70-761 Q&As

Querying Data with Transact-SQL

Pass Microsoft 70-761 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.lead4pass.com/70-761.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Microsoft Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

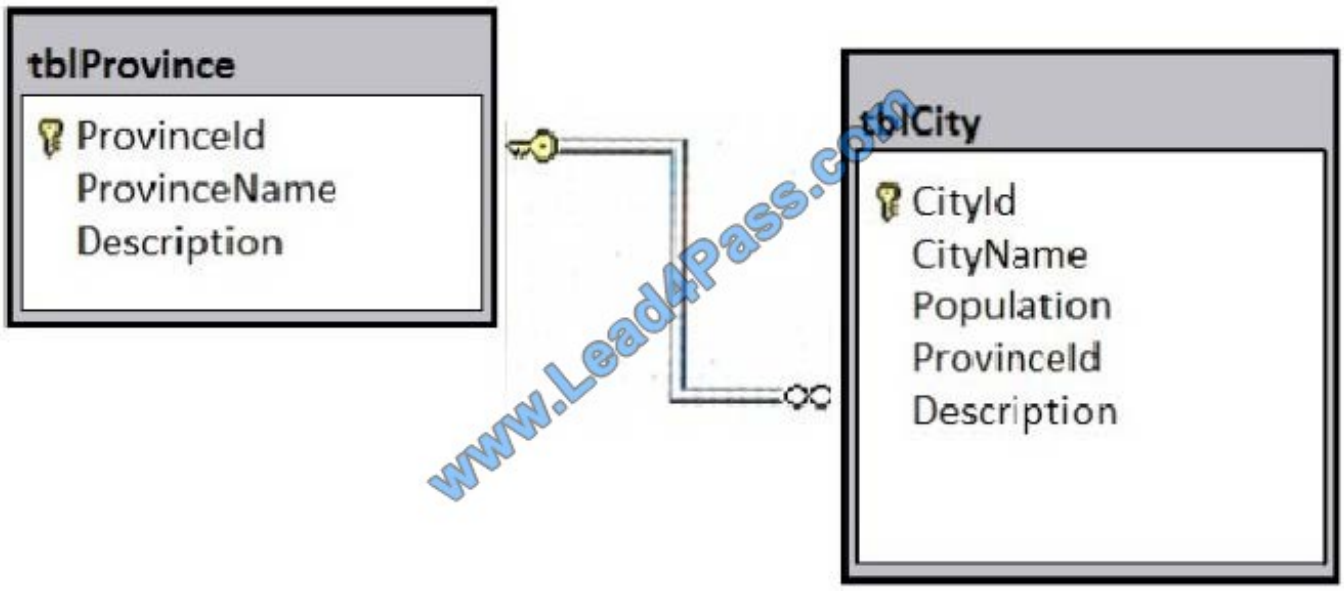
QUESTION 1

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while

others might not have a correct solution.

After you answer a question in this section. You will NOT be able to return to it. As a result, these questions will not appear in the review screen.

A database has two tables as shown in the following database diagram:



You need to list all provinces that have at least two large cities. A large city is defined as having a population of at least one million residents. The query must return the following columns: tblProvince.ProvinceId tblProvince.ProvinceName a derived column named LargeCityCount that presents the total count of large cities for the province

Solution: You run the following Transact-SQL statement:

```
SELECT P.ProvinceId, P.ProvinceName, CitySummary.LargeCityCount
FROM tblProvince P
OUTER APPLY (
   SELECT COUNT(*) AS LargeCityCount FROM tblCity C
   WHERE C.Population>=1000000 AND C.ProvinceId = P. ProvinceId
) CitySummary
```

Does the solution meet the goal?

A. Yes

B. No

Correct Answer: B

We need to list all provinces that have at least two large cities. There is no reference to this in the code.

QUESTION 2

HOTSPOT

You have the following stored procedure:

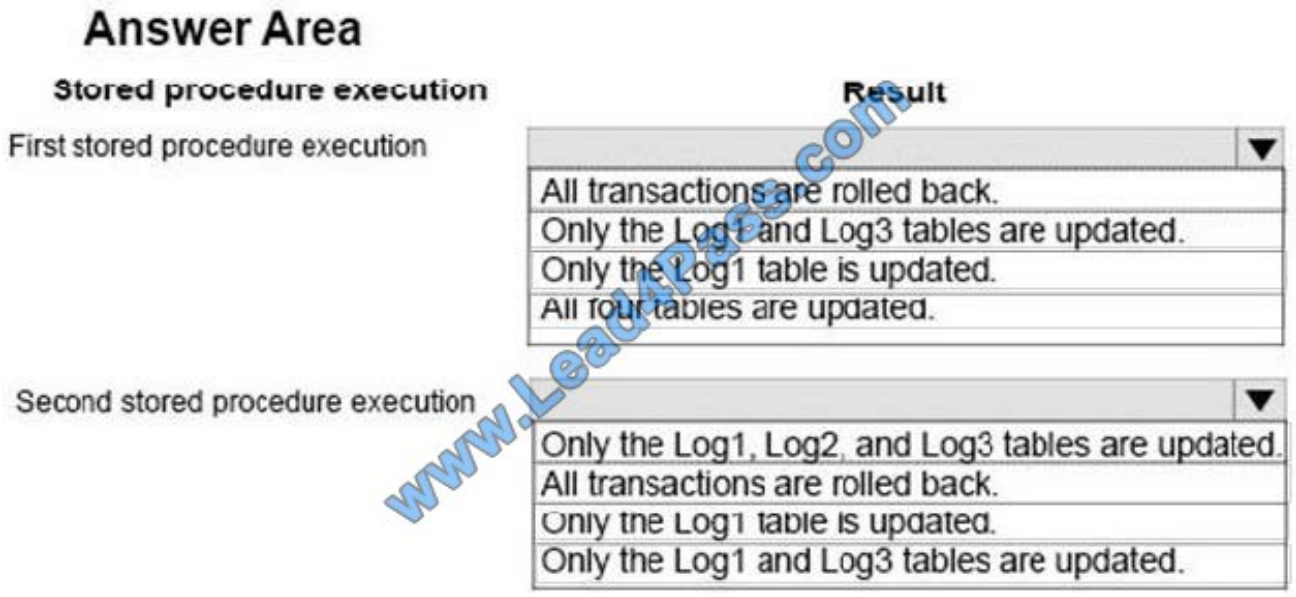
```
CREATE PROC dbo.UpdateLogs @Code char(5), @ApplicationId int, @Info varchar(1000)
AS
BEGIN
     BEGIN TRY
          BEGIN TRAN
               INSERT INTO dbo.Log1 VALUES (@Code, @ApplicationId, @Info)
               IF @Code = 'C2323 AND @ApplicationId = 1
                    RAISERROR('C2323 code from HR application!', 16, 1)
               ELSE
                    INSERT INTO dbo.Log2 VALUES (@Code, @ApplicationId, @Info)
                    INSERT INTO dbo.Log3 VALUES (@Code, @ApplicationId, @Info)
                    BEGIN TRAN
                         IF @Code = 'C2323'
                              ROLLBACK TRAN
                         ELSE
                              INSERT INTO dbo.Log4 VALUES (@Code, @ApplicationId, @Info)
                              IF @@TRANCOUNT > 0
                                   COMMIT TRAN
     END TRY
     BEGIN CATCH
          IF XACT_STATE() != 0
               ROLLBACK TRAN
     END CATCH
END
```


You run the following Transact-SQL statements:

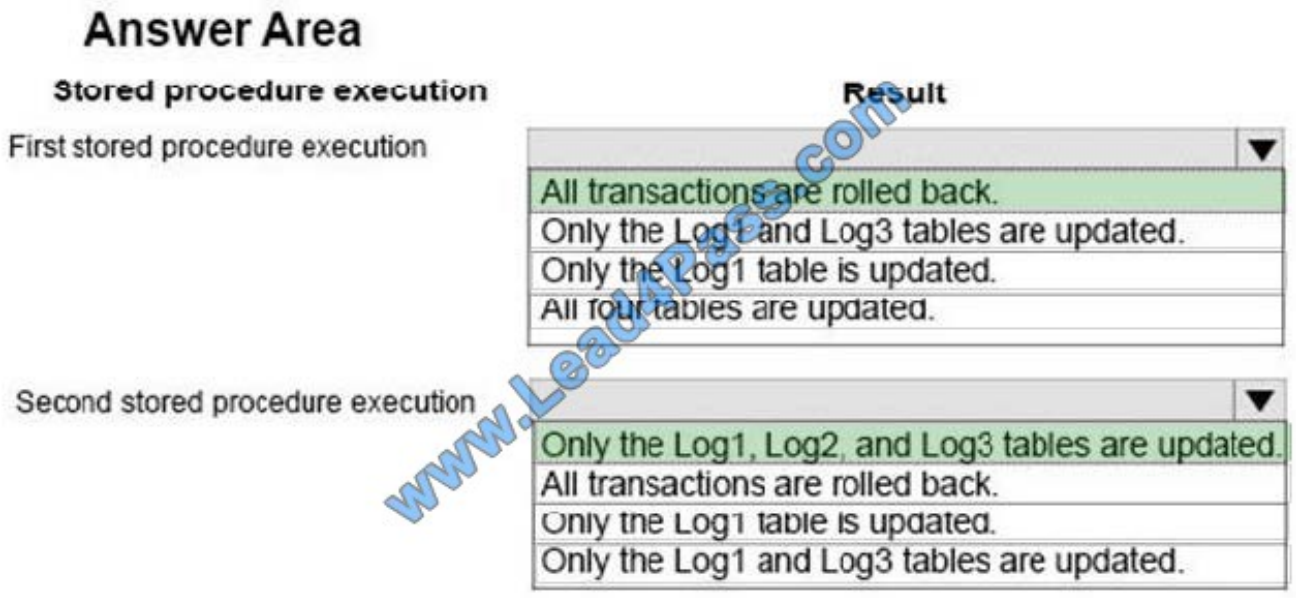
```
EXEC dbo.UpdateLogs 'C2323', 1, 'Employee records are updated.'
EXEC dbo.UpdateLogs 'C2323', 10, 'Sales process started.'
```

What is the result of each Transact-SQL statement? To answer, select the appropriate options in the answer area.

Hot Area:



Correct Answer:



Box 1: All transactions are rolled back.

The first IF-statement, IF @CODE = \\'C2323\\' AND @ApplicationID = 1, will be true, an error will be raised, the error will be caught in the CATCH block, and the only transaction that has been started will be rolled back.

Box 2: Only Log1, Log2, and Log3 tables are updated.

The second IF-statement, IF @Code = \\'C2323\\', will be true, so the second transaction will be rolled back, but log1, log2, and log3 was updated before the second transaction.

QUESTION 3

DRAG DROP

You have a project management application. The application uses a Microsoft SQL Server database to store data. You are developing a software bug tracking add-on for the application.

The add-on must meet the following requirements:

Allow case sensitive searches for product.

Filter search results based on exact text in the description.

Support multibyte Unicode characters.

```
CREATE TABLE Bug (
      Id UNIQUEIDENTIFIER NOT NULL,
      Product NVARCHAR(255) NOT NULL,
      Description NVARCHAR(max) NOT NULL,
      DateCreated DATETIME NOT NULL,
      ReportingUser VARCHAR(50) NULL
)
```

You run the following Transact-SQL statement:

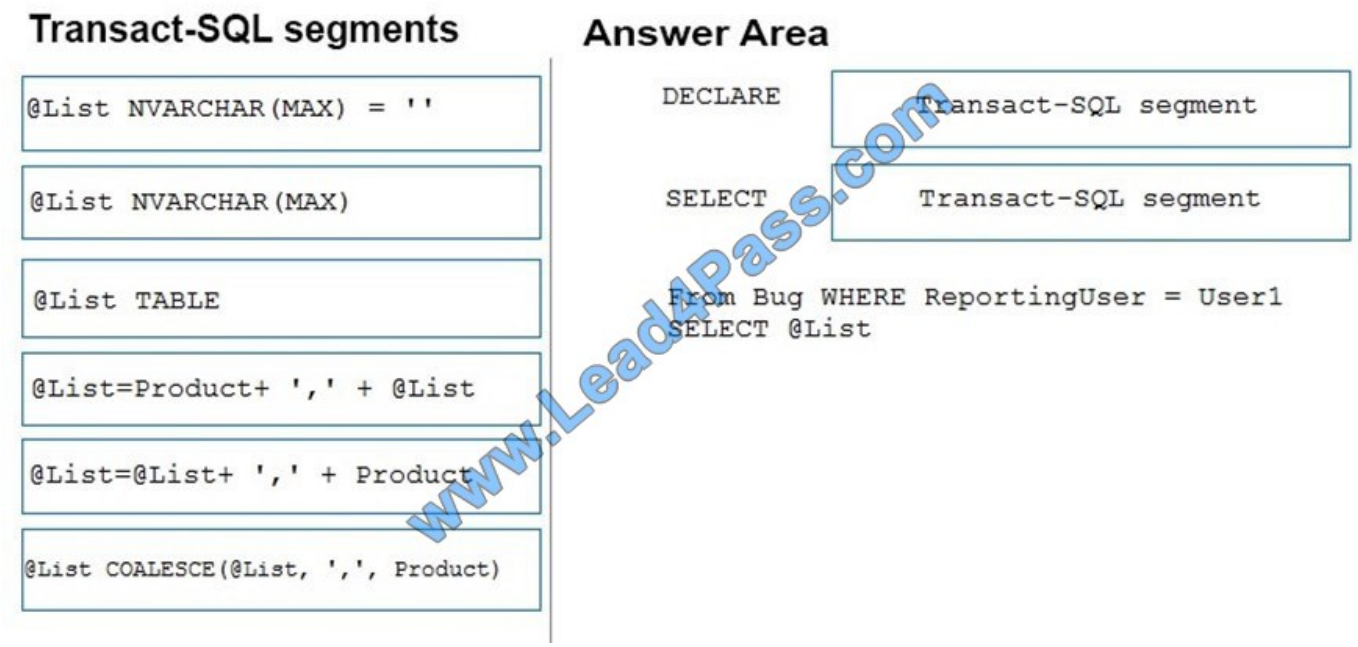
You need to display a comma separated list of all product bugs filed by a user named User1.

How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct locations. Each Transact-SQL segment may be used once, more than once, or not at all. You may need to drag

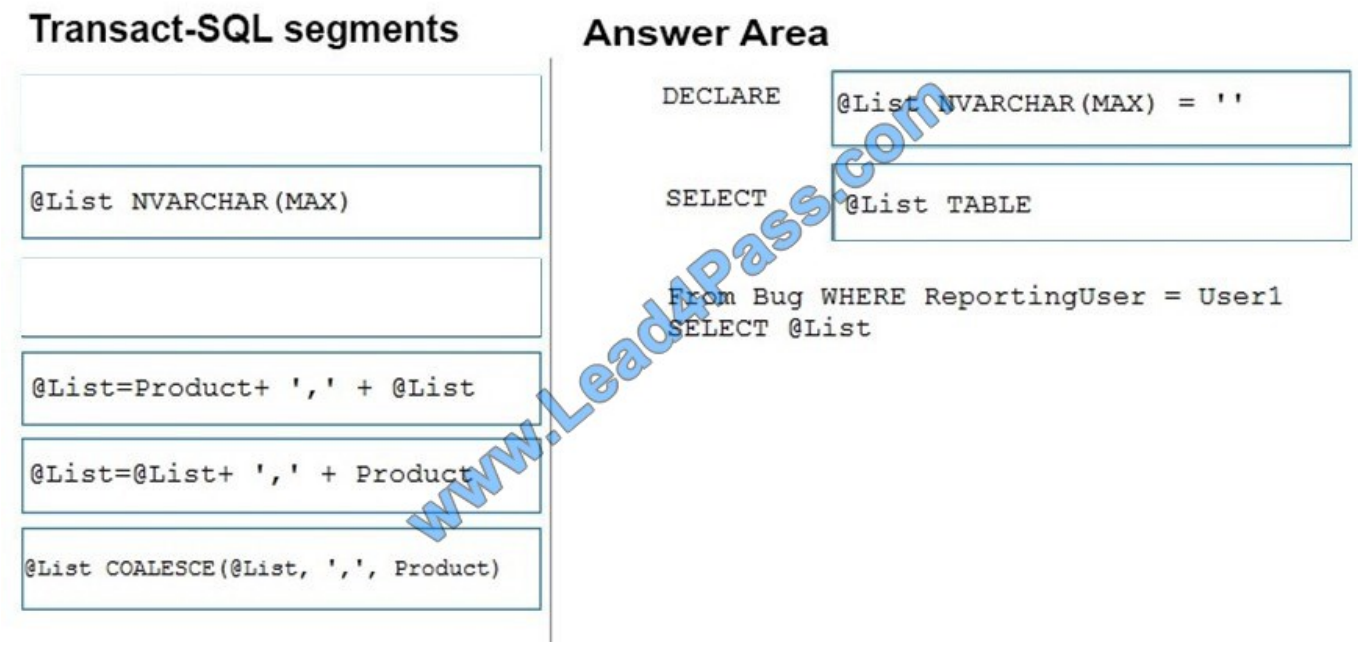
the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

Select and Place:



Correct Answer:



References: https://docs.microsoft.com/en-us/sql/t-sql/functions/string-split-transact-sql?view=sql-server-2017

QUESTION 4

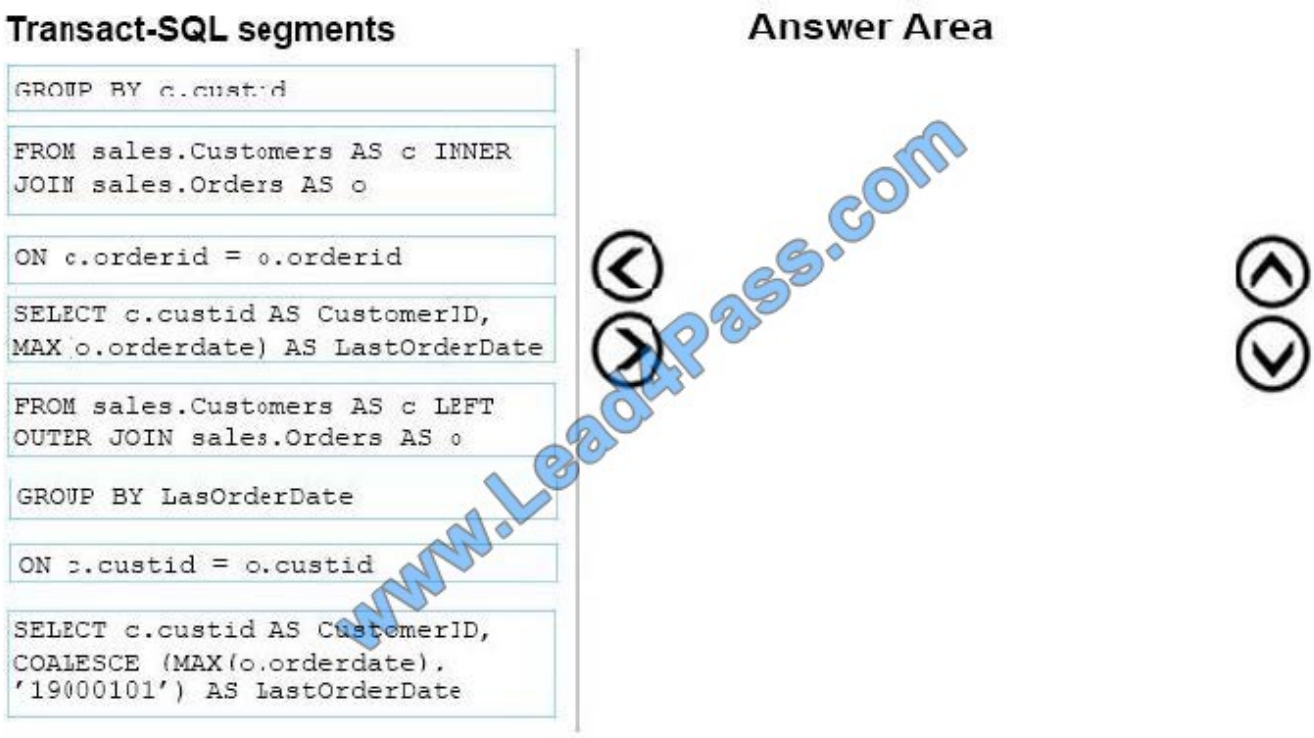
DRAG DROP

You have a database that includes the following tables:

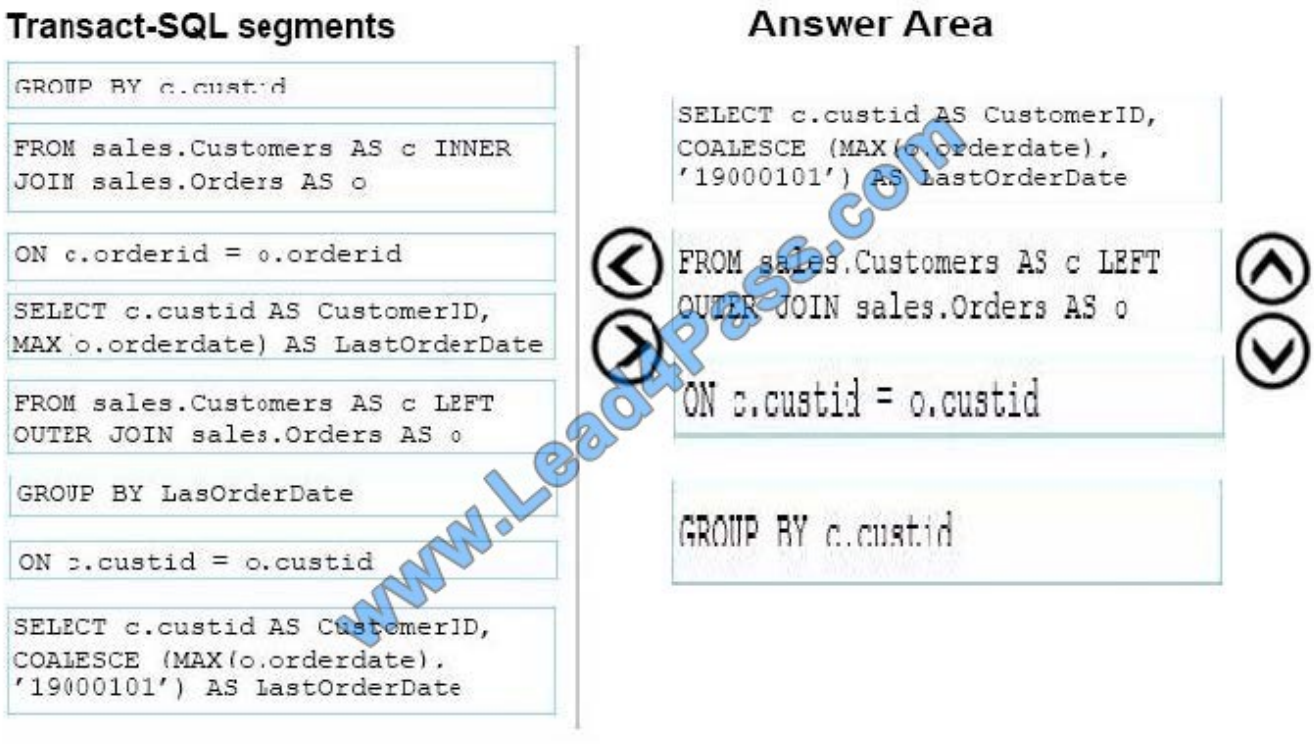
You need to create a list of all customer IDs and the date of the last order that each customer placed. If the customer has not placed any orders, you must return the date January 1, 1900. The column names must be CustomerID and LastOrderDate.

Which four Transact-SQL segments should you use to develop the solution? To answer, move the appropriate Transact-SQL segments from the list of Transact-SQL segments to the answer area and arrange them in the correct order.

Select and Place:



Correct Answer:



Box 1: SELECT..COALESCE...

The COALESCE function evaluates the arguments in order and returns the current value of the first expression that initially does not evaluate to NULL.

Box 2: ..LEFT OUTER JOIN..

The LEFT JOIN (LEFT OUTER JOIN) keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match. A customer might have no orders so the

right table must be allowed have a NULL value.

Box 3: ON c.custid = o.custid

We JOIN on the custID column, which is available in both tables.

Box 4: GROUP BY c.custid

References:

https://technet.microsoft.com/en-us/library/ms189499(v=sql.110).aspx

http://www.w3schools.com/sql/sql_join_left.asp

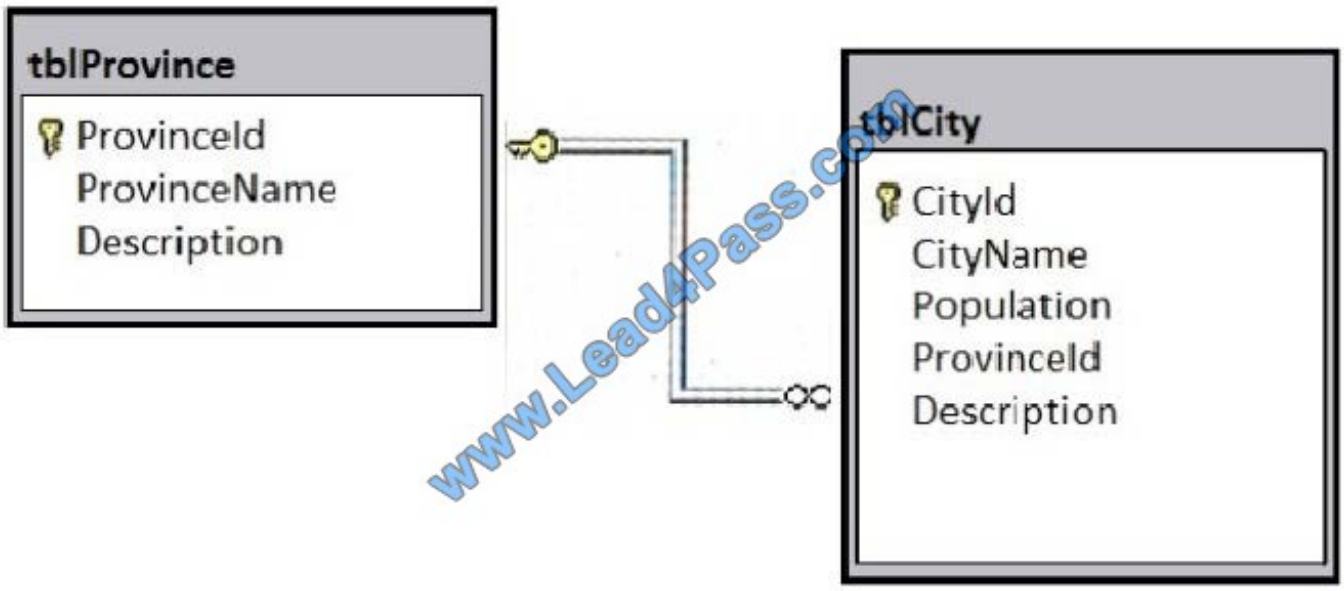
QUESTION 5

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while

others might not have a correct solution.

After you answer a question in this section. You will NOT be able to return to it. As a result, these questions will not appear in the review screen.

A database has two tables as shown in the following database diagram:



You need to list all provinces that have at least two large cities. A large city is defined as having a population of at least one million residents. The query must return the following columns:

tblProvince.ProvinceId

tblProvince.ProvinceName

a derived column named LargeCityCount that presents the total count of large cities for the province

Solution: You run the following Transact-SQL statement:

```
SELECT P.ProvinceId, P.ProvinceName, CitySummary.LargeCityCount
FROM tblProvince P
CROSS APPLY (
   SELECT COUNT(*) AS LargeCityCount FROM tblCity C
   WHERE C.Population>=1000000 AND C.ProvinceId = P.ProvinceId
) CitySummary
WHERE CitySummary.LargeCityCount >=2
```

Does the solution meet the goal?

A. Yes

B. No

Correct Answer: A

The requirement to list all provinces that have at least two large cities is meet by the WHERE CitySummary.LargeCityCount >=2 clause.

CROSS APPLY willwork fine here.

Note:

The APPLY operator allows you to invoke a table-valued function for each row returned by an outer table expression of a query. The table-valued function acts as the right input and the outer table expression acts as the left input. The right

input is evaluated for each row from the left input and the rows produced are combined for the final output. The list of columns produced by the APPLY operator is the set of columns in the left input followed by the list of columns returned by

the right input.

There are two forms of APPLY: CROSS APPLY and OUTER APPLY. CROSS APPLY returns only rows from the outer table that produce a result set from the table-valued function. OUTER APPLY returns both rows that produce a result set,

and rows that do not, with NULL values in the columns produced by the table-valued function.

References: https://technet.microsoft.com/en-us/library/ms175156(v=sql.105).aspx