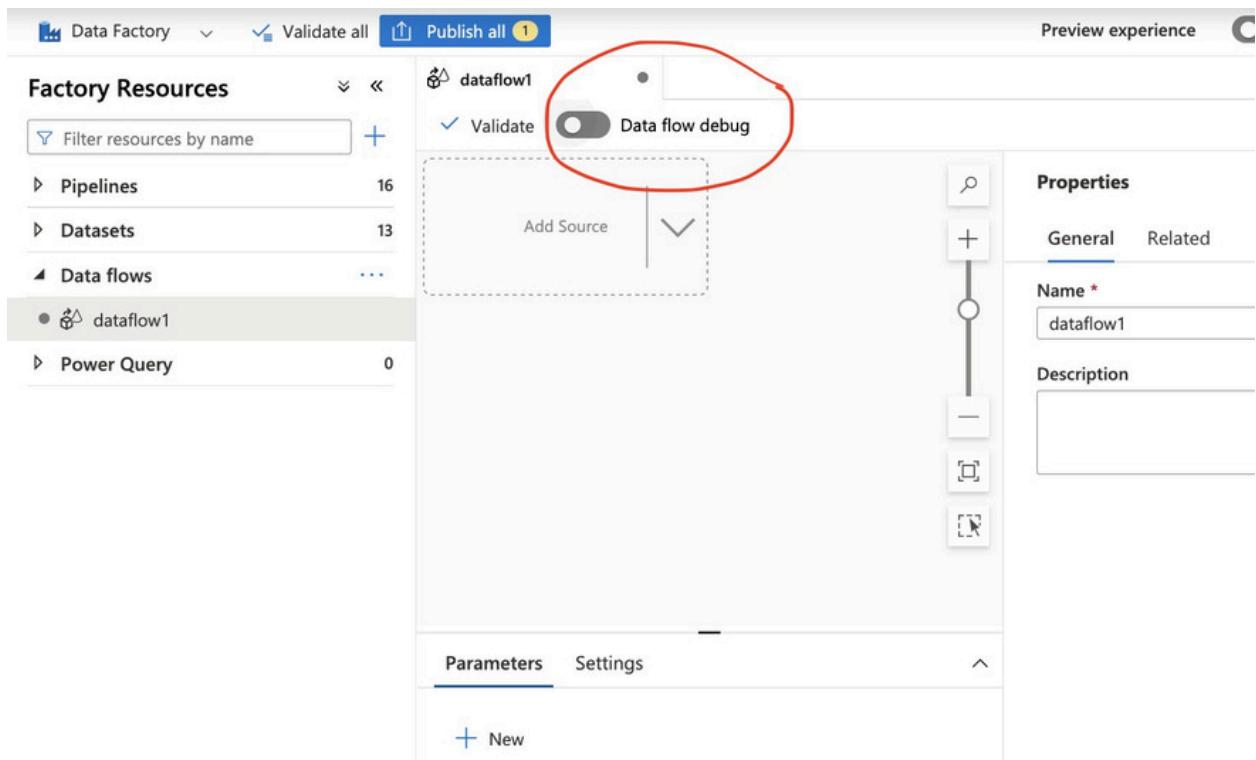


Azure Data Factory DATA FLOW:

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.



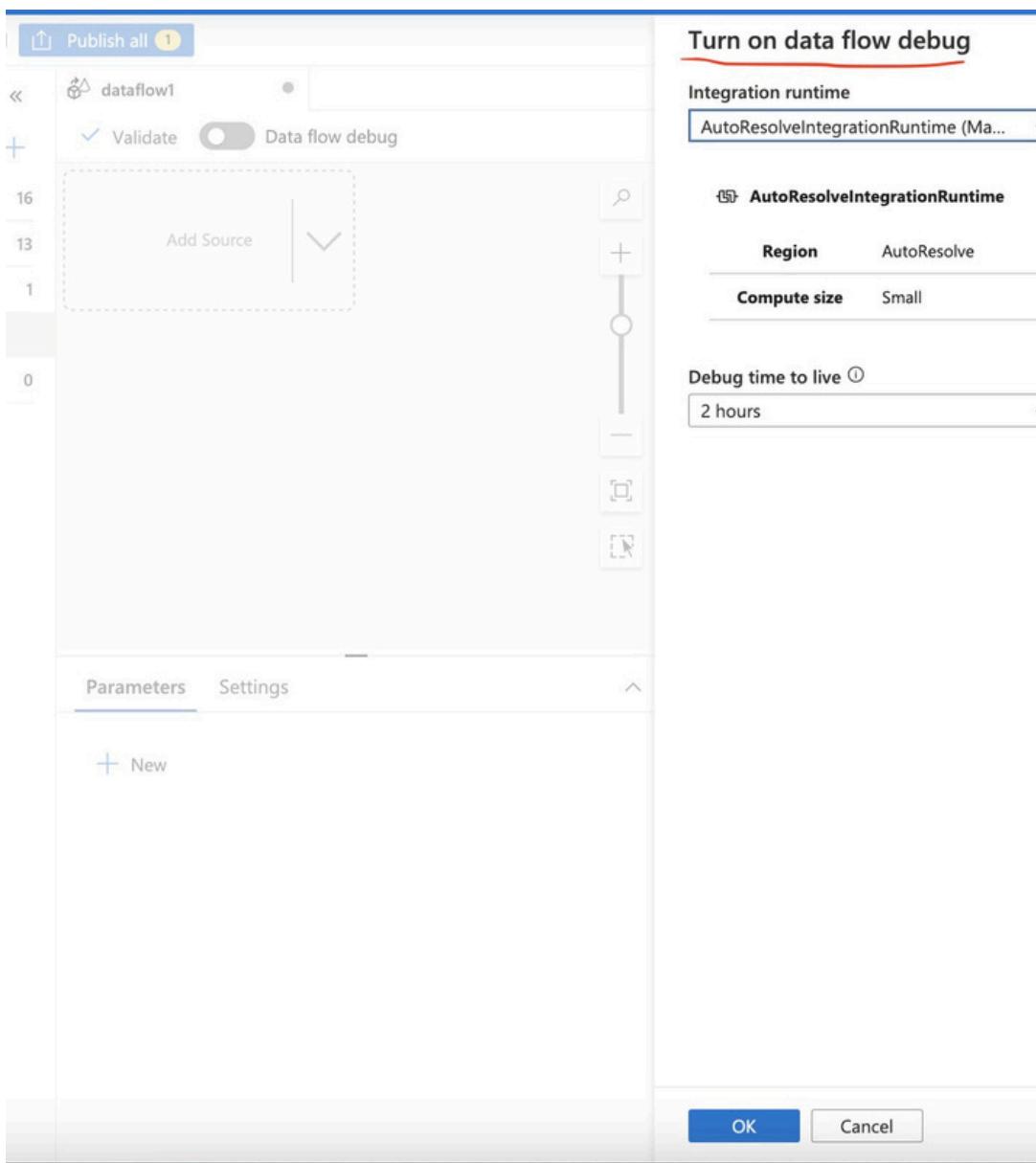
Data flow is nothing but spark code and cluster.

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:

The screenshot shows the Dataflow Example interface. At the top, there is a header with a project name "dataflowExample" and a "Validate" button. Below the header, a source component is displayed with the label "source1" and a "Columns: 0 total" message. To the right of the source is a plus sign (+) for adding more sources. A dashed box labeled "Add Source" contains a checkmark icon, indicating where a new source can be added. Below the source component, there is a navigation bar with tabs: "Source settings" (which is selected), "Source options", "Projection", "Optimize", "Inspect", and "...". Under the "Source settings" tab, there are several configuration fields:

- "Output stream name *": A text input field containing "source1" with a "Learn more" link.
- "Description": A text input field containing "Add source dataset" with a "Reset" button.
- "Source type *": A selection field with two options: "Dataset" (selected) and "Inline".
- "Dataset *": A dropdown menu with a "Select..." option and a "New" button.
- "Options": A section with a checked checkbox for "Allow schema drift" and a help icon.

Source settings Source options Projection Optimize Inspect ...

Output stream name * CustomerTable Learn more

Description Add source dataset Reset

Source type * Dataset Inline

Dataset * Select... + New

Options

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

Sampling * ⓘ Enable Disable

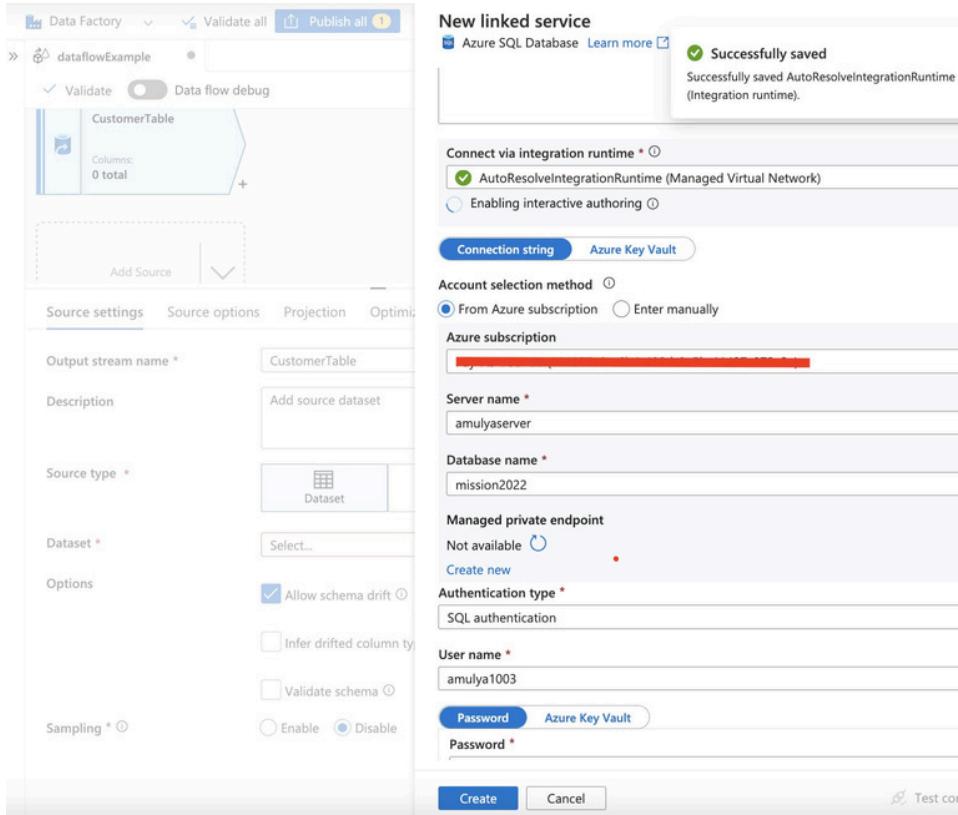
Create a new Dataset to point to customer Table:

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

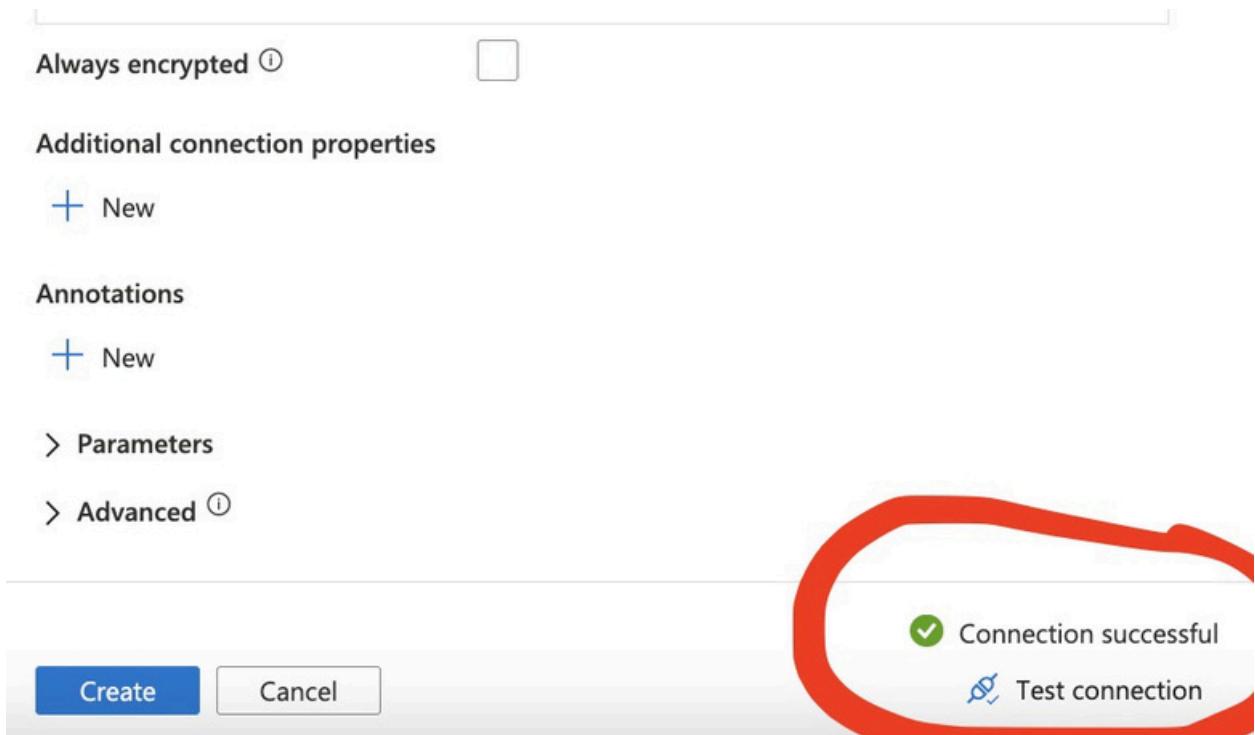
Select a data store

All	Azure	Database	File	Generic protocol	NoSQL	Services and apps
Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen1	Azure Data Lake Storage Gen2				
Azure Database for MySQL	Azure Database for PostgreSQL	Azure SQL Database				
Azure SQL Database Managed Instance	Azure Synapse Analytics	Dataverse (Common Data Service for Apps)				

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

Linked service *

Connect via integration runtime * ⓘ

 AutoResolveIntegrationRuntime (Managed Virtual Network)  Interactive authoring enabled ⓘ 

Table name

None

SalesLT.Address

SalesLT.Customer

SalesLT.CustomerAddress

SalesLT.Product

SalesLT.ProductCategory

SalesLT.ProductDescription

SalesLT.ProductModel

SalesLT.Customer

Set properties

Name

Linked service *

Connect via integration runtime * ⓘ

 AutoResolveIntegrationRuntime (Managed Virtual Network)   Interactive authoring enabled ⓘ

Table name

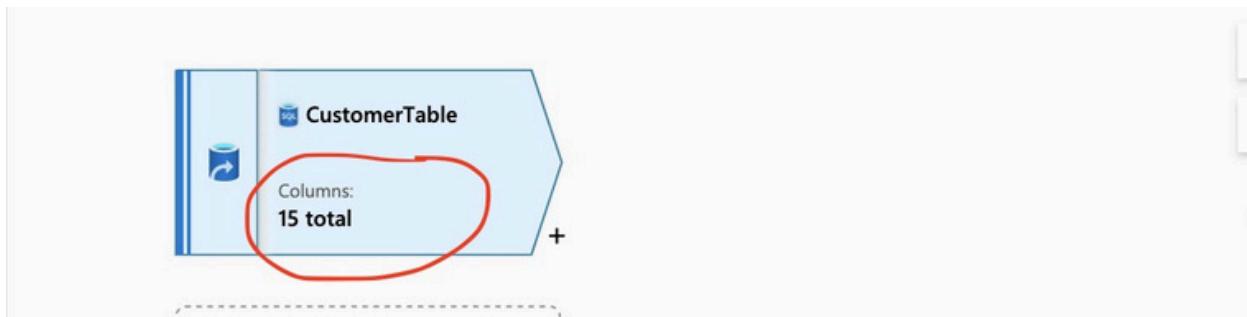
   Edit

Import schema

 From connection/store None

> Advanced

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



The screenshot shows the 'Source settings' tab of the dataset configuration. At the top, there's a preview icon and the name 'CustomerTable'. Below it, a red circle highlights the text 'Columns: 15 total'. The main configuration area includes:

- Output stream name ***: CustomerTable
- Description**: Import data from DS_Customer_SQLDB
- Source type ***: Dataset (selected)
- Dataset ***: DS_Customer_SQLDB
- Options**:
 - Allow schema drift ⓘ
 - Infer drifted column types ⓘ
 - Validate schema ⓘ
- Sampling ***: ⓘ
 - Enable
 - Disable

Source options:

The screenshot shows the 'Source options' tab selected in a UI. At the top, there are tabs: 'Source settings', 'Source options' (which is active), 'Projection', 'Optimize', 'Inspect', and '...'. Below these are several configuration items:

- Input**: A radio button group where 'Table' is selected, with 'Query' and 'Stored procedure' as other options.
- Batch size**: An input field containing a placeholder value.
- Change data capture**: A checkbox that is currently unchecked.
- Isolation level**: A dropdown menu set to 'Read uncommitted'.

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

The screenshot shows the 'Projection' tab selected in a UI. At the top, there are tabs: 'Source settings', 'Source options', 'Projection' (which is active), 'Optimize', 'Inspect', and '...'. Below these are buttons for 'Import projection', 'Reset schema', and 'Overwrite schema'. The main area displays the schema for a table named 'CustomerTable' with 15 columns:

Column name	Type
CustomerID	integer
NameStyle	boolean
Title	string
FirstName	string
MiddleName	string
LastName	string
Suffix	string
CompanyName	string
SalesPerson	string
EmailAddress	string
Phone	string
PasswordHash	string

Click on “Overwrite Schema”, to change

The screenshot shows a schema editor interface with the following components:

- Top Navigation:** Source settings, Source options, **Projection**, Optimize, Inspect.
- Middle Row Buttons:** Import projection, Reset schema, Overwrite schema (highlighted with a red oval).
- Column Headers:** Column name, Type.
- Table Data:** A list of columns with their types:
 - CustomerID: integer
 - NameStyle: boolean
 - Title: string
 - FirstName: string
 - MiddleName: string
 - LastName: string
 - Suffix: string
 - CompanyName: string
 - SalesPerson: string
 - EmailAddress: string
 - Phone: string
 - PasswordHash: string

You can change Integer to anyType:

The screenshot shows a schema editor interface. On the left, there is a list of columns with their current types: CustomerID (integer), NameStyle (string), Title (string), FirstName (string), MiddleName (string), LastName (string), Suffix (string), CompanyName (string), SalesPerson (string), EmailAddress (string), Phone (string), and PasswordHash (string). To the right of this list is a dropdown menu titled "Type" with a red circle highlighting it. The menu lists various data types: integer, string, boolean, date, timestamp, short, double, float, and another string option. The "integer" option is currently selected.

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

The screenshot shows a database configuration interface. At the top, there is a diagram of a "CustomerTable" with 15 columns. Below the diagram, there is a navigation bar with tabs: Source settings, Source options, Projection, Optimize, Inspect, and ... The "Optimize" tab is highlighted with a red circle. At the bottom, there is a section for "Partition option *". It contains three radio buttons: "Use current partitioning" (selected), "Single partition", and "Set partitioning".

Inspect shows the Actual output:

The screenshot shows the Data Inspect interface for a table named "CustomerTable". The table has 15 columns. The "Inspect" button in the toolbar is circled in red.

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

IF no transformations, Data preview will show input data:

The screenshot shows a data preview interface. At the top, there's a header bar with tabs: Source settings, Source options, Projection, Optimize, Inspect, and Data preview (which is currently selected). Below the header, there are buttons for INSERT (100), UPDATE (0), DELETE (0), UPSERT (0), and LOOKUP (0). A toolbar below these includes Refresh, Typecast, Modify, Map drifted, Statistics, Remove, and Export to CSV. The main area displays a table with 15 columns and 6 rows. The columns are labeled: Custom..., NameSt..., Title, FirstName, Middle..., and LastName. The rows contain sample data: Row 1 (Orlando, N., Gee), Row 2 (Keith, NULL, Harris), Row 3 (Donna, F., Carreras), Row 4 (Janet, M., Gates), Row 5 (Lucy, NULL, Harringt...), and Row 6 (Rosmarie, I, Carroll).

Custom...	NameSt...	Title	FirstName	Middle...	LastName
1	X	Mr.	Orlando	N.	Gee
2	X	Mr.	Keith	NULL	Harris
3	X	Ms.	Donna	F.	Carreras
4	X	Ms.	Janet	M.	Gates
5	X	Mr.	Lucy	NULL	Harringt...
6	X	Ms.	Rosmarie	I	Carroll

Click on another ADD SOURCE to test Inline:



Inline:

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

Source settings Source options Projection Optimize Inspect ...

Output stream name * CustomerTable Learn more

Description Add source dataset  Reset

Source type *  Dataset  Inline

Inline dataset type * 

Linked service * Select...

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

Sampling * ⓘ Enable Disable

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

Source settings Source options Projection Optimize Inspect ...

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

Description Import data from SqIDatabaseConnection [Reset](#)

Source type *  Dataset  Inline

Inline dataset type * DelimitedText

Linked service * AzureDataLakeStoragetoADLS

 Test connection  Edit  New

Skip line count

Sampling *  Enable  Disable

Rows limit 100

Click on source options and select the exact path location:

The screenshot shows the 'Source options' tab selected in the 'CustomerTable' configuration. On the right, a 'Browse' window is open, displaying a file tree under 'Root folder > landing'. The 'Customer.csv' file is highlighted and circled in red. Below the tree, it says 'Showing 1 - 2 of 2 items'. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

CustomerTable

Columns: 0 total

Source settings Source options Projection Optimiz:

✓ File settings

File mode File Wildcard

File path * /

Allow no files found

Change data capture

Compression type

Encoding

Column delimiter

Row delimiter

Quote character

Escape character

First row as header

Browse

Select a file or folder.

Root folder > landing

Customer.csv

HighWaterMark.txt

Showing 1 - 2 of 2 items

OK Cancel

As CSV file as header:

The screenshot shows the 'Source options' tab selected in a data import configuration window. The 'First row as header' checkbox is checked and highlighted with a red oval. Other settings like Column delimiter, Row delimiter, Quote character, Escape character, and Null value are also visible.

We can specify format of our own:

The screenshot shows the 'Projection' tab for a 'CustomerCSVFile' schema. The table lists 15 columns with their types (e.g., integer, boolean, string) and 'Format' buttons for customization.

Column name	Type	Format
CustomerID	integer	Specify format
NameStyle	boolean	Specify format
Title	string	Specify format
FirstName	string	Specify format
MiddleName	string	Specify format
LastName	string	Specify format
Suffix	string	Specify format
CompanyName	string	Specify format
SalesPerson	string	Specify format
EmailAddress	string	Specify format
Phone	string	Specify format
PasswordHash	string	Specify format

Click on + symbol to show only few columns.

The screenshot shows the 'Source options' tab for a 'CustomerCSVFile' input. On the left, there's a preview pane showing 15 total columns. A '+' icon is located next to the preview pane. Below it, the 'Source options' tab is selected. The 'File settings' section includes 'File mode' (set to 'File'), 'File path' ('landing'), and several optional checkboxes for 'Allow no files found', 'Change data capture', and compression types ('None'). To the right, a large dropdown menu titled 'Multiple inputs/outputs' is open, listing various data processing operations like Join, Conditional Split, Exists, Union, and Lookup. The 'Select' operation is currently highlighted. Other sections visible include 'Schema modifier' (with Derived Column, Select, Aggregate, Surrogate Key, Pivot, Unpivot, Window, and Rank) and a 'Projection' tab at the bottom.

CustomerCSVFile

Columns: 15 total

+

Source settings Source options Projection

File settings

File mode File W

File path * landing

Allow no files found

Change data capture

Compression type None

Encoding Default(UTF-8)

Column delimiter Comma (,)

Row delimiter Default (\r,\n)

Quote character Double quote

Escape character Backslash (\)

Multiple inputs/outputs

Join

Conditional Split

Exists

Union

Lookup

Schema modifier

Derived Column

Select

Aggregate

Surrogate Key

Pivot

Unpivot

Window

Rank

The screenshot shows the configuration interface for a 'Select' activity in Azure Data Factory. At the top, there's a preview diagram showing an incoming stream 'CustomerCSVFile' (imported from 'AzureDataLakeStorageASS6') being processed by a 'select1' activity, which outputs 15 total columns. Below the diagram are tabs: 'Select settings' (selected), 'Optimize', 'Inspect', and 'Data preview'.

The 'Select settings' tab contains the following fields:

- Output stream name ***: select1
- Description**: Renaming CustomerCSVFile to select1 with columns 'CustomerId', 'NameStyle', 'Title', 'FirstName', 'MiddleName', 'LastName'
- Incoming stream ***: CustomerCSVFile
- Options**:
 - Skip duplicate input columns
 - Skip duplicate output columns
- Input columns ***:
 - Auto mapping
 - Reset
 - Add mapping
 - Delete
 - 15 mappings: All inputs mappedA detailed mapping table follows:

CustomerCSVFile's column	Name as
123 CustomerID	CustomerId
✗ NameStyle	NameStyle
... Title	Title

Select and delete the columns you no longer needed:

The screenshot shows the Azure Data Factory mapping editor interface. At the top, there's a pipeline diagram with a source 'CustomerCSVFile' (importing from 'AzureDataLakeStorageASS6') connected to a 'UsingSelect' activity. The 'UsingSelect' activity has 15 total columns. Below the diagram is a toolbar with tabs: 'Select settings' (selected), 'Optimize', 'Inspect', 'Data preview', 'Reset' (with a help icon), 'Add mapping', and 'Delete'. A tooltip for 'Delete' says '15 mappings: All inputs mapped'. The main area is a table titled 'CustomerCSVFile's column' with columns for 'Name as' and 'Name as'. Several columns are circled in red: 'Suffix' and 'Phone' in the list, and the 'Delete' button in the toolbar.

CustomerCSVFile's column	Name as
CustomerID	CustomerID
LineStyle	LineStyle
Title	Title
FirstName	FirstName
MiddleName	MiddleName
LastName	LastName
<input checked="" type="checkbox"/> Suffix	Suffix
<input type="checkbox"/> CompanyName	CompanyName
abc SalesPerson	SalesPerson
abc EmailAddress	EmailAddress
<input type="checkbox"/> abc Phone	Phone
<input checked="" type="checkbox"/> abc PasswordHash	PasswordHash

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

The screenshot shows the Azure Data Factory mapping editor interface. At the top, there is a pipeline diagram with two main components: 'CustomerCSVFile' (Import data from AzureDataLakeStorageASS6) and 'UsingSelect'. The 'UsingSelect' component has a status bar indicating 'Columns: 11 total'. Below the pipeline diagram, there are tabs: 'Select settings' (which is selected), 'Optimize', 'Inspect', and 'Data preview'.

Under the 'Select settings' tab, there are several controls:

- 'Input columns *': A checkbox labeled 'Auto mapping' with a help icon.
- 'Reset' button with a circular arrow icon.
- 'Add mapping' button with a plus sign icon.
- 'Delete' button with a trash bin icon.
- A message: '11 mappings: 4 column(s) from the inputs left unmapped'.

A red oval highlights the message '11 mappings: 4 column(s) from the inputs left unmapped'.

CustomerCSVFile's column	Name as
123 CustomerID	CustomerID
✗ NameStyle	NameStyle
abc Title	Title
abc FirstName	FirstName
abc MiddleName	MiddleName
abc LastName	LastName
abc CompanyName	CompanyName
abc SalesPerson	SalesPerson
abc EmailAddress	EmailAddress

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

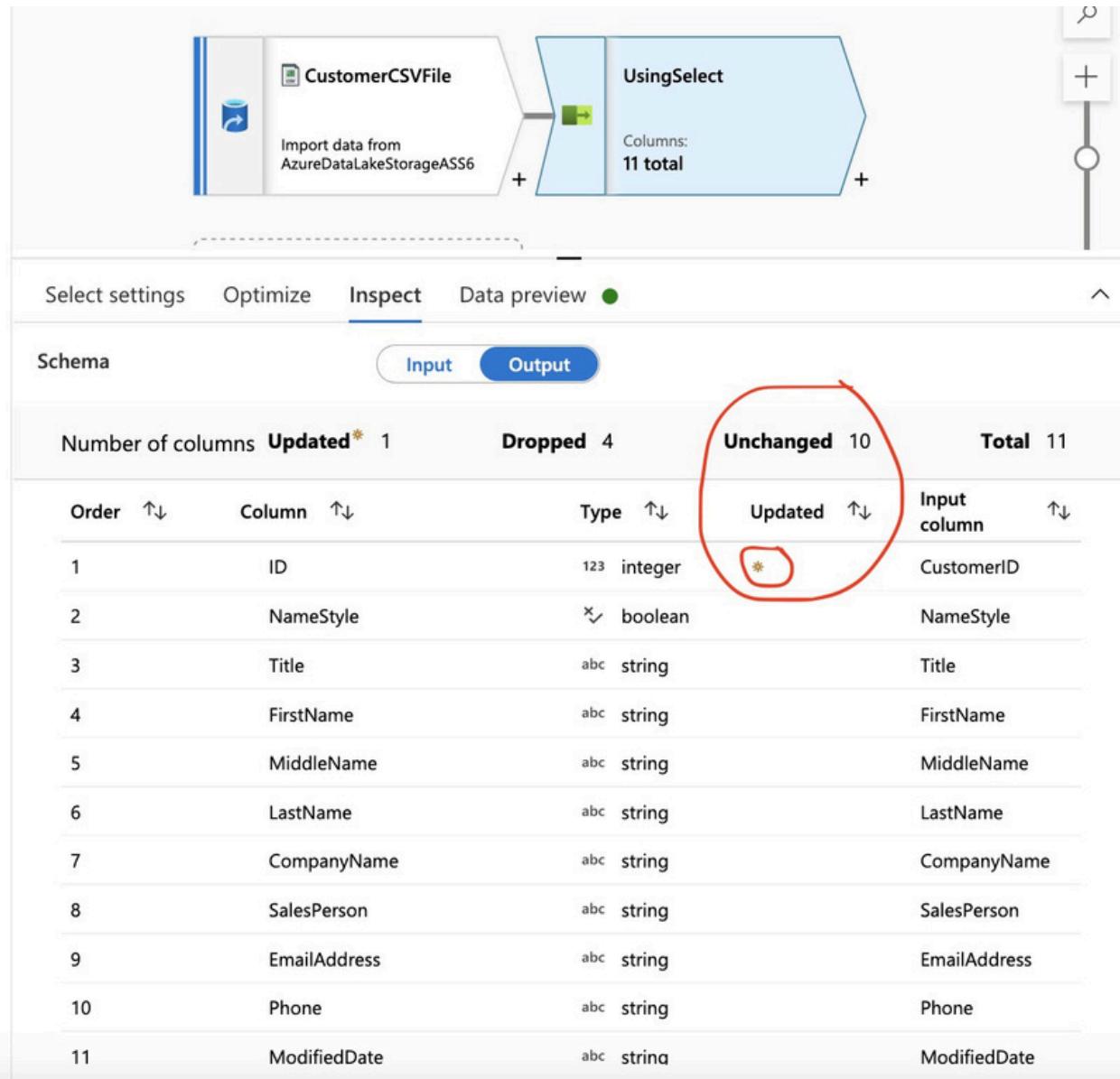
The screenshot shows the Azure Data Factory mapping editor interface. At the top, there's a pipeline diagram with a source 'CustomerCSVFile' (Import data from AzureDataLakeStorageASS6) connected to a 'UsingSelect' activity. The 'UsingSelect' activity has 11 total columns. Below the diagram, there are tabs: 'Select settings' (which is selected), 'Optimize', 'Inspect', and 'Data preview'. A checkbox for 'Skip duplicate input columns' is checked.

In the 'Select settings' tab, there are buttons for 'Auto mapping' (with a help icon), 'Reset', 'Add mapping', and 'Delete mapping'. It also displays '11 mappings: 4 column(s) from the inputs left unmapped'.

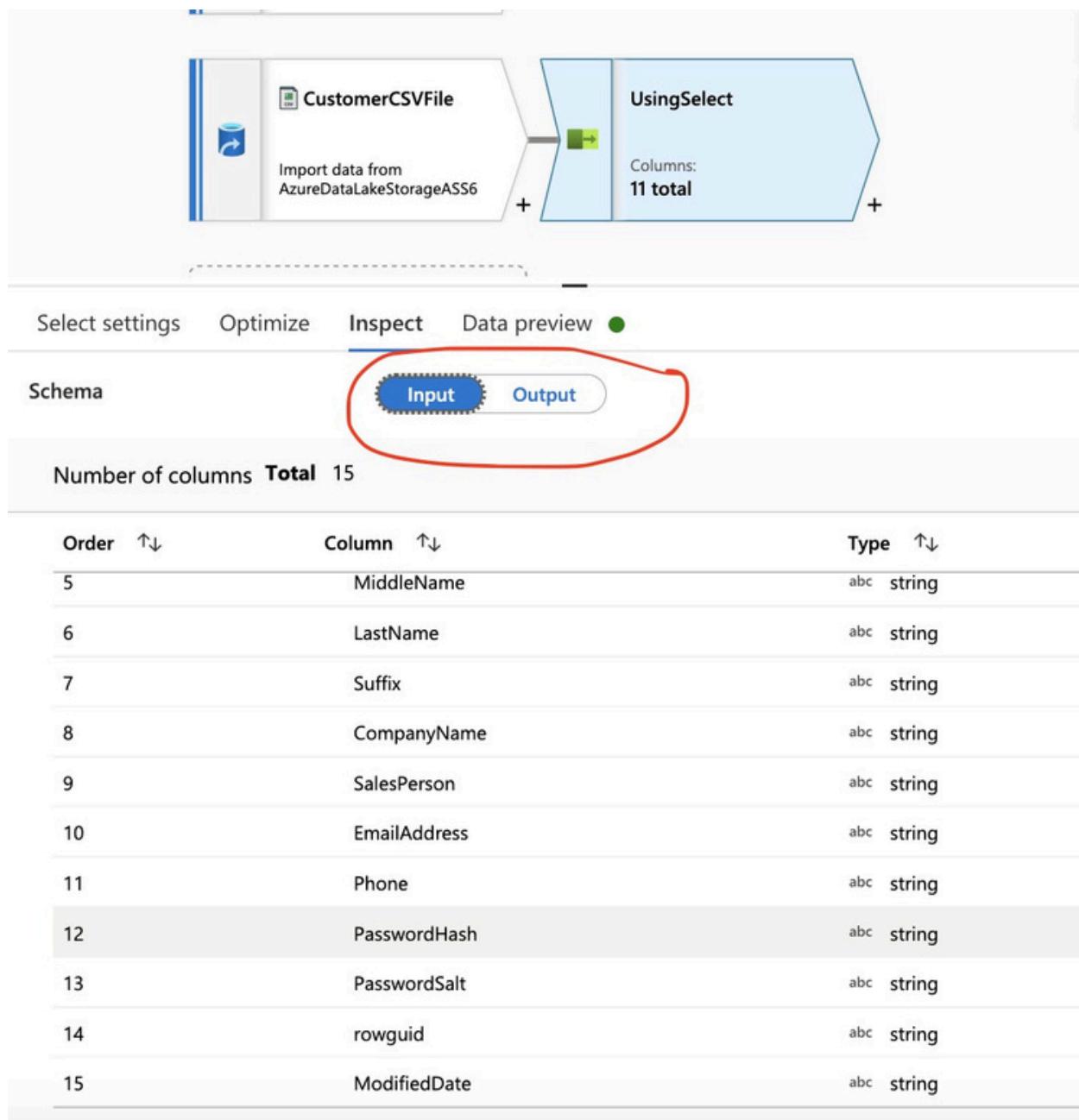
The main area shows a list of 'CustomerCSVFile's column' names on the left and their corresponding 'Name as' target names on the right. Most columns map directly, but 'CustomerID' is explicitly mapped to 'ID', which is highlighted with a red oval. The list includes:

CustomerCSVFile's column	Name as
123 CustomerID	ID
✗ NameStyle	NameStyle
abc Title	Title
abc FirstName	FirstName
abc MiddleName	MiddleName
abc LastName	LastName
abc CompanyName	CompanyName
abc SalesPerson	SalesPerson
abc EmailAddress	EmailAddress

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:

The screenshot shows the configuration interface for the 'UsingSelect' component in Azure Data Factory. At the top, there's a visual representation of the pipeline flow: 'CustomerCSVFile' (Import data from AzureDataLakeStorageASS6) → 'UsingSelect' (Columns: 11 total) → 'filter1' (Add expression for the filter). Below this, the 'Select settings' tab is selected.

Output stream name *: UsingSelect

Description: Renaming CustomerCSVFile to UsingSelect with columns 'ID, NameStyle, Title, FirstName, MiddleName, LastName, ...'

Incoming stream *: CustomerCSVFile (highlighted with a red circle)

Options:

- Skip duplicate input columns
- Skip duplicate output columns

Input columns *:

Auto mapping (unchecked) | Reset | Add mapping | Delete | 11 mappings: 4 column(s) from the inputs left unmapped

CustomerCSVFile's column	Name as
CustomerID	ID
NameStyle	NameStyle
[...]	[...]

We want to filter LastName:Gates

The screenshot shows a data processing pipeline with the following components:

- CustomerCSVFile**: An input stream from keStorageASS6.
- UsingSelect**: A transformation step that renames the input stream. It lists columns: 'ID', 'NameStyle', 'Title', 'FirstName', 'MiddleName', and 'LastName'.
- filter1**: A filtering step that takes the 'UsingSelect' stream as input. It displays 11 total columns.

Below the pipeline, there are several tabs and configuration fields:

- Filter settings** (selected)
- Optimize**
- Inspect**
- Data preview**

Configuration fields:

- Output stream name ***: filter1
- Description**: Add expression for the filter
- Incoming stream ***: UsingSelect
- Filter on ***: Enter filter... (with ANY placeholder)

Buttons at the bottom right include **Learn more**, **Reset**, and **Open expression builder**.

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(~~x~~ 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:

Expression elements

- All
- Functions
- Input schema
- Parameters**
- Cached lookup
- Data flow library functions

Expression values

 Filter by keyword

 Create new

Data preview

Save and finish Cancel Clear contents

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

```
LastNames == '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

X

Gee

X

Harris

X

Carreras

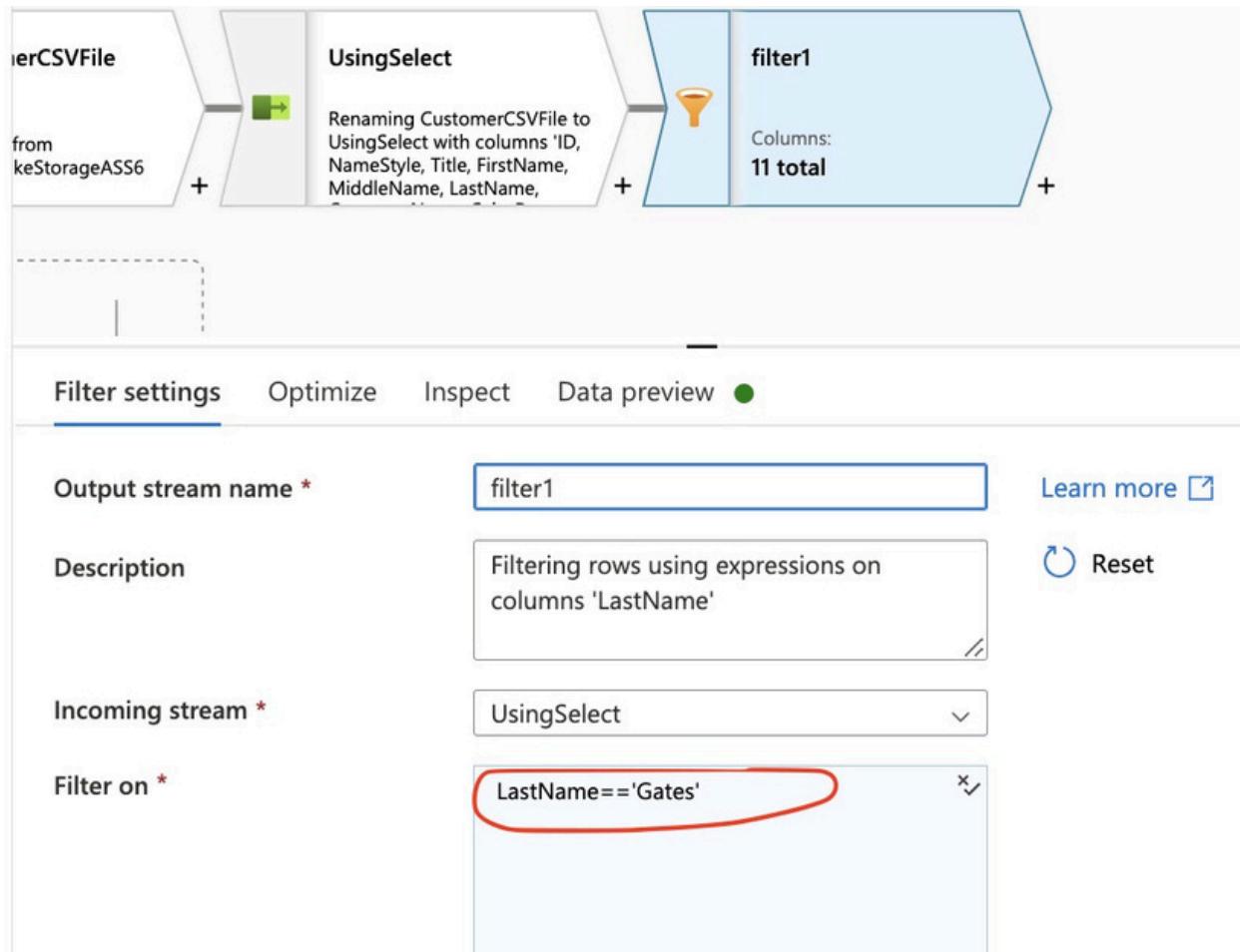
✓

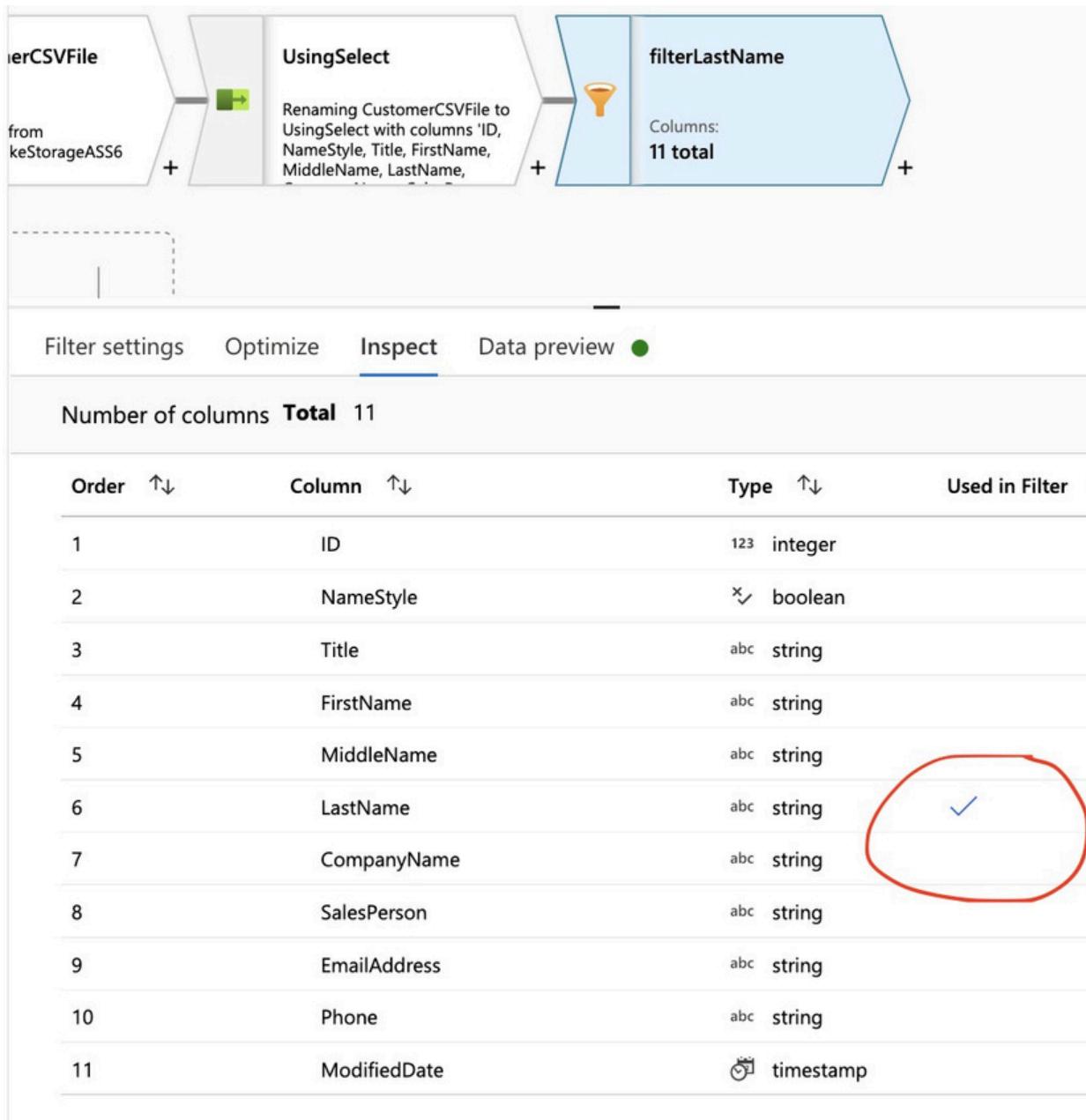
Gates

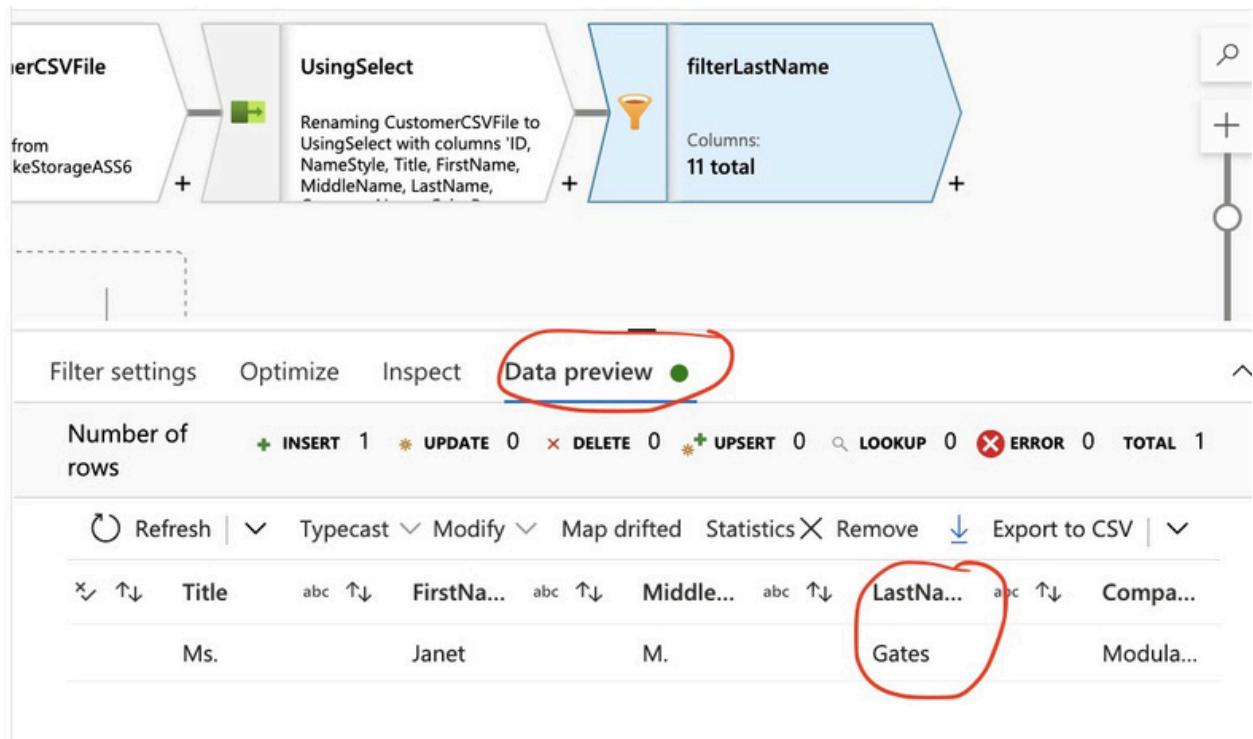
Save and finish

Cancel

Clear contents







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

The screenshot shows a data flow in Alteryx. The workflow consists of three main components:

- UsingSelect**: Renaming CustomerCSVFile to UsingSelect with columns 'ID', 'NameStyle', 'Title', 'FirstName', 'MiddleName', 'LastName'.
- filterLastName**: Filtering rows using expressions on columns 'LastName'.
- derivedColumn1**: Description: Columns: 12 total.

The 'Derived column's settings' tab is selected. The 'derivedColumn1' section shows a list of columns: NameStyle, Title, FirstName, MiddleName, LastName, CompanyName, SalesPerson, EmailAddress, Phone, ModifiedDate. An 'Open expression builder' button is highlighted with a red circle.

Column	Expression
Address	'USA'

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

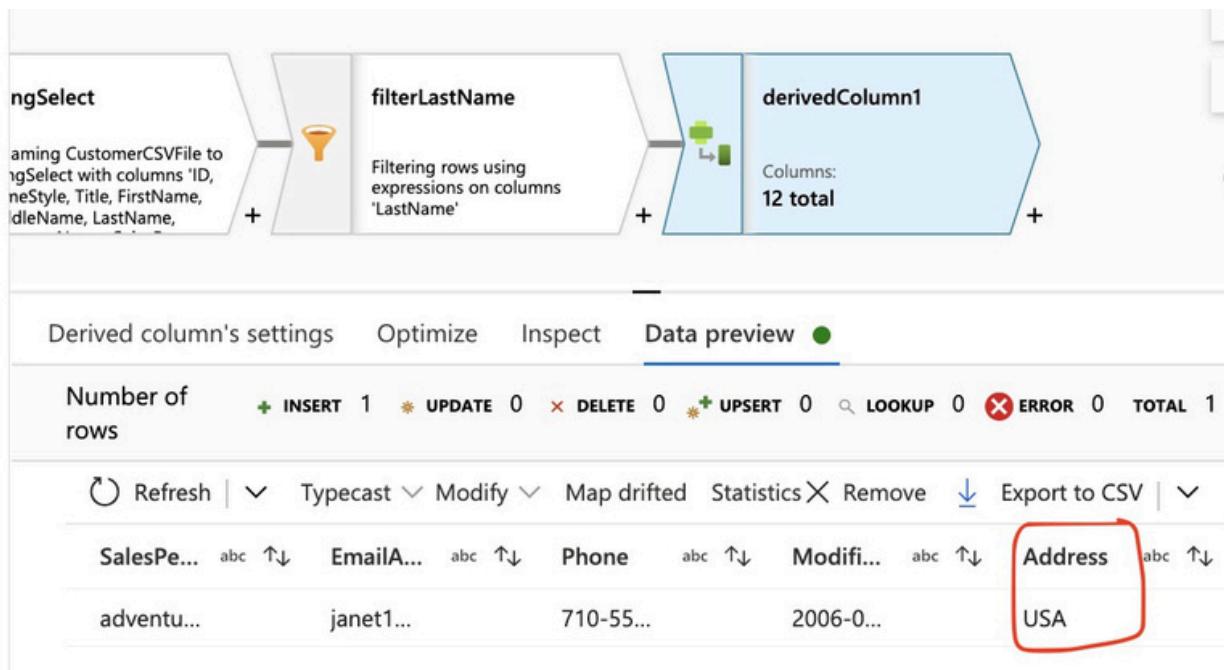
Output: Address abc

USA

Save and finish

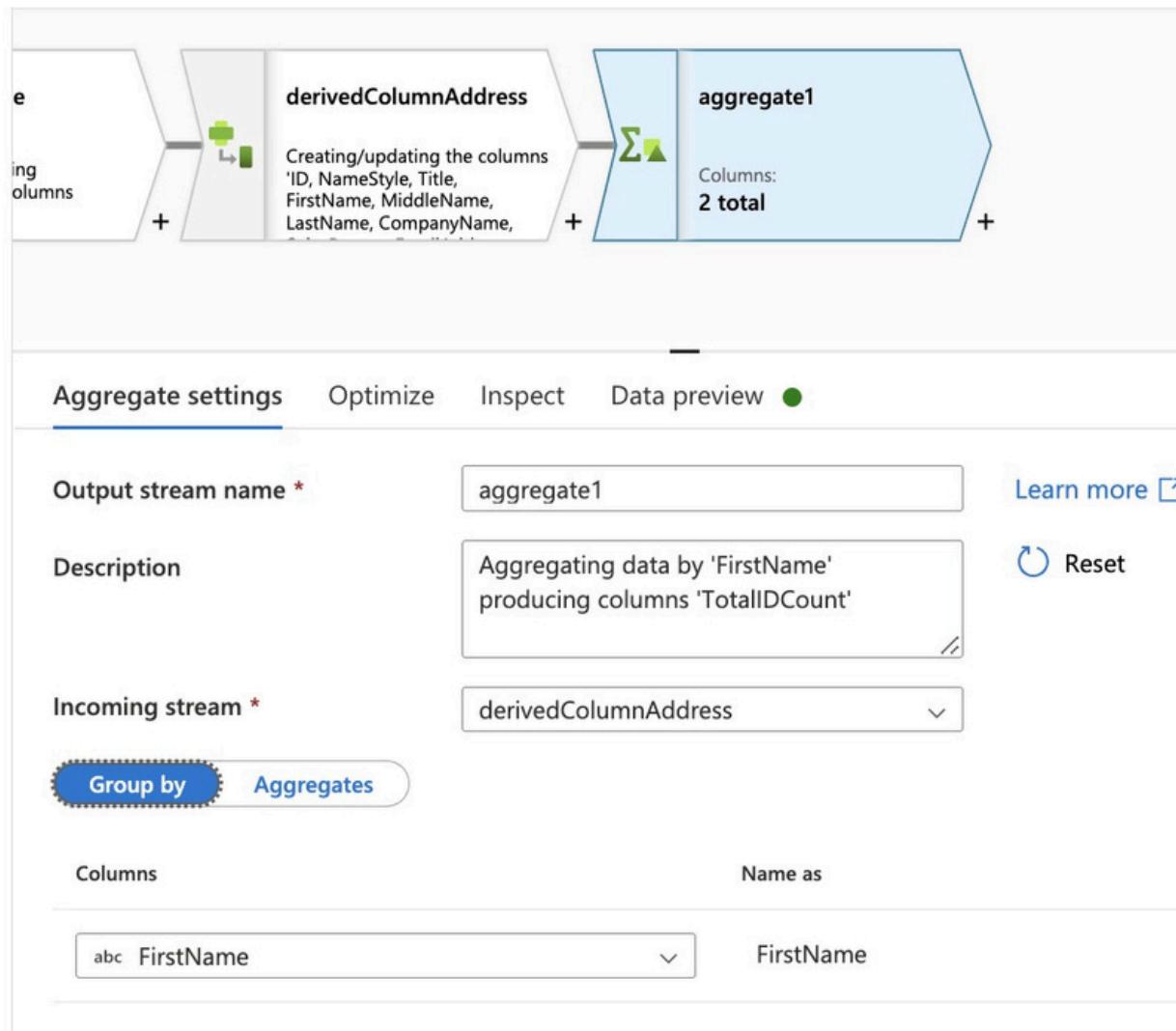
Cancel

Clear contents



Aggregate :

GroupBY “FirstName”



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

The diagram illustrates a data processing pipeline. It starts with a stream labeled "e" containing "Existing columns". This stream merges with a component labeled "derivedColumnAddress", which is described as "Creating/updating the columns 'ID, NameStyle, Title, FirstName, MiddleName, LastName, CompanyName, Address'". The output of this merge then enters an aggregate component labeled "aggregate1". The aggregate component is described as "Columns: 2 total".

Aggregate settings

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

<input type="checkbox"/> Column	Expression
<input type="checkbox"/> TotalIDCount	$\sum(ID)$

Dataflow expression builder

Σ aggregate1

Aggregate Columns

+ Create new ▾

123 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 ▾

123 ID

✗ NameStyle

abc Title

abc FirstName

abc MiddleName

abc LastName

... CompanyName

Save and finish

Cancel

Clear contents

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

The diagram illustrates a data flow process:

- Source:** 'e' (represented by a grey parallelogram)
- Transformation 1:** 'derivedColumnAddress' (represented by a grey parallelogram with a green plus sign icon). Description: "Creating/updating the columns 'ID', 'NameStyle', 'Title', 'FirstName', 'MiddleName', 'LastName', 'CompanyName', 'City', 'Country'".
- Transformation 2:** 'aggregate1' (represented by a grey parallelogram with a green sigma icon). Description: "Aggregating data by 'FirstName' producing columns 'TotalIDCount'".
- Sink:** 'sink1' (represented by a blue rounded rectangle with a blue arrow icon). Description: "Columns: 2 total".

Sink Configuration:

- Sink** tab is selected.
- Output stream name ***: sink1
- Description**: Add sink dataset
- Reset** button
- Incoming stream ***: aggregate1
- Sink type ***:
 - Dataset** (selected)
 - Inline**
 - Cache**
- Dataset ***: Select... (dropdown menu) + New
- Options**:
 - Allow schema drift ⓘ
 - Validate schema ⓘ

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

dataflowExample ADLSSASS6

Validate Data flow debug Debug Settings

derivedColumnAddress

Creating/updating the columns 'ID, NameStyle, Title, FirstName, MiddleName, LastName, CompanyName,'

aggregate1

Aggregating FirstNames, TotalCount

Sink Settings Errors Mapping Optimize Insp

Output stream name * SavetoFile

Description Export data to ADLSSASS6

Incoming stream * aggregate1

Sink type * Dataset

Dataset * ADLSSASS6

Connection successful

Test connection Open New

Skip line count

Options Allow schema drift Validate schema

Choose the format type of your data

 Avro	 DelimitedText	 JSON
 ORC	 Parquet	 Binary

Continue Back

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

Set properties

Name

Linked service *

Connect via integration runtime * ⓘ

 AutoResolveIntegrationRuntime (Managed Virtual Network)  Enabling interactive authoring ⓘ

File path

 / /  

Import schema

 From connection/store From sample file None

> Advanced

The diagram illustrates a data flow process:

- 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:

The screenshot shows the Azure Data Factory Pipeline Editor interface. At the top, there are tabs for 'Validate', 'Debug' (which is selected), 'Add trigger', and a toggle for 'Data flow debug'. Below the tabs, a 'Data flow' activity is selected, displaying a preview window titled 'Data flow1' with a blue cube icon.

The main configuration area is the 'Settings' tab, which is highlighted with a red oval. It contains the following fields:

- Data flow ***: A dropdown menu set to 'dataflowExample'.
- Run on (Azure IR) ***: A dropdown menu set to 'AutoResolveIntegrationRuntime (Ma...)'.
- Compute size ***: A dropdown menu set to 'Small'.
- Logging level ***: Radio buttons for 'Verbose' (selected), 'Basic', and 'None'.
- Sink properties**: A collapsed section indicated by a right-pointing arrow.
- Staging**: A collapsed section indicated by a right-pointing arrow.

Dataflow debug and Inspect:

All pipeline runs > DataflowPipelineExample - Activity runs > Data flow1

① 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, 1:44:46 AM

Sinks All streams

Sink	Status	Processing time	Highest processing time	Rows written	Stages	Lineage
SavetoFile	Succeeded	5s	4s 881ms	1	1	1

Click on stages:

Stages

Transformations Stages ⓘ

SavetoFile

Processing time: 5s

STAGE ID	ROWS READ	ROWS WRITTEN	TIME
77	4	1	88ms
78	1	1	4s 881ms

Close

Lineage:

Lineage

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

Close

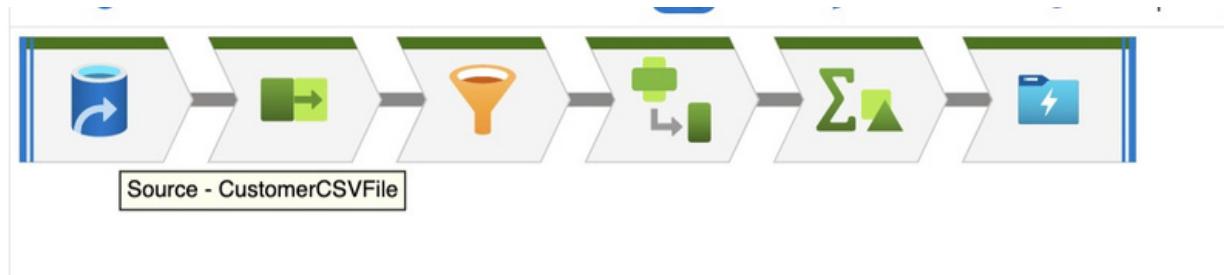
Data flow1
 Cluster startup time: 1s 398ms Number of transformations: 6 Data flow status: Success

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:

Home > adlsstgacct | Containers >

landing ...
Container

» [Upload](#) [Add Directory](#) [Refresh](#) | [Rename](#) [Delete](#) [Change tier](#) [Acquire lease](#) ...

Authentication method: Access key ([Switch to Azure AD User Account](#))
Location: landing

Search blobs by prefix (case-sensitive) Show deleted objects

Name	Modified	Access tier
<input type="checkbox"/> _SUCCESS	12/11/2022, 14:44 ...	Hot (Inferred)
<input type="checkbox"/> Customer.csv	12/7/2022, 10:33:55 ...	Hot (Inferred)
<input type="checkbox"/> HlghWaterMark.txt	12/8/2022, 11:56:51 ...	Hot (Inferred)
<input type="checkbox"/> part-00000-03f5a313-2308-40c7-86ca-b6b6c1661d3b-c000.json	12/11/2022, 1:42:40 ...	Hot (Inferred)
<input type="checkbox"/> part-00000-995cb71d-0ef7-4815-ad75-7400d9b8a91b-c000.json	12/11/2022, 1:44:39 ...	Hot (Inferred)
<input type="checkbox"/> part-00181-03f5a313-2308-40c7-86ca-b6b6c1661d3b-c000.json	12/11/2022, 1:42:40 ...	Hot (Inferred)
<input type="checkbox"/> part-00181-995cb71d-0ef7-4815-ad75-7400d9b8a91b-c000.json	12/11/2022, 1:44:39 ...	Hot (Inferred)

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

Output has more files

part-00181-03f5a313-2308-40c7-86ca-b6b6c1661d3b-c000.json

Blob

 Save  Discard  Download  Refresh  Delete

Overview Versions **Edit** Generate SAS

```
1  [{"FirstName": "Janet", "TotalIDCount": 434334}]  
2
```

Json

 Preview

Dataflow data successfully copied into ADLS after transformations .