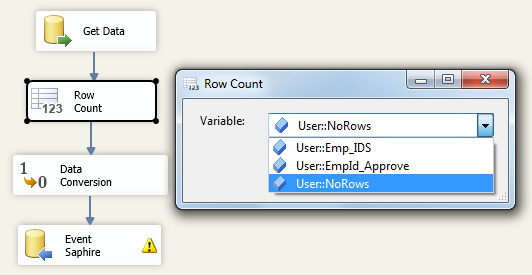
|  |  |  |
| --- | --- | --- |
| **Transformation** | **Description** | **Examples of when Transformation Would be Used** |
| Aggregate | Calculates aggregations such as SUM, COUNT, AVG, MIN and MAX based on the values of a given numeric column. This transformation produces additional output records. | Adding aggregated information to your output. This can be useful for adding totals and sub-totals to your output. |
| Audit | Includes auditing information, such as computer name where the package runs, package version ID, task name, etc in the data flow. | Creates advanced logs: where and when the package was executed, how long it took to run the package and the outcome of execution. |
| Character Map | Minor manipulations on string columns. Converts all letters to uppercase, lowercase, reverse bytes, etc. | Applying string manipulations prior to loading or while loading data. |
| **Conditional Split** | Accepts an input and determines which destination to pipe the data into based on the result of an expression. | Cleansing the data to extract specific rows from the source. If a specific column does not conform to the predefined format (perhaps it has leading spaces or zeros), move such records to the error file. |
| Copy Column | Makes a copy of a single or multiple columns which will be further transformed by subsequent tasks in the package. | Extracting columns that need to be cleansed of leading / trailing spaces, applying character map transformation to uppercase all data and then load it into the table. |
| **Data Conversion** | Converts input columns from one data type to another. | Converting columns extracted from the data source to the proper data type. |
| Data Mining Query | Queries a data mining model. Includes a query builder to assist you with development of Data Mining eXpressions (DMX) prediction queries. | Evaluating the input data set against a data mining model developed with Analysis Services. |
| **Derived Column** | Calculates new column value based on an existing column or multiple columns. | Removing leading and trailing spaces from a column. Add title of courtesy (Mr., Mrs., Dr, etc) to the name. |
| Export Column | Exports contents of large columns (TEXT, NTEXT, IMAGE data types) into files. | Saving large strings or images into files while moving the rest of the columns into a transactional database |
| Fuzzy Grouping | Finds close or exact matches between multiple rows in the data source. Adds columns to the output including the values and similarity scores. | Cleansing data by translating various versions of the same value to a common identifier. For example, "Dr", "Dr.", "doctor", "M.D." should all be considered equivalent. |
| Fuzzy Lookup | Compares values in the input data source rows to values in the lookup table. Finds the exact matches as well as those values that are similar. | Cleansing data by translating various versions of the same value to a common identifier. For example, "Dr", "Dr.", "doctor", "M.D." should all be considered equivalent. |
| Import Column | Imports contents of a file and appends to the output. Can be used to append TEXT, NTEXT and IMAGE data columns to the input obtained from a separate data source. | This transformation could be useful for web content developers. For example, suppose you offer college courses online. Normalized course meta-data, such as course\_id, name, and description is stored in a typical relational table. Unstructured course meta-data, on the other hand, is stored in XML files. You can use Import Column transformation to add XML meta-data to a text column in your course table. |
| **Lookup** | Joins the input data set to the reference table, view or row set created by a SQL statement to lookup corresponding values. If some rows in the input data do not have corresponding rows in the lookup table then you must redirect such rows to a different output. | Obtaining additional data columns. For example, the majority of employee demographic information might be available in a flat file, but other data such as department where each employee works, their employment start date and job grade might be available from a table in relational database. |
| Merge | Merges two sorted inputs into a single output based on the values of the key columns in each data set. Merged columns must have either identical or compatible data types. For example you can merge VARCHAR(30) and VARCHAR(50) columns. You cannot merge INT and DATETIME columns. | Combining the columns **from different databases / data sources** into a single row set prior to populating a dimension table in a data warehouse. With prior versions of SQL Server you had to populate the staging area first if your data warehouse had multiple transactional data sources. |
| Merge Join | Joins two sorted inputs using INNER JOIN, LEFT OUTER JOIN or FULL OUTER JOIN algorithm. You can specify columns used for joining inputs. | Combining the columns **from different databases / data sources** into a single row set prior to populating a dimension table in a data warehouse. Using Merge Join transformation saves the step of having a temporary staging area. With prior versions of SQL Server you had to populate the staging area first if your data warehouse had multiple transactional data sources. NOTE: Merge Join can only combine 2 data sets at a time. However, you can use multiple Merge Joins. |
| Multicast | Similar to the conditional split transformation, but the entire data set is piped to multiple destinations. | Populating the relational warehouse as well as the source file with the output of a derived column transformation. |
| **OLEDB Command** | Runs a SQL command for each input data row. UPDATE employee\_source SET has\_been\_loaded=1 WHERE employee\_id=? | Setting was\_loaded field to 1 after the data row has been loaded. This way the subsequent loads will only attempt importing the rows that haven't made it to the warehouse as of yet. |
| Percentage Sampling | Loads only a random subset of your data, defined as the percentage of all rows in the data source. | For testing a sample |
| Pivot | Pivots the normalized data set by certain column to create a more easily readable output. Similar to PIVOT command in Transact-SQL. You can think of this transformation as converting rows into columns. For example if your input rows have customer, account number and account balance columns the output will have the customer and one column for each account. | Creating a row set that displays the table data in a more user-friendly format. The data set could be consumed by a web service or could be distributed to users through email. |
| Row count | Counts the number of transformed rows and store in a variable. | Determining the number of rows transform |
| **Row sampling** | Loads a random subset of your data, defined as the number of rows. | For testing a sample |
| **Script Component** | Script Component allows you to write transformations for otherwise un-supported source and destination file formats. Script component also allows you to perform transformations not directly available through the built-in transformation algorithms. (In .NET) | Custom transformations can call functions in managed assemblies, including .NET framework. This type of transformation can be used when the data source (or destination) file format cannot be managed by typical connection managers. For example, some log files might not have tabular data structures. At times you might also need to parse strings one character at a time to import only the needed data elements. |
| Slowly Changing Dimension | Maintains historical values of the dimension members when new members are introduced. | Useful for maintaining dimension tables in a data warehouse when maintaining historical dimension member values is necessary. |
| Sort | Sorts input by column values. You can sort the input by multiple columns in either ascending or descending order. The transformation also allows you to specify the precedence of columns used for sorting. This transformation could also discard the rows with duplicate sort values. | Ordering the data prior to loading it into a data warehouse. This could be useful if you're ordering your dimension by member name values as opposed to sorting by member keys. You can also use Sort transformation prior to feeding the data as the input to the Merge Join or Merge transformation. |
| Term Extraction | Extracts terms (nouns and noun phrases) from the input text into the transformation output column. | Processing large text data and extracting main concepts. For example, you could extract the primary terms used in this section of SQLServerPedia by feeding the Term Extraction transformation the text column containing the entire section. |
| Term Lookup | Extracts terms from the input column with TEXT data type and match them with same or similar terms found in the lookup table. Each term found in the lookup table is scanned for in the input column. If the term is found the transformation returns the value as well as the number of times it occurs in the row. You can configure this transformation to perform case-sensitive search. | Analyzing large textual data for specific terms. For example, suppose you accept email feedback for latest version of your software. You might not have time to read through every single email messages that comes to the generic inbox. Instead you could use this task to look for specific terms of interest. |
| Union ALL | Combines multiple inputs into a single output. Rows are sorted in the order they're added to the transformation. You can ignore some columns from each output, but each output column must be mapped to at least one input column. | Import data from multiple disparate data sources into a single destination. For example, you could extract data from mail system, text file, Excel spreadsheet and Access database and populate a SQL Server table. |
|  |
| Unlike Merge and Merge Join transformations Union ALL can accept more than two inputs. |
| Unpivot | Opposite to Pivot. Converts columns into rows. It normalizes the input data set that has many duplicates values in multiple columns by creating multiple rows that have the same value in a single column. | If your input has a customer name and a separate column for checking and savings' accounts Unpivot can transform it into a row set that has customer, account and account balance columns |



**Data Conversion**

