Azure Data Factory LAB

Prerequisites

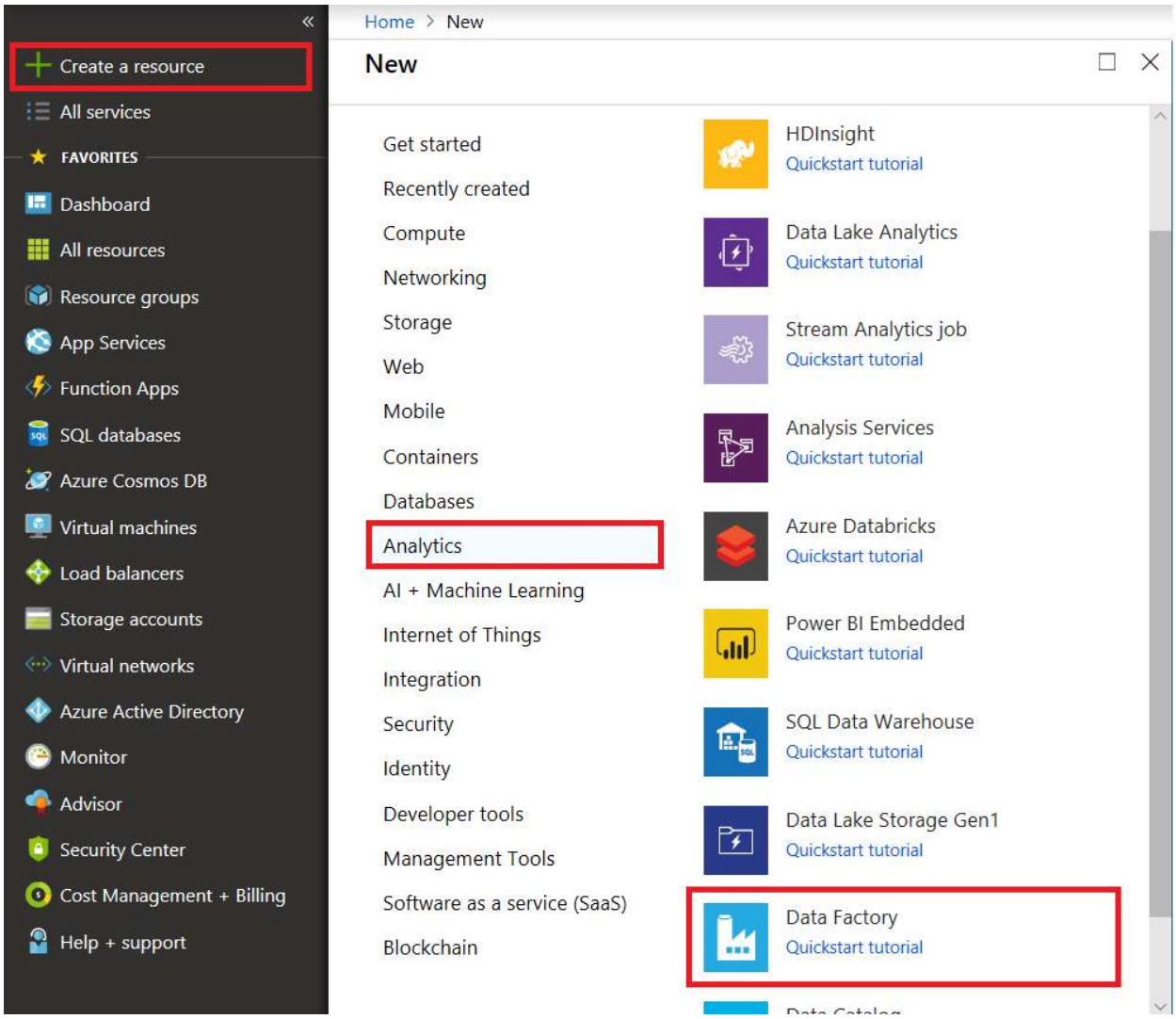
- Azure SQL Data Warehouse: The data warehouse holds the data that's copied over from the SQL database. You created your Azure SQL Data Warehouse in previous lab.
- Azure SQL Database: This tutorial copies data from an Azure SQL database with Adventure Works LT sample data. We have created this DB for you however should it be unavailable, follow this link to create it in your environment: Create an Azure SQL database.

  SQL DB name: AcademySQLDatabase
  SQL Server name: academysqlserver
  User name: academyuser
  Password: Azure4Academy2019

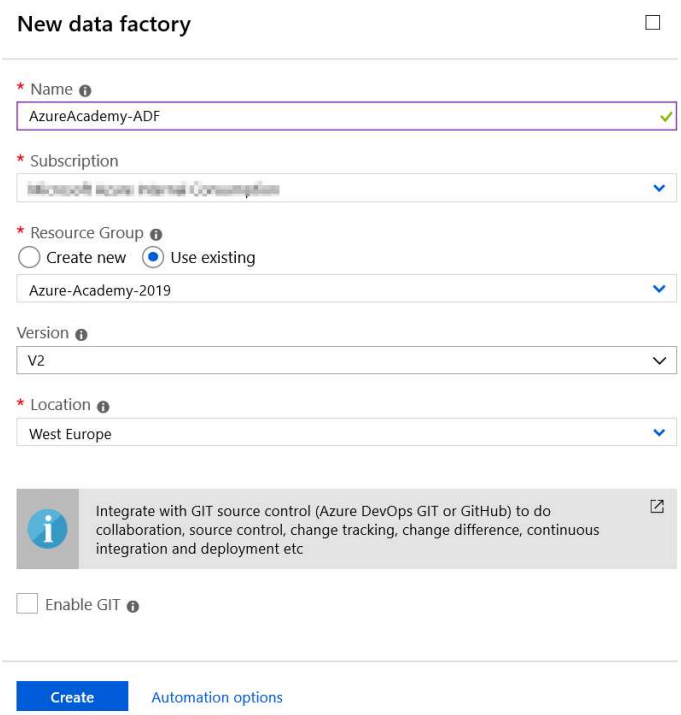
- Azure storage account: Azure Storage is used as the *staging* blob in the bulk copy operation. If you don't have an Azure storage account, see the instructions in Create a storage account.

Create a data factory

1. On the left menu, select **Create a resource** > **Data + Analytics** > **Data Factory**:

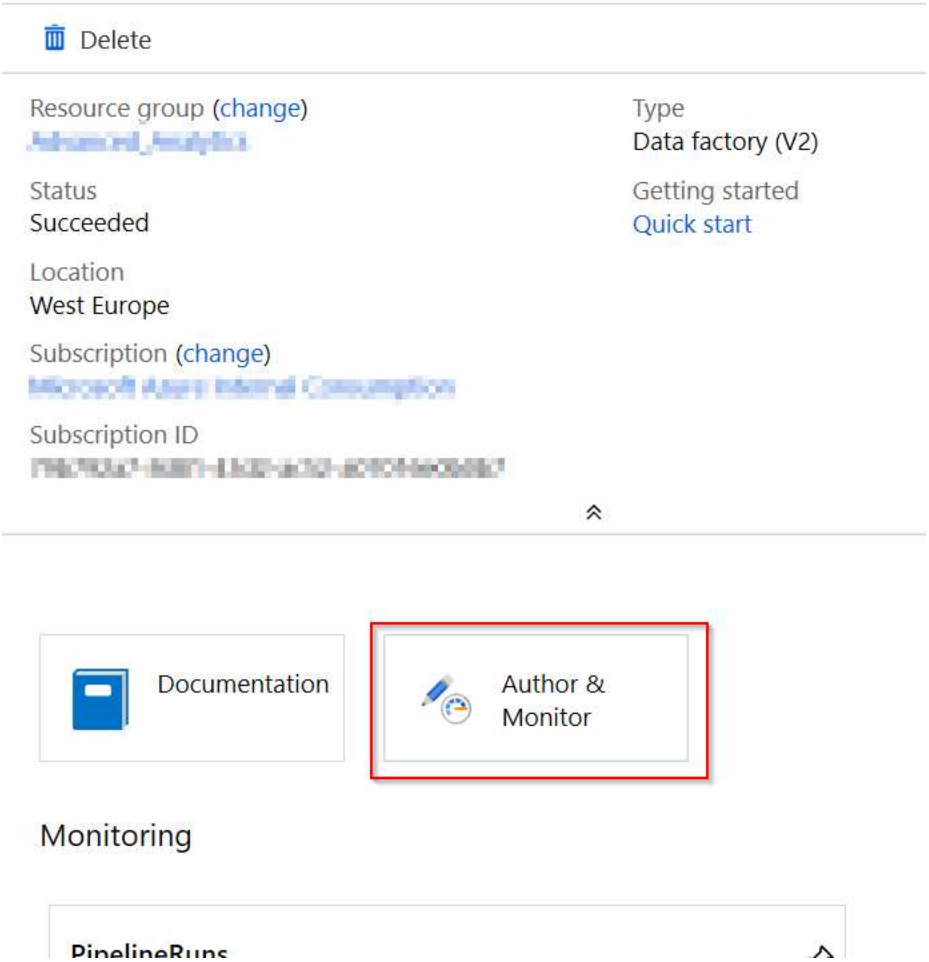


2. In the **New data factory** page, provide values for the fields that are shown in the following image:



- **Name**: Enter a globally unique name for your Azure data factory. If you receive the error "Data factory name "AzureAcademy-ADF" is not available," enter a different name for the data factory. For example, you could use the name ***yourname*ADFTutorialDataFactory**. Try creating the data factory again. For the naming rules for Data Factory artifacts, see Data Factory naming rules.
- **Subscription**: Select your Azure subscription in which to create the data factory.
- **Resource Group**: Select an existing resource group from the drop-down list, or select the **Create new** option and enter the name of a resource group. To learn about resource groups, see Using resource groups to manage your Azure resources.
- **Version**: Select **V2**.
- **Location**: Select the location for the data factory. Only supported locations are displayed in the drop-down list. The data stores that are used by data factory can be in other locations and regions. These data stores include Azure Data Lake Store, Azure Storage, Azure SQL Database, and so on.

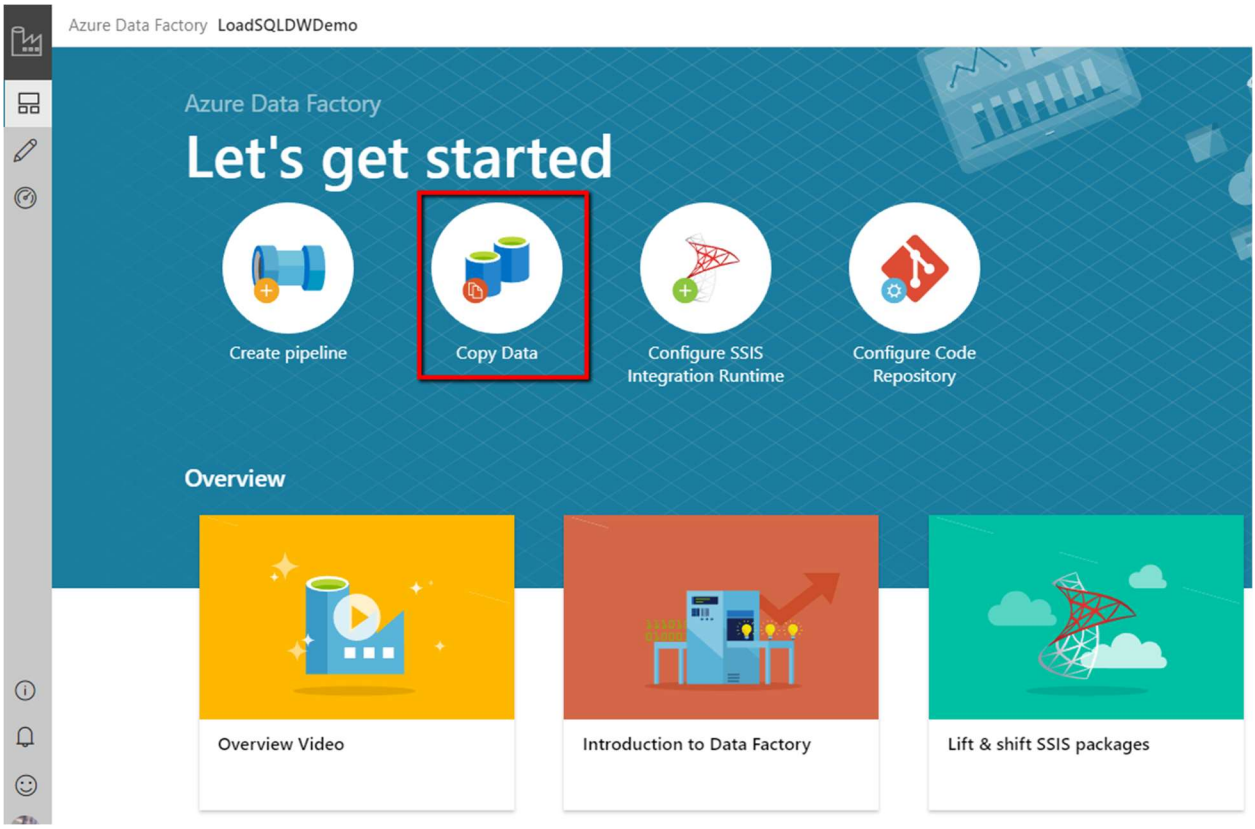
3. Select **Create**.
4. After creation is complete, go to your data factory. You see the **Data Factory** home page as shown in the following image:



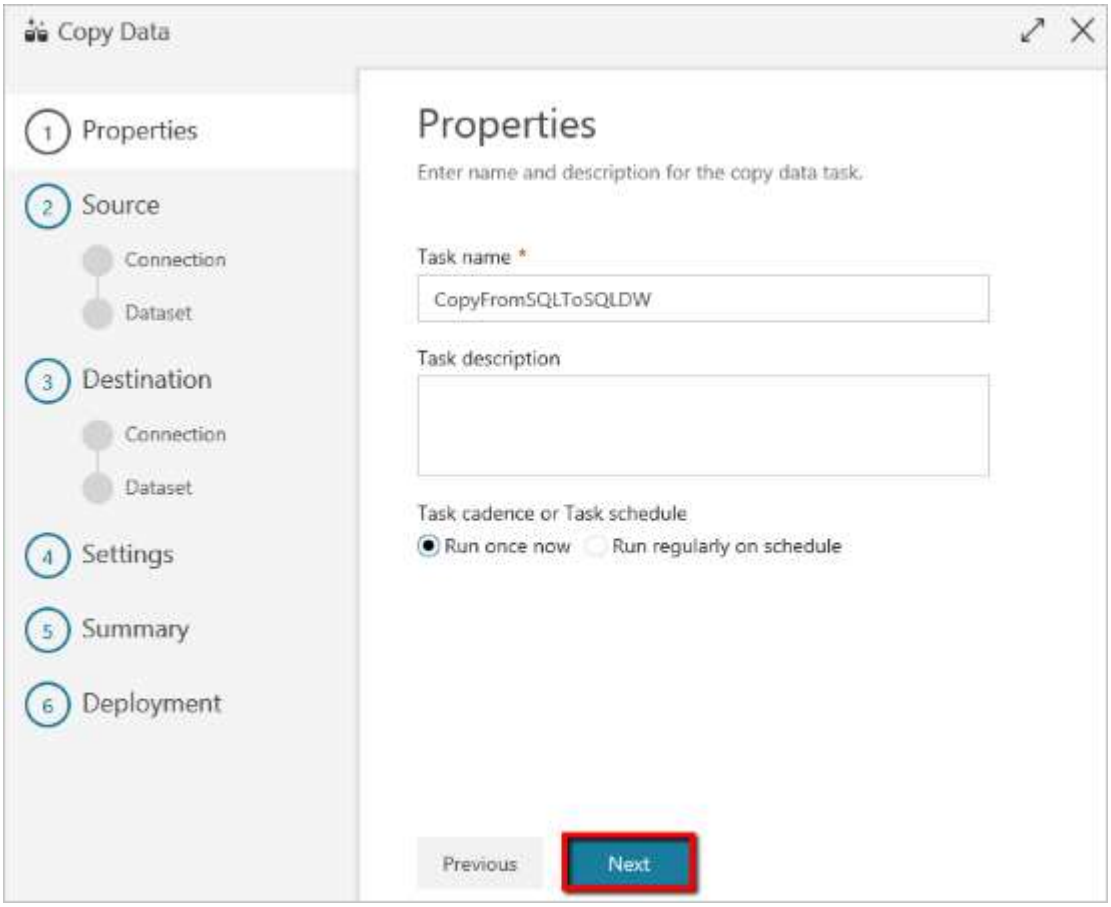
Select the **Author & Monitor** tile to launch the Data Integration Application in a separate tab.

Load data into Azure SQL Data Warehouse

1. In the **Get started** page, select the **Copy Data** tile to launch the Copy Data tool:

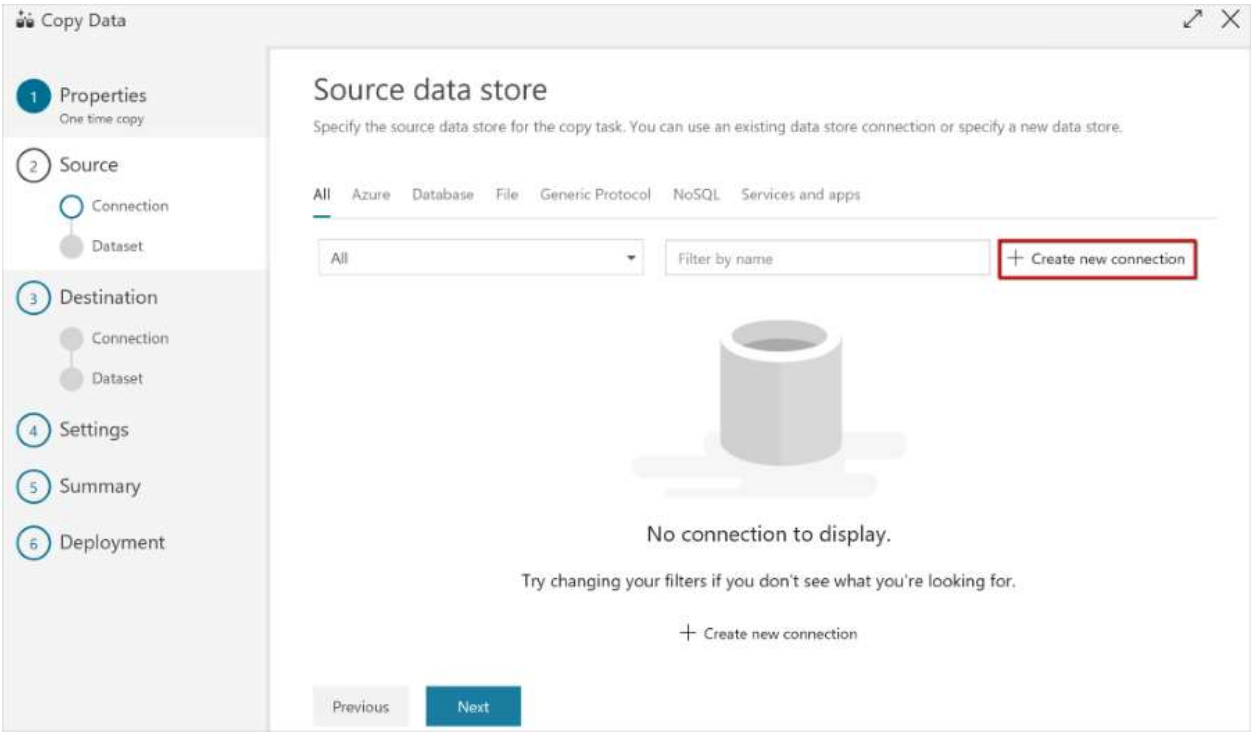


2. In the **Properties** page, specify **CopyFromSQLToSQLDW** for the **Task name** field, and select **Next**:

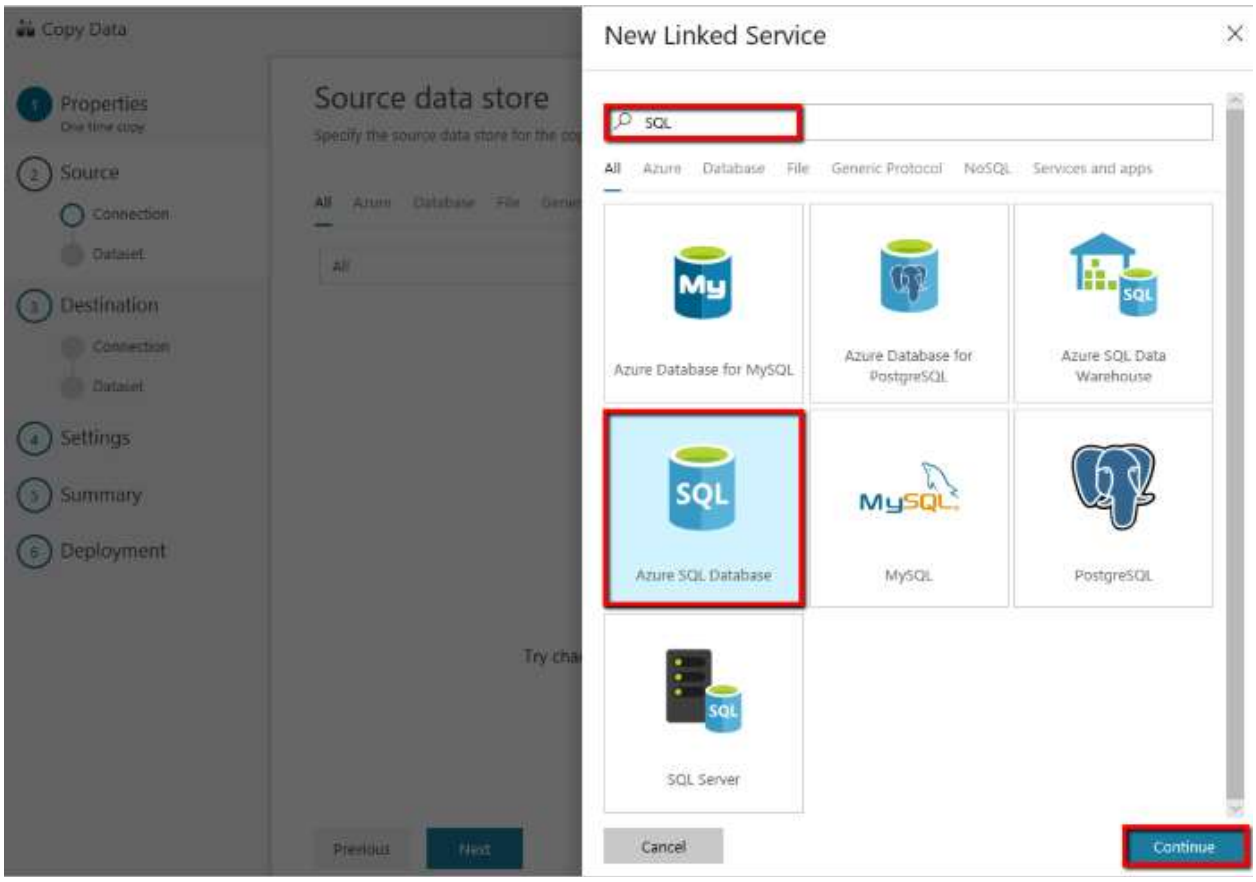


3. In the **Source data store** page, complete the following steps:

   a. click **+ Create new connection**:

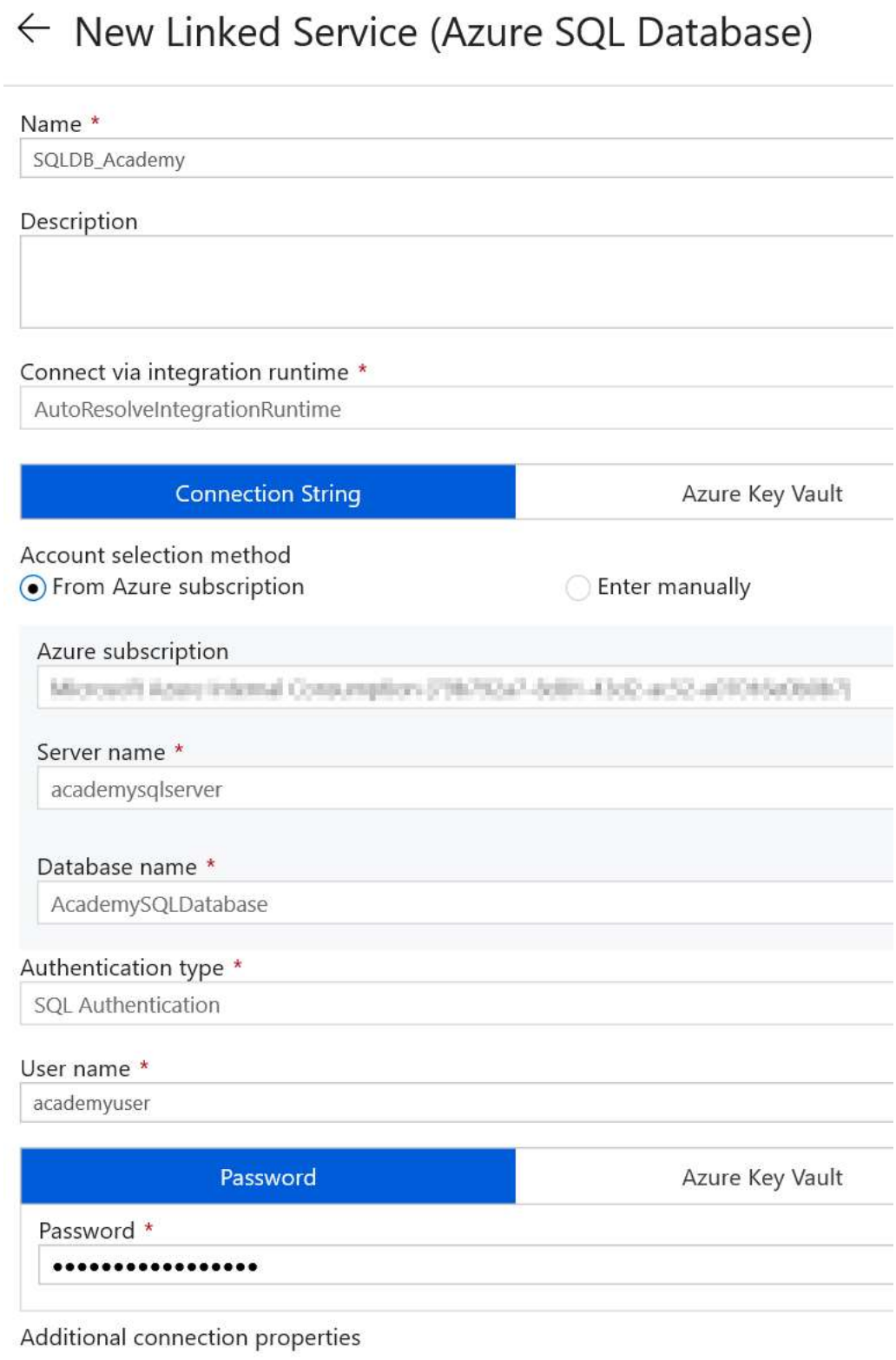


b. Select **Azure SQL Database** from the gallery and select **Continue**. You can type "SQL" in the search box to filter the connectors.



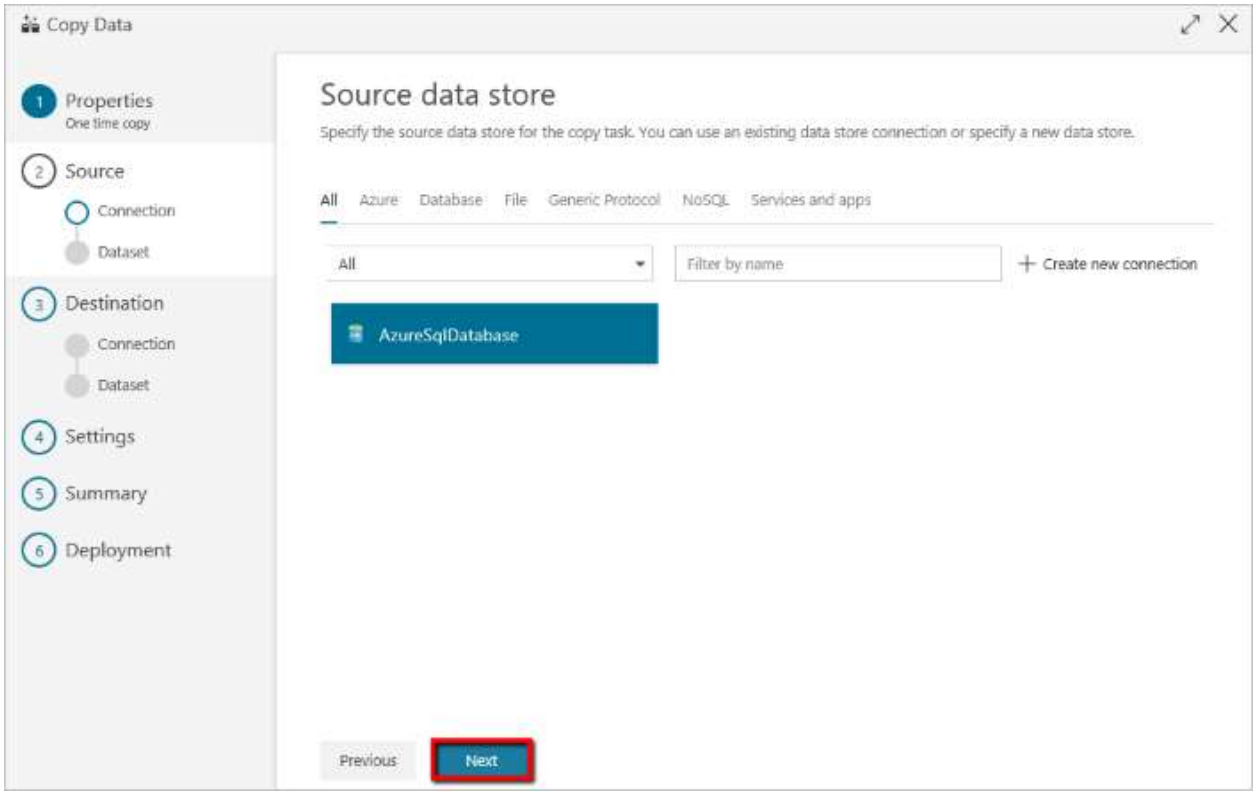
c. In the **New Linked Service** page, select your server name and DB name from the dropdown list, and specify the username and password. If you don't you're your own SQL Database, please connect to our with credentials mentioned in the beginning of this document.

Click **Test connection** to validate the settings, then select **Finish**.

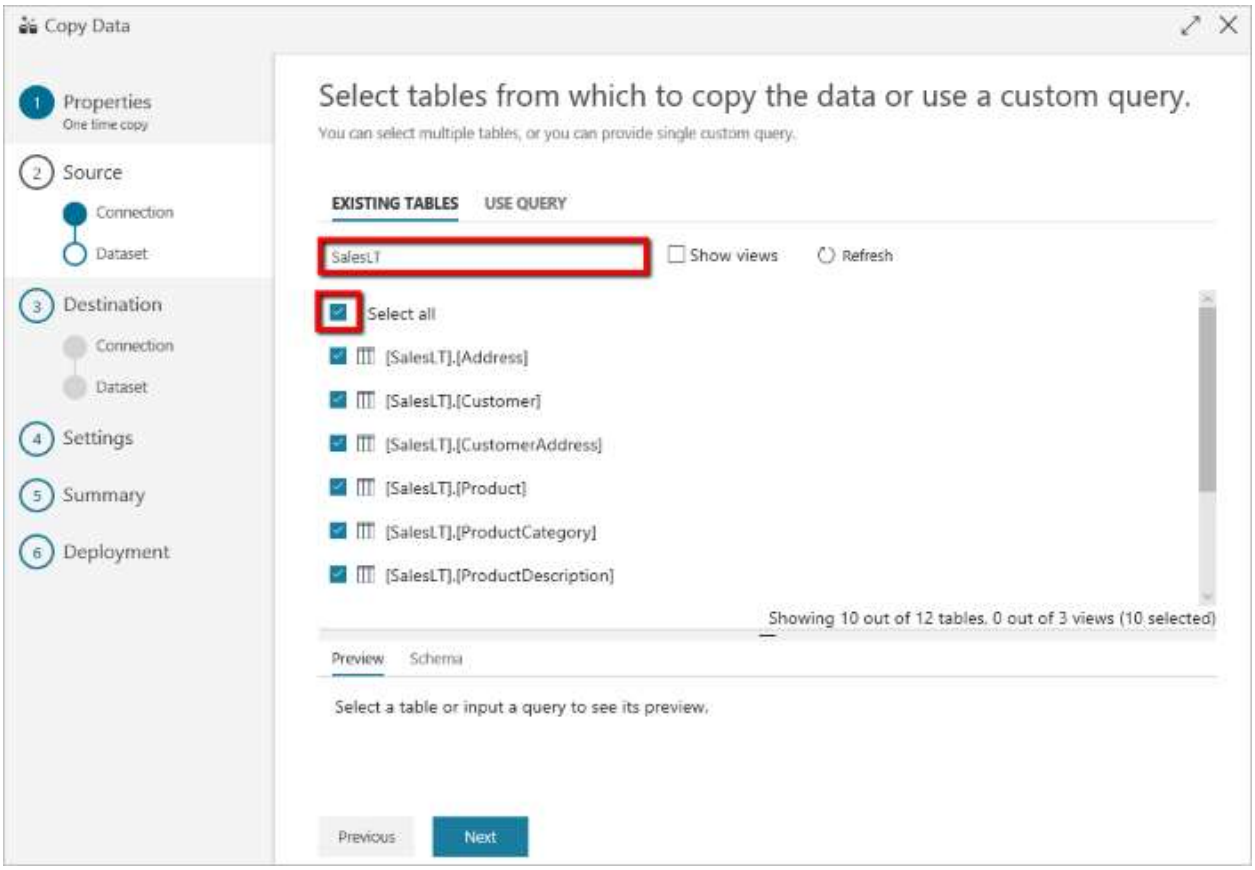


Your screen could look like this.

d. Select the newly created linked service as source, then click **Next**.

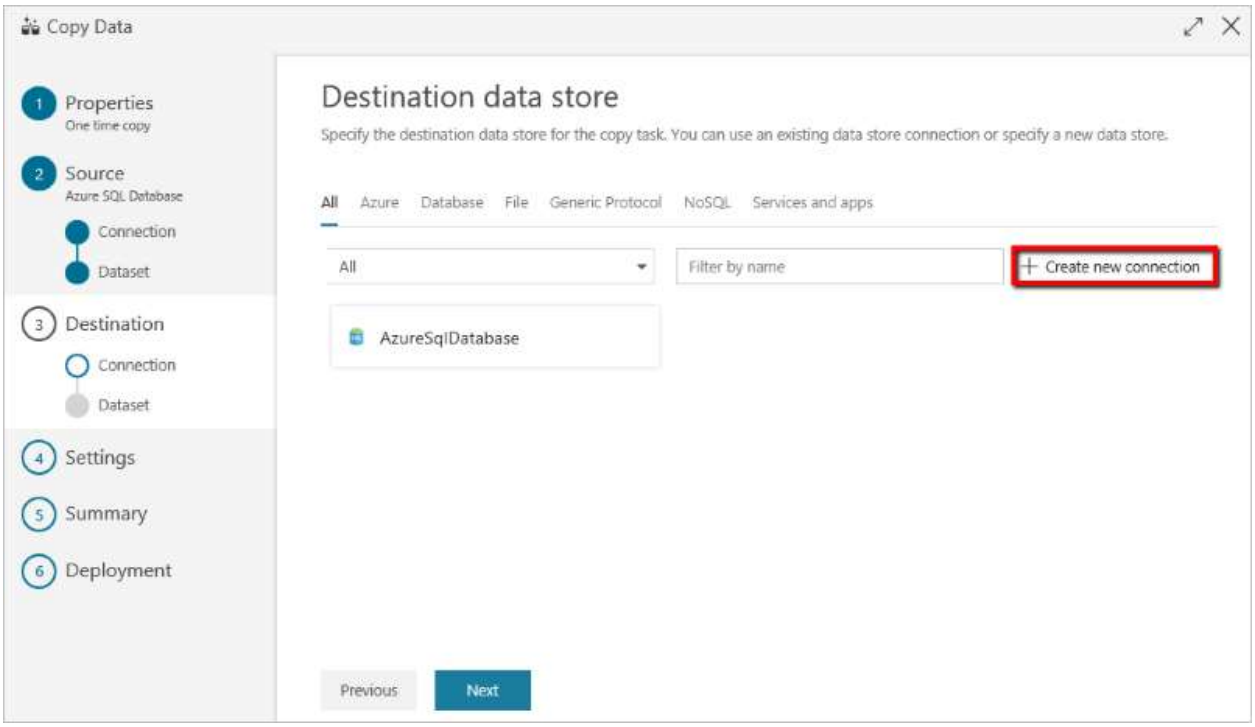


4. In the **Select tables from which to copy the data or use a custom query** page, enter **SalesLT** to filter the tables. Choose the **(Select all)** box to use all of the tables for the copy, and then select **Next**:

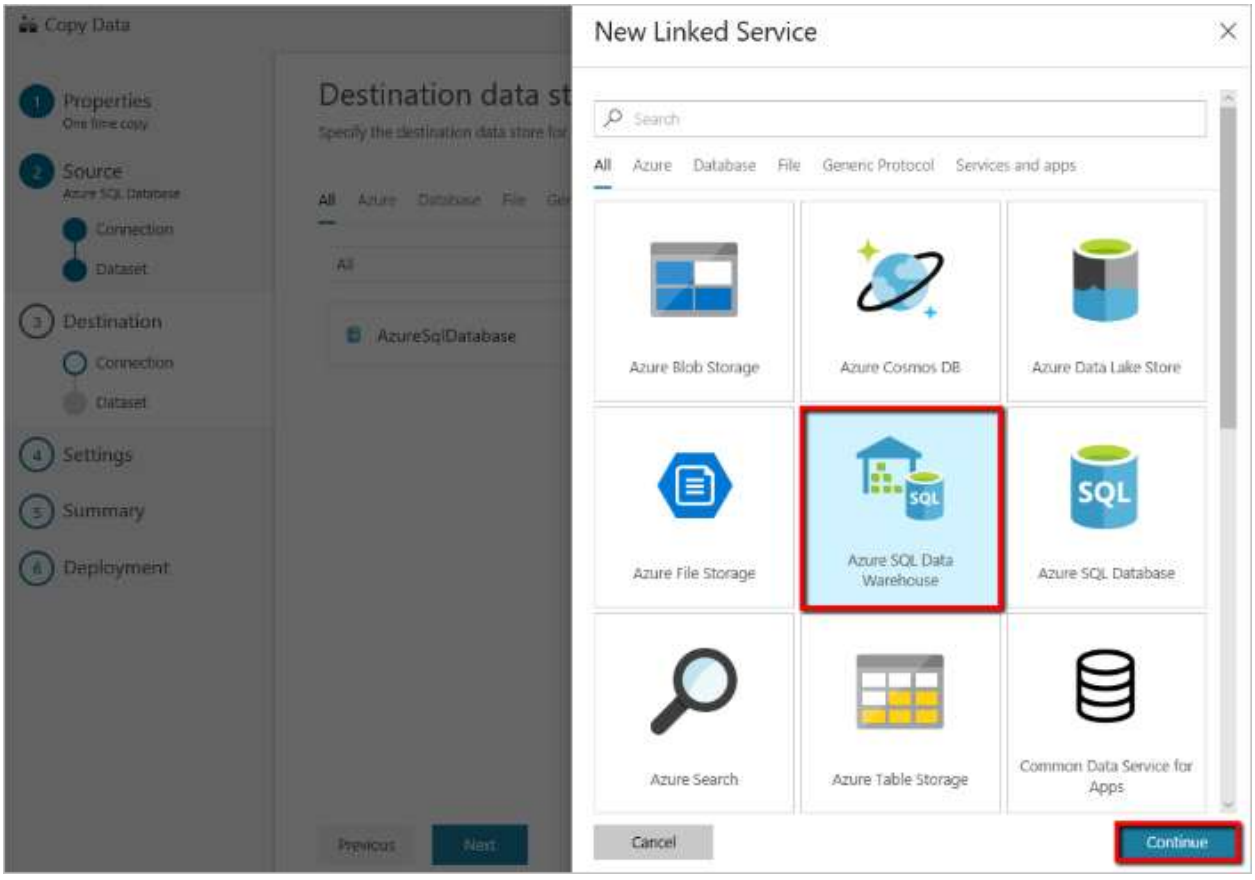


5. In the **Destination data store** page, complete the following steps:

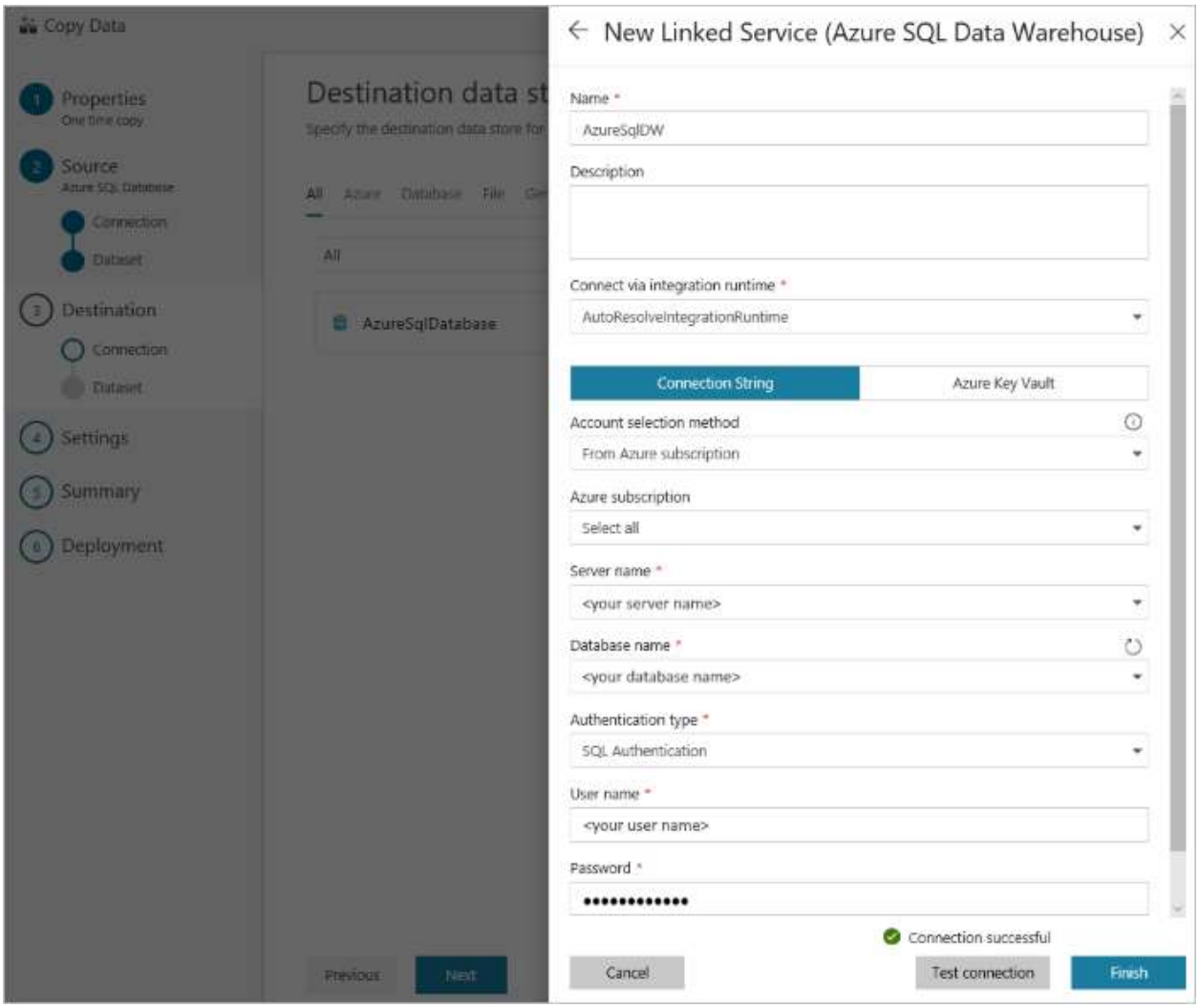
   a. Click **+ Create new connection** to add a connection



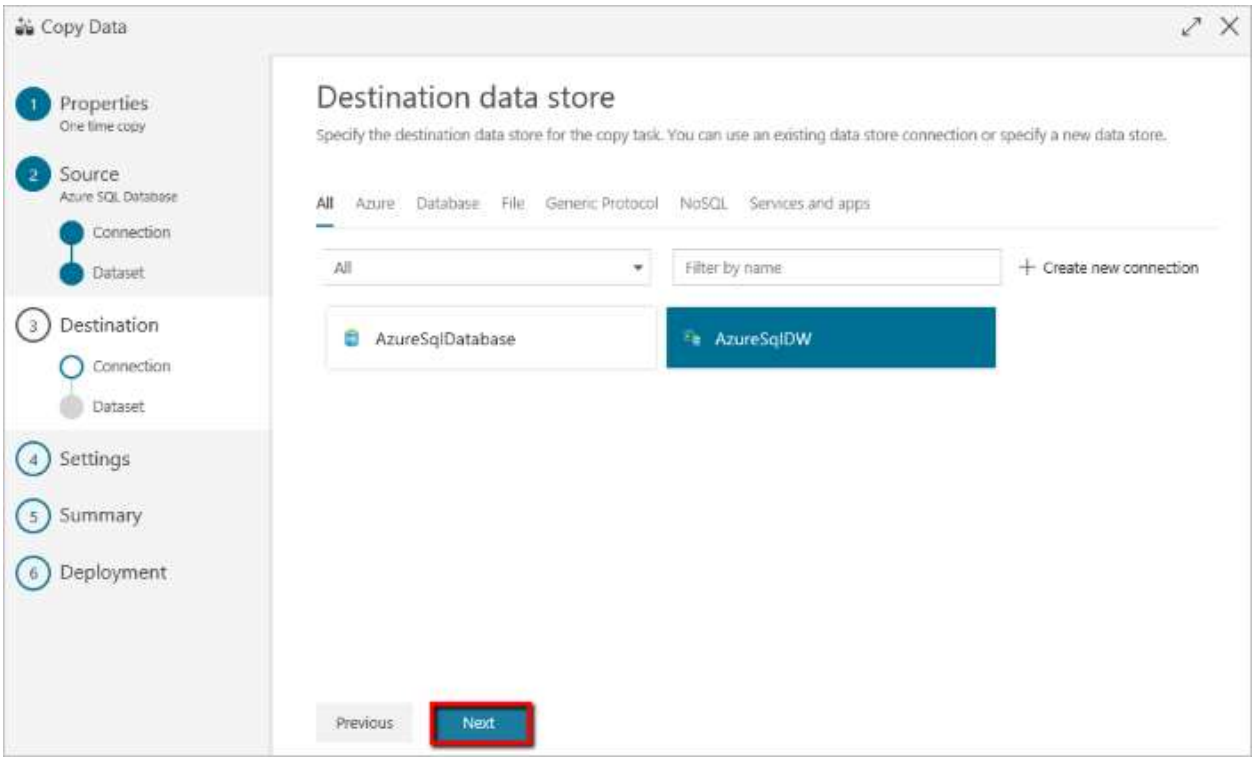
b. Select **Azure SQL Data Warehouse** from the gallery, and select **Next**.



c. In the **New Linked Service** page, select your server name and DB name from the dropdown list, and specify the username and password. Click **Test connection** to validate the settings, then select **Finish**.



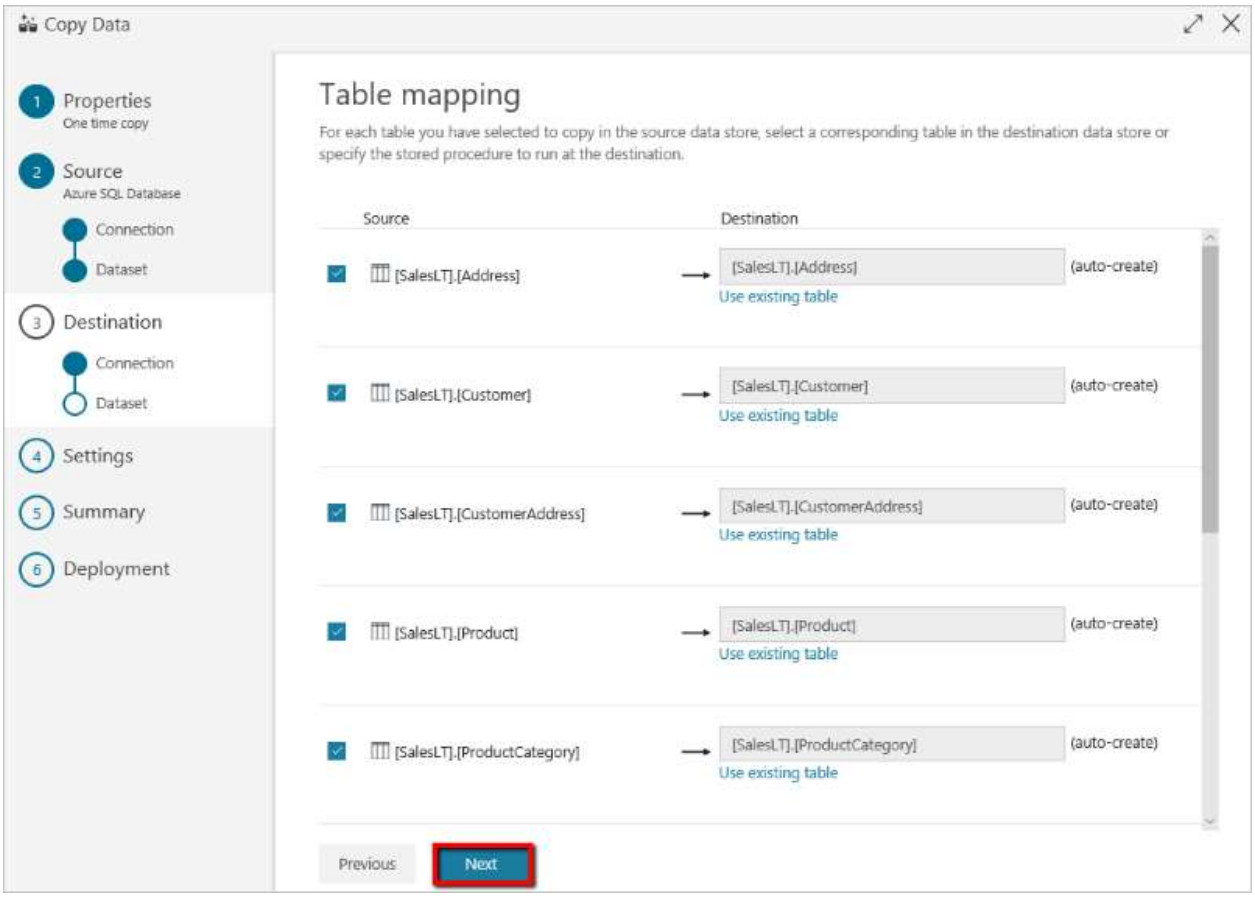
d. Select the newly created linked service as sink, then click **Next**.



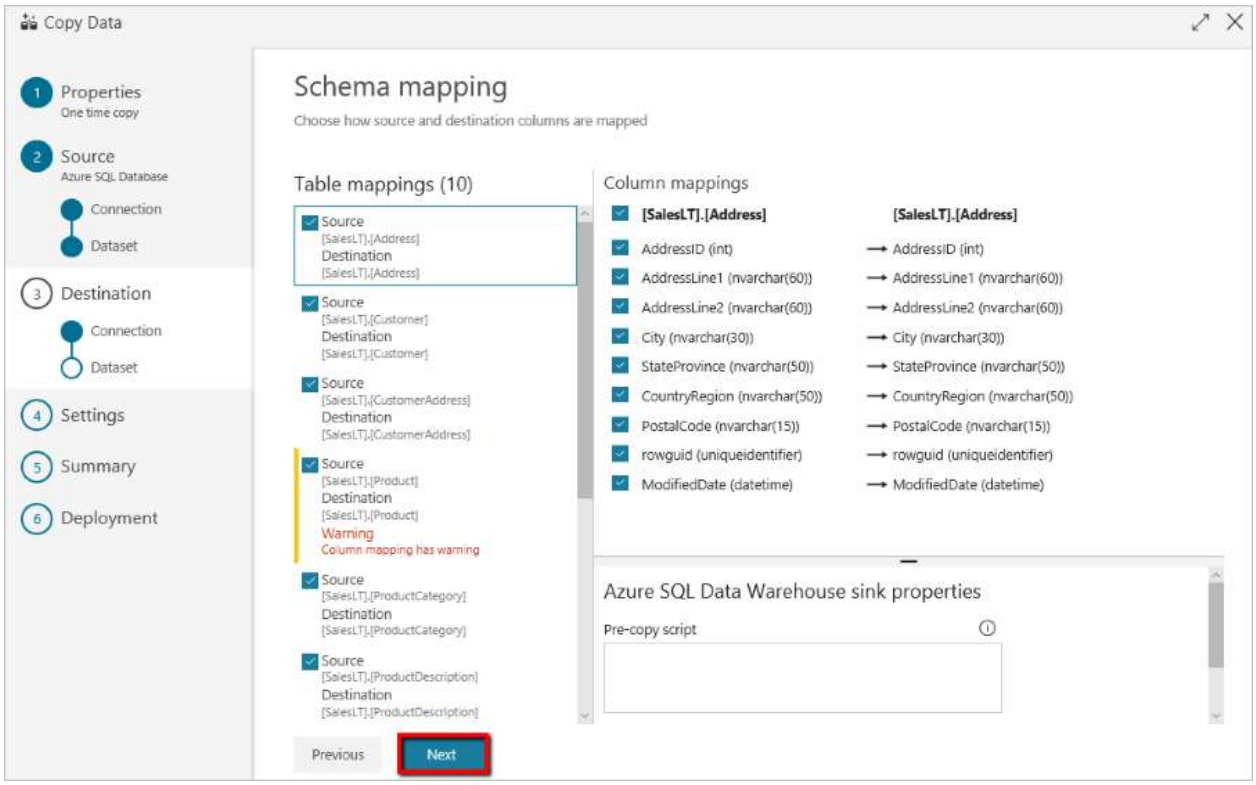
6. In the **Table mapping** page, review the content, and select **Next**. An intelligent table mapping displays. The source tables are mapped to the destination tables based on the table names. If a source table doesn't exist in the destination, Azure Data Factory creates a destination table with the same name by default. You can also map a source table to an existing destination table.

   Note

   Automatic table creation for the SQL Data Warehouse sink applies when SQL Server or Azure SQL Database is the source. If you copy data from another source data store, you need to pre-create the schema in the sink Azure SQL Data Warehouse before executing the data copy.

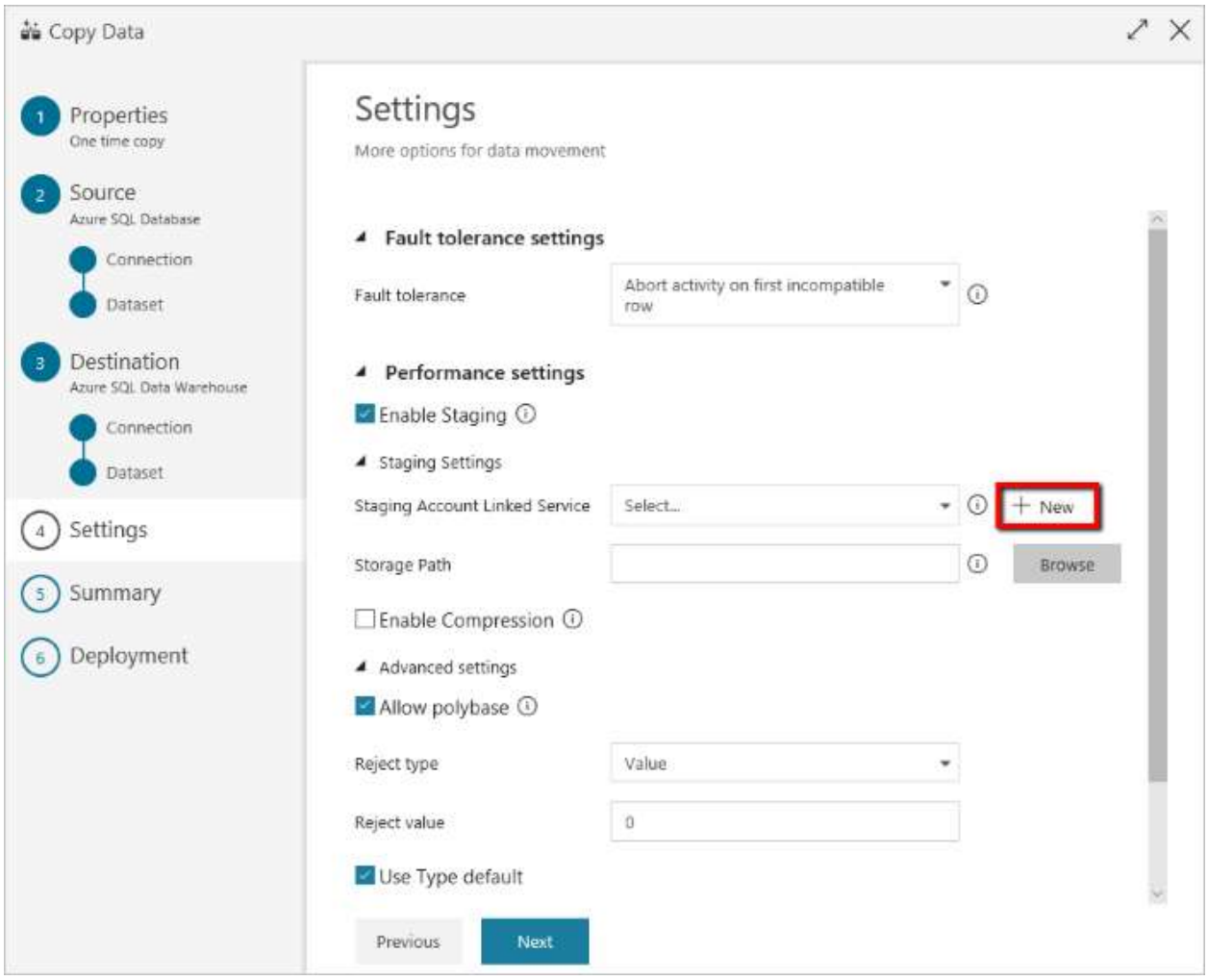


7. In the **Schema mapping** page, review the content, and select **Next**. The intelligent table mapping is based on the column name. If you let Data Factory automatically create the tables, data type conversion can occur when there are incompatibilities between the source and destination stores. If there's an unsupported data type conversion between the source and destination column, you see an error message next to the corresponding table.

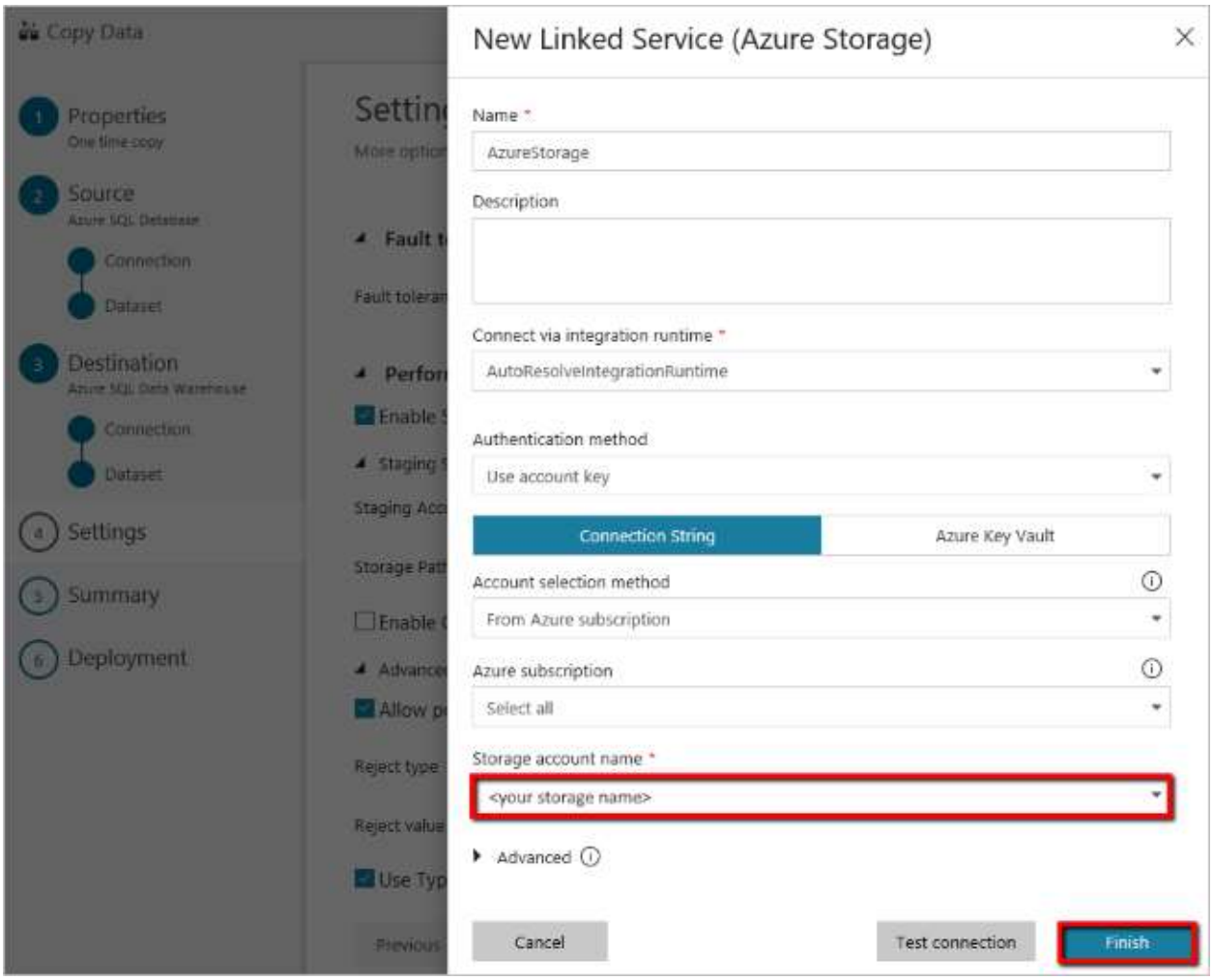


8. In the **Settings** page, complete the following steps:

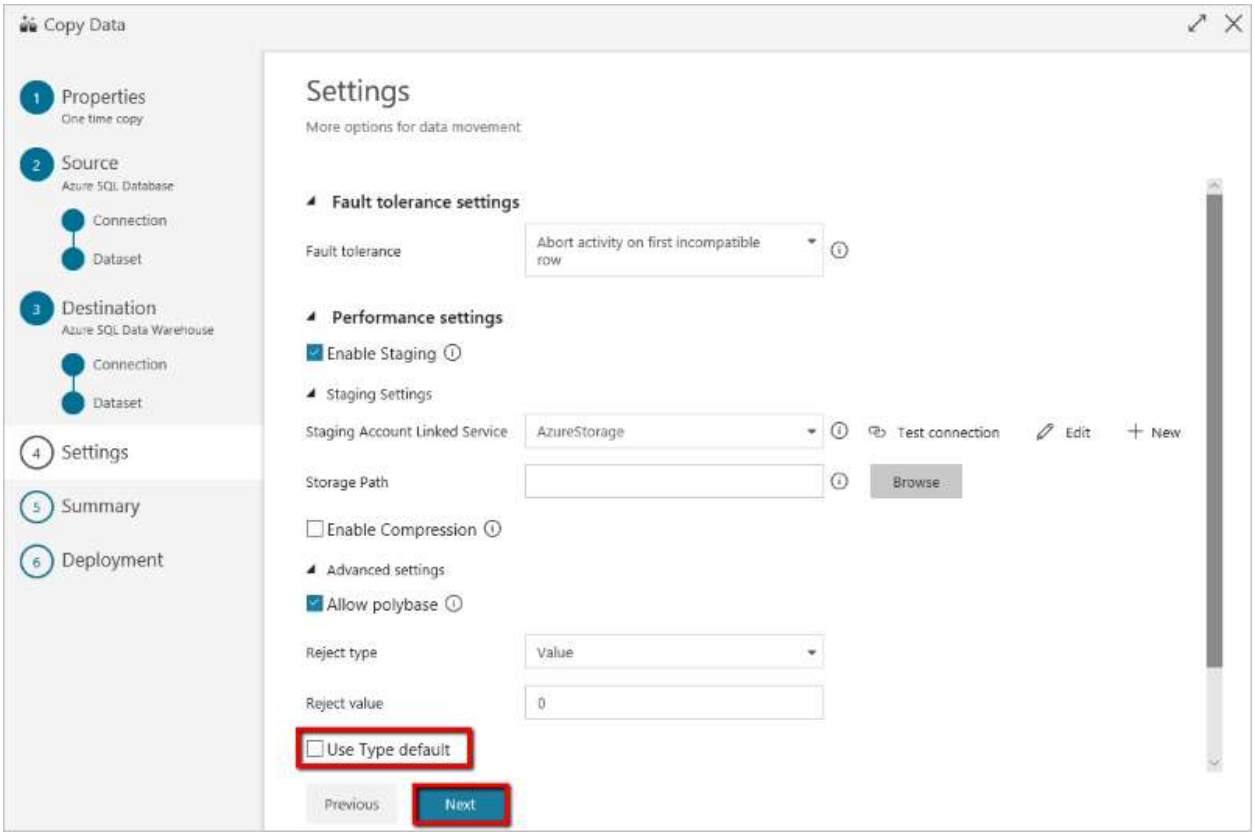
   a. In **Staging settings** section, click **+ New** to new a staging storage. The storage is used for staging the data before it loads into SQL Data Warehouse by using PolyBase. After the copy is complete, the interim data in Azure Storage is automatically cleaned up.



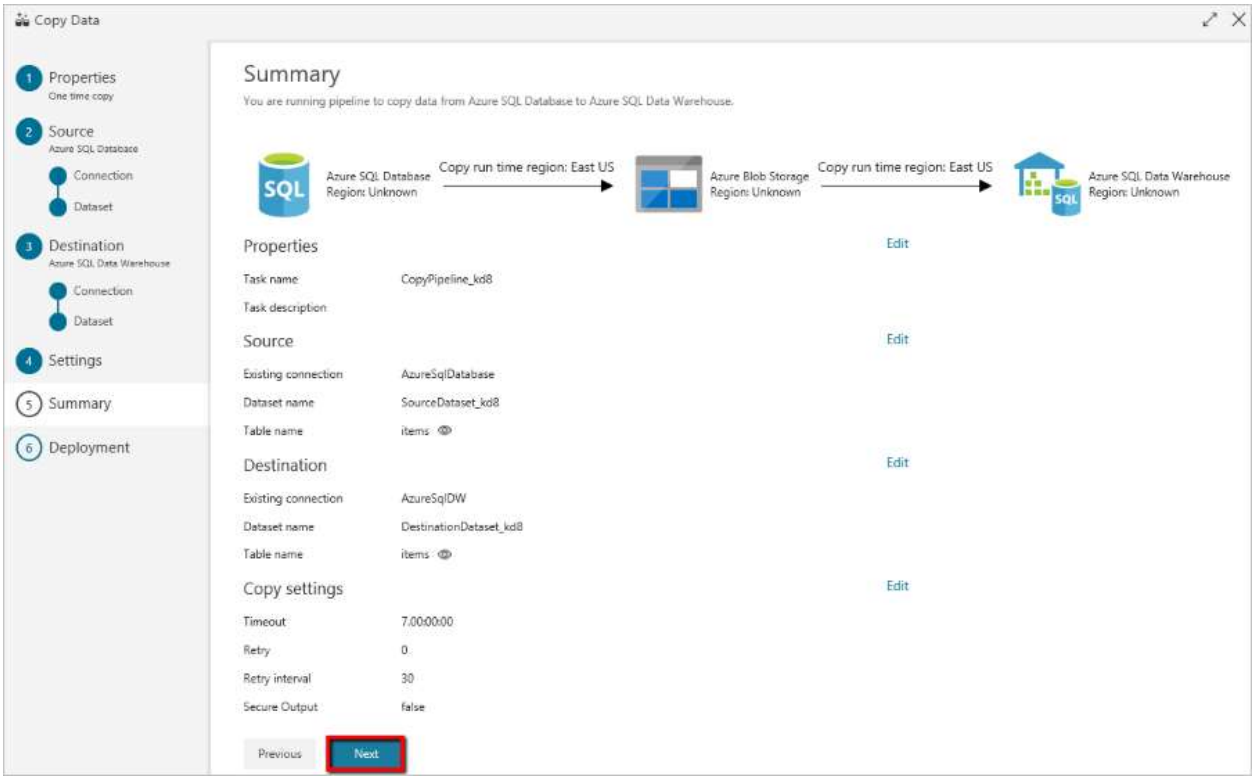
b. In the **New Linked Service** page, select your storage account, and select **Finish**.



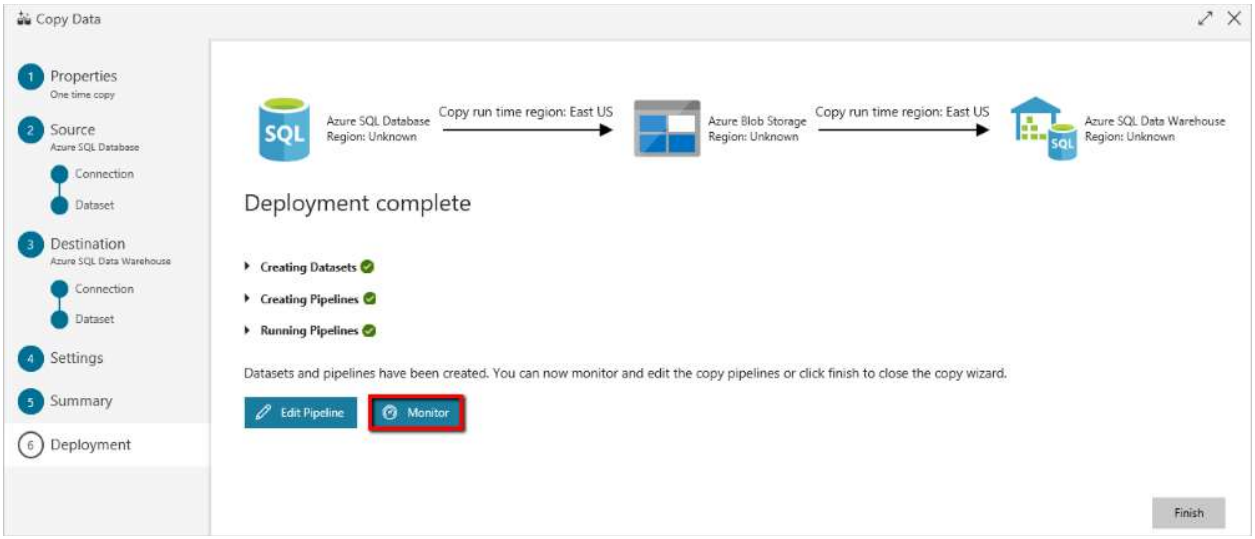
c. In the **Advanced settings** section, deselect the **Use type default** option, then select **Next**.



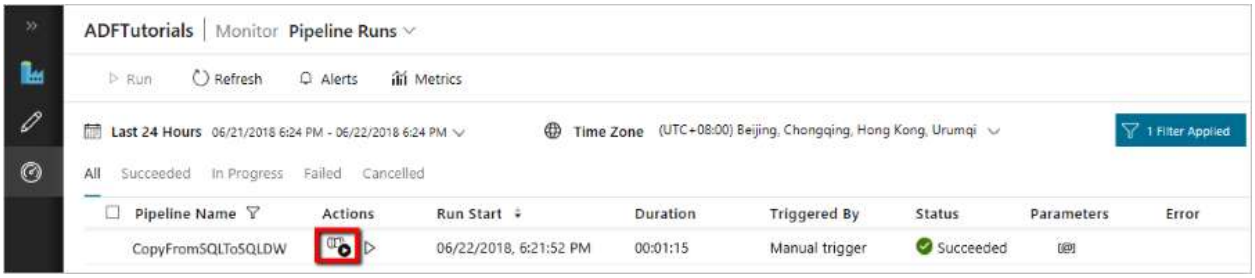
9. In the **Summary** page, review the settings, and select **Next**:



10. In the **Deployment page**, select **Monitor** to monitor the pipeline (task):

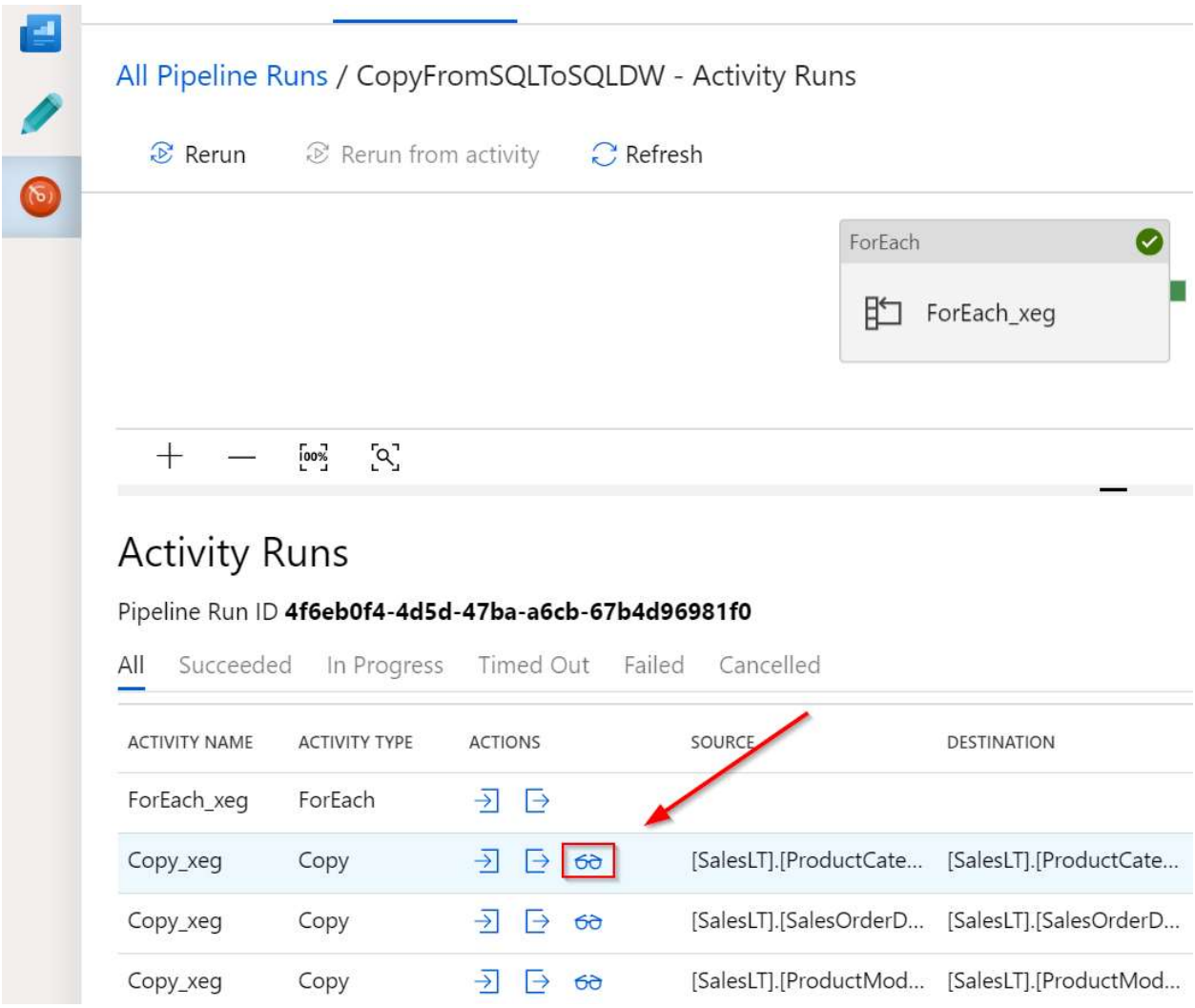


11. Notice that the **Monitor** tab on the left is automatically selected.
    The **Actions** column includes links to view activity run details and to rerun the pipeline:

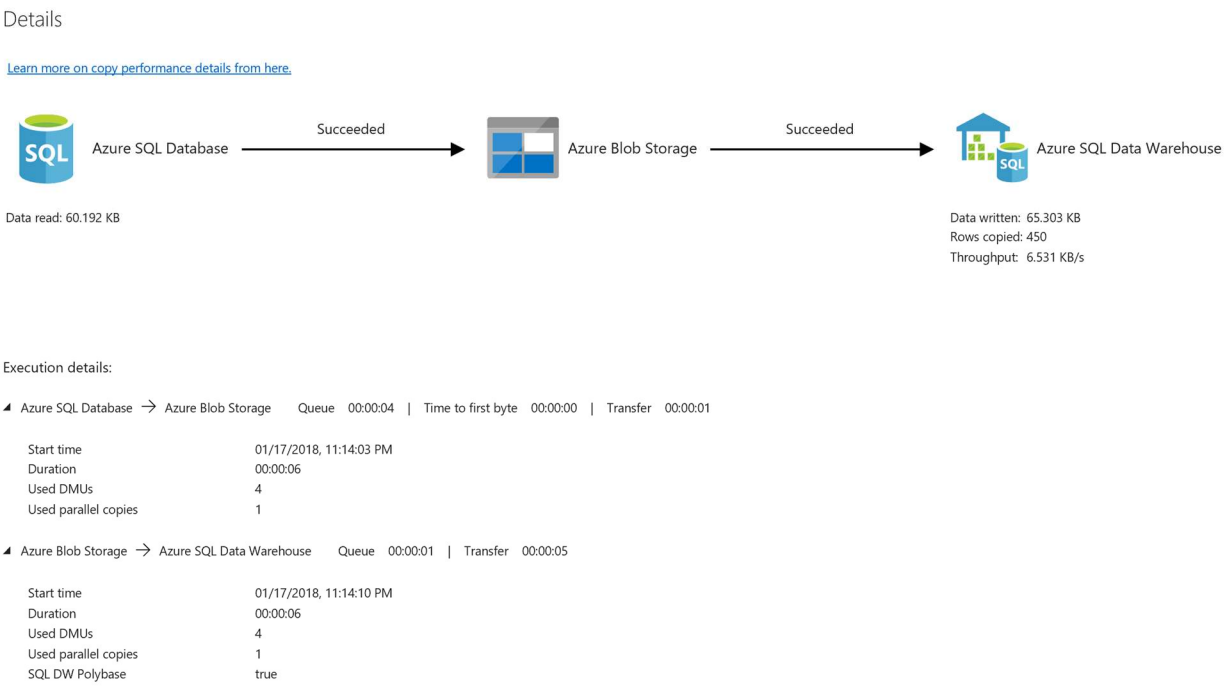


12. To view activity runs that are associated with the pipeline run, select the **View Activity Runs** link in the **Actions** column. To switch back to the pipeline runs view, select the **Pipelines** link at the top. Select **Refresh** to refresh the list.

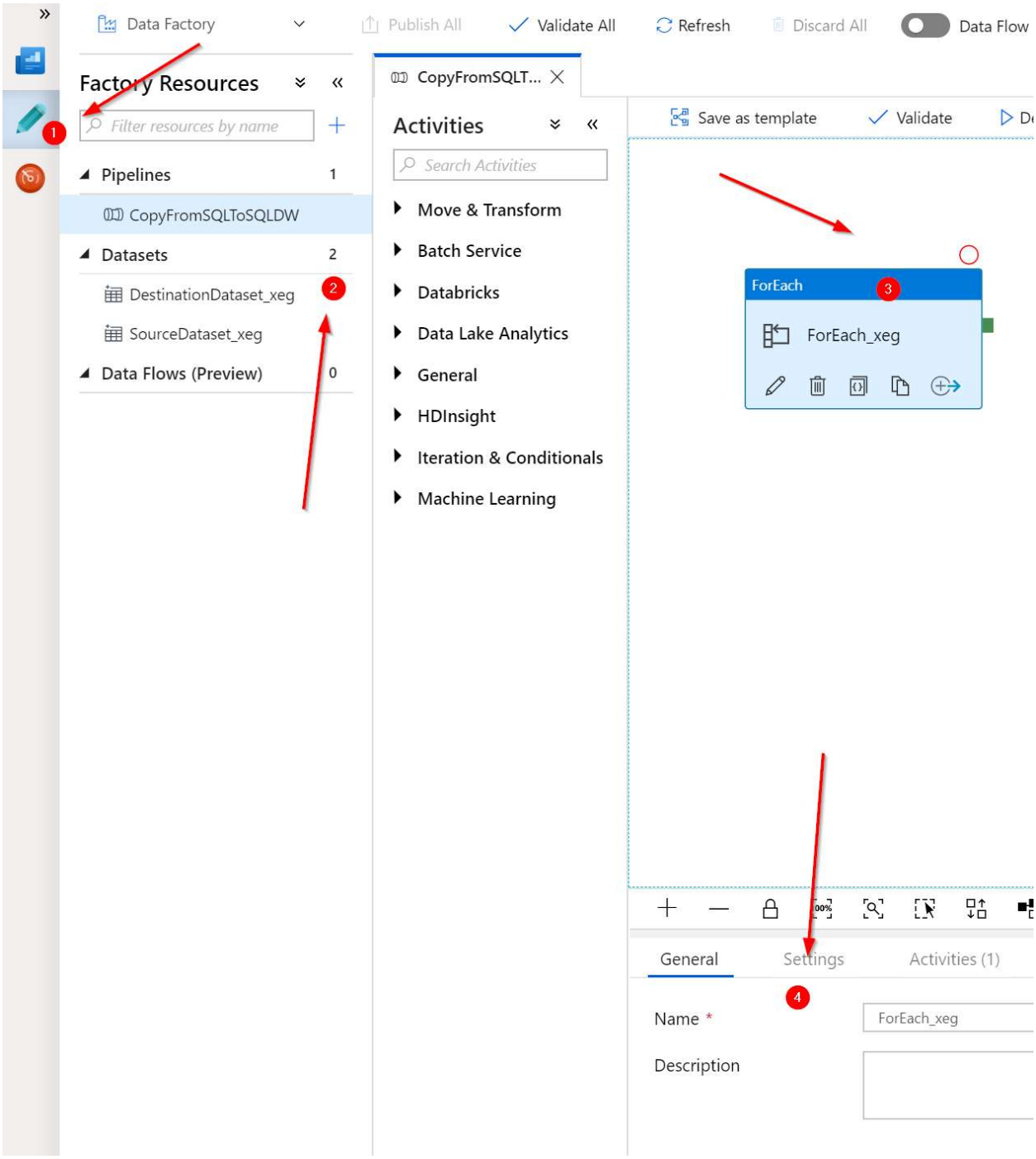
13. To monitor the execution details for each copy activity, select the **Details** link under **Actions** in the activity monitoring view. You can monitor details like the volume of data copied from the source to the sink, data throughput, execution steps with corresponding duration, and used configurations:



You should see the following details.



Review what has been created by the wizard in Azure Data Factory.


Data Factory
Publish All
Validate All
Refresh
Discard All
Data Flow
Factory Resources
Filter resources by name
Pipelines 1
CopyFromSQLToSQLDW
Datasets 2
DestinationDataset_xeg
SourceDataset_xeg
Data Flows (Preview) 0
CopyFromSQLT...
Activities
Search Activities
Move & Transform
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & Conditionals
Machine Learning
Save as template
Validate
ForEach
ForEach_xeg
General
Settings
Activities (1)
Name *
ForEach_xeg
Description
1
2
3
4

Create ADF pipeline using Mapping Data Flows

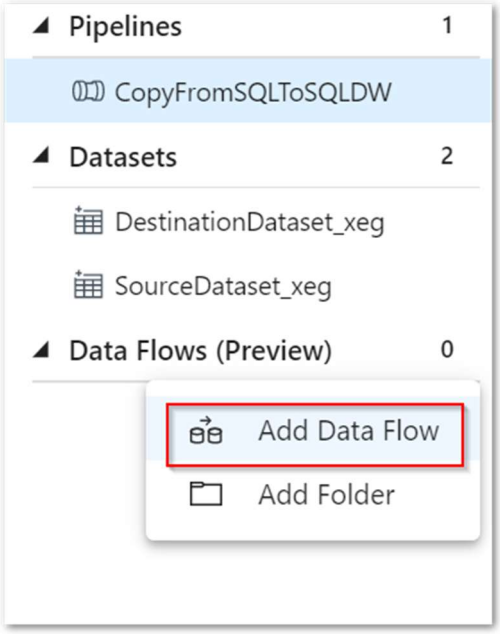
ADF Mapping data flows is a data transformation part of ADF which helps to visually define the data transformation using the Azure Databricks technology in the background.

These defined data flows could be then used in Azure Data Factory pipeline as any other object available in ADF.

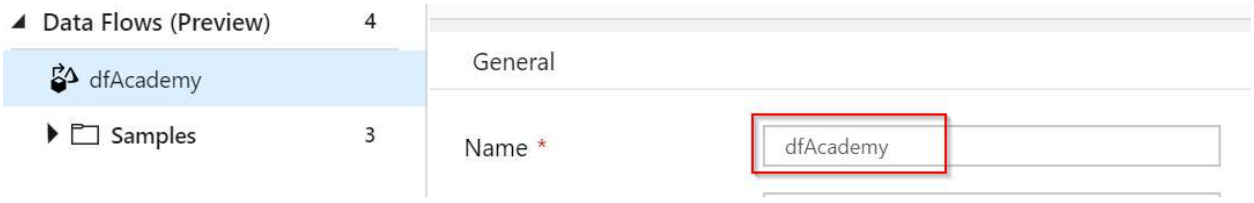
Azure Databricks cluster could be pre-provisioned in your environment or is created upon request in the background by ADF and destroyed after the pipeline finishes.

Create Mapping Data Flow

In the left panel, create new blank Data Flow



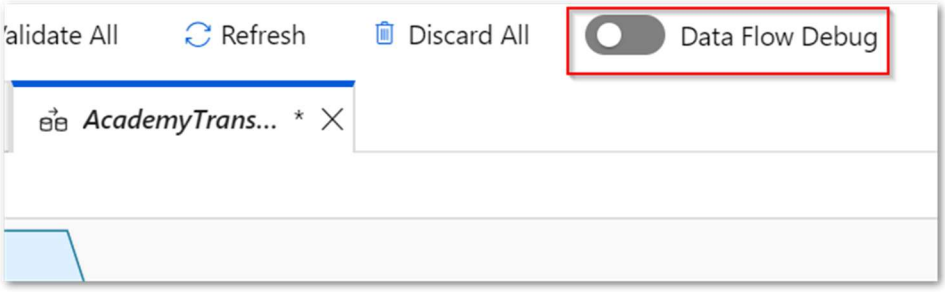
Give your new Data Flow the proper name (for example **dfAcademy**).



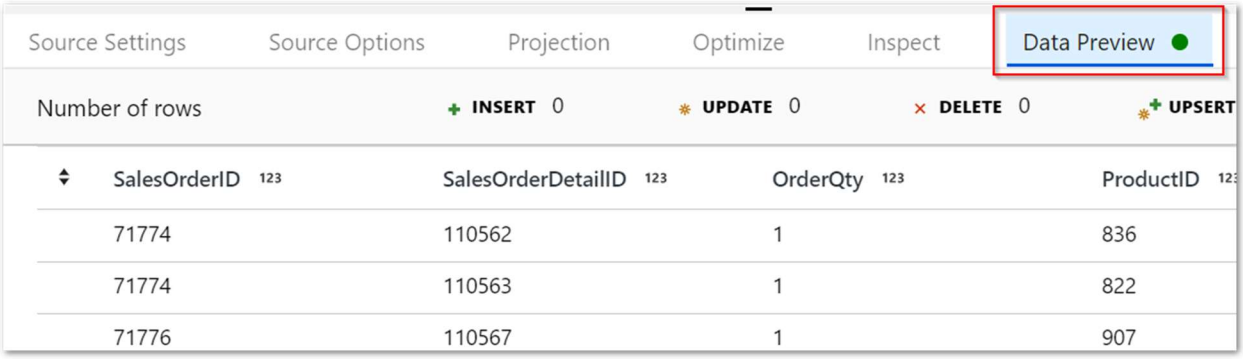
For the use of this demo, we will get some of the tables from the source SQL Database and denormalize them into one table suitable for (say) reporting.

We will create one table with order details along with data about customer and its address.

Click on the **Data Flow Debug** switch to enable seeing your data live without actually invasively touching it. This takes a few minutes as Data Factory runs small cluster under the covers and needs to start it.



After the switch is activated, you can preview all your changes and data on the **Data Preview** tab located on the right side of properties window.

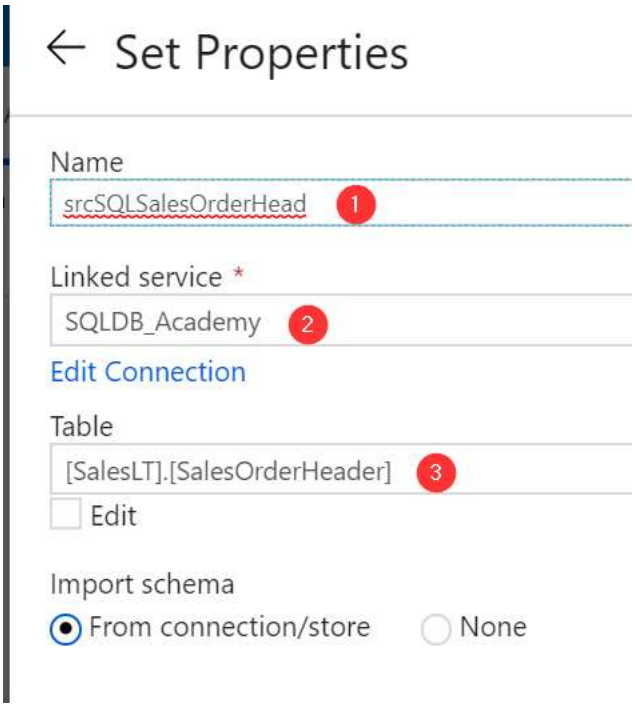


Create source datasets

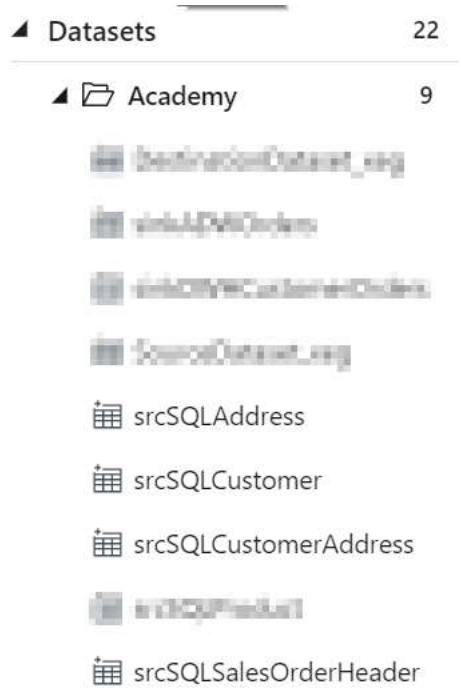
First, create the following source datasets representing the source tables from SQL database: *SalesOrderHeader, Customer, CustomerAddress, Address*. These are obviously fact table and 3 dimensions.

- Click on Add source and in the Properties window give your source dataset a name (**srcSQLSalesOrderHeader**)
- After you give it a name, click on the **+New** button to add connection to the actual table

- Choose Azure SQL Database and fill in the appropriate values on the Properties pop-up window

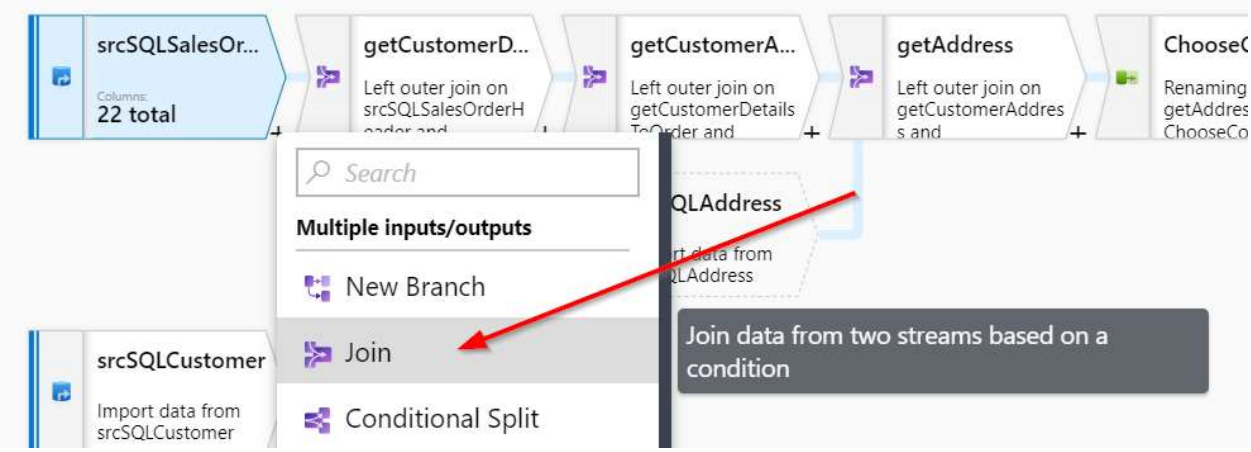


Iterate this process for all the tables – your datasets list and canvas should now look similar to this:

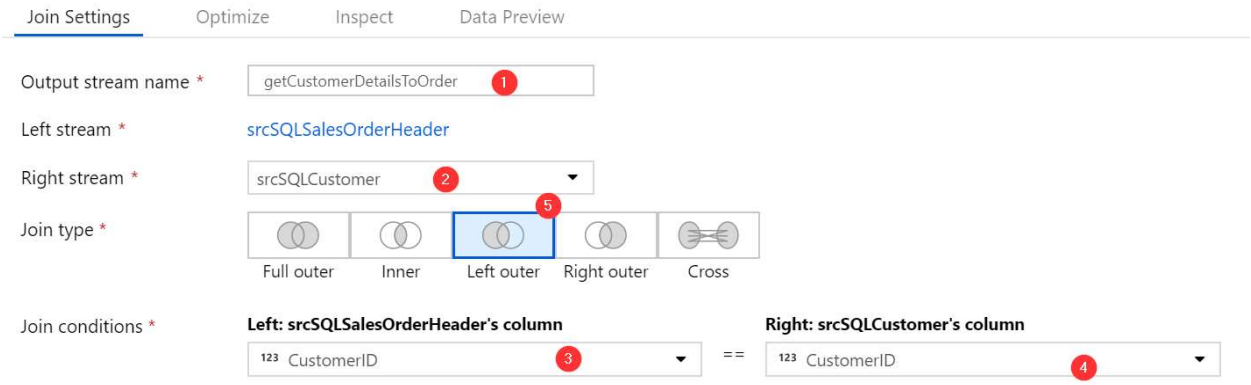


Join tables

From the first data source, click on the **+** sign, review possible activities and choose **Join** as the desired activity:



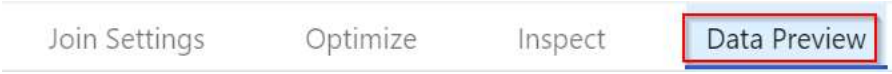
- Give your join activity proper name (**getCustomerDetailsToOrder**)
- Define the right stream – with what the join should be executed (**srcSQLCustomer**)
- On the left stream choose **CustomerID** from orders table and on the right side the **CustomerID** from the Customer table
- Regarding the join type, choose **Left outer** join in order to bring customers to their orders.



Join additional tables

Follow the same procedure as above but create joins for **CustomerAddress** and **Address** tables in addition.

You can always check our data within the flow on the **Data Preview** tab.



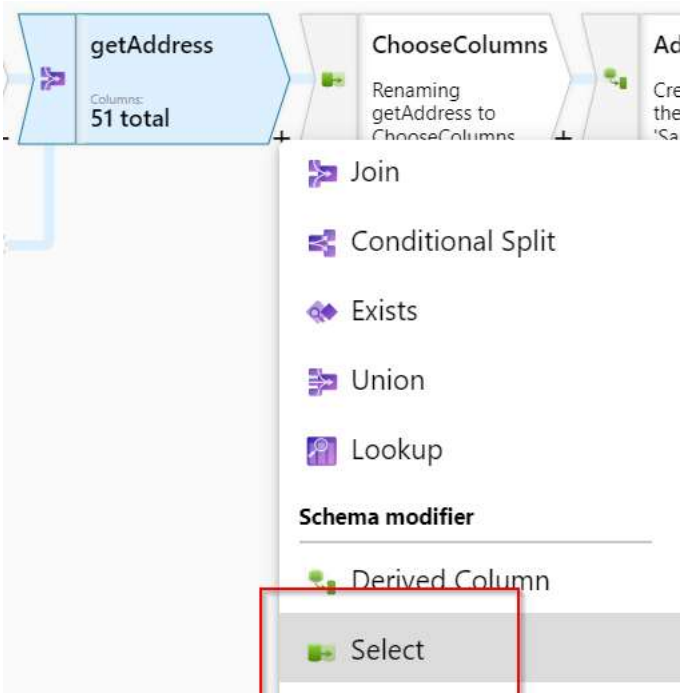
When hovering over the data flow element you see how many columns are currently there in the outgoing data set.

Your canvas should currently look something similar to this:

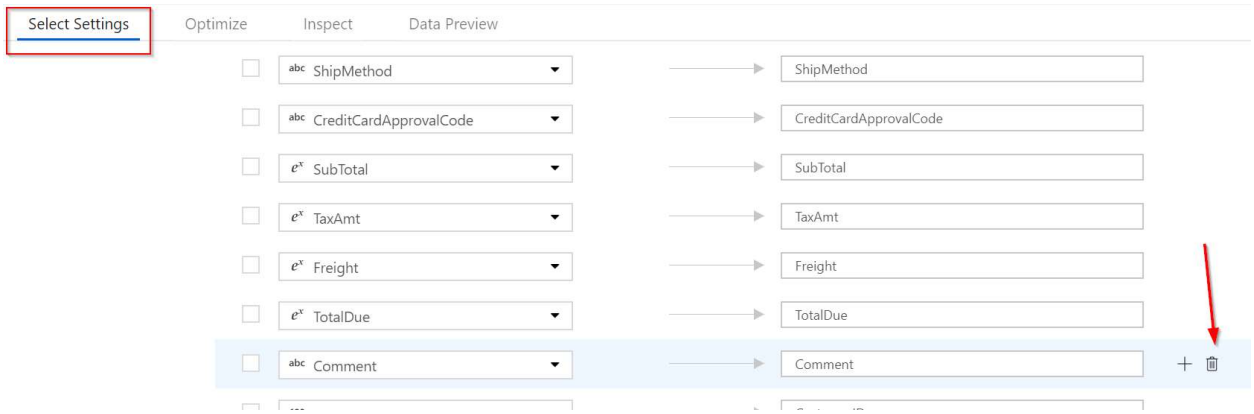
Limit columns in output

As you can imagine it is quite rough to have all the columns within the output table after joining fact table with 3 dimensions.

After the last flow, click on the + sign and choose **Select** activity.



There are some redundant columns with the same names from within multiple tables, please clean your data choosing and deleting columns you do not want in your output.



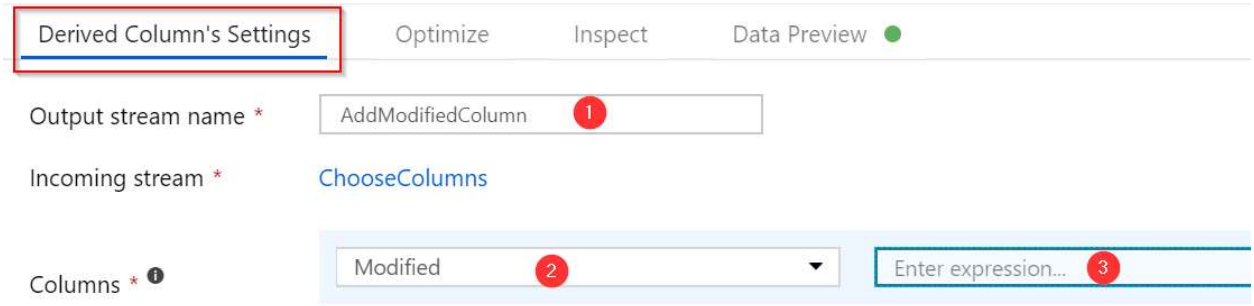
You can also change the name of the output columns if needed.

Add timestamp columns to control loading

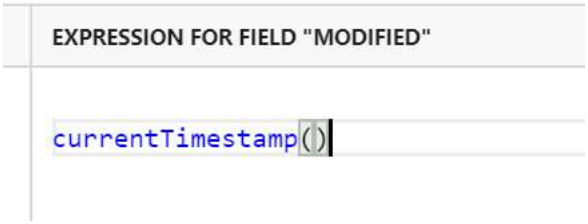
In this simple demo, we would like to show how to create custom column in our data – in this case we will create the timestamp column to see when the load into the Warehouse happened.

- After the last flow, click on the **+** sign and choose **Derived Column** as the activity

- In the properties window, give your flow proper name, define the name of your new column and click on the Expression text box to define, how your new column should look like



- Review the **Visual Expression Builder** – test, what functions are available and how the editor behaves
- For the contents of this column, choose **currentTimestamp()** function and Save the window



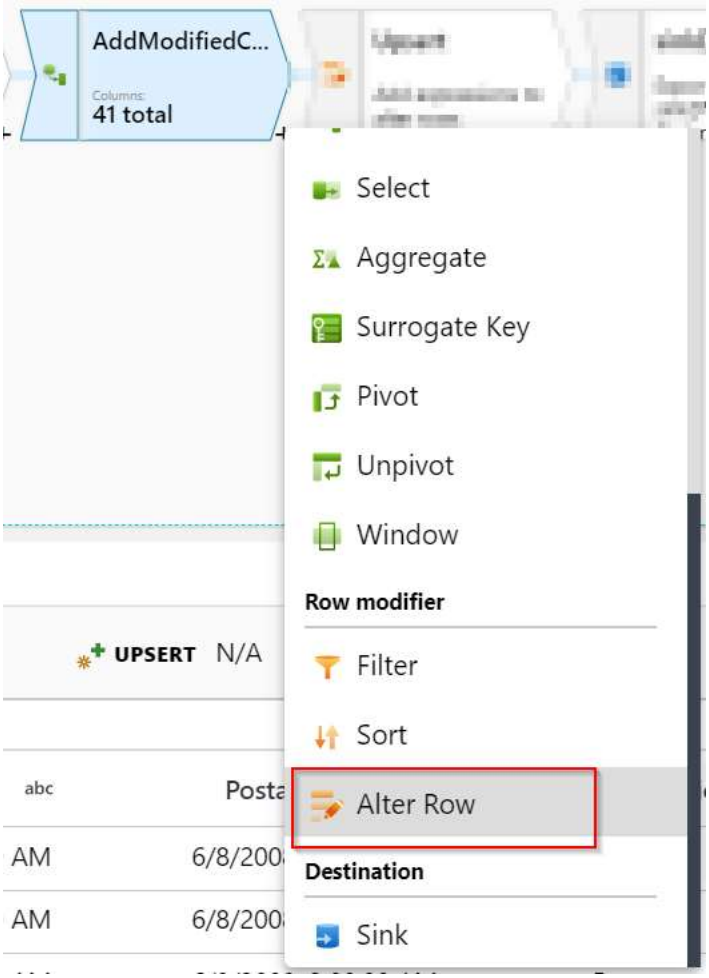
- Preview the data clicking on **Data Preview**

For the **upserts** being setup in the following chapter, this column gives you an idea, what records have been inserted and updated and when.

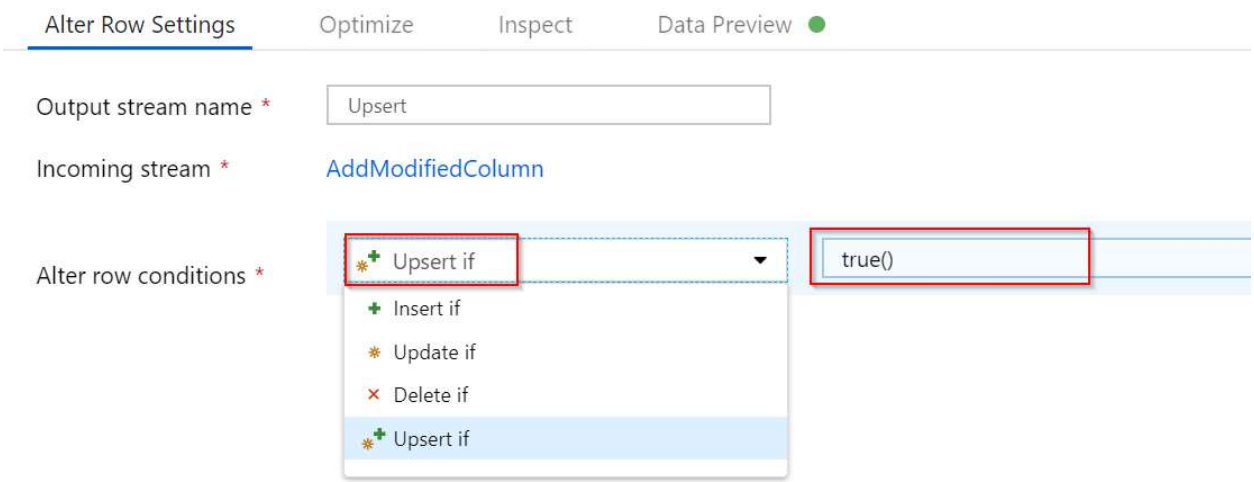
Alter row upserts conditions

Although not needed for this case, you can define much more complex conditions for your inserts, deletes, updates and upserts.

In order to achieve this, add the **Alter Row** activity



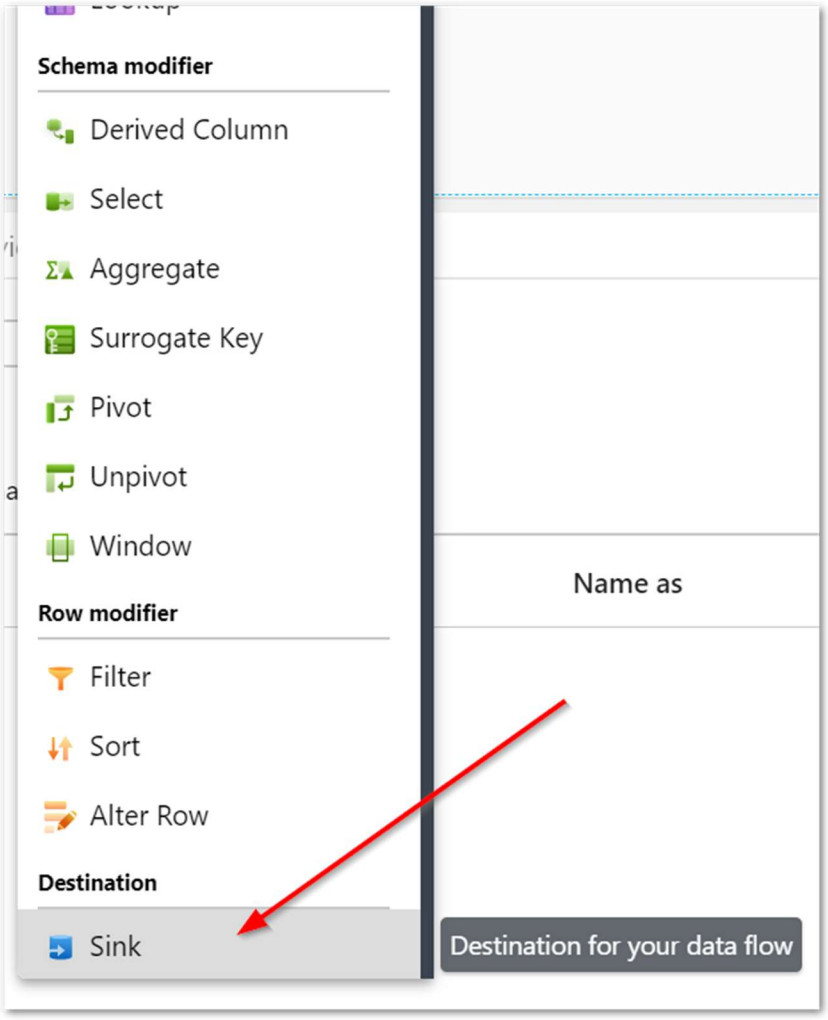
After you give your step a proper name, you can choose from the conditions (on the left) and the functions adding the business logic regurning **true()** or **false().**



For the sake of this demo, we put **true()** function (which is obviously always true) and thus does nothing – but you get the picture.

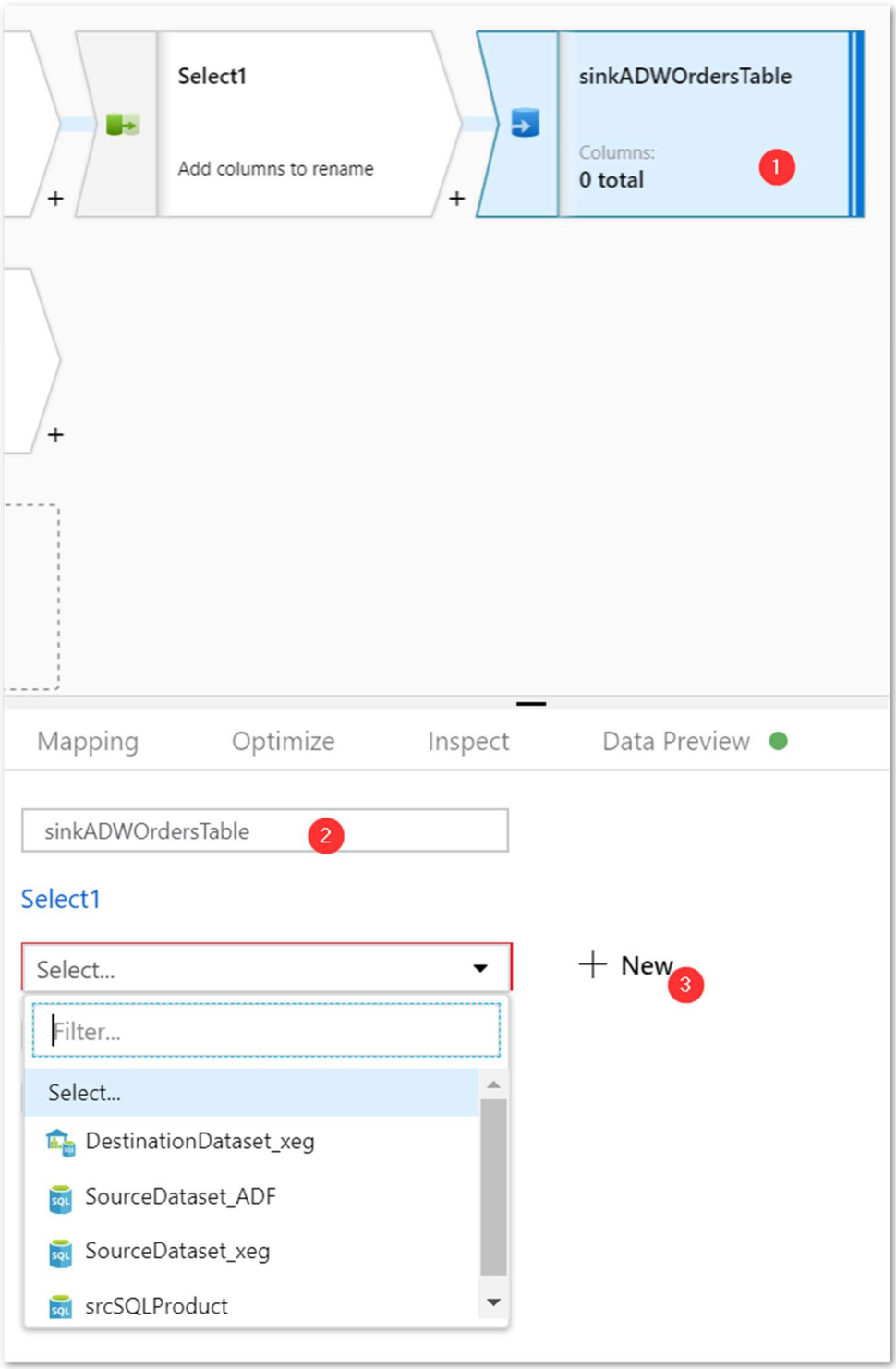
Create Azure Data Warehouse as a sink

Continuing in your flow, press **+** and choose Sink from the context menu.

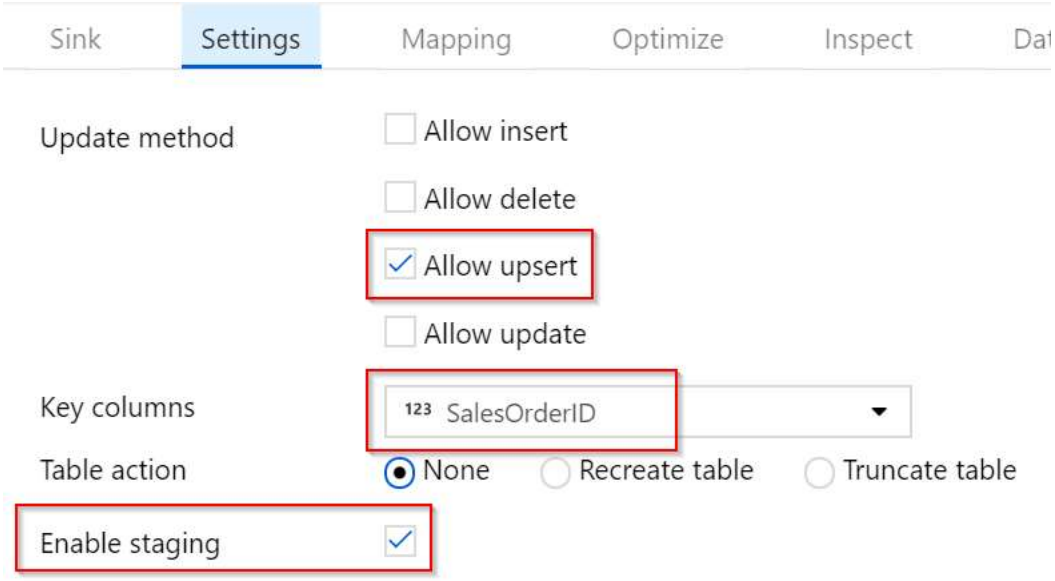


With the properties window, add the Azure Data Warehouse table as the sink for your flow. Create the new table and name it **SalesLT.CustomerOrders**.


General
Connection
Schema
Parameters
Linked service *
SQLDWH_Academy
Test connection
Table
SalesLT.CustomerOrders
Preview data
Edit



Following the “sink” settings, please make sure you allow **upserts,** choose the **SalesOrderID** as the mapping ID column and Enable staging for faster loads.

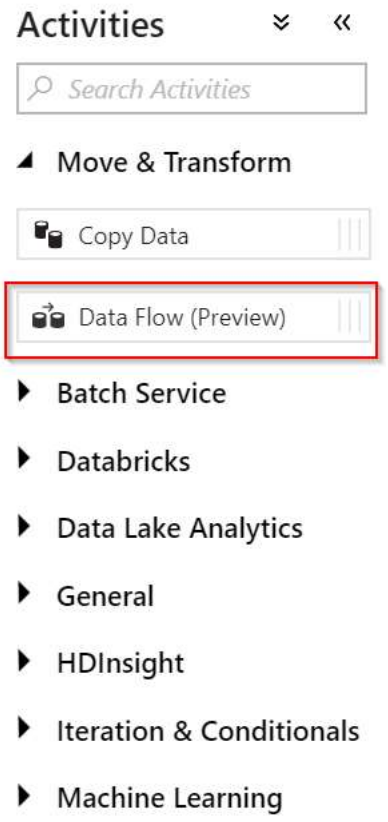


The settings above are available only when the sources and sinks supporting them are available (SQL DB & SQL DW).

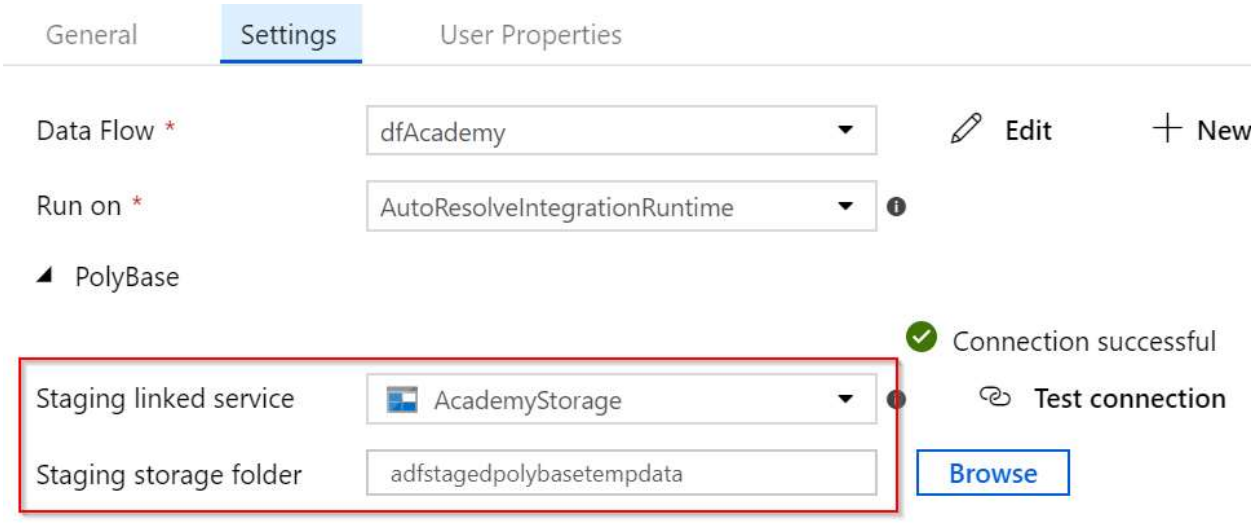
Create the pipeline to execute your data flow

ADF Mapping Data Flows are the ETL concept backed by Azure Databricks capabilities. As such, you define individual data flows and although you can debug them, you need the **Pipeline** to include the ADF Mapping Data Flow into some process. To be able to activate it upon trigger and monitor it.

- Create new pipeline and name it **CustomerOrdersSQLtoDW**.
- Choose Data Flow from the **Move & Transform** section.



- Choose your data flow within the popup window
- Click on the data flow to be able to set its “running” parameters
- On the **Settings** page, choose the proper Blob Storage created for this exercise and define the container where the data should be storage while staging



You can click Debug to run the Pipeline including your data flow or you can add trigger to activate your pipeline as usual.