## SYLLABUS

- Introduction to Data Quality
    - Installing and setting up the DQS
    - Creating a Knowledge Base
- Using Data Quality Service to cleanse data
- Using Data Quality Service to match data
- Using Scripts in SSIS
    - Script Task
        - The Script Task Dts Object
    - Script Component
- Using Custom components in SSIS

## Table of Contents

## 1. Introduction to Data Quality

Why is data quality important? It is easy to answer this simple question. Say you can't trust the data in your data warehouse, perhaps because some data is incorrect (accuracy), because some data is missing (completeness), because some data is not in sync with other data (consistency), because some data is only a rough estimate (precision), or because some data is out-of-date (timeliness). In these cases, would it still be possible for you to use the data warehouse? If you use the data for making important decisions and the data is incorrect, how much will the mistake cost the company? And how bad will it be for your career?

Further, if you cannot trust the data in your data warehouse, what is the point of having a data warehouse? To prevent this disaster, when building a data warehouse, it is important to think about data quality as early as possible, preferably at the beginning of the project. The essence of maintaining data quality is to prevent bad data from getting into the data warehouse and to fix the bad data as early as possible in the process chain (ideally at the source system). To implement this, we need to set up rules that define what "bad data" is and put additional components in the architecture to filter the bad data out and to enable reporting, monitoring, and cleaning mechanisms.

## 2. Data Quality Process

The data quality process includes the activities to make sure the data in the data warehouse is correct and complete. This is usually done by checking the data on its way into the data warehouse. The data quality process also includes the mechanism to report the bad data and to correct it. So, there are three aspects of the data quality process: checking, reporting, and correcting.

For example, in the Amadeus Entertainment case study, customers can purchase a product, or they can subscribe to a package. The date a customer subscribes to a package for the first time is called the first subscription date, and the most recent date they canceled their subscriptions is called the last cancellation date. Suppose one day the ETL process extracted a customer record with the last cancellation date earlier than the first subscription date. This is not a valid condition. Either the last cancellation date is wrong or the first subscription date is wrong, or both. The data quality process detects this condition and reports it to the people responsible for the subscription data. They then correct the data in the source system, and the data is loaded into the data warehouse.

To understand the mechanism of how the data quality process works, look at Figure 1. It shows the ODS + DDS architecture discussed in Unit 1, with the data quality components added.

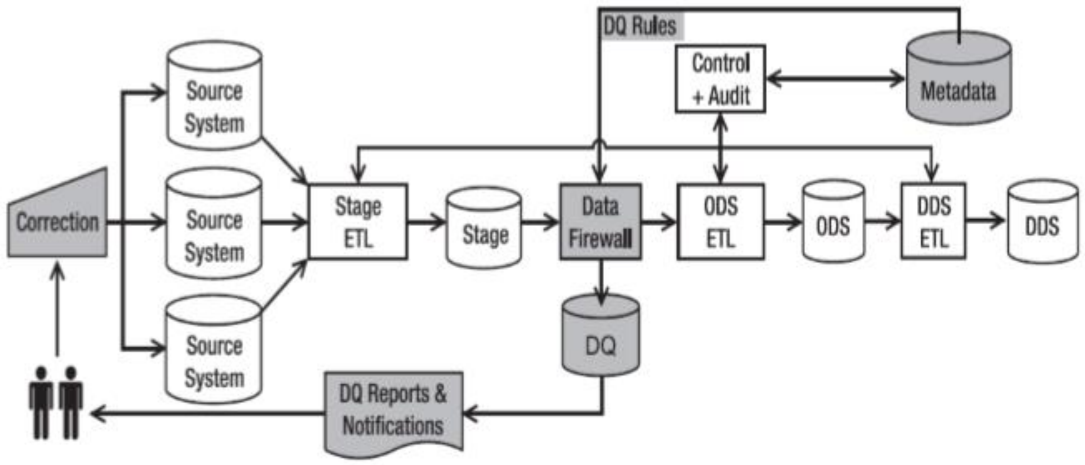


Figure 1 - Data quality components in the data warehouse architecture

The DQ components in Figure 1 are shaded and are as follows:

• The data firewall is a program that checks the incoming data. Physically it is an SSIS package or a stored procedure.

• Metadata is a database that stores the data quality rules, such as “The last cancellation date must be greater than the first subscription date.”

• The DQ database stores the bad data detected by the data firewall.

• DQ reports and notifications read the DQ database and inform the people responsible for data quality.

• Correction is a process to correct the data in the source system.

## 3. Install Data Quality Services

SQL Server Data Quality Services (DQS) contains the following two components: **Data Quality Server** and **Data Quality Client**.

**I NST AL L D ATA Q U ALI TY SE R VIC ES**

| DQS Component | Description |
|---|---|
| **Data Quality Server** | Data Quality Server is installed on top of the SQL Server Database Engine, and includes three databases: DQS_MAIN, DQS_PROJECTS, and DQS_STAGING_DATA. DQS_MAIN contains DQS stored procedures, the DQS engine, and published knowledge bases. DQS_PROJECTS contains the data quality project information. DQS_STAGING_DATA is the staging area where you can copy your source data to perform DQS operations, and then export your processed data. |
| **Data Quality Client** | Data Quality Client is a standalone application that enables you to connect to Data Quality Server, and provides you with a highly-intuitive graphical user interface to perform data quality operations, and other administrative tasks related to DQS. |

DQS installation is a three-part process:

**3.1 Pre-Installation Tasks**: Verify system requirements before you install DQS.

**3.2 Data Quality Services Installation Tasks:** Install DQS by using SQL Server Setup.

**3.3 Post-Installation Tasks**: Perform these tasks after finishing with the SQL Server Setup to finish installing DQS.

### 3.1 Pre-Installation Tasks

Before installing DQS, make sure that your computer meets the minimum system requirements. The following table provides information about the minimum system requirements for the DQS components:

| PR E-IN ST ALL ATI ON T AS KS | |
|---|---|
| **DQS Component** | **Minimum System Requirements** |
| Data Quality Server | Memory (RAM): Minimum: 2 GB / Recommended: 4 GB or more SQL Server Database Engine. For more information, see Install SQL Server Database Engine. |
| Data Quality Client | .NET Framework 4.0 (installed during the Data Quality Client installation, if not already installed)<br><br>Internet Explorer 6.0 SP1 or later |

### 3.2 Data Quality Services Installation Tasks

You have to use the SQL Server Setup to install DQS components. When you run the SQL Server Setup, you have to go through a series of installation wizard pages to select appropriate options based on your requirements. The following table lists only those pages in the installation wizard where the options that you select will affect your installation of DQS:

| DATA QUALITY SERVICES INSTALLATION TASKS | |
|---|---|
| **Page** | **Action** |
| Feature Selection | Select: **Data Quality Services** under **Database Engine Services** to install the Data Quality Server. If you select the **Data Quality Services** check box, SQL Server Setup will copy an installer file, DQSInstaller.exe, under the SQL Server instance directory on your computer. You must run this file after you have completed the SQL Server Setup to *complete* the Data Quality Server installation. Also, you must perform some additional steps to configure your Data Quality Server before you can use it. For more information, see Post-Installation Tasks.<br>**Data Quality Client** to install Data Quality Client. (Recommended) **Management Tools - Complete** under **Management Tools - Basic** to install SQL Server Management Studio. It provides you a graphical user interface to manage your SQL Server instance, and will aid you in performing additional tasks post installation as listed in the next section. |
| Database Engine Configuration | Click **Add Current User** to add your user Windows account to the sysadmin fixed server role. This is required for you to be able to run the DQSInstaller.exe file later to complete the Data Quality Server installation. |

### 3.3 Post-Installation Tasks

After you complete the SQL Server installation wizard, you must perform additional steps mentioned in this section to complete your Data Quality Server installation, configure it, and then use it.

1. To complete the Data Quality Server installation, run the DQSInstaller.exe file. On running the DQSInstaller.exe file:

o The DQS_MAIN, DQS_PROJECTS, and DQS_STAGING_DATA databases are created.

o The ##MS_dqs_db_owner_login## and ##MS_dqs_service_login## logins are created.

o The dqs_administrator, dqs_kb_editor, and dqs_kb_operator roles are created in the DQS_MAIN database.

o The DQInitDQS_MAIN stored procedure is created in the master database.

o DQS_install.log file is typically created in the C:\Program Files\Microsoft SQL Server\MSSQL13.<*instance_name*>\MSSQL\Log folder. The file contains information about the actions performed on running the DQSInstaller.exe file.

o If a Master Data Services database is present in the same SQL Server instance as Data Quality Server, a user mapped to the Master Data Services login is created, and is granted the dqs_administrator role on the DQS_MAIN database.

This completes the Data Quality Server installation.

For more information see **Run DQSInstaller.exe to Complete Data Quality Server Installation.** 2. Grant DQS

Roles to Users:

To log on to Data Quality Server using Data Quality Client, a user must have any of the following three roles on the DQS_MAIN database:

o **dqs_administrator**

o **dqs_kb_editor**

o **dqs_kb_operator**

By default, if your user account is a member of the sysadmin fixed server role, you can log on to Data Quality Server using Data Quality Client even if none of the DQS roles are granted to your user account. For information about the three DQS roles, see DQS Security.

3. Make your data available for DQS operations. Make sure that you can access your source data for the DQS operations, and can export the processed data to a table in a database.

## 4. DQS Knowledge Bases

This topic describes what a knowledge base is in Data Quality Services (DQS). To cleanse data, you have to have knowledge about the data. To prepare knowledge for a data quality project, you build and maintain a knowledge base (KB) that DQS can use to identify incorrect or invalid data. DQS enables you to use both computer-assisted and interactive processes to create, build, and update your knowledge base. Knowledge in a knowledge base is maintained in domains, each of which is specific to a data field. The knowledge base is a repository of knowledge about your data that enables you to understand your data and maintain its integrity.

DQS knowledge bases have the following benefits:

· Building knowledge about data is a detailed process. The DQS process of extracting knowledge about data automatically, from sample data, makes the process much easier.

· DQS enables you to see its analysis of the data, and to augment the knowledge in the knowledge base by creating rules and changing data values. You can do so repeatedly to improve the knowledge over time.

· You can leverage pre-existing data quality knowledge by basing a knowledge base on an existing KB, importing domain knowledge from files into the KB, importing knowledge from a project back into a KB, or using the DQS default KB, DQS Data.

· You can ensure the quality of your data by comparing it to the data maintained by a reference data provider.

· there is a clear separation between building a knowledge base and applying it in the data correction process, which gives you flexibility in how you build and update the knowledge base.

**How to Create and Build a DQS Knowledge Base**

Building a DQS knowledge base involves the following processes and components: **Knowledge Discovery**

A computer-assisted process that builds knowledge into a knowledge base by processing a data sample

## 5. Domain Management

An interactive process that enables the data steward to verify and modify the knowledge that is in knowledge base domains, each of which is associated with a data field. This can include setting field wide properties, creating rules, changing specific values, using reference data services, or setting up term based or cross-field relationships.

**Reference Data Services**
A process of domain management that enables you to validate your data against data maintained and guaranteed by a reference data provider.

**Matching Policy**

A policy that defines how DQS processes records to identify potential duplicates and non-matches, built into the knowledge base in a computer-assisted and interactive process.

## 6. Using Data Quality Service to cleanse data and Match data

What is data cleansing? Why is it important in data quality? Put simply, data cleansing, or data scrubbing, is the process of identifying and correcting dirty data. Dirty data means incomplete, wrong, duplicate, or out-of-date data. I’m not sure why, but people in the data warehousing community prefer to use the words cleanse and scrub rather than clean, which is simpler and more descriptive. An example of data cleansing is checking stores’ tables to make sure the store names, store numbers, store types, and store addresses are all correct.

Other examples are making sure that there are no duplicate customer records that the price lists are correct, that obsolete products are flagged accordingly, that all subscriptions refer to existing packages, and that the song and film titles are correctly spelled.

In data cleansing, it is important to be able to determine that one data item is the same as another data item. This is called data matching. Data matching is used to identify duplicate records when doing a lookup on reference data. Matching is particularly relevant for character-based data types, including large value data types, such as varchar(max), because for numerical or date time data types, we can simply use the equal sign. For character-based data types, it may not be that clear. For example, when loading the address table in the NDS, we may find that the city name is “Los Angles,” which does not exist in the city table. In this case, we need to match “Los Angles” to “Los Angeles.” Another example is the customer name “Robert Peterson.” We need to match/recognize that “Robert Peterson” is the same as “Bob Peterson.”

Numeric data is not as tricky as character data. We can just use the equal sign; 5 is 5 whatever way we look at it, and 5 is different from 6. So, we can just use = for matching numeric data. For example, “if A = B, then….” The only possible problem is rounding; for example, is 5.029 the same as 5.03? If the precision is two decimal digits, they are the same, but if the precision is three decimal digits, then they are different. For date and time data, if the data is stored as a date time data type, it’s not a problem.

We can just use the equal sign like the numeric data. But if it is stored as a character data type, it could be a little bit tricky because of the date format and time zone. For example, is 03/01/2008 the same as 01/03/2008? Is it the same as 2007-03-01T00:00:00Z+06? We need a little bit of logic here to match them, such as by comparing the components or by comparing the date, month, and year.

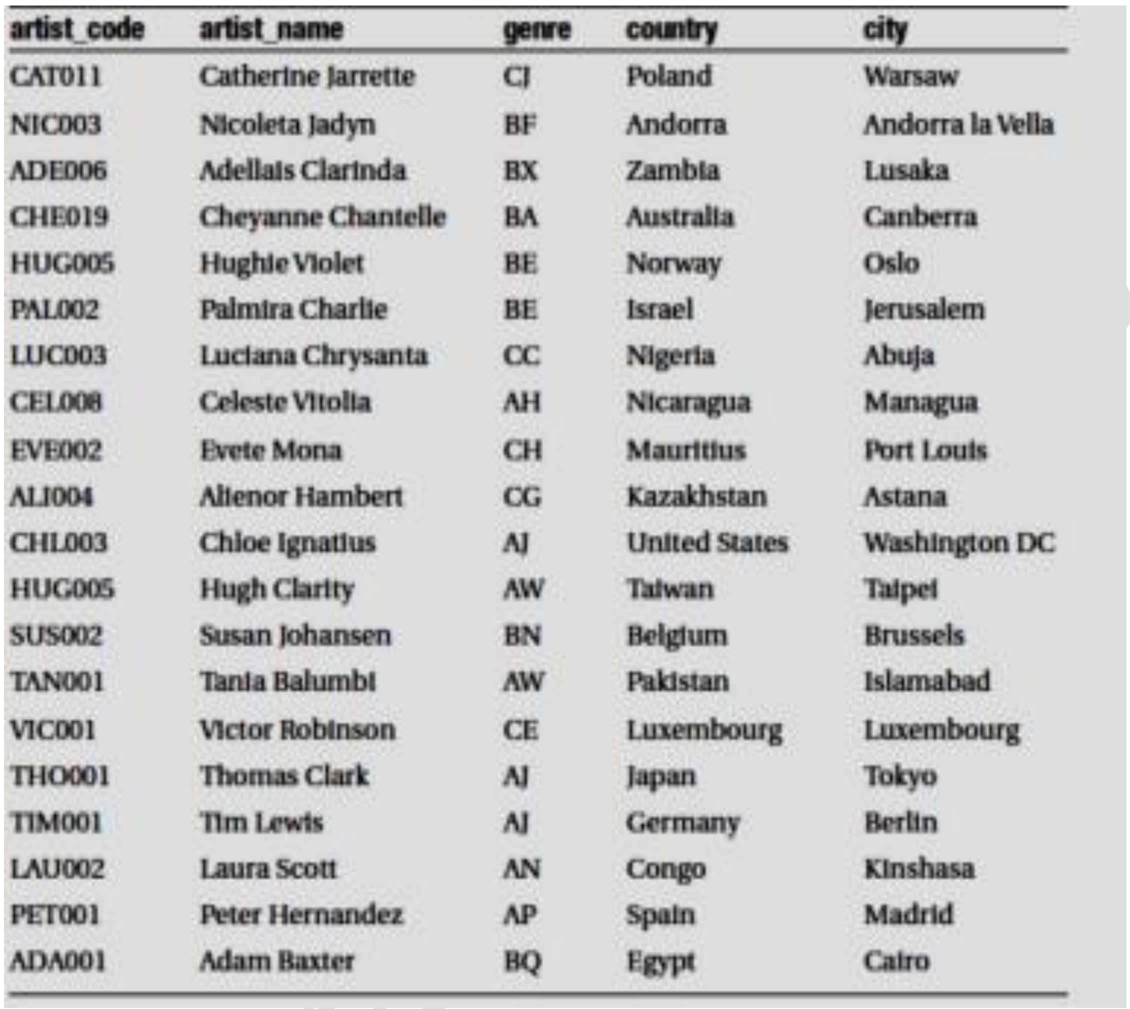
In SQL Server we have three types of matching logic: exact, fuzzy (approximate), and rule based. Exact matching is where all characters are the same, for example “Los Angeles” and “Los Angeles.” In SSIS this is done using a Lookup transformation. Fuzzy logic matching finds how similar a set of data is to another set of data. For example, using the Fuzzy Lookup transformation in SSIS, “You can’t hurry love” and “You cannot hurry love” have a similarity score of 0.81666672 and a confidence level of 0.59414238. You can then decide for example that if the similarity score is greater than 0.75 and the confidence level is greater than 0.5, then it’s a match.

To determine the threshold for the similarity score and confidence level (0.75 and 0.5 in the previous case), we need to run the data through a Fuzzy Lookup transformation with the similarity threshold set to zero. This will output the similarity scores for all the rows in the source data. We then manually examine the score to find the appropriate threshold. We will do this (finding the threshold) later in this section.

Rule-based logic is where we use certain rules and data to identify a match. For example, we can define (in a table) that for name data the string “Bill” is the same as “William” or in product names “movie” is the same as “film.” In SSIS this is implemented with database lookup. We can also define a rule, such as “For product code, omit the spaces when comparing” so that “KL 7923 M” is the same as “KL7923M.” Here’s another example: “Omit prefix 9000 in the supplier number” so that “900089123” and “89123” are the same. SSIS logic like the previous can be implemented with Script Component.

Let’s open Business Intelligence Development Studio (BIDS) and construct a fuzzy logic and a rule based logic solution. As they say, the devil is in the details. And without trying it, we wouldn’t know the details. In this scenario, we have a table in Jade called artist2. The table contains 20 artists that we are going to load into NDS. The artist2 table also contains city and country, as shown in Table 1.

Table 1 - artist2 Table in the Jade Source System



| artist_code | artist_name | genre | country | city |
|---|---|---|---|---|
| CAT011 | Catherine Jarrette | CJ | Poland | Warsaw |
| NIC003 | Nicoleta Jadyn | BF | Andorra | Andorra la Vella |
| ADE006 | Adellais Clarinda | BX | Zambia | Lusaka |
| CHE019 | Cheyanne Chantelle | BA | Australia | Canberra |
| HUG005 | Hughie Violet | BE | Norway | Oslo |
| PAL002 | Palmira Charlie | BE | Israel | Jerusalem |
| LUC003 | Luciana Chrysanta | CC | Nigeria | Abuja |
| CEL008 | Celeste Vitolia | AH | Nicaragua | Managua |
| EVE002 | Evete Mona | CH | Mauritius | Port Louis |
| ALI004 | Alienor Hambert | CG | Kazakhstan | Astana |
| CHL003 | Chloe Ignatius | AJ | United States | Washington DC |
| HUG005 | Hugh Clarity | AW | Taiwan | Taipei |
| SUS002 | Susan Johansen | BN | Belgium | Brussels |
| TAN001 | Tania Balumbi | AW | Pakistan | Islamabad |
| VIC001 | Victor Robinson | CE | Luxembourg | Luxembourg |
| THO001 | Thomas Clark | AJ | Japan | Tokyo |
| TIM001 | Tim Lewis | AJ | Germany | Berlin |
| LAU002 | Laura Scott | AN | Congo | Kinshasa |
| PET001 | Peter Hernandez | AP | Spain | Madrid |
| ADA001 | Adam Baxter | BQ | Egypt | Cairo |

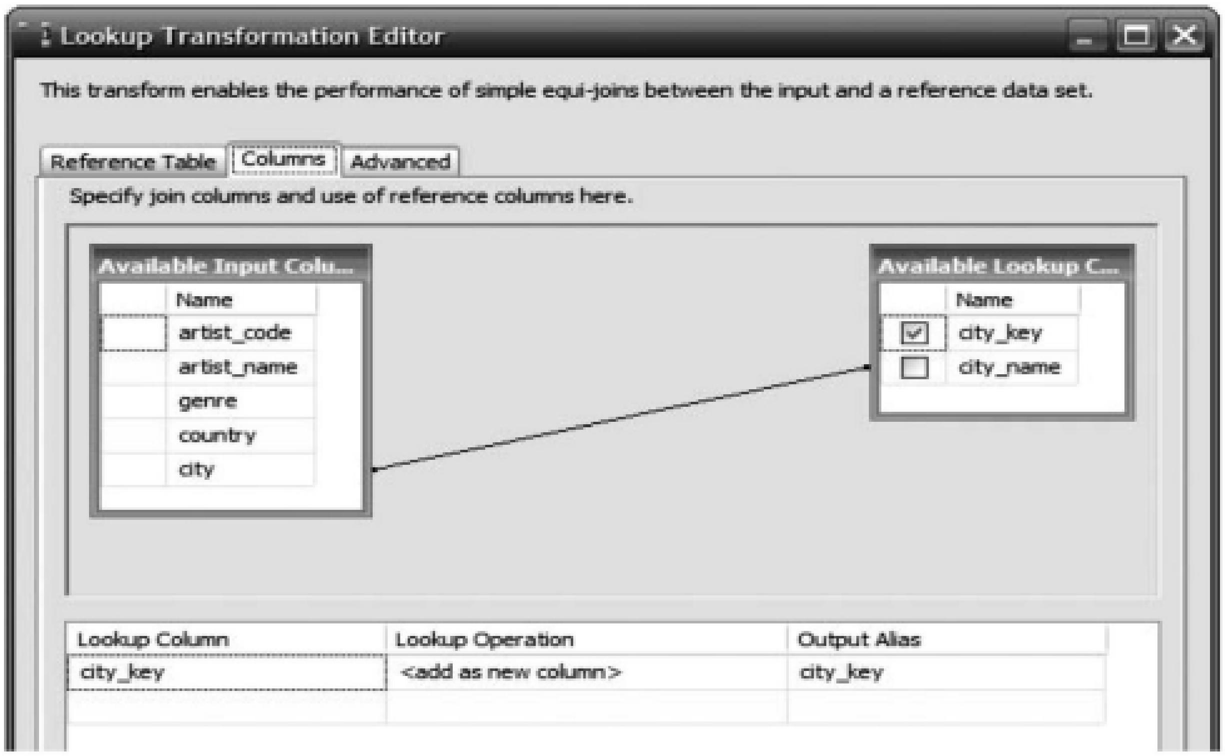
Out of these twenty artists, nine are new artists. Eleven of those twenty, however, actually already exist in the NDS, but their names are slightly different. For example, one that already exists in the NDS is "Catherine Jarrett," but the incoming data is "Catherine Jarrette."

We are going to use a Fuzzy Lookup transformation with certain similarity levels to determine whether the artists already exist in the NDS. If they exist, we update the NDS. If they don't exist, we will insert them into the NDS. In this case, we will find first what the confidence level should be by pointing the output of the Fuzzy Lookup transformation to a file and setting up a data viewer. The city and country need be translated into keys using normal lookup (exact match). As with every NDS table, we also need to populate the source_system_code, create_timestamp, and update_timestamp columns. We'll use a Derived Column transformation to populate these three columns. Using a Derived Column transformation, we can create new columns containing constant values or calculations based on the input columns.

**Using Scripts and Custom components in SSIS**

1. Define a new SSIS package, and call it NDS Artist.
2. Create a Data Flow task on the design surface, and call it Load Artist.
3. Double-click it to edit it.
4. Create an OLE DB connection, call it Jade Artist, and point this connection to the Jade database. 5. Set Data Access Mode to table or view and the name of the table to Artist2. Click Preview to check; there should be 20 rows. In the column pane, ensure that all columns are selected. 6. Create a Lookup transformation, name it Lookup City, and connect the green arrow from Jade Artist to it. Point the connection to the NDS, set the SQL As select city_key, city_name from city, and map the incoming city column to the city_name column on the lookup

table. Define one output column, and call it
city_key, as shown in Figure 3. Click OK to close Lookup City.



This transform enables the performance of simple equi-joins between the input and a reference data set.

Specify join columns and use of reference columns here.

| | Name |
|---|---|
| | artist_code |
| | artist_name |
| | genre |
| | country |
| | city |

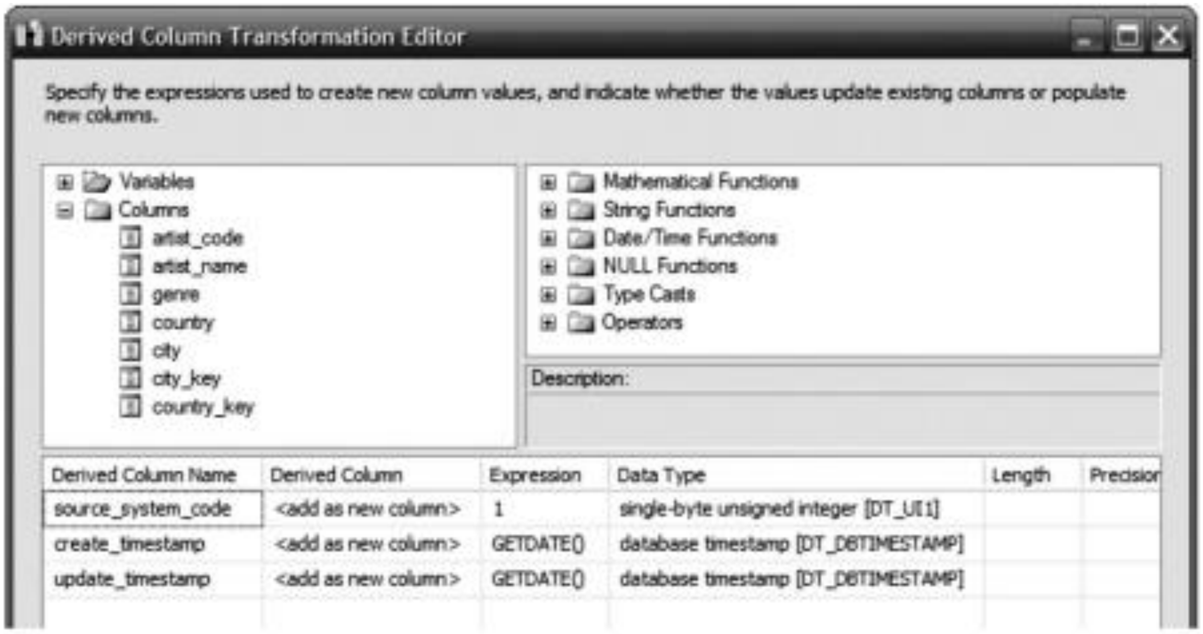
| | Name |
|---|---|
| ✓ | city_key |
| | city_name |

| Lookup Column | Lookup Operation | Output Alias |
|---|---|---|
| city_key | <add as new column> | city_key |

**Figure 3 - Looking up the city key using a normal lookup**

7. Repeat step 6 for country. When doing a lookup like this, you can (you should, really) redirect the error output to a file for further processing (for example, to prevent the next step from loading the data into the DDS or for analyzing the failed data) as well as for auditing purposes.

8. After you are done with country, create a Derived Column transformation. This is to add the source_system_code, create_timestamp, and update_timestamp columns. Name the transformation Source Code & Timestamps, and configure it as per Figure 4.


Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

| Derived Column Name | Derived Column | Expression | Data Type | Length | Precision |
|---|---|---|---|---|---|
| source_system_code | <add as new column> | 1 | single-byte unsigned integer [DT_UI1] | | |
| create_timestamp | <add as new column> | GETDATE() | database timestamp [DT_DBTIMESTAMP] | | |
| update_timestamp | <add as new column> | GETDATE() | database timestamp [DT_DBTIMESTAMP] | | |

**Figure 4. Using derived column to define source system code and timestamps**

9. The resulting data flow will be similar to the one in Figure 5. The last two boxes in Figure 5 (Lookup Artist Names and Flat File Destination) are not done yet. We'll do them next.

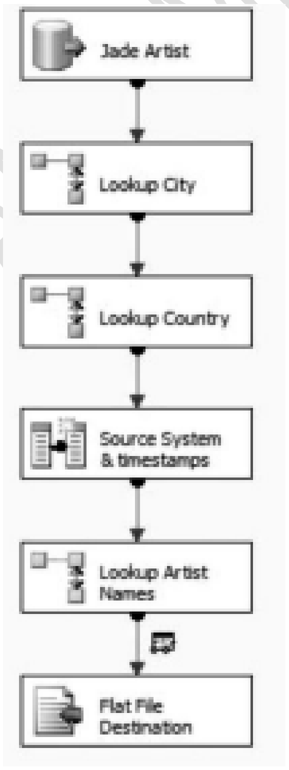
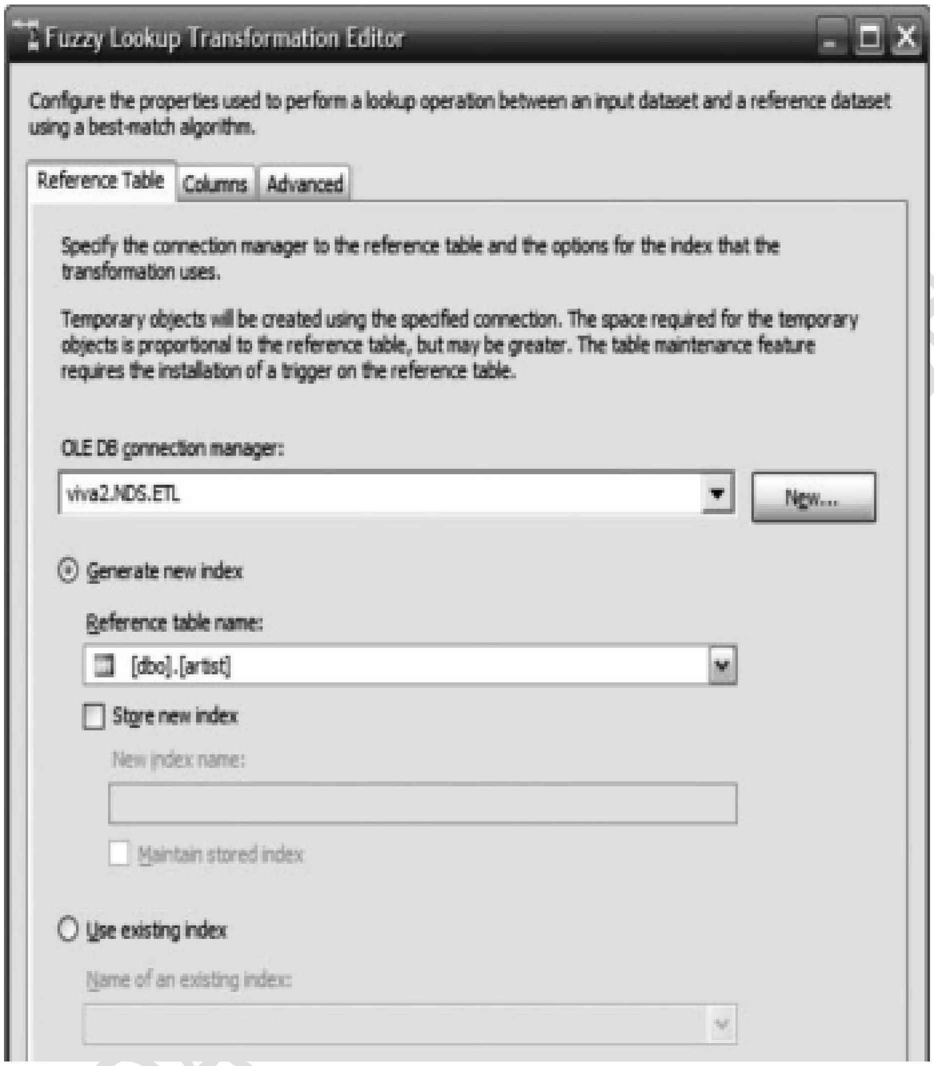


**Figure 5 - Data flow for loading the artist2 table using fuzzy lookup**

10. In the Toolbox, find the Fuzzy Lookup transformation, and drag it to the design surface. Connect the green arrow from Source Code & Timestamp to it, and call it Lookup Artist Names. Double click it to edit it, set the connection to NDS, and choose artist as the reference table, as shown in Figure 6.


Fuzzy Lookup Transformation Editor

Configure the properties used to perform a lookup operation between an input dataset and a reference dataset using a best-match algorithm.

Reference Table | Columns | Advanced

Specify the connection manager to the reference table and the options for the index that the transformation uses.

Temporary objects will be created using the specified connection. The space required for the temporary objects is proportional to the reference table, but may be greater. The table maintenance feature requires the installation of a trigger on the reference table.

OLE DB connection manager: viva2.NDS.ETL

New...

[✓ Generate new index]

Reference table name: [dbo].[artist]

☐ Store new index

New index name:

☐ Maintain stored index

○ Use existing index

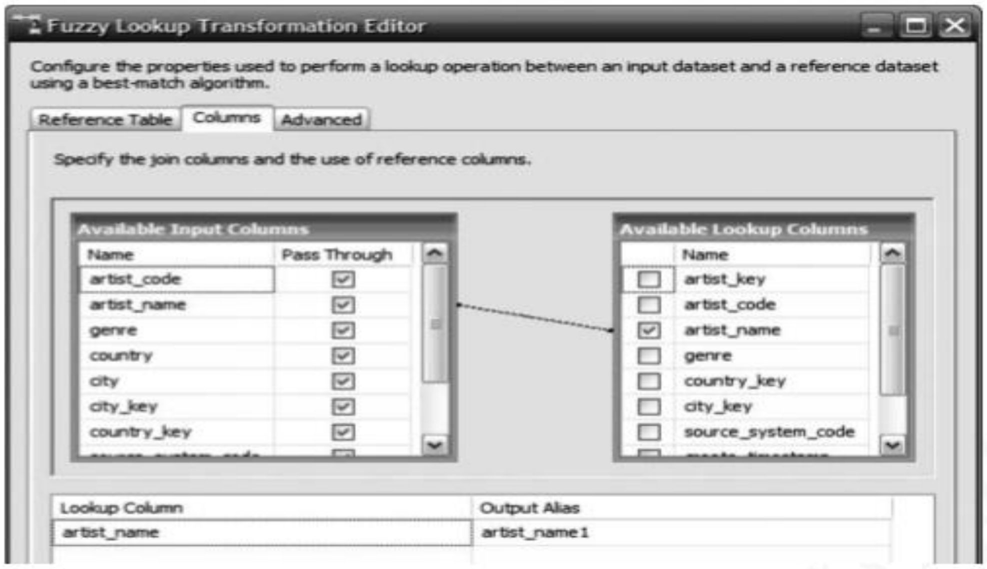
Name of an existing index:

**Figure 6 - Setting the reference table on the Fuzzy Lookup transformation**

11. Now we want to look up the artist record in the NDS (target) based on the artist name on the stage (source). For that, click the Columns tab. Ensure that all the Pass Through check boxes are checked so that the input columns are included in the output. Delete the artist_code and genre mappings, leaving only artist_name. This is because we want to do the lookup based only on the artist name. Check artist_name in the Available Lookup Columns list, and set Output Alias to artist_name1, as shown in Figure 7.

12. Click the Advanced tab. Leave the maximum number of matches at 1. Leave the similarity threshold at 0, and leave the token delimiters as they are, as shown in Figure 8.

The purpose of doing this is so that we get the similarity score for all the rows in the table. Click OK to close the Fuzzy Lookup Transformation Editor dialog box.



Fuzzy Lookup Transformation Editor

Configure the properties used to perform a lookup operation between an input dataset and a reference dataset using a best-match algorithm.

Reference Table | Columns | Advanced

Specify the join columns and the use of reference columns.

Available Input Columns

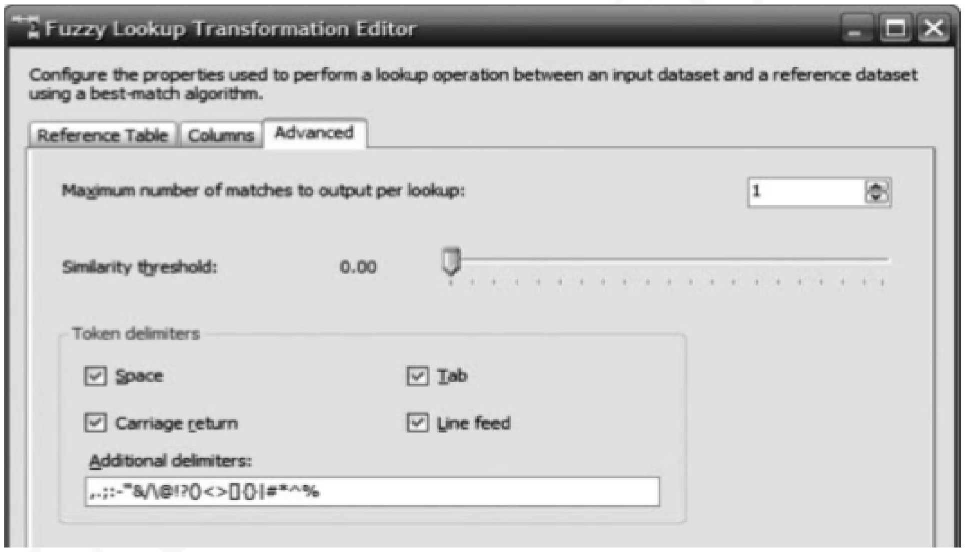
| Name | Pass Through |
|---|---|
| artist_code | ✓ |
| artist_name | ✓ |
| genre | ✓ |
| country | ✓ |
| city | ✓ |
| city_key | ✓ |
| country_key | ✓ |

Available Lookup Columns

| | Name |
|---|---|
| ☐ | artist_key |
| ☐ | artist_code |
| ✓ | artist_name |
| ☐ | genre |
| ☐ | country_key |
| ☐ | city_key |
| ☐ | source_system_code |

| Lookup Column | Output Alias |
|---|---|
| artist_name | artist_name1 |

**Figure 7 - Configuring the lookup columns on the Fuzzy Lookup transformation**



Fuzzy Lookup Transformation Editor

Configure the properties used to perform a lookup operation between an input dataset and a reference dataset using a best-match algorithm.

Reference Table | Columns | Advanced

Maximum number of matches to output per lookup: 1

Similarity threshold: 0.00

Token delimiters

[✓] Space [✓] Tab

[✓] Carriage return [✓] Line feed

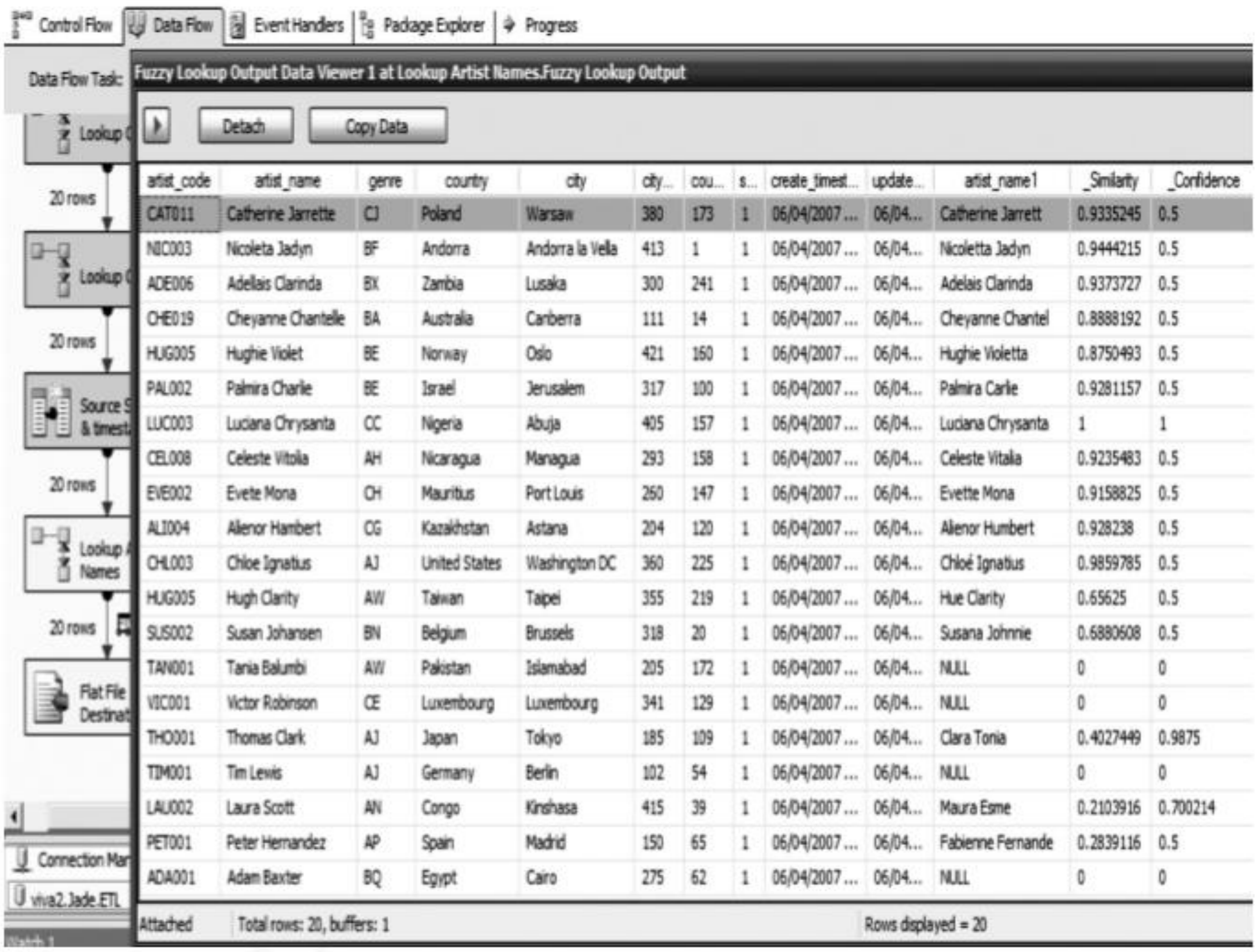
Additional delimiters:

,.;:-"&/\@!?()<>[]{}|#*^%

**Figure 8 - Advanced tab of the Fuzzy Lookup transformation**

13. Now let's add a file destination and a data viewer so we can see and examine the data, along with the similarity score and confidence level. Create a flat-file destination. Point it to a new folder called Output in your ETL directory. Set the format to Delimited. In the column pane, set the column delimiter to a vertical bar (|). Leave everything else as is, and click OK. Right-click the green connector between the Fuzzy Lookup and the Flat File destination, and select Data Viewers. Click Add, and choose Grid. Click the Grid tab, and select all columns. Click OK twice to return to the Data Flow transformation.

14. Run the package, and it will stop at the data viewer. The data viewer shows the similarity and confidence columns, as shown in Figure 9.



| artist_code | artist_name | genre | country | city | city... | cou... | s... | create_timest... | update... | artist_name1 | _Similarity | _Confidence |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAT011 | Catherine Jarrette | CJ | Poland | Warsaw | 380 | 173 | 1 | 06/04/2007 ... | 06/04... | Catherine Jarrett | 0.9335245 | 0.5 |
| NIC003 | Nicoleta Jadyn | BF | Andorra | Andorra la Vella | 413 | 1 | 1 | 06/04/2007 ... | 06/04... | Nicoletta Jadyn | 0.9444215 | 0.5 |
| ADE006 | Adelais Clarinda | BX | Zambia | Lusaka | 300 | 241 | 1 | 06/04/2007 ... | 06/04... | Adelais Clarinda | 0.9373727 | 0.5 |
| CHE019 | Cheyanne Chantelle | BA | Australia | Canberra | 111 | 14 | 1 | 06/04/2007 ... | 06/04... | Cheyanne Chantel | 0.8888192 | 0.5 |
| HUG005 | Hughie Violet | BE | Norway | Oslo | 421 | 160 | 1 | 06/04/2007 ... | 06/04... | Hughie Violetta | 0.8750493 | 0.5 |
| PAL002 | Palmira Charlie | BE | Israel | Jerusalem | 317 | 100 | 1 | 06/04/2007 ... | 06/04... | Palmira Carlie | 0.9281157 | 0.5 |
| LUC003 | Luciana Chrysanta | CC | Nigeria | Abuja | 405 | 157 | 1 | 06/04/2007 ... | 06/04... | Luciana Chrysanta | 1 | 1 |
| CEL008 | Celeste Vitola | AH | Nicaragua | Managua | 293 | 158 | 1 | 06/04/2007 ... | 06/04... | Celeste Vitala | 0.9235483 | 0.5 |
| EVE002 | Evete Mona | CH | Mauritius | Port Louis | 260 | 147 | 1 | 06/04/2007 ... | 06/04... | Evette Mona | 0.9158825 | 0.5 |
| ALI004 | Alenor Hambert | CG | Kazakhstan | Astana | 204 | 120 | 1 | 06/04/2007 ... | 06/04... | Alenor Humbert | 0.928238 | 0.5 |
| CHL003 | Chloe Ignatius | AJ | United States | Washington DC | 360 | 225 | 1 | 06/04/2007 ... | 06/04... | Chloé Ignatius | 0.9859785 | 0.5 |
| HUG005 | Hugh Clarity | AW | Taiwan | Taipei | 355 | 219 | 1 | 06/04/2007 ... | 06/04... | Hue Clarity | 0.65625 | 0.5 |
| SUS002 | Susan Johansen | BN | Belgium | Brussels | 318 | 20 | 1 | 06/04/2007 ... | 06/04... | Susana Johnnie | 0.6880608 | 0.5 |
| TAN001 | Tania Balumbi | AW | Pakistan | Islamabad | 205 | 172 | 1 | 06/04/2007 ... | 06/04... | NULL | 0 | 0 |
| VIC001 | Victor Robinson | CE | Luxembourg | Luxembourg | 341 | 129 | 1 | 06/04/2007 ... | 06/04... | NULL | 0 | 0 |
| THO001 | Thomas Clark | AJ | Japan | Tokyo | 185 | 109 | 1 | 06/04/2007 ... | 06/04... | Clara Tonia | 0.4027449 | 0.9875 |
| TIM001 | Tim Lewis | AJ | Germany | Berlin | 102 | 54 | 1 | 06/04/2007 ... | 06/04... | NULL | 0 | 0 |
| LAU002 | Laura Scott | AN | Congo | Kinshasa | 415 | 39 | 1 | 06/04/2007 ... | 06/04... | Maura Esme | 0.2103916 | 0.700214 |
| PET001 | Peter Hernandez | AP | Spain | Madrid | 150 | 65 | 1 | 06/04/2007 ... | 06/04... | Fabienne Fernande | 0.2839116 | 0.5 |
| ADA001 | Adam Baxter | BQ | Egypt | Cairo | 275 | 62 | 1 | 06/04/2007 ... | 06/04... | NULL | 0 | 0 |

Attached | Total rows: 20, buffers: 1 | Rows displayed = 20

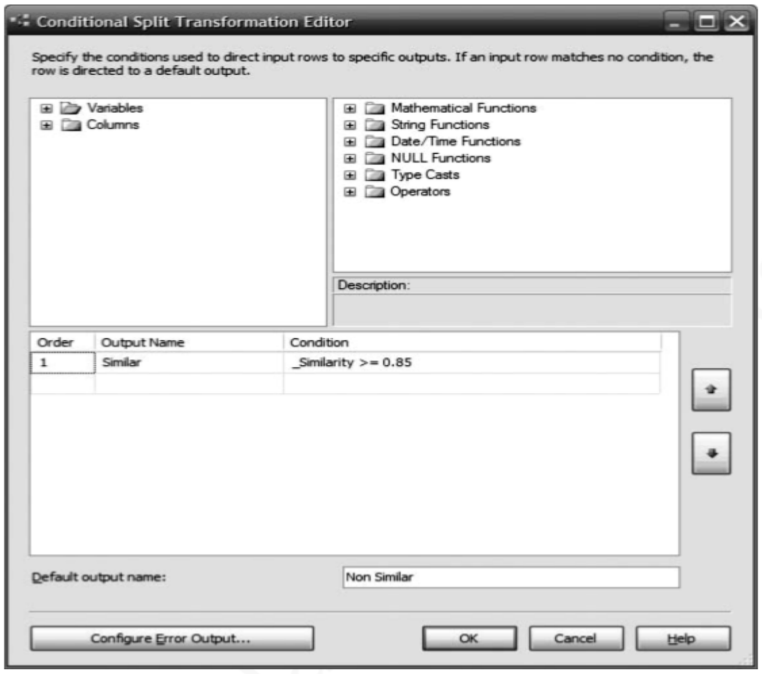
**Figure 9 - Data viewer showing the similarity and confidence columns4**

15. The data viewer shows that the similarity figures for the first 11 rows are all greater than 87 percent. This means the source data is similar to the target data in the NDS. In other words, we found a match for these 11 rows. You can compare the artist_name and artist_name1 columns to see the slight differences. artist_name is the incoming data, and artist_name1 is the data in the NDS artist table. However, the similarity for Hugh Clarity (NDS record Hue Clarity) is only 65 percent, about the same as Susan Johansen (NDS record Susana Johnnie), which gets 69 percent. Four gets 0 (none matching), two gets between 20 percent and 30 percent, and 1 gets 40 percent. This means that the Fuzzy Lookup transformation did not find similar records in the NDS. In other words, there are no matches for these records. Based on this, we can decide to Set the similarity level to about 85 percent because this threshold will allow good records to be loaded onto NDS, yet it will prevent the bad data from going through.

16. Click the green arrow in the top-left corner of the data viewer to continue executing the package. The package completes, and the flat file is populated. The first ten lines of the flat file are shown here:
artist_code|artist_name|genre|country|city|city_key|country_key|
source_system_code|create_timestamp|update_timestamp|
artist_name1|_Similarity|_Confidence

```
CAT011 |Catherine Jarrette|CJ|Poland|Warsaw|380|173|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Catherine Jarrett|0.93352449|0.5|0.93352449
NIC003 |Nicoleta Jadyn|BF|Andorra|Andorra la Vella|413|1|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Nicoletta Jadyn|0.94442153|0.5|0.94442153
ADE006 |Adellais Clarinda|BX|Zambia|Lusaka|300|241|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Adelais Clarinda|0.93737274|0.5|0.93737274
CHE019 |Cheyanne Chantelle|BA|Australia|Canberra|111|14|1|
2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Cheyanne Chantel|0.88881922|0.5|0.88881922
HUG005 |Hughie Violet|BE|Norway|Oslo|421|160|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Hughie Violetta|0.87504929|0.5|0.87504929
PAL002 |Palmira Charlie|BE|Israel|Jerusalem|317|100|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Palmira Carlie|0.92811567|0.5|0.92811567
LUC003 |Luciana Chrysanta|CC|Nigeria|Abuja|405|157|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Luciana Chrysanta|1|1|1
CEL008 |Celeste Vitolia|AH|Nicaragua|Managua|293|158|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Celeste Vitalia|0.92354828|0.5|0.92354828
EVE002 |Evete Mona|CH|Mauritius|Port Louis|260|147|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Evette Mona|0.91588253|0.5|0.91588253
ALI004 |Alienor Hambert|CG|Kazakhstan|Astana|204|120|
1|2007-04-06 22:30:42.546000000|2007-04-06 22:30:42.546000000|
Alienor Humbert|0.92823803|0.5|0.92823803
```

17. Press Shift+F5 to stop the package and return to the designer. Now let's delete the Flat File destination because we want to direct the output of the Fuzzy Lookup transformation to the NDS. To load the output into the NDS, we need to split the output into two depending on whether the artist record already exists in the NDS. If the record already exists, we update it. If it does not exist, we insert it. For that let's add a Conditional Split transformation and connect the green arrow from Fuzzy Lookup to it. Name the box Split Based on Similarity, and double-click it to edit it. Under Output Name, type Similar, and under Condition, type [_Similarity]>=0.85, as shown in Figure 10.



Conditional Split Transformation Editor

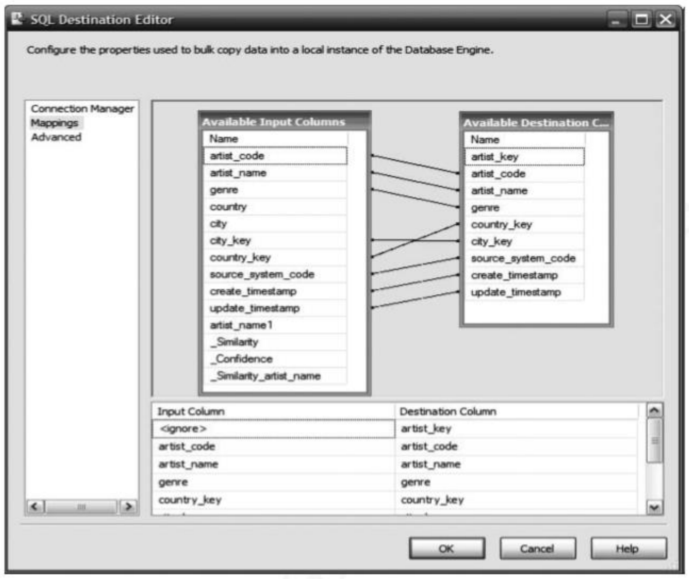
Specify the conditions used to direct input rows to specific outputs. If an input row matches no condition, the row is directed to a default output.

| Order | Output Name | Condition |
|---|---|---|
| 1 | Similar | _Similarity >= 0.85 |

Default output name: Non Similar

Figure 10 - Splitting the output of the Fuzzy Lookup transformation based on similarity

18. Now that we have split the output into two branches, we need to feed both branches into two separate destinations. The first destination is for insert, and the second one is for update. If the artist name is the same or similar to the ones in the target (the NDS), we want to update the target. On the other hand, if the artist name does not exist (Not Similar), we want to insert (create) that artist into the NDS. For that, let's set Default Output Name to Not Similar and click OK. Now we need to configure the insert and update. Let's do the insert first. Drag an SQL Server destination onto the design surface, and name it Insert to NDS Artist. Connect the green arrow from the Split Based on Similarity box, and the Input Output Selection will pop up. In the Output dropdown list, select Not Similar, and click OK. Double-click the SQL Server destination, and set the connection to NDS. Point it to the artist table, and configure the column mappings as per Figure 11.



SQL Destination Editor

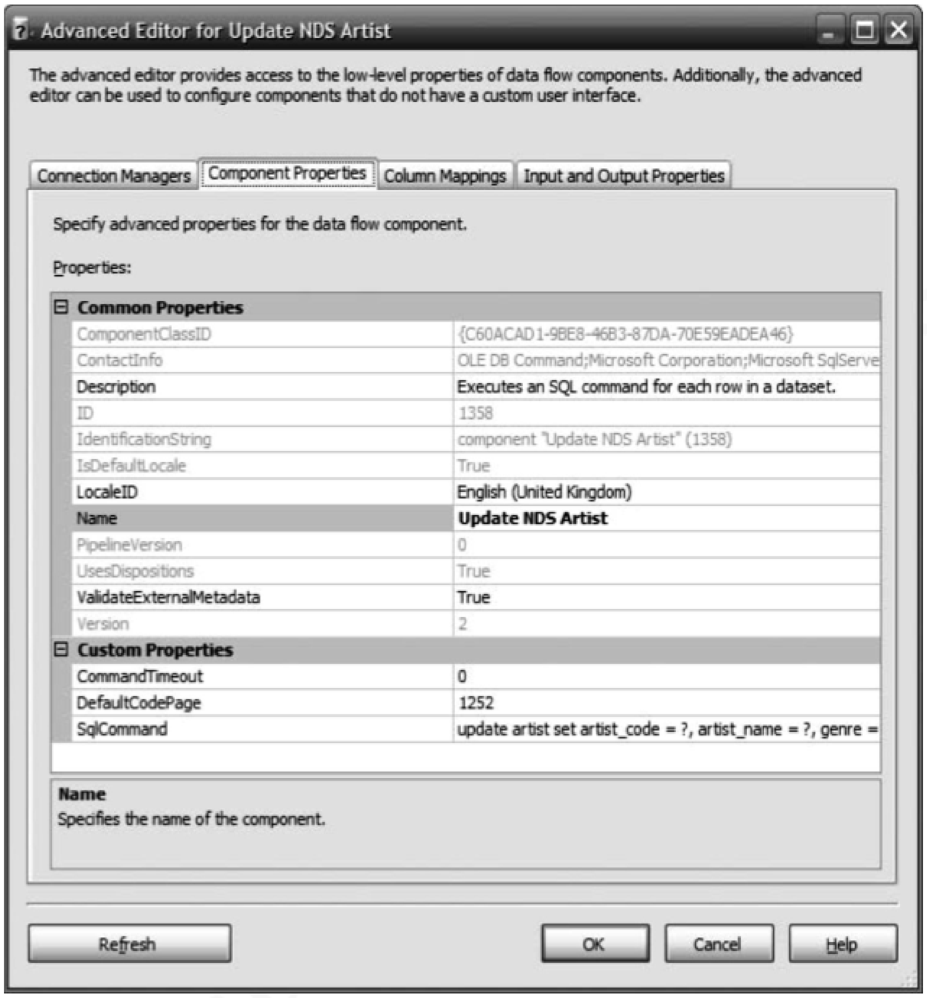
Configure the properties used to bulk copy data into a local instance of the Database Engine.

| Input Column | Destination Column |
|---|---|
| <ignore> | artist_key |
| artist_code | artist_code |
| artist_name | artist_name |
| genre | genre |
| country_key | country_key |

Figure 11 - Column mappings for inserting nonsimilar rows

19. From the Toolbox, drag an OLE DB Command transformation onto the design surface, and call it Update NDS Artist. Connect the green arrow from the Split box onto it; this time the Input Output Select doesn't pop up. It will just choose Similar because Similar is the only output left from the Split box without connection. Double-click the Update NDS Artist box to edit it. Set Connection Manager to NDS.

20. Click the Component Properties tab to display the common and custom properties, as shown in Figure 12. On this tab we need to set the SQL statement that updates the artist record in the NDS.



The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

Specify advanced properties for the data flow component.

Properties:

| | |
|---|---|
| **Common Properties** | |
| ComponentClassID | {C60ACAD1-9BE8-46B3-87DA-70E59EADEA46} |
| ContactInfo | OLE DB Command;Microsoft Corporation;Microsoft SqlServe |
| Description | Executes an SQL command for each row in a dataset. |
| ID | 1358 |
| IdentificationString | component "Update NDS Artist" (1358) |
| IsDefaultLocale | True |
| LocaleID | English (United Kingdom) |
| Name | **Update NDS Artist** |
| PipelineVersion | 0 |
| UsesDispositions | True |
| ValidateExternalMetadata | True |
| Version | 2 |
| **Custom Properties** | |
| CommandTimeout | 0 |
| DefaultCodePage | 1252 |
| SqlCommand | update artist set artist_code = ?, artist_name = ?, genre = |

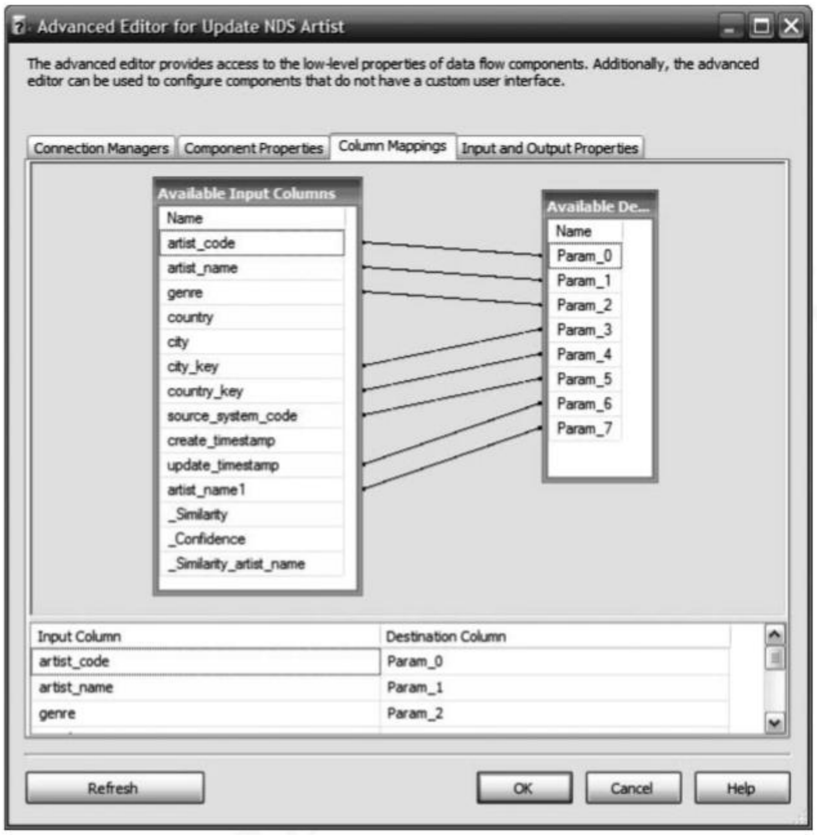
**Name**
Specifies the name of the component.

**Figure 12 - Common properties and custom properties in the Advanced Editor**

21. Set the SQL statement on the SqlCommand property, as shown in Figure 12. Please note that SqlCommand here is a property of the OLE DB Command SSIS transformation, not the .NET Framework class on the System.Data.SqlClient namespace.We need to set the SqlCommand property as follows:

```
update artist set artist_code = ?,
artist_name = ?, genre = ?,
city_key = ?, country_key = ?, source_system_code = ?,
update_timestamp = ?
where artist_name = ?
```

Notice that we don't update create_timestamp in the previous SQL because we are updating the row, not creating the row. On the Column Mappings tab, configure the mappings as shown in Figure 13, and click OK to close the Update NDS Artist OLE DB command.



| Input Column | Destination Column |
|---|---|
| artist_code | Param_0 |
| artist_name | Param_1 |
| genre | Param_2 |

**Figure 13 - Column mappings for the OLE DB command for updating similar rows**

Add two data viewers, one for the update branch and one for the insert branch, as shownin the bottom two boxes in Figure 14.

The bottom part of Figure 14 is basically an UPSERT operation (insert if it does not exist and update if it exists), but it uses fuzzy lookup to determine approximate matches. In Figure 14, the Lookup Artist Names box checks whether the incoming artist names match the NDS artist table using fuzzy lookup. The Split Based on Similarity box splits the flow based on the fuzzy lookup similarity score. The flow is split into an “insert branch” that inserts the incoming data into the artist table and an “update branch” that updates the existing rows in the artist table.

Save and execute the package, and notice that the first data row (the insert one) shows the eleven rows with similarity greater than 85 percent. Click Continue (the green arrow). The second data viewer then shows the nine rows that don’t exist in the NDS with similarity less than 70 percent. Check the NDS artist table to make sure that eleven rows in the destination artist NDS table are updated and nine are inserted.

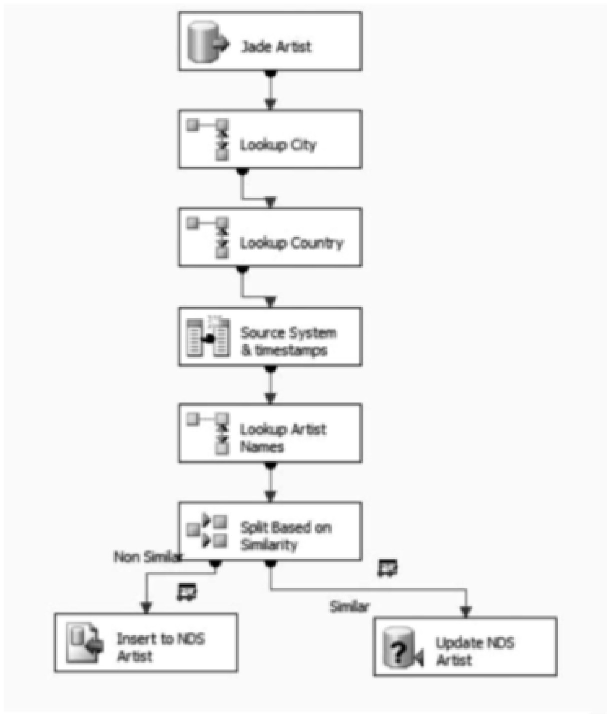


**Figure 14 - The data flow after adding split, insert, and update components**

So, we have created a SSIS package that imports the artist data from the stage table and loads it into the NDS database, using the Fuzzy Lookup transformation instead of using an exact match. If the artist name on the stage table is "similar" to the one in the NDS, we update the NDS record. Otherwise, we insert it into the NDS. We defined the "similarity" criteria by determining the similarity threshold by allowing all records from the stage to flow through the Fuzzy Logic transformation to score them. Once we determined the similarity threshold, we set the Fuzzy Logic transformation to split the data from the source into two branches. For the "similar" branch, we update the NDS based on the source data, and for the "no similar" branch we insert the source data into NDS.

**Script Component**

**The Script Component provides another area where programming logic can be applied in an SSIS package. This component, which can be used only in the Data Flow portion of an SSIS package, allows programmatic tasks to occur in the data stream. This component exists to provide, consume, or transform data using .NET code. To differentiate between the various uses of the Script Component, when you create one, you have to choose one of the following three types:**

· **Source Type Component**: The role of this Script Component is to provide data to your Data Flow Task. You can define outputs and their types and use script code to populate them. An example would be reading in a complex file format, possibly XML or something that requires custom coding to read, like HTTP or RSS Sources.

· **Destination Type Component**: This type of Script Component consumes data much like an Excel or Flat File Destination. This component is the end of the line for the data in your data stream. Here, you'll typically put the data into a Dataset variable to pass back to the Control Flow for further processing, or send the stream to custom output destinations not supported by built-in SSIS components. Examples of these output destinations can be web service calls, custom XML

Formats, and multi-record formats for mainframe systems. You can even programmatically connect and send a stream to a printer object.

· **Transformation Type Component**: This type of Script Component can perform custom transformations on data. It consumes input columns and produces output columns. You would use this component when one of the built-in transformations just isn't flexible enough.

## 7. SSIS Custom component

SSIS components are one of the basic building blocks of the SSIS framework, and can be found in the SSIS toolbox in SQL Server Data Tools. In this article we will be specifically looking at data flow components. These can read data from external data sources, transform it, and write it back to other data destinations. SQL Server Data Tools have some examples of data flow components, including the OLE DB Source, Lookup transformation and the Character Map transformation components.

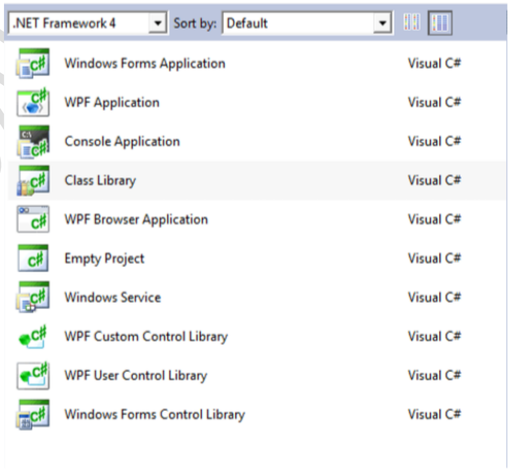
Under the hood, an SSIS data flow component is a .Net class that inherits from the Pipeline Component class, which is part of the overall SSIS Object Model. It overrides the various methods of Pipeline Component, such as those for creating inputs/outputs, adding custom properties to the component etc. The compiled class library is placed in a subfolder in the SQL Server installation folder. SQL Server Data Tools automatically recognizes any class libraries in this folder that implement Pipeline Component, and makes the component available in the SSIS Toolbox.

## 8. Building the component class

We will start by first building a bare-bones data source component, and then deploying it so that you can use it in your Integration Services package. I've already mentioned that a data source component is used to read data from an external data source, but at this stage it won't actually do anything. In the next stage, once we are satisfied that we can deploy it, we'll get it doing something useful.

Before we get into the details of building the class, note that the following description assumes that you are using Visual Studio 2012 and SQL Server Data Tools 2012. If you are using an older version of either tool, you would have to adjust accordingly. I have tried to highlight specific tools between different versions wherever possible.

With that in mind, start by launching Visual Studio, and create a new class library project, as shown in Figure 1. You can name the project Custom SSIS Component. Make sure that you select .Net Framework 4 as the .net version. The code I have presented here uses C#, but you can write it just as easily in Visual Basic also.



| | |
|---|---|
| Windows Forms Application | Visual C# |
| WPF Application | Visual C# |
| Console Application | Visual C# |
| Class Library | Visual C# |
| WPF Browser Application | Visual C# |
| Empty Project | Visual C# |
| Windows Service | Visual C# |
| WPF Custom Control Library | Visual C# |
| WPF User Control Library | Visual C# |
| Windows Forms Control Library | Visual C# |

Rename the Class1.cs file in the project to CustomSSISComponent.cs. You will also need to reference the following assemblies in your project:

| Assembly to import | Namespace to import | Can be found at |
|---|---|---|
| Microsoft.SqlServer.Pi p elineHost | Microsoft.SqlServer.Dts.Pipeline | Under the GAC_MSIL folder in the GAC |
| Microsoft.SqlServer.D T SPipelineWrap | Microsoft.SqlServer.Dts.Pipeline. W rapper | Under the GAC_MSIL folder in the GAC |
| Microsoft.SqlServer.M a nagedDTS | Microsoft.SqlServer.Dts.Runtime | Under the GAC_MSIL folder in the GAC |
| Microsoft.SqlServer.D T SRuntimeWrap | Microsoft.SqlServer.Dts.Runti me. Wrapper | Under the GAC_32 folder in the GAC |
| System.ServiceModel | System.ServiceModel | Part of the .Net framework. |

* Note that the GAC is at %windowsfolder%\Microsoft.Net\Assembly in .Net framework 4.0.
* Since we are building the component with .Net framework 4.0, make sure you refer to the assemblies at %windowsfolder%\Microsoft.Net\Assembly.
Modify the CustomSSISComponent.cs file to add the following class definition for a custom Integration services component:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.SqlServer.Dts.Pipeline;
using Microsoft.SqlServer.Dts.Pipeline.Wrapper;
using Microsoft.SqlServer.Dts.Runtime;
using Microsoft.SqlServer.Dts.Runtime.Wrapper;
using System.ServiceModel.Syndication;
using System.Data;
using System.Xml;

namespace CustomSSISComponent
{
[DtsPipelineComponent(DisplayName = "CustomSSISComponent", ComponentType = ComponentTyp public
class CustomSSISComponent : PipelineComponent
 {
 public override void AcquireConnections(object transaction)
 {
 base.AcquireConnections(transaction);
 }
 public override void PrimeOutput(int outputs, int[] outputIDs, PipelineBuffer[] buffers)  {
 base.PrimeOutput(outputs, outputIDs, buffers);
 }
 public override void PreExecute()
 {
```

```
 base.PreExecute();
 }
 public override DTSValidationStatus Validate()
 {
 return base.Validate();
 }
 public override IDTSCustomProperty100 SetComponentProperty(string propertyName, object prop {
 return base.SetComponentProperty(propertyName, propertyValue);
 }
 public override void ProvideComponentProperties()
 {
 base.ProvideComponentProperties();
 }

 }
}
```

This class inherits from PipelineComponent , which is the base class for all Integration Services data flow components. The DTSPipelineComponent attribute provides the name of the component to be shown in the SSIS Toolbox in SQLServer Data Tools, and also specifies the type of the component. We will delve into the details of the overridden methods later in this article. Since at this stage we have not provided implementations for any of the overridden methods, the component will not actually do anything. However, we can still deploy it and add it to the SSIS Toolbox.

## 9. Deploying the custom component

Follow these steps to deploy your custom SSIS component:-

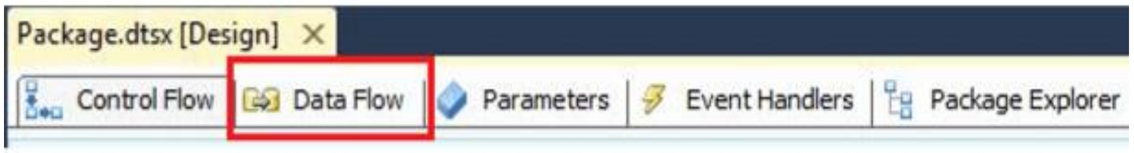
Sign the assembly with a strong name. You can either do this after building the assembly, or at build time by providing the key/value file (.snk file) in the project properties. You can find details on signing an assembly on MSDN.

Copy the strongly-named assembly to {{SQLServer Installation Folder}}\110\DTS\ PipelineComponents. On 32-bit machines, if your SQLServer installation is at the default path, this path translates to C:\Program Files\Microsoft SQLServer\110\DTS\ PipelineComponents. On 64-bit machines, the path is C:\Program Files (x86)\Microsoft SQLServer\110\DTS\ PipelineComponents

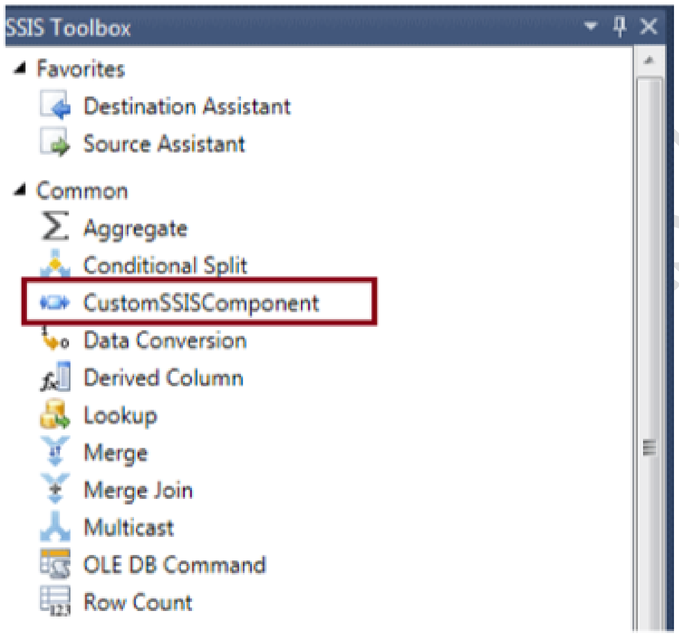
If you are developing a component for SQLServer 2008, replace the folder 110 with 100 in the above path.

Next, launch the Visual Studio developer console, and install the strongly-named assembly to the GAC using the following command –
C:\gacutil -u CustomSSISComponent
C:\gacutil -i {Solution Root folder}\CustomSSISComponent\CustomSSISComponent\bin\Debug\CustoThe above uninstalls any previous versions of the assembly and reinstalls the newer version of the DLL.
If you have multiple versions of the .Net framework installed on your computer, it is important that you use the correct version of gacutil.exe to add the file to the GAC. Since we developed the component using .Net framework 4.0, the gacutilversion should also be.Net framework 4.0.

Launch a new instance of SQLServer Data Tools, and create a blank Integration Services package. Open the package in package designer, and click on the "Data Flow" tab (shown below).



· You should be able to see your custom data source component in the SSIS Toolbox.



If everything was configured correctly, the custom component should automatically appear in the SSIS toolbox. If you don't see it, try adding it manually, via 'Tools' 'Choose Toolbox Items' 'Browse to your component DLL'.

## QUESTION BANK

### Q-1: 1 Mark Question

1. What is the use of SSIS Script task? ( July-2021)
2. Give the Full form of DQAF. ( July-2021) Data Quality Assessment Framework
3. Give the Full form of DQS. (July-2021,2019)(2023)
4. Give the Full form of GDPR. ( July-2021) General Data Protection Regulation.
5. What is Data Cleansing? ( July-2021)
6. What is Data Matching? ( July-2021)
7. Write three Script component types of SSIS. ( July-2021)
8. What is Domain Rule condition? (July-2021)
9. What is data profiling? (BKNMU- June-2021,2019)
10. List out DQS Components. (BKNMU - april-2020,2019)
11. What is Data Profiling? (BKNMU - april-2020)
12. How many type of project provided by DQS client? List out it.
13. What is data cleansing? (BKNMU - april-2022)
14. DTS stands for _________. (BKNMU - april-2022) Data Transformation Services.
15. DQKB Stands for_______ (2023)
16. DQM stands for____(2023) Data Quality Management.

### Q-2: 3 Mark Question

1. What is Data Quality Services? ( July-2021)
2. Explain Interactive process. ( July-2021)
3. Explain invalid DQS. ( July-2021)
4. Write the steps for the cycle of Data Quality Management. ( July-2021)
5. Why Data Quality is important? (April–2020)(2023)
6. Explain Features of DQS. (April–2020)(2023)
7. Explain important of data quality. (BKNMU - June-2021,2019)
8. Draw a diagram for data matching in DQS. (BKNMU - June-2021)
9. Explain data cleansing with example. (BKNMU - June-2021)
10. What is Clustering in Knowledge Base Management? (BKNMU - april-2020,2019)
11. What is DQS? Explain. (BKNMU - april-2022)
12. Discuss the benefits of DQS knowledge base. (BKNMU - april-2022)
13. Explain SSIS custom component. (BKNMU - april-2022)
14. List features provided by DQS to resolve data quality issue. (BKNMU - april-2022)
15. Write a note on data cleansing. (2023)
16. Define script task and script component. (2023)

**Q-3: 5 Mark Question**

1. Explain DQS matching process benefits. ( July-2021)
2. Explain computer assisted process. ( July-2021)
3. Explain Domain Rule Conditions in detail. (April–2020)
4. Difference between script tasks vs. component task. (BKNMU - June-2021,2019)
5. Explain data quality process with diagram. (BKNMU - June-2021,2019)(2023)
6. Explain Data Quality Service to Cleanse Data. (BKNMU - april-2020,2019)
7. Different between Script Task v/s Script Component. (BKNMU- april-2020)
8. Explain data quality service to match data. (BKNMU - april-2022)(2023)
9. Write differences between script task and script component in SSIS. (BKNMU - april-2022)