# SSIS

<https://www.youtube.com/watch?v=NGzieSedvuM&list=PL_YF--8vjjEVEXMf2hEFn0D5tEJV9kRqi&index=1&t=447s>

video series

SSIS variable

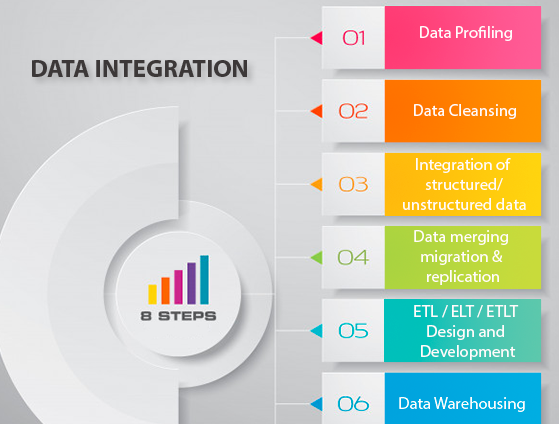
SSIS Parameter

## What is SSIS

* SSIS stands for SQL Server Integration Services.
* It is a data warehousing tool used for data extraction, loading the data into another database, transformations such as cleaning, aggregating, merging data, etc.
* SSIS tool also contains the graphical tools and window wizards workflow functions such as sending email messages, ftp operations, data sources.

## Main Functionalities

### ****Data Integration****

* Combining the data from multiple sources and provides unified data to the users
* The data can be either heterogeneous data or homogeneous data
* The data can be structured, semi-structured, or unstructured.
* 

### ****Workflow****

Workflow can be used to perform several things.

## Requirement for SSIS

* Install the SQL Server
* Install the SQL Server Data Tools

## ****Component of SSIS package****

1. **Connections**  
   SSIS package will have some connections, and these connections are used to connect to various data sources.
2. **Control flow elements**  
   Control flow elements handle workflows. Workflow means that we are performing all SSIS tasks in steps, so the sequence is done through control flow.

**Data Flow task, SQL Task, Package Task, Process task, File Task, FTP Task, Mail Task**

1. **Data flow elements**  
   The data flow elements perform transformations.

**Source🡪 Transformation🡪Target**

## SSIS Tasks under Control Flow

 A task is a unit of work and we will have different types of tasks to perform different kinds of work. There are various types of tasks, but we will discuss the most common tasks used in SSIS:

### ****Execute SQL Task****

It is used to execute the SQL statements against a relational database.

Data Flow Task  
It is mainly used to read the data from one or multiple data sources, transform the data and can also load the data to one or more destinations.

### ****Execute Package Task****

This task is used to call the other packages within the same project. You can even pass the variable values to the called package.

### ****Execute Process Task****

* run an application or batch scripts as a SQL Server Integration Services.
* open the standard application such as Microsoft Excel, Microsoft Word,
* unzip the compressed file.

### ****File System Task****

manipulations in the file system such as moving files, deleting files, renaming the files, changing the directory, etc.

### ****FTP Tasks****

to send or receive the file from the FTP server to the local directory, then the SSIS FTP task is used.

### **Script Task**

task allows you to write the .Net code that you want to perform.

### ****Send Mail Task****

 used to send an email. It is mainly used when you want to notify the users regarding the state of the task whether it is in a running state or some error has occurred.

# Connection Manager

Integration Services uses the connection manager as a logical representation of a connection. A connection manager includes the **ConnectionString** property that you set at design time; at run time, a physical connection is created using the value in the connection string property.

* There are built-in connection managers that Setup installs when you install Integration Services.
* There are connection managers that are available for download from the Microsoft website.
* You can create your own custom connection manager if the existing connection managers do not meet your needs.

## Connection Manager level

Project --It is available all the packages in the project.

Package –It is available to that specific package.

### Difference between ODBC and OLEDB

ODBC:- Only for relational databases (Sql Server, Oracle etc)

OLE DB:- For both relational and non-relational databases. (Oracle, Sql-Server, Excel, raw files, etc)

## Connection Type

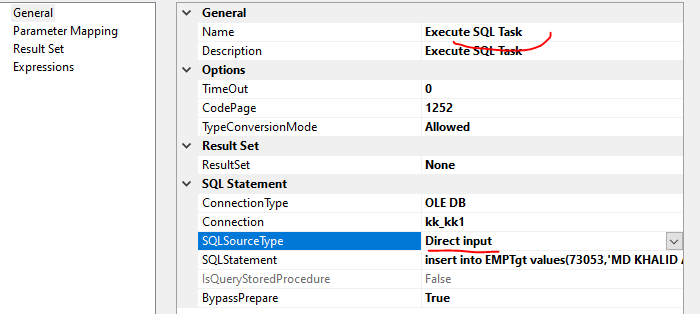
|  |  |
| --- | --- |
| Type | Description |
| OLEDB | SQL Server connection |
| ODBC | Oracle |
| EXCEL | Excel Sheet Connection |
| FLATFILE | Connect to data in a single flat file. |
| FILE | Connects to a file or a folder. |
| FTP | Connect to an FTP server. |
| SMTP | Connects to an SMTP mail server. |

# Execute SQLTask

## Variables and parameter

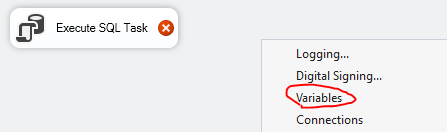
### SQLSourceType=Direct Input

Step1: SQL Task in **Control Flow** And **SQLSourceType=**Direct Input



Step2: Open variable window

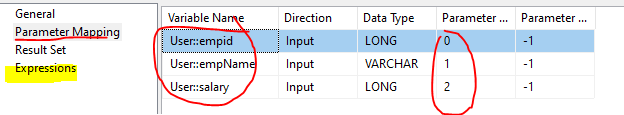
Right click in package body 🡪click on variable🡪 Create variable



Step3: add ? mark in SQLstatement like below

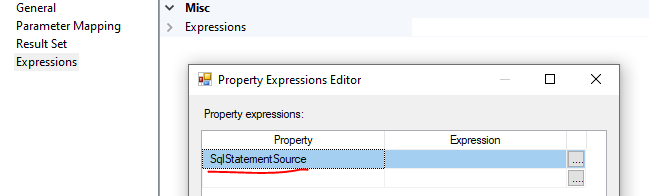


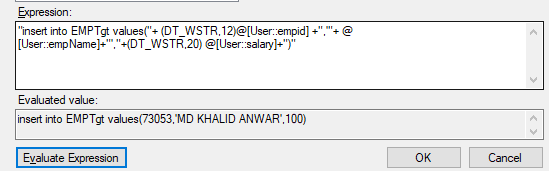
Step4: configure **Parameter Mapping**---Add parameter and parameter index as below



OR --

Step4a: Remove parameter mapping and Add **Expression**





For String variable type, there is no issue in string concatenation

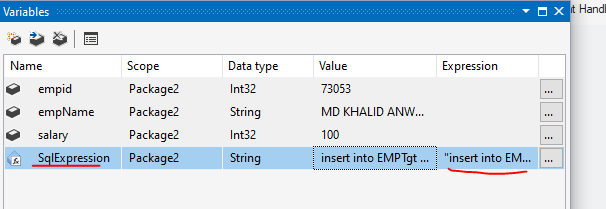
For Integer variable, you need to cast to string like (DT\_WSTR,20) variableName.

| **Data type** | **Parameter** | **Example** |
| --- | --- | --- |
| DT\_STR | *charcount*  *codepage* | (DT\_STR,30,1252) casts 30 bytes, or 30 single characters, to the DT\_STR data type using the 1252 code page. |
| DT\_WSTR | *Charcount* | (DT\_WSTR,20) casts 20 byte pairs, or 20 Unicode characters, to the DT\_WSTR data type. |
| DT\_BYTES | *Bytecount* | (DT\_BYTES,50) casts 50 bytes to the DT\_BYTES data type. |
| DT\_DECIMAL | *Scale* | (DT\_DECIMAL,2) casts a numeric value to the DT\_DECIMAL data type using a scale of 2. |
| DT\_NUMERIC | *Precision*  *Scale* | (DT\_NUMERIC,10,3) casts a numeric value to the DT\_NUMERIC data type using a precision of 10 and a scale of 3. |
| DT\_TEXT | *Codepage* | (DT\_TEXT,1252) casts a value to the DT\_TEXT data type using the 1252 code page. |

### SQLSourceType=variable

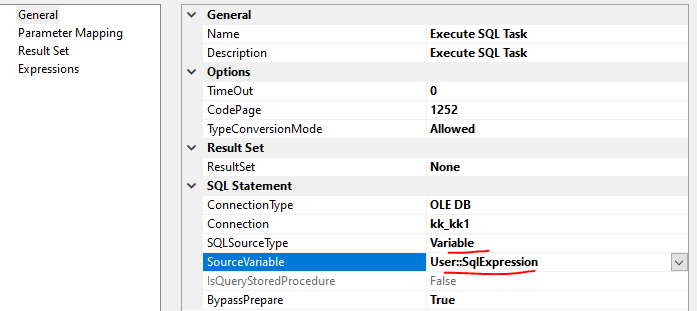
Step1: SQL Task in **Control Flow** And **SQLSourceType=**variable

Step2: Create variable and add expression as below



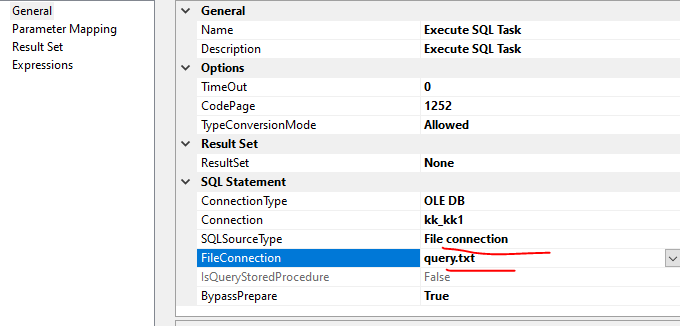
Step3: Remove other parameter mapping or any expression

Use this variable name as below



### SQLSourceType=File Connecion

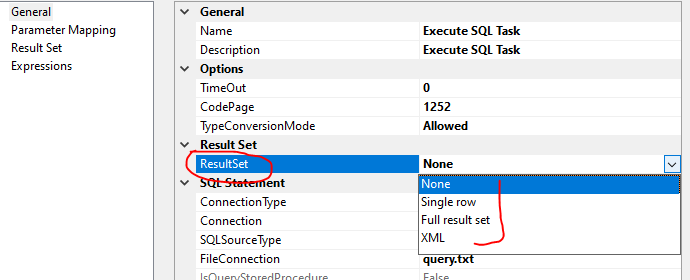
We can execute SQL query file using this parameter



## Direct input=Direct Input

The next section on the **General** page is **Result Set**. Notice that this section includes only the**ResultSet** property. The property lets you select one of the following four options:

* **None:** The query returns no result set.
* **Singlerow:** The query returns a single-row result set.
* **Fullresultset:** The query returns a result set that can contain multiple rows.
* **XML:** The query returns a result set in an XML format.



The option you select depends on the results of the query you pass into the **Execute SQL** task. For this exercise, our query will return only a single value. Consequently, we will choose the **Single row** option.

Next, we need to configure the properties in the **SQL Statement** section. Table 1 shows the values you should use to configure these properties.

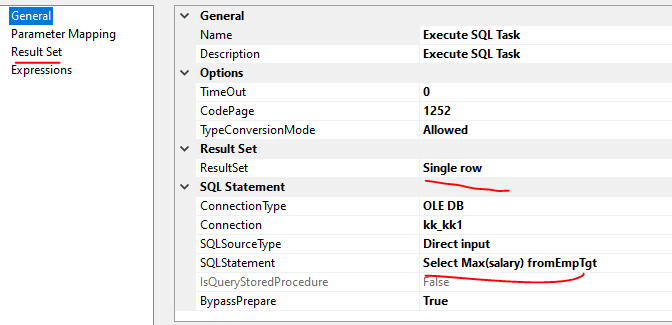
| Property | Value |
| --- | --- |
| Connection | AW2008 (or whatever you named the connection manager you created earlier) |
| SQLSourceType | **Direct input**  This means we’ll type the code straight in and not use a stored procedure. |
| SQLStatement | Because we’ve selected the Direct input option, we need to enter a T-SQL statement for this option. I’ve used the following statement, which returns a single value:      SELECT MAX(EmployeeID) AS [MaxEmpID]   FROM HumanResources.Employee |
| IsQueryStoredProcedure | This option is greyed out because we selected Direct input for theSQLSourceType property. Had we selected Stored Procedure, this property would be available and the SQLStatement property would be greyed out. |
| BypassPrepare | The property defaults to False. If you change the value to True, you can click the Parse Query button to verify that your T-SQL statement is valid. |

Our next step is to associate our result set value with a variable that will store the value we retrieve from the database.

### Result Set(storing query output into variable)

To do this, go to the **Result Set** page of the **Execute SQL Task Editor**

The main grid of the **Result Set** page contains two columns: **Result Name** and **Variable Name**. Click the **Add** button to add a row to the grid. In the **Result Name** column, enter the column name returned by your query (**MaxEmpID**). In the **Variable Name** column, select the **User:: EmpNum**variable.



### ****Parameter Mapping(reading variable value into query)****

* **Variable Name:** The variable that contains the value to be used for the parameter. In this case, we’ll use the **User:: EmpNum** variable, which contains the result set value returned by the first **Execute SQL** task.
* **Direction:** Determines whether to pass a value into a parameter (input) or return a value through the parameter (output)
* **Data Type:** Determines the type of data provided from the variable. This will default to the type used when setting up the variable.
* **Parameter Name:** The name of the parameter. The way in which parameters are named depends on your connection type. When running a T-SQL statement against a SQL Server database through an OLE DB connection, as we’re doing here, we use numerical values to represent the statement’s parameters, in the order they appear in the statement, starting with**0**. In this case, because there’s only one parameter, we use **0**.
* **Parameter Size:** The size of the parameter if it can be a variable length. The default is **-1**, which lets SQL Server determine the correct size.

## Full result set

“Full result set” to a set variable of Object in Execute SQL TASK

In the General tab, set ResultSet to **Full result set**.

Query used

Select id,name from emp where emp>0—It should return more than one record

Create variable with data type as object

In the Result Set tab, add  a result set and set 0 for Result Name, and User::vLoopSet as Variable Name. 0 (zero) is the index number from the results of the SQL query. Index 0 (zero) will give us the first column from the SQL query.

# For Each Loop Container

## Foreach File Enumerator

Enumerates files in a folder;

Fully qualified:

The fully qualified file name should be returned when a file in the enumerated list is being referenced.

Name and extension:

The file name with its extension should be returned when a file in the enumerated list is being referenced.

Name only:

The file name without its extension should be returned when a file in the enumerated list is being referenced.

## Foreach Item Enumerator:

1. Enumerates items in a collection, such as the executables specified in an Execute Process task.

## Foreach ADO Enumerator:

1. Enumerates rows in a table, such as the rows in an ADO recordset.

## Foreach ADO.NET Schema Rowset Enumerator:

1. Enumerates schema information about a data source.

## Foreach From Variable Enumerator:

1. Enumerates a list of objects in a variable, such as an array or ADO.NET DataTable.

## ForeachNodeList Enumerator:

1. Enumerates the result set of an XML Path Language (XPath) expression.

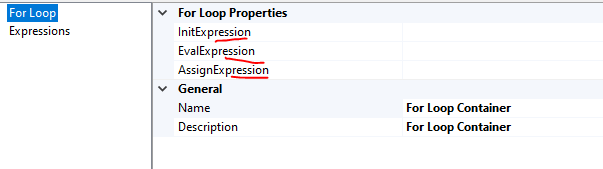
## Foreach SMO Enumerator:

1. Enumerates a list of SQL Server Management Objects (SMO) objects, such as a list of views in a database.

# For Loop Container

The red circle with white X indicates that we need to configure the For Loop Container. Double-click on the For Loop Container to display the For Loop Editor window. We will set values for the

* **InitExpression**
* **EvalExpression**
* **AssignExpression**

****

**Step1**:Create variable

I default value=0

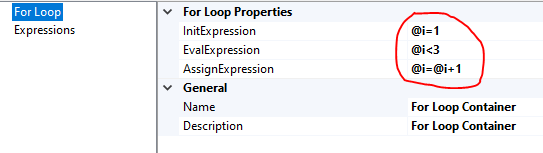
**Step2**: Open For loop container properties and put details as below

InitExpression ===@i=1

EvalExpression===@i<5

AssignExpression== @i=@i +1

Step3: Whatever task is available inside container then it will be executed 5 times



# Sequence Container

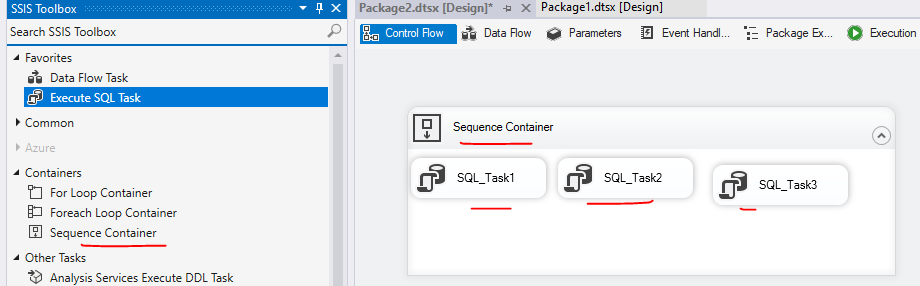
To run multiple task in parallel, we use this container. As per below screen , All 3 SQL Task will be executed in parallel.

Below task will be executed in Parallel

SQL\_Task1

SQL\_Task2

SQL\_Task3



# Debug package

Right click in designer page

Click on edit breakpoints

Check box correspondingly

# Data viewers

Right click on arrow in data flow

Click on data viewer

Click on add and select grid view

# Transformation

## Aggregate

An Asynchronous full blocking transformation, Aggregate transformation allows to aggregate data from Data Flow to apply certain T-SQL functions that are done in a GROUP BY statement. we have applied SUM aggregation but Aggregation transformation provides other options to aggregate data like Count, Count distinct, Average, Minimum and Maximum.

Check columns for grouping and aggregation

In operation column, select what operation has to be performed either group by or aggregation like sum etc

## Conditional Split

Synchronous transformation, allows you to send the data from a single data path to various outputs or paths based on conditions that use the SSIS expressions.

Order: 1,2 3 etc

Output name: aliase name of splitted data

Condtion: this is where we can have filter condition like col>= or col== “kk” etc

## Data Conversion

Synchronous transformation is used for data conversion. It is a similar function to the Convert or Cast functions in T-SQL. It is a very useful transformation if we are pulling same data from multiple sources.

# Merge Join (like Inner join or Left Outer join)

SSIS merge join also combines data from different sources (Source of same type or different type (heterogeneous)) into single output or merged output. It uses merge concept based on (Inner, Left and full). We can use Merge Join based on specific condition like combining data on matching keys with that Inner, Left and full. Merge Join component accepts only 2 sorted (compulsory) inputs and one output and one error output.

Unlike Merge, Merge Join combines data depending on matching keys or string name.

There will be two source data

Both must be sorted on joining key

Merge join can be done on

Inner join

Left outer join --table can be swaped to keep one table in left side

Full outer join

Select columns list from both table as output column

# Merge (like UNION ALL)

SSIS merge is one of the component of SSIS, available in toolbox. SSIS merge works similar way to a SQL join it merges the two or more different sources (Sources can be of same type or different type / heterogeneous) into a single output. SSIS is all about collecting data from different sources, If you want to merge the collected data from different sources then we can use merge component. Merge component accepts only 2 sorted (compulsory) inputs. if there is more than 2 inputs then it best to use Union All transformation component. Also Merge transformations have only 1 output and does not have any Error output.

Input Requirements

* The Merge Transformation requires sorted data for its inputs.
* The Merge transformation also requires that the merged columns in its inputs have matching metadata. For example, you cannot merge a column that has a numeric data type with a column that has a character data type. If the data has a string data type, the length of the column in the second input must be less than or equal to the length of the column in the first input with which it is merged.
* In the SSIS Designer, the user interface for the Merge transformation automatically maps columns that have the same metadata. You can then manually map other columns that have compatible data types.

The Merge transformation is similar to the Union All transformations. Use the Union All transformation instead of the Merge transformation in the following situations:

* The transformation inputs are not sorted.
* The combined output does not need to be sorted.
* The transformation has more than two inputs.

# Lookup

The Lookup transformation tries to perform an equi-join between values in the transformation input and values in the reference dataset. (An equi-join means that each row in the transformation input must match at least one row from the reference dataset.) If an equi-join is not possible, the Lookup transformation takes one of the following actions:

* If there is no matching entry in the reference dataset, no join occurs. By default, the Lookup transformation treats rows without matching entries as errors. However, you can configure the Lookup transformation to redirect such rows to a no match output.
* If there are multiple matches in the reference table, the Lookup transformation returns only the first match returned by the lookup query. If multiple matches are found, the Lookup transformation generates an error or warning only when the transformation has been configured to load all the reference dataset into the cache. In this case, the Lookup transformation generates a warning when the transformation detects multiple matches as the transformation fills the cache.
* If there are multiple matches in the source table then all record from source table displayed
* The join can be a composite join, which means that you can join multiple columns in the transformation input to columns in the reference dataset. The transformation supports join columns with any data type

The Lookup transformation has the following inputs and outputs:

* Input.
* Match output. The match output handles the rows in the transformation input that match at least one entry in the reference dataset.
* No Match output. The no match output handles rows in the input that do not match at least one entry in the reference dataset. If you configure the Lookup transformation to treat the rows without matching entries as errors, the rows are redirected to the error output. Otherwise, the transformation would redirect those rows to the no match output.

Now it’s a time to use Term lookup transformation to find out the terms in Feedback source column, but the Term Lookup transformation will except only Unicode String [DT\_WSTR] or Unicode text stream [DT\_NTXT] as its data type . To convert the data type of the column we use data Conversion transformation.

Step1: add lookup transaformation

Step2: configure reference table

Step3: configure lookup column and reference column for matching

Step3: select reference column that you want to display along with all lookup columns

# Difference between Lookup and Merge Join

**Regarding Lookup:**

If you want to find rows matching in source 2 based on source 1 input and if you know there will be only one match for every input row, then I would suggest to use Lookup operation. An example would be you OrderDetails table and you want to find the matching Order Id and Customer Number, then Lookup is a better option.

**Regarding Merge Join:**

If you want to perform joins like fetching all Addresses (Home, Work, Other) from Address table for a given Customer in the Customer table, then you have to go with Merge Join because the customer can have 1 or more addresses associated with them.



# Fuzzy Lookup

Fuzzy Lookup matches input records that are "dirty" (because of misspellings, truncations, missing or inserted tokens, null fields, unexpected abbreviations, and other irregularities) with clean records in a reference table.

Fuzzy Lookup enables you to match input records with clean, standardized records in a reference table. The matching process is resilient to errors that are present in the input records. Fuzzy Lookup returns the closest match and indicates the quality of the match. For example, customer information (name and address) that is input during a new sales transaction may not match exactly with any record in the Customers reference table, which consists of all current customers, because of typographical or other errors in the input data. Fuzzy Lookup returns the best matching record from the Customers reference table even if no exact match exists, and provides measures to indicate the match quality.

# Fuzzy Group

The Fuzzy Grouping Transformation in [SSIS](https://www.tutorialgateway.org/ssis/) is used to replace the wrongly typed words with correct words. This transformation is same as [Fuzzy Lookup Transformation](https://www.tutorialgateway.org/fuzzy-lookup-transformation-in-ssis/) but Fuzzy Grouping Transformation does not require any reference table to correct the data. It will use the grouping technique to check for the wrongly typed words (type mistakes) and correct them. For example, if 98 people typed country name as India and 2 people types as Indi then Fuzzy Grouping Transformation will replace the Indi with India.

* This transformation uses Grouping technique to replace the wrong word in source data with the correct word.
* This transformation allows us to use only DT\_WSTR and DT\_STR Data type column for Fuzzy Matching and Exact matching can be applied to any data type except DT\_TEXT, DT\_NTEXT, and DT\_IMAGE.
* It creates temporary tables and indexes in the SQL Server database at runtime.
* This transformation will only use [OLE DB Connection Manager](https://www.tutorialgateway.org/ole-db-connection-manager-in-ssis/) to establish connection to store the temporary tables and indexes.
* To configure the transformation, you must select the Match Type (Fuzzy or Exact) for an input columns

Comparison Flags

From the below screenshot you can see the available options in this.

**Ignore case:**If we check mark this option then Fuzzy Grouping will ignore the case. Both XYZ and xyz will be same.

**Ignore kana type:**If we check mark this option then Fuzzy Grouping will ignore the difference between the hiragana and katakana in Japanese.

**Ignore non spacing characters:**If we check mark this option then Fuzzy Grouping will ignore the difference between the diacritics and character

**Ignore character width:** If we check mark this option then Fuzzy Grouping will ignore the difference between single-byte character and double-byte character.

**Ignore symbols:**If we check mark this option then Fuzzy Grouping will ignore the difference between the letters and symbols (white spaces, punctuations, currency symbols and mathematical symbols). For example, \*xy is treated same as xy

**Sort punctuation as symbols:** If we check mark this option then Fuzzy Grouping will all the punctuation symbols (except apostrophe and hyphen) write before the letters. For example .xyz will sort before the xyz

--

## ****Numerals****

In this option, We have to specify the significance of starting and ending numerals while comparing the column data. For example, if leading numerals are significant, “93 New lands Street” will not be grouped with “99 New lands Street”

|  |  |
| --- | --- |
| **Value** | **Description** |
| Neither | Numerals at the starting and ending position will be ignored while grouping. |
| Leading | Numerals at the starting position will be ignored while grouping. |
| Trailing | Numerals at the ending position will be ignored while grouping. |
| LeadingAndTrailing | Numerals at the starting and ending position will not be ignored while grouping. |

Within the Advanced Tab we have to configure the Similarity Threshold. Similarity threshold ranges between 0 and 1 where 1 is exact match. The Fuzzy Grouping Transformation Editor provides slider to adjust the similarity between 0 and 1. If similarity threshold is closer to 1 then source column should match more accurately to reference data. In this example, We are doing Fuzzy Grouping on Country Name and find the fuzzy match. If we give Similarity threshold as 0.76 the string column values should match more than 76% then only it will treat as a valid record.

# Pivot

PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output, and performs aggregations where they are required on any remaining column values that are wanted in the final output

Step1:Click on pivot

Step2: go to “input columns” select which all input columns are required for processing

Step3: Go to “input and output properties”>>pivot default input>>custome properties>>pivot usage as below

Enter PivoteUsage corresponding to each column as per below

|  |  |
| --- | --- |
| **OPTIONS** | **DESCRIPTION** |
| **0** | Column values will pass through the transformation |
| **1** | Set key. All the columns with the same set key |
| **2** | This is a **Pivot column** (These Column values become the Column names) |
| **3** | **(Aggregation** )Values from this column will be placed in the new columns created by the Pivot Transformation |

Notice that you must have at least one input column with PivotUsage 2, one input column with PivotUsage 3, one input column with PivotUsage 0 OR 1 at least in your pivot transformation.

After setting input columns, go to Pivot Default output, and under output columns

**Add columns**: ProductID, Name, 2005, 2006, 2007, 2008 using **Add Column** button

Enter **PivoteKeyValue** and **sourcecolumn**  as per below guidance

Lineageid of input column=sourcecolumn (if it is direct output)s

Lineageid of input column(aggregation column)=sourcecolumn (if it is derrived )

# Recordset Destination

The Recordset destination does not save data to an external data source. Instead, the Recordset destination saves data in memory in a recordset that is stored in an Integration Services package variable of the **Object** data type.

After the Recordset destination saves the data, you typically use a Foreach Loop container with the Foreach ADO enumerator to process one row of the recordset at a time. The Foreach ADO enumerator saves the value from each column of the current row into a separate package variable. Then, the tasks that you configure inside the Foreach Loop container read those values from the variables and perform some action with them.

# Data Type in SSIS

|  |  |
| --- | --- |
| SQL Server data type | SSIS data type |
| INT | DT\_I4 (four-byte signed integer)--int32 |
| NVARCHAR(50) | DT\_WSTR (Unicode string) |
| BIT | DT\_BOOL (Boolean) |
| SMALLINT | DT\_I2 (two-byte signed integer)--int16 |
| MONEY | DT\_CY (currency)--int64 |
| DECIMAL(8,2) | DT\_NUMERIC (numeric) |
| NCHAR(2) | DT\_WSTR (Unicode string) |
| DATETIME | DT\_DBTIMESTAMP (database timestamp) |
| DATE | DT\_DBDATE (database date) |
| UNIQUEIDENTIFIER | DT\_GUID (unique identifier) |