# Data Tools

## Contents

## 1. Conditional Split

Splits the data based on certain conditions being met. For example, this transformation could be instructed to send data down a different path if the State column is equal to Florida. The Conditional Split Transformation is a fantastic way to add complex logic to your Data Flow. This transformation enables you to send the data from a single data path to various outputs or paths based on conditions that use the SSIS expression language. For example, you could configure the transformation to send all products with sales that have a quantity greater than 500 to one path, and products that have more than 50 sales down another path. Lastly, if neither condition is met, the sales would go down a third path, called "Small Sale," which essentially acts as an ELSE statement in T-SQL. You can drag and drop the column or expression code snippets from the tree in the top-right panel. After you complete the condition, you need to name it something logical, rather than the default name of Case 1. You'll use this case name later in the Data Flow. You also can configure the "Default output name," which will output any data that does not fit any case. Each case in the transform and the default output name will show as a green line in the Data Flow and will be annotated with the name you typed in.

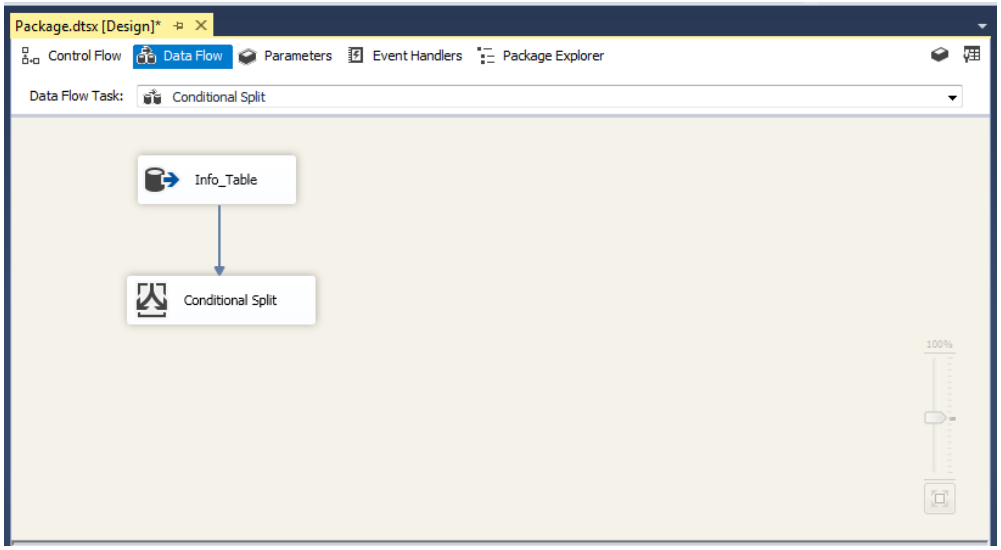
**Example:** suppose you have the following table:

**Info**

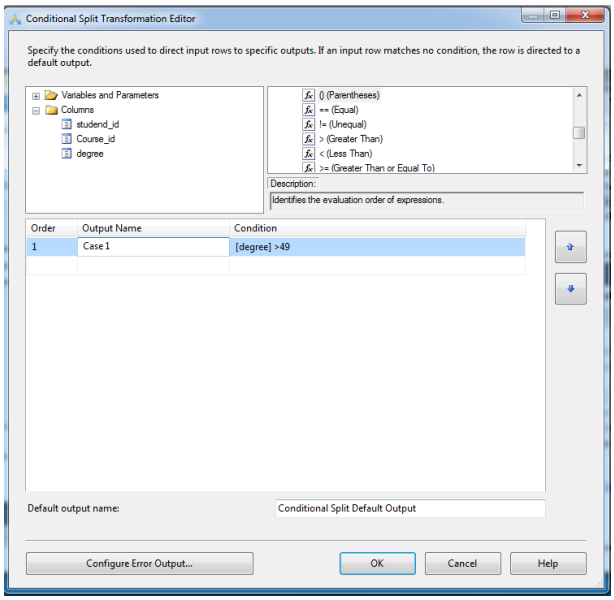
| Student_id | Course_ID | degree |
|---|---|---|
| 1 | IS202 | 80 |
| 2 | IS205 | 47 |
| 3 | IS401 | 46 |
| 4 | IS401 | 87 |
| 5 | IS402 | 88 |

Design an ETL to store the data of success students in a table call Success_Stud and the other data in Failed_Stud.

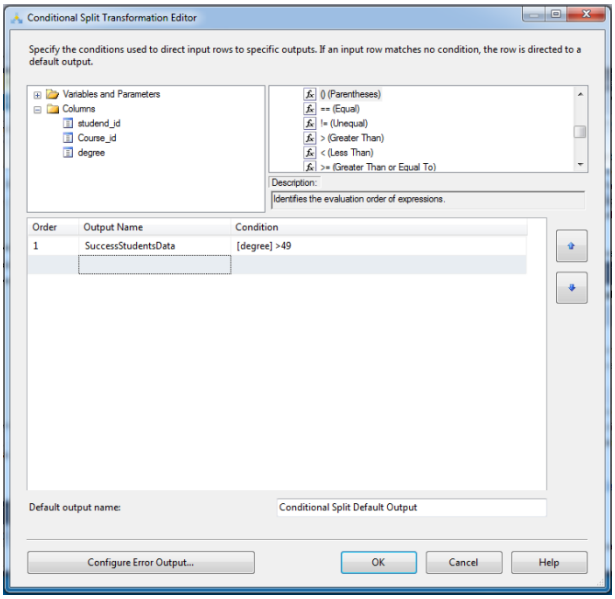
Sol: Drag and drop a control task (Data flow task) and open it, add the source (info table) and configure it, then add a data flow task (conditional split) and connect the data source with it:



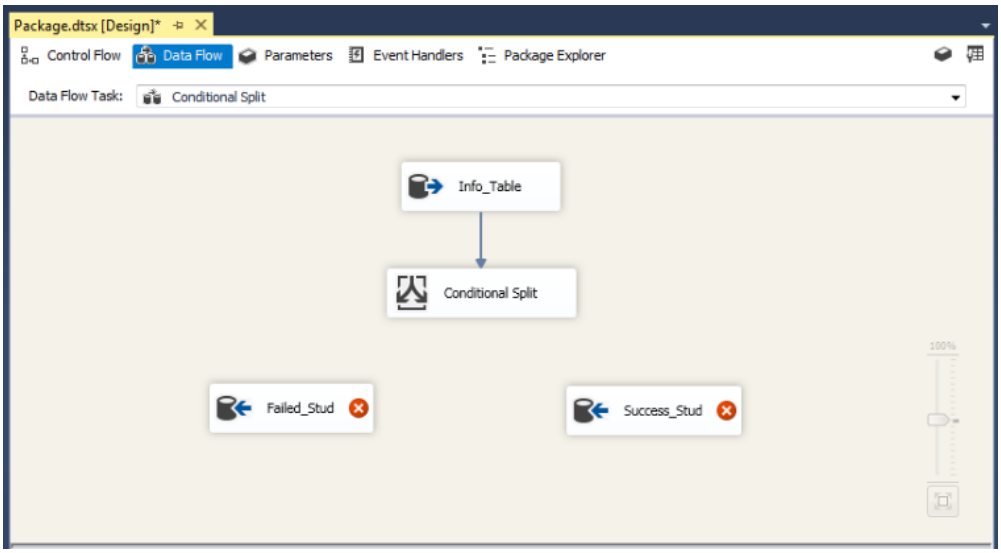

Configure the Conditional Split task by open it and add new condition for the data flow of success students. Drag and drop the column (degree) and add the comparison (>) from operators then add the value (49), so the result condition will be like:



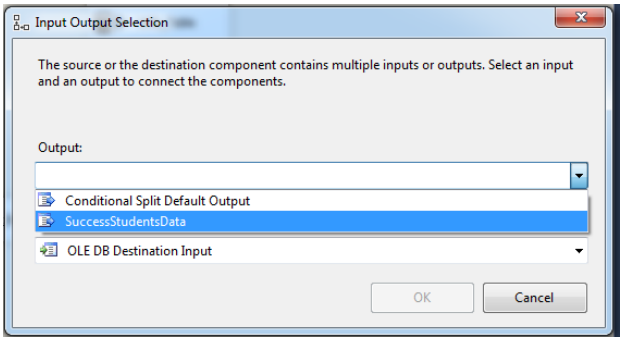
You can change the output name into any name, and the preferable name can be the name that describe the nature of data, so the name will be (SuccessStudentsData):




We can add another condition to filter the other data, but seems not reasonable to add this condition since the other data will go to the other path. Click OK and add two destinations for two tables (Success_Stud and Failed_Stud).




Connect the conditional split to Success_Stud destination and configure the output path by selecting SuccessStudentsData which has been configured already:




Click OK and create new destination table to save the data of success students. Connect the Conditional Split Default Output to Failed_Stud and create new destination table to save the data then click Run to split the data of the source table and store them in the destination tables (Success_Stud and Failed_Stud).

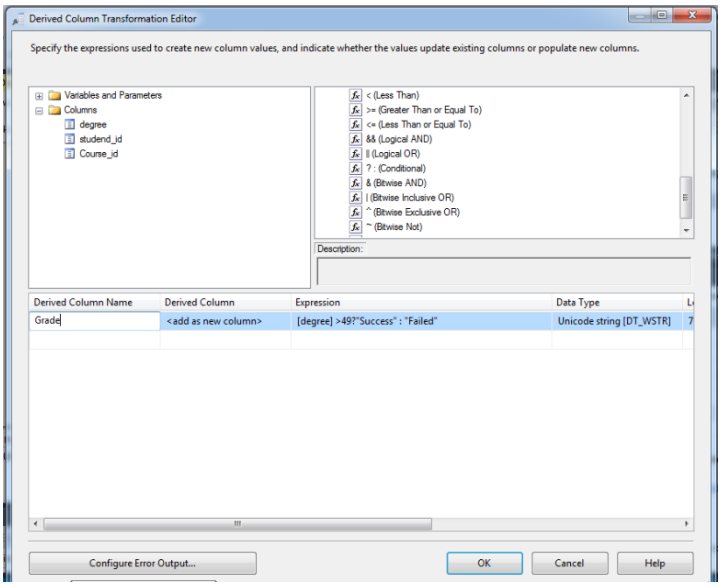
## 2. Derived Column

The Derived Column Transformation creates a new column that is calculated (derived) from the output of another column or set of columns. It is one of the most important transformations in your Data Flow arsenal. You may wish to use this transformation, for example, to multiply the quantity of orders by the cost of the order to derive the total cost of the order. You can also use it to find out the current date or to fill in the blanks in the data by using the ISNULL function. This is one of the top five transformations that you will find yourself using to alleviate the need for T-SQL scripting in the package.

You'll find all the available functions for the expression language in the top-right pane of the editor. There are no hidden or secret expressions in this C# variant expression language. We use the expression language much more throughout this and future chapters so don't worry too much about the details of the language yet.

Example: based on the previous table (info), create new table (Grade_table) that holds new column called grade which takes the value (Success) when the degree>49 and (Failed) when degree<=49.

Sol: add the source table and derived column in the data flow task and connect the source to derived column and configure it:




Rename the derived Column Name into Grade, from Derived Colum select add as new column, from the Expression add the conditional (from operators). The conditional is (Degree>49?"Succes":"Failed"). The condition is (Degree>49), when it is true the value will be "Success" and when it is false the value will be "Failed". Add new destination and connect the derived column to the destination and configure the destination to create new table to store the new derived column with old columns.

## 3. Merge

The Merge Transformation can merge data from two paths into a single output. This transformation is useful when you wish to break out your Data Flow into a path that handles certain errors and then merge it back into the main Data Flow downstream after the errors have been handled. It's also useful if you wish to merge data from two Data Sources. This transformation is similar to the Union All Transformation, but the Merge Transformation has some restrictions that may cause you to lean toward using Union All:

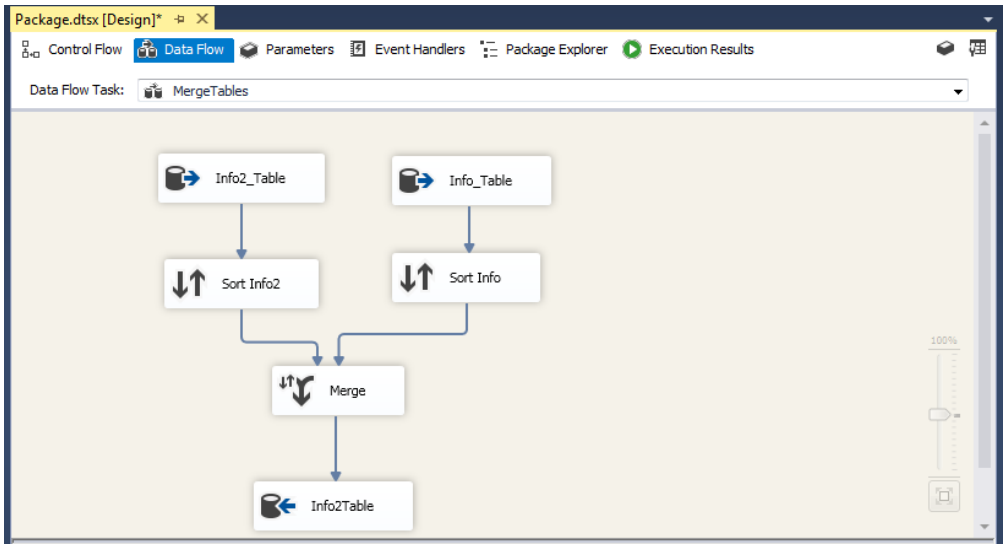
➤➤The data must be sorted before the Merge Transformation. You can do this by using the Sort Transformation prior to the merge or by specifying an ORDER BY clause in the source connection.

➤➤The metadata must be the same between both paths. For example, the CustomerID column can't be a numeric column in one path and a character column in another path.

➤➤If you have more than two paths, you should choose the Union All Transformation. To configure the transformation, ensure that the data is sorted exactly the same on both paths and drag the path onto the transform. You'll be asked if the path you want to merge is Merge Input 1 or 2. If this is the first path you're connecting to the transformation, select Merge Input 1. Next, connect the second path into the transformation. The transformation will automatically configure itself. Essentially, it maps each of the columns to the column from the other path, and you have the option to ignore a certain column's data.

**Example**: merge two tables (info and info2) in one table info3.

**Sol**: the data flow task should be like the following:



The table info should be sorted by any column (student_id in the example), the info2 table should also sorted based on the same column in the first table (student_id) in order to be merged. The merge receive two input, the data of first input table will be inserted first in the destination table and followed by data of the second table.