# Compute Daily Product Revenue

## Description

This project is an Azure Data Engineering project that calculates total revenue of orders per product based on the given order month (year and month, yyyy-MM).

I used Azure Data Factory, Azure Data Studio, Azure SQL Database, Azure Synapse Analytics, SQL and Azure Blob Storage.

## Dataset

I downloaded the dataset from Durga's (Udemy Lecturer) repository, https://github.com/dgadiraju/data/tree/master/retail\_db, I used the Orders and Order Items folder, the data is comma delimited.

In the above folder I also referred to the create\_db\_tables\_og.sql script for the schema.

## Data Flows

### DFFileFormatConverterOrders

This data flow has two activities.

The first activity is the data source, it reads data from the Orders table that is stored in Azure Blob Storage and sends it to the sink.

The second/last activity receives the data from the incoming stream (Orders), saves the data to OrdersParquet(data set) in Azure Blob storage in Parquet format.

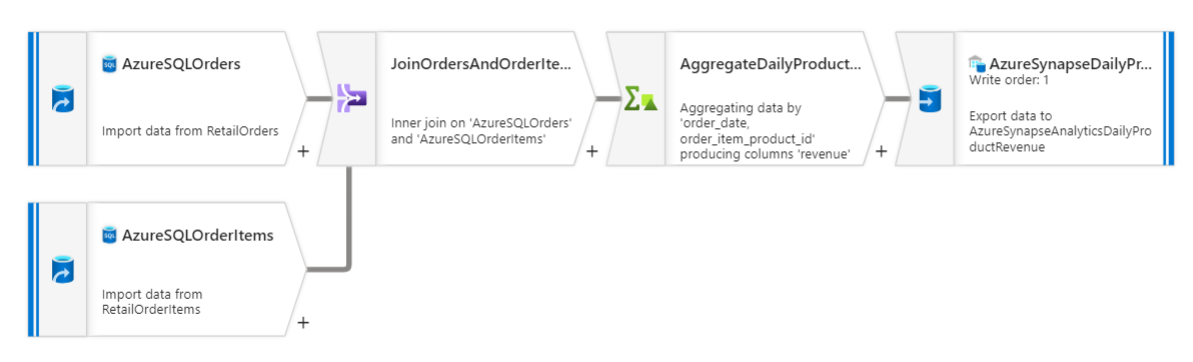
### DFFileFormatConverterOrderItems

This data flow has two activities.

The first activity is the data source, it reads data from the OrderItems table that is stored in Azure Blob Storage and sends it to the sink.

The second/last activity receives the data from the incoming stream (OrderItems), saves the data to OrderItemsParquet(data set) in Azure Blob storage.

### ComputeDailyProductRevenue

This data flow has five activities.

The first two activities are for data sources (Orders and Order Items).

The third activity is for joining the two tables.

The forth one is an aggregate activity that calculates the total revenue of orders per product.

The last activity is the sink which uses an Azure Synapse data set for storing the results in a table.

## Pipeline

### CopyCSVDataToSQLTable

This pipeline has two Copy Data activities.

The first copy activity copies the Orders CSV file to an Orders dataset in Azure SQL Database.

The second copy activity copies the Order Items CSV file to an OrderItems dataset in Azure SQL Database.

### PLFileFormatConverterOrders

This pipeline has one data flow activity.

It is used to execute the DFFileFormatConverterOrders data flow.

### PLFileFormatConverterOrderItems

This pipeline has one data flow activity.

It is used to execute the DFFileFormatConverterOrderItems data flow.

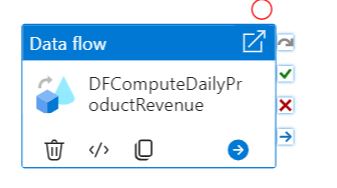
### PLComputeDailyProductRevenueParams

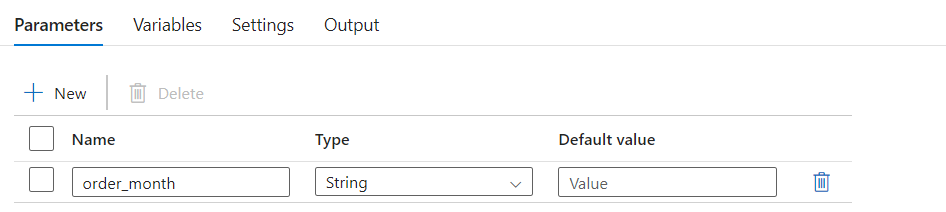
This pipeline has one data flow activity.

It is used to execute the ComputeDailyProductRevenue data flow.

It uses the customized IR(Integrated Runtime) with a time to live of 30min to reduce wait time of compute when executing and testing.

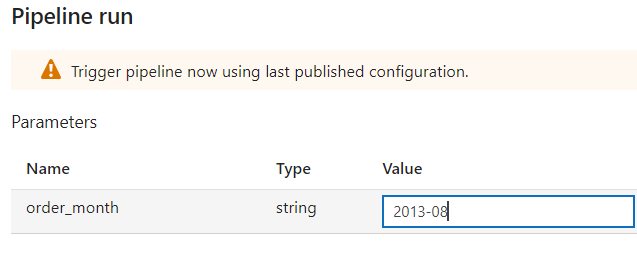
It has one parameter that will be used to specify the order month (Year and month yyyy-MM) that will be used by the data flow to filter the data.

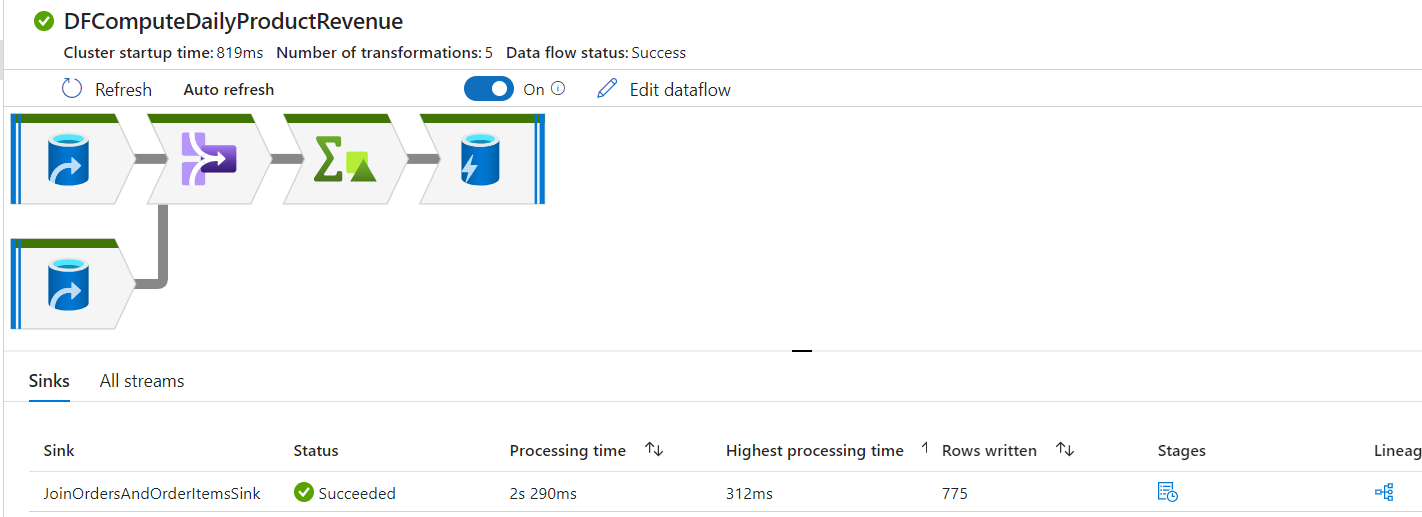




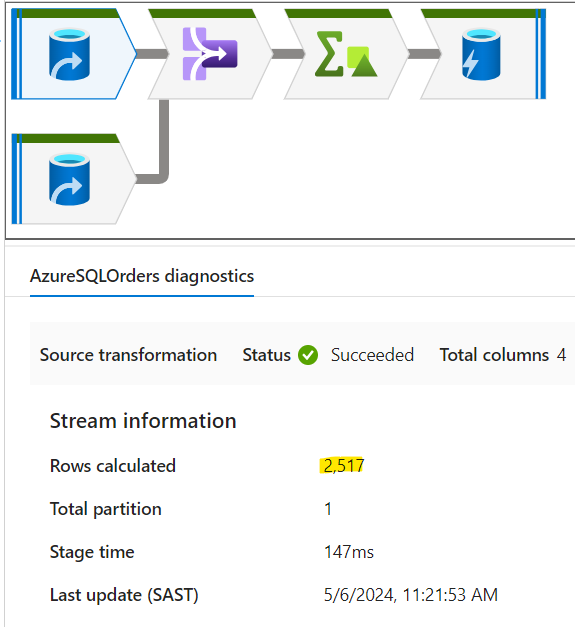
## Test results:

I triggered the PLComputeDailyProductRevenueParams pipeline and used ‘2013-08’ as the order\_month (parameter).





The first activity is the Orders data source, it filters the data by order status to only return orders that have a complete or closed status. Then it filters by order date to only return those that were ordered in August 2013.



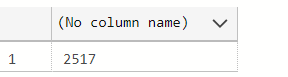
Verified the rows returned by running the below quey in Azure Data Studio:

SELECT COUNT(\*)

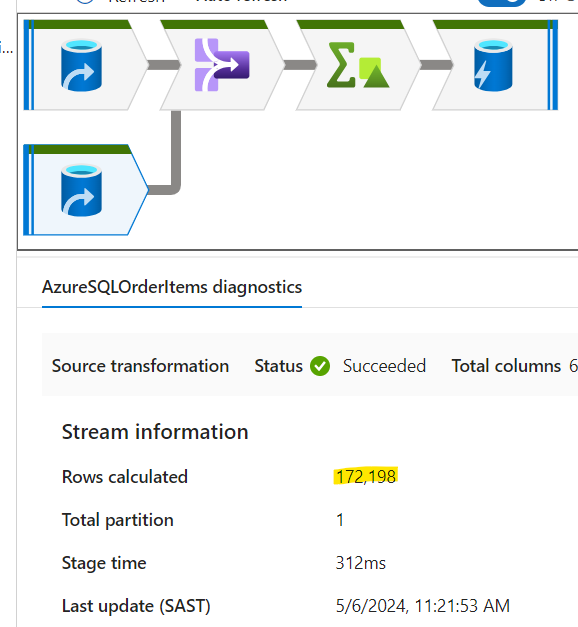
FROM orders AS o

WHERE o.order\_status IN ('COMPLETE', 'CLOSED')

AND o.order\_date LIKE '2013-08%';

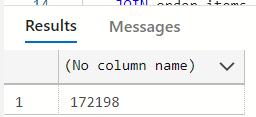


The second activity ran was the Order Items data source, it read all the data in the table.

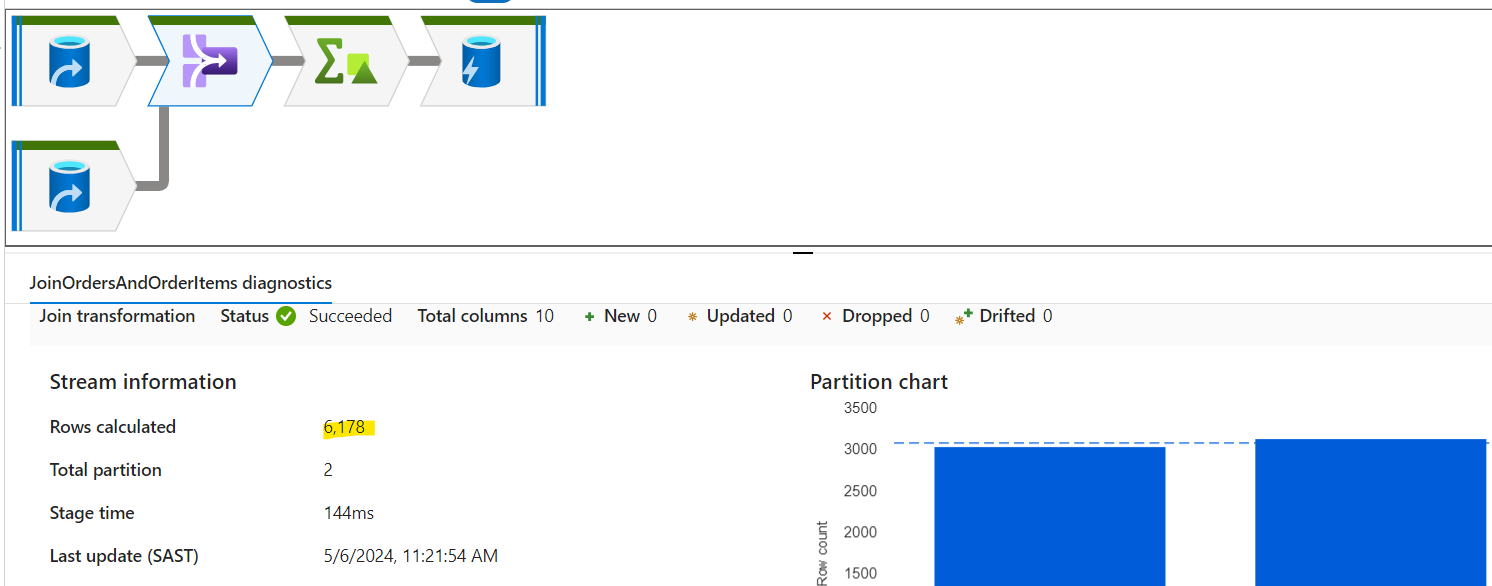


Verified results by running the query below:

SELECT COUNT(\*) FROM order\_items;



The third activity is a join activity that uses inner join to combine both tables. I used 2 partitions on the setup as the number of rows returned is small.

Verified results by running the below query:

SELECT COUNT(\*)

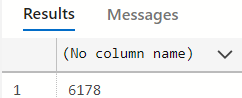
FROM orders AS o

JOIN order\_items AS oi

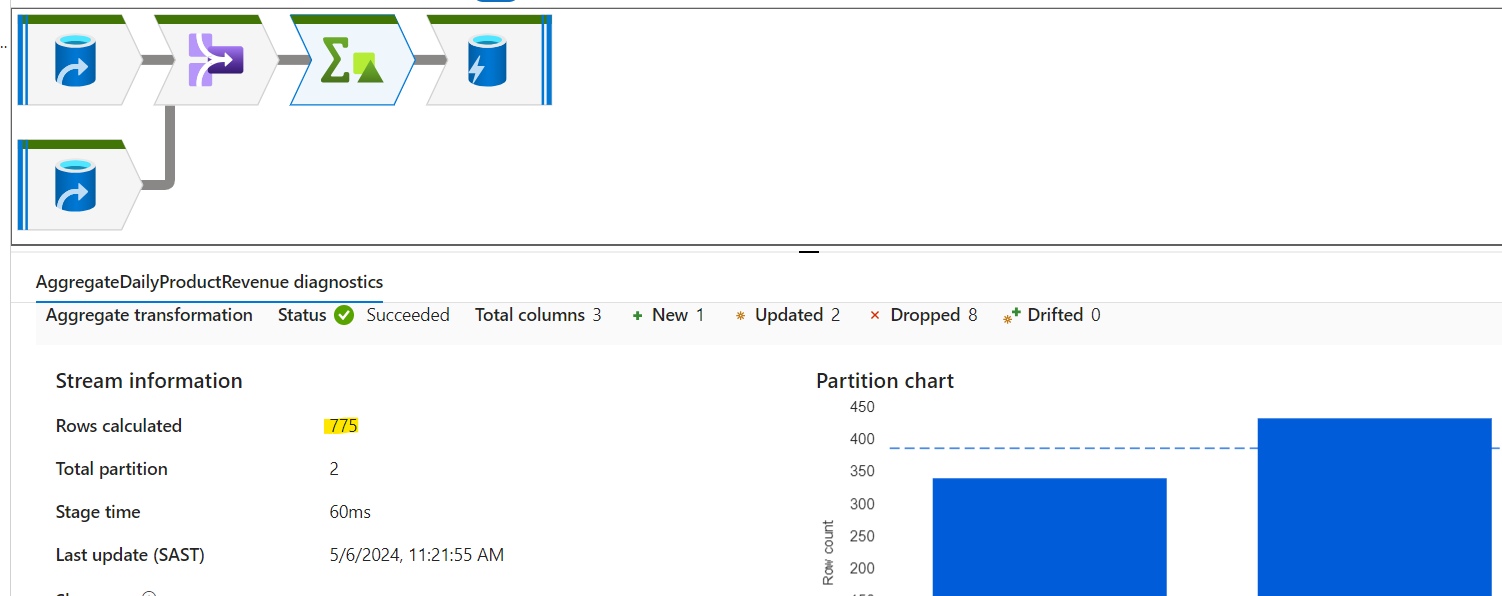
ON o.order\_id = oi.order\_item\_order\_id

WHERE o.order\_status IN ('COMPLETE', 'CLOSED')

AND o.order\_date LIKE '2013-08%';



The forth activity is the aggregate activity that calculates the total sum of the order subtotal and groups the results by order date then by product ID.



Verified rows returned by running below query in Azure Data Studio:

SELECT o.order\_date, oi.order\_item\_product\_id, ROUND(SUM(oi.order\_item\_subtotal), 2) AS revenue

FROM orders AS o

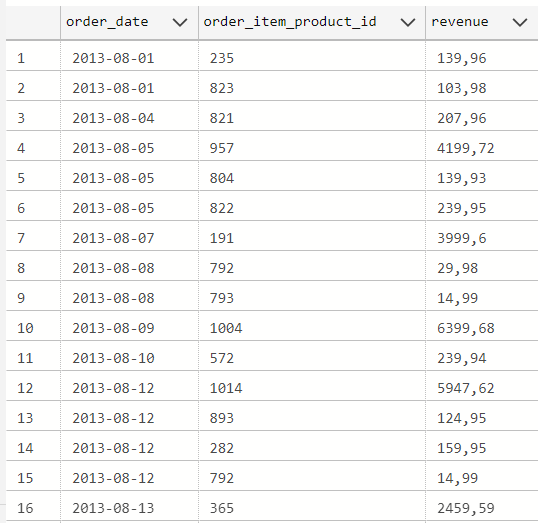
JOIN order\_items AS oi

ON o.order\_id = oi.order\_item\_order\_id

WHERE o.order\_status IN ('COMPLETE', 'CLOSED')

AND o.order\_date LIKE '2013-08%'

GROUP BY o.order\_date, oi.order\_item\_product\_id;





The last activity is the sink activity that stores results in Azure Synapse Dataset

