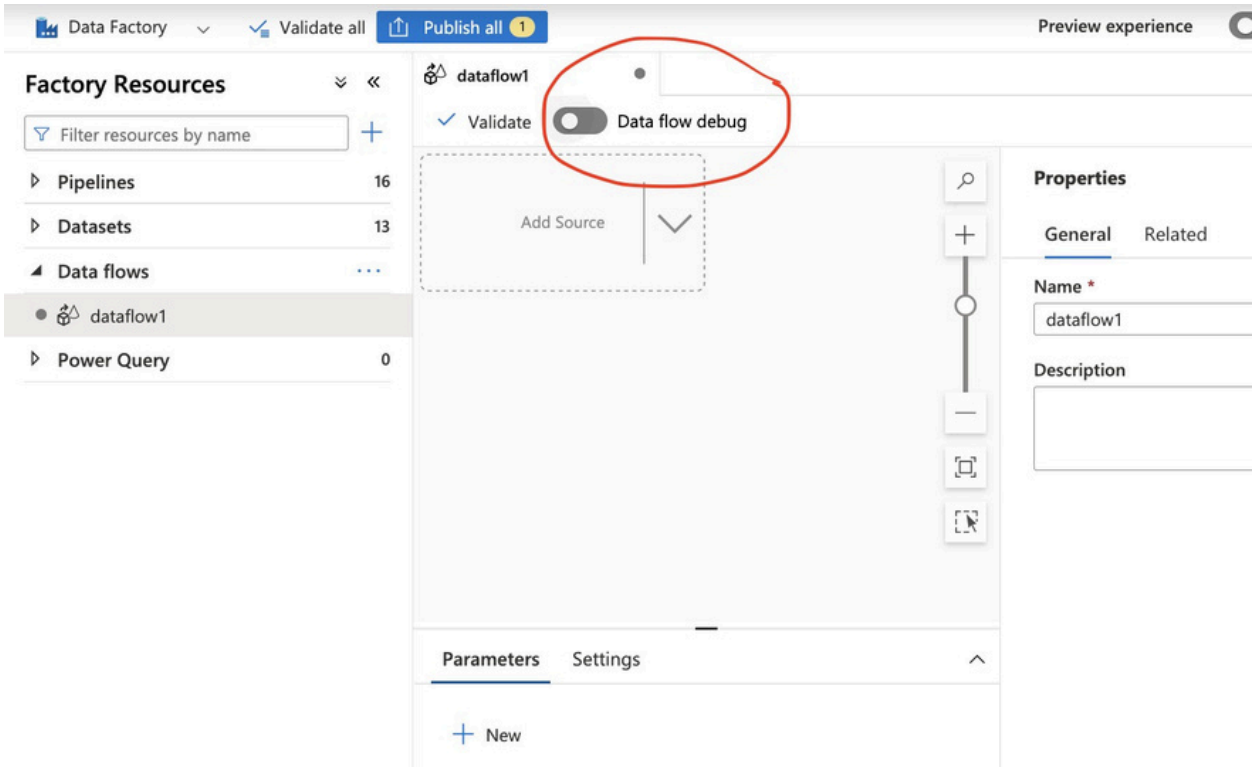
Azure Data Factory DATA FLOW:

Mapping data flows are visually designed data transformations in Azure Data Factory.
Data flows allow data engineers to develop data transformation logic without writing code.
The resulting data flows are executed as activities within Azure Data Factory pipelines that use scaled-out Apache Spark clusters.



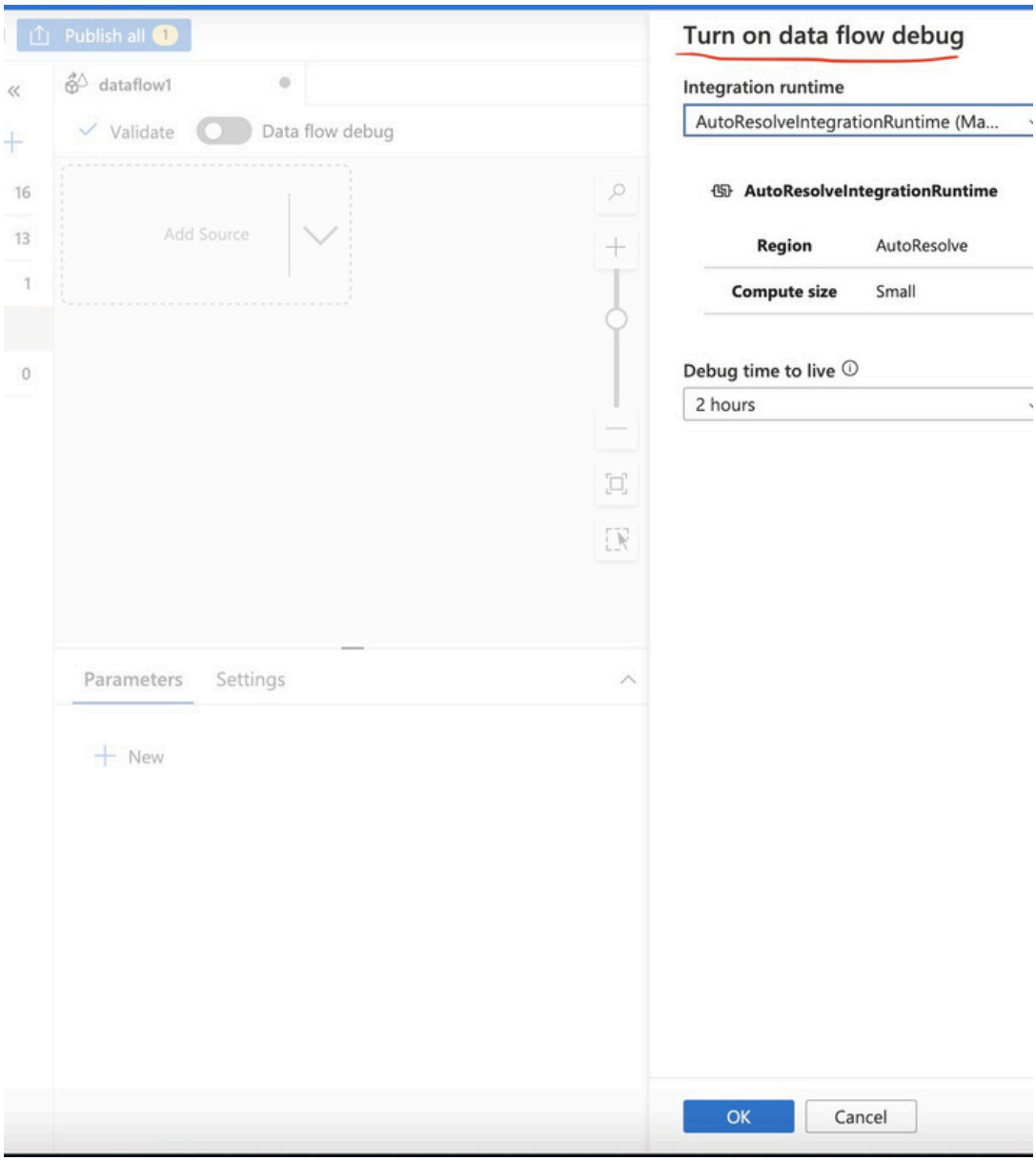
Data flow is nothing but spark code and cluster.

Behind the scene Whatever we do in dataflow, Under the hood it will run into spark code and run on spark cluster.

Spark is a big data technology it can process huge amount of data in fastest way.

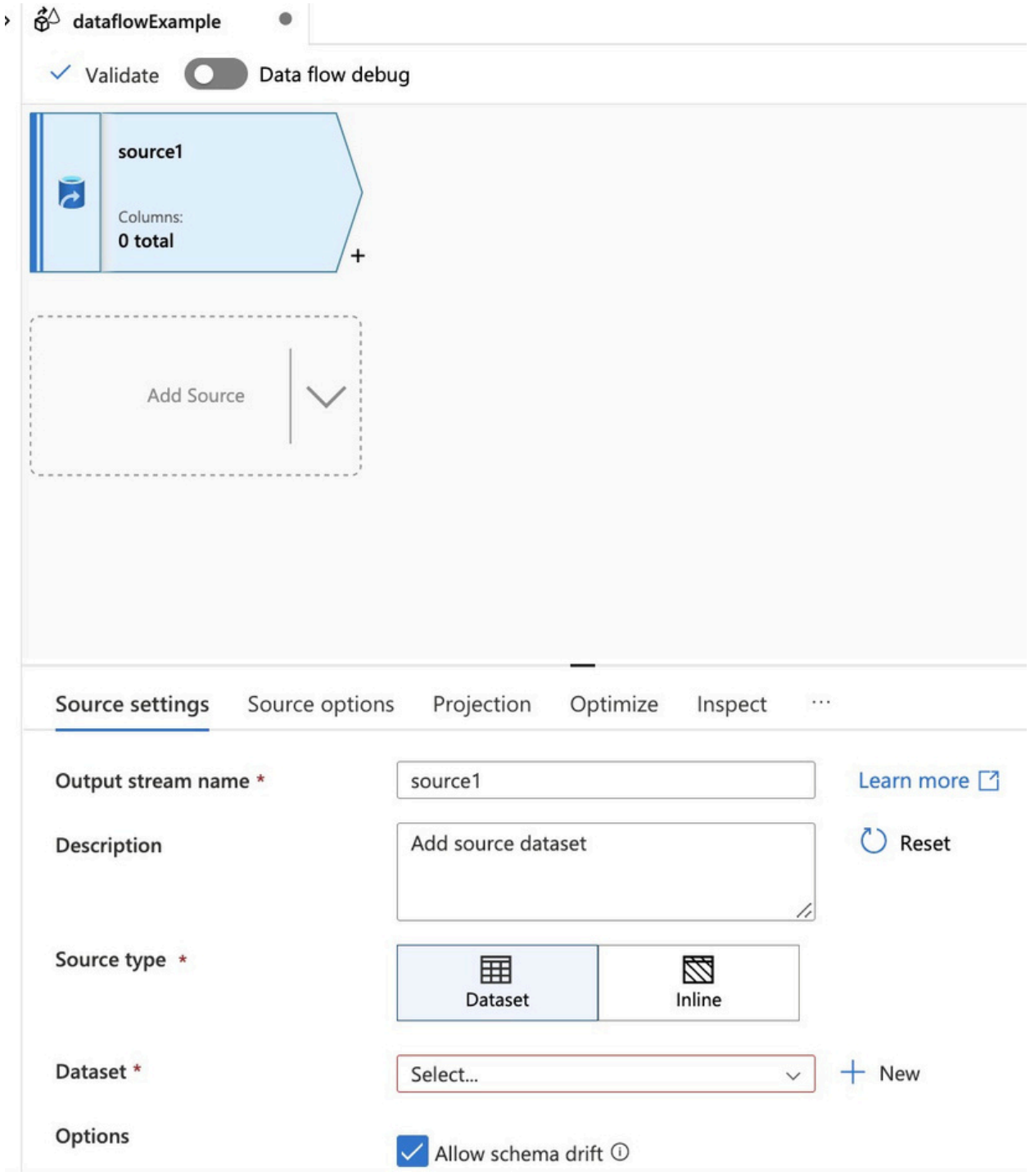
Why do we need to learn spark Instead doing all transformation in ADF dataflow?

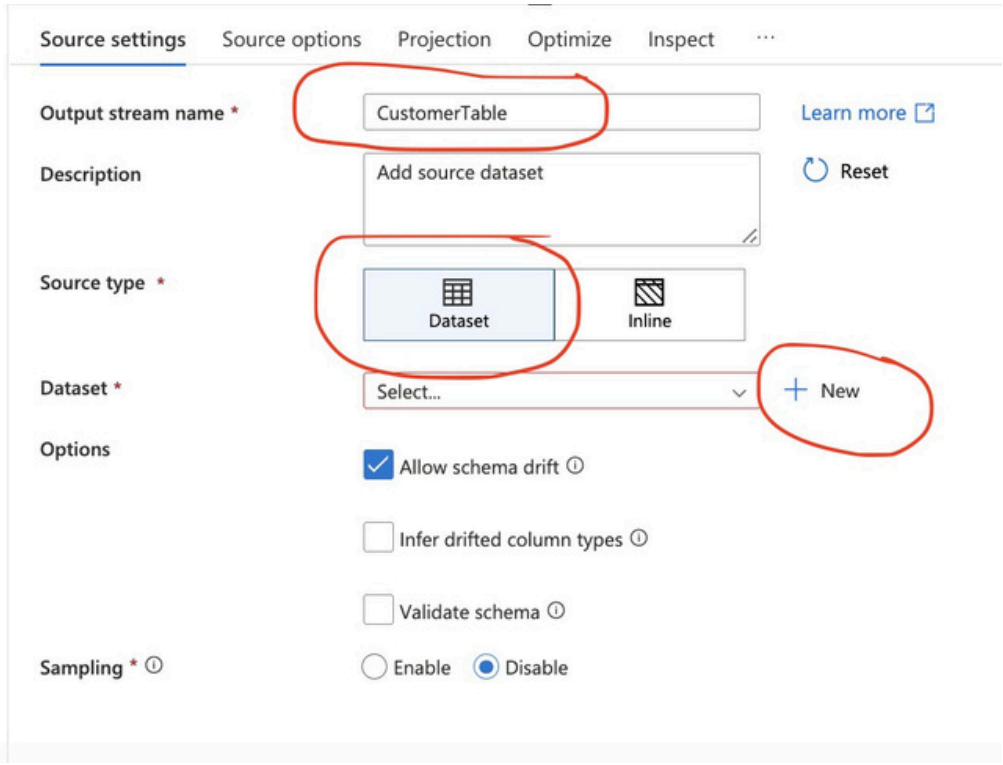
To do more complex and customize , it is prefer to use spark over dataflow as code is very easy and better way.



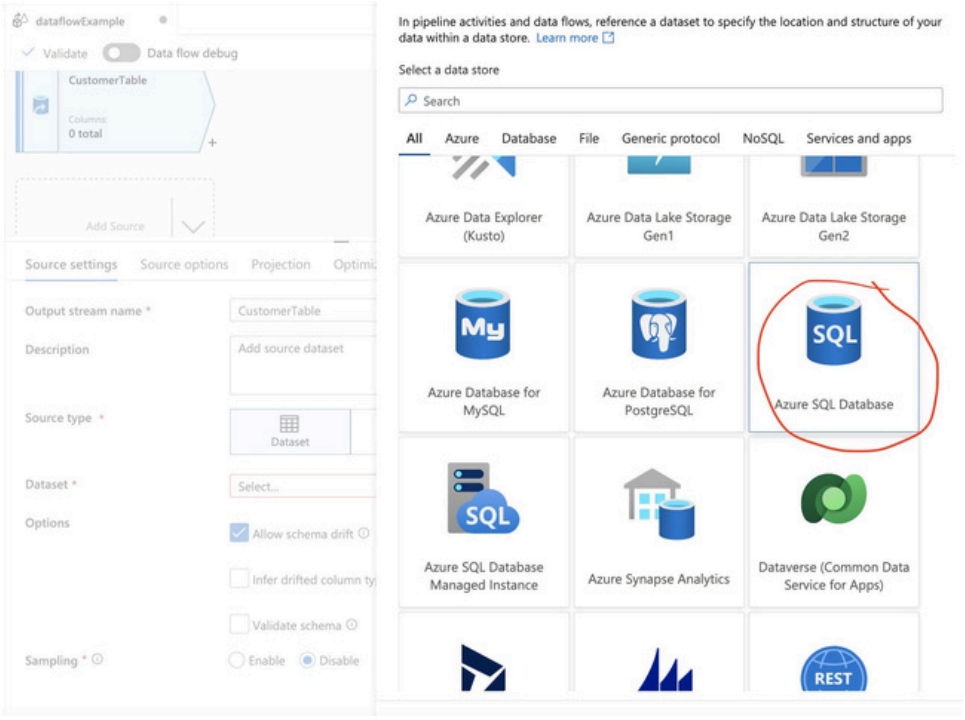
Under the hood it creates the cluster when we enable dataflow debug.

For every transformation , we need to add source:

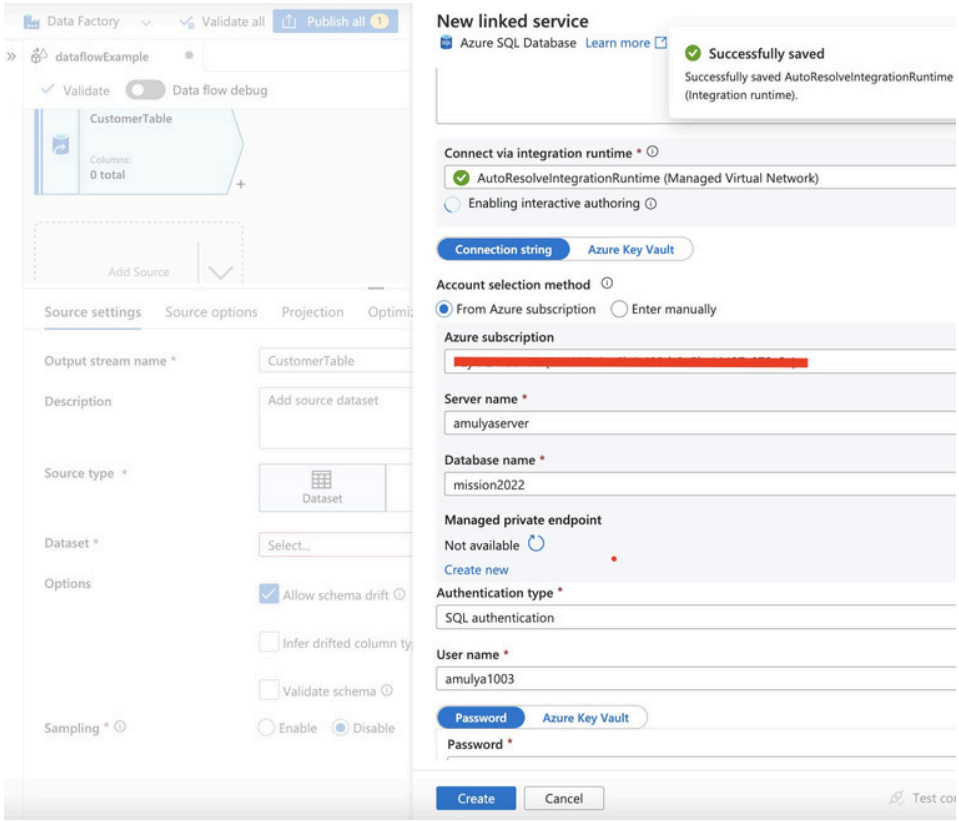




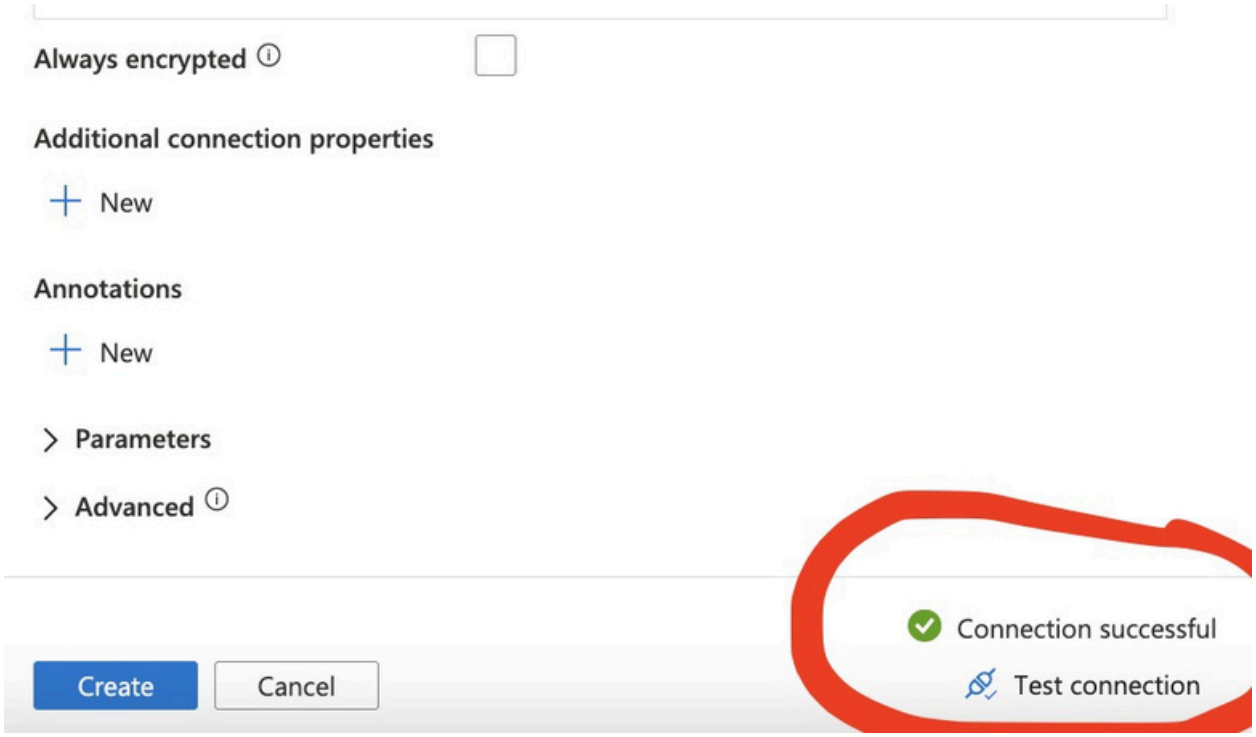
Create a new Dataset to point to customer Table:



In Dataset, Click on linked service to create.



Click on Test Connection to verify Linked Service connected successfully:



Click on Create.

Select the Table Name , you want to point it to:

Set properties

Name

DS_Customer_SQLDB

Linked service *

AzureSqlDatabase1

Connect via integration runtime *

AutoResolveIntegrationRuntime (Managed Virtual Network)

Interactive authoring enabled

Table name

SalesLT.Customer

Filter...

None

SalesLT.Address

SalesLT.Customer

SalesLT.CustomerAddress

SalesLT.Product

SalesLT.ProductCategory

SalesLT.ProductDescription

SalesLT.ProductModel

Set properties

Name

DS_Customer_SQLDB

Linked service *

AzureSqlDatabase1

Connect via integration runtime *

AutoResolveIntegrationRuntime (Managed Virtual Network)

Interactive authoring enabled

Table name

SalesLT.Customer

Edit

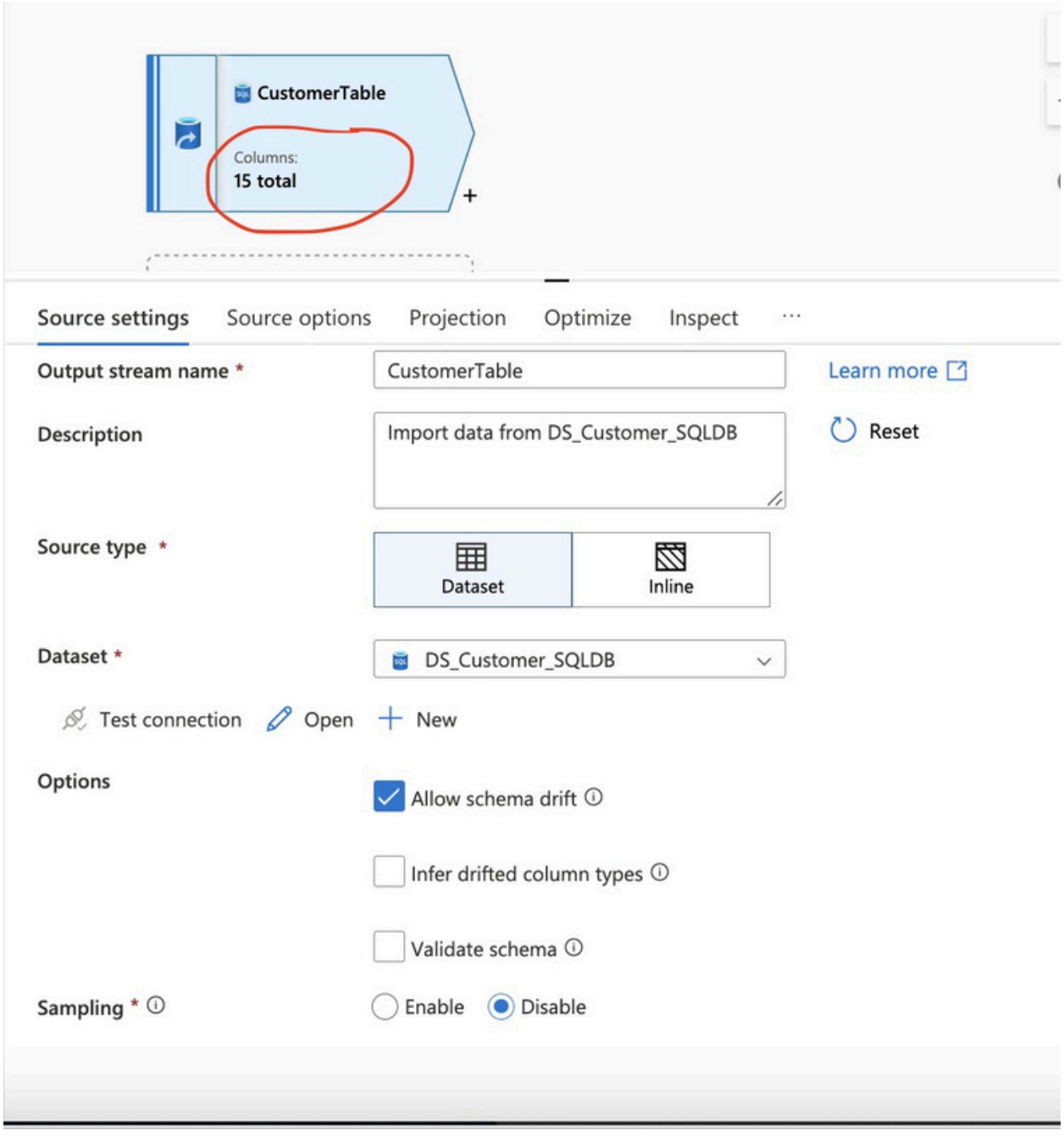
Import schema

From connection/store None

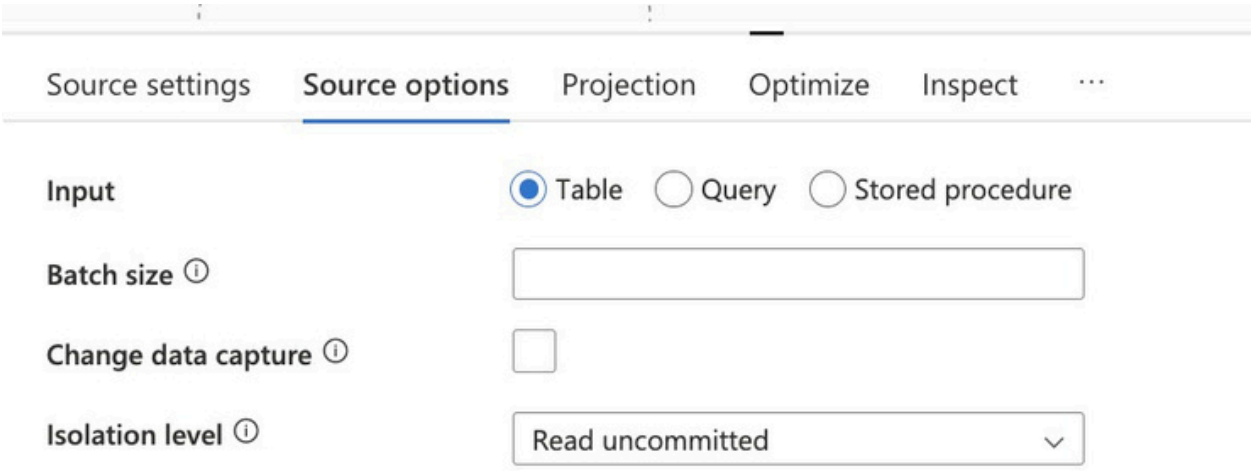
Advanced

OK Back Cancel

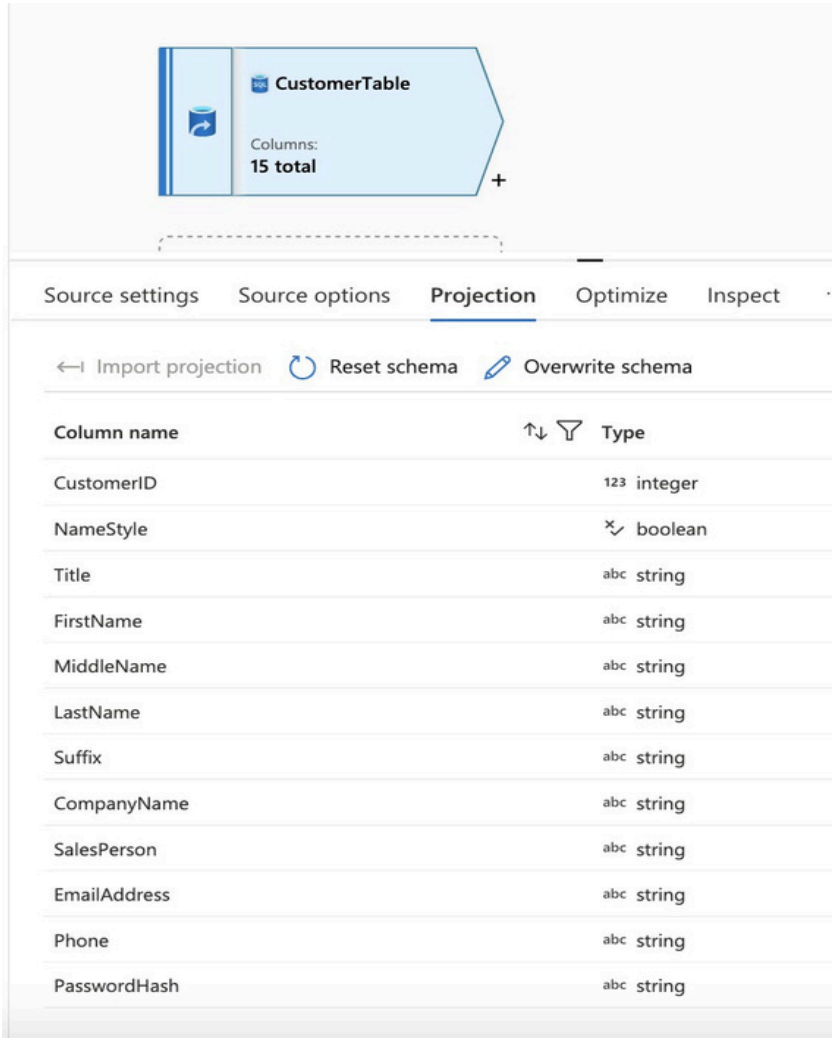
Once dataset created successfully, you can notice 15 columns in Customer Table.



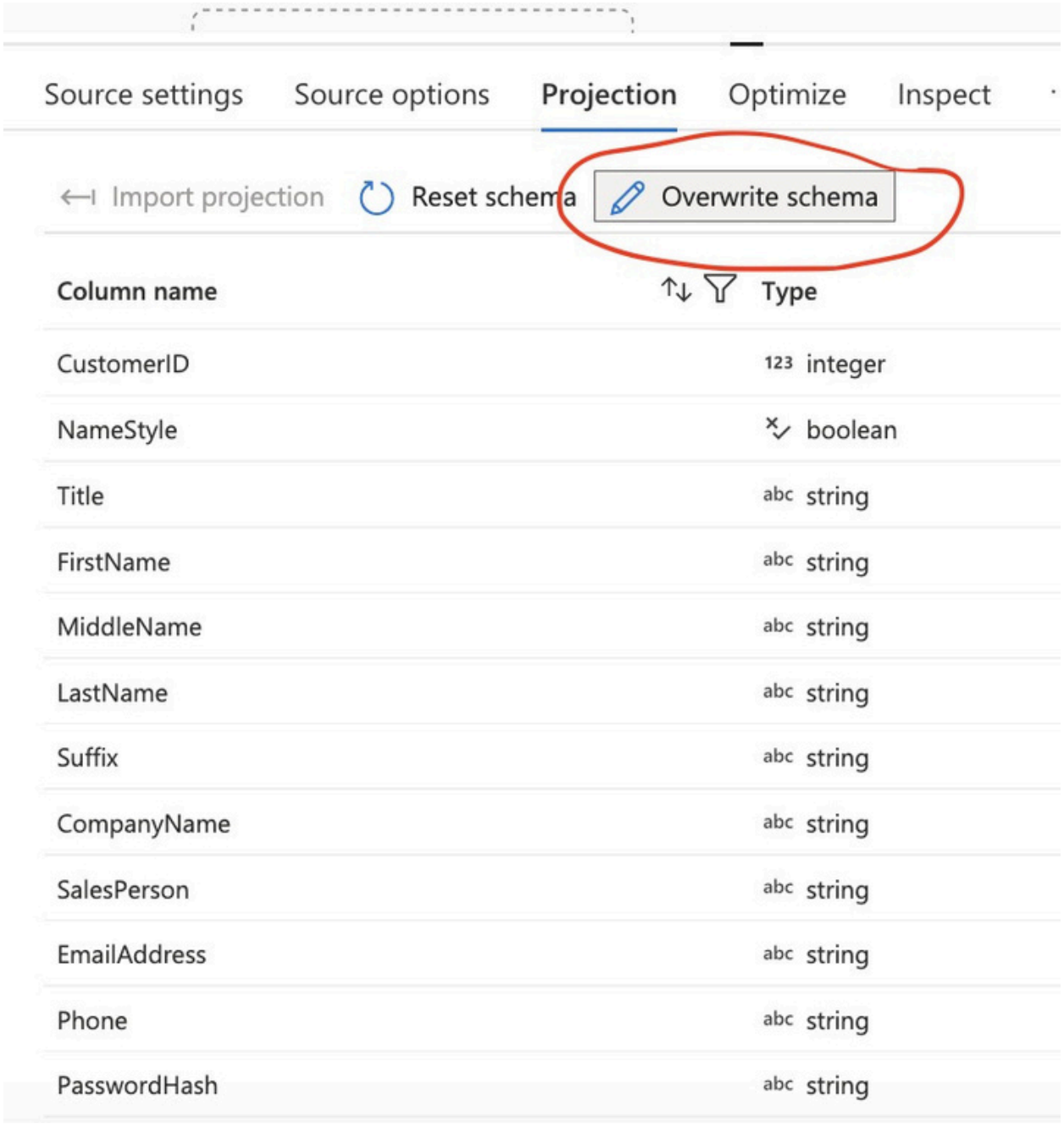
Source options:



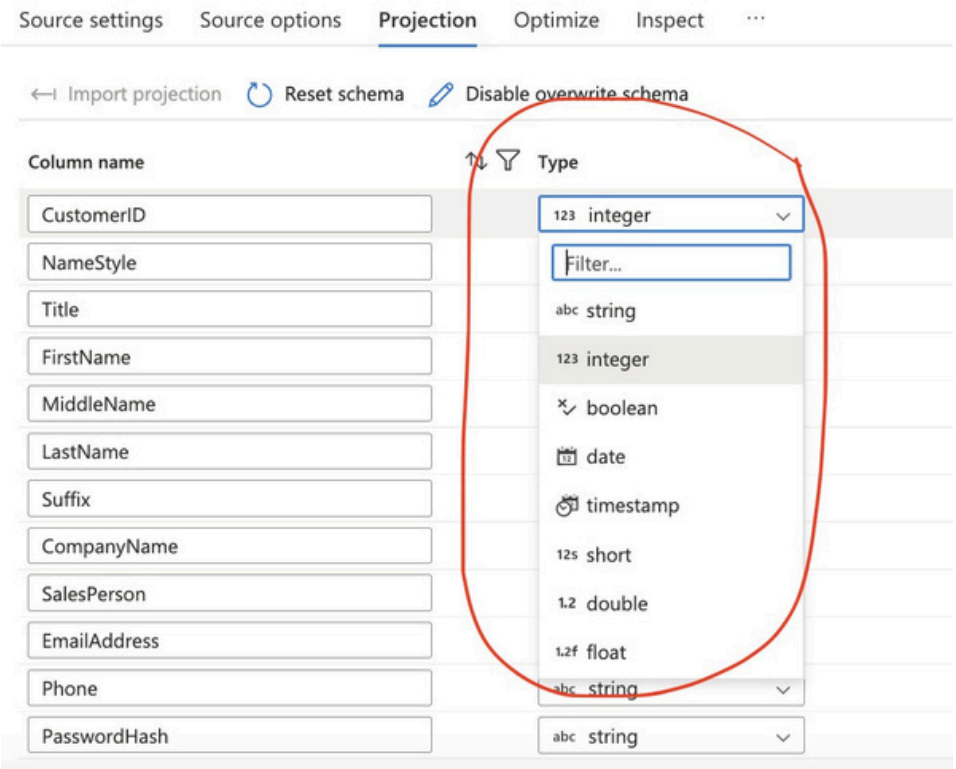
Projection contains all metadata information. In this we can overwrite the schema:



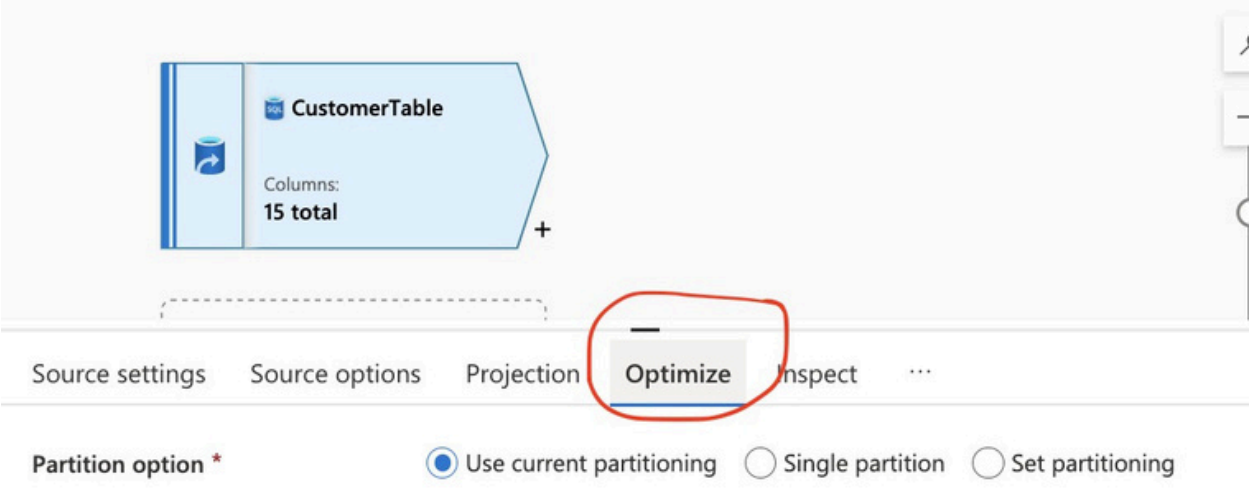
Click on “Overwrite Schema”, to change



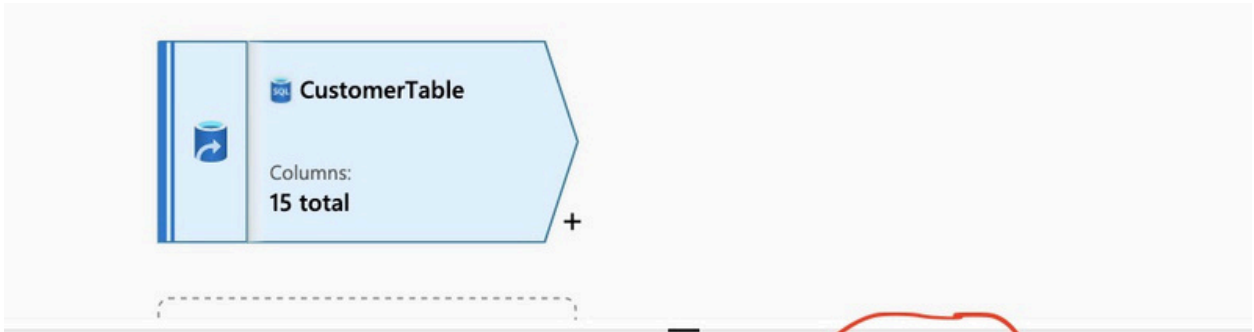
You can change Integer to anyType:



Most of the Time, current partitioning gives the best performance:



Inspect shows the Actual output:

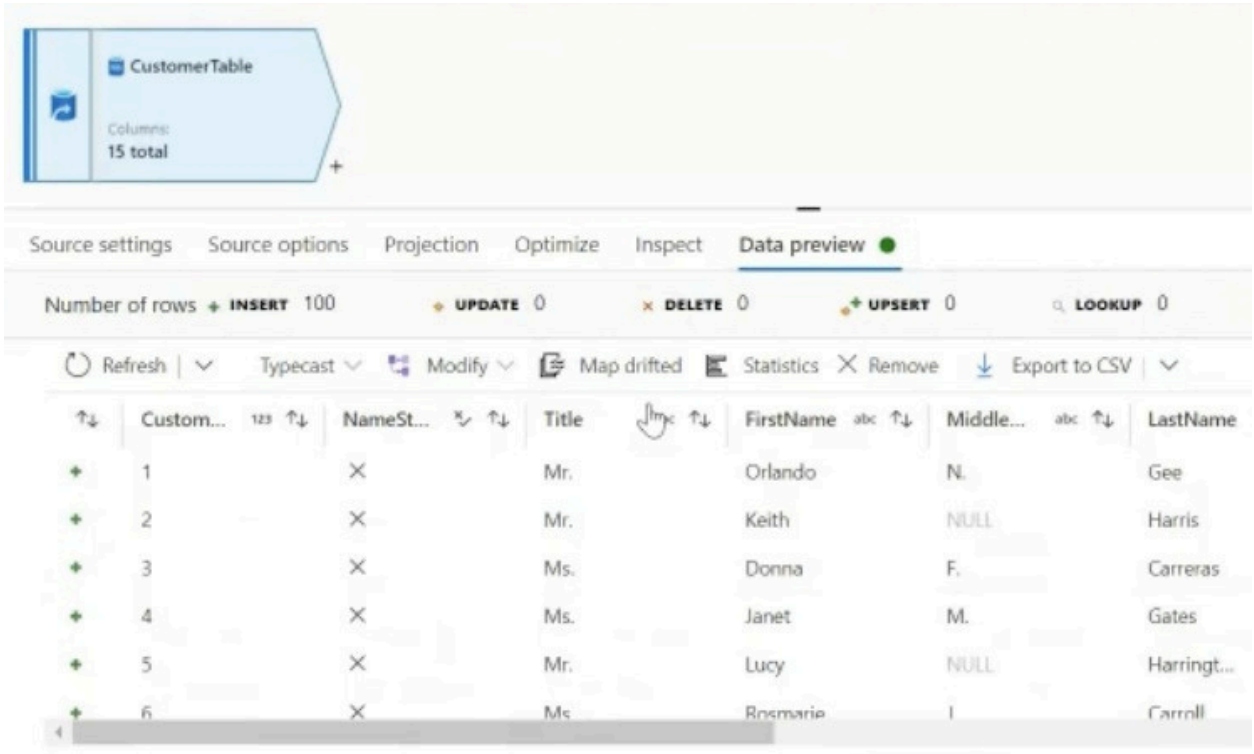


Source settings | Source options | Projection | Optimize | **Inspect** | ···

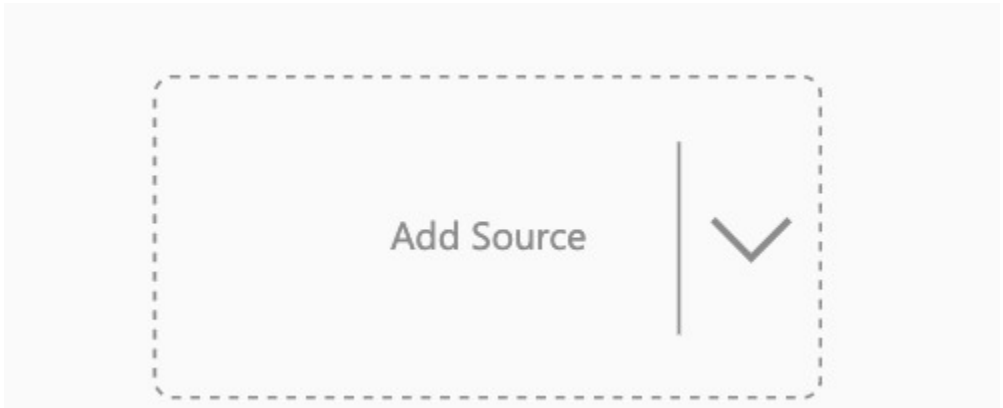
Number of columns **Total** 15

Order	Column	Type
1	CustomerID	123 integer
2	NameStyle	boolean
3	Title	abc string
4	FirstName	abc string
5	MiddleName	abc string
6	LastName	abc string
7	Suffix	abc string
8	CompanyName	abc string
9	SalesPerson	abc string
10	EmailAddress	abc string
11	Phone	abc string

IF no transformations, Data preview will show input data:

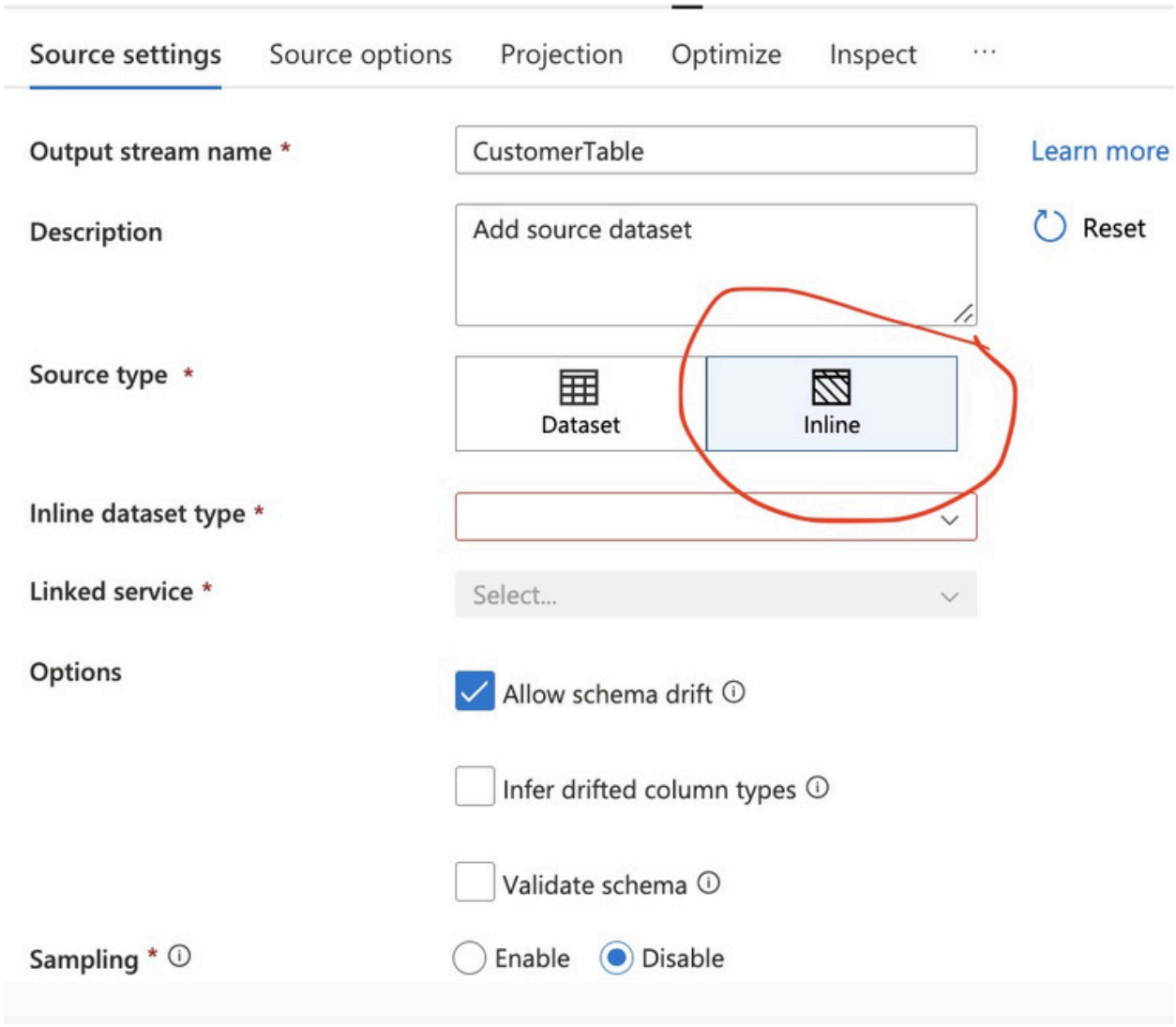


Click on another ADD SOURCE to test Inline:

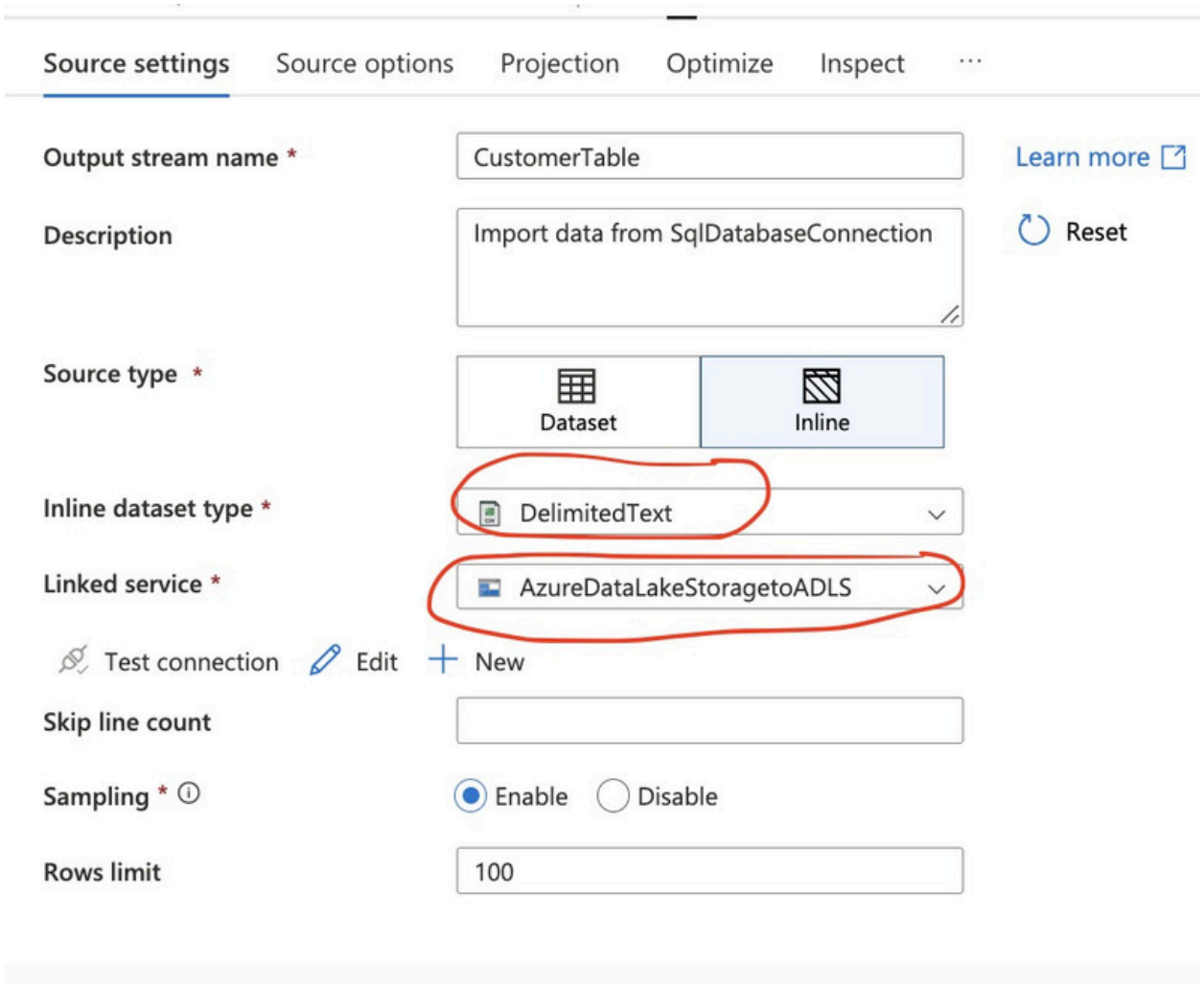


Inline:

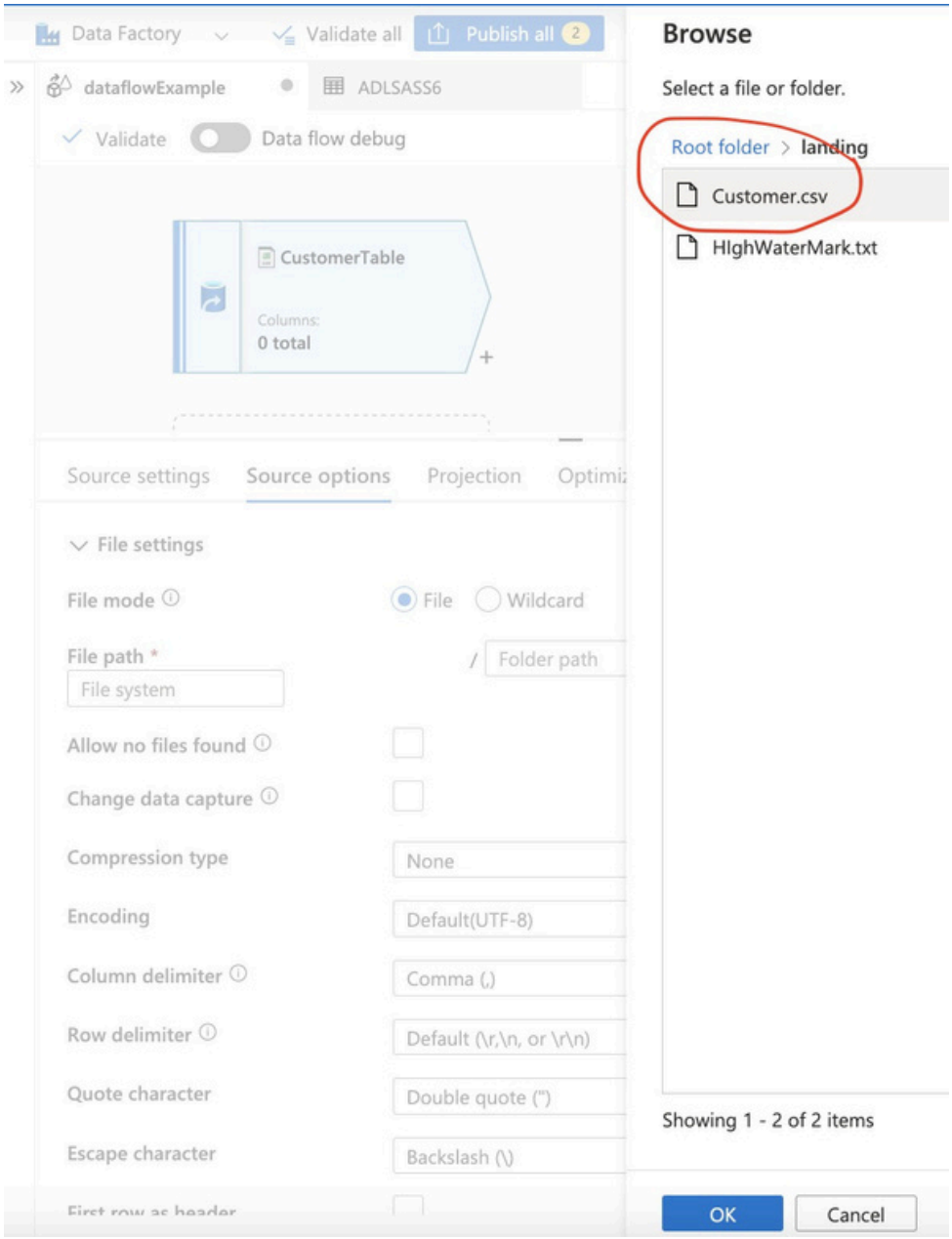
If you want some file or dataset is no reused , then we can use Inline



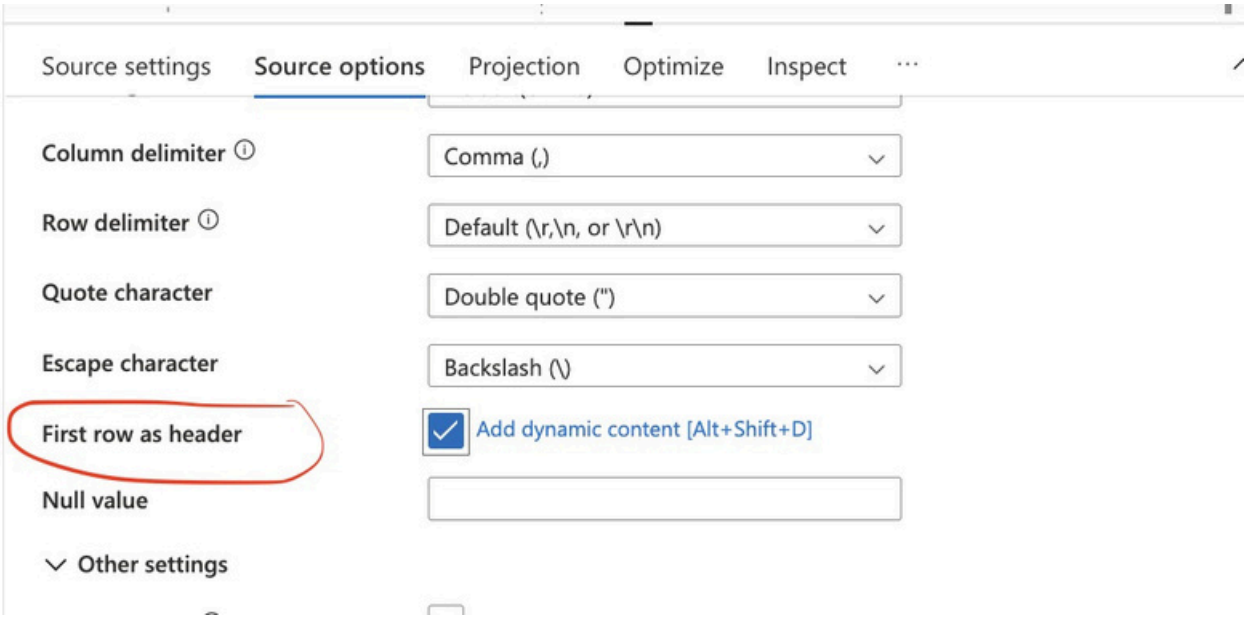
For CSV file, select “DelimitedText” and linked service connected ADLS as shown below:



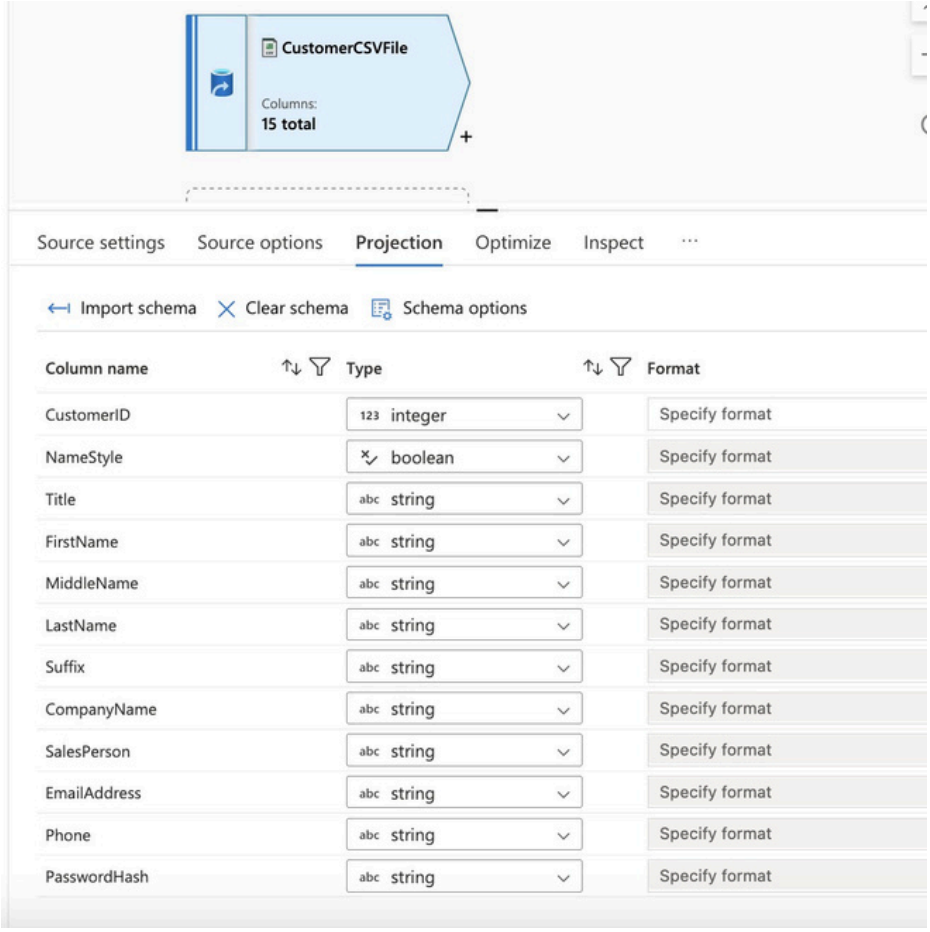
Click on source options and select the exact path location:



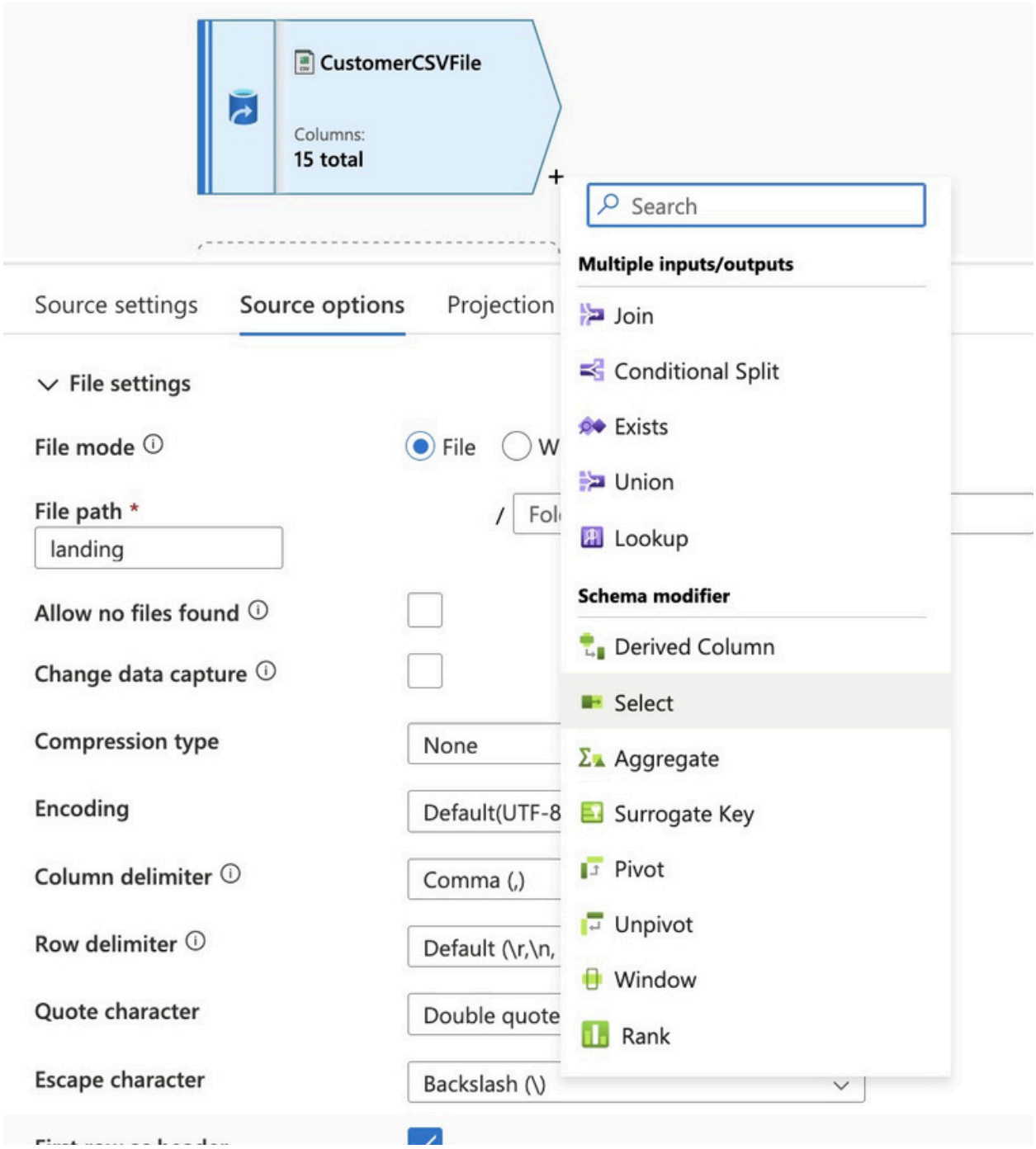
As CSV file as header:

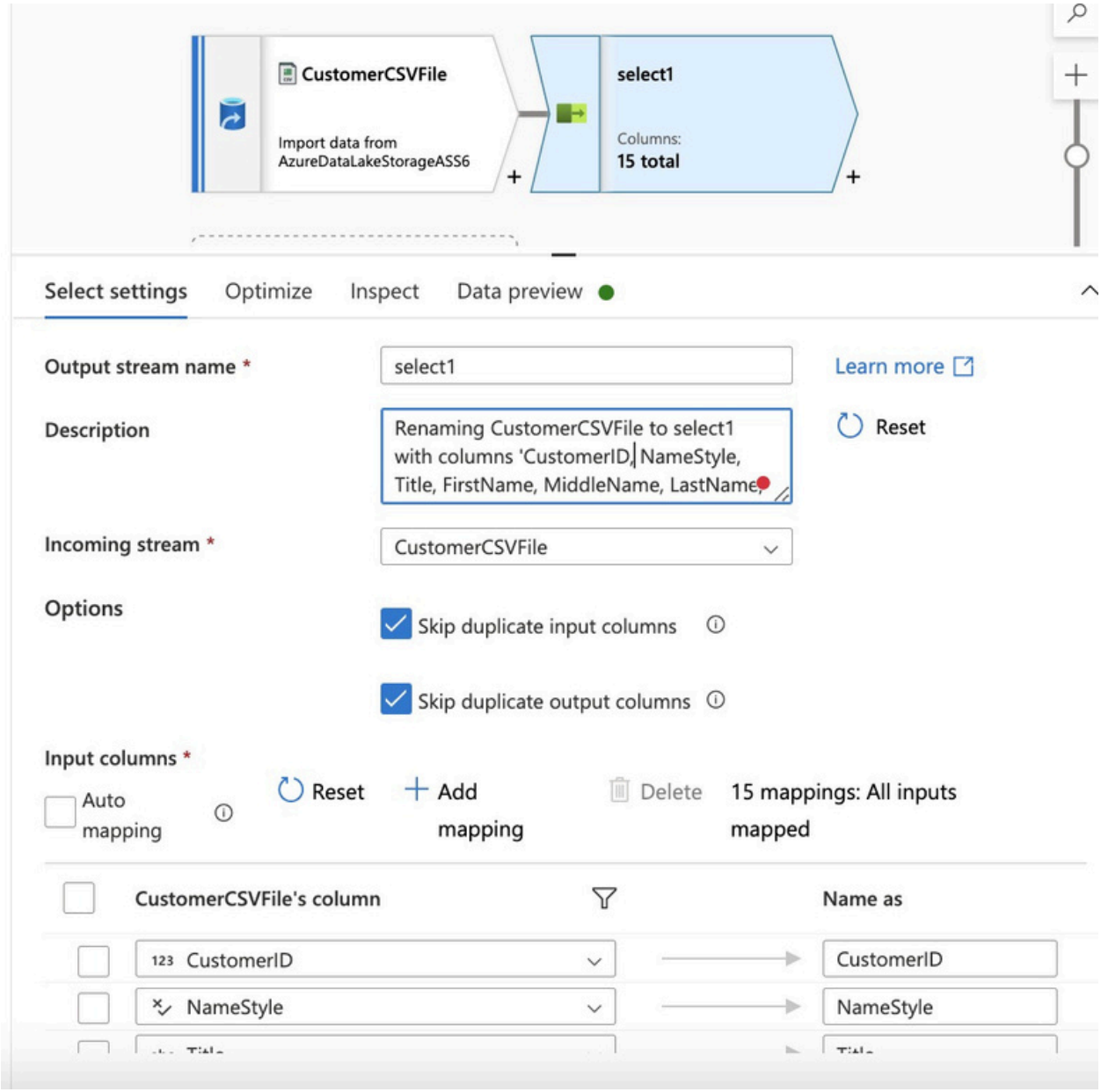


We can specify format of our own:

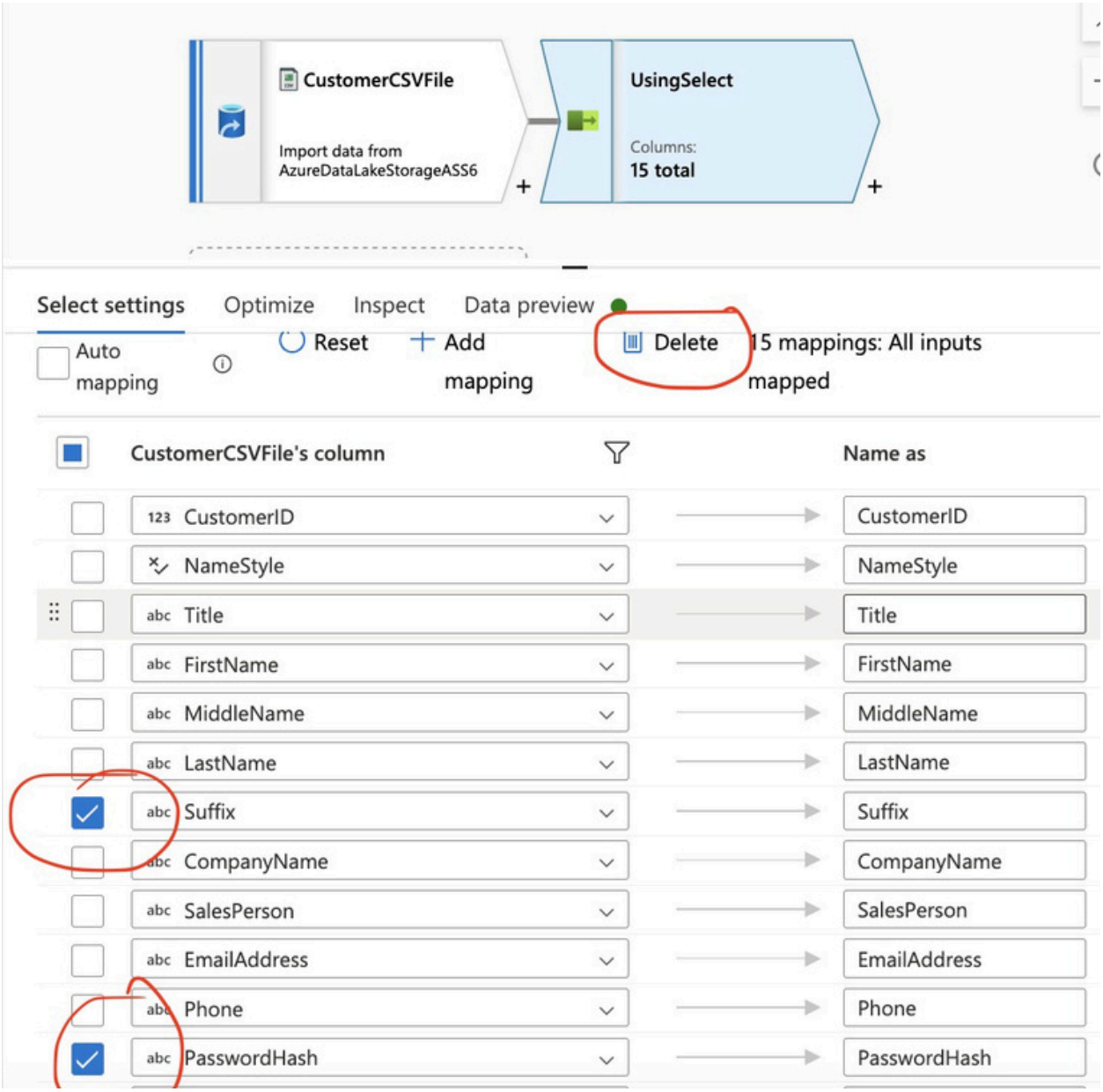


Click on + symbol to show only few columns.

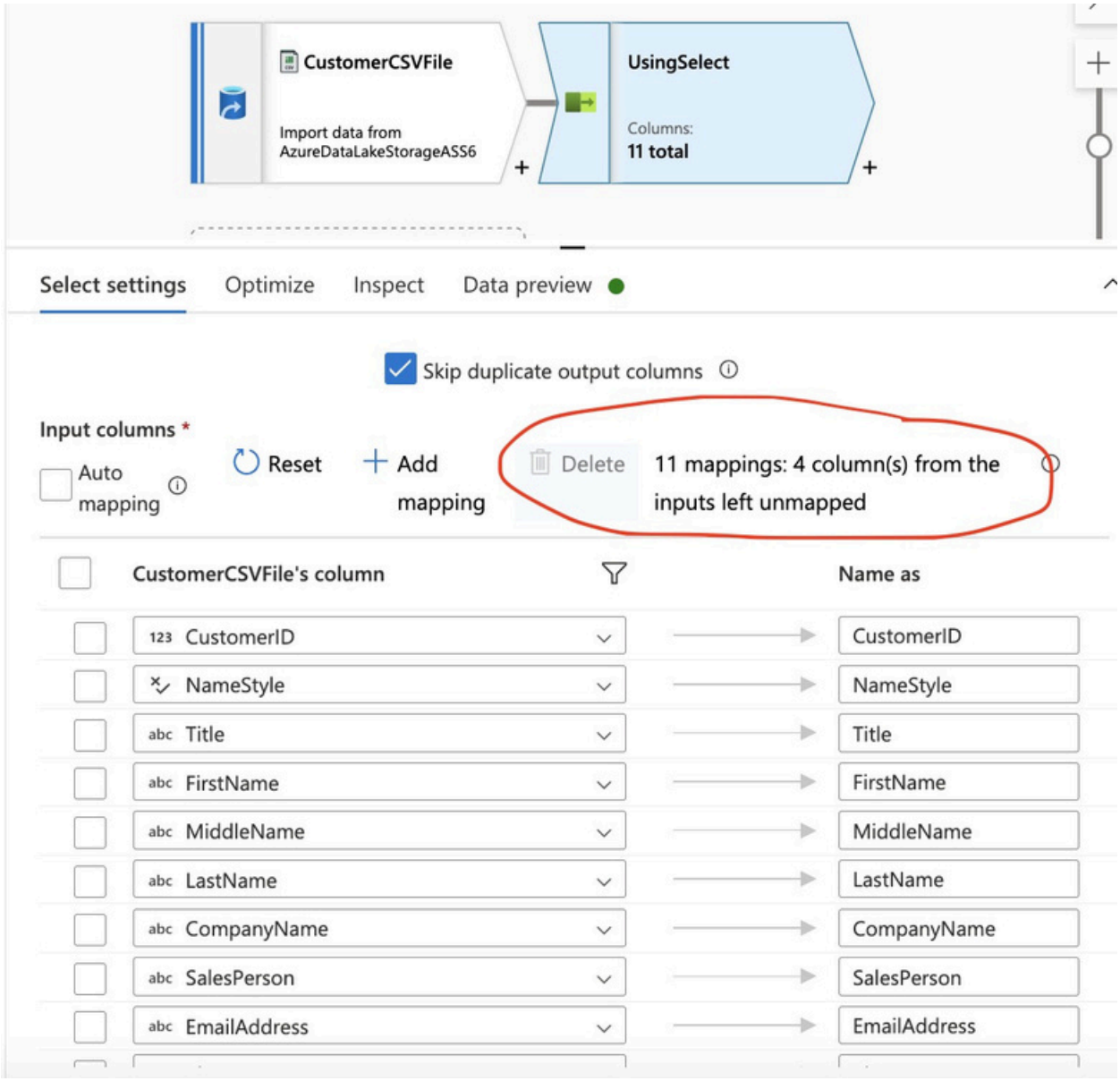



CustomerCSVFile
Import data from AzureDataLakeStorageASS6
select1
Columns:
15 total
Select settings
Optimize
Inspect
Data preview
Output stream name *
select1
Learn more
Description
Renaming CustomerCSVFile to select1 with columns 'CustomerID, NameStyle, Title, FirstName, MiddleName, LastName,
Reset
Incoming stream *
CustomerCSVFile
Options
Skip duplicate input columns
Skip duplicate output columns
Input columns *
Auto mapping
Reset
Add mapping
Delete
15 mappings: All inputs mapped
CustomerCSVFile's column
Name as
CustomerID
CustomerID
NameStyle
NameStyle

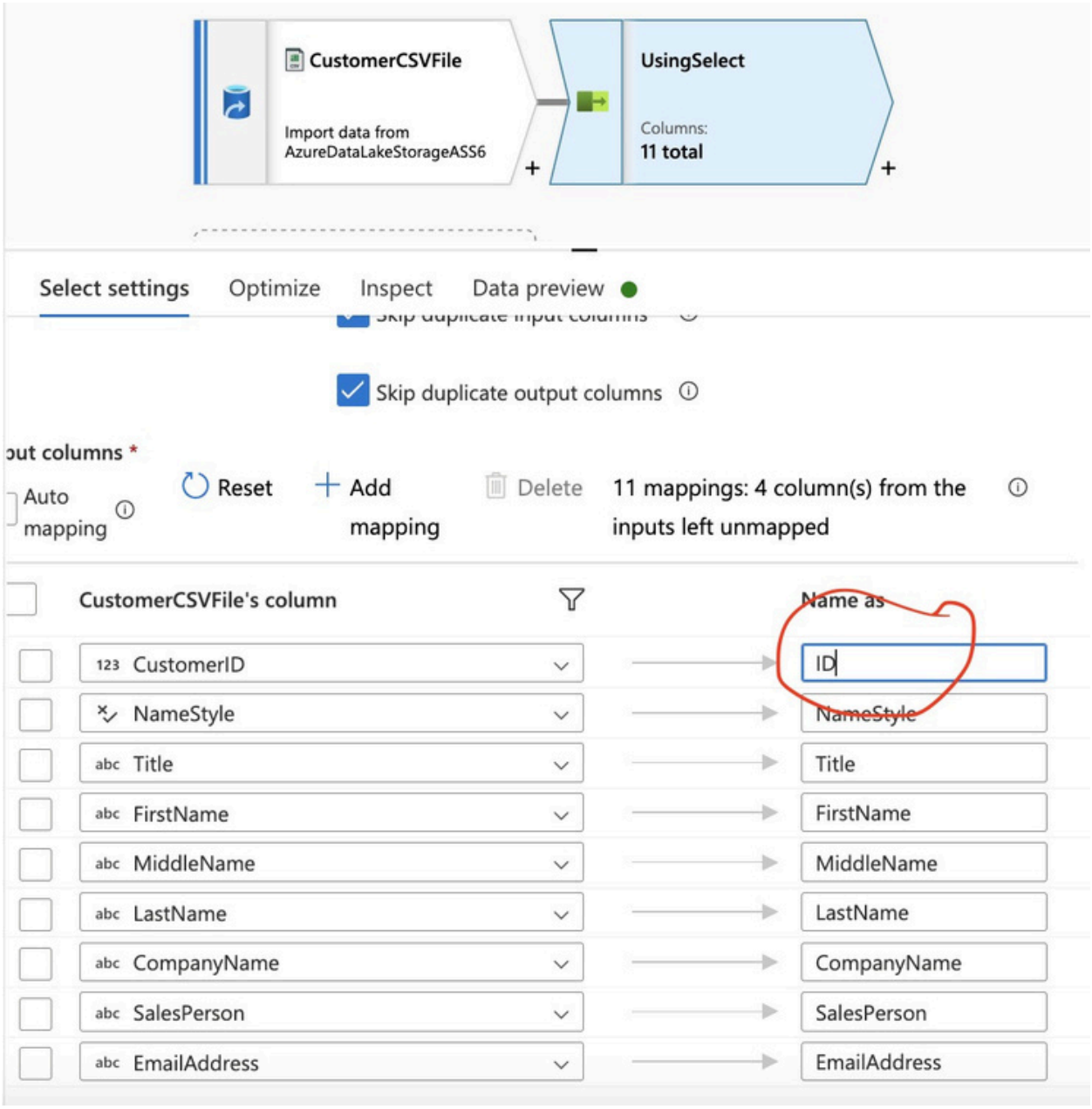
Select and delete the columns you no longer needed:



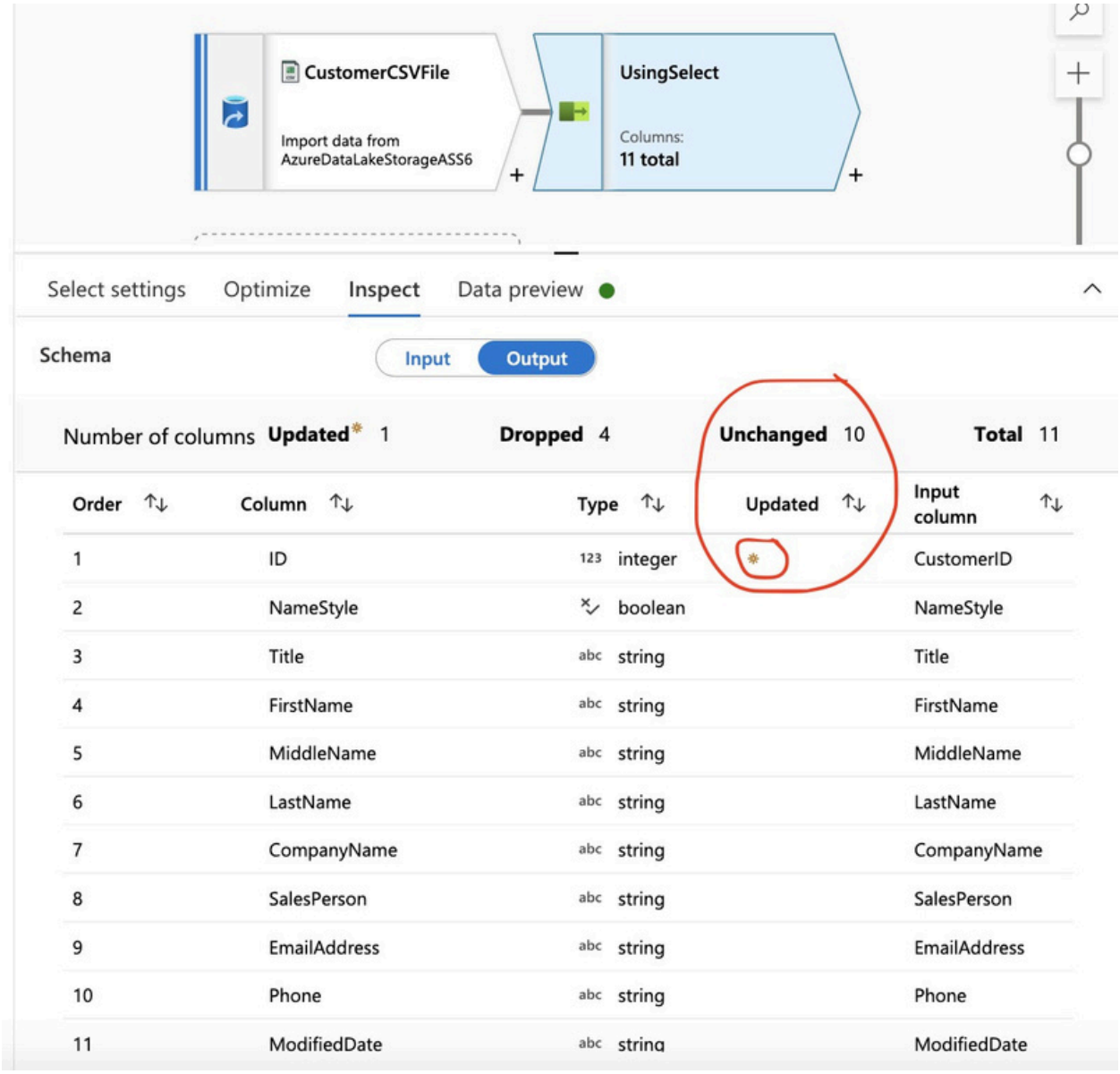
I selected 4 columns to be deleted, Now we can see 11 columns:



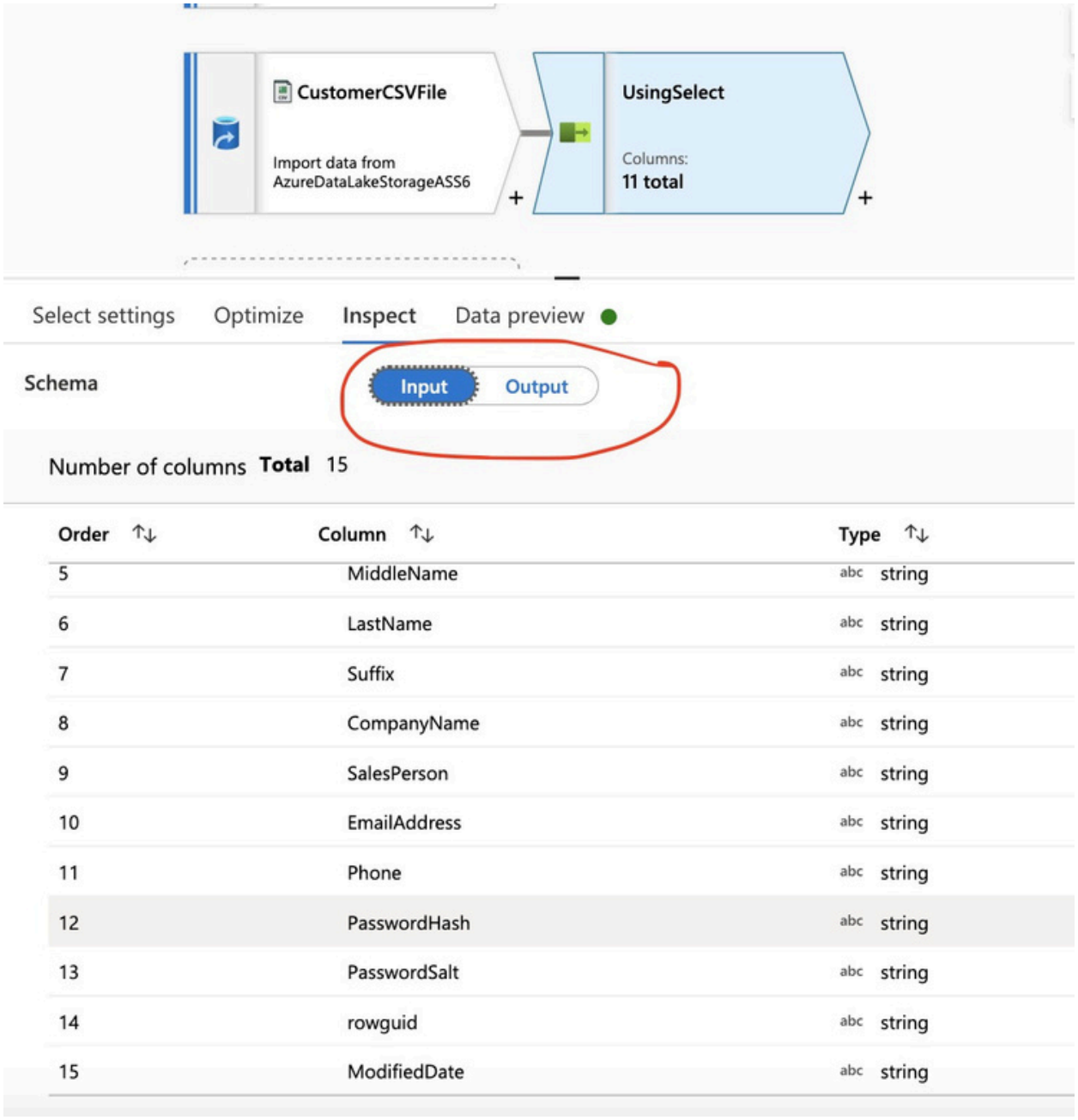
We can also change the column name as per the requirement:



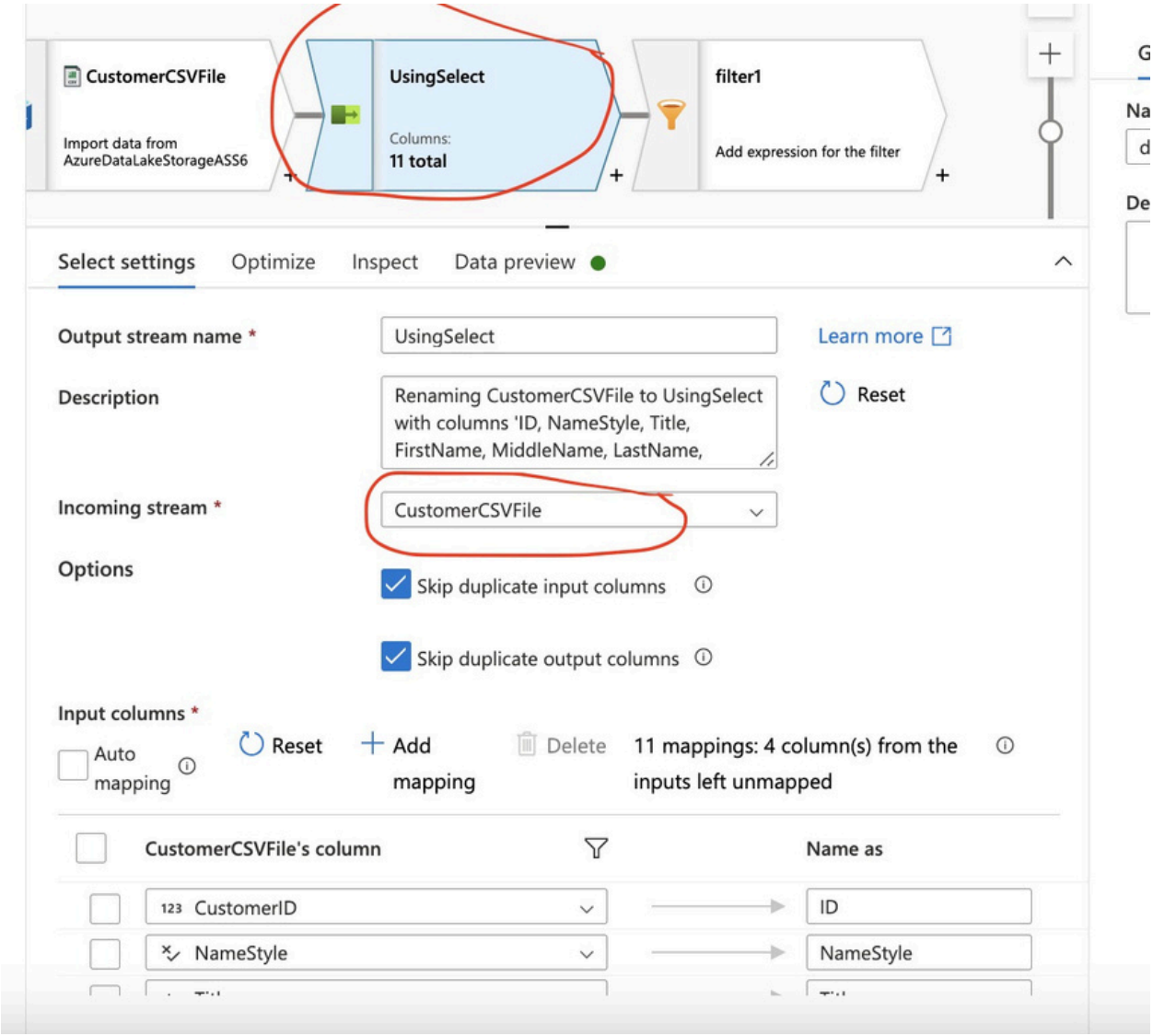
Updated Column , will show *, we can see dropped, updated and total in below screen shot:



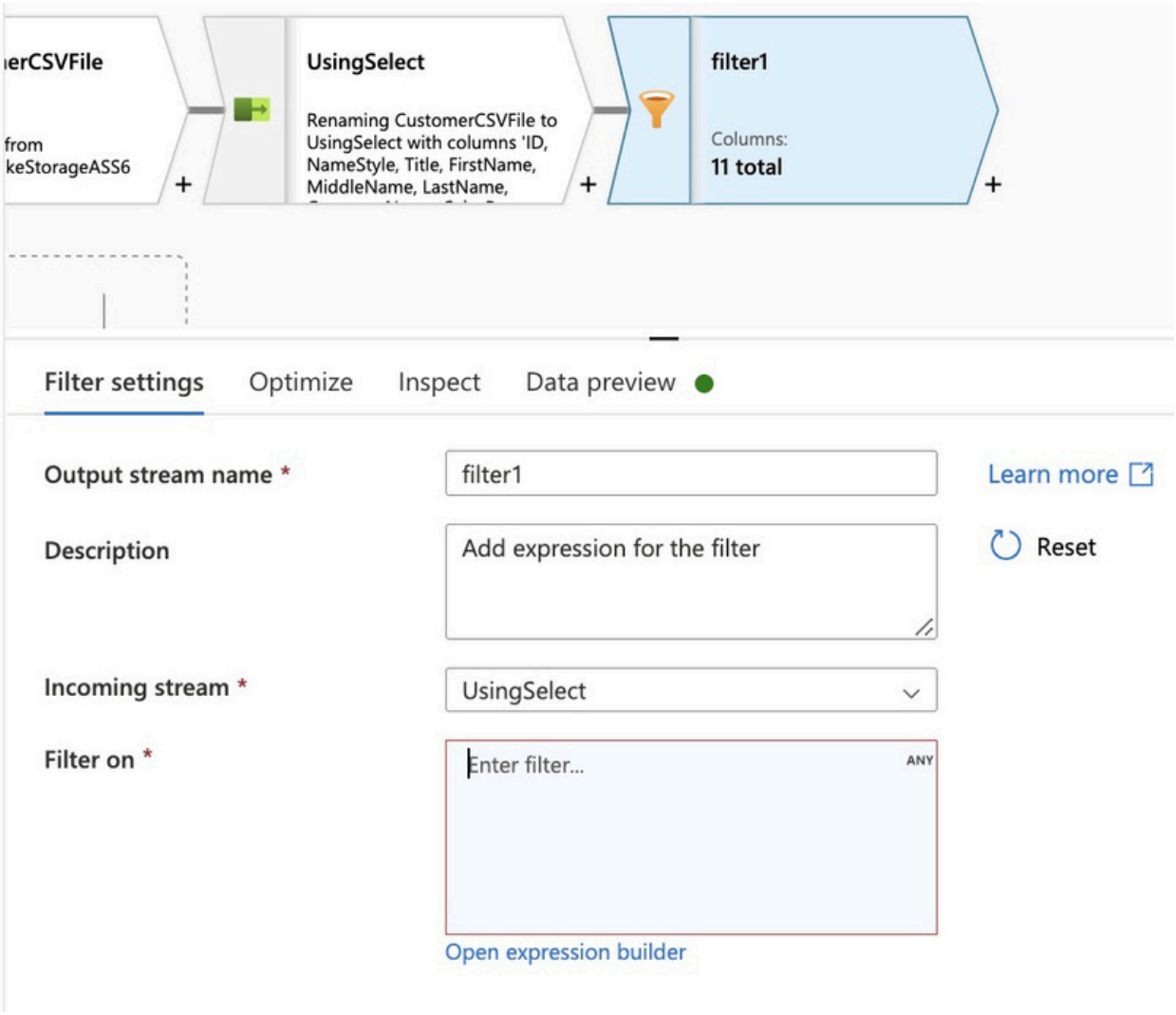
We can switch to input to check the input columns:

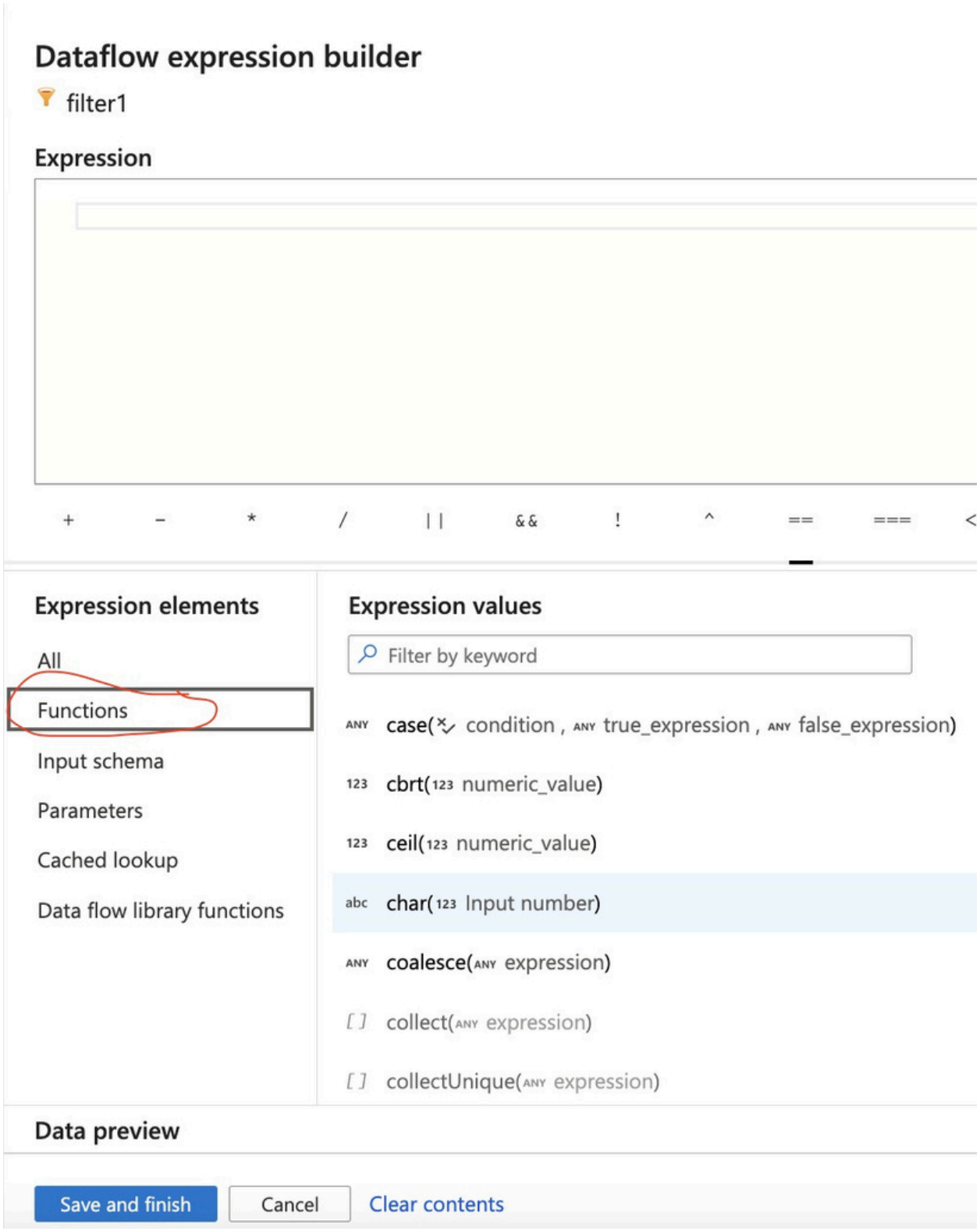


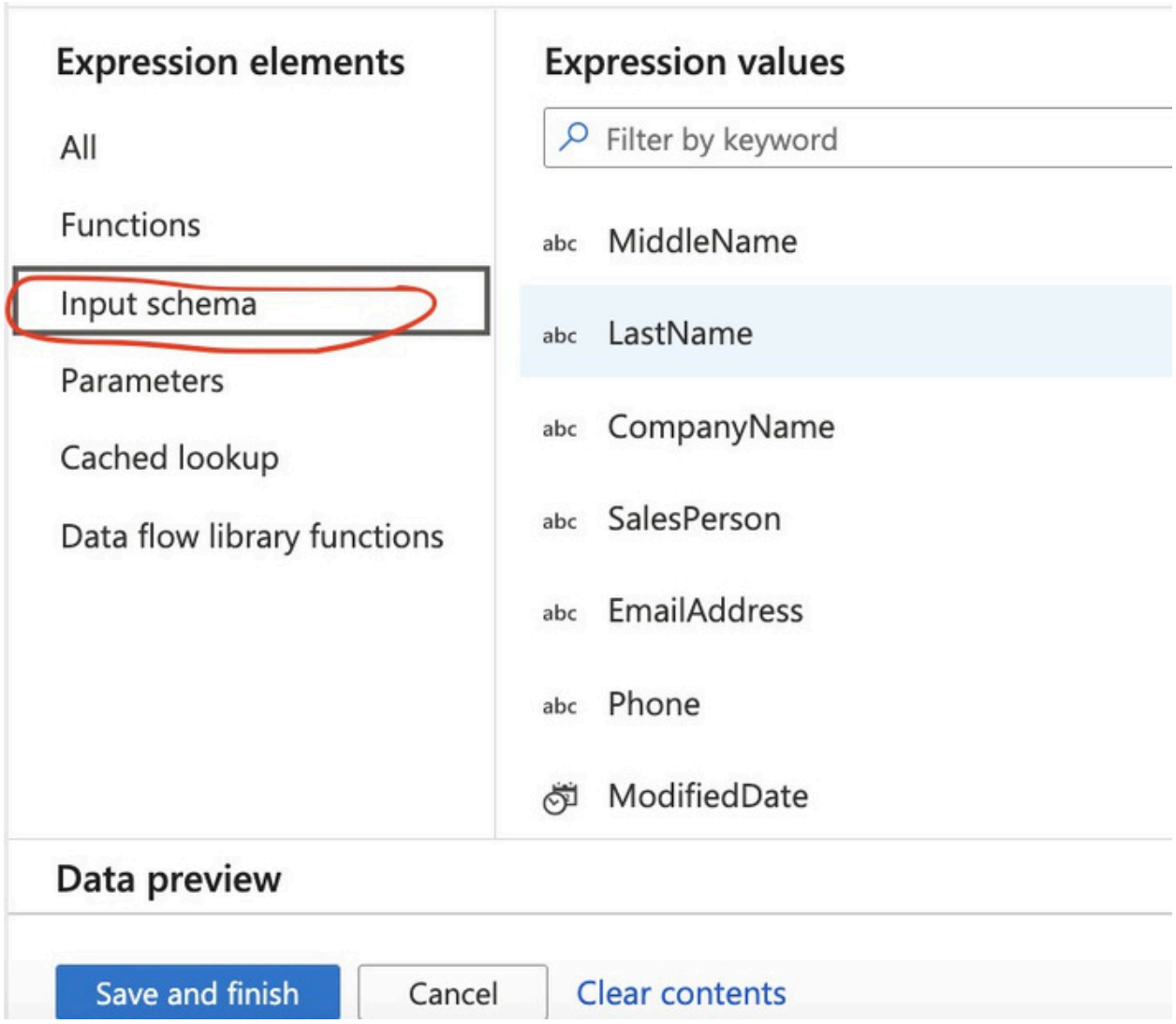
We can also change Incoming stream:



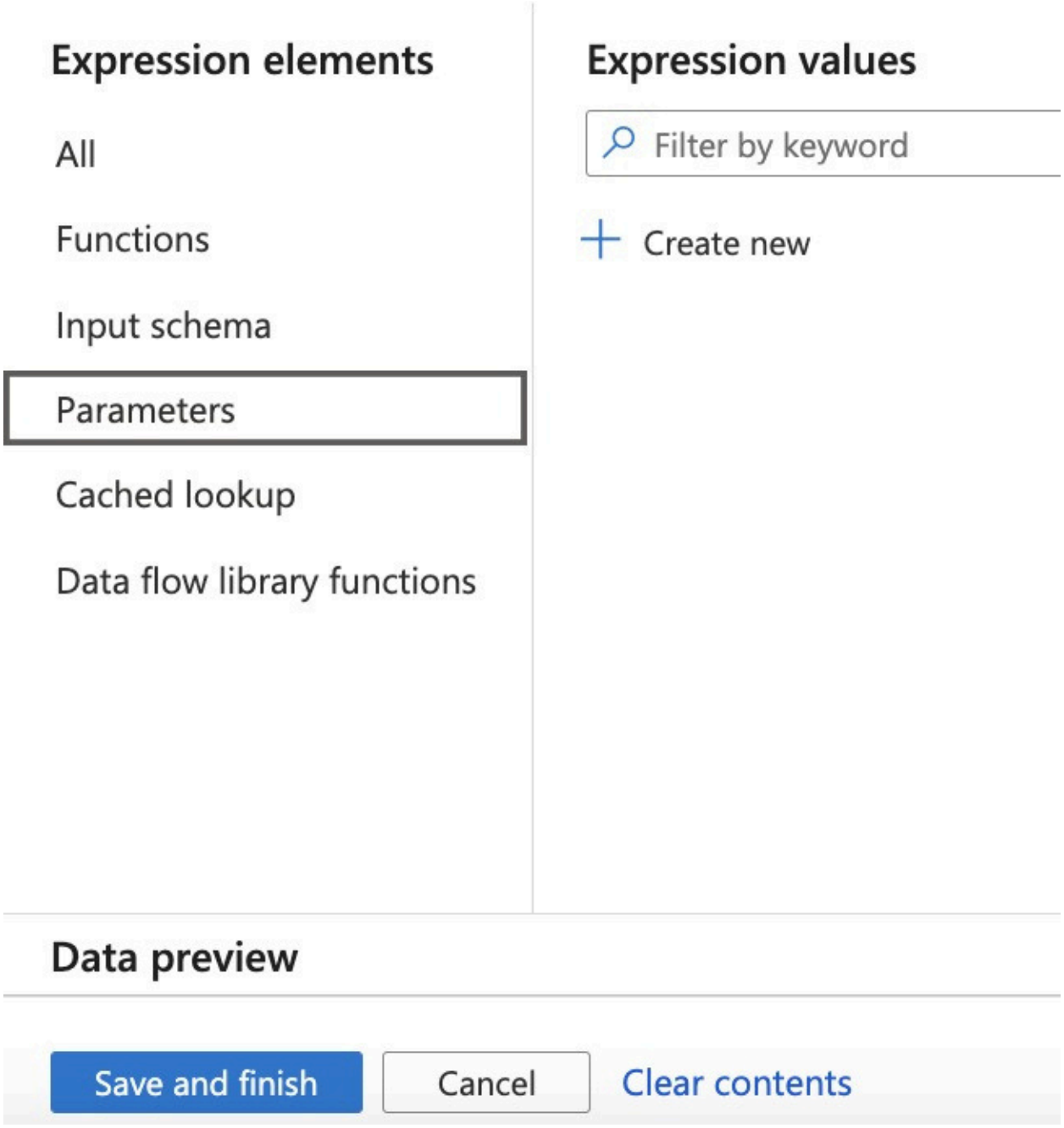
We want to filter LastName:Gates

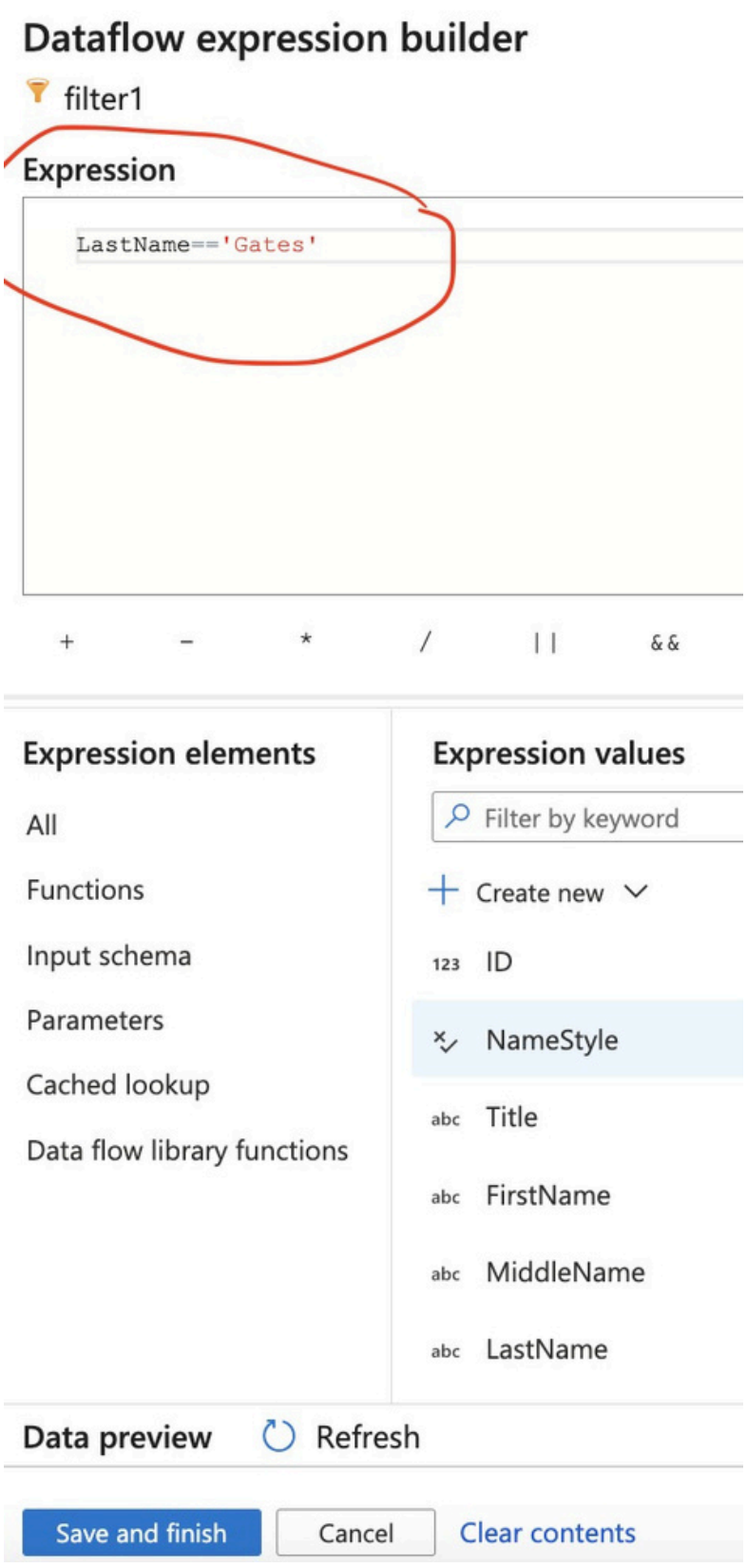



Dataflow expression builder
filter1
Expression
+ - * / || && ! ^ == === <
Expression elements
All
Functions
Input schema
Parameters
Cached lookup
Data flow library functions
Expression values
Filter by keyword
ANY case(condition , ANY true_expression , ANY false_expression)
123 cbrt(123 numeric_value)
123 ceil(123 numeric_value)
abc char(123 Input number)
ANY coalesce(ANY expression)
[ ] collect(ANY expression)
[ ] collectUnique(ANY expression)
Data preview
Save and finish
Cancel
Clear contents


Expression elements
All
Functions
Input schema
Parameters
Cached lookup
Data flow library functions
Expression values
Filter by keyword
abc MiddleName
abc LastName
abc CompanyName
abc SalesPerson
abc EmailAddress
abc Phone
ModifiedDate
Data preview
Save and finish
Cancel
Clear contents

We can create parameters at filter level:




Dataflow expression builder
filter1
Expression
LastName=='Gates'
+ - * / || &&
Expression elements
All
Functions
Input schema
Parameters
Cached lookup
Data flow library functions
Expression values
Filter by keyword
Create new
123 ID
NameStyle
abc Title
abc FirstName
abc MiddleName
abc LastName
Data preview
Refresh
Save and finish
Cancel
Clear contents

Dataflow expression builder

filterLastName

Expression

```
LastName=='Gates'
```

+ - * / || &&

Expression elements

All

Functions

Input schema

Parameters

Cached lookup

Data flow library functions

Expression values

Filter by keyword

Create new

123 ID

NameStyle

abc Title

abc FirstName

abc MiddleName

abc LastName

Data preview Refresh

Save and finish | Cancel | Clear contents

Dataflow expression builder

filterLastName

Expression

```
LastName=='Gates'
```

+ - * / || && ! ^ == === <=>

Expression elements

All

Functions

Input schema

Parameters

Cached lookup

Data flow library functions

Expression values

Filter by keyword

Create new

123 ID

NameStyle

abc Title

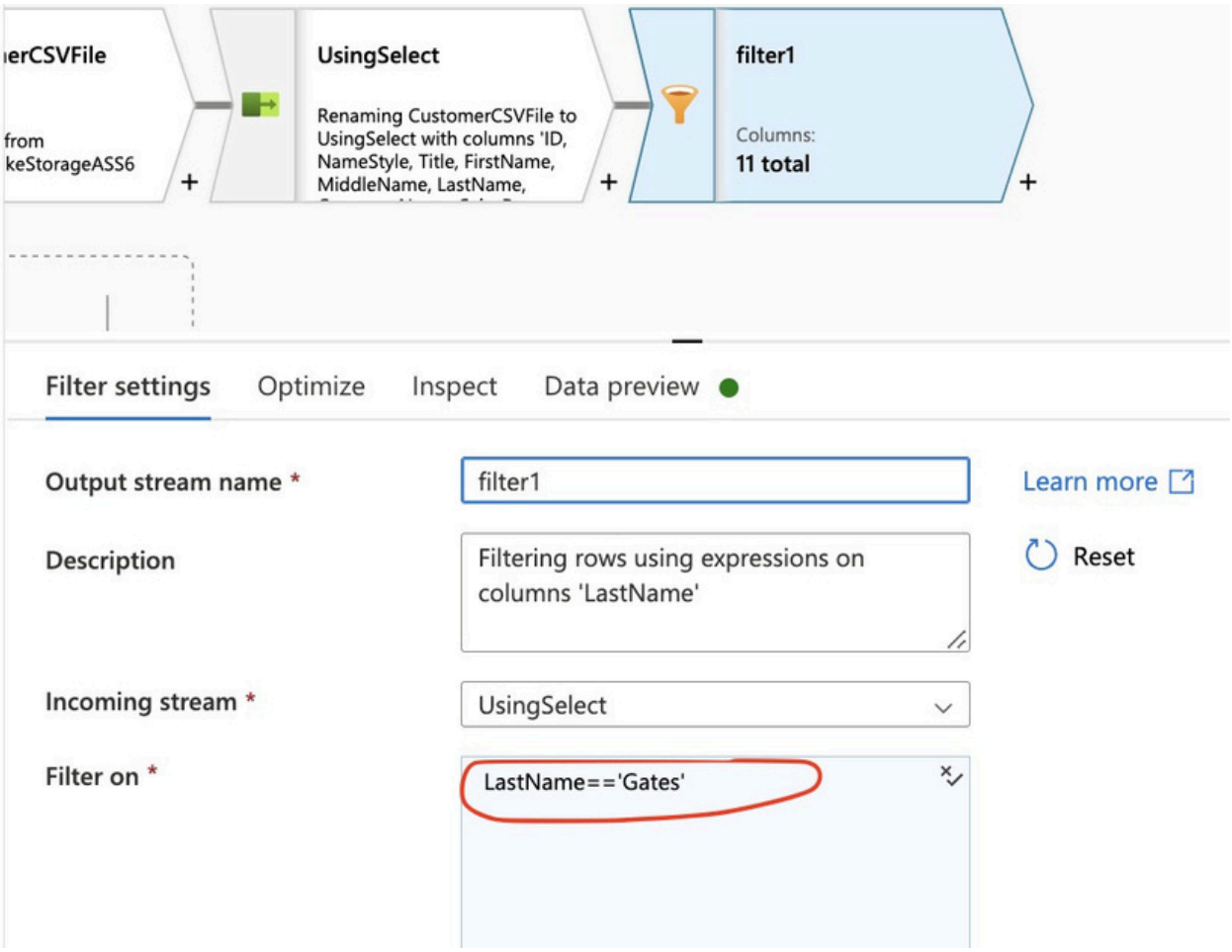
abc FirstName

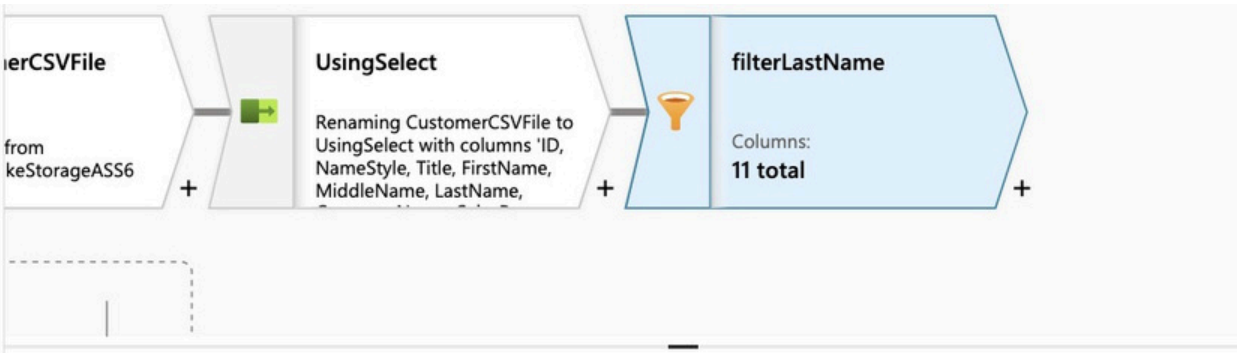
Data preview

Refresh

Output:	LastName abc
✗	Gee
✗	Harris
✗	Carreras
✓	Gates

Save and finish | Cancel | Clear contents

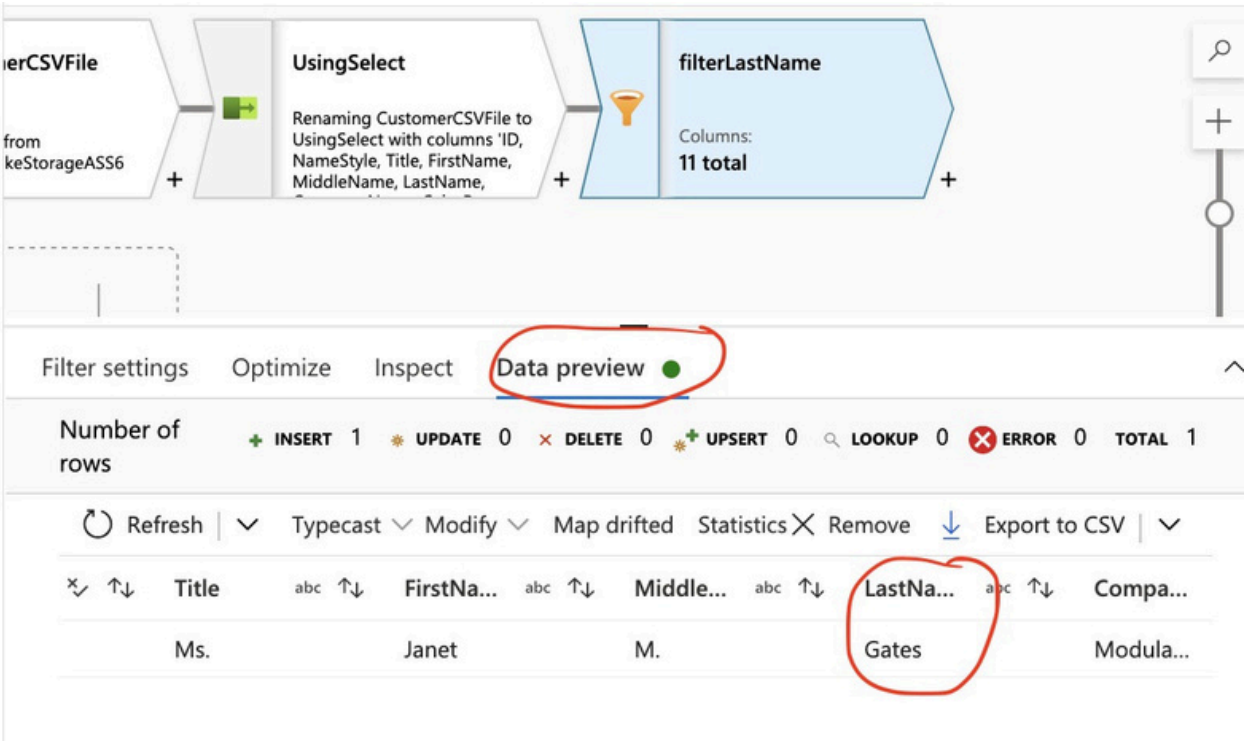

erCSVFile
from
keStorageASS6
UsingSelect
Renaming CustomerCSVFile to UsingSelect with columns 'ID, NameStyle, Title, FirstName, MiddleName, LastName,
filter1
Columns:
11 total
Filter settings
Optimize
Inspect
Data preview
Output stream name *
filter1
Learn more
Description
Filtering rows using expressions on columns 'LastName'
Reset
Incoming stream *
UsingSelect
Filter on *
LastName=='Gates'



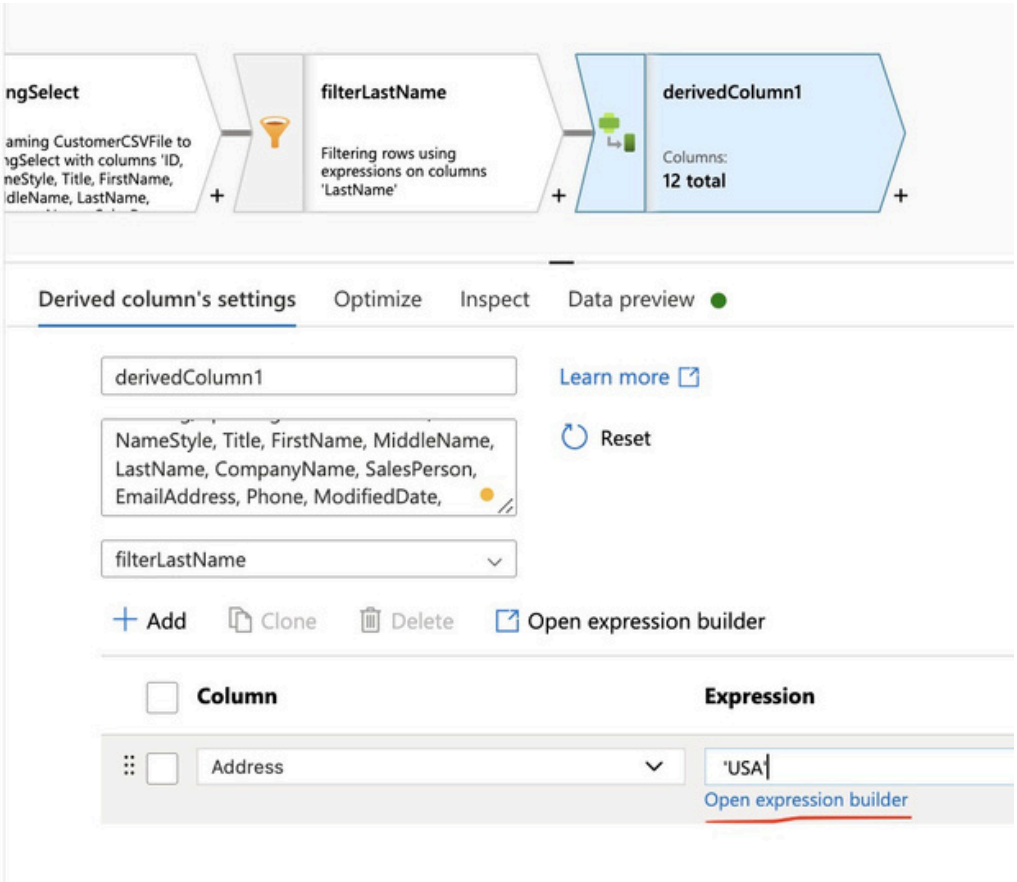
Filter settings | Optimize | **Inspect** | Data preview

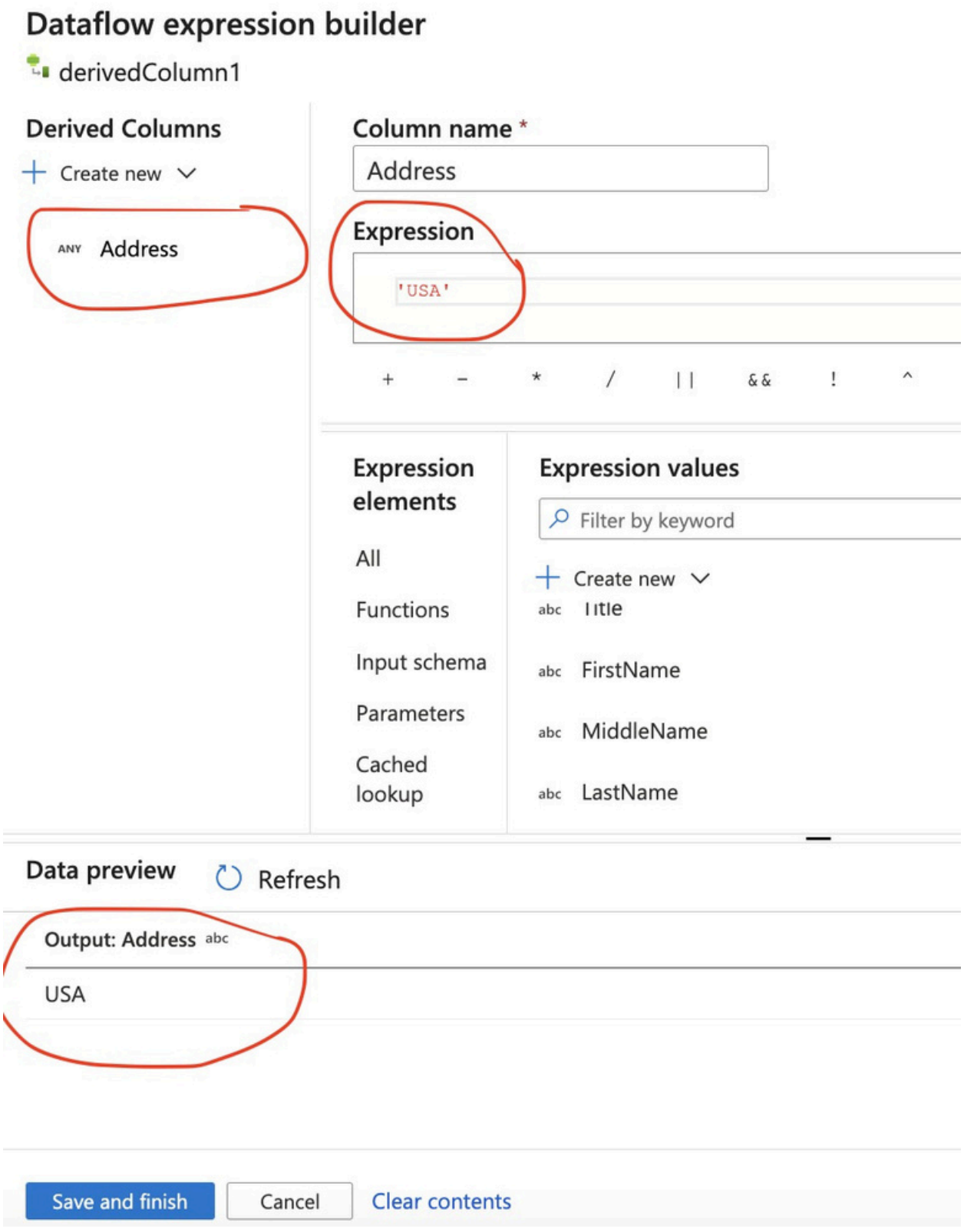
Number of columns **Total** 11

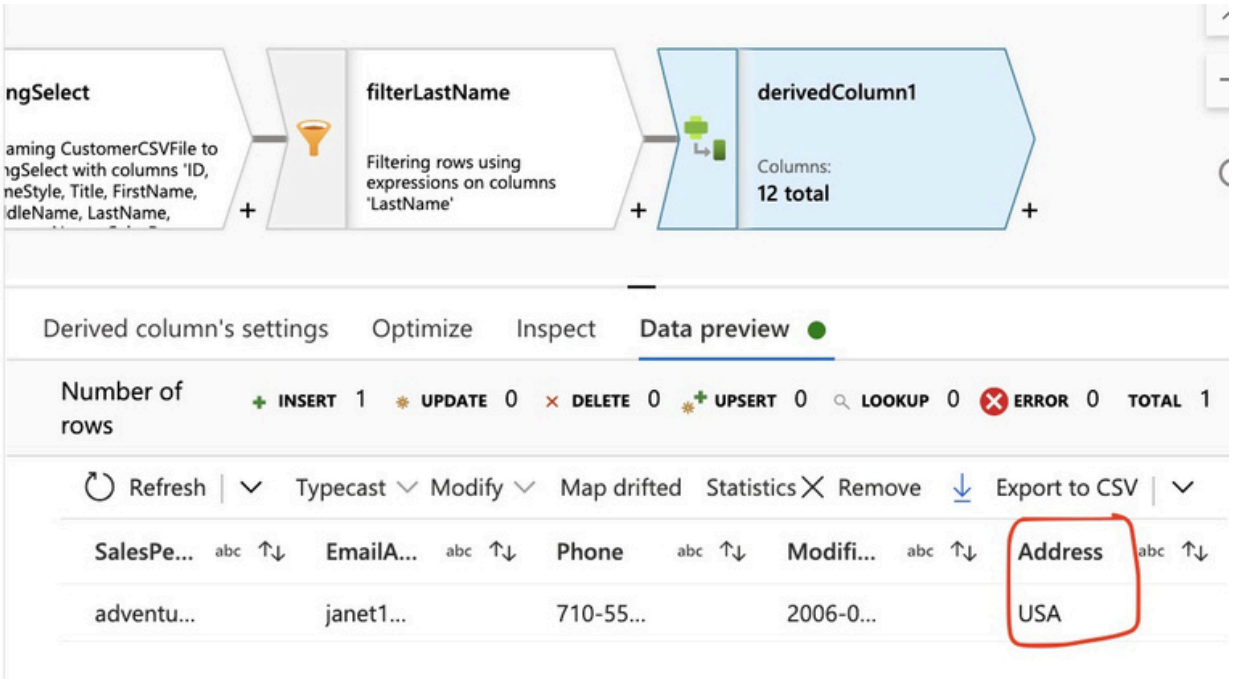
Order	Column	Type	Used in Filter
1	ID	123 integer	
2	NameStyle	boolean	
3	Title	abc string	
4	FirstName	abc string	
5	MiddleName	abc string	
6	LastName	abc string	✓
7	CompanyName	abc string	
8	SalesPerson	abc string	
9	EmailAddress	abc string	
10	Phone	abc string	
11	ModifiedDate	timestamp	



Click on derived Column , If we want to add column of our own:

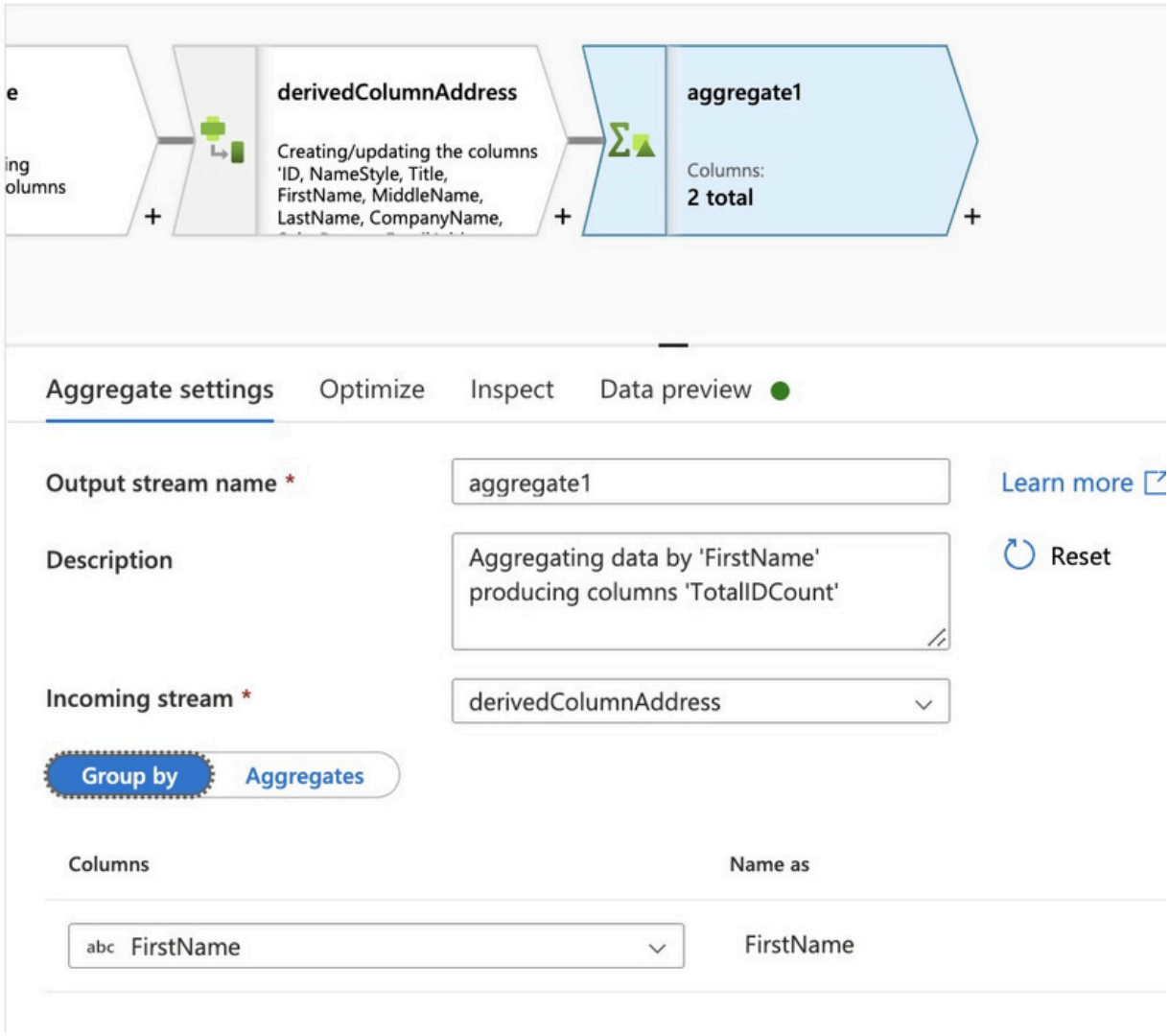



Dataflow expression builder
derivedColumn1
Derived Columns
Create new
ANY Address
Column name *
Address
Expression
'USA'
+ - * / || && ! ^
Expression elements
All
Functions
Input schema
Parameters
Cached lookup
Expression values
Filter by keyword
Create new
abc Title
abc FirstName
abc MiddleName
abc LastName
Data preview
Refresh
Output: Address abc
USA
Save and finish
Cancel
Clear contents

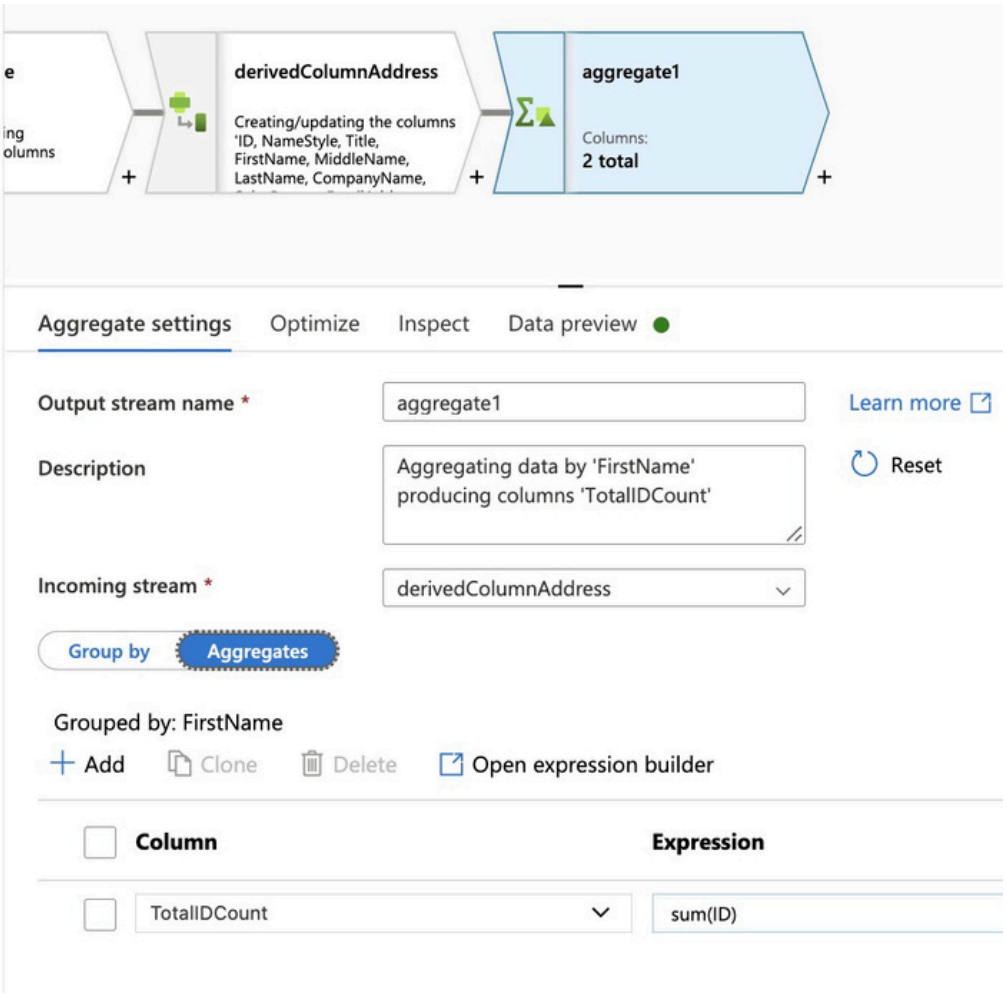

ngSelect
aming CustomerCSVFile to
ngSelect with columns 'ID,
neStyle, Title, FirstName,
dleName, LastName,
filterLastName
Filtering rows using expressions on columns 'LastName'
derivedColumn1
Columns:
12 total
Derived column's settings
Optimize
Inspect
Data preview
Number of rows
INSERT 1
UPDATE 0
DELETE 0
UPSERT 0
LOOKUP 0
ERROR 0
TOTAL 1
Refresh
Typecast
Modify
Map drifted
Statistics
Remove
Export to CSV
SalesPe...
EmailA...
Phone
Modifi...
Address
adventu...
janet1...
710-55...
2006-0...
USA

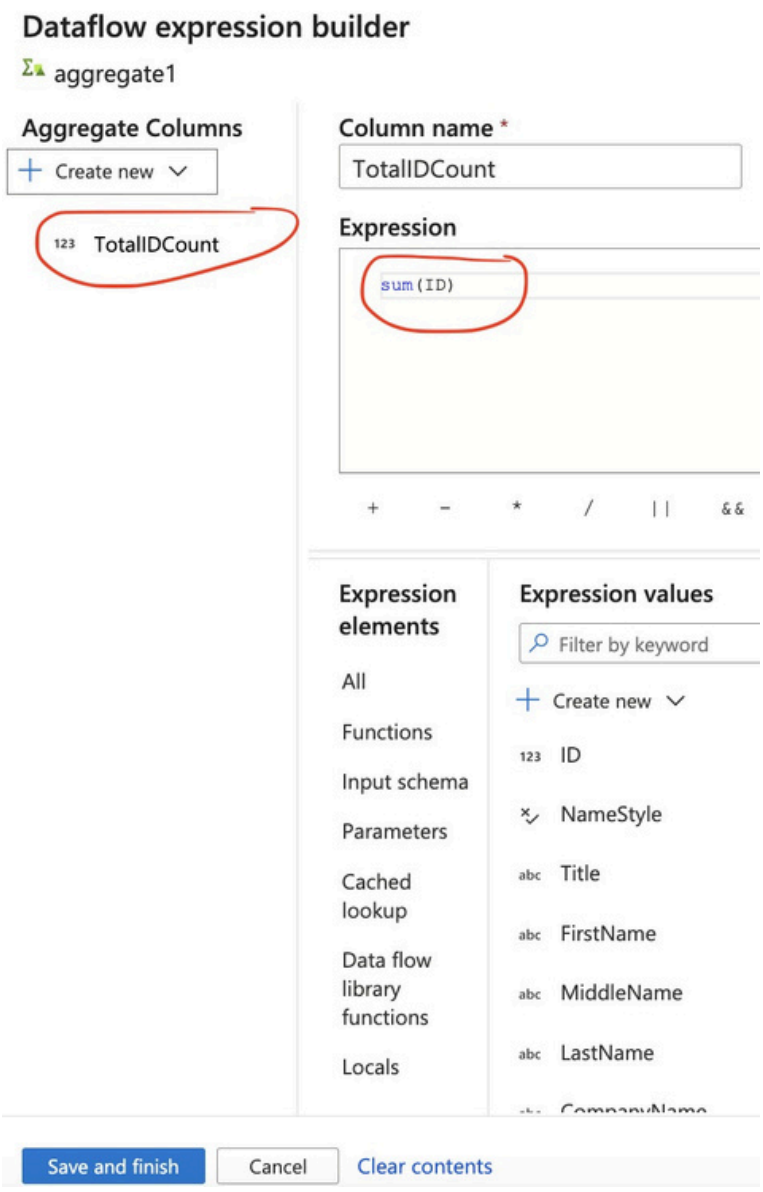
Aggregate :

GroupBY “FirstName”

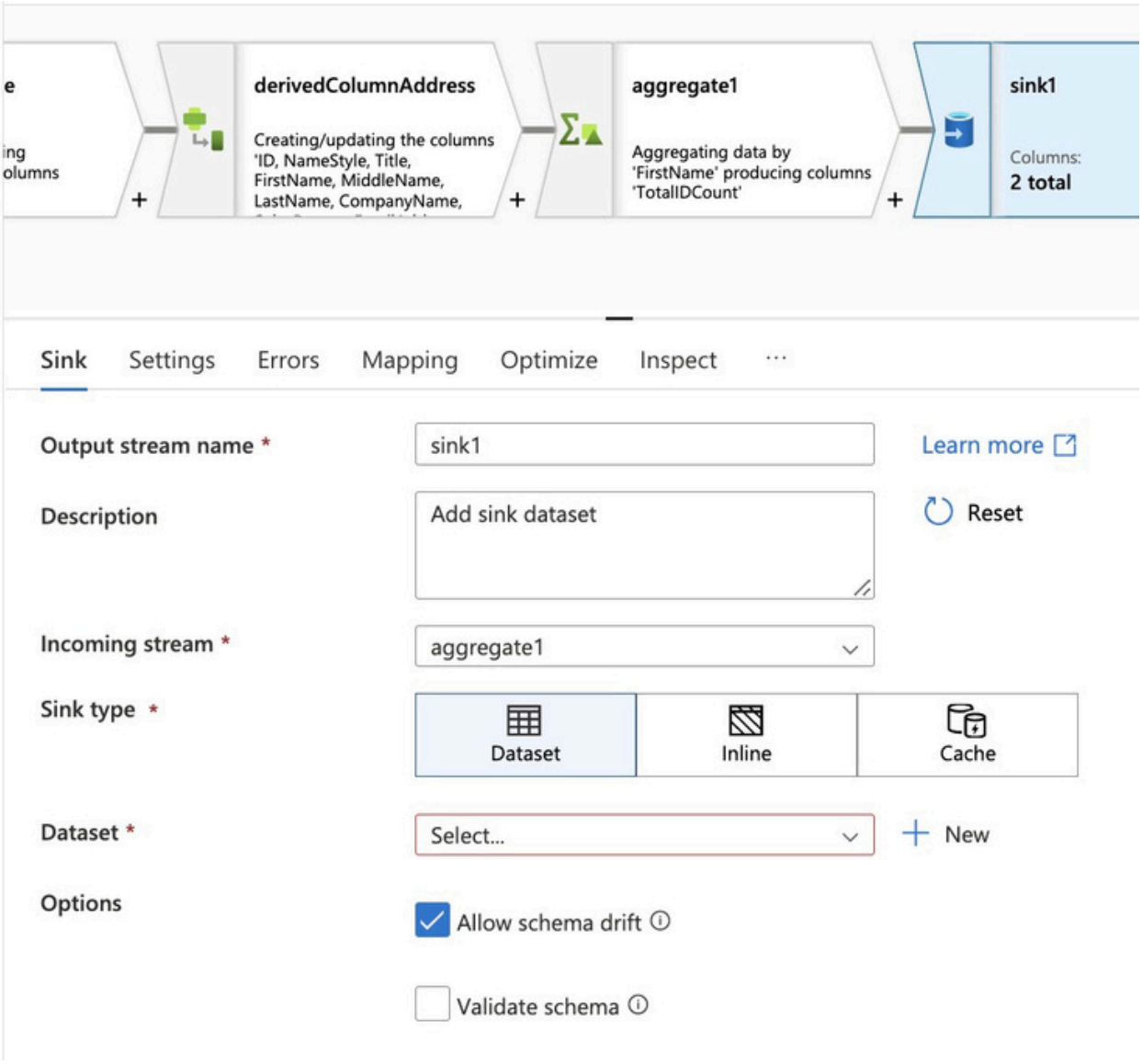


For Example: In aggregates to change columnName and do sum(ID):

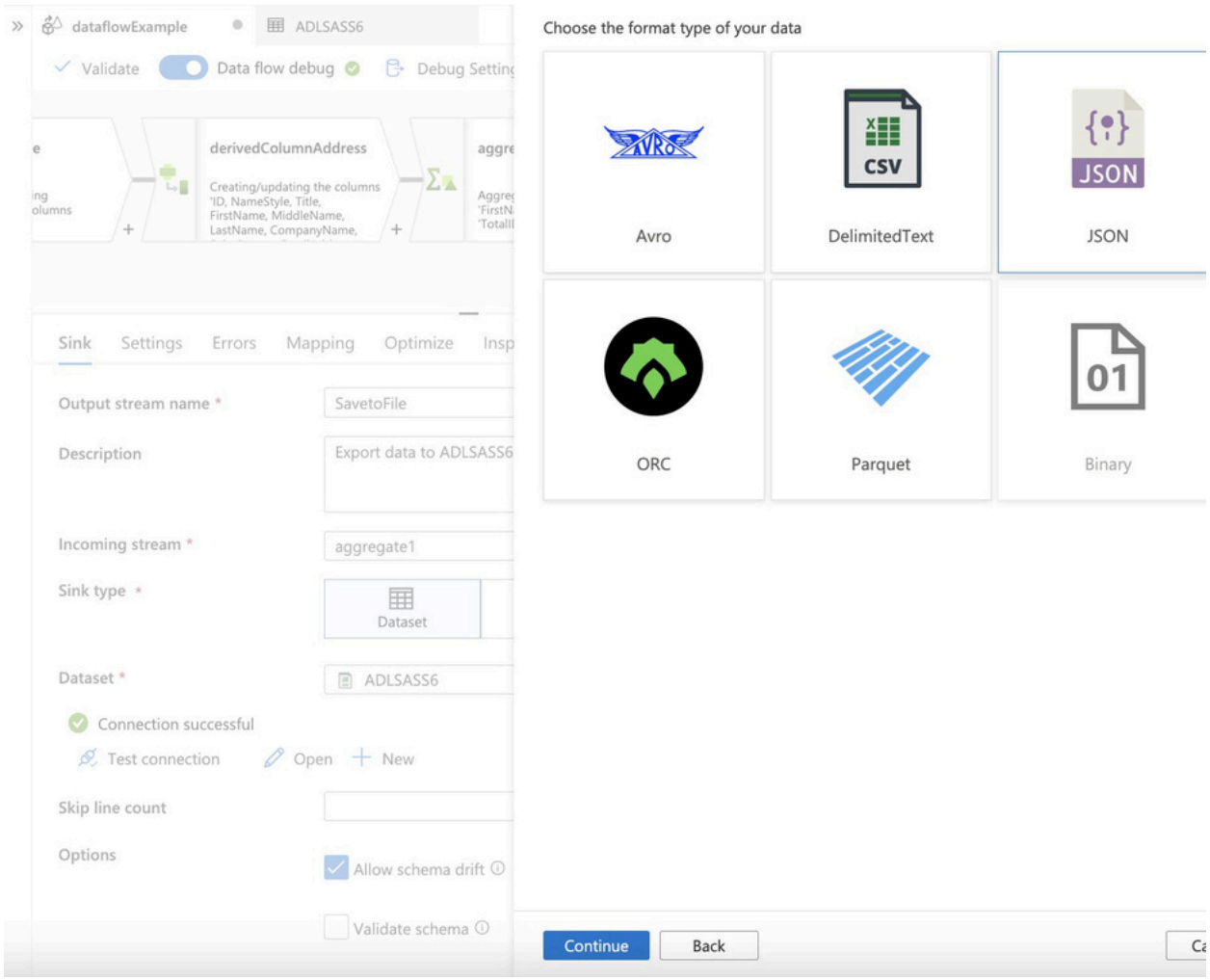

e
ing
olumns
derivedColumnAddress
Creating/updating the columns 'ID, NameStyle, Title, FirstName, MiddleName, LastName, CompanyName,
aggregate1
Columns:
2 total
Aggregate settings
Optimize
Inspect
Data preview
Output stream name *
aggregate1
Learn more
Description
Aggregating data by 'FirstName' producing columns 'TotalIDCount'
Reset
Incoming stream *
derivedColumnAddress
Group by
Aggregates
Grouped by: FirstName
Add
Clone
Delete
Open expression builder
Column
Expression
TotalIDCount
sum(ID)


Dataflow expression builder
aggregate1
Aggregate Columns
Create new
TotalIDCount
Column name *
TotalIDCount
Expression
sum(ID)
+ - * / || &&
Expression elements
All
Functions
Input schema
Parameters
Cached lookup
Data flow library functions
Locals
Expression values
Filter by keyword
Create new
ID
NameStyle
Title
FirstName
MiddleName
LastName
Save and finish
Cancel
Clear contents

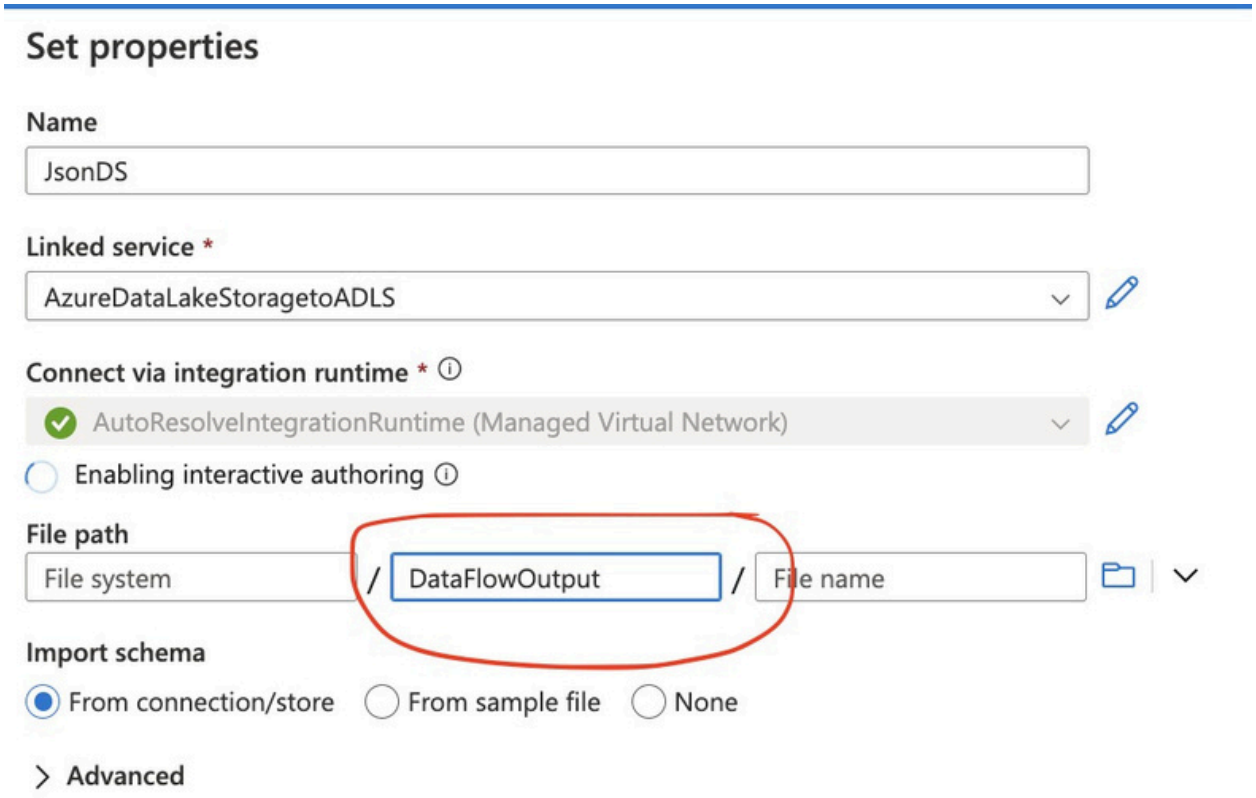
We can go and write all the transformations we have done so far in **sink**:

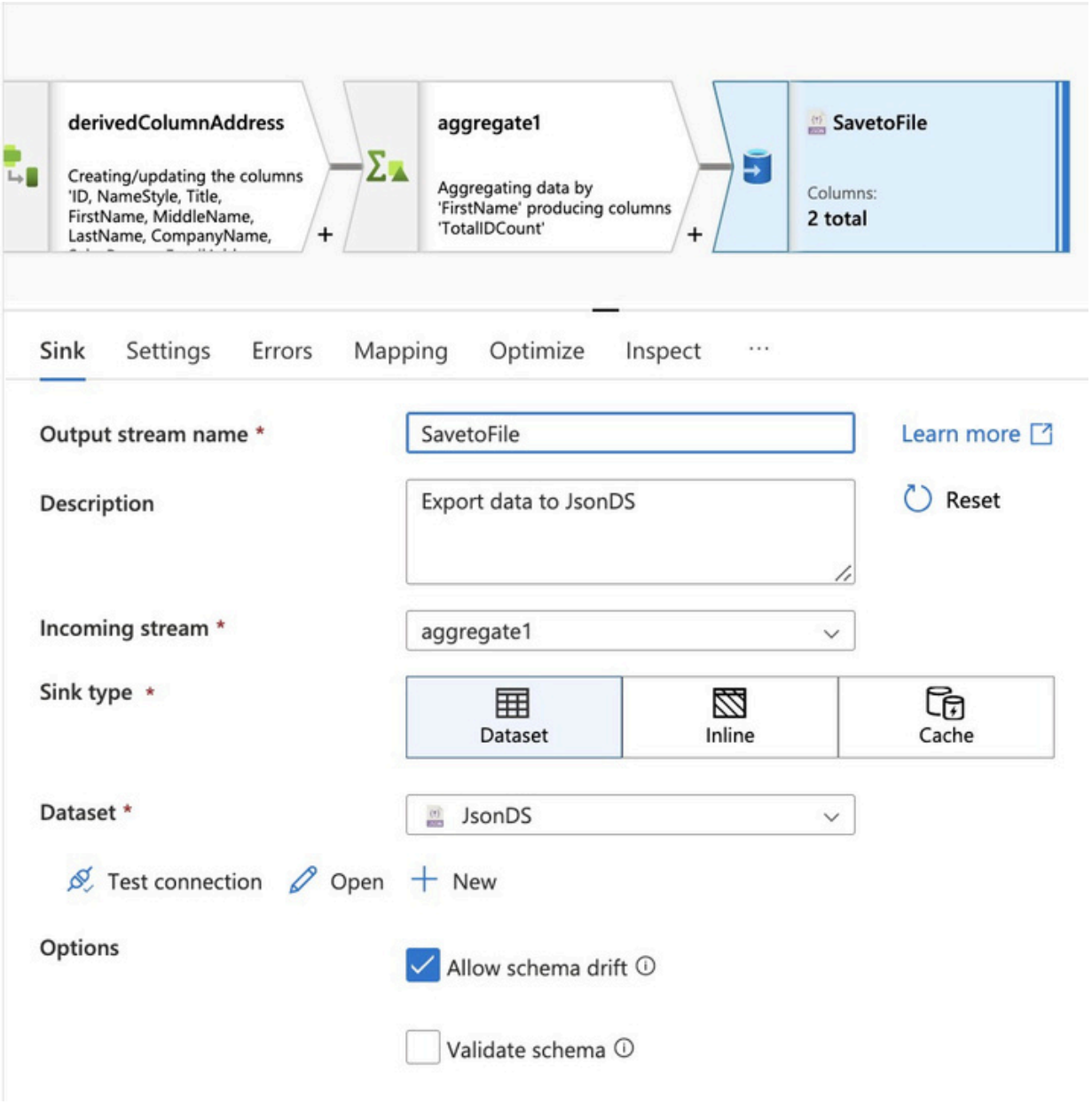


Click on new dataset->ADLS->JSON(If you want to store in JSON format):

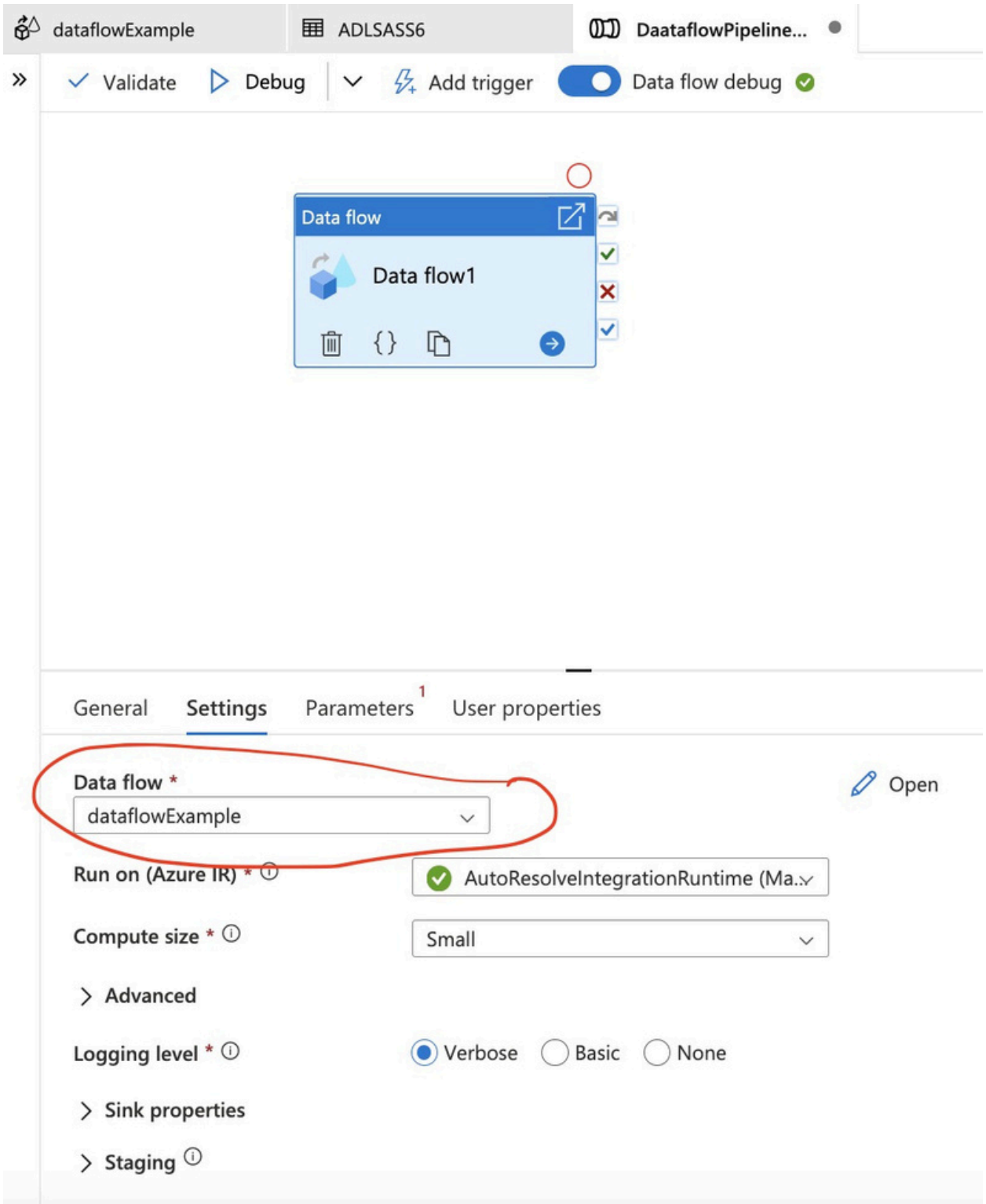


To store into one particular folder, select the directory Name:

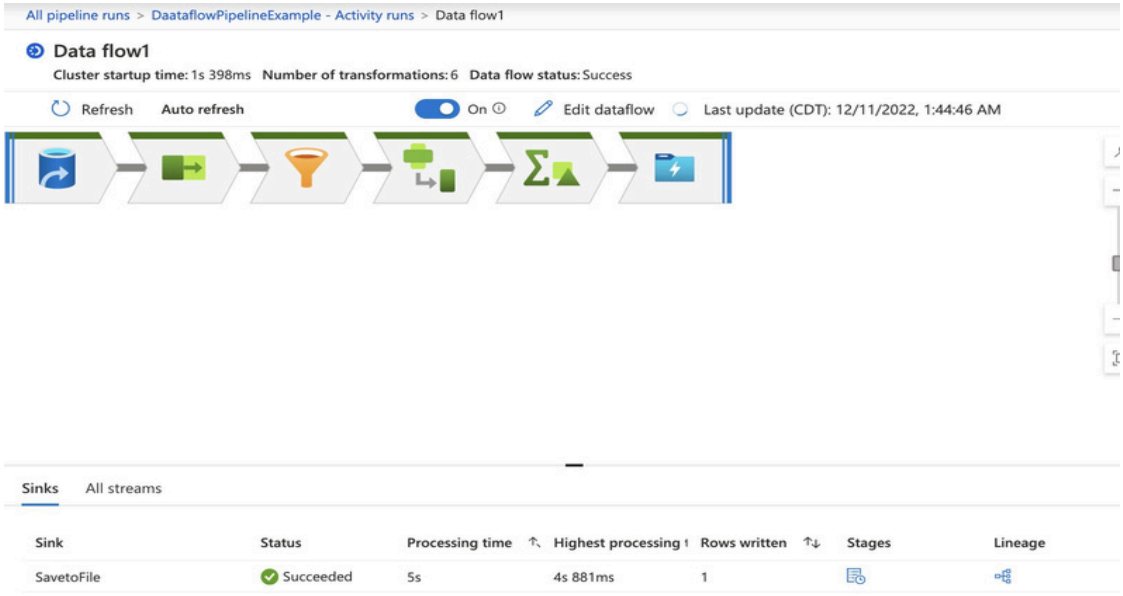



derivedColumnAddress
Creating/updating the columns 'ID, NameStyle, Title, FirstName, MiddleName, LastName, CompanyName,
aggregate1
Aggregating data by 'FirstName' producing columns 'TotalIDCount'
SavetoFile
Columns:
2 total
Sink
Settings
Errors
Mapping
Optimize
Inspect
Output stream name *
SavetoFile
Learn more
Description
Export data to JsonDS
Reset
Incoming stream *
aggregate1
Sink type *
Dataset
Inline
Cache
Dataset *
JsonDS
Test connection
Open
New
Options
Allow schema drift
Validate schema

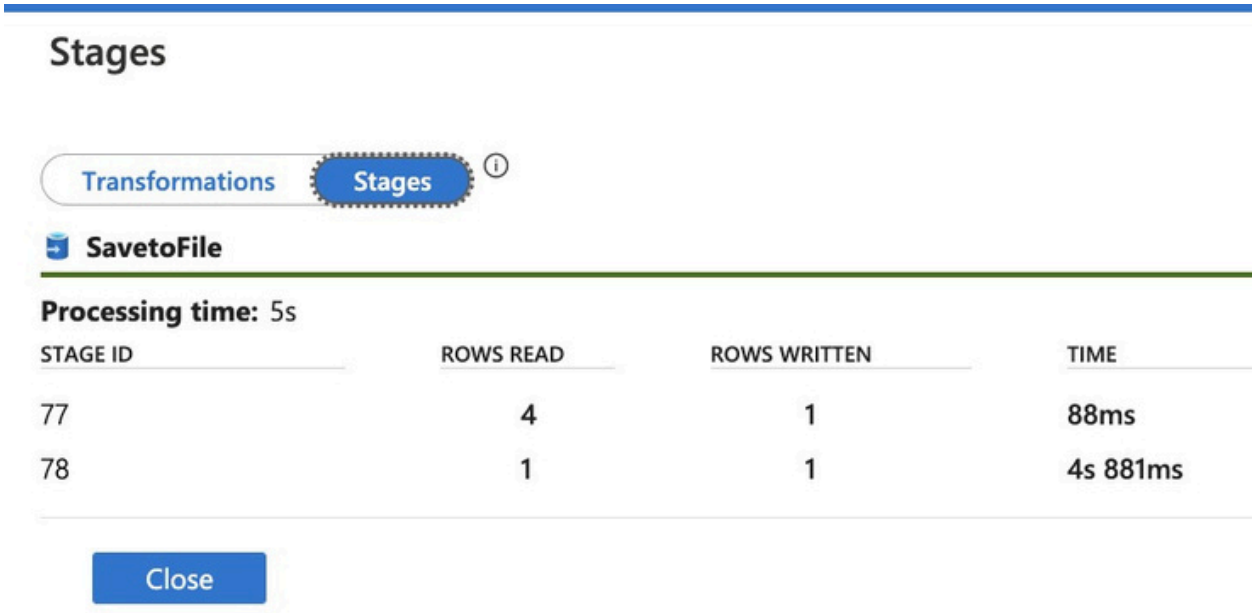
Dataflow itself is a complete Activity, so create pipeline to run this dataflow:



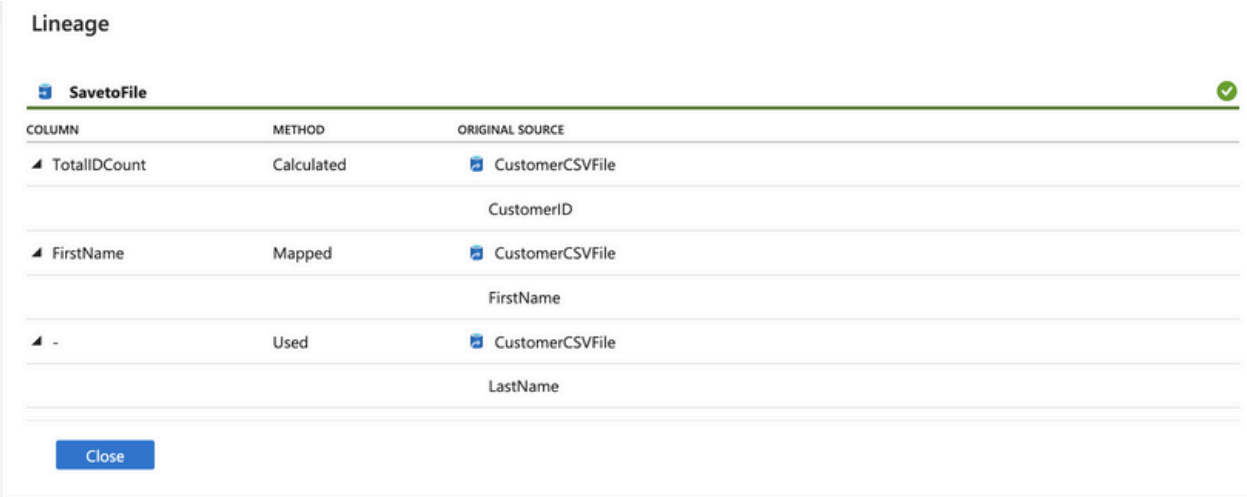
Dataflow debug and Inspect:

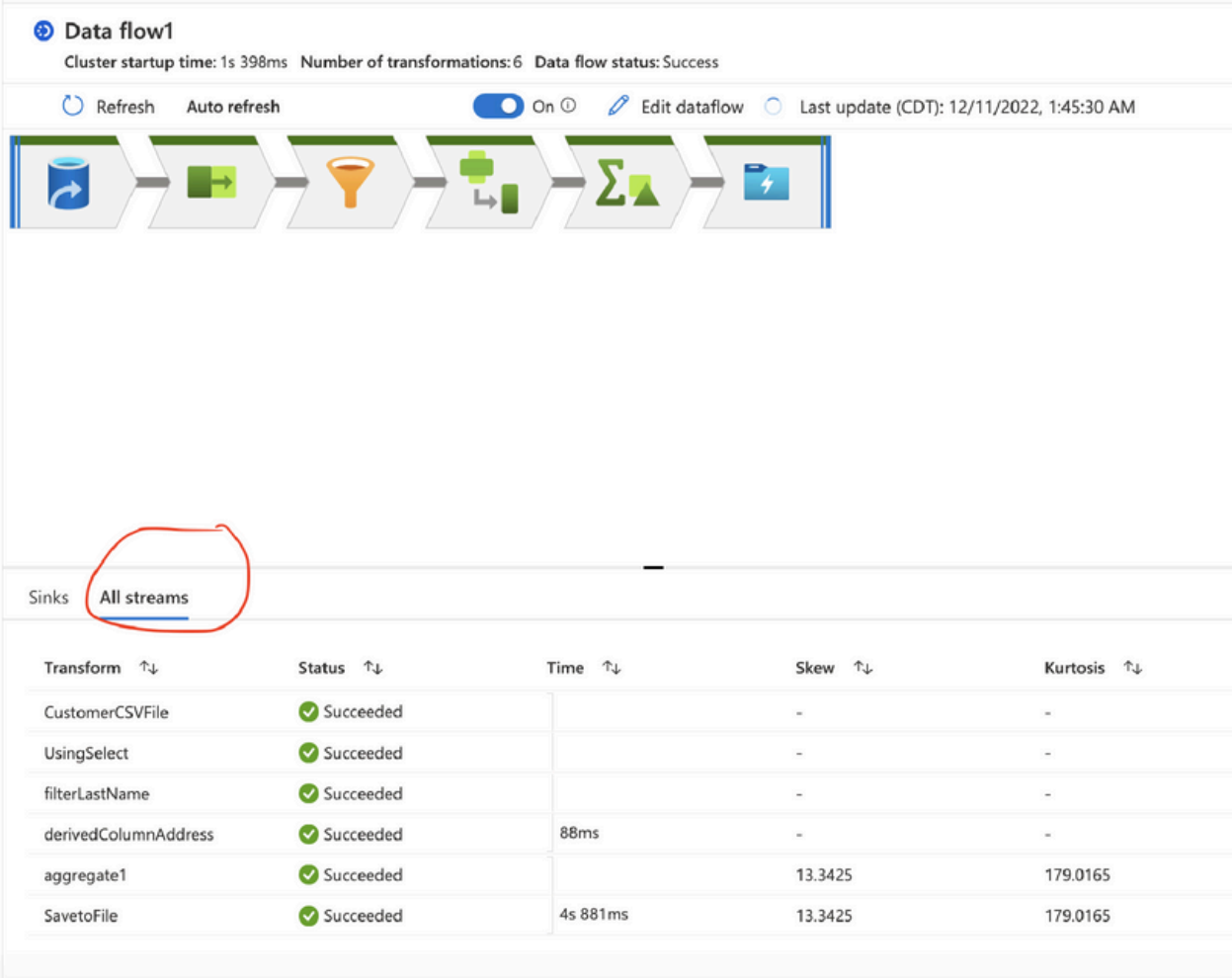


Click on stages:

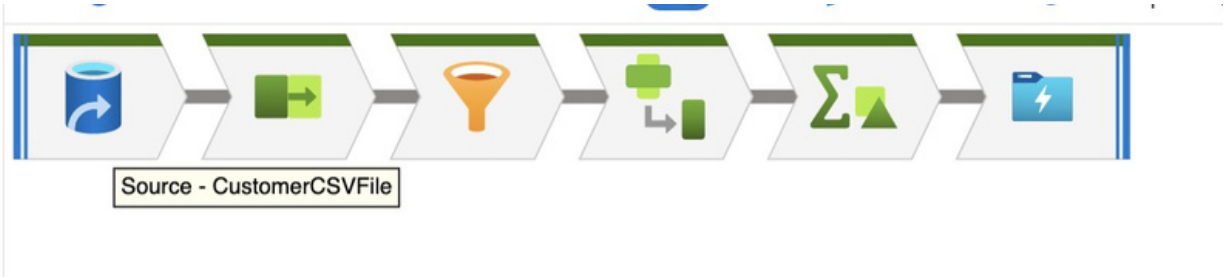


Lineage:

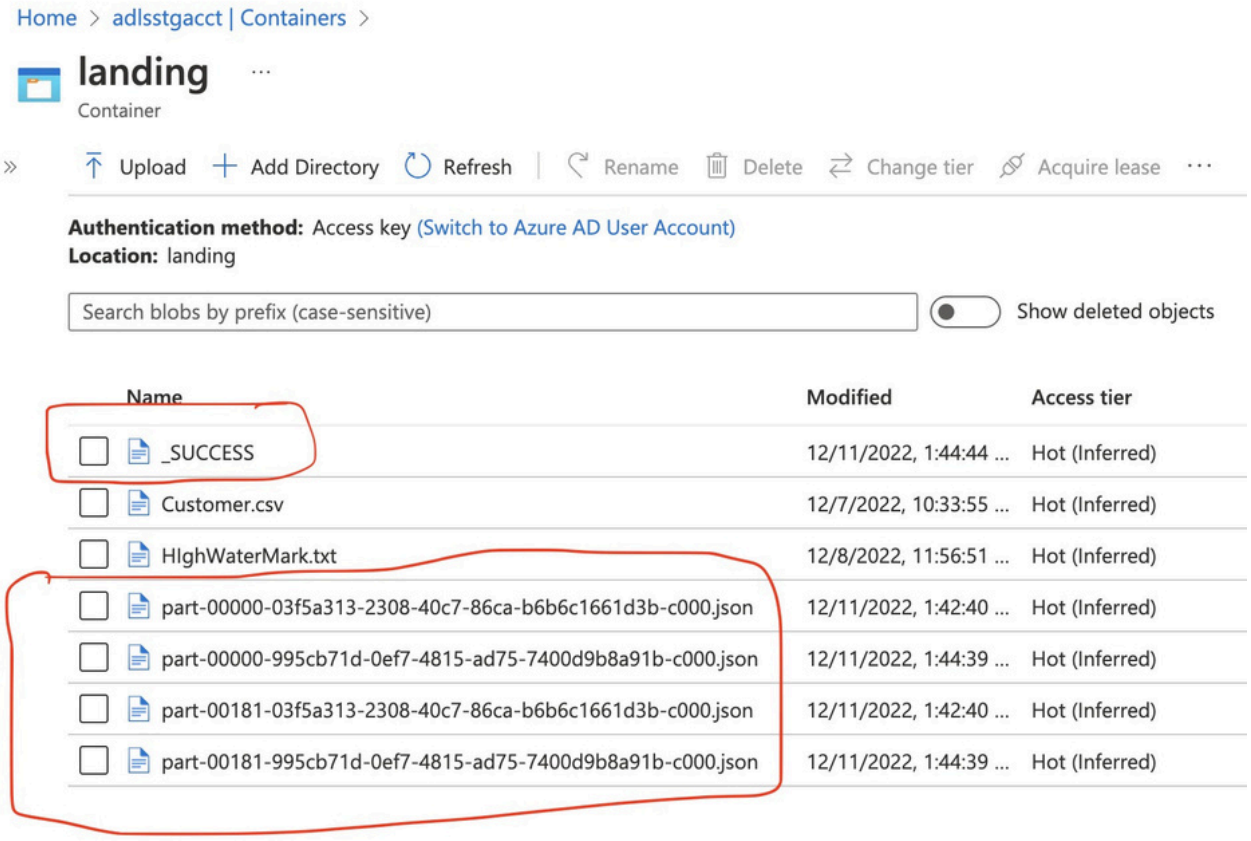

Lineage
SavetoFile
COLUMN
METHOD
ORIGINAL SOURCE
TotalIDCount
Calculated
CustomerCSVFile
CustomerID
FirstName
Mapped
CustomerCSVFile
FirstName
-
Used
CustomerCSVFile
LastName
Close


Data flow1
Cluster startup time: 1s 398ms
Number of transformations: 6
Data flow status: Success
Refresh
Auto refresh
On
Edit dataflow
Last update (CDT): 12/11/2022, 1:45:30 AM
Sinks
All streams
Transform
Status
Time
Skew
Kurtosis
CustomerCSVFile
Succeeded
-
-
UsingSelect
Succeeded
-
-
filterLastName
Succeeded
-
-
derivedColumnAddress
Succeeded
88ms
-
-
aggregate1
Succeeded
13.3425
179.0165
SavetoFile
Succeeded
4s 881ms
13.3425
179.0165

Hover on each dataflow to see the transformation , we have used.

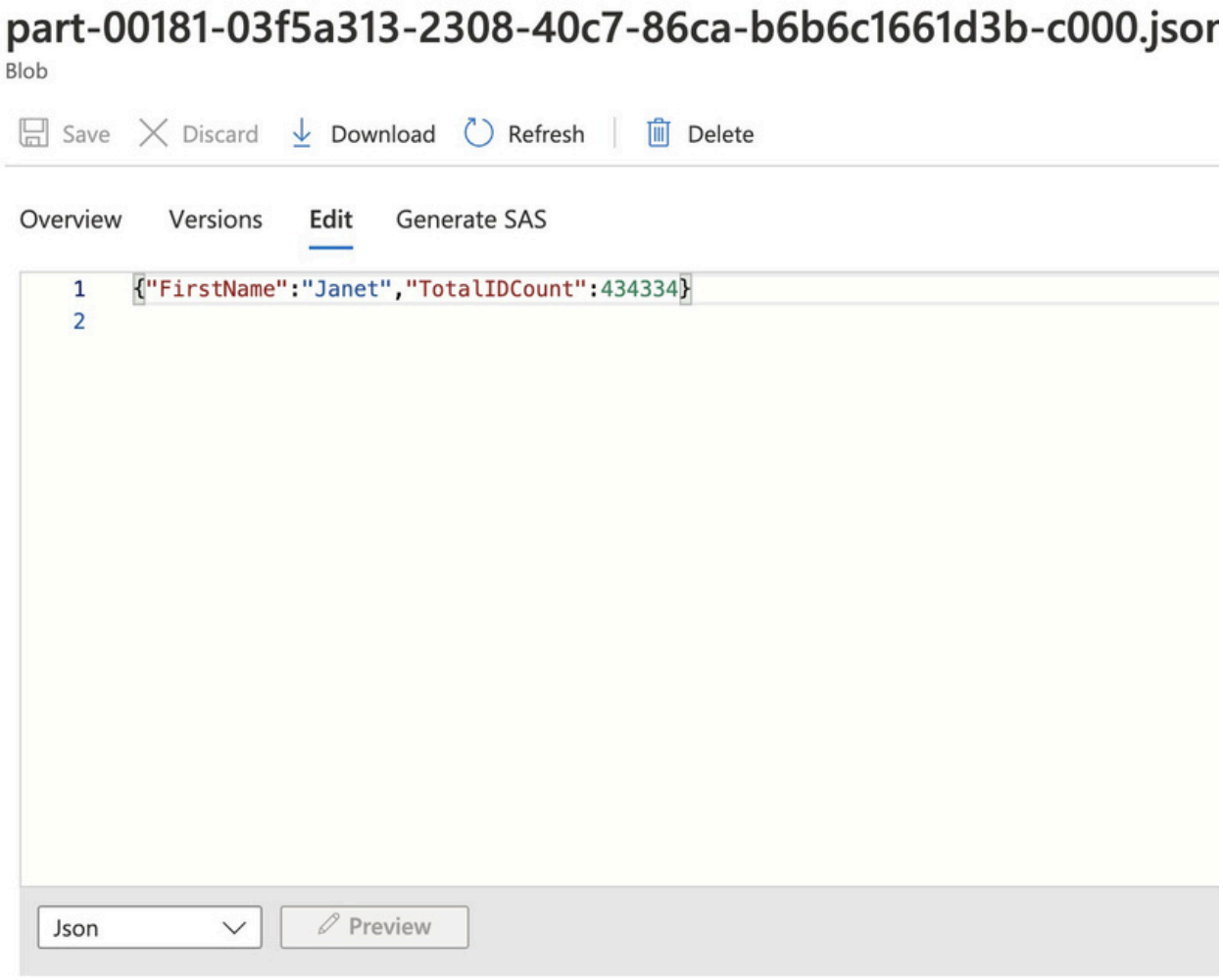


Output:



NOTE: some parts will be empty as it is running in parallel.

Output has more files



Dataflow data successfully copied into ADLS after transformations .