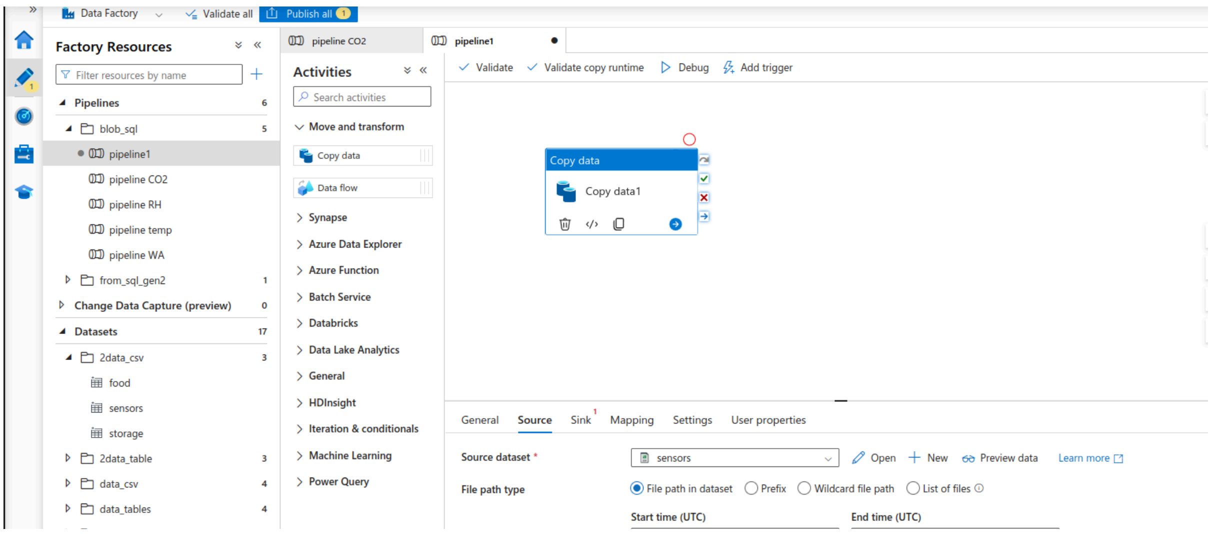
After designating the table name, Data Factory has the capability to extract the columns from the designated table. These columns serve as guidelines for Data Factory to insert the data into the database.



Column name	Type
event_id	int
temp	float
rh	float
co2	float
wa	float
station	varchar
datetime	datetime

Upon finishing these steps, proceed to set up the copy activity within the pipeline. Include the Copy file located in the pipeline Activities tab, and allow Data Factory to manage the copying process.

First select the source (csv files)

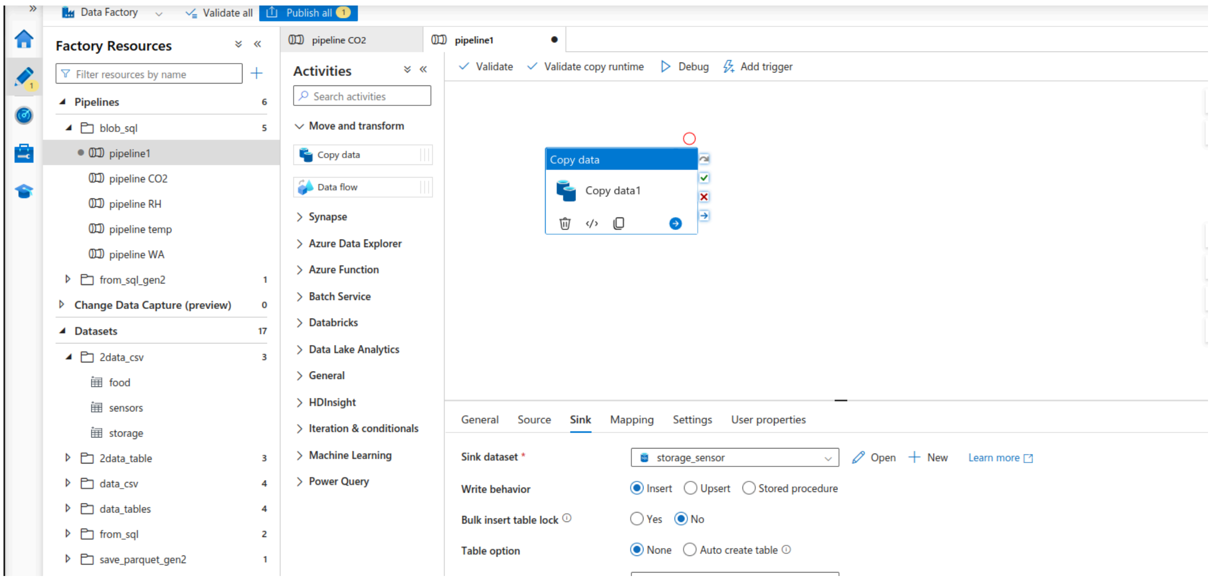


Source dataset * sensors

File path type [✓ File path in dataset] Prefix Wildcard file path List of files

Choose the file destination (sink) where all tables will be saved as csv files in the folder labelled 2data_table. Next, select the table and proceed with the Mapping.

Sink (destination sql database)



Sink dataset * storage_sensor

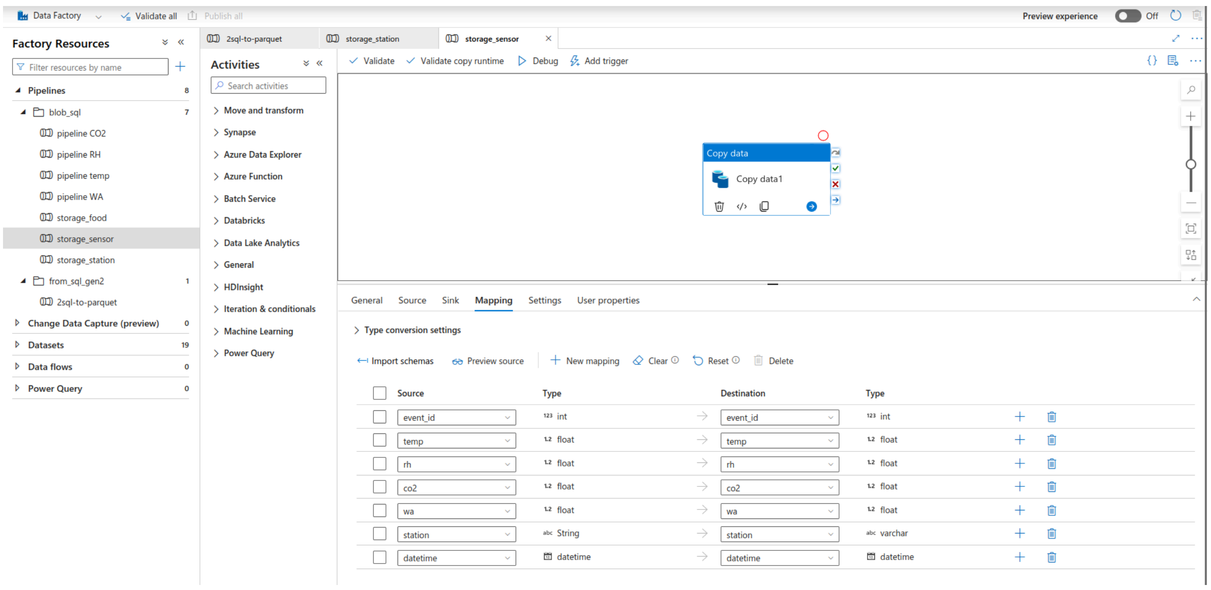
Write behavior [✓ Insert] Upsert Stored procedure

Bulk insert table lock Yes [✓ No]

Table option [✓ None] Auto create table

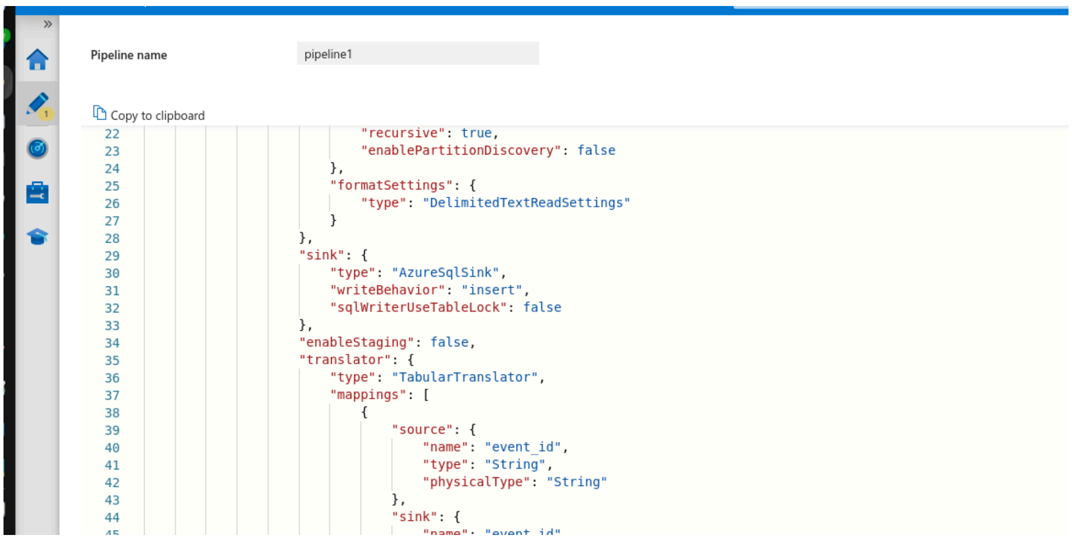
The mapping section provides us with a convenient way to visualise the transformation and loading of our source data into the destination database. By importing the schemas, Data Factory can automatically align the columns from our CSV file with the corresponding columns in our database table. This smooth mapping procedure guarantees the precise transfer of our data without the need for any manual intervention.

If the schema does not match, you have the option to manually edit the json file. The json file can be located by clicking on the icon (**{ }**) →



Source	Type	Destination	Type
event_id	int	event_id	int
temp	float	temp	float
rh	float	rh	float
co2	float	co2	float
wa	float	wa	float
station	String	station	varchar
datetime	datetime	datetime	datetime

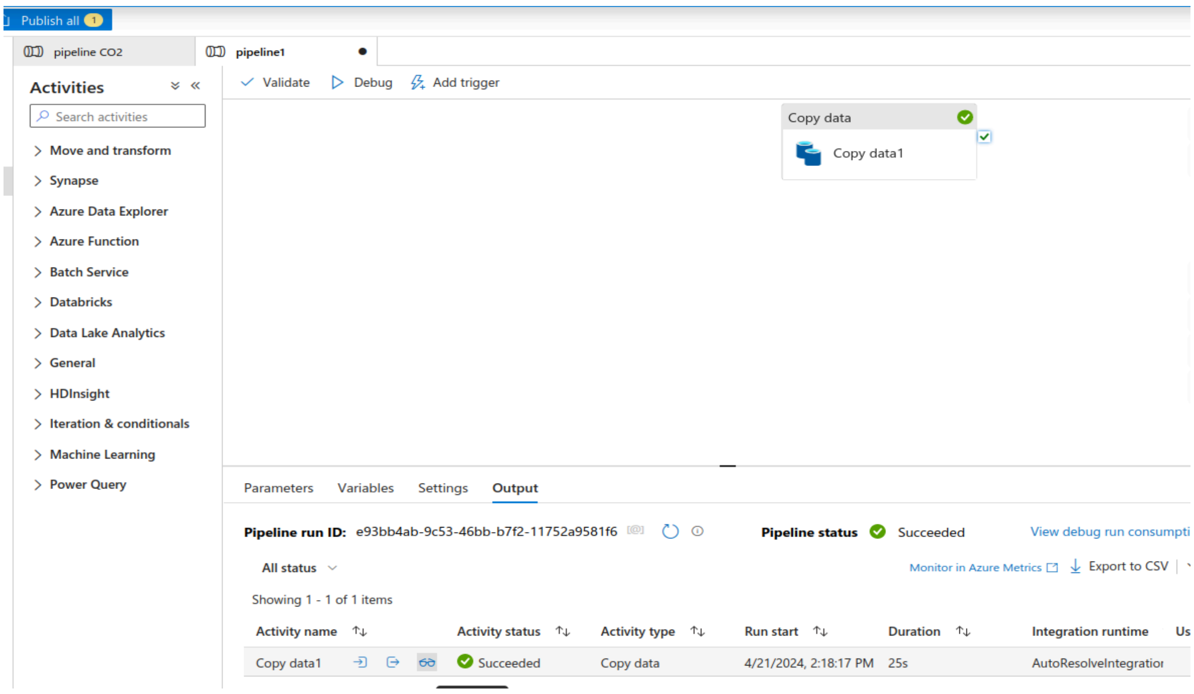
Editing JSON file to match the files schema



```
                "recursive": true,
                "enablePartitionDiscovery": false
            },
            "formatSettings": {
                "type": "DelimitedTextReadSettings"
            }
        },
        "sink": {
            "type": "AzureSqlSink",
            "writeBehavior": "insert",
            "sqlWriterUseTableLock": false
        },
        "enableStaging": false,
        "translator": {
            "type": "TabularTranslator",
            "mappings": [
                {
                    "source": {
                        "name": "event_id",
                        "type": "String",
                        "physicalType": "String"
                    },
                    "sink": {
                        "name": "event_id"
```

Copy to SQL

Once the csv file has been successfully copied to the database, proceed to validate the pipeline. If all checks pass, you may proceed with debugging, which will initiate the data transfer to the SQL database.



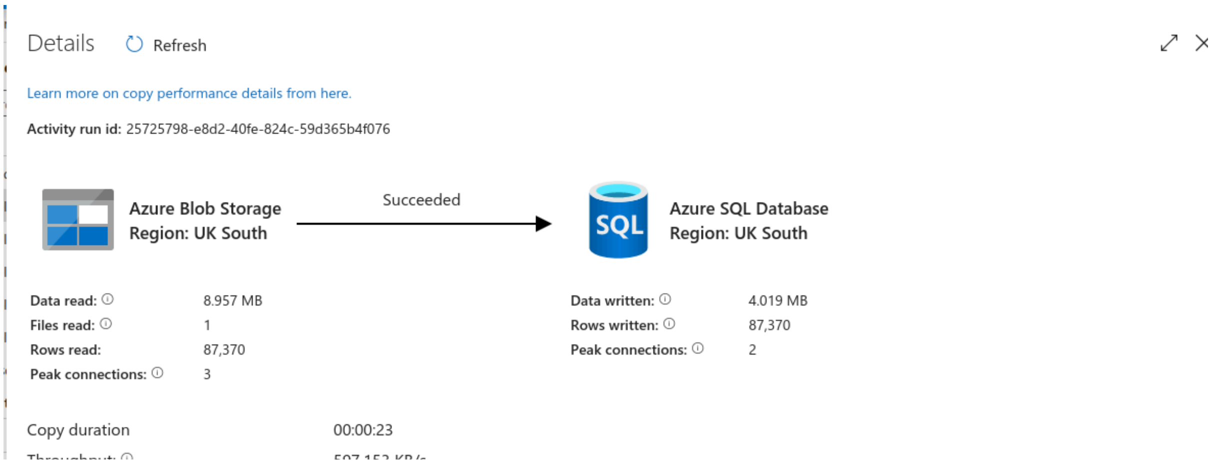
Pipeline run ID: e93bb4ab-9c53-46bb-b7f2-11752a9581f6

Pipeline status ✓ Succeeded

Showing 1 - 1 of 1 items

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
Copy data1	✓ Succeeded	Copy data	4/21/2024, 2:18:17 PM	25s	AutoResolveIntegratior

If the data copying process was successful, you will be able to view the total amount of data transferred, which is indicated by the number of rows read being equal to the number of rows written.



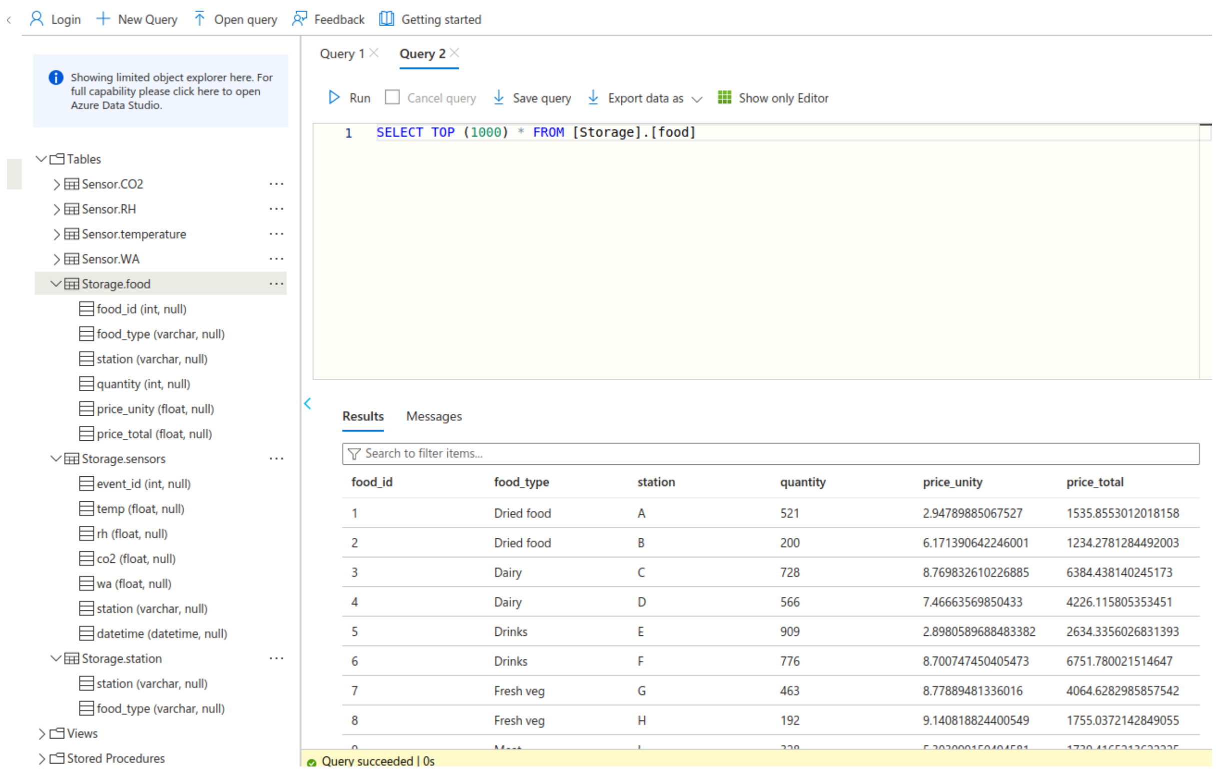
Activity run id: 25725798-e8d2-40fe-824c-59d365b4f076

Data read:	8.957 MB
Files read:	1
Rows read:	87,370
Peak connections:	3

Data written:	4.019 MB
Rows written:	87,370
Peak connections:	2

Copy duration	00:00:23

Now you can use data from SQL databases!



Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

```sql
SELECT TOP (1000) * FROM [Storage].[food]
```

food_id	food_type	station	quantity	price_unity	price_total
1	Dried food	A	521	2.94789885067527	1535.8553012018158
2	Dried food	B	200	6.171390642246001	1234.2781284492003
3	Dairy	C	728	8.769832610226885	6384.438140245173
4	Dairy	D	566	7.46663569850433	4226.115805353451
5	Drinks	E	909	2.8980589688483382	2634.3356026831393
6	Drinks	F	776	8.700747450405473	6751.780021514647
7	Fresh veg	G	463	8.77889481336016	4064.6282985857542
8	Fresh veg	H	192	9.140818824400549	1755.0372142849055

Query succeeded | 0s

Summary

I have provided a detailed explanation of how to transfer data from Azure Blob Storage to a SQL database using Data Factory. This process involves several steps, including creating data sets, configuring the copy activity, and debugging the schema mapping. The data has been successfully transferred and is now ready for further analysis or processing.