LAB - Handle SQL truncation error rows in Data Factory mapping data flows

A common scenario in Data Factory when using mapping data flows, is to write your transformed data to a database in Azure SQL Database. In this scenario, a common error condition that you must prevent against is possible column truncation.

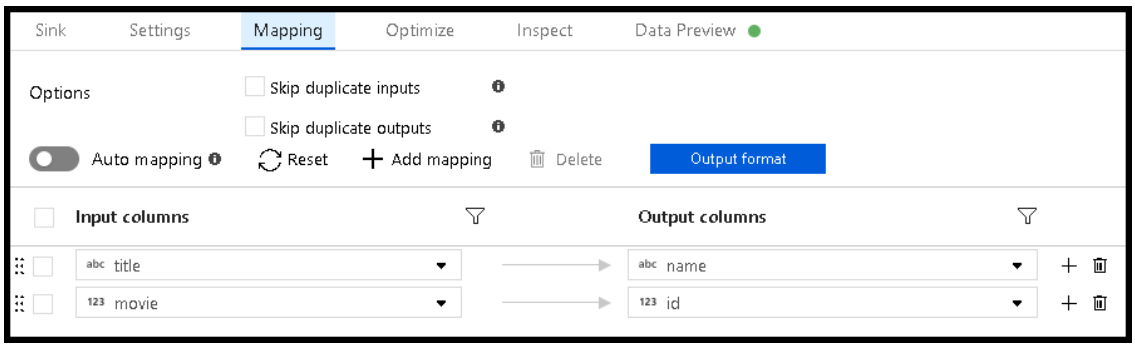
There are two primary methods to graceful handle errors when writing data to your database sink in ADF data flows:

- Set the sink error row handling to "Continue on Error" when processing database data. This is an automated catch-all method that does not require custom logic in your data flow.
- Alternatively, follow the steps below to provide logging of columns that won't fit into a target string column, allowing your data flow to continue.

> **Note**
>
> When enabling automatic error row handling, as opposed to the method below of writing your own error handling logic, there will be a small performance penalty incurred by and additional step taken by ADF to perform a 2-phase operation to trap errors.

Scenario

1. We have a target database table that has an `nvarchar(5)` column called "name".
2. Inside of our data flow, we want to map movie titles from our sink to that target "name" column.



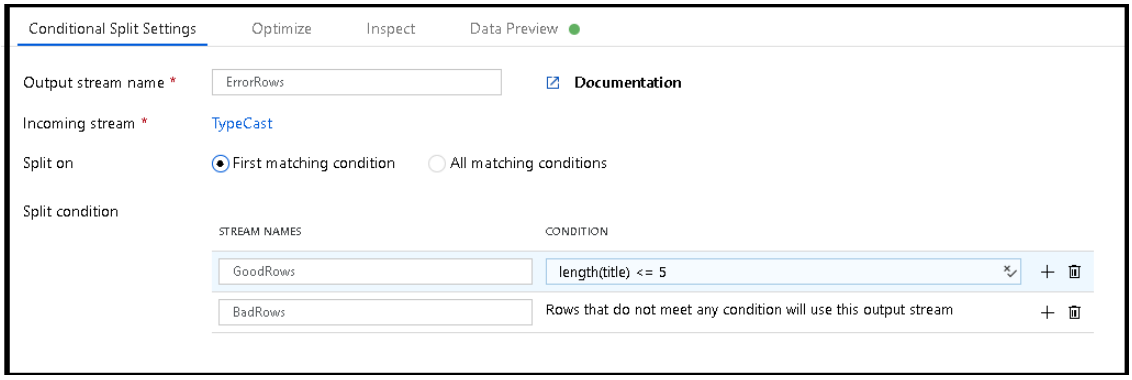
3. The problem is that the movie title won't all fit within a sink column that can only hold 5 characters. When you execute this data flow, you will receive an error like this one: `"Job failed due to reason: DF-SYS-01 at Sink 'WriteToDatabase': java.sql.BatchUpdateException: String or binary data would be truncated. java.sql.BatchUpdateException: String or binary data would be truncated."`

This video walks through an example of setting-up error row handling logic in your data flow:

https://www.microsoft.com/en-us/videoplayer/embed/RE4uOHj

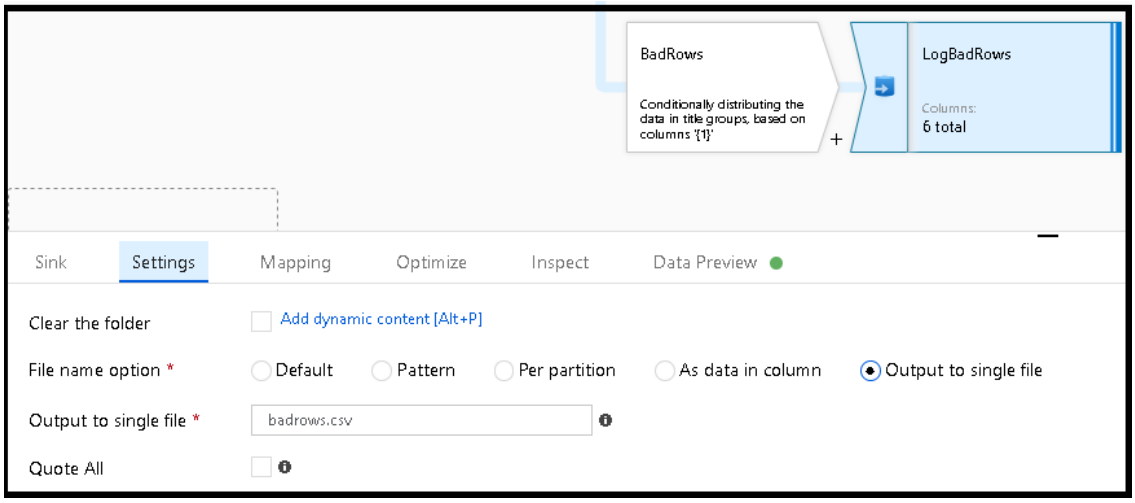
How to design around this condition

1. In this scenario, the maximum length of the "name" column is five characters. So, let's add a conditional split transformation that will allow us to log rows with "titles" that are longer than five characters while also allowing the rest of the rows that can fit into that space to write to the database.

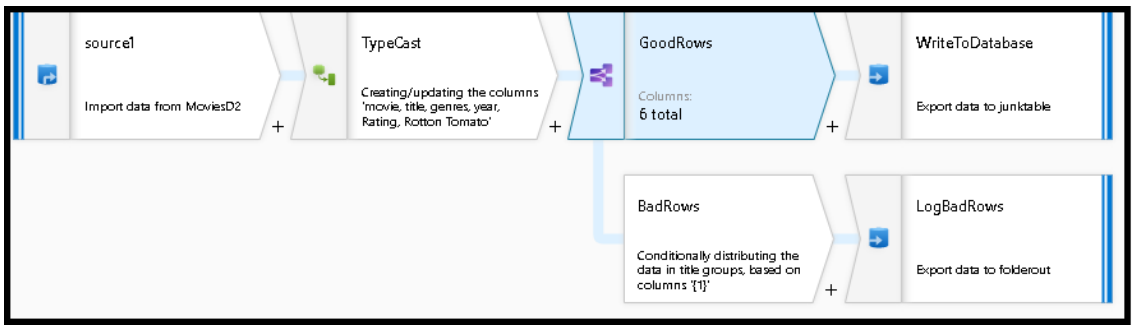


2. This conditional split transformation defines the maximum length of "title" to be five. Any row that is less than or equal to five will go into the `GoodRows` stream. Any row that is larger than five will go into the `BadRows` stream.

3. Now we need to log the rows that failed. Add a sink transformation to the `BadRows` stream for logging. Here, we'll "auto-map" all of the fields so that we have logging of the complete transaction record. This is a text-delimited CSV file output to a single file in Blob Storage. We'll call the log file "badrows.csv".



4. The completed data flow is shown below. We are now able to split off error rows to avoid the SQL truncation errors and put those entries into a log file. Meanwhile, successful rows can continue to write to our target database.



5. If you choose the error row handling option in the sink transformation and set "Output error rows", ADF will automatically generate a CSV file output of your row data along with the driver-reported error messages. You do not need to add that logic manually to your data flow with that alternative option. There will be a small performance penalty incurred with this option so that ADF can implement a 2-phase methodology to trap errors and log them.

	A	B	C	D	E	F
1	name	id	genres	DF_SQL_Op	DF_Error_Details	
2	Trip to the [redacted]	32898	Action|Adventure|Fantasy|Sci-Fi	INSERT	[{\ErrorCode\":\"2628\"	\"Message\":\"String or binary data would be truncated in table [dbo].[junk]
3	Birth of a Na[redacted]	7065	Drama|War	INSERT	[{\ErrorCode\":\"2628\"	\"Message\":\"String or binary data would be truncated in table [dbo].[junk]
4	Intolerance:[redacted]	7243	Drama	INSERT	[{\ErrorCode\":\"2628\"	\"Message\":\"String or binary data would be truncated in table [dbo].[junk]