Lab Answer Key: Module 6: Importing and Exporting Data

Lab: Importing and Exporting Data

Exercise 1: Using the SQL Server Import and Export Wizard

Task 1: Prepare the Lab Environment

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are both running, and then log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Labfiles\Lab06\Starter folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

Task 2: Use the SQL Server Import and Export Wizard to Export Data

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. In Object Explorer, expand **Databases**. Then right-click the **InternetSales** database, point to **Tasks**, and click **Export Data**.
3. On the **Welcome to SQL Server Import and Export Wizard** page, click **Next**.
4. On the **Choose a Data Source** page, in the **Data source** drop-down list, select **SQL Server Native Client 11.0**. Then ensure that the **MIA-SQL** server is selected, that **Use Windows Authentication** is selected, and that the **InternetSales** database is selected; and click **Next**.
5. On the **Choose a Destination** page, in the **Data source** drop-down list, select **Microsoft**

Excel. Then in the **Excel file path** box type **D:\Labfiles\Lab06\Starter\Sales.xls**, ensure that **First row has column names** is selected, and click **Next**.

6. On the **Specify Table Copy or Query** page, select **Write a query to specify the data to transfer** and click **Next**.

7. On the **Provide a Source Query** page, click **Browse** and open the **Query.sql** script file in the **D:\Labfiles\Lab06\Starter** folder. Then, on the **Provide a Source Query** page, click **Next**.

8. On the **Select Source Tables and Views** page, replace **'Query'** in the **Destination** column with **'Sales'**. Then click **Next**.

9. On the **Review data Type Mapping** page, review the default mappings and click **Next.**

10. On the **Save and Run Package** page, ensure that **Run immediately** is selected, and click **Next**.

11. On the **Complete the Wizard** page, click **Finish**. Then, when the execution is successful, click **Close**.

12. Start Excel and open the **Sales.xls** file in the D:\Labfiles\Lab06\Starter folder and view the data that has been exported. Then close Excel without saving the file.

Result: After this exercise, you should have exported data from InternetSales to an Excel workbook named Sales.xls.

Exercise 2: Using the bcp Utility

Task 1: Create a Format File

1. In SQL Server Management Studio, in Object Explorer, expand the **HumanResources** database and its **Tables** folder, and then right-click the **dbo.JobCandidate** table and click **Select Top 1000 Rows**.

2. View the existing data in the table, noting that some of the columns include Unicode

characters.

3. Open a command prompt and enter the following command to create a format file:

```
bcp HumanResources.dbo.JobCandidate format nul -S MIA-SQL -T
-w  -t \t -r \n -x -f D:\Labfiles\Lab06\Starter
\JobCandidateFmt.xml
```

4. Start Notepad and open **JobCandidateFmt.xml** in the D:\Labfiles\Lab06\Starter folder. Then view the XML format file and close notepad.

Task 2: Use bcp to Import Data

1. Use Notepad to view the contents of the **JobCandidates.txt** file in the D:\Labfiles \Lab06 folder. Note that this file contains new candidate data. Then close Notepad.
2. In the command prompt window, enter the following command to import data the new candidate data into the **dbo.JobCandidate** table in the **HumanResources** database.

```
bcp HumanResources.dbo.JobCandidate in D:\Labfiles\Lab06
\Starter\JobCandidates.txt -S MIA-SQL -T -f D:\Labfiles\Lab06
\Starter\JobCandidateFmt.xml
```

3. Close the command prompt.
4. In SQL Server Management Studio, re-execute the query that retrieves the top 1000 rows from the **dbo.JobCandidate** table and verify that the new data has been imported.

Result: After this exercise, you should have created a format file named JobCandidateFmt.xml, and imported the contents of the JobCandidates.txt file into the HumanResources database.

Exercise 3: Using the BULK INSERT Statement

Task 1: Disable Indexes

1. In SQL Server Management Studio, in Object Explorer, expand the **InternetSales** database and its **Tables** folder, right-click **dbo.CurrencyRate**, and click **Select Top 1000 Rows**. Note that the table is currently empty.
2. Expand the **dbo.CurrencyRate** table, and then expand its **Indexes** folder. Note that the table has indexes defined.
3. Click **New Query**, and then in the new query pane, enter the following Transact-SQL code to disable indexes:

```
ALTER INDEX ALL ON InternetSales.dbo.CurrencyRate
DISABLE;
GO
```

4. Click **Execute**.

Task 2: Use the BULK INSERT Statement to Import Data

1. Use Excel to view the contents of the **CurrencyRates.csv** file in the M:\ folder, and note that it contains currency rate data. Then close Excel.
2. In SQL Server Management Studio, in the query pane, under the existing code to disable indexes and constraints, enter the following Transact-SQL code:

```
BULK INSERT InternetSales.dbo.CurrencyRate
FROM 'M:\CurrencyRates.csv'
WITH
      (
         FIELDTERMINATOR =',',
         ROWTERMINATOR ='\n'
```

```
);
```

3. Click **Execute** and note the number of rows affected.

4. Switch to the query pane that retrieves the top 1000 rows from the **dbo.CurrencyRate** table, remove the **Top 1000** clause from the query, and click **Execute** to run modified the SELECT query. Note that the table is now populated with the same number of rows as you noted in the previous step.

Task 3: Rebuild Indexes

1. In SQL Server Management Studio, in the query pane, under the existing code to import data, enter the following Transact-SQL code:

```
ALTER INDEX ALL ON InternetSales.dbo.CurrencyRate
REBUILD;
GO
```

2. Click **Execute**.

Result: After this exercise, you should have used the BULK INSERT statement to load data into the CurrencyRates table in the InternetSales database.

Exercise 4: Using the OPENROWSET Function

Task 1: Copy Data Files to the Server

1. Use Notepad to view the **JobCandidates2.txt** file in the D:\Labfiles\Lab06\Starter folder and note that it contains data for three candidates, only two of which have supplied email addresses. Then close Notepad without saving the file.

2. Copy the **JobCandidates2.txt** and **JobCandidatesFmt.xml** files from the D:\Labfiles \Lab06\Starter folder to the M:\ folder.

Note: In this lab environment, the client and server are the same. However, in a real environment you would need to upload data and format files from your local workstation to a volume that is accessible from the server. In this scenario, M: represents a volume in a SAN that would be accessible from the server.

Task 2: Disable Indexes and Constraints

1. In SQL Server Management Studio, in Object Explorer, under the **HumanResources** database right-click the **dbo.JobCandidate**, table and click **Select Top 1000 Rows**. Note the number of rows currently in the table.
2. Expand the **dbo.JobCandidate** table, and then expand both its **Constraints** folder and its **Indexes** folder. Note that the table has indexes and constraints defined.
3. Click **New Query**, and then in the new query pane, enter the following Transact-SQL code to disable the non-clustered indexes and constraints:

```
ALTER INDEX idx_JobCandidate_City ON
HumanResources.dbo.JobCandidate
DISABLE;
GO
ALTER INDEX idx_JobCandidate_CountryRegion ON
HumanResources.dbo.JobCandidate
DISABLE;
GO
ALTER TABLE HumanResources.dbo.JobCandidate
NOCHECK CONSTRAINT ALL;
GO
```

4. Click **Execute**.

Task 3: Use the OPENROWSET Function to Import data

1. In SQL Server Management Studio, in the query pane, under the existing code to disable indexes and constraints, enter the following Transact-SQL code:

```
INSERT INTO HumanResources.dbo.JobCandidate
SELECT * FROM OPENROWSET (BULK 'M:\JobCandidates2.txt',
FORMATFILE = 'M:\JobCandidateFmt.xml') AS rows
WHERE EmailAddress IS NOT NULL;
```

2. Click **Execute** and note the number of rows affected.

3. Switch to the query pane that retrieves the top 1000 rows from the **dbo.JobCandidate** table and click **Execute** to re-run the SELECT query. Verify that the records for candidates with an email address have been inserted.

Task 4: Re-Enable Indexes and Constraints

1. In SQL Server Management Studio, in the query pane, under the existing code to import data, enter the following Transact-SQL code:

```
ALTER INDEX idx_JobCandidate_City ON
HumanResources.dbo.JobCandidate
REBUILD;
GO
ALTER INDEX idx_JobCandidate_CountryRegion ON
HumanResources.dbo.JobCandidate
REBUILD;
GO
ALTER TABLE HumanResources.dbo.JobCandidate
CHECK CONSTRAINT ALL;
GO
```

2. Click **Execute**.

Result: After this exercise, you should have imported data from JobCandidates2.txt into the dbo.JobCandidates table in the HumanResources database.