



Mapping data flow (creating fact table)

Mapping Data Flows in **Azure Data Factory (ADF)** are a visual and data-transformation tool designed to enable scalable and reusable data transformations without requiring extensive code. They are particularly useful for ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform) scenarios. Here's a detailed overview:

Key Features

1. No-Code Visual Design:

- Mapping Data Flows provide a drag-and-drop interface for designing data transformations.
- Users can build data transformation logic without writing code, making it accessible to non-developers.

2. Integration with Azure Data Factory Pipelines:

- Data flows are part of ADF pipelines and can be triggered and monitored alongside other pipeline activities.

3. Scalable Execution:

- Transformations are executed on a managed Spark cluster provided by ADF, allowing for distributed and high-performance processing.

4. Wide Range of Transformation Options:

- **Filter:** Remove rows based on conditions.
- **Derived Columns:** Add or modify columns.
- **Aggregate:** Perform aggregation operations like sum, average, etc.
- **Join:** Combine data from multiple sources.
- **Lookup:** Enrich data by looking up values from another dataset.
- **Pivot and Unpivot:** Reshape data structures.
- **Sort, Split, and Conditional Split:** Organize and divide data.

5. Connectors for Diverse Data Sources:

- Supports integration with various data sources such as Azure Blob Storage, SQL databases, Data Lake Storage, and more.

6. Parameterization:

- Mapping Data Flows can be parameterized, making them reusable for different datasets or scenarios.

7. Data Preview:

- Allows users to preview data transformations at different stages without executing the full pipeline.

Use Cases

1. Data Cleansing:

- Remove duplicates, filter out unwanted data, or standardize formats.

2. Data Enrichment:

- Join and enrich datasets from multiple sources.

3. Data Aggregation:

- Summarize or group data for analytics or reporting.

4. ETL/ELT Workflows:

- Extract, transform, and load data into a data warehouse or data lake.

Execution and Monitoring

- Mapping Data Flows are executed using Azure Data Factory's integration runtime with Spark clusters.
- Users can monitor performance and debug issues using the **ADF Monitoring** dashboard, which provides insights into data flow execution status, duration, and errors.

Advantages

- **Ease of Use:** No need for Spark expertise; transformations are visual.
- **Scalability:** Handles large-scale data processing with Spark.
- **Integration:** Seamlessly works with Azure ecosystem services.
- **Flexibility:** Customizable with parameterization and branching.

Limitations

- **Dependency on Spark:** Execution relies on Azure's Spark infrastructure, which may introduce latency in certain scenarios.
- **Cost:** Running Spark clusters incurs additional costs, especially for complex or large-scale transformations.

Mapping Data Flows is ideal for organizations looking to simplify complex data transformation workflows without heavy coding.

To begin with the Lab:

1. In this lab we are going to create a mapping data flow for the fact table. Also, we'd seen how to create a fact table while working with Azure Synapse. The approach is the same we will use the same script to create the data flow but here we will just use the column names to add and remove columns while working which you will see by completing this lab.
2. Also, we can only create fact and dimension tables using the **sample database** from **Azure SQL Database**. So, if you have deleted your SQL database then create one.
3. Below is the script that we are using to choose columns for both tables.

```
1 SELECT
2     hd.[SalesOrderID],hd.[OrderDate],hd.[CustomerID],hd.[SubTotal],hd.[TaxAmt],hd.[Freight],hd.[TotalDue],
3     dt.[OrderQty],dt.[ProductID],dt.[UnitPrice],dt.[UnitPriceDiscount],dt.[LineTotal]
4 FROM [SalesLT].[SalesOrderHeader] hd INNER JOIN [SalesLT].[SalesOrderDetail] dt
5 ON hd.[SalesOrderID]=dt.[SalesOrderID];
```

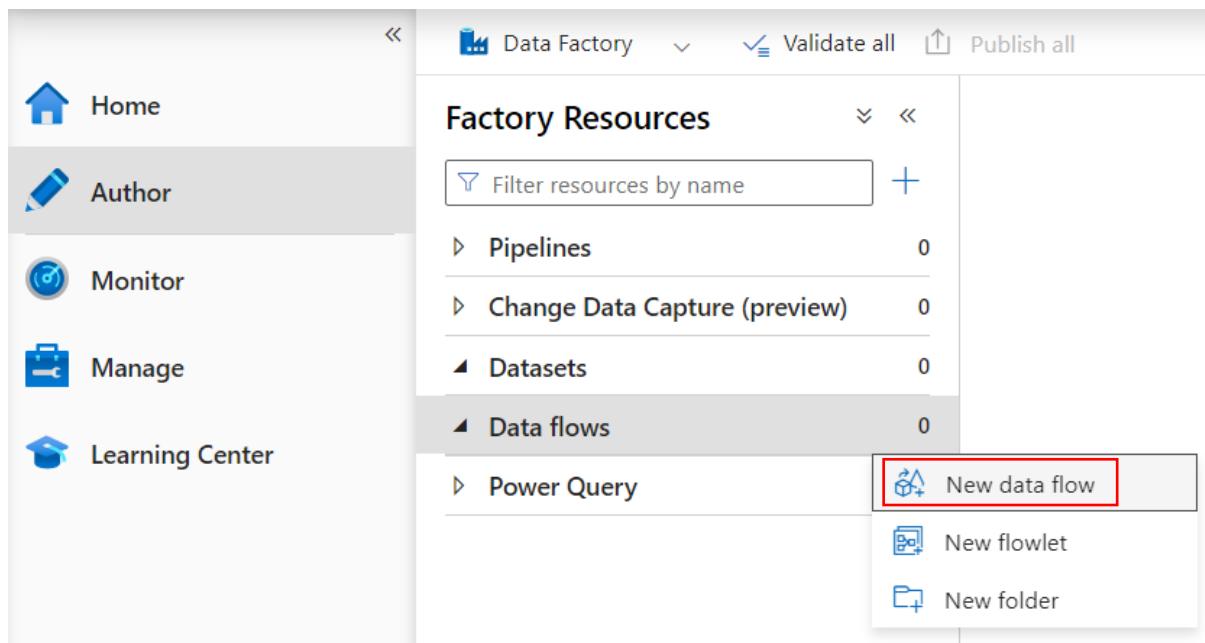
4. Now before creating our table using mapping data flow first, we have to create a fact table inside our dedicated SQL Pool, like we did while working with pipelines in Azure Synapse.
5. Also, from **GitHub** you can download the scripts that we are using for the labs.

```

1  CREATE TABLE factSales
2  (
3      SalesOrderID int NOT NULL,
4      OrderDate datetime NOT NULL,
5      CustomerID int NOT NULL,
6      SubTotal money NOT NULL,
7      TaxAmt money NOT NULL,
8      Freight money NOT NULL,
9      TotalDue money NOT NULL,
10     OrderQty int,
11     ProductID int NOT NULL,
12     UnitPrice money NOT NULL,
13     UnitPriceDiscount money NOT NULL,
14     LineTotal decimal NOT NULL
15 )
16 WITH
17 (
18     DISTRIBUTION=HASH(ProductID)
19 )

```

6. So, to create a mapping data flow in Azure Data Factory, go to the author tab and choose data flows then click on new data flow to create one.



7. You'll be on the canvas where you can add a source. Directly click on it to add a source. Also, you can change the name of your data flow using the properties tab.



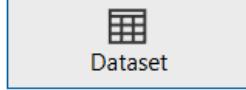
8. Here we need to choose the source. So, to create a fact table, our data comes from two tables: sales order detail and sales order header table. So, we need to make two sources.
9. The first source will be for the sales order header and the second for the sales order detail table.
10. In the source settings we will change the name of the output stream then we have to create a new dataset.

 salesorderheaderstream
Columns:
0 total

Source settings Source options Projection Optimize Inspect Data preview

Output stream name * salesorderheaderstream [Learn more](#)

Description Add source dataset [Reset](#)

Source type *  Dataset  Inline

Dataset * [New](#)

Options Allow schema drift ⓘ Infer drifted column types ⓘ Validate schema ⓘ

Sampling * ⓘ Enable Disable

11. Then click on new to create a new dataset choose Azure SQL database.

New dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

Azure sql

All Azure Database File Generic protocol

Azure SQL Database Azure SQL Database Managed Instance

12. Then choose the same settings which you can see below. Also, you have to create a new linked service for SQL Database.

Set properties

Name

sqlserver_salesOrderheader_table

Linked service *

sqlserver120



Table name

Select...



Enter manually

Import schema

From connection/store None

> Advanced

13. Once the connection is created, you will see it in the dataset section, open it.

Source settings Source options Projection Optimize Inspect Data preview

Output stream name * [Learn more](#)

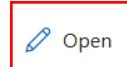
Description [Reset](#)

Source type * Dataset Inline

Dataset * [Test connection](#) [Open](#) [New](#)

Options Allow schema drift [①](#) Infer drifted column types [①](#) Validate schema [①](#)

Sampling * [①](#) Enable Disable



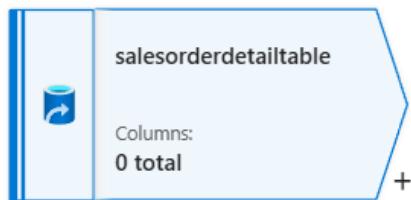
14. Choose the sales **order header table**, go to **Schema**, and click on **Import Schema**.

Connection Schema Parameters

Linked service * [Test connection](#) [Edit](#) [New](#) [Learn more](#)

Table [Refresh](#) [Preview data](#) Enter manually

15. Then we will create another source for the sales order detail table and choose to create a new dataset.



Source settings Source options Projection Optimize Inspect Data preview

Output stream name * [Learn more](#)

Description [Reset](#)

Source type *

 Dataset

 Inline

Dataset *

[New](#)

Options

Allow schema drift [①](#)

Infer drifted column types [①](#)

Validate schema [①](#)

Sampling * [①](#)

Enable Disable

16. Here, give a name to your dataset, choose the linked service, and click Ok to create it.

Set properties

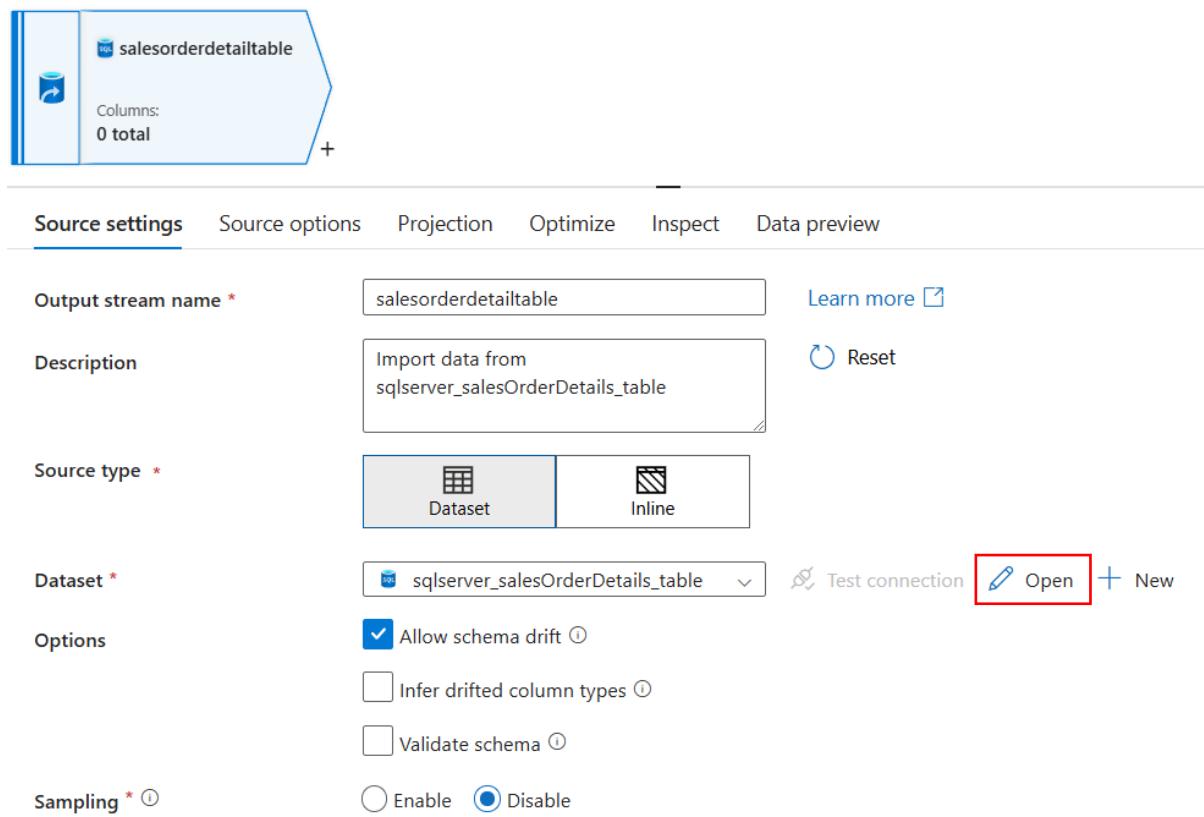
Name
sqlserver_salesOrderDetails_table

Linked service *
sqlserver120 

Table name
Select... 
 Enter manually

Import schema
 From connection/store None
[Advanced](#)

17. Once it is created you will see it in the dataset section click on open.



salesorderdetailtable
Columns: 0 total

Source settings Source options Projection Optimize Inspect Data preview

Output stream name * salesorderdetailtable Learn more 

Description Import data from sqlserver_salesOrderDetails_table 

Source type *  Dataset  Inline

Dataset *  sqlserver_salesOrderDetails_table  Test connection  Open 

Options Allow schema drift 
 Infer drifted column types 
 Validate schema 

Sampling *  Enable  Disable

18. Choose the **sales order detail table** go to **Schema**, and click on **Import Schema**.

Connection Schema Parameters

Linked service *

[Test connection](#) [Edit](#) [New](#) [Learn more](#)

Table Refresh Preview data

Enter manually

19. Then we need to perform the join between the sales order header and sales order detail table. Click on the plus icon and choose the join.



20. Then choose the same JOIN settings as shown below. Choose the right stream as sales order detail and for join type select Inner join.

Join settings Optimize Inspect Data preview

Output stream name * [Learn more](#)

Description

Left stream *

Right stream *

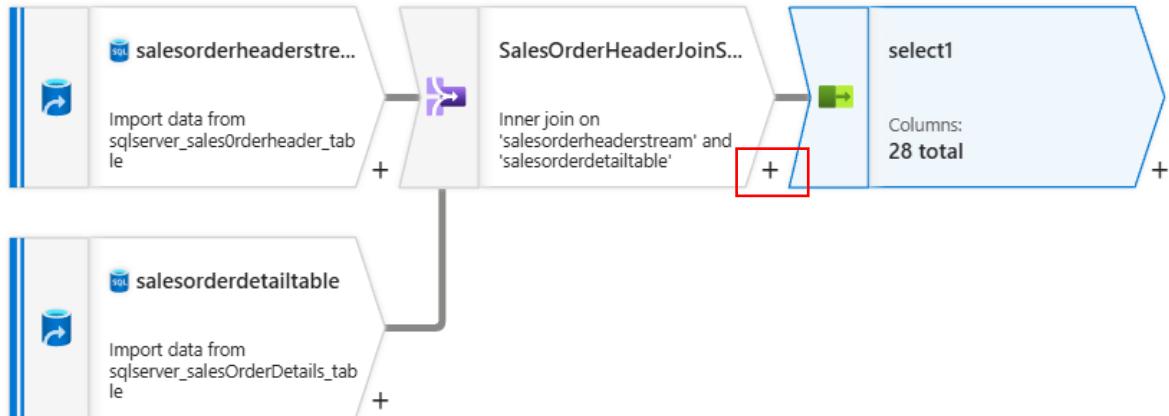
Join type * Inner Full outer Left outer Right outer Custom (cross)

Use fuzzy matching [?](#)

Join conditions *

[+](#) [-](#)

21. Then we will click on the plus icon from our join and from the schema modifier choose the **Select** function to select the columns of our choice.



22. Then what we need to do is delete the columns that are not needed. You can make use of the script and remove the columns.

Select settings Optimize Inspect Data preview

Output stream name * [Learn more](#)

Description

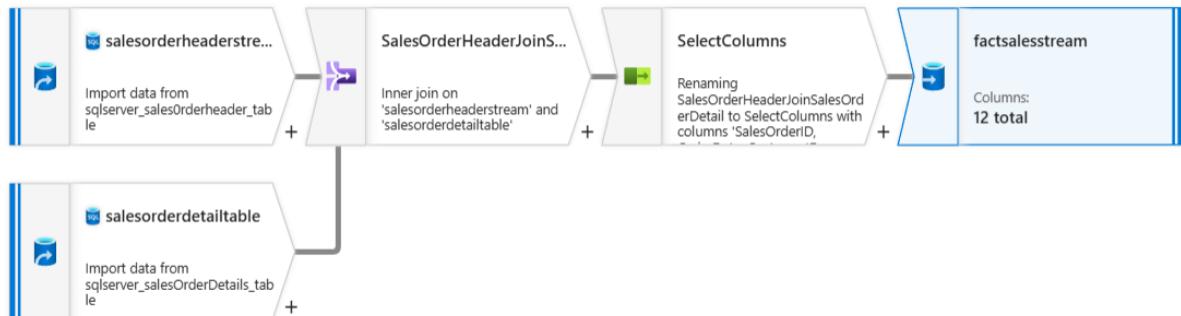
Incoming stream *

Options Skip duplicate input columns [?](#) Skip duplicate output columns [?](#)

Input columns * Auto mapping [?](#) [Reset](#) [+](#) Add mapping [Delete](#) 12 mappings: 19 column(s) from the inputs left unmapped [?](#)

SalesOrderHeaderJoinSalesOrderDetail's column	Name as
123 salesorderheaderstream@SalesOrderID	SalesOrderID
OrderDate	OrderDate
123 CustomerID	CustomerID
e ^x SubTotal	SubTotal
⋮	
e ^x TaxAmt	TaxAmt
e ^x Freight	Freight
e ^x TotalDue	TotalDue
123 OrderQty	OrderQty
123 ProductID	ProductID
e ^x UnitPrice	UnitPrice
e ^x UnitPriceDiscount	UnitPriceDiscount
e ^x LineTotal	LineTotal

23. Now choose Sink or the destination from the select column area. Click on the plus icon from the select column function and choose the sink.



24. For the sink give it a name and, in the dataset, click on new to create a new linked service for your dedicated SQL Pool and if you already have one then you can choose it.

25. After that click on Open to go inside the dataset and choose your fact sales table.

Sink Settings Errors Mapping Optimize Inspect Data preview

Output stream name * factsalesstream [Learn more](#)

Description Export data to synapseworkspace2361_dataPool [Reset](#)

Incoming stream * SelectColumns

Sink type * [Dataset](#) [Inline](#) [Cache](#)

Dataset * [synapseworkspace2361_dataPool](#) [Test connection](#) [Open](#) [New](#)

Options Allow schema drift [①](#) Validate schema [①](#)

26.In the end inside your sink, go to mapping and disable auto mapping to check the mappings manually.

Sink Settings Errors **Mapping** Optimize Inspect Data preview

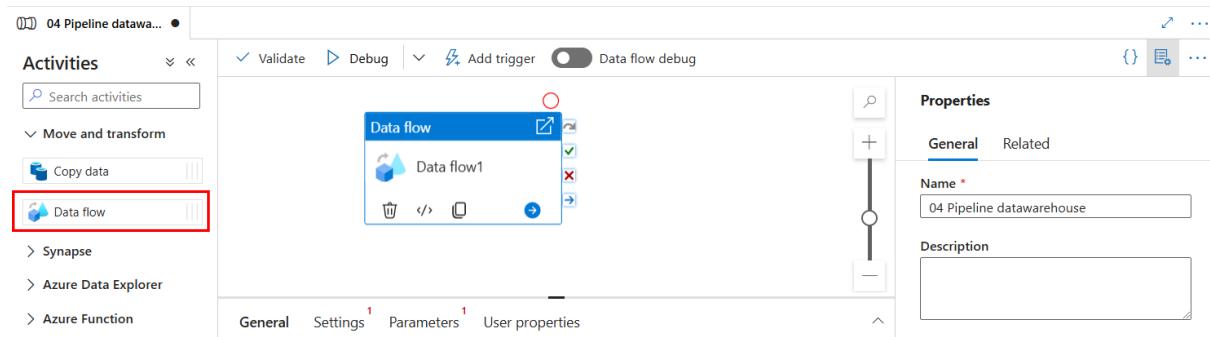
Options Skip duplicate input columns [①](#) Skip duplicate output columns [①](#)

Auto mapping [①](#) [Reset](#) [Add mapping](#) [Delete](#) [Output format](#) 12 mappings: All c

Input columns	Output columns
123 SalesOrderID	123 SalesOrderID
OrderDate	OrderDate
CustomerID	CustomerID
e ^x SubTotal	e ^x SubTotal
e ^x TaxAmt	e ^x TaxAmt
e ^x Freight	e ^x Freight
e ^x TotalDue	e ^x TotalDue
OrderQty	OrderQty
ProductID	ProductID
e ^x UnitPrice	e ^x UnitPrice

27.We have created a mapping data flow which has the visual representation of creating our fact table. How do we run mapping data flow?

28.For that we need to create a pipeline. So, we will go to move and transform the area from here we will choose Data flow.



29. Here in the setting, you need to choose the same settings as shown below.

First, choose your data flow name and choose a Staging folder.

30. So, for staging you have to go to the storage account and create a new container with name staging this container will be used as a staging container and it will hold the temporary data in it.

Name	Last modified	Anonymous access level	Lease state
\$logs	12/26/2024, 10:17:56 AM	Private	Available
data	12/26/2024, 10:18:11 AM	Private	Available
staging	12/27/2024, 3:34:31 PM	Private	Available

31. As you can see below, we have chosen the sating linked service which is our data lake and from it, we have the staging folder. Then publish your changes and trigger the pipeline.

Setting	Value
Data flow *	01_Dataflow_datawarehouse
Run on (Azure IR) *	AutoResolveIntegrationRuntime
Compute size *	Small
Logging level *	Verbose
Staging linked service	gen2datafactory2163
Staging storage folder	staging

32. The duration of the pipeline will take around 3-4 minutes. This is because an Apache Spark cluster will be created, the data flow will run, the pipeline will run, and, in the end, the Apache Spark cluster will be deleted because it runs for a temporary period.

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run	Param
01 Fact and dimension Tables	12/28/2024, 2:38:16 PM	12/28/2024, 2:41:36 PM	3m 20s	Manual trigger	✓ Succeeded	Original	

33. After the pipeline run is successful, you can check the data in your table.

SalesOrderID	OrderDate	CustomerID	SubTotal	TaxAmt	Freight	TotalDue
71782	2008-06-01T00:00:00.000Z	29485	39785.3304	3182.8264	994.6333	43962.7901
71780	2008-06-01T00:00:00.000Z	30113	38418.6895	3073.4952	960.4672	42452.6519
71782	2008-06-01T00:00:00.000Z	29485	39785.3304	3182.8264	994.6333	43962.7901
71780	2008-06-01T00:00:00.000Z	30113	38418.6895	3073.4952	960.4672	42452.6519
71776	2008-06-01T00:00:00.000Z	30072	78.8100	6.3048	1.9703	87.0851
71782	2008-06-01T00:00:00.000Z	29485	39785.3304	3182.8264	994.6333	43962.7901
71783	2008-06-01T00:00:00.000Z	29957	83858.4261	6708.6741	2096.4607	92663.5609

00:00:02 Query executed successfully.