

**ELT with Azure Data Factory**

**And**

**Mapping Data Flows**

Hands-on lab step-by-step

Feb 2020

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

# Contents

[Azure Data Factory hands-on lab 1](#azure-data-factory-hands-on-lab)

[ELT with Mapping Dataflows, SmartFood’s “Items(foods)” and Customer dimensions 2](#elt-with-mapping-dataflows)

[Task 1: Create a Parquet dataset to write SmartFood’s DW Blob container 3](#X92dceeaec1c58b4ccee2ab529e1ea5356576018)

[Task 2: Create SQL Database Dataset 3](#X94dbdfd29895d07c4b2068c02750486b8dbbafc)

[Task 3: Create Foods Dimension 5](#X731831f276c359de55e1d4a7ac872ef735f03c6)

[Task 4: (Challenge Task) Create customer dimension 9](#challenge-task-create-customer-dimension)

[Task 5: Create SmartFoods Invoice fact tables 12](#create-smartfoods-invoice-fact-tables)

# Azure Data Factory hands-on lab

## ELT with Mapping Dataflows

#### Solution overview

**Solution requirements:**

In previous exercise you ingested *customer*, *orders* and *orderlines* data from WWI to blob storage. Also, you ingested *Customer*, *Transactions* and *reference* data from SmartFoods systems to Blob storage. In this exercise you will use ADF Mapping DF to cleans, transform, enrich and store this data to be served using PowerBI to business users. Plus, data needs to be prepared for SmartFoods customer facing application to which displays accumulated loyalty points and comprehensive nutritional information and suggestions.

***Analytics Reporting:***

Since WWI business users are keen to setup a self-serve reporting environment, it means the serving layer storage solution should support the following requirements:

* Role based access control plus row and column level security so data can be made available to all users and controlled at group level which rows and columns will be made available to each user group.
* Dynamic Data Masking, certain PII information can be masked for certain user groups while they still have access to the rest of the data.

With the above requirements and considering this is only a POC, they decided to use *Azure SQL Database* for serving layer storage solution. The team acknowledges that after successful POC they will move this part of the solution to *Azure Synapse Analytics*.

***Data Science:***

In addition, the data science team decided to use *Azure ML services* to build ML/AI applications, particularly to support the nutritional suggestions based on the SmartFoods data. Hence, to avoid the need to export the cleansed data from SQL DB they requested the data to be stored in *Blob storage* as well if possible, at loading time. After considering all this requirement it was decided to use *Parquet files* on Azure Blob Storage (after POC to be replaced with Azure Data Lake Storage Gen2)

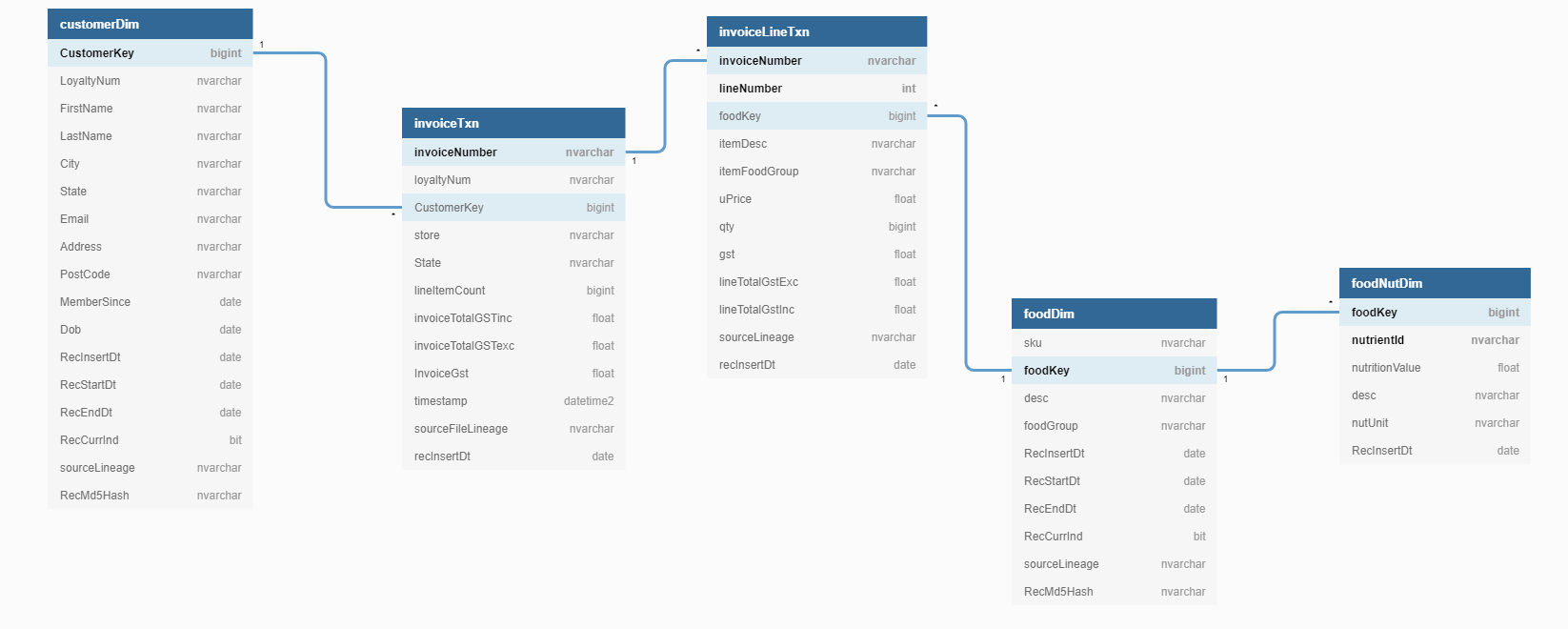
***SmartFoods App:***

Finally, the SmartFoods application (Web and Mobile) will need to access the data through an API and their primary requirements are performance, scalability and availability. After considering all this requirement it was decided to use *Azure CosmosDB* for application data storage.

In the Blob container we copied for SmartFoods there are multiple CSV files which represents SmartFood’s reference data for the transactions that comes through the HTTP API.

#### Create Data warehouse tables for SmartFoods in Azure SQLDB

Here is the initial star schema we are building for SmartFoods DW. Later we will also introduce some aggregate tables for easier reporting.



Either using Query Editor in Azure Portal or using SSMS connect to your Azure SQL DB and create a schema for SmartFoods DW and all the tables by running the following SQL script.

**Note:** You may need to add your Client IP Address to your SQL DB through “Set Firewall” page.

CREATE SCHEMA SmartFoodsDW;  
GO  
CREATE TABLE [SmartFoodsDW].[customerDim](  
 [CustomerKey] [bigint],  
 [LoyaltyNum] [nvarchar](max),  
 [FirstName] [nvarchar](max) NULL,  
 [LastName] [nvarchar](max) NULL,  
 [City] [nvarchar](max) NULL,  
 [State] [nvarchar](max) NULL,  
 [Email] [nvarchar](max) NULL,  
 [Address] [nvarchar](max) NULL,  
 [PostCode] [nvarchar](max) NULL,  
 [MemberSince] [date] NULL,  
 [Dob] [date] NULL,  
 [RecInsertDt] [date] NULL,  
 [RecStartDt] [date] NULL,  
 [RecEndDt] [date] NULL,  
 [RecCurrInd] [bit] NULL,  
 [sourceLineage] [nvarchar](max),  
 [RecMd5Hash] [nvarchar](max)   
) ;  
GO  
CREATE TABLE [SmartFoodsDW].[foodDim](  
 [sku] [nvarchar](max),  
 [foodKey] [bigint],  
 [desc] [nvarchar](max) NULL,  
 [foodGroup] [nvarchar](max) NULL,  
 [RecInsertDt] [date] NULL,  
 [RecStartDt] [date] NULL,  
 [RecEndDt] [date] NULL,  
 [RecCurrInd] [bit] NULL,  
 [sourceLineage] [nvarchar](max),  
 [RecMd5Hash] [nvarchar](max)   
) ;  
GO  
CREATE TABLE [SmartFoodsDW].[foodNutDim](  
 [foodKey] [bigint],  
 [nutrientId] [nvarchar](max),  
 [nutritionValue] [float] NULL,  
 [desc] [nvarchar](max) NULL,  
 [nutUnit] [nvarchar](60) NULL,  
 [RecInsertDt] [date] NULL  
);  
GO  
  
CREATE TABLE [SmartFoodsDW].[invoiceLineTxn](  
 [invoiceNumber] [nvarchar](max),  
 [lineNumber] [int],  
 [foodKey] [bigint],  
 [itemDesc] [nvarchar](max) NULL,  
 [itemFoodGroup] [nvarchar](max) NULL,  
 [uPrice] [float],  
 [qty] [bigint],  
 [gst] [float],  
 [lineTotalGstExc] [float],  
 [lineTotalGstInc] [float],  
 [sourceLineage] [nvarchar](max),  
 [recInsertDt] [date]   
);  
GO  
CREATE TABLE [SmartFoodsDW].[invoiceTxn](  
 [invoiceNumber] [nvarchar](max),  
 [loyaltyNum] [nvarchar](max) NULL,  
 [CustomerKey] [bigint] NULL,  
 [store] [nvarchar](max) NULL,  
 [State] [nvarchar](max) NULL,  
 [lineItemCount] [bigint],  
 [invoiceTotalGSTinc] [float],  
 [invoiceTotalGSTexc] [float],  
 [InvoiceGst] [float],  
 [timestamp] [datetime2](7),  
 [sourceFileLineage] [nvarchar](max),  
 [recInsertDt] [date]   
) ;  
GO

#### Introduction to Mapping Data Flows

Mapping Data Flows is a new feature of Azure Data Factory that allows you to build data transformations in a visual user interface (code-free or very low amount coding).

Mapping data flows are visually designed data transformations in Azure Data Factory. Data flows allow data engineers to develop graphical 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 activities can be engaged via existing Data Factory scheduling, control, flow, and monitoring capabilities. More info in this [article](https://docs.microsoft.com/en-us/azure/data-factory/concepts-data-flow-overview).

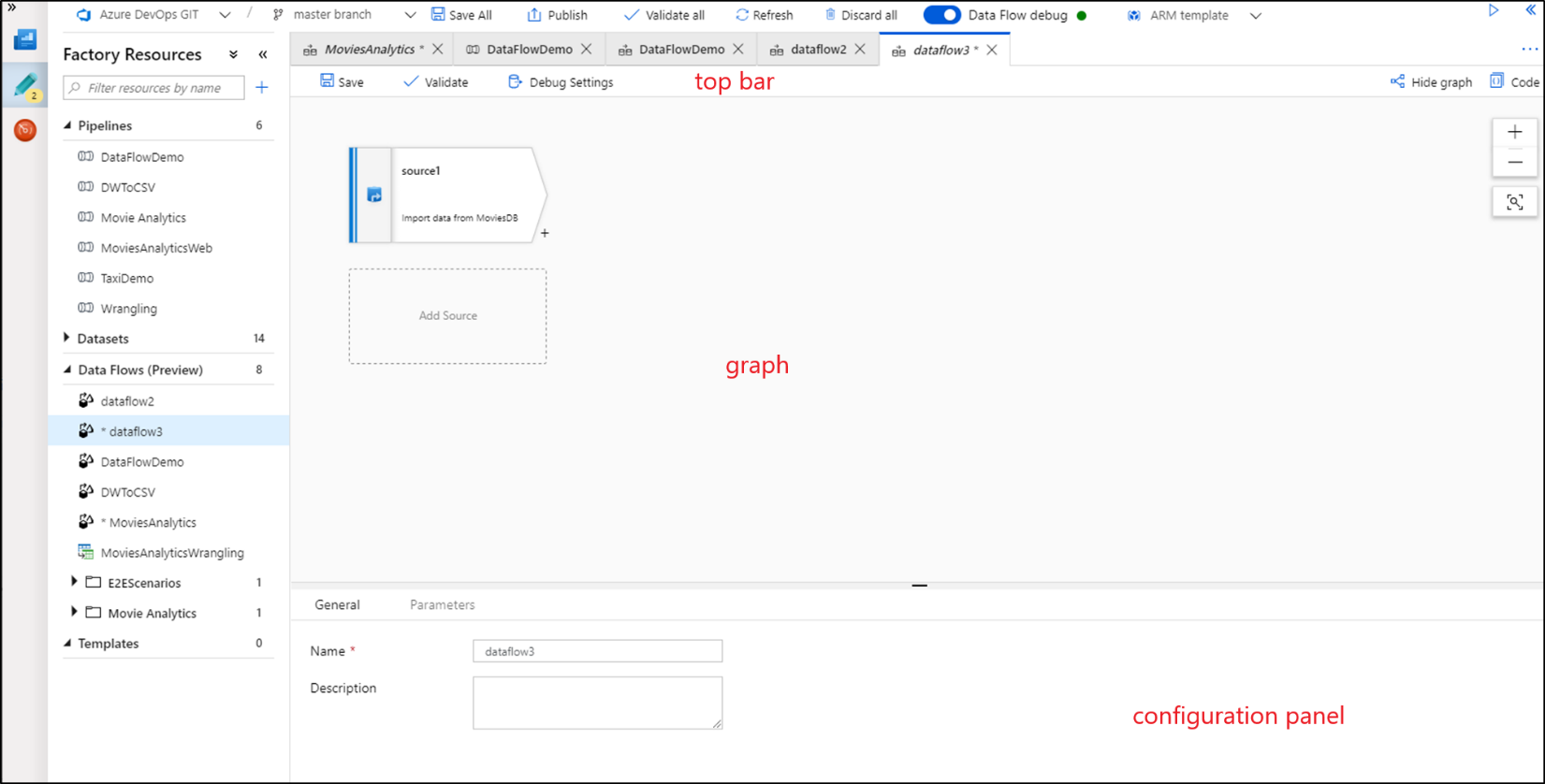
Mapping data flows provide an entirely visual experience with no coding required. Your data flows run on your execution cluster for scaled-out data processing. Azure Data Factory handles all the code translation, path optimization, and execution of your data flow jobs.

ADF translates the flow built in the visual interface to Apache Spark code which will run on serverless Spark cluster than we can configure in terms of count and type of worker nodes.

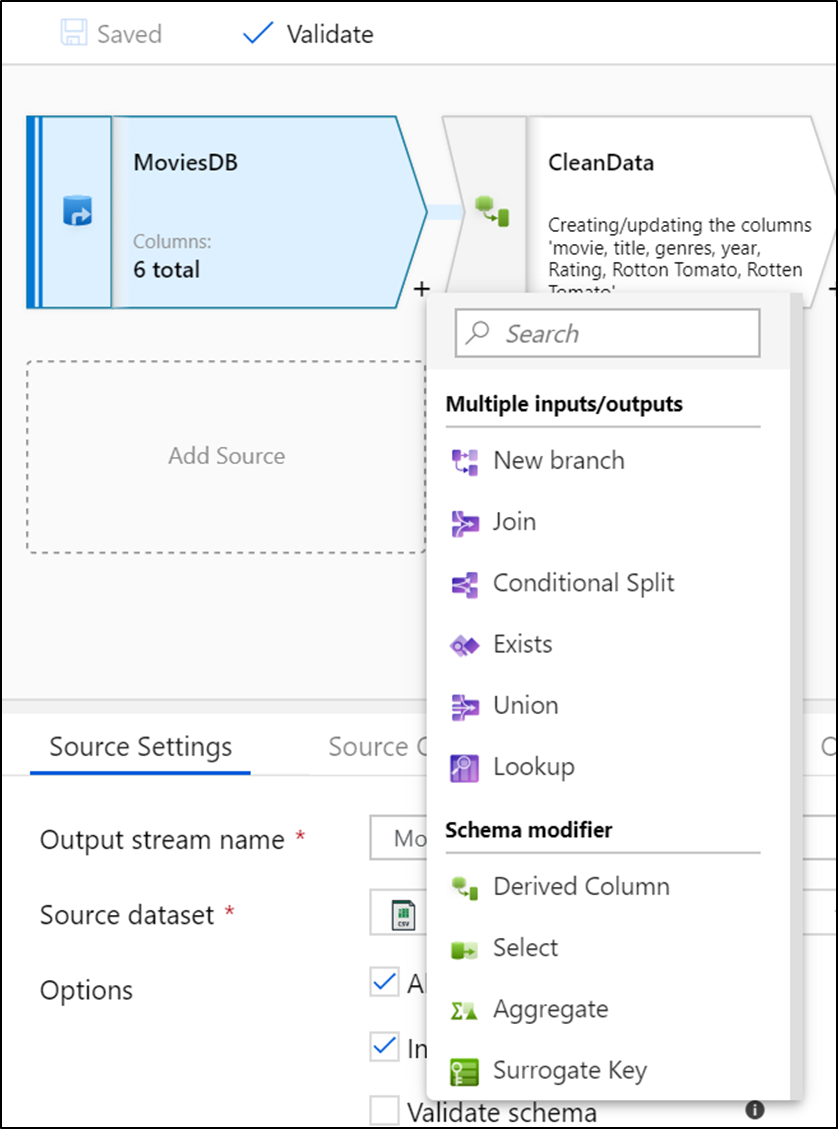
**Serverless Spark cluster**: The Apache Spark cluster will be deployed on Azure Integration Runtime and like Azure IR, which is serverless, the cluster is fully managed by Azure and charged per number of seconds the job takes to run.

**Mapping DF on SH-IR?** ADF Spark clusters are only deployable on Azure and currently there is no option for deploying on-prem.

**Data flow canvas:** Here is what the Data flow canvas look like. It is separated to three parts



**The Graph:** The graph displays the transformation stream. It shows the lineage of source data as it flows into one or more sinks. To add a new source, select Add source. To add a new transformation, select the plus sign on the lower right of an existing transformation.



## Slowly changing dimension type 2 withMapping dataflow (customerDim)

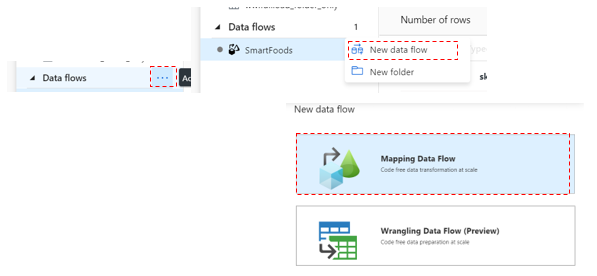
#### Create a new Mapping Dataflow and add source dataset

You loaded SmartFoods’ customer staging data from API to Blob storage in CSV format in the following location “smartfoodsstaging/customer/smartfoods\_customers\_<date>.csv”

The ultimate table will look like below:

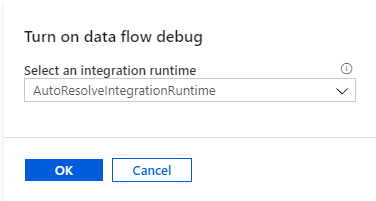
|  |  |  |
| --- | --- | --- |
| Field | Description | Source |
| CustomerKey bigint – primary key | Surrogate Key | Generated by ELT |
| LoyaltyNum nvarchar | Source Key | Existing field in source |
| FirstName nvarchar | From Name field | Generated by ELT |
| LastName nvarchar | From Name field | Generated by ELT |
| City nvarchar |  | Existing field in source |
| State nvarchar |  | Existing field in source |
| Email nvarchar |  | Existing field in source |
| Address nvarchar |  | Existing field in source |
| PostCode nvarchar |  | Existing field in source |
| MemberSince date |  | Existing field in source |
| Dob date |  | Existing field in source |
| RecInsertDt date | Actual ELT running date | Generated by ELT |
| RecStartDt date | Record validity start date = batch date | Generated by ELT |
| RecEndDt date | Record validity end date = batch date | Generated by ELT |
| RecCurrInd Boolean | Record validity indicator | Generated by ELT |
| sourceLineage nvarchar | Name of source file | Generated by ELT |
| RecMd5Hash nvarchar | MD5 Hash of all source fields except natural key | Generated by ELT |

1. Create a mapping Dataflow by clicking on new Data flow button and rename it to “SmartFoodsCustomerELT”



1. At the top of the page turn on the “data flow debug” -> select *AutoResolveIntegrationRuntime*

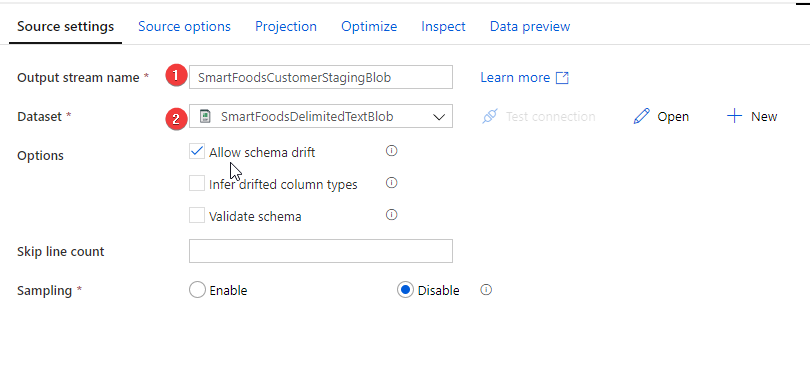




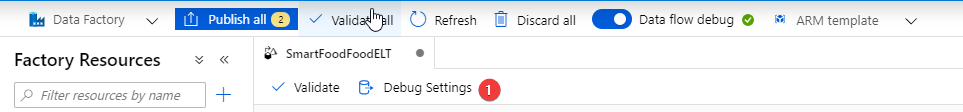
**Debug mode:** Azure Data Factory mapping data flow's debug mode allows you to interactively watch the data shape transform while you build and debug your data flows. The debug session can be used both in Data Flow design sessions as well as during pipeline debug execution of data flows. To turn on debug mode, use the "Data Flow Debug" button at the top of the design surface. For more information check [this](https://docs.microsoft.com/en-us/azure/data-factory/concepts-data-flow-debug-mode) article.

**Note:** By turning on Debug mode, ADF deploys a Spark cluster on the same region as your Integration Runtime.

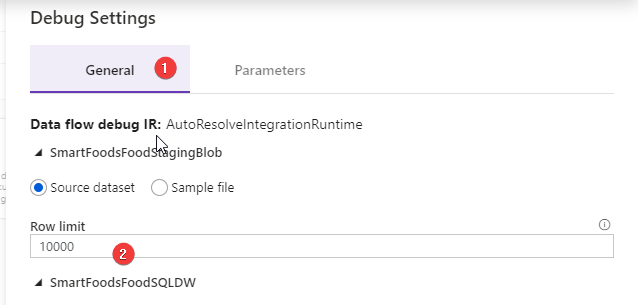
1. Click “Add Source” on canvas
2. Change the output stream name to: “SmartFoodsCustomerStagingBlob”
3. Select the ‘*SmartFoodsDelimitedTextBlob’*



1. Click “Debug Settings” on the top task bar

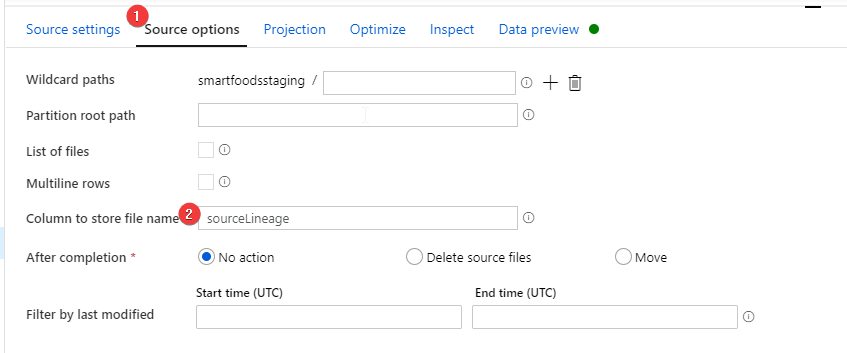


1. Under “General” increase “*Row limit”* to 10,000



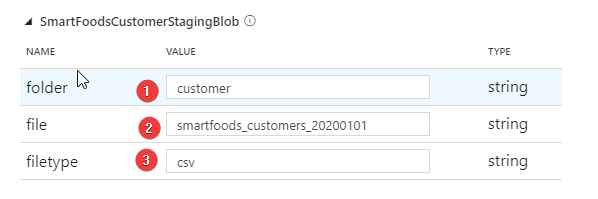
**Row Limit:** This is the maximum number of rows that will be retrieved from the source data set in when we try to preview the transformation results. If you are working with multiple datasets that needs to be joined, it is best to increase this to a higher limit, otherwise the join result in “preview” will have a lot of missing records.

1. Under source options change “Column to store file name” to “sourceLineage”



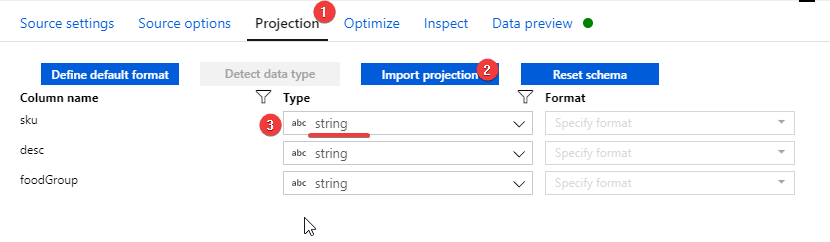
**Note:** This enables the option to take the processing file name and pass it on as a column for every row in the dataset. This is really valuable information for testing, lineage and debugging.

1. Under parameters provide
   * “folder” = customer
   * “file” = smartfoods\_customers\_20200101
   * “fileType” = csv



**Where did the parameters came from?** We are reusing the parametrized dataset created previously for importing data from source systems to blob storage. These are the parameters that dataset needs to operate.

1. Go to “projection tab”
   1. click “*import Projection*”
   2. Change the data type for “Postcode” column from “short” to “string”



**Note:** When we load data from a delimited text or flat file, it is always recommended to double check the schema ADF detected and if needed fix it.

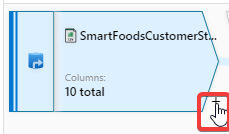
**Why changing SKU from Short to String?** 1. The destination sink (Azure SQLDB) does not recognized values of type short. 2. Postcodes can have leading 0 in it which gets eliminated in non-string type fields

1. Now go to “Data Preview” tab and refresh it to get a preview of the dataset

Debug Cluster: We need “Debug” mode running for 1. Importing data projection (schema) 2. Running preview task.

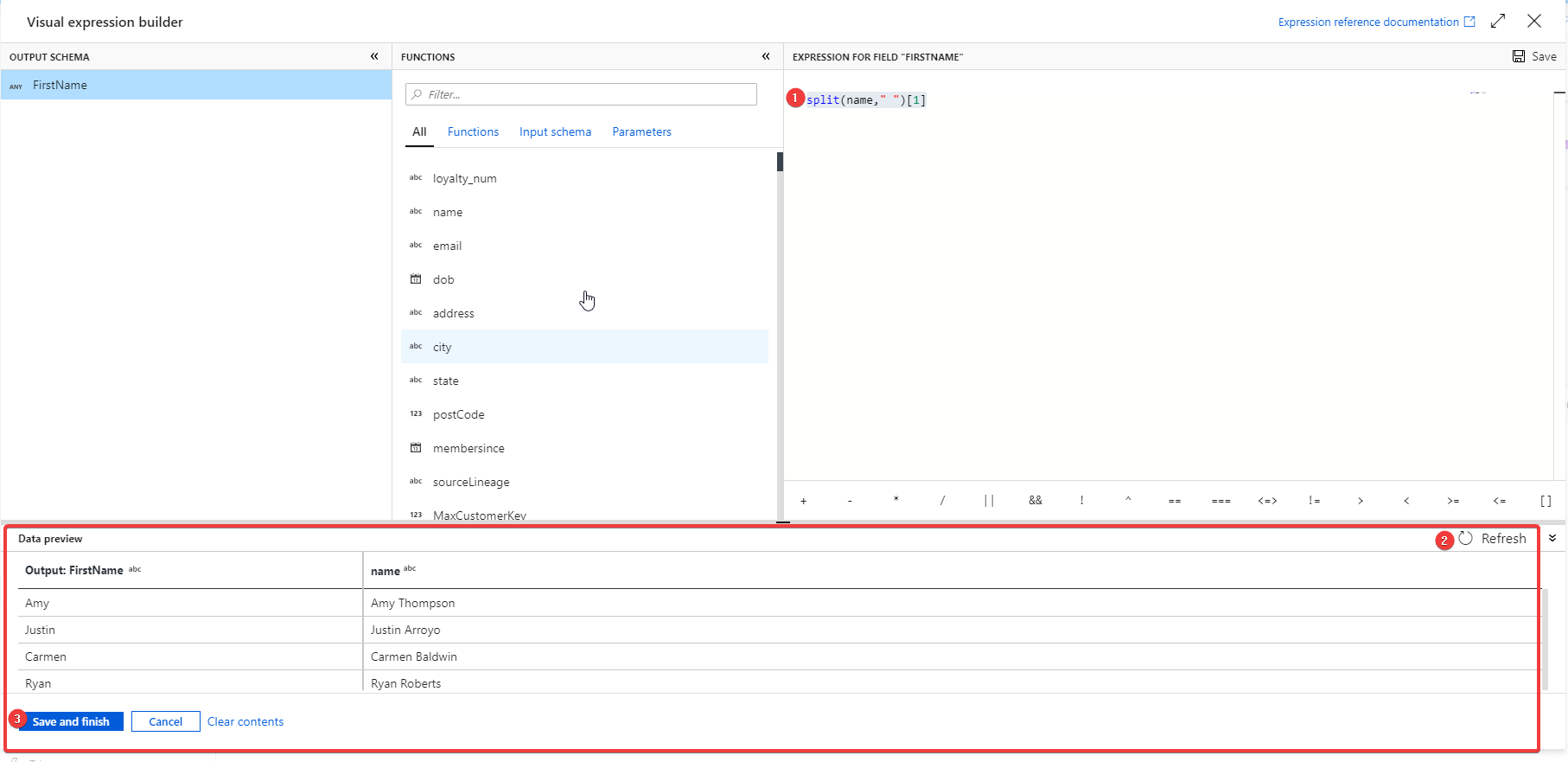
#### Break the Name field to firstName and lastName fields

1. Click the plus sign on the lower right-hand side of source transformation to add the next transformation.



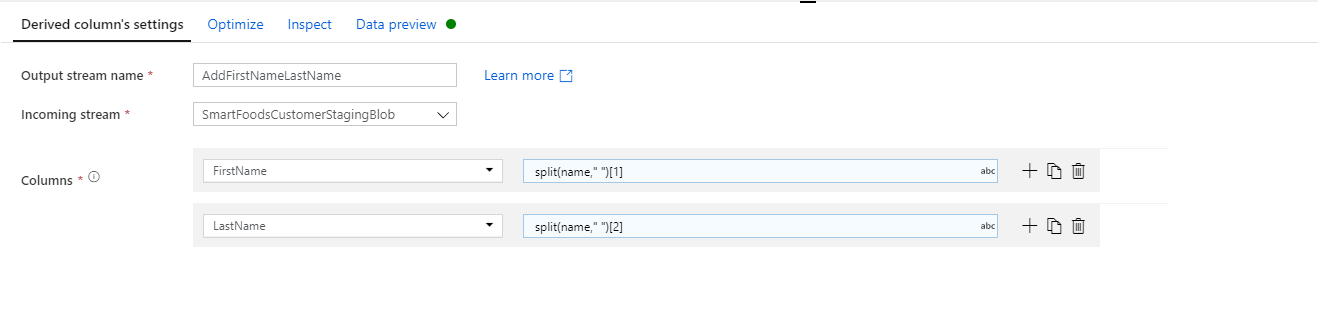
1. Select “Derived column” transformation
2. Rename it to “AddFirstNameLastName”
3. Add a new column and for name type “FirstName” for value click on the right box and it opens the expression editor and enter following expression.
4. Click refresh to see the result of the expression on the data.

split(name," ")[1]



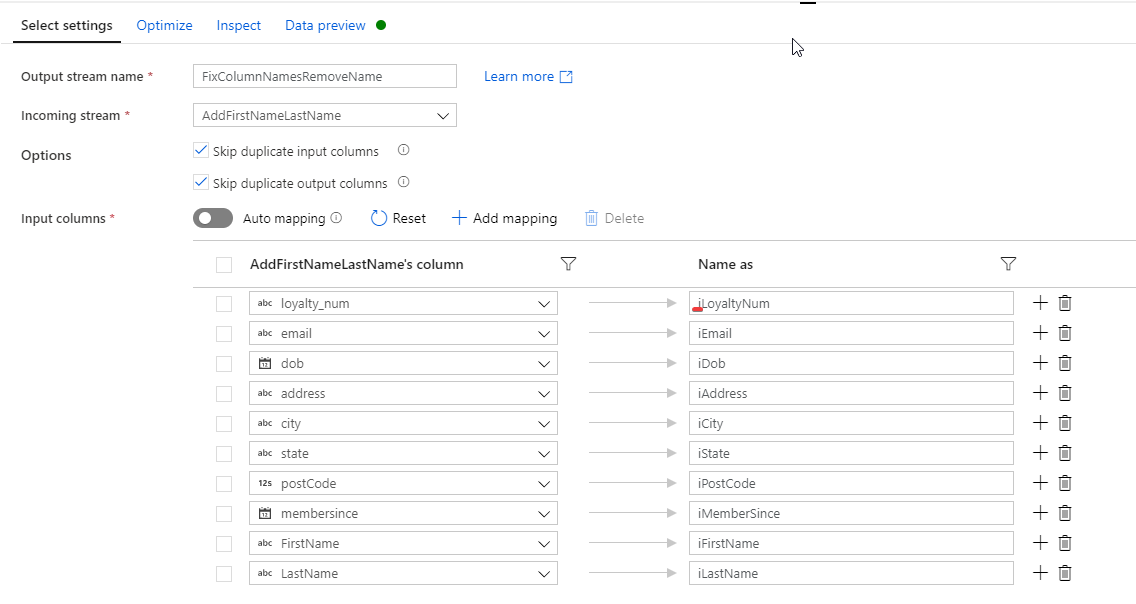
1. Create another column for “LastName” using below expression

split(name," ")[2]

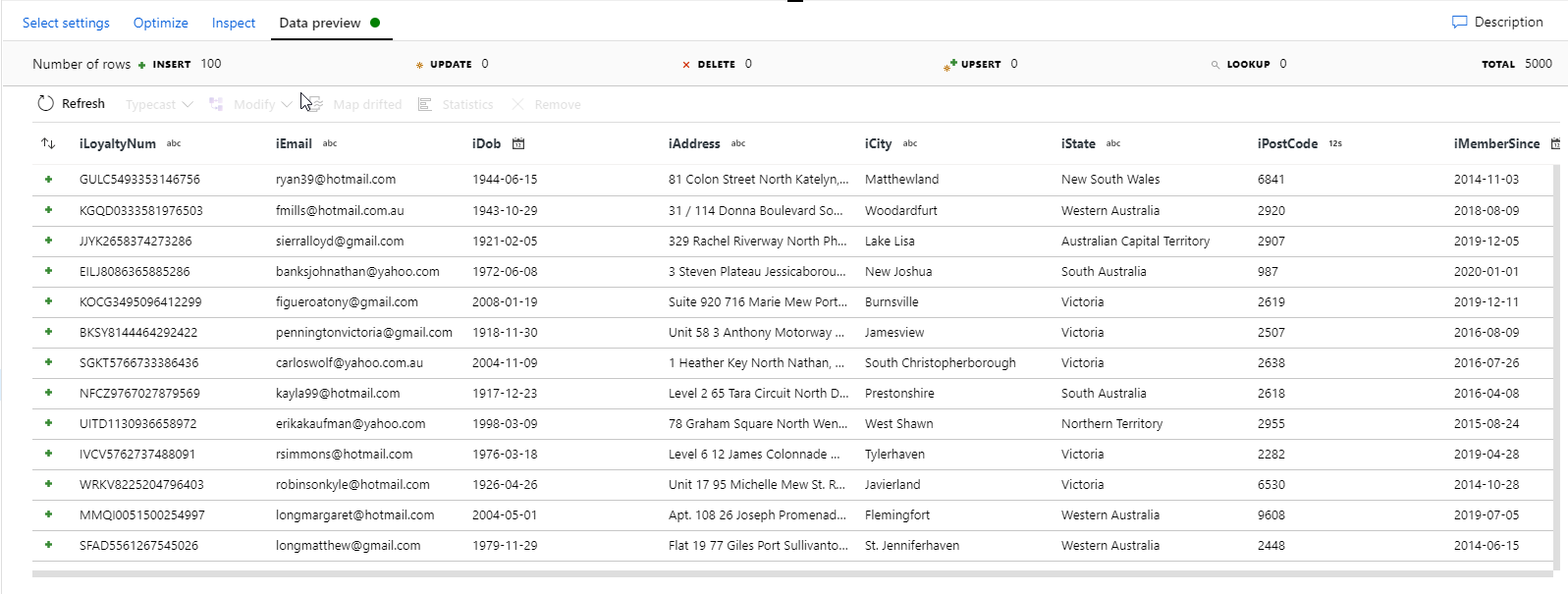


#### Remove extra columns and Rename columns using “Select” transformation

1. Add a “Select” transformation after the previous transformation to
   1. Remove the “name” column in favor or “firstName” and “LastName”.
   2. We are going to compare this data with existing data in the dimension to identify changes an newly added rows so for easier identification of columns we add an ‘i’ in front of all columns.
2. Rename your “Select” transformation to “FixColumnNamesRemoveName”
3. Set it up as below screenshot



1. Preview the output of this transformation



#### Calculate MD5 Hash of all non-key columns

**Note:** If you are not familiar with Slowly Changing Dimensions in Data Warehousing read this article in Wikipedia <https://en.wikipedia.org/wiki/Slowly_changing_dimension>

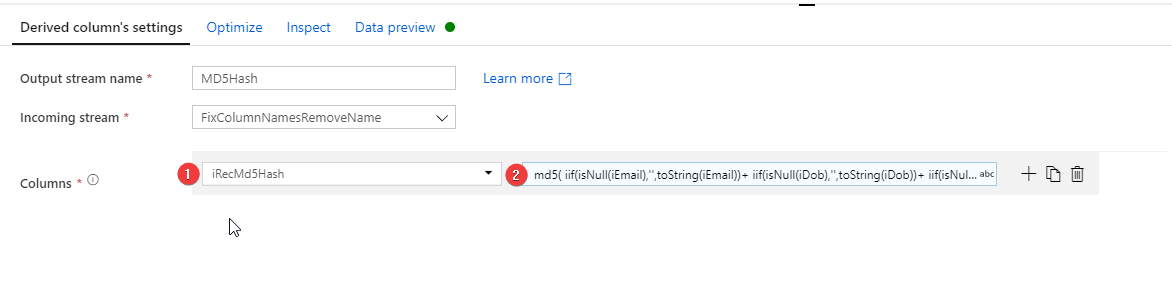
We are trying to build a SCD type 2 for customers table. This means no data will be discarded from the DW table. If a record changes in the OLTP source system this change gets captured by adding a new record with updated values and marking the old record as inactive(usually referred to as closing record)(recCurrInd), plus adding the date/timestamp of the record closure (recEndDate).

There are multiple technics to identify updated records, one is to compare the MD5 Hash of the existing records against the newly received record from source and if they do not match it is considered a change.

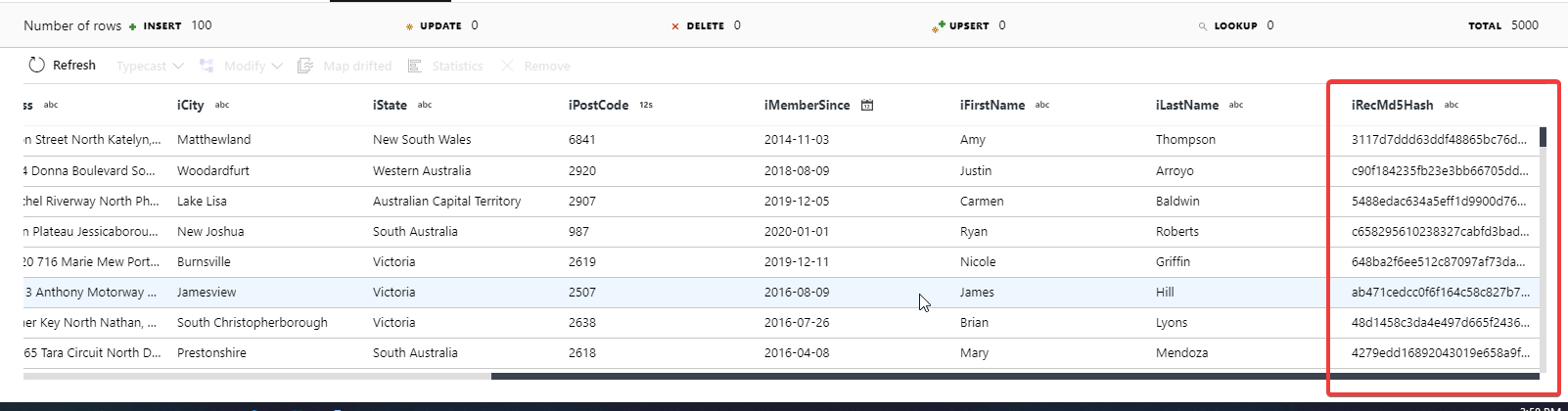
1. Add a ‘Derived column’ transformation to calculate MD5 Hash of all non-key columns
2. Rename it to “MD5Hash”
3. Add a column “iRecMd5Hash”
4. For expression use

md5( iif(isNull(iEmail),'',toString(iEmail))+   
iif(isNull(iDob),'',toString(iDob))+   
iif(isNull(iAddress),'',toString(iAddress))+   
iif(isNull(iCity),'',toString(iCity))+   
iif(isNull(iState),'',toString(iState))+   
iif(isNull(iPostCode),'',toString(iPostCode))+   
iif(isNull(iMemberSince),'',toString(iMemberSince))+   
iif(isNull(iFirstName),'',toString(iFirstName))+   
iif(isNull(iLastName),'',toString(iLastName)))

**Expression explanation:** For every non-key column first, we replace all Nulls with an empty string, then convert all fields to string and concatenate them together. Finally use the md5 method to calculate the hash of the whole concatenated string.



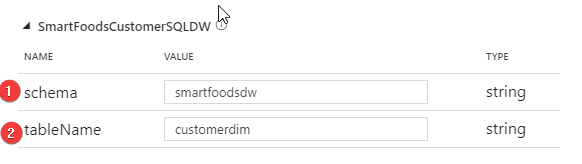
1. Preview the output



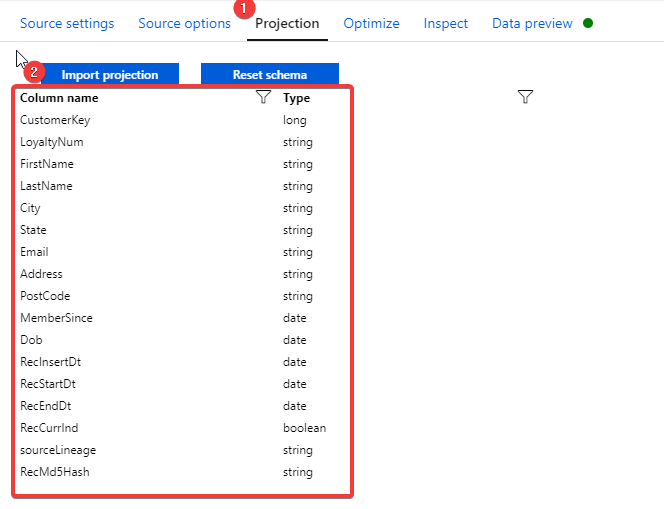
#### Add DW table source

To perform the comparison and identify updated or new records we need to also retrieve the existing data from DW dimension table (at first run the table is empty, hence every row will be determined as new).

1. On the far-left hand side of the canvas under the first source click “Add source” to add a new source to the flow.
2. Rename it to “SmartFoodsCustomerSQLDW”
3. For dataset Select “AzureSqlTable1” (You created this SQL Dataset at the beginning of the lab)
4. Go to “Debug Settings” and provide the below parameters:



1. Go to Projection tab and import the dataset projection

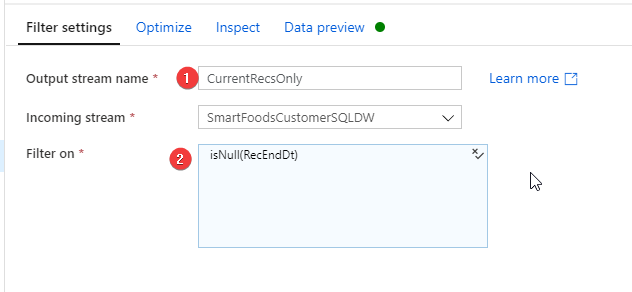


1. Add a filter transformation after the DW source

**Note:** The filter transformation is used to only get ‘active’ rows (RecEndDt is Null) from the table.

1. Rename it to “CurrentRecordsOnly”
2. Filter on:

isNull(RecEndDt)

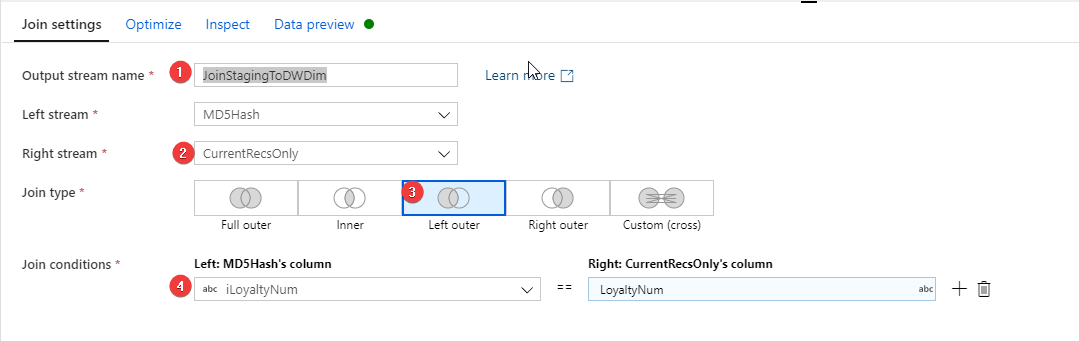


#### Compare staging records with DW records to identify updates and inserts

Next step is to compare the records from staging and DW. We use a “join” transformation for this purpose. As the staging dataset will have records that do not exists in DW dataset we need to use an “left outer join” the sudo code for the join is:

SELECT \* FROM staging s  
LEFT OUTER JOIN  
dw\_table d  
ON  
s. key = d.key

1. Add a join transformation after “MD5Hash” transformation
2. Rename it to “JoinStagingToDWDim”
3. For right stream select “CurrRecsOnly”
4. Join type: “Left outer”
5. Join conditions: iLoyaltyNum == LoyaltyNum



**Note:** Since we renamed the staging columns with an ‘i’ in front of them it is quite easy to find the right column for joins here.

#### Identify Updates/Inserts using “Conditional Split” transformation

A “Conditional Split” transformation allows us to split an incoming dataset into multiple outgoing datasets based on some logical criteria.

Here we need to find out

* If a record is new (if the right-hand side [records from DW] of left outer join is null)
* If a record has Changed (if the primary key existed within both staging and DW datasets but MD5hashes are not matching).
* If a record has Not Changed (if the primary key existed within both staging and DW datasets and MD5hashes are matching).

1. Add a “Conditional split” transformation after “JoinStagingToDWDim” transformation.
2. Rename it to “SDC2Split”
3. For Split on option set it to “First matching condition”

**Split On**: If we set this to “First matching condition” the first condition a record fits with will be pushed to that stream and condition(s) after that will not be tested on the record. First. This option is more efficient in processing but has two implications: 1. The order of conditions becomes important (stricter conditions should be placed above less strict ones) 2. Every record only gets passed into a single stream. If your workflow logic requires input records to get passed into multiple output streams choose “All matching conditions”

1. Split Condition
   1. New: isNull(LoyaltyNum)
   2. Changed: !(isNull(LoyaltyNum)) && (iRecMd5Hash \!=RecMd5Hash)
   3. Unchanged: !(isNull(LoyaltyNum)) && (iRecMd5Hash == RecMd5Hash)

#### (Challenge Task) Create customer dimension

In the SmartFoods Blob container that we copied previously there was a file named “customer.csv” which contains the customers’ reference data.

We would like to create a dimension table for this data source as below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CustomerKey | LoyaltyNum | Firstname | Lastname | City | State | EmailAddress | MemberSince | Dob | RecInsertDt |

Note 1: The source is providing “name” field, which is full name, but we need to separate first name and last name

Note 2: We know some of the email addresses of customers are NOT the right format ([abc@xyz.com](mailto:abc@xyz.com)) and we need to replace these with NULL instead

**Optional Extra challenge:** WWI also likes to calculate the age of the customer as well and store in “Age” column can you used Mapping Data flows Expression Language to calculate it?

**Table DDL:**

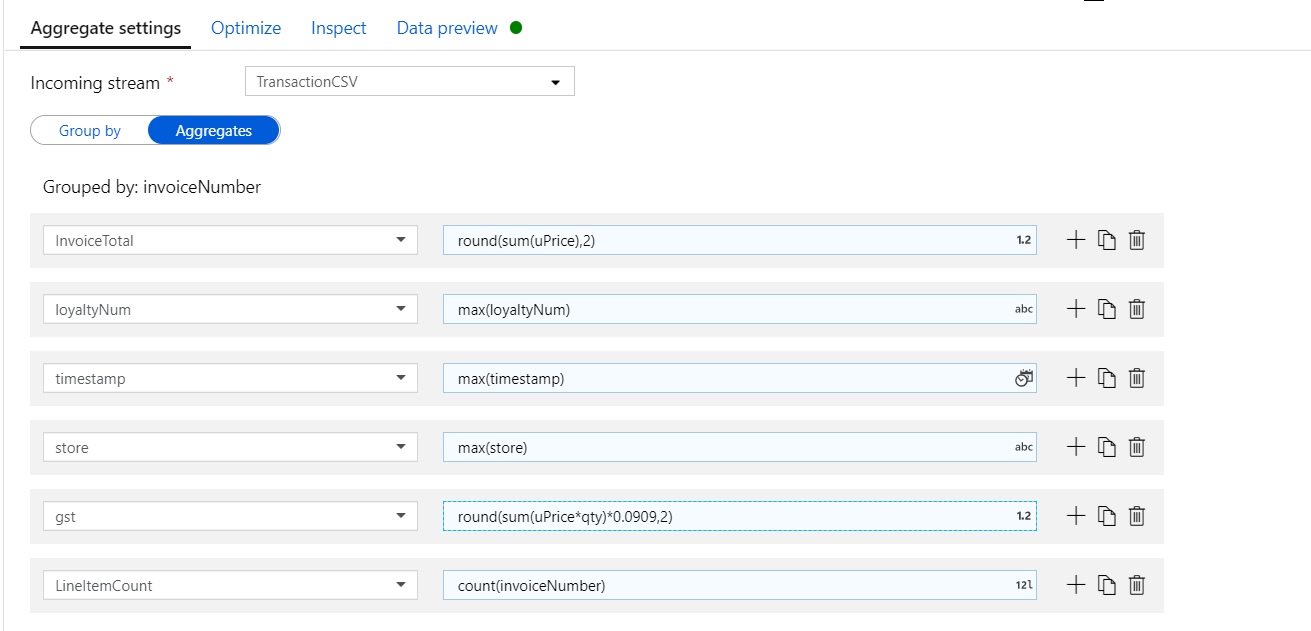
CREATE TABLE [SmartFoodsDW].[customer](  
 [CustomerKey] [bigint] NULL,  
 [LoyaltyNum] [nvarchar](max) NULL,  
 [FirstName] [nvarchar](max) NULL,  
 [LastName] [nvarchar](max) NULL,  
 [City] [nvarchar](max) NULL,  
 [State] [nvarchar](max) NULL,  
 [Email] [nvarchar](max) NULL,  
[Address] [nvarchar](max),  
[PostCode] [nvarchar](max),  
 [MemberSince] [date] NULL,  
 [Dob] [date] NULL,  
 [RecInsertDt] [date],  
 [RecStartDt] [date],  
 [RecEndDt] [date] NULL,  
 [RecCurrentInd] [bit],  
 [RecMd5Hash] [nvarchar](max)  
) ;  
GO

**Final Data Flow:**

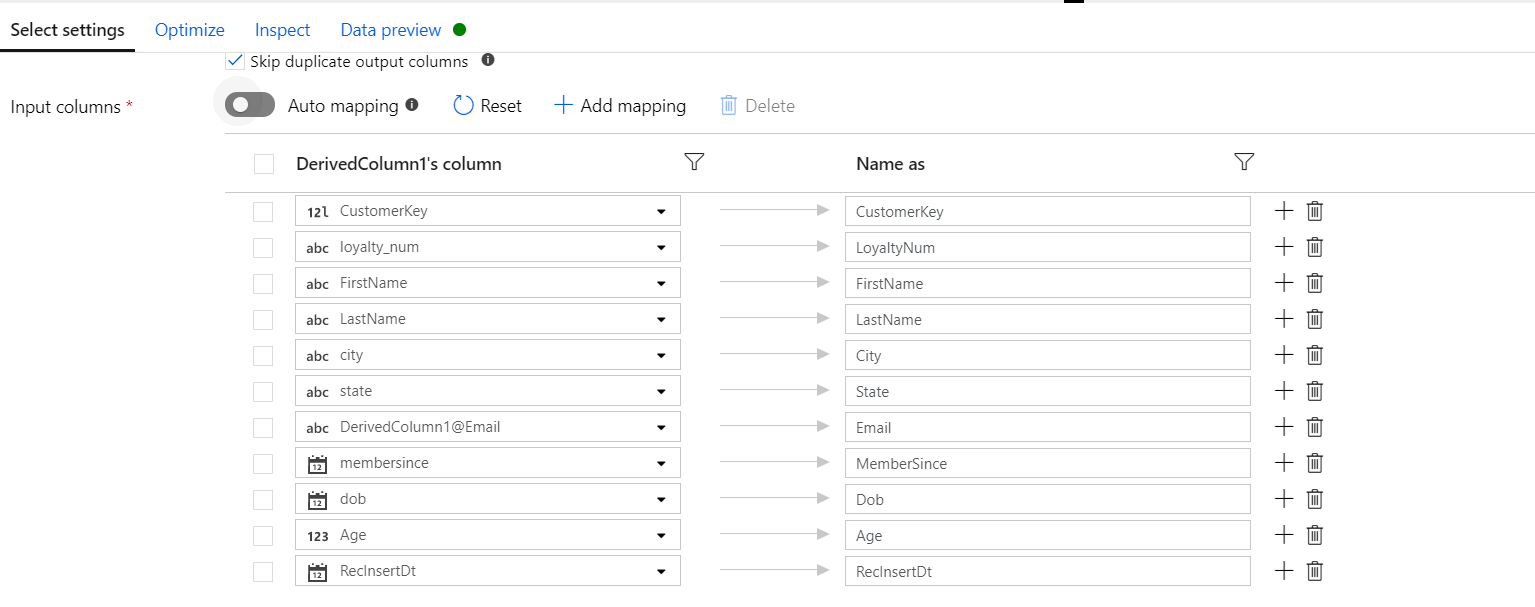


\*\*If you are stuck or want to double check your answer the solution for Expression Language and Select transformation is in the next page.  
\*\*

**Derived column expressions solution:**

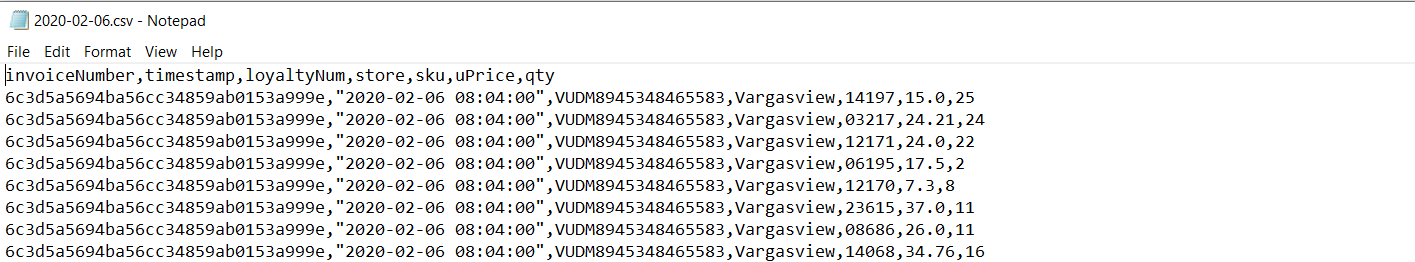


**Select transformation:**



#### Create SmartFoods Invoice fact tables

The Data that we retrieved in the previous exercise from SmartFoods Transaction API seems to be in an uncommon format for invoices. Usually invoice data has an invoice header and an invoice item lines but for the case of SmartFoods the API is only capable of providing the data in form of line items with repeated invoice header information.



The requirement is to create two separate tables in following form:

Invoice

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| InvoiceNumber | CustomerKey | Store | InvoiceDtts | InvoiceTotal | Gst | NumLineItems | RecInsertDt |

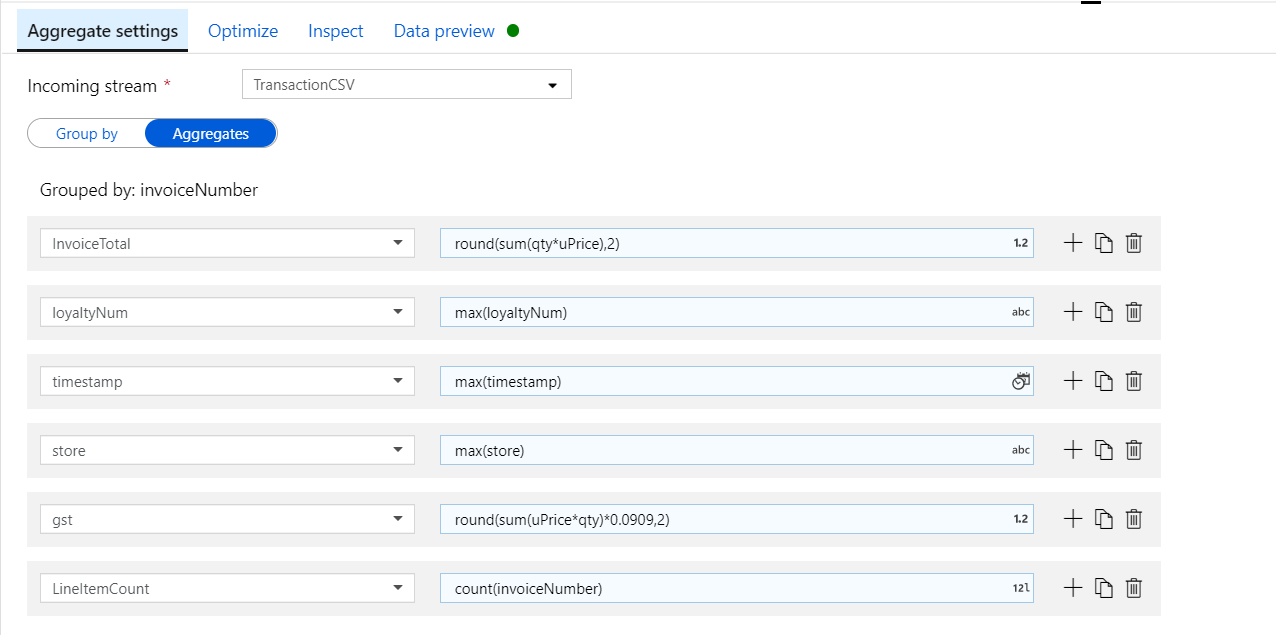
InvoiceLine

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| InvoiceNumber | ItemKey | ItemDescription | UnitPrice | Qty | Gst | RecInsertDt |

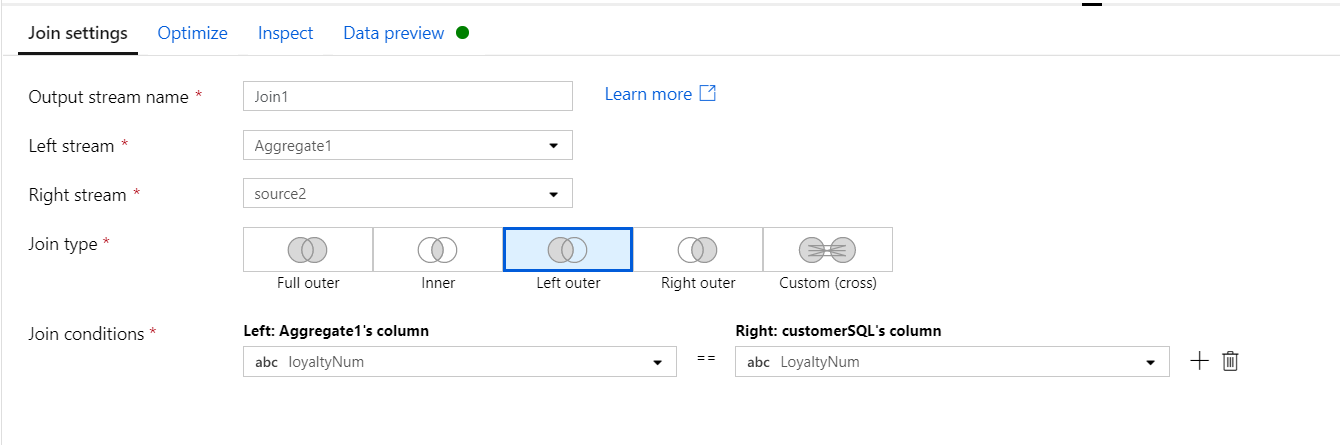
1. **For Invoice Table Overall Data flow looks:**



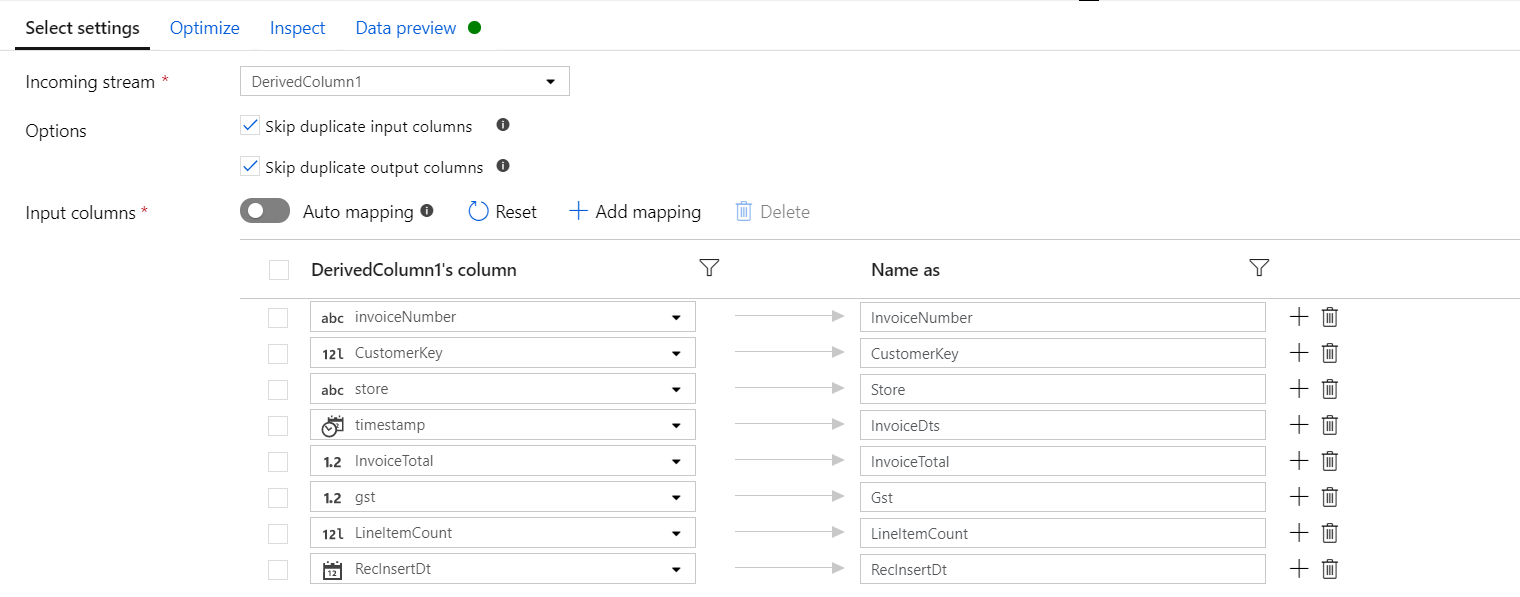
Aggregate transformation:



Join transformation:

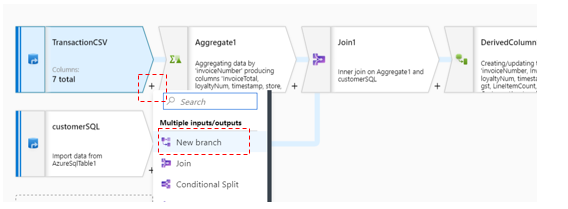


Select Transformation:

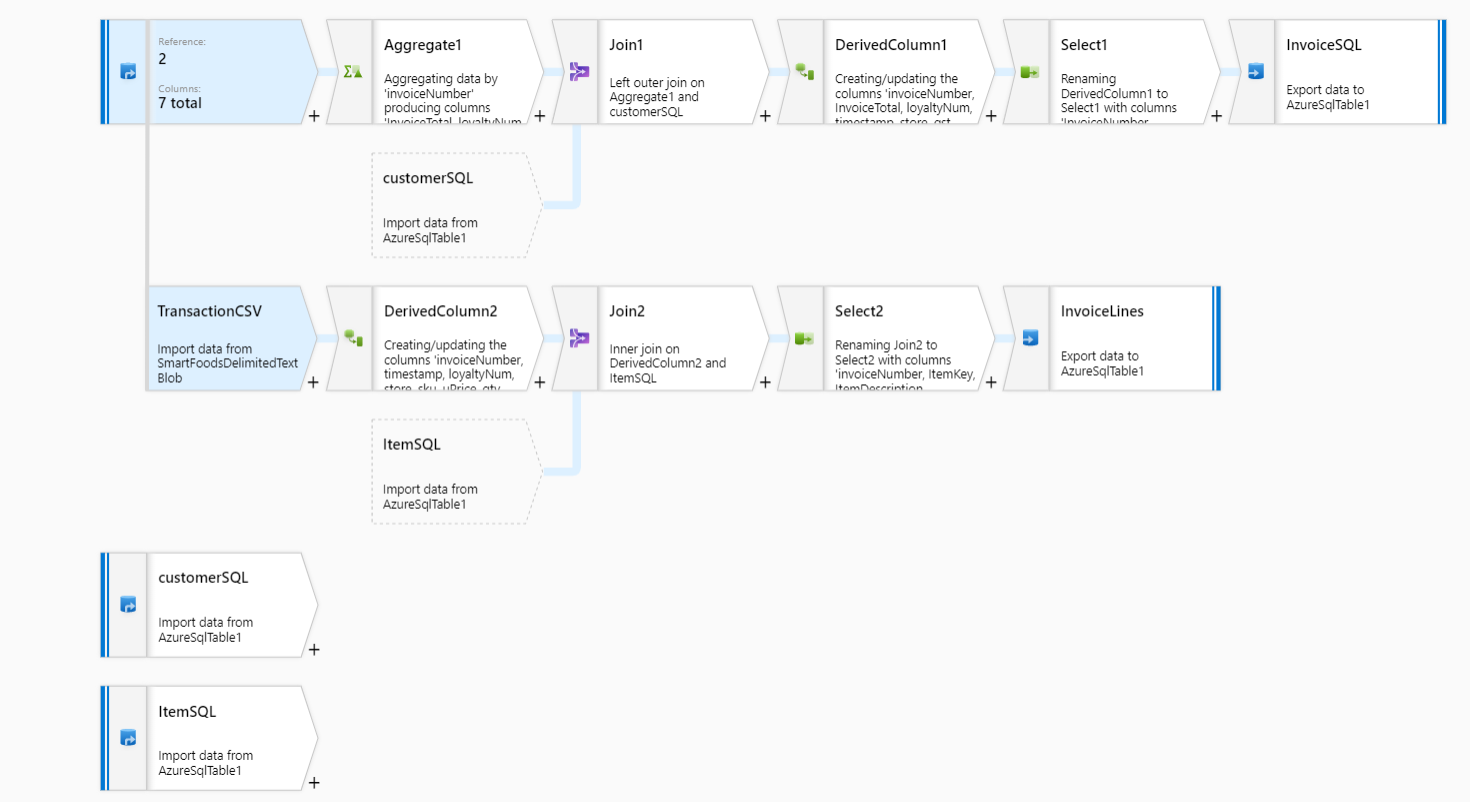


1. **For Invoice Lines:**

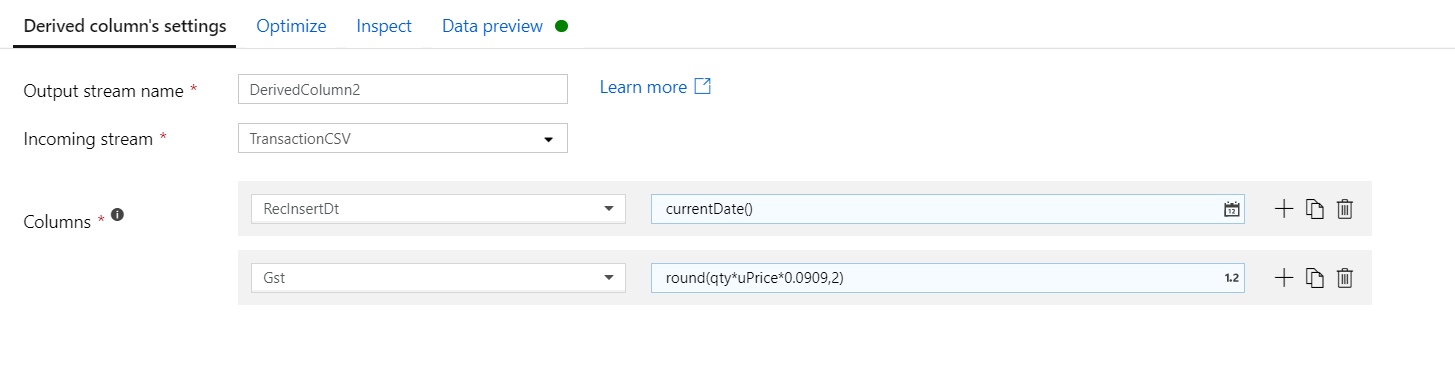
In the **same** data flow after your source CSV add a new branch transformation. This will branch the same data source to two different pathes



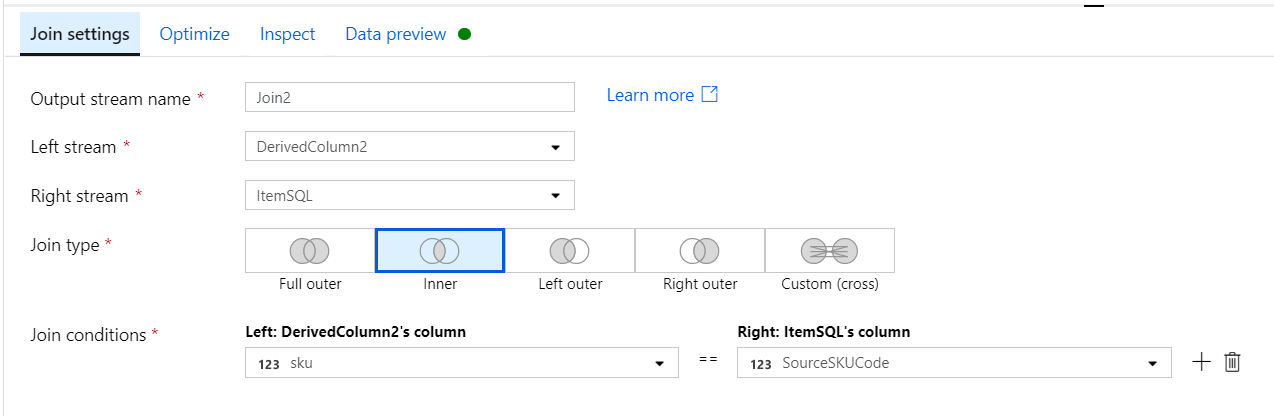
**Final Data flow for invoice and invoice line:**



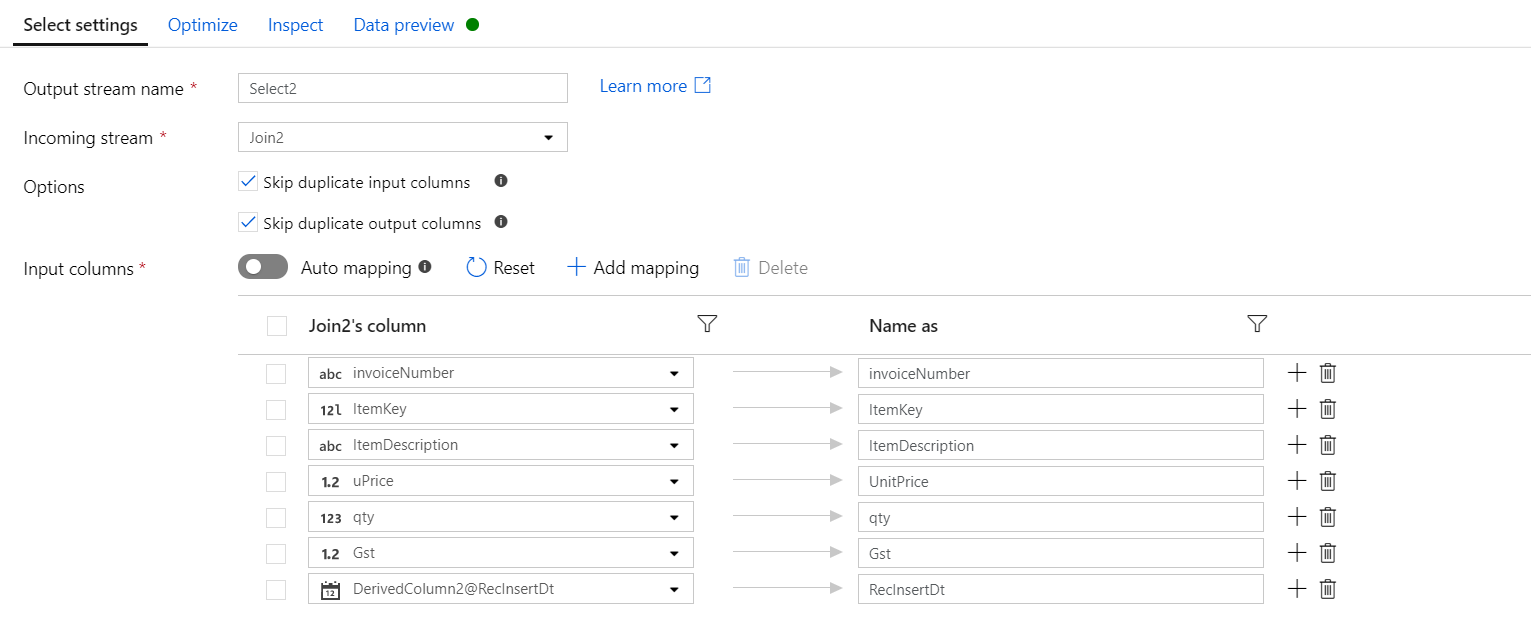
**Derived Column Transformation:**



**Join transformation:**



**Select Transformation:**



**DDLS for InvoiceLine table:**

CREATE TABLE [smartfoods].[invoiceline](  
 [invoiceNumber] [nvarchar](max) NULL,  
 [ItemKey] [bigint] NULL,  
 [ItemDescription] [nvarchar](max) NULL,  
 [UnitPrice] [float] NULL,  
 [qty] [int] NULL,  
 [Gst] [float] NULL,  
 [RecInsertDt] [date] NULL  
);  
GO