Microsoft

OFFICIAL MICROSOFT LEARNING PRODUCT

20767A

Implementing a SQL Data Warehouse

Product Number: 20767A

Part Number (if applicable):

Released: 04/2016

Module 1

Introduction to Data Warehousing

Contents:

Lesson 1

Overview of Data Warehousing

Contents:

Question and Answers

Put the following steps in a data warehouse project in order by numbering each to indicate the correct order.

	Steps
	Work with business stakeholders to identify key questions.
	Identify the data that may be able to answer the identified questions.
	Identify where data can be sourced.
	Prioritize questions that need answers.
	Identify the people or groups who require access to the produced data.

Answer:

	Steps
1	Work with business stakeholders to identify key questions.
2	Identify the data that may be able to answer the identified questions.
3	Identify where data can be sourced.
4	Prioritize questions that need answers.
5	Identify the people or groups who require access to the produced data.

Lesson 2

Considerations for a Data Warehouse Solution

Contents:

Question and Answers

Which of the following are optimal places to perform data transformations in an ETL solution?

Items	
1	Perform simple transformations in the query that is used to extract the data.
2	All the data transformations should be completed by the derived reports.
3	Implement transformations in a data flow task, within an ETL process, when complex row-by-row processing is required.
4	There is no optimal place to perform data transformations.
5	Perform complex transformations in a staging database, where it is possible to combine data from multiple sources.

Category 1		Category 2
Optimal solution		Suboptimal solution

Answer:

Category 1		Category 2
Optimal solution		Suboptimal solution
Perform simple transformations in the query that is used to extract the data. Implement transformations in a data flow task, within an ETL process, when complex row-by-row processing is required. Perform complex transformations in a staging database, where it is possible to combine data from multiple sources.		All the data transformations should be completed by the derived reports. There is no optimal place to perform data transformations.

Module Review and Takeaways

Question: Why might you consider including a staging area in your ETL solution?

Answer: You should consider using a staging area if you need to extract data from multiple data sources that have differing acquisition windows, or if you need to perform data transformations, validation, cleansing, or deduplication before loading the data into the data warehouse.

Question: Which project role would be best assigned to a business user?

( ) Solution Architect

( ) Data Modeler

( ) Data Steward

( ) Tester

Answer:

( ) Solution Architect

( ) Data Modeler

(√) Data Steward

( ) Tester

Module 2

Planning Data Warehouse Infrastructure

Contents:

Lesson 1

Considerations for Data Warehouse Infrastructure

Contents:

Question and Answers

Question: Which of these options is not a consideration for system sizing?

( ) Data volume

( ) Number of users

( ) Data center location

( ) Analysis and reporting complexity

( ) Availability requirements

Answer:

( ) Data volume

( ) Number of users

(√) Data center location

( ) Analysis and reporting complexity

( ) Availability requirements

Question: Which four factors determine the size of a BI solution?

Answer:

- Data volume.
- Analysis and reporting complexity.
- Number of users.
- Availability requirements.

Question: Discuss the considerations for deciding between single-server and distributed architecture.

Answer:

- Higher hardware costs, software costs, and configuration complexity.
- Better scalability and performance, and better flexibility.

Lesson 2

Planning Data Warehouse Hardware

Contents:

Question and Answers

Question: Which of these equations defines maximum consumption rate?

( ) (average logical writes ÷ average CPU time) × 4 ÷ 1024

( ) (average logical reads ÷ maximum CPU time) × 12 ÷ 1024

( ) (average physical reads ÷ minimum CPU time) × 8 ÷ 1024

( ) (average logical reads ÷ average CPU time) × 8 ÷ 1024

( ) (average physical reads ÷ average CPU time) × 8 ÷ 1024

Answer:

( ) (average logical writes ÷ average CPU time) × 4 ÷ 1024

( ) (average logical reads ÷ maximum CPU time) × 12 ÷ 1024

( ) (average physical reads ÷ minimum CPU time) × 8 ÷ 1024

(√) (average logical reads ÷ average CPU time) × 8 ÷ 1024

( ) (average physical reads ÷ average CPU time) × 8 ÷ 1024

Question: What factors are used in estimating CPU requirements?

Answer:

- Average query size in megabytes.
- Maximum consumption rate.
- Number of concurrent users.
- Target response time.

Question: What factors are involved when planning a storage solution?

Answer:

- Disk size
- Disk speed
- RAID
- DAS or SAN

Demonstration: Calculating Maximum Consumption Rate (MCR)

Demonstration Steps

Create tables for benchmark queries

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and then log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod02 folder, run **Setup.cmd** as Administrator.
3. In the **User Account Control** window, click **Yes**.
4. Start SQL Server Management Studio, and then connect to the **MIA-SQL** database engine by using Windows authentication.

5. In SQL Server Management Studio, open the **Create BenchmarkDB.sql** query file from the D:\Demofiles\Mod02 folder.

Execute a query to retrieve I/O statistics

1. In SQL Server Management Studio, open the **Measure MCR.sql** query file from the D:\Demofiles\Mod02 folder.
2. Click **Execute**, and then wait for query execution to complete. The queries retrieve an aggregated value from each table, and are performed twice. This ensures that on the second execution (for which statistics are shown), the data is in cache, so the I/O statistics do not include disk reads. Note that the MAXDOP=1 clause ensures that only a single core is used to process the query.

Calculate MCR from the I/O statistics

1. In the results pane, click the **Messages** tab. This shows the statistics for the queries.
2. Add the **logical reads** value for the two queries together, and then divide the result by two to find the average.
3. Add the **CPU time** value for the two queries together, and then divide the result by two to find the average. Divide the result by 100 to convert it to seconds.
4. Calculate MCR by using the following formula:

```
(average logical reads / average CPU time) * 8 / 1024
```

Module Review and Takeaways

Question: In a growing number of organizations, virtualization has become a core platform for infrastructure. Microsoft Hyper-V®, which is included as a feature from Windows 10, together with enterprise operations and management software such as Microsoft System Center, have enabled IT departments to benefit from simpler provisioning, management, mobility, and recoverability of services.

What components of a BI infrastructure would you consider virtualizing, and why?

Answer: Many database professionals are resistant to virtualization, particularly regarding data warehouses, because of the additional layer of abstraction that it adds between the database server and the physical hardware (in particular, the disk subsystem). However, advances in virtualization, such as support for virtual host bus adapters in Windows Server, mean that a virtualized environment can provide near-equivalent performance to a physical server. There are also advantages with portability and recoverability because the entire data warehouse server can be easily copied to new physical hardware or backed up.

For Reporting Services and Analysis Services, a virtualized infrastructure could be extremely beneficial because you can easily move entire virtual servers across physical hosts for hardware maintenance operations or to recover from a hardware failure. There is also the ability to dynamically scale up virtual hardware resources to match demand. For example, suppose an organization experiences a sharp rise in demand for report processing at the end of the financial year, and a corresponding drop in specific analysis as users shift their attention to generating year-end reports. If necessary, the allocation of physical memory and processor resources to the virtual server that is hosting Reporting Services could be increased by reducing the resources that are allocated to the virtual Analysis Services server. Then, after the year-end reporting activity is over, the resource allocations could be readjusted to support the normal workload balance.

 **The SQLCAT team at Microsoft has conducted research into the performance of the SQL Server database engine and Analysis Services on virtualized infrastructure. You can review their findings at:**

http://aka.ms/m6cnit

Lab Review Questions and Answers

Lab: Planning Data Warehouse Infrastructure

Question and Answers

Lab Review

Question: Review **DWHardwareSpec.xlsx** in the D:\Labfiles\Lab02\Solution folder. How does the hardware specification in this workbook compare to the one that you created in the lab?

Answer: Answers will vary. Key points about the suggested solution are:

- The amount of suggested memory is 64 GB per processor, and totals more than 20 percent of the data volume.
- The suggested storage solution is extensible (more disks can be added to the SAN) and offloads I/O processing overheads to the SAN. It also enables the solution to balance disk I/O for the storage arrays symmetrically with the processors.

Module 3

Designing and Implementing a Data Warehouse

Contents:

Lesson 1

Data Warehouse Design Overview

Contents:

Question and Answers

Question: What is the difference between a snowflake schema and a star schema?

> **Answer:** A star schema has all the dimension attributes linked directly to the fact measures. A snowflake schema has some of the dimension attributes linked to each other, in addition to others linked directly to the fact measures.

Resources

The Data Warehouse Design Process

 **Additional Reading:** For a detailed exploration of how to apply the Kimball dimensional modeling methodology to a SQL Server-based data warehouse design, read *The Microsoft Data Warehouse Toolkit* (Wiley, 2011).

Dimensional Modeling

 **Additional Reading:** For more information about using a bus matrix as part of a data warehouse design project, read *The Microsoft Data Warehouse Toolkit* (Wiley, 2011).

Lesson 2

Designing Dimension Tables

Contents:

Question and Answers

Categorize each item into the appropriate category which is a dimension type. Indicate your answer by writing the category number to the right of each item.

Items	
1	Gender
2	Fiscal Year
3	Invoice Number
4	Month
5	Out of Stock Indicator

Category 1		Category 2		Category 3
Slicer		Time Dimension		Junk Dimension

Answer:

Category 1		Category 2		Category 3
Slicer		Time Dimension		Junk Dimension
Gender		Fiscal Year Month		Invoice Number Out of Stock Indicator

Lesson 3

Designing Fact Tables

Contents:

Question and Answers

Question: What kind of measure is a stock count?

( ) Additive

( ) Semi-additive

( ) Non-additive

Answer:

( ) Additive

(√) Semi-additive

( ) Non-additive

Lesson 4

Physical Design for a Data Warehouse

Contents:

Question and Answers

Question: List three reasons for partitioning a large table.

1. **Answer:** Improved query performance.
2. More granular manageability.
3. Improved data load performance.

Resources

Table Partitioning

 **Additional Reading:** For information about how to implement partitioning, see *Partitioned Tables and Indexes* in SQL Server Books Online.

Data Compression

 **Best Practice:** Best practices for data compression in a data warehouse.
When planning tables, partitions, and indexes in a data warehouse, consider the following best practices for data compression:

- Use page compression on all dimension tables and fact table partitions.
- If performance is CPU-bound, revert to row compression on frequently-accessed partitions.

Demonstration: Partitioning a Fact Table

Demonstration Steps

Create a Partitioned Table

1. Ensure that the MSL-TMG1, 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are all running, and then log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod03 folder, run Setup.cmd as Administrator.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.
4. In the Command Prompt window press **y**, and then press **Enter**.
5. Start Microsoft® SQL Server® Management Studio and connect to the **MIA-SQL** instance of the database engine by using Windows authentication.
6. On the **File** menu, point to **Open**, and then click **File**. In the **Open File** dialog box, navigate to the **D:\Demofiles\Mod03** folder, click **Partitions.sql**, and then click **Open**.
7. Select the code under the comment **Create partition function and scheme**, and then click **Execute**.

   This creates a partition function that defines four ranges of values (less than 20140101, 20140101 to 20150100, 20150101 to 20160100, and 20160101 and higher), and a partition scheme that maps these ranges to the FG0000, FG2014, FG2015, and FG2016 filegroups.
8. Select the code under the comment **Create a partitioned table**, and then click **Execute**. This creates a partitioned table on the partition scheme you created previously.

9. Select the code under the comment **Insert data into the partitioned table**, and then click **Execute**. This inserts four records into the table.

View Partition Metadata

1. Select the code under the comment **Query the table**, and then click **Execute**. This retrieves rows from the table and uses the **$PARTITION** function to show which partition the **datekey** value in each row is assigned to. This function is useful for determining which partition of a partition function a specific value belongs in.
2. Select the code under the comment **View filegroups, partitions, and rows**, and then click **Execute**. This code uses system tables to show the partitioned storage and the number of rows in each **partition**. Note that there are two empty partitions, one at the beginning of the table, and one at the end.

Split a Partition

1. Select the code under the comment **Add a new filegroup and make it the next used**, and then click **Execute**. This creates a new filegroup named FG2017 and adds it to the partition scheme as the next used partition.
2. Select the code under the comment **Split the empty partition at the end**, and then click **Execute**. This creates a new partition for values of 20170101 and higher and assigns it to the next used filegroup (**FG2017**), leaving an empty partition for values between 20160101 and 20170100.
3. Select the code under the comment **Insert new data**, and then click **Execute**. This inserts two new rows into the partitioned table.
4. Select the code under the comment **View partition metadata**, and then click **Execute**. This shows that the two rows inserted in the previous step are in partition 4, and that partition 5 (on **FG2017**) is empty.

Merge Partitions

1. Select the code under the comment **Merge the 2014 and 2015 partitions**, and then click **Execute**. This merges the partition that contains the value 20140101 into the previous partition.
2. Select the code under the comment **View partition metadata**, and then click **Execute**. This shows that partition 2 (on **FG2014**) now contains four rows, and that the partition previously on FG2015 has been removed.
3. Close **Partitions.sql** but keep SQL Server Management Studio open for the next demonstration.

Demonstration: Creating Indexes

Demonstration Steps

Create Indexes on Dimension Tables

1. Ensure that you have completed the previous demonstration in this module.
2. In **SQL Server Management**, on the **File** menu, point to **Open**, and then click **File**. In the **Open File** dialog box, navigate to the **D:\Demofiles\Mod03** folder, click **Indexes.sql**, and then click **Open**.
3. Select the code under the comment **Create indexes on the DimDate table**, and then click **Execute**. This creates a clustered index on the surrogate key column, and non-clustered indexes on commonly queried attribute columns.
4. Select the code under the comment **Create indexes on the DimCustomer table**, and then click **Execute**. This creates a clustered index on the surrogate key column, and non-clustered indexes on commonly queried attribute columns.

5. Select the code under the comment **Create indexes on the DimProduct table**, and then click **Execute**. This creates a clustered index on the surrogate key column, and non-clustered indexes on a commonly queried attribute column.

View Index Usage and Execution Statistics

1. Select the code under the comment **Create a fact table**, and then click **Execute**. This creates a fact table named **FactOrder** that contains more than 7.5 million rows from the existing data in the dimension tables.
2. On the toolbar, click **Include Actual Execution Plan**.
3. Select the code under the comment **View index usage and execution statistics**, and then click **Execute**.

This enables statistics messages, and then queries the tables in the data warehouse to view orders for the previous six months.

4. After query execution completes, in the **Results** pane, click the **Messages** tab. Note the CPU time and elapsed time for the query. Note the logical reads from each table. The number from the **FactOrder** table should be considerably higher than the number from the dimension tables.
5. Click the **Execution plan** tab, which shows a visualization of the steps the query optimizer used to execute the query. Scroll to the right-hand side and to the bottom, and note that a table scan was used to read data from the **FactOrder** table. Then hold the mouse pointer over each of the **Index Scan** icons for the dimension tables to see which indexes were used.
6. Execute the selected code again and compare the results when the data is cached.

Create Indexes on a Fact Table

1. Select the code under the comment **Create traditional indexes on the fact table**, and then click **Execute**. This creates a clustered index on the date dimension key, and non-clustered indexes on the other dimension keys (the operation can take a long time).
2. Select the code under the comment **Empty the cache**, and then click **Execute**. This clears any cached data.
3. Select the code under the comment **Test the traditional indexes**, and then click **Execute**. This executes the same query as earlier.
4. Click the **Messages** tab and compare the number of logical reads for the **FactOrders** table and the CPU and elapsed time values with the previous execution. They should all be lower.
5. Click the **Execution plan** tab and note that the clustered index on the date key in the fact table was used.
6. Execute the selected code again and compare the results when the data is cached.

Create a Columnstore Index

1. Select the code under the comment **Create a copy of the fact table with no indexes**, and then click **Execute**. This creates an un-indexed copy of the **FactOrder** table named **FactOrderCS**.
2. Select the code under the comment **Create a columnstore index on the copied table**, and then click **Execute**. This creates a columnstore index on all columns in the **FactOrderCS** table.
3. Select the code under the comment **Empty the cache again**, and then click **Execute**. This clears any cached data.
4. Select the code under the comment **Test the columnstore index**, and then click **Execute**. This executes the same query as earlier.

5. Click the **Messages** tab and compare the number of logical reads for the **FactOrdersCS** table and the CPU and elapsed time values with the previous execution. They should all be lower.
6. Click the **Execution plan** tab and note that the columnstore index on the fact table was used.
7. Execute the selected code again and compare the results when the data is cached.
8. Close Indexes.sql but keep SQL Server Management Studio open for the next demonstration.

Demonstration: Implementing Data Compression

Demonstration Steps

Create Uncompressed Tables and Indexes

1. Ensure that you have completed the previous demonstrations in this module.
2. Use **Windows Explorer** to view the contents of the **D:\Demofiles\Mod03** folder, and set the folder window to **Details** view and resize it if necessary so that you can see the **Size** column.
3. In **SQL Server Management**, on the **File** menu, point to **Open**, and then click **File**. In the **Open File** dialog box, navigate to the **D:\Demofiles\Mod03** folder, click **Compression.sql**, and then click **Open**.
4. Select the code under the comment **Create the data warehouse** (from line2 to line 113 in the script), and then click **Execute**. This creates a database with uncompressed tables.
5. While the script is still executing, view the contents of the D:\Demofiles\Mod03 folder and note the increasing size of **DemoDW.mdf**. This is the data file for the database.
6. When execution is complete (after approximately three minutes), note the final size of **DemoDW.mdf** and return to SQL Server Management Studio.

Estimate Compression Savings

1. Select the code under the comment **Estimate size saving** (line 119 in the script), and then click **Execute**. This uses the **sp_estimate_data_compression_savings** system stored procedure to compress a sample of the **FactOrder** table (which consists of one clustered and two non-clustered indexes).
2. View the results returned by the stored procedure, noting the current size and estimated compressed size of each index.

Create Compressed Tables and Indexes

1. Select the code under the comment **Create a compressed version of the data warehouse** (from line 125 to line 250 in the script), and then click **Execute**. This creates a database with compressed tables and indexes.
2. While the script is still executing, view the contents of the D:\Demofiles\Mod03 folder and note the increasing size of **CompressedDemoDW.mdf**. This is the data file for the database.
3. When execution is complete (after approximately one minute), compare the final size of **CompressedDemoDW.mdf** with **DemoDW.mdf** (the file for the compressed database should be significantly smaller) and return to SQL Server Management Studio.

Compare Query Performance

1. Select the code under the comment **Compare query performance** (from line 255 to line 277 in the script), and then click **Execute**. This executes an identical query in the compressed and uncompressed databases and displays execution statistics.

2. When execution is complete, click the **Messages** tab and compare the statistics for the two queries. The execution time statistics (the second and third set of figures labeled "SQL Server Execution Time") should be similar, but the second query (in the compressed database) should have used considerably fewer logical reads for each table than the first query.
3. Close SQL Server Management Studio.

Module Review and Takeaways

Question: When designing a data warehouse, is it better or worse to have a strong background in transactional database design?

Answer: Answers will vary.

Module 4

Columnstore Indexes

Contents:

Lesson 1

Introduction to Columnstore Indexes

Contents:

Question and Answers

Categorize each index property into the appropriate index type. Indicate your answer by writing the category number to the right of each property.

Items	
1	Perform the best when seeking for specific data
2	A high degree of compression is possible, due to data being of the same category
3	Can improve the performance of database queries
4	Implemented as a b-tree index structure
5	Perform best when aggregating data
6	Can be stored in memory optimized tables

Category 1		Category 2		Category 3
Rowstore Index		Columnstore Index		Applies to both types of index

Answer:

Category 1		Category 2		Category 3
Rowstore Index		Columnstore Index		Applies to both types of index
Perform the best when seeking for specific data Implemented as a b-tree index structure		A high degree of compression is possible, due to data being of the same category Perform best when aggregating data		Can improve the performance of database queries Can be stored in memory optimized tables

Demonstration: The Benefits of Using Columnstore Indexes

Demonstration Steps

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are running, and then log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Run D:\Demofiles\Mod04\Setup.cmd as an administrator to revert any changes.
3. In the virtual machine, on the taskbar, click **SQL Server Management Studio**.
4. In the Connect to Server window, in the **Server name** box, type **MIA-SQL**. Ensure **Windows Authentication** is selected in the **Authentication** box.
5. Click **Connect**.
6. On the **File** menu, point to **Open**, click **File**, navigate to **D:\Demofiles\Mod04\Demo\Demo.sql** script file, and then click **Open**.
7. Select the code under **Step 1**, and then click **Execute**.
8. Select the code under **Step 2**, and then click **Execute**.
9. Select the code under **Step 1**, and then click **Execute**.
10. Select the code under **Step 3**, and then click **Execute**.
11. Select the code under the comment **Data space used**, and then click **Execute**.

Lesson 2

Creating Columnstore Indexes

Contents:

Question and Answers

Question: How will you create your indexes in a database—with SSMS or Transact-SQL?

Answer: There are advantages to both approaches.

Resources

Creating a Clustered Columnstore Index

Common Issues and Troubleshooting Tips

Common Issue	Troubleshooting Tip
Unable to create columnstore index on an Azure SQL Database.	Ensure you are using at least V12 of an Azure SQL Database. The pricing tier of the database also has to be a minimum of Premium.

Demonstration: Creating Columnstore Indexes Using SQL Server Management Studio

Demonstration Steps

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are running and then log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the virtual machine, on the taskbar, click **SQL Server Management Studio**.
3. In the Connect to Server window, in the **Server name** box, **type MIA-SQL**. Ensure **Windows Authentication** is selected in the **Authentication** box.
4. Click **Connect**.
5. In Object Explorer, expand **Databases**, expand **AdventureWorksDW**, expand **Tables**, and then expand **dbo.AdventureWorksDWBuildVersion**.
6. Right-click **Indexes**, point to **New Index**, and then click **Clustered Columnstore Index**.
7. In the **New Index** dialog box, click **OK** to create the index.
8. In Object Explorer, expand **Indexes** to show the new clustered index.
9. In Object Explorer, expand **dbo.FactResellersSales**.
10. Right-click **Indexes**, point to New Index, and then click **Non-Clustered Columnstore Index**.
11. In the **Columnstore Columns** table click **Add**.
12. Select the following columns: **SalesOrderNumber**, **UnitPrice**, and **ExtendedAmount**.
13. Click **OK**.
14. In Object Explorer, expand **Indexes** to show the created non-clustered index.

Lesson 3

Working with Columnstore Indexes

Contents:

Question and Answers

Question: When would you consider converting a rowstore table, containing dimension data in a data warehouse, to a columnstore table?

( ) When mission critical analytical queries join one or more fact tables to the dimension table—and those fact tables are columnstore tables.

( ) When the data contained in the dimension table has a high degree of randomness and uniqueness.

( ) When the dimension table has very few rows.

( ) When the dimension table has many millions of rows, with columns containing small variations in data.

( ) It is never appropriate to convert a dimension table to a columnstore table.

Answer:

( ) When mission critical analytical queries join one or more fact tables to the dimension table—and those fact tables are columnstore tables.

( ) When the data contained in the dimension table has a high degree of randomness and uniqueness.

( ) When the dimension table has very few rows.

(√) When the dimension table has many millions of rows, with columns containing small variations in data.

( ) It is never appropriate to convert a dimension table to a columnstore table.

Lab Review Questions and Answers

Lab: Using Columnstore Indexes

Question and Answers

Lab Review

Question: Why do you think the disk space savings were so large?

Answer: A clustered columnstore index re-orders data on the disk into columns in pages. These pages can be compressed far more efficiently because the data stored in columns can be very similar.

Module 5

Implementing an Azure SQL Data Warehouse

Contents:

Lesson 1

Advantages of Azure SQL Data Warehouse

Contents:

Question and Answers

Question: You are working on a data warehouse that sometimes requires executions with very high workloads, but also goes for long periods of time with a very low workload requirement. If you migrate this data warehouse to Azure SQL Data Warehouse, how can you improve the efficiency of resource allocation?

> **Answer:** You can set the DWU performance to a high level during the periods when the workload requirements are high, and set the DWU performance to a low level during the periods when the workload requirements are low.

Lesson 2

Implementing an Azure SQL Data Warehouse Database

Contents:

Question and Answers

Question: Which of the following statements about a logical server is correct?

( ) You must create a new logical server each time you create a new database.

( ) A logical server is not a physical server.

( ) You are charged for each logical server you create.

( ) The more databases that a logical server hosts, the more the performance of the logical server will decrease.

( ) You can change the DWU performance of the logical server.

Answer:

( ) You must create a new logical server each time you create a new database.

(√) A logical server is not a physical server.

( ) You are charged for each logical server you create.

( ) The more databases that a logical server hosts, the more the performance of the logical server will decrease.

( ) You can change the DWU performance of the logical server.

Demonstration: Creating and Configuring an Azure SQL Data Warehouse Database

Demonstration Steps

Create an Azure SQL Data Warehouse Database and Server

1. Ensure that the MSL-TMG1, 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and then log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod05 folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and then wait for the script to finish.
4. Open Microsoft Internet Explorer® and go to **https://portal.azure.com/**
5. Sign in to the Azure portal with your Azure pass or Microsoft account credentials.
6. Click **New**, click **Data + Storage**, and then click **SQL Data Warehouse**.
7. On the **SQL Data Warehouse** blade, in the **Name** box, type **BikeSalesDW**.
8. Drag the Performance slider to **200 DWU**.
9. Under **Server** click **Configure required settings**, and then click **Create a new server**.
10. On the **New server** blade, in the **Server name** box, type **20767a** followed by your initials, followed by today's date. For example, if your initials are CN and the date is 15 March 2016, type **20767acn15mar16**. The server name must be unique; if the name is unique and valid, a green tick appears.
11. In the **Server admin login** box, type **BikeSalesadmin**.
12. In the **Password** and **Confirm password** boxes, type **Pa$$w0rd**.
13. Under **Location**, select a region nearest your current geographical location, and then click **OK**.

14. On the **SQL Data Warehouse** blade, click **Resource group**.
15. On the **Create resource group** blade, in the **Name** box, type **BikeSalesRG**, and then click **OK**.
16. On the **SQL Data Warehouse** blade, verify that **Server** has the value of the server name you specified, **Select source** has the value **Blank database**, and that **Resource group** has the value **BikeSalesRG**, and then click **Create**.
17. It will take some time for the new server and database to be created. The Azure portal will notify you when this step is finished.

Change the Performance Settings

1. On the **Microsoft Azure** blade, click **Resource groups**, click **BikeSalesRG**.
2. On the **BikeSalesRG** blade, in the **Summary** section, click **BikeSalesDW**.
3. On the **BikeSalesDW** blade, click **Scale**.
4. On the **Scale** blade, drag the slider to **100 DWU**, click **Save**, and then, in the **Save** message box, click **Yes**.
5. Close the Scale blade.

Configure the Azure Firewall

1. On the **Microsoft Azure** blade, click **All resources**.
2. In the **All resources** blade, click **<Server Name>** (where <Server Name> is the name of the server name you created).
3. In the **<Server Name>** blade, click **Show firewall settings**.
4. In the **Firewall settings** blade, click **Add client IP**, and then click **Save**.
5. When the firewall changes are complete, click **Ok**.
6. Leave Internet Explorer open.

Connect to the Azure Server with SQL Server Management Studio

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows® authentication.
2. On the **File** menu, click **Connect Object Explorer**.
3. In the **Connect to Server** dialog box, enter the following values, and then click **Connect**:
   - **Server name**: <Server Name>.database.windows.net (where <Server Name> is the name of the server you created.)
   - **Authentication**: SQL Server Authentication
   - **Login**: BikeSalesadmin
   - **Password**: Pa$$w0rd
4. In Object Explorer, under **<Server Name>.database.windows.net**, expand **Databases**. Note that the **BikeSalesDW** database is listed.
5. Leave SQL Server Management Studio open.

Lesson 3

Developing an Azure SQL Data Warehouse

Contents:

Question and Answers

Question: A gender column in a table would be a suitable hashed column. True or false?

( ) True

( ) False

Answer:

( ) True

(√) False

Lesson 4

Migrating to an Azure SQL Data Warehouse

Contents:

Question and Answers

Question: If you are migrating a schema using the Data Warehouse Migration Utility, and some of the data in the schema has features that are incompatible with Azure SQL Data Warehouse, the migration will continue.

( ) True

( ) False

Answer:

(√) True

( ) False

Demonstration: Migrating a Database to Azure SQL Data Warehouse

Demonstration Steps

Install the Data Warehouse Migration Utility

1. In the D:\Demofiles\Mod05\Installdwmigrationutility folder, double-click **Data WarehouseMigrationUtility.msi**.
2. If the **Open File – Security Warning** dialog box appears, click **Run**.
3. In the **Data Warehouse Migration Utility** dialog box, on the **Select Installation Folder** page, click **Next**.
4. On the **License Agreement** page, review the License Agreement, click **I Agree**, and then click **Next**.
5. In the **User Account Control** dialog box, click **Yes**.
6. On the **Installation Complete** page, click **Close**.
7. Minimize all open applications and note the **Data Warehouse Migration Utility** icon on the desktop.

Check Compatibility of the Legacy Database

1. In SQL Server Management Studio, in Object Explorer, under **MIA-SQL**, expand **Databases**, expand **BikeSalesDW**, and then expand **Tables**.
2. Right-click each of the following tables, click **Select Top 1000 Rows**, and then review the data they contain:
   - **DimCustomers**
   - **DimProducts**
   - **DimResellers**
   - **FactInternetSales**
   - **FactResellerSales**
3. Minimize all windows, and then, on the desktop, double-click the **Data Warehouse Migration Utility** icon.
4. In the **DATA WAREHOUSE MIGRATION UTILITY** dialog box, note the following settings, and then click **Next**.
   - **Source Type**: SQL Server
   - **Destination Type**: Azure SQL Data Warehouse

5. In the **DATA WAREHOUSE MIGRATION UTILITY MIGRATION SOURCE** dialog box, in the **Server Name** box, type **MIA-SQL**, and then click **Connect**.
6. In the **Database List** pane, click **BikeSalesDW**, and then click **Check Compatibility**.
7. In the **Save As** dialog box, save the **BikeSalesDW – DatabaseCompatibilityReport.xlsx** file to the D:\Demofiles\Mod05 folder.
8. In the **File Saved Successfully** dialog box, click **Yes**.
9. View the data in the Excel spreadsheet, and then close Excel without saving changes.

Migrate the Schema

1. In the **Database List** pane, click **BikeSalesDW**, and then click **Migrate Selected**.
2. In the **Migrate Database** pane, select the **Table Name** check box. Note that the check boxes for all tables in the database are now selected, and then click **Migrate Schema**.
3. In the **Migrate Schema** pane, note the script that has been generated for each table, and then click **Run Script**.
4. In the **DATA WAREHOUSE MIGRATION UTILITY MIGRATION DESTINATION** dialog box, enter the following values, click **Ok**, and wait for the script to run:
   - **Database**: BikeSalesDW
   - **Server Name**: <Server Name>.database.windows.net (where <Server Name> is the name of the server you created previously)
   - **User Name**: BikeSalesadmin
   - **Password**: Pa$$w0rd
5. In the **Script Applied Successfully** message box, click **OK**.

Migrate the Data

1. In the **Migrate Schema** pane, click **Migrate Data**.
2. In the **Migrate Data** pane, click **Run Migration**.
3. In the **DATA WAREHOUSE MIGRATION UTILITY** dialog box, in the **Package Path** box, type **D:\Demofiles\Mod05**, and then click **Next**.
4. In the **DATA WAREHOUSE MIGRATION UTILITY MIGRATION DESTINATION** dialog box, click **Generate**.
5. In the **Package(s) Generated Successfully** message box, click **OK**.
6. In the D:\Demofiles\Mod05 folder, right-click **BikeSalesDW_Export.bat**, and then click **Run as administrator**.
7. In the **User Account Control** message box, click **Yes**.
8. In the D:\Demofiles\Mod05 folder, right-click **BikeSalesDW_Import.bat**, and then click **Run as administrator**.
9. In the **User Account Control** message box, click **Yes**, and wait for the bat file to run.
10. Close the DATA WAREHOUSE MIGRATION UTILITY window.

Check the Data Has Been Migrated Successfully

1. In SQL Server Management Studio, in **Object Explorer**, under <**Server Name>.database.windows.net,** expand **Databases**, right-click **BikeSalesDW**, and then click **New Query** (where <Server Name> is the name of the server you created previously).
2. If the **Unable to Apply connection settings** message box is displayed, click **OK**.
3. In the query pane, type the following SQL:

```
SELECT * FROM DimCustomers
SELECT * FROM DimProducts
SELECT * FROM DimResellers
SELECT * FROM FactInternetSales
SELECT * FROM FactResellerSales
```

4. To check the data has been migrated to each of the five tables, highlight the first line, click **Execute**, review the data, and then repeat the process for the four other lines.
5. Close SQL Server Management Studio without saving any changes, and any other windows that are open.

Lesson 5

Copying Data with the Azure Data Factory

Contents:

Question and Answers

Question: In this lesson, the pipeline activity that copies data from the on-premises database to the Azure SQL Data Warehouse database does not contain any connection strings for the databases. Why is it not necessary to specify the connection strings?

Answer: The activity receives the connection strings indirectly by referring to the source and destination datasets. Each dataset references a linked server, which contains a connection string.

Module Review and Takeaways

Review Question(s)

Question: How do you add other logins to an Azure SQL Data Warehouse logical server?

Answer: After you have created the logical server, you can connect to it using SQL Server Management Studio. You can then add the logons in the same way as you would with a SQL Server.

Module 6

Creating an ETL Solution

Contents:

Lesson 1

Introduction to ETL with SSIS

Contents:

Question and Answers

Question: The Connection Managers pane in SSDT shows all of the connection managers in the current project. True or false?

( ) True

( ) False

Answer:

( ) True

(√) False

Lesson 2

Exploring Source Data

Contents:

Question and Answers

Question: You are examining a database that has a Sales table containing a column named Product. It appears that this column references a column named ProductKey in a Product table. You want to use the Data Profiling task to check whether all values of the Product column in the Sales table have a matching value in the ProductKey column of the Product table. What type of profiling request can you use to achieve this?

( ) Candidate Key

( ) Column Null Ratio

( ) Value Inclusion

( ) Column Statistics

Answer:

( ) Candidate Key

( ) Column Null Ratio

(√) Value Inclusion

( ) Column Statistics

Demonstration: Exploring Source Data

Demonstration Steps

Extract Data with the Import and Export Data Wizard

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and then log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod06 folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. When you are prompted to confirm, click **Yes**, and then wait for the batch file to complete.
4. On the **Start** screen, type **Import and Export**, and then click the **SQL Server 2016 CTP3.3 Import and Export Data (64-bit)**.
5. On the **Welcome to SQL Server Import and Export Wizard** page, click **Next**.
6. On the **Choose a Data Source** page, set the following options, and then click **Next**:
   - **Data source**: SQL Server Native Client 11.0
   - **Server name**: localhost
   - **Authentication**: Use Windows Authentication
   - **Database**: ResellerSales
7. On the **Choose a Destination** page, select the following options, and then click **Next**:
   - **Destination**: Flat File Destination
   - **File name**: D:\Demofiles\Mod06\Top 500 Resellers.csv
   - **Text qualifier**: " (this is used to enclose exported text values in quotation marks. This is required because some European address formats include a comma, and these must be distinguished from the commas that are used to separate each column value in the exported text file)
8. On the **Specify Table Copy or Query** page, select **Write a query to specify the data to transfer**, and then click **Next**.

9. On the **Provide a Source Query** page, enter the following Transact-SQL code, and then click **Next**:

```
SELECT TOP 500 * FROM Resellers
```

10. On the **Configure Flat File Destination** page, click **Next**.
11. On the **Save and Run Package** page, select only **Run immediately**, and then click **Next**.
12. On the **Complete the Wizard** page, click **Finish**.
13. When the data extraction has completed successfully, click **Close**.

Explore Source Data in Excel

1. Start Excel.
2. On the Excel start screen, click **Open Other Workbooks**, and then click **Browse**.
3. In the **Open** dialogue box, change the drop-down box from **All Excel Files** to **Text Files**, and then open **Top 500 Resellers.csv** in the D:\Demofiles\Mod06 folder.
4. On the **Home** tab of the ribbon, click **Format as Table**, and then select a table style for the data.
5. In the **Format As Table** dialog box, ensure that the **My table has headers** check box is selected, and then click **OK**.
6. Adjust the column widths so that you can see all the data.
7. View the drop-down filter list for the **CountryRegionCode** column (column J), and note the range of values.
8. Clear the **(Select All)** check box, select the **FR** check box and then click **OK**. Note that the table is filtered to show only the resellers in France. Note also that many of the addresses include a comma. If no text qualifier had been selected in the Import and Export Data Wizard, these commas would have created additional columns in these rows, making the data difficult to examine as a table.
9. Clear the filter for the **CountryRegionCode** column.
10. In cell O1, enter the following formula:

```
=MIN(Table1[YearOpened])
```

11. Note that this formula shows the earliest year that a store in this sample data was opened.
12. Close Excel without saving any changes.

Demonstration: Using the Data Profiling Task

Demonstration Steps

Use the Data Profiling Task

1. Ensure you have completed the previous demonstration in this module.
2. Start SQL Server Data Tools, and on the **File** menu, point to **New**, and then click **Project**.
3. In the **New Project** dialog box, select the following values, and then click **OK**.
   - **Project Template**: Integration Services Project
   - **Name**: ProfilingDemo
   - **Location**: D:\Demofiles\Mod06
   - **Solution name**: ProfilingDemo

4. If the Getting Started (SSIS) window is displayed, close it.
5. In the Solution Explorer pane, right-click **Connection Managers**, and then click **New Connection Manager**.
6. In the **Add SSIS Connection Manager** dialog box, click **ADO.NET**, and then click **Add**.
7. In the **Configure ADO.NET Connection Manager** dialog box, click **New**.
8. In the **Connection Manager** dialog box, enter the following values, and then click **OK**:
   - **Server name**: localhost
   - **Log on to the server**: Use Windows Authentication
   - **Select or enter a database name**: ResellerSales
9. In the **Configure ADO.NET Connection Manager** dialog box, verify that a data connection named **localhost.ResellerSales** is listed, and then click **OK**.
10. If the SSIS Toolbox pane is not visible, on the **SSIS** menu, click **SSIS Toolbox**. Then, in the SSIS Toolbox pane, in the **Common** section, double-click **Data Profiling Task** to add it to the Control Flow surface.
11. Double-click the **Data Profiling Task** icon on the Control Flow surface.
12. In the **Data Profiling Task Editor** dialog box, on the **General** page, in the **Destination** property drop-down list, click **<New File connection...>**.
13. In the **File Connection Manager Editor** dialog box, in the **Usage type** drop-down list, click **Create file**. In the **File** box, type **D:\Demofiles\Mod06\Reseller Sales Data Profile.xml**, and then click **OK**.
14. In the **Data Profiling Task Editor** dialog box, on the **General** page, set **OverwriteDestination** to **True**.
15. In the **Data Profiling Task Editor** dialog box, on the **Profile Requests** page, in the **Profile Type** drop-down list, click **Column Statistics Profile Request**, and then click the **RequestID** column.
16. In the Request Properties pane, set the following property values:
    - **ConnectionManager**: localhost.ResellerSales
    - **TableOrView**: [dbo].[SalesOrderHeader]
    - **Column**: OrderDate
17. In the **Data Profiling Task Editor** dialog box, click **OK**.
18. On the **File** menu, point to **Open** and click **Project/ Solution**.
19. In the **Open Project** dialog box, open the **Explore Reseller Sales.sln** file in the D:\DemoFiles\Mod06\Profilerequests folder, saving your changes to the previous solution if you are prompted.
20. In Solution Explorer, expand **SSIS Packages** and double-click **Package.dtsx**.
21. Double-click the **Data Profiling Task** icon on the Control Flow surface.
22. In the **Data Profiling Task Editor** dialog box, click the **Profile Requests** page, and then note the three profile requests.

23. Click the **Column Length Distribution Profile Request** row and note the following settings in the Request Properties pane:
    - **ConnectionManager**: localhost.ResellerSales
    - **TableOrView**: [dbo].[Resellers]
    - **Column**: AddressLine1
24. Click the **Column Null Ratio Profile Request** row and note the following settings in the Request Properties pane:
    - **ConnectionManager**: localhost.ResellerSales
    - **TableOrView**: [dbo].[Resellers]
    - **Column**: AddressLine2
25. In the **Profile Type** list, click below **Column Null Ratio Profile Request**, in the drop-down list, click **Value Inclusion Profile Request**, and then click the **RequestID** column.
26. In the Request Properties pane, set the following property values:
    - **ConnectionManager**: localhost.ResellerSales
    - **SubsetTableOrView**: [dbo].[SalesOrderHeader]
    - **SupersetTableOrView**: [dbo].[PaymentTypes]
    - **InclusionColumns**:
      - **Subset side Columns**: PaymentType
      - **Superset side Columns**: PaymentTypeKey
    - **InclusionThresholdSetting**: None
    - **SupersetColumnsKeyThresholdSetting**: None
27. In the **Data Profiling Task Editor** dialog box, click **OK**.
28. On the **Debug** menu, click **Start Debugging**.

View a Data Profiling Report

1. When the Data Profiling task has completed, with the package still running, double-click **Data Profiling Task**, and then in the **Data Profiling Task Editor**, click **Open Profile Viewer**.
2. Maximize the **Data Profile Viewer** window and under the **[dbo].[SalesOrderHeader]** table, click **Column Statistics Profiles**. Review the minimum and maximum values for the **OrderDate** column.
3. Under the **[dbo].[Resellers]** table, click **Column Length Distribution Profiles.**. Click the bar chart for any of the column lengths, and then click the **Drill Down** button (at the right edge of the title area for the middle pane) to view the source data that matches the selected column length.
4. Close the **Data Profile Viewer** window, and then in the **Data Profiling Task Editor** dialog box, click **Cancel**.
5. On the **Debug** menu, click **Stop Debugging**, and then close SSDT, saving your changes if prompted.
6. On the Start screen, type **Data Profile**, and then click **SQL Server 2016 CTP3.3 Data Profile Viewer**. When the app starts, maximize it.
7. On the toolbar, click **Open**, and open **Reseller Sales Data Profile.xml** in the D:\Demofiles\Mod06 folder.

8. Under the **[dbo].[Resellers]** table, click **Column Null Ratio Profiles** and view the null statistics for the **AddressLine2** column. Select the **AddressLine2** column, and then click the **Drill Down** button to view the source data.
9. Under the **[dbo].[SalesOrderHeader]** table, click **Inclusion Profiles** and review the inclusion statistics for the **PaymentType** column. Select the inclusion violation for the payment type value of **0**, and then click the **Drill Down** button to view the source data. **Note**: The **PaymentTypes** table includes two payment types, using the value 1 for invoice-based payments and 2 for credit account payments. The Data Profiling task has revealed that for some sales, the value 0 is used, which may indicate an invalid data entry or may be used to indicate some other kind of payment that does not exist in the **PaymentTypes** table.
10. Close the Data Profile Viewer window.

Lesson 3

Implementing Data Flow

Contents:

Question and Answers

Question: In a lookup transformation component, the source of the lookup columns is a table, view, or SQL statement that you must specify. Why do you not need to specify the source of the input columns?

> **Answer:** When a source/transformation component is connected to the next component in the data flow, all output columns from the source/transformation component are input into the next component. The input columns in a lookup transformation are therefore determined by the columns output from the previous source/ transformation component in the data flow.

Demonstration: Implementing a Data Flow

Demonstration Steps

Configure a Data Source Component

1. Ensure you have completed the previous demonstrations in this module.
2. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine instance using Windows authentication.
3. In Object Explorer, expand **Databases**, expand **Products**, and then expand **Tables**. Right-click each of the following tables, click **Select Top 1000 Rows,** and then review the data they contain.
   - **dbo.Product**
   - **dbo.ProductCategory**
   - **dbo.ProductSubcategory**
4. In Object Explorer, under **Databases**, expand **DemoDW**, and then expand **Tables**. Right-click **dbo.DimProduct**, and then click **Select Top 1000 Rows** to verify that this table is empty.
5. Start SQL Server Data Tools and create a new Integration Services project named **DataFlowDemo** in the D:\Demofiles\Mod06 folder.
6. If the **Getting Started (SSIS)** window is displayed, close it.
7. In Solution Explorer, expand **SSIS Packages**, right-click **Package.dtsx,** and then click **Rename**. Change the package name to **ExtractProducts.dtsx.**
8. In Solution Explorer, right-click **Connection Managers**, and then click **New Connection Manager**. Add a new **OLEDB** connection manager with the following settings:
   - **Server name**: localhost
   - **Log on to the server**: Use Windows Authentication
   - **Select or enter a database name**: Products
9. In the **SSIS Toolbox**, in the **Favorites** section, double-click **Data Flow Task** to add it to the Control Flow surface.
10. Right-click **Data Flow Task**, and then click **Rename**. Change its name to **Extract Products**.
11. Double-click **Extract Products** to switch to the **Data Flow** tab.
12. In the **SSIS Toolbox**, in the **Favorites** section, double-click **Source Assistant** to add a source component to the Data Flow surface.
13. In the **Source Assistant - Add New Source** dialog box, in the list of types, click **SQL Server**. In the list of connection managers, click **localhost.Products**, and then click **OK**.
14. Rename the **OLE DB Source** to **Products**, and then double-click it to edit its settings.

15. In the **OLE DB Source Editor** dialog box, on the **Connection Manager** page, view the list of available tables and views in the drop-down list.
16. Change the data access mode to **SQL Command**, and then enter the following Transact-SQL code:

```
SELECT ProductKey, ProductName FROM Product
```

17. Click **Build Query** to open the **Query Builder** dialog box.
18. In the **Product** table, select the **ProductSubcategoryKey**, **StandardCost**, and **ListPrice** columns, and then click **OK**.
19. In the **OLE DB Source Editor** dialog box, click **Preview** to see a data preview, and then click **Close** to close the **Preview Query Results** dialog box.
20. In the **OLE DB Source Editor** dialog box, on the **Columns** page, view the list of external columns that the query has returned and the output columns generated by the data source, and then click **OK**.

Use a Derived Column Transformation

1. In the SSIS Toolbox pane, in the **Common** section, double-click **Derived Column** to add a derived column transformation component to the Data Flow surface, and then position it below the **Products** source.
2. Rename the **Derived Column** transformation component to **Calculate Profit**.
3. Select the **Products** source component, and then drag the blue output arrow to the **Calculate Profit** transformation component.
4. Double-click the **Calculate Profit** transformation component to edit its settings, and then in the **Derived Column Name** box, type **Profit**.
5. Ensure that **<add as new column>** is selected in the **Derived Column** box.
6. Expand the **Columns** folder, and then drag the **ListPrice** column to the Expression box.
7. In the Expression box, after [ListPrice], type a minus sign (–), and then drag the **StandardCost** column to the Expression box to create the following expression:

```
[ListPrice]-[StandardCost]
```

8. Click the **Data Type** box, ensure that it is set to **Currency [DT_CY]**, and then click **OK**.

Use a Lookup Transformation

1. In the SSIS Toolbox pane, in the **Common** section, double-click **Lookup** to add a lookup transformation component to the Data Flow surface, and then position it below the **Calculate Profit** transformation component.
2. Rename the **Lookup** transformation component to **Lookup Category**.
3. Select the **Calculate Profit** transformation component, and then drag the blue output arrow to the **Lookup Category** transformation component.
4. Double-click the **Lookup Category** transformation component to edit its settings.
5. In the **Lookup Transformation Editor** dialog box, on the **General** page, in the **Specify how to handle rows with no matching entries** list, select **Redirect rows to no match output**.
6. In the **Lookup Transformation Editor** dialog box, on the **Connection** page, ensure that the **localhost.Products** connection manager is selected, and then click **Use results of an SQL query**.

7. Click **Browse**, and then in the D:\Demofiles\Mod06 folder, open the **LookupProductCategories.sql** query.
8. Click **Preview** to view the product category data, note that it includes a **ProductSubcategoryKey** column, and then click **Close** to close the preview.
9. In the **Lookup Transformation Editor** dialog box, on the **Columns** page, in the **Available Input Columns** list, drag **ProductSubcategoryKey** to **ProductSubCategoryKey** in the **Available Lookup Columns** list.
10. Select the **ProductSubcategoryName** and **ProductCategoryName** columns to add them as new columns to the data flow, and then click **OK**.

Configure a Destination Component

1. In Solution Explorer, create a new OLE DB connection manager with the following settings:
   - **Server name**: localhost
   - **Log on to the server**: Use Windows Authentication
   - **Select or enter a database name**: DemoDW
2. In the SSIS Toolbox pane, in the **Favorites** section, double-click **Destination Assistant** to add a destination component to the Data Flow surface.
3. In the **Destination Assistant - Add New Destination** dialog box, in the list of types, click **SQL Server**. In the list of connection managers, click **localhost.DemoDW,** and then click **OK**.
4. Rename the OLE DB destination component to **DemoDW** and position it below the **Lookup Category** transformation component.
5. Select the **Lookup Category** transformation component, and then drag the blue output arrow to the **DemoDW** destination component.
6. In the **Input Output Selection** dialog box, in the **Output** list, click **Lookup Match Output**, and then click **OK**.
7. Double-click the **DemoDW** destination component to edit its settings, and then in the **Name of the table or the view** list, click **[dbo].[DimProduct]**.
8. In the **OLE DB Destination Editor** dialog box, on the **Mappings** page, note that input columns are automatically mapped to destination columns with the same name.
9. In the **Available Input Columns** list, drag the **ProductKey** column to the **ProductID** column in the **Available Destination Columns** list, and then click **OK**.
10. In the SSIS Toolbox pane, in the **Other Destinations** section, double-click **Flat File Destination** to add a destination component to the Data Flow surface, and then position it to the right of the **Lookup Category** transformation component.
11. Rename the flat file destination component **Uncategorized Products**.
12. Select the **Lookup Category** transformation component, and then drag the blue output arrow to the **Uncategorized Products** destination component. The **Lookup No Match Output** output is automatically selected.
13. Double-click the **Uncategorized Products** destination component to edit its settings, and then click **New**. In the **Flat File Format** dialog box, select **Delimited** and click **OK**.

14. In the **Flat File Connection Manager Editor** dialog box, name the new connection manager **Unmatched Products,** specify the file name **D:\Demofiles\Mod06\UnmatchedProducts.csv**, and then click **OK**.
15. In the **Flat File Destination Editor** dialog box, click the **Mappings** page and note that the input columns are mapped to destination columns with the same names, and then click **OK**.
16. On the **Debug** menu, click **Start Debugging**, and observe the data flow as it runs. Note that the number of rows transferred along each path is shown in front of the arrows connecting the components in the data flow.
17. When the data flow has completed, on the **Debug** menu, click **Stop Debugging**.
18. Close SSDT, saving your changes if you are prompted.
19. Start Excel, and open the **Unmatched Products.csv** flat file in the D:\Demofiles\Mod06 folder. Note that there were no unmatched products.
20. Use SQL Server Management Studio to view the contents of the **DimProduct** table in the **DemoDW** database, and note that the product data has been transferred.
21. Close SQL Server Management Studio without saving any files.

Module Review and Takeaways

Question: How could you determine the range of OrderDate values in a data source to plan a time dimension table in a data warehouse?

Answer: You could use SQL Server Management Studio to query for the MIN and MAX values of the OrderDate column. Alternatively, you could extract the data from the table to Excel and use formulas to determine the minimum and maximum values. Lastly, you could retrieve the Column Statistics profile request in the Data Profiling task.

Module 7

Implementing Control Flow in an SSIS Package

Contents:

Lesson 1

Introduction to Control Flow

Contents:

Question and Answers

Question: Which of the following is one of the key advantages of developing multiple packages within an SSIS solution?

( ) It isolates individual units of workflow in the solution.

( ) It creates reusable units of workflow which you can use multiple times in an ETL process.

( ) Multiple packages result in a less cluttered workspace.

( ) Multiple small packages use less resources than a single large package.

Answer:

( ) It isolates individual units of workflow in the solution.

(√) It creates reusable units of workflow which you can use multiple times in an ETL process.

( ) Multiple packages result in a less cluttered workspace.

( ) Multiple small packages use less resources than a single large package.

Demonstration: Implementing Control Flow

Demonstration Steps

Add Tasks to a Control Flow

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and then log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the **D:\Demofiles\Mod07** folder, run **Setup.cmd** as Administrator.
3. Start **SQL Server Data Tools** and open **ControlFlowDemo.sln** from the **D:\Demofiles\Mod07** folder.
4. In Solution Explorer, double-click **Control Flow.dtsx**.
5. If the SSIS Toolbox is not visible, on the **SSIS** menu, click **SSIS Toolbox**. Then, from the SSIS Toolbox, drag a **File System Task** to the control flow surface.
6. Double-click **File System Task**, configure the following settings, and then click **OK**.
   - **Name**: Delete Files
   - **Operation**: Delete directory content
   - **SourceConnection**: A new connection with a **Usage type** of **Create folder**, and a **Folder** value of **D:\Demofiles\Mod07\Demo**. (Do not create this folder)
7. From the SSIS Toolbox, drag a second **File System Task** to the control flow surface. Then double-click **File System Task**, configure the following settings, and then click **OK**.
   - **Name**: Delete Folder
   - **Operation**: Delete directory
   - **SourceConnection**: Demo
8. From the SSIS Toolbox, drag a third **File System Task** to the control flow surface. Then double-click **File System Task**, configure the following settings, and then click **OK**.
   - **Name**: Create Folder
   - **Operation**: Create directory

- **UseDirectoryIfExists**: True
- **SourceConnection**: Demo

9. From the SSIS Toolbox, drag a fourth **File System Task** to the control flow surface. Then double-click the **File System Task**, configure the following setting, and then click **OK**.
   - **Name:** Copy File
   - **Operation:** Copy file
   - **DestinationConnection:** Demo
   - **OverwriteDestination**: True
   - **SourceConnection**: A new connection with a **Usage type** of **Existing file**, and a File value of **D:\Demofiles\Mod07\Demo.txt**
10. From the SSIS Toolbox, drag a **Send Mail Task** to the control flow surface. Then double-click the **Send Mail Task**, configure the following setting, and then click **OK**.
    - **Name** (on the **General** tab): Send Failure Notification
    - **SmtpConnection** (on the **Mail** tab): Create a new SMTP connection manager with a Name property of **Local SMTP Server** and an **SMTP Server** property of **localhost**. Use the default values for all other settings
    - **From** (on the **Mail** tab): demo@adventureworks.msft
    - **To** (on the **Mail** tab): student@adventureworks.msft
    - **Subject** (on the **Mail** tab): Control Flow Failure
    - **MessageSource** (on the **Mail** tab): A task failed

Use Precedence Constraints to Define a Control Flow

1. Select the **Delete Files** task and drag its green arrow to the **Delete Folder** task. Then connect the **Delete Folder** task to the **Create Folder** task and the **Create Folder** task to the **Copy File** task.
2. Connect each of the File System tasks to the **Send Failure Notification** task.
3. Right-click the connection between **Delete Files** and **Delete Folder**, and then click **Completion**.
4. Right-click the connection between **Delete Folder** and **Create Folder** and click **Completion**.
5. Click the connection between the **Delete Files** task and the **Send Failure Notification** task to select it. Then hold the **Ctrl** key and click each connection between the remaining File System tasks and the **Send Failure Notification** task while holding the **Ctrl** key to select them all.
6. Press F4 and in the Properties pane, set the **Value** property to **Failure**.
7. Click anywhere on the control flow surface to clear the current selection, and then double-click any of the red constraints connected to the **Send Failure Notification** task. Then in the **Precedence Constraint Editor** dialog box, in the **Multiple constraints** section, click **Logical OR. One constraint must evaluate to True**, and click **OK**. Note that all connections to the **Send Failure Notification** task are now dotted to indicate that a logical OR operation is applied.
8. Right-click the control flow surface next to the **Send Failure Notification** task and click **Add Annotation**. Then type **Send an email message if a task fails**.
9. Select the **Delete Files** and **Delete Folder** tasks, then right-click either of them and click **Group**. Drag the group to rearrange the control flow so you can see that the **Delete Folder** task is still connected to the **Create Folder** task.

10. On the **Debug** menu, click **Start Debugging** to run the package, and note that the **Delete Files** and **Delete Folder** tasks failed because the specified folder did not previously exist. This caused the **Send Failure Notification** task to be executed.
11. You can view the email message that was sent by the **Send Failure Notification** task in the **C:\inetpub\mailroot\Drop** folder. Double-click it to open with Outlook.
12. In SQL Server Data Tools, on the **Debug** menu, click **Stop Debugging**, and then run the package again. This time, all the File System tasks should succeed because the folder was created during the previous execution. Consequently, the **Send Failure Notification** task is not executed.
13. Stop debugging and close **SQL Server Data Tools**. Save the solution files if prompted.

Lesson 2

Creating Dynamic Packages

Contents:

Demonstration: Using Variables and Parameters

Demonstration Steps

Create a Variable

1. Ensure you have completed the previous demonstration in this module.
2. Start **SQL Server Data Tools** and open the **VariablesAndParameters.sln** solution in the **D:\Demofiles\Mod07** folder.
3. In Solution Explorer, double-click **Control Flow.dtsx**.
4. On the **View** menu, point to **Other Windows**, and click **Variables**.
5. In the Variables pane, click **Add Variable** and add a variable with the following properties:
   - **Name**: fName
   - **Scope**: Control Flow
   - **Data type**: String
   - **Value**: Demo1.txt

Create a Parameter

1. In Solution Explorer, double-click **Project.params**.
2. In the Project.params [Design] pane, click **Add Parameter** and add a parameter with the following properties:
   - **Name:** FolderPath
   - **Data type:** String
   - **Value**: D:\Demofiles\Mod07\Files\
   - **Sensitive**: False
   - **Required**: True
   - **Description**: Folder containing text files

Note: Be sure to include the trailing "\" in the Value property.

3. Save all files and close the **Project.params [Design]** window.

Use a Variable and a Parameter in an Expression

1. On the **Control Flow.dtsx** package design surface, in the Connection Managers pane, click the **Demo.txt** connection manager, and then press F4.
2. In the Properties pane, in the **Expressions** property box, click the ellipsis (**…**) button. In the **Property Expressions Editor** dialog box, in the **Property** box, select **ConnectionString** and in the **Expression box**, click the ellipsis (**…**) button.
3. In the **Expression Builder** dialog box, expand the **Variables and Parameters** folder, and then drag the **$Project::folderPath** parameters to the **Expression** box. In the **Expression** box, type a plus (**+**) symbol, and then drag the **User::fName** variable to the **Expression** box to create the following expression:

   @[$Project::folderPath]+[@User::fName]

4. In the **Expression Builder** dialog box, click **Evaluate Expression** and verify that the expression produces the result **D:\Demofiles\Mod07\Files\Demo1.txt**.

5. Click **OK** to close the **Expression Builder** dialog box, and then in the **Property Expressions Editor** dialog box, click **OK**.
6. Run the project, and when it has completed, stop debugging and close **SQL Server Data Tools**.
7. View the contents of the **D:\Demofiles\Mod07\Demo** folder and verify that **Demo1.txt** has been copied.

Lesson 3

Using Containers

Contents:

Demonstration: Using a Sequence Container

Demonstration Steps

Use a Sequence Container

1. Ensure you have completed the previous demonstrations in this module.
2. Start **SQL Server Data Tools** and open the **SequenceContainer.sln** solution in the D:\Demofiles\Mod07 folder.
3. In Solution Explorer, double-click **Control Flow.dtsx**.
4. Right-click the **Group** indicator around the **Delete Files** and **Delete Folder** tasks, and then click **Ungroup** to remove it.
5. Drag a **Sequence Container** from the SSIS Toolbox to the control flow design surface.
6. Right-click the precedence constraint that connects **Delete Files** to **Send Failure Notification**, and click **Delete**. Then delete the precedence constraints connecting the **Delete Folder** to **Send Failure Notification** and **Delete Folder** to **Create Folder**.
7. Click and drag around the **Delete Files** and **Delete Folder** tasks to select them both, and then drag into the Sequence Container.
8. Drag a precedence constraint from the Sequence Container to **Create Folder**. Then right-click the precedence constraint and click **Completion**.
9. Drag a precedence constraint from the Sequence Container to **Send Failure Notification**. Then right-click the precedence constraint and click **Failure**.
10. Run the package and view the results, then stop debugging.
11. Click the Sequence container and press F4. Then in the Properties pane, set the **Disable** property to **True**.
12. Run the package again and note that neither of the tasks in the Sequence container is executed. Then stop debugging and close **SQL Server Data Tools**.

Demonstration: Using a For Loop Container

Demonstration Steps

Use a For Loop Container

1. Ensure you have completed the previous demonstrations in this module.
2. Start **SQL Server Data Tools** and open the **ForLoopContainer.sln** solution in the **D:\Demofiles\Mod07** folder.
3. In Solution Explorer, double-click **Control Flow.dtsx**.
4. If the Variables window is not open, on the **View** menu, point to **Other Windows**, and then click **Variables**. Then add a variable with the following properties:
   - **Name**: counter
   - **Scope**: Control Flow
   - **Data type**: Int32
   - **Value**: 0
5. From the SSIS Toolbox, drag a **For Loop Container** to the control flow design surface.

6. Double-click the **For Loop Container**, set the following properties, and then click **OK**:
   - **InitExpression**: @counter = 1
   - **EvalExpression**: @counter < 4
   - **AssignExpression**: @counter = @counter + 1
7. From the SSIS Toolbox, drag an **Execute Process Task** into the For Loop container.
8. Double-click the **Execute Process Task**, set the following properties, and then click **OK**:
   - **Name** (on the **General** tab): Open File
   - **Executable** (on the **Process** tab): Notepad.exe
   - **Expressions** (on the **Expressions** tab): Use the **Property Expressions Editor** to set the following expression for the **Arguments** property:

@[$Project::folderPath] + "Demo" + (DT_WSTR,1)@[User::counter] + ".txt"

9. Drag a precedence constraint from the **For Loop Container** to the **Sequence Container** and rearrange the control flow if necessary.
10. Run the package, and note that the For Loop starts Notepad three times, opening the text file with the counter variable value in its name (Demo1.txt, Demo2.txt, and Demo3.txt). Close Notepad each time it opens, and when the execution is complete, stop debugging.
11. Close **SQL Server Data Tools**, saving the solution files if prompted.

Demonstration: Using a Foreach Loop Container

Demonstration Steps

Use a Foreach Loop Container

1. Ensure you have completed the previous demonstrations in this module.
2. Start **SQL Server Data Tools** and open the **ForeachLoopContainer.sln** solution in the **D:\Demofiles\Mod07** folder.
3. In Solution Explorer, double-click **Control Flow.dtsx**.
4. From the SSIS Toolbox, drag a **Foreach Loop Container** to the control flow design surface. Then double-click the **Foreach loop Container** to view the **Foreach Loop Editor** dialog box.
5. On the **Collection** tab, in the **Enumerator** list, click **Foreach File Enumerator**. In the **Expressions** box, click the ellipsis (**…**) button. In the **Property Expressions Editor** dialog box, in the **Property** list, click **Directory** and in the **Expression** box click the ellipsis (**…**) button.
6. In the **Expression Builder** dialog box, expand the **Variables and Parameters** folder and drag the **$Project::folderPath** parameter to the **Expression** box to specify that the loop should iterate through files in the folder referenced by the **folderPath** project parameter. Click **OK** to close the **Expression Builder** and then in the **Property Expressions Editor** dialog box, click **OK**.
7. In the **Foreach Loop Editor** dialog box, on the **Collection** tab, in the **Retrieve file name** section, select **Name and extension** to return the file name and extension for each file the loop finds in the folder.
8. In the **Foreach Loop Editor** dialog box, on the **Variable Mappings** tab, in the **Variable** list, click **User::fName** and in the **Index** column, select **0** to assign the file name of each file found in the folder to the **fName** variable. Click **OK**.

9. Remove the precedence constraints that are connected to and from the **Copy File** task, and then drag the **Copy File** task into the **Foreach Loop Container**.
10. Create a precedence constraint from the **Create Folder** task to the **Foreach Loop Container**, and a precedence constraint from the **Foreach Loop Container** to the **Send Failure Notification** task.
11. Right-click the constraint between the **Foreach Loop Container** and the **Send Failure Notification** task, and click **Failure**.
12. Run the package, closing each instance of Notepad as it opens. When the package execution has completed, stop debugging and close SQL Server Data Tools, saving the solution files if prompted.
13. Verify that the **D:\Demofiles\Mod07\Demo** folder contains each of the files in the **D:\Demofiles\Mod07\Files** folder.

Lesson 4

Managing Consistency

Contents:

Question and Answers

Question: You have developed an SSIS package containing two **Execute SQL Task** tasks. The first task extracts data and the second loads a table. A checkpoint is configured for the package. During execution, the second task fails after loading 50 percent of the table data. What will happen when the package is next executed (assuming the error which caused the failure has been rectified)?

( ) The package will rerun all tasks from the beginning.

( ) The package will fail.

( ) The package will rerun the second task, loading the remaining 50 percent of the table data.

( ) The package will rerun the second task from the beginning.

( ) The package will rerun the first task but skip the second task to avoid another failure.

Answer:

( ) The package will rerun all tasks from the beginning.

( ) The package will fail.

( ) The package will rerun the second task, loading the remaining 50 percent of the table data.

(√) The package will rerun the second task from the beginning.

( ) The package will rerun the first task but skip the second task to avoid another failure.

Demonstration: Using a Transaction

Demonstration Steps

Use a Transaction

1. If you did not complete the previous demonstrations in this module, ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and log on to **20767A-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**. Then, in the D:\Demofiles\Mod07 folder, run **Setup.cmd** as administrator.
2. Start **SQL Server Management Studio** and connect to the **localhost** database engine instance using Windows authentication.
3. In Object Explorer, expand **Databases**, expand **DemoDW**, and then expand **Tables**.
4. Right-click **dbo.StagingTable** and click **Select Top 1000 Rows** to verify that it contains product data.
5. Right-click **dbo.ProductionTable** and click **Select Top 1000 Rows** to verify that it is empty.
6. Start **SQL Server Data Tools** and open the **Transactions.sln** solution in the **D:\Demofiles\Mod07** folder.
7. In Solution Explorer, double-click **Move Products.dtsx**. Note that the control flow consists of a Data Flow task named **Copy Products** that moves products from a staging table to a production table, and an **SQL Command** task named **Update Prices** that sets the product price.
8. On the **Debug** menu, click **Start Debugging** to run the package and note that the **Update Prices** task fails. Then on the **Debug** menu, click **Stop Debugging**.
9. In **SQL Server Management Studio**, select the top 1,000 rows from the **dbo.ProductionTable** table, noting that it now contains product data but the prices are all set to 0.00. You want to avoid having products with invalid prices in the production table, so you need to modify the SSIS package to ensure that, when the price update task fails, the production table remains empty.

10. On the **File** menu, point to **New**, and then click **Query with Current Connection**, type the following Transact-SQL code, and then click **Execute**. This deletes all rows in the **dbo.ProductionTable** table:

```
TRUNCATE TABLE DemoDW.dbo.ProductionTable;
```

11. **In SQL Server Data Tools**, click anywhere on the Control Flow surface and press F4. Then in the Properties pane, set the **TransactionOption** property to **Required**.
12. Click the **Copy Products** task, and in the Properties pane, set the **FailPackageOnFailure** property to **True** and ensure the **TransactionOption** property is set to **Supported**.
13. Repeat the previous step for the **Update Prices** task.
14. Run the package and note that the **Update Prices** task fails again. Then stop debugging.
15. In **SQL Server Management Studio**, select the top 1,000 rows from the **dbo.ProductionTable** table, noting that it is empty, even though the **Copy Products** task succeeded. The transaction has rolled back the changes to the production table because the **Update Prices** task failed.
16. In SQL Server Data Tools, double-click the **Update Prices** task and change the **SQLStatement** property to **UPDATE ProductionTable SET Price = 100**. Then click **OK**.
17. Run the package and note that all tasks succeed. Then stop debugging and close SQL Server Data Tools.
18. In **SQL Server Management Studio**, select the top 1,000 rows from the **dbo.ProductionTable** table, noting that it now contains products with valid prices.

Demonstration: Using a Checkpoint

Demonstration Steps

Use a Checkpoint

1. If you did not complete the previous demonstrations in this module, ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and log on to 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**. Then, in the **D:\Demofiles\Mod07** folder, run **Setup.cmd** as administrator.
2. Start **SQL Server Management Studio** and connect to the **localhost** database engine instance using Windows authentication.
3. In Object Explorer, expand **Databases**, expand **DemoDW**, and expand **Tables**.
4. Right-click **dbo.StagingTable** and click **Select Top 1000 Rows** to verify that it contains product data.
5. Start **Excel** and open **Products.csv** in the **D:\Demofiles\Mod07** folder. Note that it contains details for three more products. Then close Excel.
6. Start **SQL Server Data Tools 2015** and open the **Checkpoints.sln** solution in the **D:\Demofiles\Mod07** folder.
7. In Solution Explorer, double-click **Load Data.dtsx**. Note that the control flow consists of a File System task to create a folder, a second File System task to copy the products file to the new folder, and a Data Flow task that loads the data in the products file into the staging table.

8. Click anywhere on the Control Flow surface to select the package, and press F4. Then in the Properties pane, set the following properties:
   - **CheckpointFileName**: D:\Demofiles\Mod07\Checkpoint.chk
   - **CheckpointUsage**: IfExists
   - **SaveCheckpoints**: True
9. Set the **FailPackageOnFailure** property for all three tasks in the control flow to **True**.
10. On the **Debug** menu, click **Start Debugging** to run the package and note that the **Load to Staging Table** task fails. Then on the **Debug** menu click **Stop Debugging**.
11. In the **D:\Demofiles\Mod07** folder, note that a file named **Checkpoint.chk** has been created, and that the File System tasks that succeeded have created a folder named **Data** and copied the **Products.csv** file into it.
12. In SQL Server Data Tools, view the **Data Flow** tab for the **Load to Staging Table** task, and double-click the **Derive Columns** transformation. Change the expression for the **NewPrice** column to **100**, and then click **OK**.
13. View the **Control Flow** tab, and then on the **Debug** menu, click **Start Debugging**. Note that the **Create Folder** and **Copy File** tasks, which succeeded previously, are not re-executed. Only the **Load to Staging Table** task is executed.
14. On the **Debug** menu, click **Stop Debugging**, and verify that the **Checkpoint.chk** file is deleted now that the package has been executed successfully.
15. In SQL Server Management Studio, select the top 1,000 rows from the **dbo.StagingTable,** and note that it now contains data about six products.
16. Close SQL Server Management Studio and SQL Server Data Tools.

Module Review and Takeaways

Question: You have an existing SSIS package containing three tasks. You want Task 3 to run if Task 1 or Task 2 fails. How can you accomplish this?

Answer: Add Failure precedence constraints from Task 1 and Task 2 to Task 3. Then configure the precedence constraints to use a Logical OR to determine whether to run Task 3.

Question: Which container should you use to perform the same task once for each file in a folder?

Answer: A Foreach Loop container.

Question: Your package includes an FTP task that downloads a large file from an FTP folder and a Data Flow task that inserts data from the file into a database. The Data Flow task may fail if the database is unavailable, in which case you plan to run the package again, after bringing the database online. How can you avoid downloading the file again when the package is re-executed?

Answer: Configure the package to use a checkpoint.

Module 8

Debugging and Troubleshooting SSIS Packages

Contents:

Lesson 1

Debugging an SSIS Package

Contents:

Question and Answers

Question: You have executed a package in Visual Studio and a task failed unexpectedly. Where can you review information about the package execution to help determine the cause of the problem?

> **Answer:** A good place to start is the Execution Results tab of the package designer, or the Output window.

Demonstration: Debugging a Package

Demonstration Steps

Add a Breakpoint

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are started, and log onto 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod08 folder, run **Setup.cmd** as Administrator. Click **Yes** when prompted.
3. At the command prompt, type **y**, and then press Enter.
4. Start Visual Studio 2015 and open the **Debugging.sln** solution in the D:\Demofiles\Mod08 folder.
5. In Solution Explorer, double-click **Debugging Demo.dtsx**. This package includes a control flow that performs the following tasks:
   - Copies a text file using a variable named **User::sourceFile** to determine the source path and a variable named **User::copiedFile** to determine the destination path.
   - Uses a data flow to extract the data from the text file, convert columns to appropriate data types, and load the resulting data into a database table.
   - Deletes the copied file.
6. On the **Debug** menu, click **Start Debugging**, note that the first task fails, and on the **Debug** menu, click **Stop Debugging**.
7. Click the **Copy Source File** task and on the **Debug** menu, click **Toggle Breakpoint**.
8. Right-click the **Copy Source File** task and click **Edit Breakpoints**. Note that you can use this dialog box to control the events and conditions for breakpoints in your package. When you toggle a breakpoint, by default it is enabled for the **OnPreExecute** event with a **Hit Count Type** value of **Always**. Then click **OK**.
9. On the **Debug** menu, click **Start Debugging**, note that execution stops at the breakpoint.

View Variables while Debugging

1. With execution stopped at the breakpoint, on the **Debug** menu, point to **Windows**, and then click **Locals**.
2. In the **Locals** pane, expand **Variables** and find the **User::copiedFile** variable. Right-click it, and then click **Add Watch**. The **Watch 1** pane is then shown with the **User::copiedFile** variable displayed.
3. Click the Locals pane, right-click the User::sourceFile variable, and then click Add Watch. The Watch 1 pane is then shown with the User::copiedFile and User::sourceFile variables displayed.
   Note that the value of the User::sourceFile variable is D:\\Demofiles\\Mod08\\Products.txt ("\" is used as an escape character, so "\\" is used to indicate a literal "\" string). In the D:\Demofiles\Mod08 folder, note that the file is actually named Products.csv.
4. On the **Debug** menu, click **Stop Debugging**.
5. On the **SSIS** menu, click **Variables**.

6. In the **Variables** window, change the value for the **sourceFile** variable to **D:\Demofiles\Mod08\Products.csv**.
7. Close the **Variables** window.
8. On the **Debug** menu, click **Start Debugging**, observe the variable values in the **Watch 1** pane.

   Note that the **sourceFile** variable now refers to the correct file.
9. On the **Debug** menu, click **Continue** and note that the **Load Data** task fails.
10. On the **Debug** menu, click **Stop Debugging**.

Enable a Data Viewer

1. Double-click the **Load Data** task to view the data flow design surface.
2. Right-click the data flow path between **Products File** and **Data Conversion**, and click **Enable Data Viewer**.
3. Double-click the data flow path between **Products File** and **Data Conversion**, and in the **Data Flow Path Editor** dialog box, click the **Data Viewer** tab. Note that you can use this tab to enable the data viewer and specify which columns should be included, and that by default, all columns are included. Then click **OK**.
4. Click the **Control Flow** tab and verify that a breakpoint is still enabled on the **Copy Source File** task. Then, on the **Debug** menu, click **Start Debugging**.
5. When execution stops at the breakpoint, on the **Debug** menu, click **Continue**.
6. When the data viewer window is displayed, resize it so you can see the data it contains, and note that the Price column for the second row contains a "-" character instead of a number.
7. In the data viewer window, click **Copy Data**. Then click the green **continue** button in the data viewer window. Close the window.
8. When execution stops because the data flow task has failed, on the **Debug** menu, click **Stop Debugging**.
9. Start Excel, and create a new blank workbook.
10. With cell A1 selected, on the **Home** tab of the ribbon, click **Paste**. Then view the data you have pasted from the data viewer.
11. Close Excel without saving the workbook, and close Visual Studio.

Lesson 2

Logging SSIS Package Events

Contents:

Question and Answers

Question: Can you think of some advantages and disadvantages of the different SSIS log providers?

Answer: There are many advantages and disadvantages to each log provider. Some are detailed below:

Windows Event Log

- Advantages
  - Very easy to configure.
  - Can be read remotely without any special configuration.
- Disadvantages
  - Log messages are mixed in with messages from other applications.
  - Windows event log needs to be sized to deal with SSIS log messages.

Text File

- Advantages
  - Generic format which can be read easily without specialist software.
- Disadvantages
  - Can be difficult to carry out detailed analysis.

XML File

- Advantages
  - Portable.
  - Easy to validate.
- Disadvantages
  - XML language is very verbose, which can lead to larger files than with other file-based log providers.

SQL Server

- Advantages
  - Very easy to analyze even large amounts of log data with SQL queries.
- Disadvantages
  - Requires a database connection.

SQL Server Profiler

- Advantages
  - Enables viewing of log data step by step.
  - Log data can be replayed on an environment.
- Disadvantages
  - Trace files can be very large and difficult to work with.

Demonstration: Logging Package Execution

Demonstration Steps

Configure SSIS Logging

1. Ensure you have completed the previous demonstration in this module.
2. Start Visual Studio and open the **Logging.sln** solution in the D:\Demofiles\Mod08 folder.
3. In Solution Explorer, double-click **Logging Demo.dtsx**.
4. On the **SSIS** menu, click **Logging**. If an error message displays, click **OK**.
5. In the **Configure SSIS Logs: Logging Demo** dialog box, in the **Provider type** list, select **SSIS log provider for Windows Event Log** and click **Add**. Then select **SSIS log provider for SQL Server** and click **Add**.
6. In the Configuration column for the SSIS log provider for SQL Server, click the drop-down arrow, and then click the (local).DemoDW connection manager.
   Note that the Windows Event Log provider requires no configuration.
7. In the **Containers** tree, select the **Logging Demo** check box, and then, with **Logging Demo** selected, on the **Providers and Logs** tab, select the for the **SSIS log provider for Windows Event Log** check box.
8. With **Logging Demo** selected, on the **Details** tab, select the **OnError** and **OnInformation** events.
9. In the **Containers** tree, clear the **Load Data** check box.
10. In the **Containers** tree, select the **Load Data** check box. This can override the inherited logging settings for the **Load Data** task.
11. With **Load Data** selected, on the **Providers and Logs** tab, select the **SSIS log provider for SQL Server** check box.
12. With **Load Data** selected, on the **Details** tab, select the **OnError** and **OnInformation** events. Then click **Advanced** and clear the **Operator** column for the two selected events. Then click **OK**.

View Logged Events

1. On the **Debug** menu, click **Start Debugging**. Then, when the **Load Data** task fails, on the **Debug** menu click **Stop Debugging**.
2. On the **SSIS** menu, click **Log Events**. This shows the events that have been logged during the debugging session (if the log is empty, re-run the package and then view the Log Events window again).
3. On the **Start** screen, type **Event Viewer** and then press Enter.
4. Expand **Windows Logs**, and click **Application**. Note the log entries with a source of **SQLISPackage130**. These are the logged events for the package.
5. Start SQL Server Management Studio and connect to the **localhost** instance of the database engine by using Windows authentication.
6. In Object Explorer, expand **Databases**, expand **DemoDW**, expand **Tables**, and expand **System Tables**. Then right-click **dbo.sysssislog** and click **Select Top 1000 Rows**.
7. View the contents of the table, noting that the **operator** column is empty.
8. Close SQL Server Management Studio without saving any files, then close Event Viewer and Visual Studio.

Lesson 3

Handling Errors in an SSIS Package

Contents:

Question and Answers

Question: In what situations might you use the package level OnError event handler?

> **Answer:** The package level OnError event handler should be used to catch errors that cannot be resolved elsewhere.

Demonstration: Handling Errors

Demonstration Steps

Implement an Event Handler

1. Ensure you have completed the previous demonstration in this module.
2. Start Visual Studio 2015 and open the **ErrorHandling.sln** solution in the D:\Demofiles\Mod08 folder.
3. In Solution Explorer, double-click **Error Handling Demo.dtsx**, and then click the **Event Handlers** tab.
4. On the **Event Handlers** tab, in the **Executable** list, ensure **Error Handling Demo** is selected, and, in the **Event handler** list, ensure **OnError** is selected. Click the hyperlink in the middle of the design surface.
5. On the **SSIS** menu, click **SSIS Toolbox**.
6. In the SSIS Toolbox, double-click **Send Mail Task**. Then, on the design surface, right-click **Send Mail Task**, click **Rename**, and change the name to **Notify administrator**.
7. Double-click **Notify administrator**.
8. In the **Send Mail Task Editor** dialog box, on the **Mail** tab, configure the following properties:
   - **SmtpConnection**: A new connection to the mia-sql.adventureworks.msft SMTP server with default settings.
   - **From**: etl@adventureworks.msft
   - **To**: administrator@adventureworks.msft
   - **Subject**: An error has occurred
9. On the **Expressions** tab, in the **Expressions** box, click the ellipsis (**…**).
10. In the **Property Expressions Editor** dialog box, in the **Property** list, click **MessageSource**, and in the **Expression** box, click the ellipsis (**…**).
11. In the **Expression Builder** dialog box, expand **Variables and Parameters**, expand **System Variables**, and drag **System::ErrorDescription** to the **Expression** box. Click **OK**.
12. In the **Property Expressions Editor** dialog box, in the row under the **MessageSource** property, in the **Property** list, click **FileAttachments**. Then in the **Expression** box, click the ellipsis (**…**).
13. In the **Expression Builder** dialog box, expand **Variables and Parameters**, and drag **User::sourceFile** to the **Expression** box. Then click **OK**.
14. In the **Property Expressions Editor** dialog box, click **OK**.
15. In the **Send Mail Task Editor** dialog box, click **OK**.
16. Click the **Control Flow** tab and then, on the **Debug** menu, click **Start Debugging**. When the **Load Data** task fails, click the **Event Handlers** tab to verify that the **OnError** event handler has been executed and then, on the **Debug** menu, click **Stop Debugging**.
17. In the **C:\inetpub\mailroot\Drop** folder, double-click the most recent email message to open it in Outlook and view its contents. Close Outlook.

Redirect Failed Rows

1. In Visual Studio, click the **Data Flow** tab, in the **Data Flow Task** drop-down list, ensure **Load Data** is selected.
2. Double-click **Data Conversion** and then, in the **Data Conversion Transformation Editor** dialog box, click **Configure Error Output**.
3. In the **Configure Error Output** dialog box, click the **Error** cell for the Numeric **ProductID** column, then hold the **Ctrl** key and click the **Error cell** for the **Numeric Price** column so that both cells are selected. In the **Set this value to the selected cells** list, click **Redirect row** and click **Apply**.
4. In the **Configure Error Output** dialog box, click **OK**.
5. In the **Data Conversion Transformation Editor** dialog box, click **OK**.
6. In the SSIS Toolbox, in the **Other Destinations** section, double-click **Flat File Destination**.
7. On the design surface, right-click **Flat File Destination**, click **Rename**, and change the name to **Invalid Rows**.
8. Move **Invalid Rows** to the right of **Data Conversion**, then click **Data Conversion** and drag the red data path from **Data Conversion** to **Invalid Rows**.
9. In the **Configure Error Output** dialog box, verify that the **Numeric ProductID** and **Numeric Price** columns both have an **Error** value of **Redirect row**, and click **OK**.
10. Double-click **Invalid Rows** and then click **New**.
11. In the **Flat File Format** dialog box, ensure **Delimited** is selected and click **OK**.
12. In the **Flat File Connection Manager Editor** dialog box, in the **Connection manager name** box, type **Invalid Rows CSV File**. In the **File name** box, type **D:\Demofiles\Mod08\InvalidRows.csv**, and click **OK**.
13. In the **Flat File Destination Editor** dialog box, click the **Mappings** tab and note that the input columns include the columns from the data flow, an **ErrorCode** column, and an **ErrorColumn** column. Click **OK**.
14. Click the **Control Flow** tab, on the **Debug** menu, click **Start Debugging**. Note that all tasks succeed, and on the **Data Flow** tab note the row counts that pass through each data flow path.
15. On the **Debug** menu, click **Stop Debugging**.
16. Start Excel and open **InvalidRows.csv** in the D:\Demofiles\Mod08 folder. View the rows that were redirected, and then close Excel without saving the workbook.

Enable DiagnosticEX Event Logging

1. On the **SSIS** menu click **Logging**.
2. In the **Configure SSIS Logs: Error Handling Demo**, in the **Provider type** menu, select **SSIS Log Provider for XML Files** and click **Add**.
3. In the **Configuration** column for the **SSIS log provider for XML Files**, click the drop-down arrow, and then click **<New connection>**.
4. In the **File Connection Manager Editor** dialog box, in the **Usage type** list, click **Create file**, then in the **File** box, type **D:\DemoFiles\Mod08\DiagnosticEX.xml**, and click **OK**.
5. In the **Containers** tree, select the **Error Handling Demo** check box, and then, with **Error Handling Demo** selected, on the **Providers and Logs** tab, select the **SSIS Log Provider for XML Files** check box.

6. With **Error Handling Demo** selected, on the **Details** tab, select the **DiagnosticEX** check box and click **OK**.
7. Click the **Control Flow** tab and on the **Debug** menu, click **Start Debugging**. Note that all tasks succeed.
8. Start Excel and open **InvalidRows.csv** in the D:\Demofiles\Mod08 folder. Note the value 10 in column E—this is the ErrorColumn value.
9. Start Word and open **DiagnosticEX.xml** in the D:\Demofiles\Mod08 folder. If a **Microsoft Word** dialog box appears, click **OK**. In the file find the text **DTS:ID="10"**. This will be followed by the text **DTS:IdentificationString="Data Conversion.Outputs[Data Conversion Output].Columns[Numeric Price]"** which details the column details for the error row.
10. Close Word, Excel.
11. In Visual Studio, stop debugging, and close without saving any changes.

Module Review and Takeaways

Question: You have configured logging with the SSIS log provider for SQL Server. Where can you view the logged event information?

Answer: The **sysssislog** system table in the database referenced by the connection manager that the log provider is configured to use.

Question: You suspect a data flow is failing because some values in a source text file are too long for the columns in the destination. How can you handle this problem?

Answer: There are a number of possible solutions, depending on how important it is to retain the full text value in the destination. If truncated values are acceptable, you could add a derived column transformation to the data flow that extracts a substring from the input column, so that the value passed to the destination is always within the required length constraint. Alternatively, you could configure the transformations and destination in the data flow to ignore truncation errors. If it is not acceptable to truncate the string, you could configure the first transformation, or destination where the issue occurs, to redirect rows containing truncation errors to an alternative destination for later examination.

Module 9

Implementing a Data Extraction Solution

Contents:

Lesson 1

Introduction to Incremental ETL

Contents:

Question and Answers

Question: What are the missing words indicated by<xx xx xx> in the following statement about a data warehouse ETL flow?

"For BI solutions that involve loading large volumes of data, a <xx xx xx> process is recommended. In this data flow architecture, the data is initially extracted to tables that closely match the source system schemas (often referred to as a landing zone)."

Answer: Three-stage ETL

Question: Which of the following questions is NOT a consideration when planning extraction windows?

( ) How much impact on business is likely during the extraction window?

( ) How frequently is new data generated in the source systems, and for how long is it retained?

( ) During what time periods are source systems least heavily used?

( ) How long does data extraction take?

( ) What latency between changes in source systems and reporting is tolerable?

Answer:

(√) How much impact on business is likely during the extraction window?

( ) How frequently is new data generated in the source systems, and for how long is it retained?

( ) During what time periods are source systems least heavily used?

( ) How long does data extraction take?

( ) What latency between changes in source systems and reporting is tolerable?

Question: Which three features does SQL Server offer for SCD control?

Answer:

SQL Server uses three methods for SCD version control:

1. **Control Data Change**.
2. **Change Tracking**.
3. **Temporal or System-Versioned Tables**.

Lesson 2

Extracting Modified Data

Contents:

Question and Answers

Question: What are the four steps used when extracting rows based on a datetime column?

Answer: The high level steps your ETL process must perform to use the high water mark technique are:

1. Note the current time.
2. Retrieve the date and time of the previous extraction from a log table.
3. Extract records where the modified date column is later than the last extraction time, but before or equal to the current time you noted in step 1. This disregards any insert or update operations that have occurred since the start of the extraction process.
4. In the log, update the last extraction date and time with the time you noted in step 1.

Question: In the following code to enable CDC in a table two system stored procedures have been omitted (<XXXXXX> and <YYYYY>). Choose the correct stored procedures from the options below.

```
IF (SELECT is_cdc_enabled FROM sys.databases WHERE name='Customers') = 'FALSE'
   BEGIN
    EXEC <xxxxxx>
    EXEC <yyyyyy>
      @source_schema       = N'dbo',
      @source_name         = N'Customers',
      @role_name           = NULL,
      @supports_net_changes = 1
   END
GO
```

( ) <XXXXXX> = sys.sp_cdc_enable_table
<YYYYYY> = sys.sp_cdc_enable_db

( ) <XXXXXX> = sys.sp_cdc_enable_db
<YYYYYY> = sys.sp_cdc_create_table

( ) <XXXXXX> = sys.sp_cdc_enable_db
<YYYYYY> = sys.sp_cdc_enable_table

( ) <XXXXXX> = sys.sp_cdc_enable_db
<YYYYYY> = sys.sp_cdc_mark_table

( ) <XXXXXX> = sys.sp_cdc_start_db
<YYYYYY> = sys.sp_cdc_add_table

Answer:

( ) <XXXXXX> = sys.sp_cdc_enable_table
<YYYYYY> = sys.sp_cdc_enable_db

( ) <XXXXXX> = sys.sp_cdc_enable_db
<YYYYYY> = sys.sp_cdc_create_table

(√) <XXXXXX> = sys.sp_cdc_enable_db
<YYYYYY> = sys.sp_cdc_enable_table

( ) <XXXXXX> = sys.sp_cdc_enable_db
<YYYYYY> = sys.sp_cdc_mark_table

( ) <XXXXXX> = sys.sp_cdc_start_db
<YYYYYY> = sys.sp_cdc_add_table

Question: What three steps would you expect to find in a data flow for extracting data from a CT-enabled data source?

Answer:

1. An SQL command that retrieves the previously extracted version from a log table and assigns it to a variable.
2. A data flow that contains a source to extract records that have been modified since the previously extracted version, and return the current version.
3. An SQL command that updates the logged version number with the current version.

Demonstration: Using a Datetime Column

Demonstration Steps

Use a Datetime Column to Extract Modified Data

1. Ensure 20767A-MIA-DC and 20767A-MIA-SQL are started, and log onto 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod09 folder, run **Setup.cmd** as Administrator. In the **User Account Control** dialog box, click **Yes**.
3. Start SQL Server Management Studio and connect to the **MIA-SQL** instance of the database engine using Windows authentication.
4. In Object Explorer, expand **Databases**, expand **DemoDW**, and expand **Tables**. Note that the database includes tables in three schemas (**dw**, **src**, and **stg**) to represent the data sources staging database, and data warehouse in an ETL solution.
5. Right-click each of the following tables and click **Select Top 1000 Rows**:
   - **stg.Products**: this table is used for staging product records during the ETL process, and is currently empty.
   - **stg.ExtractLog**: this table logs the last extraction date for each source system.
   - **src.Products**; this table contains the source data for products, including a **LastModified** column that records when each row was last modified.
6. Start Visual Studio and open the **IncrementalETL.sln** solution in the D:\Demofiles\Mod09 folder.

7. In Solution Explorer, double-click the **Extract Products.dtsx** SSIS package.
8. On the **SSIS** menu, click **Variables**, and note that the package contains two user variables named **CurrentTime** and **LastExtractTime**, and then close the window.
9. On the control flow surface, double-click **Get Current Time**.
10. In the **Expression Builder** dialog box, note that in the **Expression** pane of this task sets the **CurrentTime** user variable to the current date and time. Then click **Cancel**.
11. Double-click **Get Last Extract Time**, and note the following configuration settings. Then click **Cancel**:
    - On the **General** tab, the **ResultSet** property is set to return a single row, and the **SQLStatement** property contains a query to retrieve the maximum **LastExtractTime** value for the products source in the **stg.ExtractLog** table.
    - On the **Result Set** tab, the **LastExtractTime** value in the query results row is mapped to the **User::LastExtractTime** user variable.
12. Double-click **Staged Products** to view the data flow surface, and then double-click **Products Source** and note that the SQL command used to extract products data includes a WHERE clause that filters the query results. Then click **Parameters**, and note that the parameters in the Transact-SQL query are mapped to the **LastExtractTime** and **CurrentTime** variables.
13. Click **Cancel** in all dialog boxes, and then click the **Control Flow** tab.
14. On the control flow surface, double-click **Update Last Extract Time**, and note the following configuration settings. Then click **Cancel**:
    - On the **General** tab, the **SQLStatement** property contains a Transact-SQL UPDATE statement that updates the **LastExtractTime** in the **stg.ExtractLog** table.
    - On the **Parameter Mapping** tab, the **CurrentTime** user variable is mapped to the parameter in the Transact-SQL statement.
15. In SQL Server Management Studio, open **Modify Products.sql** in the D:\Demofiles\Mod09 folder. Then **Execute** the script to modify some rows in the **src.Products** table.
16. In Visual Studio, on the **Debug** menu, click **Start Debugging**. Then, when package execution is complete, on the **Debug** menu, click **Stop Debugging**.
17. In SQL Server Management Studio, right-click each of the following tables and click **Select Top 1000 Rows**:
    - **stg.ExtractLog**: note that the **LastExtractTime** for the **Products** data source has been updated.
    - **stg.Products**: note the rows that have been extracted from the **src.Products** table.
18. Minimize SQL Server Management Studio and Visual Studio.

Demonstration: Using Change Data Capture

Demonstration Steps

Enable CDC on the Customers Table

1. Ensure you have completed the previous demonstration in this module.
2. Maximize SQL Server Management Studio, and in Object Explorer, in the **DemoDW** database, right-click the **src.Customers** table and click **Select Top 1000 Rows**. This table contains source data for customers.

3. Open **Using CDC.sql** in the D:\Demofiles\Mod09 folder, and in the code window, select the Transact-SQL code under the comment **Enable CDC on src.Customers table**, and then click **Execute**. This enables CDC in the **DemoDW** database, and starts logging modifications to data in the **src.Customers** table.

Use CDC to Extract Modified Data

1. Select the Transact-SQL code under the comment **Insert a new customer**, and then click **Execute**. This code inserts a new customer record.
2. Select the Transact-SQL code under the comment **Make a change to a customer,** and then click **Execute**. This code updates a customer record.
3. Select the Transact-SQL code under the comment **Now see the net changes**, and then click **Execute**. This code uses CDC functions to map dates to log sequence numbers, and retrieve records in the **src.Customers** table that have been modified between the last logged extraction in the **stg.ExtractLog** table, and the current time. Two records are returned.
4. Wait 10 seconds. Then select the Transact-SQL code under the comment Check for changes in an interval with no database activity, and then click Execute.
   Because there has been no activity in the database during the specified time interval, the system returns There has been no logged database activity in the specified time interval. This demonstrates the importance of checking for a null log sequence number value when using CDC.
5. Minimize SQL Server Management Studio.

Demonstration: Using CDC Components

Demonstration Steps

Perform an Initial Extraction

1. Ensure you have completed the previous demonstrations in this module.
2. Maximize SQL Server Management Studio and open the **CDC Components.sql** script file in the D:\Demofiles\Mod09 folder.
3. Note that the script enables CDC for the **src.Shippers** table, and then click **Execute**.
4. In Object Explorer, right-click each of the following tables in the **DemoDW** database, and click **Select Top 1000 Rows** to view their contents:
   - **src.Shippers**: this table should contain four records.
   - **stg.ShipperDeletes**: this table should be empty.
   - **stg.ShipperInserts**: this table should be empty.
   - **stg.ShipperUpdates**: this table should be empty.
5. Maximize Visual Studio, in which the **IncrementalETL.sln** solution should be open, and in Solution Explorer, double-click the **Extract Initial Shippers.dtsx** SSIS package. Note that the CDC Control Tasks in the control flow contain errors, which you will resolve.
6. Double-click the **Mark Initial Load Start** CDC Control Task, and in its editor, set the following properties. Then click **OK**:
   - **SQL Server CDC database ADO.NET connection manager**: localhost DemoDW ADO NET.
   - **CDC control operation**: Mark initial load start.
   - **Variable containing the CDC state**: click **New** and create a new variable named **CDC_State**.

- **Automatically store state in a database table**: selected.
- **Connection manager for the database where the state is stored**: localhost DemoDW ADO NET.
- **Table to use for storing state**: click **New**, and then click **Run** to create the **cdc_states** table.
- **State name**: CDC_State.

7. Double-click the **Extract All Shippers** data flow task, and on the Data Flow surface, note that an ADO.NET source named **Shippers** is used to extract all rows from the **src.Shippers** table, and an ADO.NET destination named **Shipper Inserts** is used to load the extracted rows into the **stg.ShipperInserts** table.
8. On the **Control Flow** tab, double-click the **Mark Initial Load End** CDC Control Task and set the following properties. Then click **OK**:
   - **SQL Server CDC database ADO.NET connection manager**: localhost DemoDW ADO NET.
   - **CDC control operation**: Mark initial load end (make sure you do not select Mark initial load start).
   - **Variable containing the CDC state**: Select **User::CDC_State** from the list (this is the variable you created earlier).
   - **Automatically store state in a database table**: selected.
   - **Connection manager for the database where the state is stored**: localhost DemoDW ADO NET.
   - **Table to use for storing state**: [dbo].[cdc_states] (the table you created earlier).
   - **State name**: CDC_State.
9. On the **Debug** menu, click **Start Debugging** and wait for the package execution to complete. Then on the **Debug** menu, click **Stop Debugging**.
10. In SQL Server Management Studio, right-click the **Tables** folder for the **DemoDW** database and click **Refresh**. Note that a table named **dbo.cdc_states** has been created.
11. Right-click **dbo.cdc_states** and click **Select Top 1000 Rows** to view the logged **CDC_State** value (which should begin "ILEND…").
12. Right-click **stg.ShipperInserts** and click **Select Top 1000 Rows** to verify that the initial set of shippers has been extracted.

Extract Changes

1. In SQL Server Management Studio, open **Update Shippers.sql** in the D:\Demofiles\Mod09 folder.
2. Select the code under the comment **Cleanup previous extraction**, noting that it truncates the **stg.ShipperInserts** table, and click **Execute**.
3. In Visual Studio, in Solution Explorer, double-click **Extract Changed Shippers.dtsx**.
4. On the **Control Flow** tab, double-click the **Get Processing Range** CDC Control Task. Note it gets the processing range, storing it in the **CDC_State** variable and the **cdc_states** table. Then click **Cancel**.
5. Double-click the **Extract Modified Shippers** data flow task and on the **Data Flow** tab, view the properties of the **Shipper CDC Records** CDC Source component, noting that it extracts modified records, based on the range stored in the **CDC_State** variable.

6. Note that the **CDC Splitter** transformation has three outputs, one each for inserts, updates, and deletes. Each of these is connected to an ADO.NET destination that loads the records into the **stg.ShipperInserts**, **stg.ShipperUpdates**, and **stg.ShipperDeletes** tables respectively.
7. On the **Control Flow** tab, double-click the **Mark Processed Range** CDC Control Task and note it updates the **CDC_State** variable and the **cdc_states** table when the extraction is complete. Then click **Cancel**.
8. On the **Debug** menu, click **Start Debugging**. When execution is complete, stop debugging and, then double-click the **Extract Modified Shippers** data flow task and note that no rows are transferred, because the source data is unchanged since the initial extraction.
9. In SQL Server Management Studio, in the **Update Shippers.sql** script file, select the code under the comment **Modify source data**, noting that it performs an INSERT, an UPDATE, and a DELETE operation on the **src.Shippers** table, and click **Execute**.
10. In Visual Studio, click the **Control Flow** tab, and then on the **Debug** menu, click **Start Debugging**. When execution is complete, view the **Extract Modified Shippers** data flow task and note the number of rows transferred (if no rows were transferred, stop debugging and re-run the package). When three rows have been transferred (one to each output of the CDC Splitter transformation), stop debugging and close Visual Studio.
11. In SQL Server Management Studio, right-click each of the following tables and click **Select Top 1000 Rows** to view their contents. Each table should contain a single row:
    - stg.ShipperDeletes
    - stg.ShipperInserts
    - stg.ShipperUpdates
12. Minimize SQL Server Management Studio.

Demonstration: Using Change Tracking

Demonstration Steps

Enable Change Tracking

1. Ensure you have completed the previous demonstrations in this module.
2. Maximize SQL Server Management Studio, and in Object Explorer, in the **DemoDW** database, right-click the **src.Salespeople** table and click **Select Top 1000 Rows**. This table contains source data for sales employees.
3. Open **Using CT.sql** in the D:\Demofiles\Mod09 folder. Then select the Transact-SQL code under the comment **Enable Change Tracking**, and click **Execute**. This enables CT in the **DemoDW** database, and starts logging changes to data in the **src.Salespeople** table.
4. Select the Transact-SQL code under the comment **Obtain the initial data and log the current version number**, and then click **Execute**. This code uses the CHANGE_TRACKING_CURRENT_VERSION function to determine the current version, and retrieves all records in the **src.Salespeople** table.
5. Select the Transact-SQL code under the comment **Insert a new salesperson**, and then click **Execute**.
6. Select the Transact-SQL code under the comment **Update a salesperson**, and then click **Execute**.
7. Select the Transact-SQL code under the comment **Retrieve the changes between the last extracted and current versions**, and then click **Execute**.

8. In Object Explorer, in the **DemoDW** database, right-click the **src.Salespeople** table and click **Select Top 1000 Rows**.
9. Close SQL Server Management Studio.

Lesson 3

Loading Modified Data

Contents:

Question and Answers

Question: Which of these options is NOT used in a MERGE statement?

( ) WHEN NOT MATCHED

( ) USING

( ) ON

( ) WHEN MATCHED

( ) OFF

Answer:

( ) WHEN NOT MATCHED

( ) USING

( ) ON

( ) WHEN MATCHED

(√) OFF

Question: Is the following statement true or false?

"The Lookup transformation uses an in-memory cache to optimize performance."

( ) True

( ) False

Answer:

(√) True

( ) False

Demonstration: Using CDC Output Tables

Demonstration Steps

Load Data from CDC Output Tables

1. Ensure you have completed the previous demonstrations in this module.
2. Start SQL Server Management Studio and connect to the **MIA-SQL** instance of the SQL Server database engine by using Windows authentication.
3. In Object Explorer, expand **Databases**, expand **DemoDW**, and expand **Tables**. Then right-click each of the following tables and click **Select Top 1000 Rows**:
   - **dw.DimShipper**. This is the dimension table in the data warehouse.
   - **stg.ShipperDeletes**. This is the table of records that have been deleted in the source system.
   - **stg.ShipperInserts**. This is the table of new records in the source system.
   - **stg.ShipperUpdates**. This is the table of rows that have been updated in the source system.
4. Start Visual Studio and open the **IncrementalETL.sln** solution in the D:\Demofiles\Mod09 folder. Then in Solution Explorer, double-click the **Load Shippers.dtsx** SSIS package.

5. On the control flow surface, double-click the **Load Inserted Shippers** Execute SQL task. Note that the SQL Statement inserts data into **dw.DimShippers** from the **stg.ShipperInserts** table. Then click **Cancel**.
6. On the control flow surface, double-click the **Load Updated Shippers** Execute SQL task. Note that the SQL Statement updates data in **dw.DimShippers** with new values from the **stg.ShipperUpdates** table. Then click **Cancel**.
7. On the control flow surface, double-click the **Load Deleted Shippers** data flow task. On the data flow surface, note that the task extracts data from the **stg.ShipperDeletes** table, and then uses an OLE DB Command transformation to update the **Deleted** column in **dw.DimShippers** for the extracted rows.
8. On the **Debug** menu, click **Start Debugging**, and observe the control flow as it executes. When execution is complete, on the **Debug** menu, click **Stop Debugging** and minimize Visual Studio.
9. In SQL Server Management Studio, right-click the **dw.DimShipper** table and click **Select Top 1000 Rows** to review the changes that have been made.
10. Minimize SQL Server Management Studio.

Demonstration: Using the Lookup Transformation

Demonstration Steps

Use a Lookup Transformation to Insert Rows

1. Ensure you have completed the previous demonstrations in this module.
2. Maximize Visual Studio, and in Solution Explorer, double-click the **Load Geography.dtsx** SSIS package.
3. On the control flow surface, double-click **Load Geography Dimension** to view the data flow surface.
4. On the data flow surface, double-click **Staged Geography Data**. Note that the SQL command used by the OLE DB source extracts geography data from the **stg.Customers** and **stg.Salespeople** tables, and then click **Cancel**.
5. On the data flow surface, double-click **Lookup Existing Geographies** and note the following configuration settings of the Lookup transformation. Then click **Cancel**:
   - On the **General** tab, unmatched rows are redirected to the no match output.
   - On the **Connection** tab, the data to be matched is retrieved from the **dw.DimGeography** table.
   - On the **Columns** tab, the **GeographyKey** column is retrieved for rows where the input columns are matched.
6. On the data flow surface, note that the data flow arrow connecting **Lookup Existing Geographies** to **New Geographies** represents the no match data flow.
7. Double-click **New Geographies**, and note that the rows in the no match data flow are inserted into the **dw.DimGeography** table. Then click **Cancel**.
8. On the **Debug** menu, click **Start Debugging**, and observe the data flow as it executes. Note that, while four rows are extracted from the staging tables, only one does not match an existing record. The new record is loaded into the data warehouse, and the rows that match existing records are discarded. When execution is complete, on the **Debug** menu, click **Stop Debugging**.

Use a Lookup Transformation to Insert and Update Rows

1. In Visual Studio, in Solution Explorer, double-click the **Load Products.dtsx** SSIS package. Then on the control flow surface, double-click **Load Product Dimension** to view the data flow surface.
2. On the data flow surface, double-click **Staged Products**, note that the SQL command used by the OLE DB source extracts product data from the **stg.Products** table, and then click **Cancel**.
3. On the data flow surface, double-click **Lookup Existing Products** and note the following configuration settings of the Lookup transformation, then click **Cancel**:
   - On the **General** tab, unmatched rows are redirected to the no match output.
   - On the **Connection** tab, the data to be matched is retrieved from the **dw.DimProduct** table.
   - On the **Columns** tab, the **ProductKey** column is retrieved for rows where the **ProductBusinessKey** column in the staging table matches the **ProductAltKey** column in the data warehouse dimension table.
4. On the data flow surface, note that the data flow arrow connecting **Lookup Existing Products** to **Insert New Products** represents the no match data flow. The data flow arrow connecting **Lookup Existing Products** to **Update Existing Products** represents the match data flow.
5. Double-click **Insert New Products**, and note that the rows in the no match data flow are inserted into the **dw.DimProduct** table. Then click **Cancel**.
6. Double-click **Update Existing Products**, and note the following configuration settings. Then click **Cancel**:
   - On the **Connection Managers** tab, the OLE DB Command transformation connects to the **DemoDW** database.
   - On the **Component Properties** tab, the **SQLCommand** property contains a parameterized Transact-SQL statement that updates the **ProductName**, **ProductDescription**, and **ProductCategoryName** columns for a given **ProductKey**.
   - On the **Column Mapping** tab, the **ProductName**, **ProductDescription**, **ProductCategoryName**, and **ProductKey** input columns from the match data flow are mapped to the parameters in the SQL command.
7. On the **Debug** menu, click **Start Debugging**, and observe the data flow as it executes. Note the number of rows extracted from the staging tables, and how the Lookup transformation splits these rows to insert new records and update existing ones.
8. When execution is complete, on the **Debug** menu, click **Stop Debugging**. Then minimize Visual Studio.

Demonstration: Using the Merge Statement

Demonstration Steps

Use the MERGE Statement

1. In the D:\Demofiles\Mod09\LoadModifiedData folder, run **SetupB.cmd** as Administrator. In a **User Account Control** dialog box, click **Yes**.
2. In SQL Server Management Studio, in Object Explorer, right-click the **stg.SalesOrders** table and click **Select Top 1000 Rows**. This table contains staged sales order data.

3. Right-click the **dw.FactSalesOrders** table and click **Select Top 1000 Rows**. This table contains sales order fact data. Note that the staged data includes three order records that do not exist in the data warehouse fact table (with **OrderNo** and **ItemNo** values of 1005 and 1; 1006 and 1; and 1006 and 2 respectively), and one record that does exist but for which the Cost value has been modified (OrderNo 1004, ItemNo 1).
4. Open the **Merge Sales Orders.sql** file in the D:\Demofiles\Mod09\LoadModifiedData folder and view the Transact-SQL code it contains, noting the following details:
   - The MERGE statement specifies the **DemoDW.dw.FactSalesOrders** table as the target.
   - A query that returns staged sales orders and uses joins to look up dimension keys in the data warehouse is specified as the source.
   - The target and source tables are matched on the **OrderNo** and **ItemNo** columns.
   - Matched rows are updated in the target.
   - Unmatched rows are inserted into the target.
5. Click **Execute** and note the number of rows affected.
6. Right-click the **dw.FactSalesOrders** table and click **Select Top 1000 Rows**. Then compare the contents of the table with the results of the previous query you performed in step 3.
7. Minimize SQL Server Management Studio.

Demonstration: Partition Switching

Demonstration Steps

Split a Partition

1. Ensure you have completed the previous demonstration in this module (*Using the Merge Statement)*.
2. Maximize SQL Server Management Studio and open the **Load Partition.sql** file in the D:\Demofiles\LoadModifiedData folder.
3. Select the code under the comment **Create a partitioned table**, and then click **Execute**. This creates a database with a partitioned fact table, on which a columnstore index has been created.
4. Select the code under the comment **View partition metadata**, and then click **Execute**. This shows the partitions in the table with their starting and ending range values, and the number of rows they contain. Note that the partitions are shown once for each index (or for the heap if no clustered index exists). Note that the final partition (4) is for key values of 20020101 or higher and currently contains no rows.
5. Select the code under the comment **Add a new filegroup and make it the next used**, and then click **Execute**. This creates a filegroup, and configures the partition scheme to use it for the next partition to be created.
6. Select the code under the comment **Split the empty partition at the end**, and then click **Execute**. This splits the partition function to create a new partition for keys with the value 20030101 or higher.
7. Select the code under the comment **View partition metadata** again, and then click **Execute**. This time the query is filtered to avoid including the same partition multiple times. Note that the table now has three empty partitions (1, 4 and 5).

Create a Load Table

1. Select the code under the comment **Create a load table**, and then click **Execute**. This creates a table on the same filegroup as partition 4, with the same schema as the partitioned table.
2. Select the code under the comment **Bulk load new data**, and then click **Execute**. This inserts the data to be loaded into the load table (in a real solution, this would typically be bulk loaded from staging tables).
3. Select the code under the comment **Add constraints and indexes**, and then click **Execute**. This adds a check constraint to the table that matches the partition function criteria, and a columnstore index that matches the index on the partitioned table.

Switch a Partition

1. Select the code under the comment **Switch the partition**, and then click **Execute**. This switches the load table with the partition on which the value 20020101 belongs. Note that the required partition number is returned by the $PARTITION function.
2. Select the code under the comment **Clean up and view partition metadata**, and then click **Execute**. This drops the load table and returns the metadata for the partitions. Note that partition 4 now contains two rows that were inserted into the load table.
3. Close SQL Server Management Studio.

Lesson 4

Temporal Tables

Contents:

Question and Answers

Question: When you create a System-Versioned table, what is the minimum SQL Server creates in terms of tables, columns and keys?

- **Answer: Current Data Table**.
- **Historical Data Table**.
- **SysStartTime Column**.
- **SysEndTime Column**.
- **Primary Key**.

Question: Which of these options is not a sub-clause of FOR SYSTEM_TIME?

( ) AS OF <date_time>

( ) BEFORE <date_time>

( ) FROM <start_date_time> TO <end_date_time>

( ) BETWEEN <start_date_time> AND <end_date_time>

( ) CONTAINED IN (<start_date_time>, <end_date_time>)

Answer:

( ) AS OF <date_time>

(√) BEFORE <date_time>

( ) FROM <start_date_time> TO <end_date_time>

( ) BETWEEN <start_date_time> AND <end_date_time>

( ) CONTAINED IN (<start_date_time>, <end_date_time>)

Demonstration: Creating System-Versioned Tables

Demonstration Steps

Create System-Versioned Tables

1. In the D:\Demofiles\Mod09\Temporal folder, run **SetupC.cmd** as Administrator. In a **User Account Control** dialog box, click **Yes**. When prompted, press any key to continue.
2. Start SQL Server Management Studio and connect to the **MIA-SQL** instance of the database engine using Windows authentication.
3. Open the **Demo.ssmssln** solution in the D:\Demofiles\Mod09\Temporal\Demo folder.
4. If a **Microsoft SQL Server Management Studio** dialog box, appears, click **OK**.
5. In Solution Explorer, open the **6 – Temporal Tables.sql** script file.
6. Select the Transact-SQL code under the comment **Step 1: Connect to AdventureWorks**, and then click **Execute**.
7. Select the Transact-SQL code under the comment **Step 2: Create System-Versioned Table**, and then click **Execute**.

8. In Object Explorer, expand **Databases**, expand **AdventureWorks**, expand **Tables**, point out that the object name has "(System-Versioned)" in it to help identify these tables. Expand the table to show the history table located under the main table node. Expand the columns to show the tables are the same, other than the PK.
9. Select the Transact-SQL code under the comment **Step 4 - Alter Existing Table**, and then click **Execute**.
10. In Object Explorer, point out that the object name has "(System-Versioned)" in it to help identify these tables. Expand the table to show the history table located under the main table node. Expand the columns to show the tables are the same, other than the PK.
11. Select the Transact-SQL code under the comment **Step 6 - Show Dates**, and then click **Execute**.
12. Close SQL Server Management Studio. If prompted, do not save changes.

Module Review and Takeaways

Review Question(s)

Question: What should you consider when choosing between Change Data Capture and Change Tracking?

- **Answer:** CDC is available in all editions of SQL Server 2012, 2014, and 2016.
- CT is available in all editions of SQL Server 2008, 2008R2, 2012, 2014, and 2016.
- With CDC, you can record every change made to a record within a specified interval.
- With CT, you can retrieve only the most recent changes.

Question: What should you consider when deciding whether or not to use the MERGE statement to load staging data into a data warehouse?

Answer: The MERGE statement requires access to both the source and target tables, so can only be used in the following scenarios:

- The staging tables and data warehouse are co-located in the same SQL Server database.
- The staging and data warehouse tables are located in multiple databases in the same SQL Server instance, and the credentials used to execute the MERGE statement have appropriate user rights in both databases.
- The staging tables are located in a different SQL Server instance than the data warehouse, but a linked server has been defined that enables the MERGE statement to access both databases.

Additionally, you should consider whether you need to track historical changes, as this may require extremely complex Transact-SQL. In this scenario, an SCD transformation or a custom SSIS data flow may be easier to develop and maintain.

Module 10

Enforcing Data Quality

Contents:

Lesson 1

Introduction to Data Quality

Contents:

Question and Answers

Question: What are the three data quality issues that require addressing?

Answer:

Invalid Data Values

Inconsistencies

Duplicate Business Entities

Put the following steps in the correct order by numbering each one.

	Steps
	Create a free account key at the Windows Azure Marketplace.
	Subscribe to a free or paid-for RDS provider's service at the Marketplace.
	Configure the reference data service details in DQS.
	Map your domain to the RDS.
	Use the knowledge base containing the domain that maps to the RDS to cleanse the data.

Answer:

	Steps
1	Create a free account key at the Windows Azure Marketplace.
2	Subscribe to a free or paid-for RDS provider's service at the Marketplace.
3	Configure the reference data service details in DQS.
4	Map your domain to the RDS.
5	Use the knowledge base containing the domain that maps to the RDS to cleanse the data.

Demonstration: Creating a Knowledge Base

Demonstration Steps

Create a Knowledge Base

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and log into the 20767A-MIA-SQL virtual machine as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**. Then, in the D:\Demofiles\Mod10 folder, right-click **Setup.cmd** and click **Run as administrator**. If prompted to allow changes, click **Yes**. Wait for the script to finish, and then press any key to close the command prompt.
2. On the taskbar, click **SQL Server 2016 CTP3.3 Data Quality Client**. When prompted, enter the server name **MIA-SQL**, and click **Connect**.
   Please note that you may have to choose LOCAL.

3. In SQL Server Data Quality Services, in the **Knowledge Base Management** section, click **New Knowledge Base**.
4. In the **Name** box, type **Demo KB**.
5. In the **Create Knowledge Base from** list, click **Existing Knowledge Base**.
   - In the **Select Knowledge Base** list, ensure **DQS Data** is selected.
   - In the **Select Activity** section, ensure **Domain Management** is selected, and then click **Next**.
   - Select the **US - State** domain. Then, on the **Domain Properties** tab, change the domain name to **State**.
6. On the **Domain Values** tab, note the existing values. The leading value for each state is the full state name. Other possible values that should be corrected to the leading value are indented beneath each leading value.

Perform Knowledge Discovery

1. In SQL Server Data Quality Services, under **Recent Knowledge Base**, click **Demo KB**, and then click **Knowledge Discovery**.
2. On the **Map** page, in the **Data Source** drop-down list, click **Excel File**.
3. In the **Excel File** box, click **Browse**, and browse to **D:\Demofiles\Mod10**, and then double-click **Stores.xls**.
4. In the **Worksheet** drop-down list, ensure **Sheet1$** is selected, and ensure **Use first row as header** is selected. This worksheet contains a sample of store data that needs to be cleansed.
5. In the **Mappings** table, in the **Source Column** list, click **State (String)**, and in the **Domain** list, click **State**.
6. In the **Mappings** table, in the **Source Column** list, click **City (String)**, and then click the **Create a domain** icon.
7. In the **Create Domain** dialog box, in the **Domain Name** box, type **City**, and then click **OK**.
8. Repeat the previous step to map the **StoreType (String)** source column to a new domain named **StoreType**.
9. On the **Map** page, click **Next**.
10. On the **Discover** page, click **Start** and wait for the knowledge discovery process to complete. When the process has finished, note that 11 new **City** and **StoreType** records were found and that there were three unique **City** values, five unique **State** values, and four unique **StoreType** values. Then click **Next**.
11. On the **Manage Domain Values** page, with the **City** domain selected, note the new values that were discovered.
12. Select the **State** domain and note that no new values were discovered.
13. Clear the **Show Only New** check box and note that all possible values for the **State** domain are shown. Note the **Frequency** column for **California**, **CA**, and **Washington**, **WA** indicates that the data was updated.
14. Select the **StoreType** domain and note the values discovered.
15. In the **Value** column, click **Retail**, hold the CTRL key and click **Resale**, then click **Set selected domain values as synonyms**.
16. Right-click **Retail**, and click **Set as Leading**.

17. In the list of values, note that **Whalesale** has a red spelling checker line. Right-click **Whalesale**, and in the list of suggested spelling corrections, click **Wholesale**. Note that the **Type** for the **Whalesale** value changes to **Error** and the **Correct to** value is automatically set to **Wholesale**.
18. Click **Finish**. If prompted to review more values, click **No**. When prompted to publish the knowledge base, click **No**.

Perform Domain Management

1. In SQL Server Data Quality Services, under **Recent Knowledge Base**, click **Demo KB**, and then click **Domain Management**.
2. In the **Domain** list, click **StoreType**.
3. On the **Domain Values** tab, note that the values discovered in the previous task are listed with appropriate leading values and correction behavior.
4. Click Add new domain value, and then enter the value Reseller.
   Note that the Reseller value now has a star icon, as values were found in the last discovery run (hover over the star).
5. Click the **Retail** leading value, hold the CTRL key and click the new **Reseller** value. Click **Set selected domain values as synonyms** (depending on the screen resolution, may be in a drop-down list at the end of the toolbar above the table). Note that **Reseller** becomes a valid value that is corrected to the **Retail** leading value.
6. Click **Finish**, and when prompted to publish the knowledge base, click **Publish**.
7. When publishing is complete, click **OK**.

Lesson 2

Using Data Quality Services to Cleanse Data

Contents:

Question and Answers

Question: Which of these is not a mandatory part of creating a data cleansing project?

( ) Select the source containing the data to be cleansed and map its columns to the domains in the knowledge base.

( ) Export the cleansed data to a database table, comma-delimited file, or Excel workbook.

( ) Select the knowledge base to use and specify that the action to be performed is cleansing.

( ) Run the data cleansing process, then review the suggestions and corrections generated by DQS. The data steward can then approve or reject the suggestions and corrections.

( ) Create a document to describe the knowledge base.

Answer:

( ) Select the source containing the data to be cleansed and map its columns to the domains in the knowledge base.

( ) Export the cleansed data to a database table, comma-delimited file, or Excel workbook.

( ) Select the knowledge base to use and specify that the action to be performed is cleansing.

( ) Run the data cleansing process, then review the suggestions and corrections generated by DQS. The data steward can then approve or reject the suggestions and corrections.

(√) Create a document to describe the knowledge base.

Demonstration: Cleansing Data

Demonstration Steps

Create a Data Cleansing Project

1. Ensure you have completed the previous demonstration in this module.
2. In SQL Server Data Quality Services, in the **Data Quality Projects** section, click **New Data Quality Project**.
3. In the **Name** box, type **Cleansing Demo**.
4. In the **Use Knowledge Base** list, ensure **Demo KB** is selected.
5. In the **Select Activity** section, ensure **Cleansing** is selected, and then click **Next**.
6. On the **Map** page, in the **Data Source** list, ensure **SQL Server** is selected.
7. In the **Database** list, click **DemoDQS**.
8. In the **Table/View** list, click **Stores**.
9. In the **Mappings** table, perform the following mappings, and then click **Next**:

Source Column	Domain
City (varchar)	City
StoreType (varchar)	StoreType
State (varchar)	State

10. On the **Cleanse** page, click **Start**. When the cleansing process has completed, view the data in the **Profiler** tab, noting the number of corrected and suggested values for each domain, and then click **Next**.
11. On the **Manage and View Results** page, ensure that the **City** domain is selected, and on the **Suggested** tab, note that DQS has suggested correcting the value **New Yrk** to **New York**. Click the **Approve** option to accept this suggestion, and then click the **Corrected** tab to verify that the value has been corrected.
12. Click the **State** and **StoreType** domains in turn, and on the **Corrected** tab, note the corrections that have been applied, based on the values defined in the knowledge base. Then click **Next**.
13. On the **Export** page, view the output data preview.
14. In the **Export cleansing results** section, in the **Destination Type** list, click **Excel File**.
15. In the **Excel file name** box, type **D:\Demofiles\Mod10\CleansedStores.xls**, ensure that **Standardize Output** is selected, ensure that the **Data and Cleansing Info** option is selected, and click **Export**.
16. In the Exporting dialog box, when the file download has completed, click **Close**.

View Cleansed Data

17. Open **D:\Demofiles\Mod10\CleansedStores.xls** in Excel.
18. Note that the output includes the following types of column:
    - **Output**: the values for all fields after data cleansing.
    - **Source**: the original value for fields that were mapped to domains and cleansed.
    - **Reason**: the reason the output value was selected by the cleansing operation.
    - **Confidence**: an indication of the confidence DQS estimates for corrected values.
    - **Status**: the status of the output column (correct or corrected).
19. Close Excel without saving any changes.

Lesson 3

Using Data Quality Services to Match Data

Contents:

Question and Answers

Question: What are the rules for survivorship that a data steward can specify?

Answer:

- Pivot record.
- Most complete and longest record.
- Most complete record.
- Longest record.

Question: For each domain in the matching rule, the data steward defines the following settings:

Similarity: you can specify that the rule should look for similar values based on fuzzy logic comparison, or an exact match.

Weight: a percentage score to apply if a domain match succeeds.

Prerequisite: indicates that this particular domain must match for the records to be considered duplicates.

( ) True

( ) False

Answer:

(√) True

( ) False

Demonstration: Matching Data

Demonstration Steps

Create a Matching Policy

1. Ensure you have completed the previous demonstrations in this module.
2. Start the **SQL Server 2016 Data Quality Client application**, and connect to **MIA-SQL**.
3. In SQL Server Data Quality Services, under **Recent Knowledge Base**, click **Demo KB**, and then click **Matching Policy**.
4. On the **Map** page, in the **Data Source** drop-down list, select **Excel File**, in the **Excel File** box, browse to **D:\Demofiles\Mod10\Stores.xls**. In the **Worksheet** drop-down list, ensure **Sheet1$** is selected, and ensure **Use first row as header** is selected. This worksheet contains a sample of store data that needs to be matched.
5. In the **Mappings** table, perform the following mappings to existing domains:

Source Column	Domain
City (String)	City
StoreType (String)	StoreType
State (String)	State

6. In the **Mappings** table, add the following mappings by selecting a **Source Column** and creating a **New Domain**, then click **Next**. **Note**: When the **Mappings** table is full, click **Add a column mapping** to add an additional row.

Source Column	Domain
PhoneNumber (String)	PhoneNumber
StreetAddress (String)	StreetAddress
StoreName (String)	StoreName

7. On the **Matching Policy** page, click **Create a matching rule**. Then in the **Rule Details** section, change the rule name to **Is Same Store**.
8. In the **Rule Editor** table, click **Add a new domain element**. Then in the **Domain** column, ensure that **StoreName** is selected. In the **Similarity** column, ensure that **Similar** is selected, in the **Weight** column, enter **20**, and leave the **Prerequisite** column unselected.
9. Repeat the previous steps to add the following rules:
   - **StreetAddress**: Similar: 20%: Not a prerequisite.
   - **City**: Exact: 20%: Not a prerequisite.
   - **PhoneNumber**: Exact: 30%: Not a prerequisite.
   - **StoreType:** Similar: 10%: Not a prerequisite
   - **State**: Exact: 0%: Prerequisite selected.
   - Click **Start**, wait for the matching process to complete, and note that one match is detected in the sample data (Store 1 is the same as Store One). Then click **Next**.

Create a Data Matching Project

1. In SQL Server Data Quality Services, in the **Data Quality Projects** section, click **New Data Quality Project**, and create a new project named **Matching Demo** based on the **Demo KB** knowledge base. Ensure the **Matching** activity is selected, and then click **Next**.
2. On the **Map** page, in the **Data Source** list, ensure **SQL Server** is selected. Then in the **Database** list, click **DemoDQS**, and in the **Table/View** list, click **Stores**.
3. In the **Mappings** table, perform the following mappings to existing domains, then click **Next**:

Source Column	Domain
City (varchar)	City
StoreType (varchar)	StoreType
State (varchar)	State
PhoneNumber (varchar)	PhoneNumber
StreetAddress (varchar)	StreetAddress
StoreName (varchar)	StoreName

4. On the **Matching** page, click **Start**, and when matching is complete, note that two matches were detected (Store 1 is the same as Store One and Store 16 is the same as Store Sixteen). Then click **Next**.
5. On the **Export** page, in the **Destination Type** drop-down list, select **Excel File**. Then select the following content to export:
   - **Matching Results**: D:\Demofiles\Mod10\MatchedStores.xls
   - **Survivorship Results**: D:\Demofiles\Mod10\SurvivingStores.xls
6. Select the **Most complete record survivorship** rule, and click **Export**. Select **Yes** if prompted to overwrite existing files. Then when the export has completed successfully, click **Close**.

View Data Matching Results

1. Open **D:\Demofiles\Mod10\MatchedStores.xls** in Excel. Note that this file contains all the records in the dataset with additional columns to indicate clusters of matched records. In this case, there are two clusters, each containing two matches.
2. Open **D:\Demofiles\Mod10\SurvivingStores.xls** in Excel. Note this file contains the records that were selected to survive the matching process. The data has been deduplicated by eliminating duplicates and retaining only the most complete record.
3. Close Excel without saving any changes.

Module Review and Takeaways

Review Question(s)

Question: Who is responsible for ensuring the consistency and accuracy of data in solutions you have implemented or managed?

Answer: There is no single correct answer. Many organizations entrust users with particular knowledge of specific business areas, to curate data sources and perform the role of a data steward.

Module 11

Master Data Services

Contents:

Lesson 1

Introduction to Master Data Services

Contents:

Question and Answers

Question: What is the main data issue within an organization that can be resolved using Data Quality Services, and what are the advantages?

Answer: Inconsistency of data across multiple systems is a considerable issue for most organizations. Using Master Data Services to address this issue can lead to better business decisions, as more accurate and complete information is provided to the business.

Lesson 2

Implementing a Master Data Services Model

Contents:

Question and Answers

Question: What type of attribute would you create if you wanted the values to be dependent on values in another entity?

( ) File.

( ) Domain-based.

( ) An entity cannot be dependent on another entity.

( ) Free-form.

Answer:

( ) File.

(√) Domain-based.

( ) An entity cannot be dependent on another entity.

( ) Free-form.

Question: Which two attributes are automatically created for each new entity?

( ) ID and Name.

( ) Code and Name.

( ) Name and Description.

( ) Code and ID.

Answer:

( ) ID and Name.

(√) Code and Name.

( ) Name and Description.

( ) Code and ID.

Demonstration: Creating a Master Data Services Model

Demonstration Steps

Create a Model

1. Ensure that the 20767A-MIA-DC and 20767A-MIA-SQL virtual machines are both running, and log into the 20767A-MIA-SQL virtual machine as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod11 folder, right-click **Setup.cmd** and click **Run as administrator**. When prompted, click **Yes**.
3. Start Internet Explorer and browse to **http://localhost:81/MDS**.
4. On the **Master Data Services** home page, click **System Administration**.
5. On the **Manage Models** page, on the **Manage** menu, click **Models**.
6. On the **Manage Models** page, click **Add**.
7. In the **Add Model** pane, in **name** box, type **Customers**, clear the **Create entity with same name as model** check box, and then click **Save**.

Create an Entity

1. On the **Manage Models** page, click the **Customers** Model and click **Entities**.
2. On the **Manage Entities** page, in the **Model** list, click **Customers**, and then click **Add**.
3. In the **Add Entity** pane, in the **Name** box, type **Customer**, and then click **Save**.

Create Attributes

1. On the **Manage Entities** page, in the **Entity** table, click **Customer**, and then click **Attributes**.
2. On the **Manage Attributes** page, click **Add**.
3. In the **Add Attribute** pane, in the **Name** box, type **Address**, in the **Attribute Type** list, click **Free-form**, in the **Data Type** list, click **Text**, in the **Length** box, type **400**, and then click **Save**.
4. On the **Manage Attributes** page, click **Add**.
5. In the **Add Attribute** pane, in the **Name** box, type **Phone**, in the **Attribute Type** list, click **Free-form**, in the **Data Type** list, click **Text**, in the **Length** box, type **20**, and then click **Save**.
6. Click the **SQL Server 2016** text to return to the home page.

Add and Edit Members

1. On the **Master Data Services** home page, in the **Model** drop-down list, click **Customers**, and then click **Explorer**.
2. Click **Add Member**, and in the **Details** pane, enter the following data:
   - **Name**: Ben Smith
   - **Code**: 1235
   - **Address**: 1 High St, Seattle
   - **Phone**: 555 12345
   - **Annotations**: Initial data entry
3. Click **OK**, and then click the entity row you have added. Then in the **Details** pane, edit the following fields:
   - **Phone**: 555 54321
   - **Annotations**: Changed phone number
4. Click **OK**, and then click **View History**, noting the list of transactions for this entity. Click each transaction and view the text in the **Annotations** tab before clicking **Close**.
5. Close Internet Explorer.

Demonstration: Editing a Model in Excel

Demonstration Steps

Connect to a Master Data Services Model in Excel

1. Ensure you have completed the previous demonstration in this module.
2. Start **Microsoft Excel** and create a new blank document.
3. On the **File** tab, click **Options**. Then in the **Excel Options** dialog box, on the **Add-ins** tab, in the **Manage** drop-down list, select **COM Add-ins** and click **Go**.
4. In the **COM Add-ins** dialog box, if **Master Data Services Add-In for Excel** is not selected, select it. Then click **OK**.

5. On the **MASTER DATA** tab of the ribbon, in the **Connect and Load** section, click the drop-down arrow under **Connect** and click **Manage Connections**.
6. In the **Manage Connections** dialog box, click **New**, in the **Add New Connection** dialog box, enter the description **Demo MDS Server** and the MDS server address **http://localhost:81/mds**, and click **OK**. Then click **Close**.
7. On the **Master Data** tab of the ribbon, in the **Connect and Load** section, click the drop-down arrow under **Connect** and click **Demo MDS Server**.
8. In the Master Data Explorer pane, in the **Model** list, select **Customers**.
9. In the Master Data Explorer pane, click the **Customer** entity. Then on the ribbon, in the **Connect and Load** section, click **Refresh**.
   Note that the Customer entity, including the member you created in the previous demonstration, is downloaded into a new worksheet.

Add a Member

1. On the **Customer** worksheet, click cell **F4** (which should be an empty cell under Ben Smith).
2. Enter the following details in row 4:
   - **F4**: Andrew Sinclair
   - **G4**: 2600
   - **H4**: 2 Main St, Ontario
   - **I4**: 555 11111
3. Note that the row for the new member you have created has an orange background to indicate that the data has not been published to Master Data Services.
4. On the ribbon, in the **Publish and Validate** section, click **Publish**. If a **Publish and Annotate** dialog box appears, select the **Do not show this dialog box again** check box, and then click **Publish**. Note that the data is published and the orange background is removed.

Add a Free-Form Attribute to an Entity

1. On the **Customer** worksheet, click cell **N2** (which should be an empty cell to the right of the **Phone** attribute header).
2. Type **CreditLimit** and press Enter. This adds a new attribute named **CreditLimit** (which is shown with a green background because it is not yet saved to the model).
3. Click cell **N2** to re-select it, and on the **Master Data** tab of the ribbon, in the **Build Model** section, click **Attribute Properties**. Then in the **Attribute Properties** dialog box, in the **Attribute type** drop-down list, click **Number**, note the default **Decimal places** value (2), and click **OK**. Note that the changes are uploaded to the data model and the cell background changes to blue.
4. In cell **N3**, enter **1000** as the **CreditLimit** value for the **Andrew Sinclair** member, and in cell **N4**, enter **500** as the **CreditLimit** value for the **Ben Smith** member. Note that the cell background is orange to indicate that the data has not been published to Master Data Services.
5. On the ribbon, in the **Publish and Validate** section, click **Publish**.

Add a Domain-Based Attribute and Related Entity

1. On the **Customer** worksheet, click cell **O2** (which should be an empty cell to the right of the **CreditLimit** attribute header).
2. Type **AccountType** and press Enter. This adds a new attribute named **AccountType** (which is shown with a green background because it is not yet saved to the model).
3. In cell **O3**, enter **2** as the **AccountType** value for **Andrew Sinclair**. Then in cell **O4**, enter **1** as the **AccountType** value for **Ben Smith**.
4. Click cell **O2** to re-select it, and on the ribbon, in the **Build Model** section, click **Attribute Properties**. Then in the **Attribute Properties** dialog box, in the **Attribute type** drop-down list, click **Constrained list (Domain-based)**, in the **Populate the attribute with values from** list, ensure **the selected column** is selected, in the **New entity name** box, type **Account Type**, and click **OK**. Note that the **AccountType** column now contains the values **1 {1}** and **2 {2}**.
5. On the **Master Data** tab of the ribbon, in the **Connect and Load** section, click the drop-down arrow under **Connect** and click **Demo MDS Server**.
6. In the Master Data Explorer pane, in the **Model** list, select **Customers**.
7. In the Master Data Explorer pane, click the **Account Type** entity, which was created when you added a domain-based attribute to the Customer entity. Then, on the ribbon, in the **Connect and Load** section, click **Refresh**.
   Note that the Account Type entity is downloaded into a new worksheet with two members.
8. Change the **Name** attribute for the existing members as follows:

Name	Code
Standard	1
Premier	2

9. On the ribbon, in the **Publish and Validate** section, click **Publish**.
10. Click the **Customer** worksheet tab, and on the ribbon, in the **Connect and Load** section, click **Refresh**. Note that the **AccountType** attribute values are updated to show the new Name values you specified for the members in the **Account Type** entity.
11. Click cell **O3**, and note that you can select the **AccountType** value for each member from a list that looks up the related name and code in the **Account Type** entity. Leave the selected **AccountType** for Andrew Sinclair as 2 {Premier}.
12. Close Excel without saving the workbook.

Lesson 3

Hierarchies and Collections

Contents:

Question and Answers

Question: When you have created business rules in Master Data Services, how might you prevent business users from entering data which violates those rules?

Answer: You cannot prevent a user adding data that violates business rules.

Demonstration: Creating and Applying Business Rules

Demonstration Steps

Create Business Rules

1. Ensure you have completed the previous demonstrations in this module.
2. Start Internet Explorer and browse to **http://localhost:81/mds**.
3. On the **Master Data Services** home page, click **System Administration**, and then on the **Manage** menu, click **Business Rules**.
4. On the **Manage Business Rules** page, in the **Model** list, ensure that **Customers** is selected, in the **Entity** list, ensure that **Customer** is selected. In the **Member Type** list, click **Leaf**.
5. Click **Add** to create a new business rule.
6. On the **Create Business Rule** pane, in the **Name** box, type **Check Credit**.
7. In the **Description** box, type **Check that credit limit is valid**. You will use this rule to ensure that all customer credit limits are greater than or equal to zero.
8. Under **Then** click the **Add** link.
9. On the **Create Action** pane, in the **Attribute** list, click **CreditLimit**, in the **Operator** list, click **must be greater than or equal to**, in the **Must be greater than or equal to** list, click **Attribute value**, in the **Attribute value** box, type **0**, click **Save** and then on the **Create Business Rule** pane, click **Save**.
10. Click **Add** to create a new business rule.
11. In the **Create Business Rule** pane, in the **Name** box, type **Check Premier Status**.
12. In the **Description** box, type **Check account type for high credit limits**. You will use this rule to check that customers with a credit limit value greater than 1,000 have a premier account type.
13. Under **If**, click the **Add** link.
14. On the **Create Condition** pane, in the **Attribute** box, click **CreditLimit**, in the **Operator** box, click **is greater than**, in the **Is greater than** box, click **Attribute value**, in the **Attribute value** box, type **1000** and then click **Save**.
15. Under **Then**, click the **Add** link.
16. On the **Create Action** pane, in the **Attribute** list, click **AccountType**, In the **Operator** list, click **must be equal to**, in the **Must be equal to** list, click **Attribute value** and then click the **Attribute value** box.
17. On the Choose an Attribute Value pane, in the **Version** list, click **VERSION_1**, in the **Attribute value** list, click **2** and then click **Save**.

Publish Business Rules

1. On the **Manage Business Rules** page, click **Publish All**, and then in the **Message from webpage** dialog box, click **OK**.
2. In the **Business Rule State** column, check that the value displayed for both rules is **Active**.

3. On the **Manage Business Rules** page, click the **Microsoft SQL Server 2016** logo to return to the home page.

Apply Business Rules in Explorer

1. On the Master Data Services home page, click **Explorer**.
2. In the **Entities** menu, click **Customer**.
3. Click **Apply Rules**, and note that a green tick is displayed next to all valid records.
4. Click the row for the Andrew Sinclair member, and then in the Details tab, change the CreditLimit value to -200 and click OK.
   Note that a validation error message is displayed, and that the green tick for this member record changes to a red exclamation mark.
5. Change the **CreditLimit** value for Andrew Sinclair back to **500**, and click **OK**, noting that the validation message disappears and the red exclamation mark changes back to a green tick.
6. Close Internet Explorer.

Apply Business Rules in Excel

1. Start Microsoft Excel and create a new blank workbook.
2. On the **Master Data** tab of the ribbon, in the **Connect and Load** section, click the drop-down arrow under **Connect** and click **Demo MDS Server**.
3. In the Master Data Explorer pane, in the **Model** list, click **Customers**.
4. In the Master Data Explorer pane, click the **Customer** entity. Then on the ribbon, in the **Connect and Load** section, click **Refresh**. The **Customer** entity members are downloaded into a new worksheet.
5. In the ribbon, in the **Publish and Validate** section, click **Show Status**. This reveals columns that show the validation and input status for all records.
6. In the ribbon, in the **Publish and Validate** section, click **Apply Rules**. This refreshes the validation status for all rows. Currently, validation is successful for all records.
7. In cell **J4**, change the **CreditLimit** value for Ben Smith to **1200**. Note that the **AccountType** value for this record is currently 1 {Standard}.
8. In the ribbon, in the **Publish and Validate** section, click **Publish**.
9. Note that the validation status in cell **B3** is **Validation failed**.
10. Click cell **K3**, and then in the drop-down list, click **2 {Premier}** to change the **AccountType** value for Ben Smith.
11. In the ribbon, in the **Publish and Validate** section, click **Publish**.
12. Note that validation has now succeeded for the Ben Smith member. Then close Excel without saving the workbook.

Lesson 4

Creating a Master Data Hub

Contents:

Question and Answers

Question: You have a product entity; assuming default table naming was used, in which staging table would you insert leaf member data?

( ) dbo.Product_Leaf

( ) stg.Entity_Leaf

( ) stg.product_Leaf

( ) stg.Leaf_Product

Answer:

( ) dbo.Product_Leaf

( ) stg.Entity_Leaf

(√) stg.product_Leaf

( ) stg.Leaf_Product

Demonstration: Importing Master Data

Demonstration Steps

Use an SSIS Package to Import Master Data

1. Ensure you have completed the previous demonstrations in this module.
2. Start Visual Studio® and open the **MDS Import.sln** solution in the D:\Demofiles\Mod11 folder.
3. In Solution Explorer, in the **SSIS Packages** folder, double-click the **Import Customers.dtsx** SSIS package to view its control flow.
4. On the control flow surface, double-click **Load Staging Tables** to view the data flow.
5. On the data flow surface, double-click **Customers Source**, and in the **Flat File Source Editor** dialog box, on the **Columns** tab, note the columns that will be imported from the source system. Then click **Cancel**.
6. On the data flow surface, double-click **Add MDS Columns**, and in the **Derived Column Transformation Editor** dialog box, note that the transformation generates additional columns named **ImportType** (with a value of 0), **ImportStatus_ID** (with a value of 0), and **BatchTag** (with a unique string value derived from the **ExecutionInstanceGUID** system variable). Then click **Cancel**.
7. On the data flow surface, double-click **Staging Table**, and in the **OLE DB Destination Editor** dialog box, on the **Connection Manager** tab, note that the data is loaded into the **[stg].[Customer_Leaf]** table in the **MDS** database, and on the **Mappings** tab, note the column mappings. Then click **Cancel**.
8. Click the **Control Flow** tab, and on the control flow surface, double-click **Load Staged Members**. On the **General** tab, note that the **SqlStatement** property is set to execute the **stg.udp_Customer_Leaf** stored procedure with the values 'VERSION_1', 0, and a parameter, and on the **Parameter Mapping** tab, note that the **ExecutionInstanceGUID** system variable is mapped to the **@BatchTag** parameter. Then click **Cancel**.

View Import Status

1. Start Internet Explorer and browse to **http://localhost:81/mds**.
2. On the Master Data Services home page, click **Integration Management**.

3. On the **Integration** page, ensure that the **Customers** model is selected and note the batches listed. There should be a single batch for the data import you performed in the previous exercise. Note the **Started**, **Completed**, **Records**, **Status** and **Errors** values for this batch.

Validate Imported Data

1. On the Master Data Services home page, click **Explorer**.
2. In the **Entities** menu, click **Customer**. Note that nine new member records have been imported, and that their validation status is indicated by a yellow question mark.
3. Click **Apply Rules**, note that the business rules in the model are applied to all records, and that the validation status for the new records changes to a green tick for records that have passed validation, and an exclamation mark for records that have failed validation.
4. Close Internet Explorer.

Demonstration: Using Subscription Views

Demonstration Steps

Create a Subscription View

1. Ensure you have completed the previous demonstrations in this module.
2. In Internet Explorer, browse to **http://localhost:81/MDS**. Then, on the Master Data Services home page, click **Integration Management**.
3. Click **Create Views**, and on the **Subscription Views** page, click **Add** to add a subscription view.
4. On the **Subscription Views** page, in the **Create Subscription View** section, in the **name** box, type **MasterCustomers**. In the **Model** list, click **Customers**, in the **Version** list, click **VERSION_1**, in the **Entity** list, click **Customer**, in the **Format** list, click **Leaf members**, and then click **Save**.
5. Close Internet Explorer.

Query a Subscription View

1. Start SQL Server Management Studio. When prompted, connect to the **MIA-SQL** instance of SQL Server by using Windows authentication.
2. Click **New Query** and enter the following Transact-SQL in the query editor:

```
USE MDS

GO

SELECT * FROM mdm.MasterCustomers
```

3. Click **Execute** to run the query, and review the results, and then close SQL Server Management Studio without saving any items.

Module Review and Takeaways

Question: In your experience, do you think that business users in a data steward capacity will be mostly comfortable using the web-based interface, the Excel add-in, or a combination of both tools to manage master data?

Answer: There is no single correct answer.

Module 12

Extending SQL Server Integration Services

Contents:

Lesson 1

Using Scripts in SSIS

Contents:

Question and Answers

Question: You are creating a control flow script task. How do you ensure your script task can write log entries programmatically?

( ) Enable logging, and then enable the onError event handler.

( ) No special action is needed as all control flow tasks can write log entries.

( ) Enable logging, and then enable the ScriptTaskLogEntry event.

( ) Write code within the script to write to a log.

Answer:

( ) Enable logging, and then enable the onError event handler.

( ) No special action is needed as all control flow tasks can write log entries.

(√) Enable logging, and then enable the ScriptTaskLogEntry event.

( ) Write code within the script to write to a log.

Demonstration: Implementing a Script Task

Demonstration Steps

Configure a Script Task

1. Ensure 20767A-MIA-DC and 20767A-MIA-SQL are started, and log onto 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. Start Visual Studio and open the **ScriptDemo.sln** solution in the D:\Demofiles\Mod12 folder.
3. In Solution Explorer, double-click the **Package.dtsx** SSIS package to open it in the SSIS designer. Then view the control flow, and notice that it includes a script task.
4. On the **SSIS** menu, click **Logging**, then on the **Providers and Logs** tab, in the **Provider type** list, select **SSIS log provider for Windows Event Log** and click **Add**.
5. In the **Containers** tree, select the **Script Task** check box.
6. On the **Providers and Logs** tab select the **SSIS log provider for Windows Event Log** check box.
7. On the **Details** tab, select the **ScriptTaskLogEntry** check box, and click **OK**.
8. On the control flow surface, double-click **Script Task**.
9. In the **Script Task Editor** dialog box, verify that the following settings are selected:
   - **ScriptLanguage:** Microsoft Visual C# 2015
   - **EntryPoint:** Main
   - **ReadOnlyVariables:** User::Results and System::StartTime
   - **ReadWriteVariables:** User::MyVar

Implement a Script Task

1. In the **Script Task Editor** dialog box, click **Edit Script**, and wait for the Visual Studio VstaProjects editor to open. If a message that the Visual C++ Language Manager Package did not load correctly is displayed, prompting you to continue to show this error, click **No**.
2. In the **VstaProjects – Microsoft Visual Studio** window, scroll through the **ScriptMain.cs** code until you can see the whole of the **public void Main()** method. Note that the code performs the following tasks:
   - Assigns the value of the System::StartTime variable to the User::MyVar variable.
   - Writes the value of the User::MyVar variable to the log.
   - Creates a string that contains the values in the User::Results variable (which is an array of strings), and displays it in a message box.
3. Close the VstaProjects – Microsoft Visual Studio window; in the **Script Task Editor** dialog box, click **OK**.
4. On the **Debug** menu, click **Start Debugging**, and observe the package as it executes. When the **Results** message box is displayed, click **OK**.
5. When package execution has completed, on the **Debug** menu, click **Stop Debugging**; then minimize Visual Studio.
6. Right-click the **Start** button and click **Event Viewer**.
7. In Event Viewer, expand the **Windows Logs** folder, click **Application**, select the first Information event with the source **SQLISPackage130**, and in the General tab, note that it contains the start time of the package, which was written by the script component.
8. Close Event Viewer.

Demonstration: Using a Script Component in a Data Flow

Demonstration Steps

Implement a Source

1. Ensure you have completed the previous demonstration in this module.
2. Maximize Visual Studio, and view the control flow of the **Package.dtsx** SSIS package.
3. On the control flow surface, double-click **Data Flow Task** to view the data flow.
4. On the data flow surface, double-click **Script Source**.
5. In the **Script Transformation Editor** dialog box, on the Inputs and Outputs page, expand **MyOutput** and the **Output Columns** folder. Then, select **Column1** and **Column2** in turn and note the DataType property of each column.
6. In the **Script Transformation Editor** dialog box, on the Script page, click **Edit Script**, and wait for the Visual Studio VstaProjects editor to open. If a message that the Visual C++ Language Manager Package did not load correctly is displayed, prompting you to continue to show this error, click **No**.
7. In the VstaProjects – Microsoft Visual Studio window, view the script. Note that the CreateNewOutputRows method uses a loop to create six rows of data. Then close the VstaProjects – Microsoft Visual Studio window, and in the **Script Transformation Editor** dialog box, click **Cancel**.

Implement a Transformation

1. On the data flow surface, double-click **Script Transformation**.
2. In the **Script Transformation Editor** dialog box, on the Inputs and Outputs page, note that the component has a single input named **Input 0**, which consists of two columns named Column1 and Column2. There is no output defined for the component, which means that it will pass the columns in the input buffer to the output buffer without adding any new columns.
3. In the **Script Transformation Editor** dialog box, on the Input Columns page, note that both Column1 and Column2 are selected and that the Usage Type for Column1 is ReadOnly, while the Usage Type for Column2 is ReadWrite. This configuration enables the script to make changes to Column2 as it flows through the pipeline.
4. In the **Script Transformation Editor** dialog box, on the **Script page**, click **Edit Script,** and wait for the Visual Studio VstaProjects editor to open. If a message that the Visual C++ Language Manager Package did not load correctly is displayed, prompting you to continue to show this error, click **No**.
5. In the VstaProjects – Microsoft Visual Studio window, view the script. Note that the Input0_ProcessInoutRow method modifies Column2 by making its value upper case. Then close the VstaProjects – Microsoft Visual Studio window, and in the **Script Transformation Editor** dialog box, click **Cancel**.

Implement a Destination

1. On the data flow surface, double-click **Script Destination**.
2. In the **Script Transformation Editor** dialog box, on the Inputs and Outputs page, note that the component has a single input named Input 0, which consists of two columns named Column1 and Column2.
3. In the **Script Transformation Editor** dialog box, on the Input Columns page, note that both Column1 and Column2 are selected, and that the Usage Type for both columns is ReadOnly. Since the component represents a destination, there is no need to modify the rows in the buffers.
4. In the **Script Transformation Editor** dialog box, on the Script page, note that the ReadWriteVariables property gives access to the User::Results variable, and then click **Edit Script** and wait for the Visual Studio VstaProjects editor to open. If a message that the Visual C++ Language Manager Package did not load correctly is displayed, prompting you to continue to show this error, click **No**.
5. In the VstaProjects – Microsoft Visual Studio window, view the script, and note the following details:
   - The PreExecute method initializes an array variable for six string elements.
   - The Input0_ProcessInputRow method adds the value of Column2 to the next available empty element in the array.
   - The PostExecute method assigns the array variable in the script to the User::Results package variable.
6. Close the VstaProjects – Microsoft Visual Studio window, and in the **Script Transformation Editor** dialog box, click **Cancel**.
7. On the data flow surface, right-click the data flow path between Script Source and Script Transformation, and click **Enable Data Viewer**. Then repeat this step for the data flow path between Script Transformation and Script Destination.
8. On the Debug menu, click **Start Debugging**.

9. When the first data viewer window is displayed, view the rows in the pipeline and click the green **Continue** button. These rows were generated by the Script Source component.
10. When the second data viewer window is displayed, view the rows in the pipeline and click the green **Continue** button. Note that Column2 was formatted as upper case by the Script Transformation component.
11. When the Results message box is displayed, note that it contains the Column2 values that were passed to the Script Destination component. You may need to click the program icon on the taskbar to bring the message box to the front of the open windows.
12. When package execution has completed, on the Debug menu, click **Stop Debugging**. Then close Visual Studio.

Lesson 2

Using Custom Components in SSIS

Contents:

Question and Answers

Question: A software developer has developed a custom SSIS component for you which does not have its own installer. How would you install this component so it was available in SQL Server Data Tools?

Answer: Gacutil.exe command-line utility.

Module Review and Takeaways

Question: What might you consider when deciding whether to implement a custom process as a script or a custom component?

Answer: If the required process can be implemented as a script task in a control flow, or as a script source, transformation, or destination in a data flow, you might consider implementing it with a script. However, if you do not have sufficient scripting expertise, or the task is complex and likely to be required in multiple packages, you might consider looking for a commercially-available custom component or contracting a developer to create one for you.

Module 13

Deploying and Configuring SSIS Packages

Contents:

Lesson 1

Overview of SSIS Development

Contents:

Question and Answers

Question: How many different deployment models are available when using SSIS in SQL Server 2016? Give the name and a brief description of each of the models.

Answer:

There are two deployment models available when using SSIS in SQL Server 2016:

1. Package Deployment—you can deploy one or more packages at once.
2. Project Deployment—you can deploy a whole project.

Lesson 2

Deploying SSIS Projects

Contents:

Question and Answers

Question: You can add an environment variable to an SSIS Project, and assign a value to it, which may be, for example, **Test** or **Production**.

( ) True

( ) False

Answer:

(√) True

( ) False

Demonstration: Deploying an SSIS Project

Demonstration Steps

Configure the SSIS Environment

1. Ensure 20767A-MIA-DC and 30767-MIA-SQL are started, and log onto 20767A-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Demofiles\Mod13 folder, run **Setup.cmd** as Administrator. When prompted press **y**, and then press **Enter**.
3. Start SQL Server Management Studio and connect to the **localhost** database engine using Windows authentication.
4. In Object Explorer, right-click **Integration Services Catalogs**, and click **Create Catalog.**
5. In the **Create Catalog** dialog box, note that you must enable CLR integration when creating an SSIS catalog, and that you can also choose to enable automatic execution of the stored procedures used by the catalog when SQL Server starts. Then enter and confirm the password **Pa$$w0rd**, and click **OK**.
6. In Object Explorer, expand **Integration Services Catalogs**, and then right-click the **SSISDB** node that has been created, and click **Create Folder**.
7. In the **Create Folder** dialog box, in the **Folder name** box, type **Demo**, and click **OK**.
8. Expand the **SSIDB** node to see the folder, and then minimize SQL Server Management Studio.

Deploy an SSIS Project

1. Start Visual Studio, and open the **DeploymentDemo.sln** solution in the D:\Demofiles\Mod13 folder. This project contains the following two packages:
   - **Extract Login List.dtsx** – a package that uses a data flow to extract a list of logons from the master database and save them in a text file.
   - **Extract DB List.dtsx** – a package that extracts a list of databases from the **master** database and saves them in a text file.

     Both packages use a project-level connection manager to connect to the **master** database, and a project-level parameter to determine the folder where the text files containing the extracted data should be saved.
2. On the **Build** menu, click **Build Solution**.
3. When the build has succeeded, on the **Project** menu, click **Deploy**.
4. In the **Integration Services Deployment Wizard** dialog box, on the **Introduction** page, click **Next**.

5. On the **Select Destination** page, in the **Server name** box, type **localhost** and in the **Path** box, browse to the **SSISDB\Demo** folder you created earlier, and then click OK.
6. On the **Select Destination** page, click **Next**.
7. On the **Review** page, click **Deploy**. Then, when deployment has completed, click **Close** and close Visual Studio.

Create Environments and Variables

1. In SQL Server Management Studio, expand the **Demo** folder you created earlier, and expand its **Projects** folder. Note that the **DeploymentDemo** project has been deployed.
2. Right-click the **Environments** folder, and click **Create Environment**. Then in the **Create Environment** dialog box, enter the environment name **Test**, and click **OK**.
3. Repeat the previous step to create a second environment named **Production**.
4. Expand the **Environments** folder to see the environments you have created, and then right-click the **Production** environment, and click **Properties**.
5. In the **Environment Properties** dialog box, on the **Variables** page, add a variable with the following settings:
   - **Name**: DBServer
   - **Type**: String
   - **Description**: Server
   - **Value**: MIA-SQL
   - **Sensitive**: No
6. Add a second variable with the following settings (making sure to include the trailing "\" in the value), and then click **OK**:
   - **Name**: FolderPath
   - **Type**: String
   - **Description**: Folder
   - **Value**: D:\Demofiles\Mod13\Production\
   - **Sensitive**: No
7. Right-click the **Test** environment and click **Properties**.
8. In the **Environment Properties** dialog box, on the **Variables** page, add a variable with the following settings:
   - **Name**: DBServer
   - **Type**: String
   - **Description**: Server
   - **Value:** localhost
   - **Sensitive**: No

9. Add a second variable with the following settings (making sure to include the trailing "\" in the value), and then click **OK**:
   - **Name**: FolderPath
   - **Type**: String
   - **Description**: Folder
   - **Value**: D:\Demofiles\Mod13\Test\
   - **Sensitive**: No
10. In the **Projects** folder, right-click **DeploymentDemo** and click **Configure**.
11. In the **Configure** – **DeploymentDemo** dialog box, on the **References** page, click **Add** and add the **Production** environment. Then click **Add** again and add the **Test** environment.
12. In the **Configure** – **DeploymentDemo** dialog box, on the **Parameters** page, in the **Scope** list, select **DeploymentDemo**.
13. On the **Parameter**s tab, click the ellipses (**…**) button for the **OutputFolder** parameter, and in the **Set Parameter Value** dialog box, select **Use environment variable**. Click **FolderPath** in the list of variables, and click **OK**.
14. In the **Configure – DeploymentDemo** dialog box, on the **Connection Managers** tab, click the ellipses button (**…**) for the **ServerName** property, and in the **Set Parameter Value** dialog box, select **Use environment variable**. Click **DBServer** in the list of variables, and click **OK**.
15. In the **Configure – DeploymentDemo** dialog box, click **OK**.

Run an SSIS Package

1. Expand the **DeploymentDemo** package and its **Packages** folder, and then right-click **Extract DB List.dtsx** and click **Execute**.
2. In the Execute Package dialog box, select the Environment check box, and in the drop-down list, select **.\Test**. Then view the **Parameters** and Connection Managers tabs and note that the **FolderPath** and **DBServer** environment variables are used for the **OutputFolder** parameter and **ServerName** property.
3. Click **OK** to run the package. Click **No** when prompted to open the **Overview Report**.
4. Right-click **Extract Login List.dtsx** and click **Execute**.
5. In the **Execute Package** dialog box, select the **Environment** check box and in the drop-down list, select **.\Production**. Then view the **Parameters** and **Connection Managers** tabs and note that the **FolderPath** and **DBServer** environment variables are used for the **OutputFolde**r parameter and **ServerName** property.
6. Click **OK** to run the package. Click **No** when prompted to open the **Overview Report**.
7. View the contents of the D:\Demofiles\Mod13\Test folder and note that it contains a file named **DBs.csv** that was produced by the **Extract DB List.dtsx** package when it was executed in the **Test** environment.
8. View the contents of the D:\Demofiles\Mod13\Production folder and note that it contains a file named **Logins.csv** that was produced by the **Extract Login List.dtsx** package when it was executed in the **Production** environment.

View Execution Information

1. In SQL Server Management Studio, in Object Explorer, under **Integration Services Catalogs**, right-click **SSISDB**, point to **Reports**, point to **Standard Reports**, and click **Integration Services Dashboard**.
2. In the **Packages Detailed Information (Past 24 Hours)** list, notice that the two most recent package executions succeeded, and then click the **Overview** link for the first package in the list.
3. In the **Overview** report, in the **Execution Information** table, note the environment that was used, and in the **Parameters Used** table, note the values you configured with environment variables.
4. At the top of the report, click the **Navigate Backward** button to return to the **Integration Services Dashboard** report.
5. In the **Packages Detailed Information (Past 24 Hours)** list, click the **Overview** link for the second package in the list.
6. In the **Overview** report, in the **Execution Information** table, note the environment that was used, and in the **Parameters Used** table, note the values you configured with environment variables.
7. Close SQL Server Management Studio.

Lesson 3

Planning SSIS Package Execution

Contents:

Question and Answers

Question: Which of the following methods for running an SSIS package would be most appropriate if you want to schedule the package to run at regular intervals?

( ) SQL Server Management Studio

( ) Dtexec

( ) Dtexecui

( ) Windows PowerShell

( ) SQL Server Agent

Answer:

( ) SQL Server Management Studio

( ) Dtexec

( ) Dtexecui

( ) Windows PowerShell

(√) SQL Server Agent

Module Review and Takeaways

Question: What are the advantages of the project deployment model?

Answer: The project deployment model makes the process of managing your deployed project more straightforward. You can administer everything in SQL Server Management Studio, organize projects into folders, and use environments to pass values to parameters.

Question: What are the advantages of environments?

Answer: You can create multiple scenarios, such as **test** and **production**, using environments. You can assign variable values to these environments, and then switch between them with a single property edit.

Question: Where would you handle notifications?

Answer: This discussion topic has no right or wrong answer. Consider the implications of putting notifications in Jobs and SSIS projects. Consider that an SSIS package might not always run in an SQL Server Agent job, and that an SQL Server Agent job might have other steps apart from the SSIS package.

Question: Describe the new feature available in SQL Server 2016 for Deploying SSIS Packages. What are the benefits?

Answer: Selected packages within a project can be deployed using SQL Server 2016. This should save time as previously the only options were to deploy one package at a time or a whole project.

Module 14

Consuming Data in a Data Warehouse

Contents:

Lesson 1

Introduction to Business Intelligence

Contents:

Question and Answers

Which of the following are possible benefits of a data warehouse? Indicate your answer by writing the category number to the right of each item.

Items	
1	Data quality and accuracy
2	High-speed processing of transactions
3	Data availability
4	User-friendly interface for inputting core data
5	Completeness of data
6	Up-to-date data
7	Query performance
8	Accessibility of data by suitably trained non-technical users

Category 1		Category 2
Possible Benefits		Not Benefits

Answer:

Category 1		Category 2
Possible Benefits		Not Benefits
Data quality and accuracy Data availability Completeness of data Up-to-date data Query performance Accessibility of data by suitably trained non-technical users		High-speed processing of transactions User-friendly interface for inputting core data

Lesson 2

Introduction to Data Analysis

Contents:

Question and Answers

Question: Which of the following is NOT an Analysis Services data model?

( ) Multidimensional data model.

( ) Tabular data model.

( ) Entity data model.

Answer:

( ) Multidimensional data model.

( ) Tabular data model.

(√) Entity data model.

Lesson 3

Introduction to Reporting

Contents:

Question and Answers

Question: In general, only IT professionals create Reporting Services reports. True or false?

( ) True

( ) False

Answer:

( ) True

(√) False

Lesson 4

Analyzing Data with Azure SQL Data Warehouse

Contents:

Question and Answers

Question: How can you use Azure to manage and analyze data in your organization?

Answer: There is no single correct answer.

Module Review and Takeaways

Question: What are the issues you should consider when designing a data warehouse that must support self-service BI?

Answer:

Answers will vary, but some specific issues include:

Enforcing data access security.

Designing user-friendly table and column names.

Abstracting the data warehouse schema through views.

You should also consider the system resource overhead created by self-service workloads and the implications for data warehouse performance.