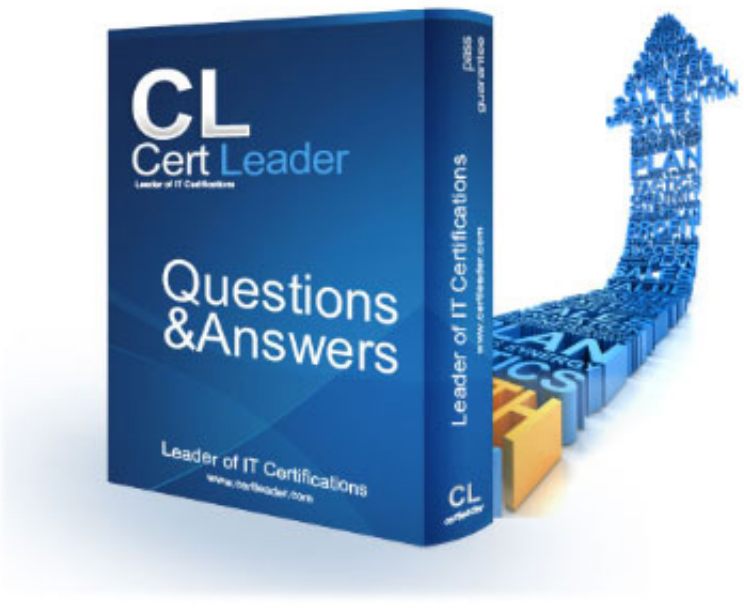
70-761 Dumps

Querying Data with Transact-SQL (beta)

https://www.certleader.com/70-761-dumps.html



NEW QUESTION 1
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You have a database that contains tables named Customer_CRMSystem and Customer_HRSystem. Both tables use the following structure:

Column name	Data type	Allow null
CustomerID	int	No
CustomerCode	char(4)	Yes
CustomerName	varchar(50)	No

The tables include the following records: Customer_CRMSystem

CustomerID	CustomerCode	CustomerName
1	CUS1	Roya
2	CUS9	Yossi
3	CUS4	Jack
4	NULL	Jane
5	NULL	Francisco

Customer_HRSystem

CustomerID	CustomerCode	CustomerName
1	CUS1	Roya
2	CUS2	Jose
3	CUS9	Yossi
4	NULL	Jane

Records that contain null values for CustomerCode can be uniquely identified by CustomerName. You need to display distinct customers that appear in both tables.
Which Transact-SQL statement should you run?

A

```
SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
FROM Customer_CRMSystem c
INNER JOIN Customer_HRSystem h
ON c.CustomerCode = h.CustomerCode AND c.CustomerName = h.CustomerName
```

B

```
SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
INTERSECT
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem
```

C

```
SELECT c.CustomerCode, c.CustomerName
FROM Customer_CRMSystem c
LEFT OUTER JOIN Customer_HRSystem h
ON c.CustomerCode = h.CustomerCode
WHERE h.CustomerCode IS NULL AND c.CustomerCode IS NOT NULL
```

D

```
SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
EXCEPT
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem
```

E

```
SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
UNION
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem
```

F

```
SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
UNION ALL
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem
```

G

```
SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
FROM Customer_CRMSystem c
CROSS JOIN Customer_HRSystem h
```

H

```
SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
FROM Customer_CRMSystem c
FULL OUTER JOIN Customer_HRSystem h
ON c.CustomerCode = h.CustomerCode AND c.CustomerName = h.CustomerName
```

A. Option A
B. Option B
C. Option C
D. Option D
E. Option E
F. Option F
G. Option G
H. Option H

Answer: H

Explanation:
To retain the nonmatching information by including nonmatching rows in the results of a join, use a full outer join. SQL Server provides the full outer join operator, FULL OUTER JOIN, which includes all rows from both tables, regardless of whether or not the other table has a matching value.

NEW QUESTION 2
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.
You have a table named Products that stores information about products your company sells. The table has a column named ListPrice that stores retail pricing information for products.
Some products are used only internally by the company. Records for these products are maintained in the Products table for inventory purposes. The price for each of these products is $0.00. Customers are not permitted to order these products.
You need to increase the list price for products that cost less than $100 by 10 percent. You must only increase pricing for products that customers are permitted to order.
Solution: You run the following Transact-SQL statement:

```
UPDATE Production.Product
SET ListPrice = ListPrice + 1.1
WHERE ListPrice
BETWEEN 0 and 100
```

Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:
Products with a price of $0.00 would also be increased.

NEW QUESTION 3
You have a table named Table1 that contains 200 million rows. Table1 contains a column named SaleDate that has a data type of DateTime2(3).
Users report that the following query runs slowly.

```
Select SalesPerson, count(*)
FROM table1
Where year(SaleDate) = 2017
GROUP BY SalesPerson
```

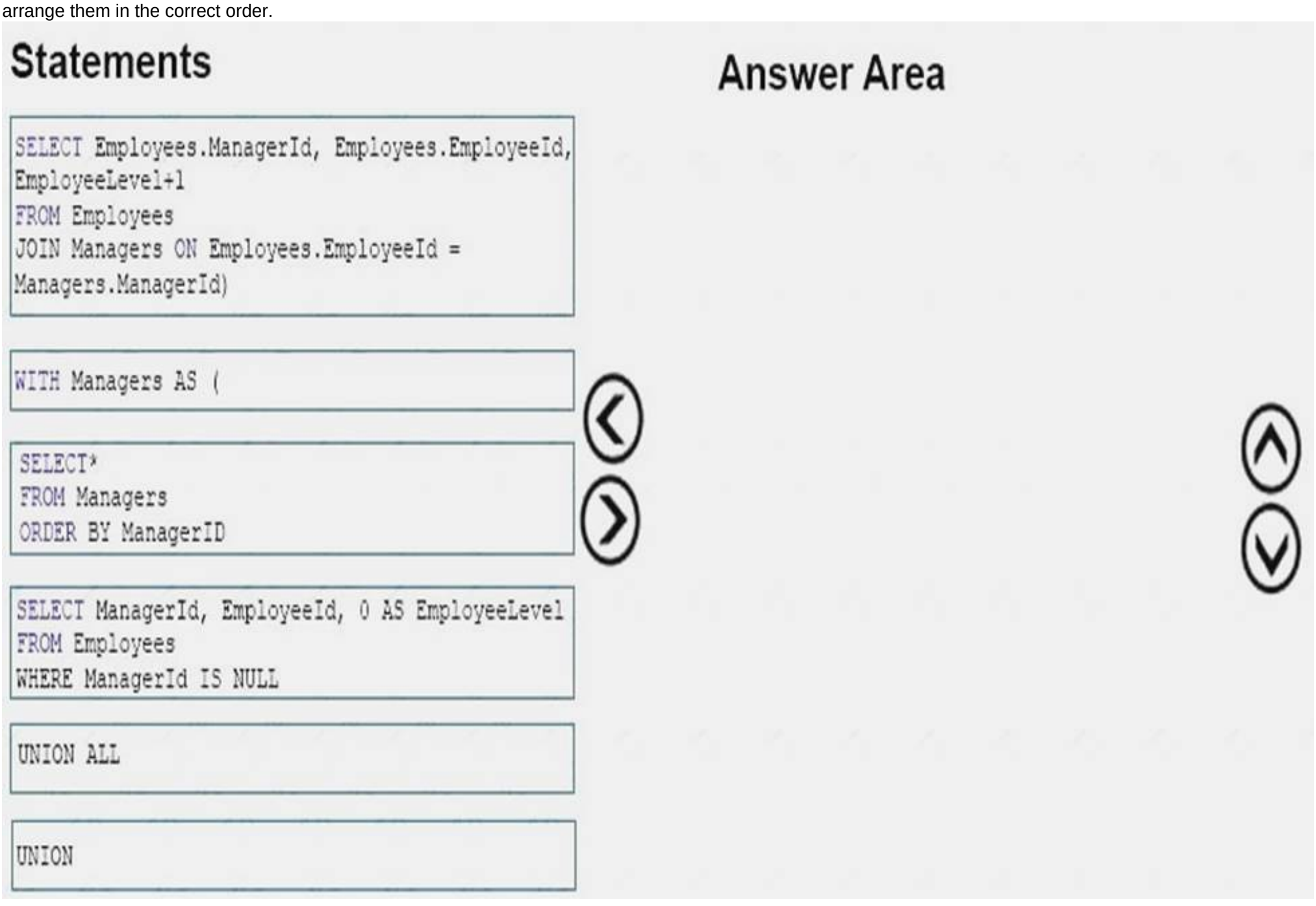
You need to reduce the amount of time it takes to run the query. What should you use to replace the WHERE statement?

A. WHERE SaleDate >= '2017-01-01' AND SaleDate < '2018-01-01'
B. WHERE cast(SaleDate as varchar(10)) BETWEEN '2017-01-01' AND '2017-12-31'
C. WHERE cast(SaleDate as date) BETWEEN '2017-01-01' AND '2017-12-31'
D. WHERE 2017 = year(SaleDate)

Answer: C

Explanation:
References: https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?view=sql-server-2017

NEW QUESTION 4
You have a database named DB1 that contains a table named HR.Employees. HR.Employees contains two columns named ManagerID and EmployeeID.
ManagerID refers to EmployeeID.
You need to create a query that returns a list of all employees, the manager of each employee, and the numerical level of each employee in your organization's hierarchy.
Which five statements should you add to the query in sequence? To answer, move the appropriate statements from the list of statements to the answer area and arrange them in the correct order.



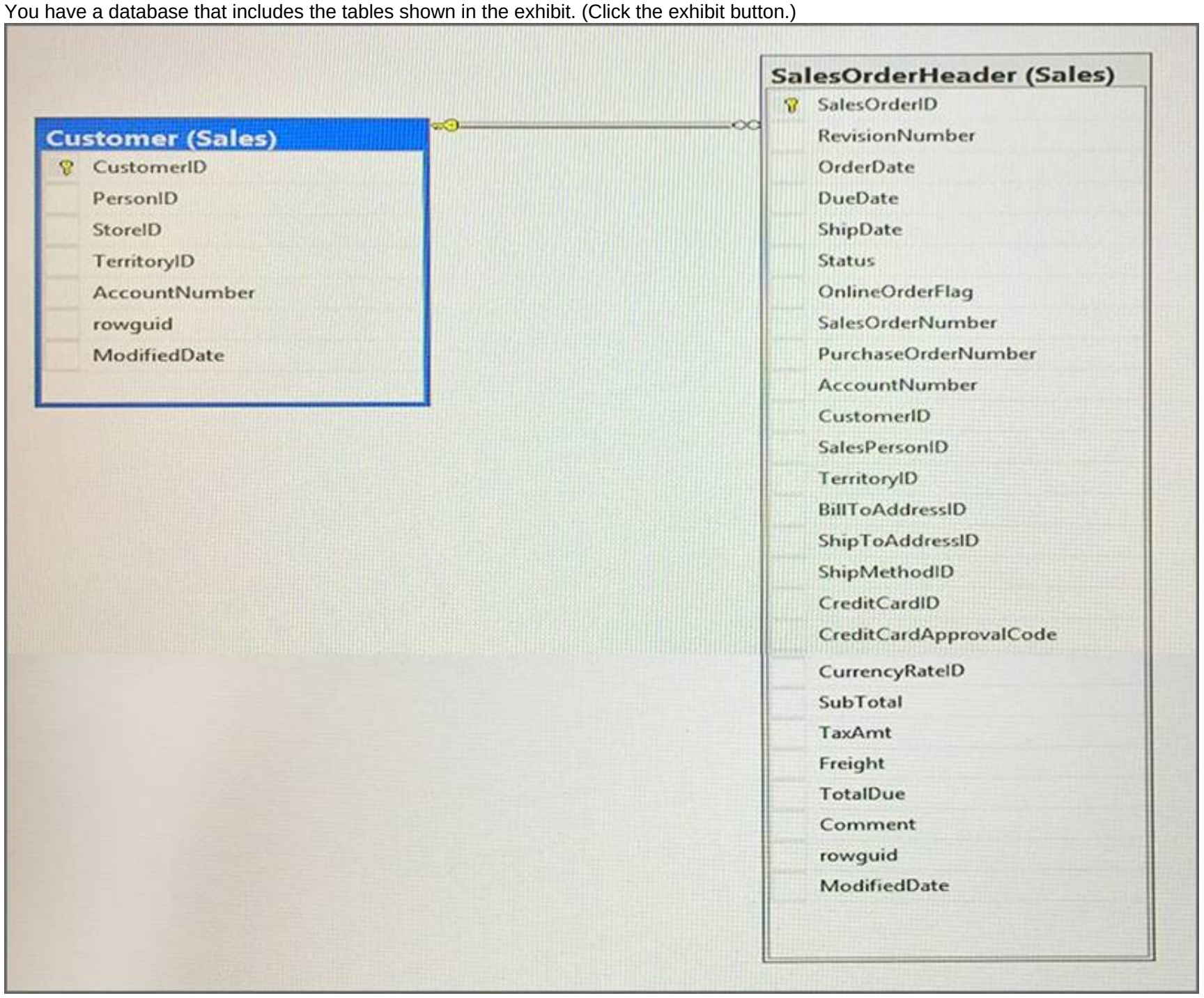
A. Mastered
B. Not Mastered

Answer: A

Explanation:
References:
https://blog.sqlauthority.com/2012/04/24/sql-server-introduction-to-hierarchical-query-using-a-recursive-cte-a-p

NEW QUESTION 5

You have a database that includes the tables shown in the exhibit. (Click the exhibit button.)



You need to create a list of all customers, the order ID for the last order that the customer placed, and the date that the order was placed. For customers who have not placed orders, you must substitute a zero for the order ID and 01/01/1990 for the date.
Which Transact-SQL statement should you run?

A
```
SELECT C.CustomerID, ISNULL(SOH.SalesOrderID, 0) AS OrderID,ISNULL(MAX(OrderDate), '')
FROM Sales.Customer C LEFT OUTER JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID, SOH.SalesOrderID
ORDER BY C.CustomerID
```

B
```
SELECT C.CustomerID, SOH.SalesOrderID, MAX(OrderDate)
FROM Sales.Customer C INNER JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID, SOH.SalesOrderID
ORDER BY C.CustomerID
```

C
```
SELECT C.CustomerID, SOH.SalesOrderID, MAX(OrderDate)
FROM Sales.Customer C CROSS JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID, SOH.SalesOrderID
ORDER BY C.CustomerID
```

D
```
SELECT C.CustomerID, SOH.SalesOrderID, MAX(OrderDate)
FROM Sales.Customer C RIGHT OUTER JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID, SOH.SalesOrderID
ORDER BY C.CustomerID
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

Explanation:
ISNULL Syntax: ISNULL (check_expression , replacement_value) author:"Luxemburg, Rosa"
The ISNULL function replaces NULL with the specified replacement value. The value of check_expression is returned if it is not NULL; otherwise, replacement_value is returned after it is implicitly converted to the type of check_expression.
References: https://msdn.microsoft.com/en-us/library/ms184325.aspx

NEW QUESTION 6
You have a database that includes the following tables. HumanResources.Employee

Column	Data type	Notes
BusinessEntityID	int	primary key

Sales.SalesPerson

Column	Data type	Notes
BusinessEntityID	int	primary key
CommissionPct	smallmoney	does not allow null values

The HumanResources.Employee table has 2,500 rows, and the Sales.SalesPerson table has 2,000 rows.
You review the following Transact-SQL statement:

```
SELECT e.BusinessEntityID
FROM HumanResources.Employee AS e
WHERE 0.015 IN
      (SELECT CommissionPct
      FROM Sales.SalesPerson AS sp
      WHERE e.BusinessEntityID = sp.BusinessEntityID)
```

You need to determine the performance impact of the query.
How many times will a lookup occur on the primary key index on the Sales.SalesPerson table?

A. 200
B. 2,000
C. 2,500
D. 5,500

Answer: C

NEW QUESTION 7
You have a database that contains a table named Products in the sales schema. The table was created by running the following Transact-SQL statement:

```
CREATE TABLE Products (
  ProductID bigint NOT NULL PRIMARY KEY,
  ProductName nvarchar(100) NOT NULL,
  ProductPrice decimal(18, 2) NOT NULL,
  ProductsInStock int NOT NULL,
  ProductsOnOrder int NOT NULL
)
```

The table includes the data shown below:

ProductID	ProductName	ProductPrice	ProductsInStock	ProductsOnOrder
1	ProductA	10.00	10	15
2	ProductB	30.00	20	0
3	ProductC	15.00	5	20

You are developing a report that displays the following values and column headers in the order listed below:
• average price of a product named Average
• the smallest number of products in stocK-named LowestNumber
• the highest product price named HighestPrice
You need to write a query to return the results for the report The query must meet the following requirements: You need to write a query to return the results for the report. The query must meet the following requirements:
• Use built-in, aggregate anfa*mathematical functions.
• Use two-part names and tables.
• Use the table alias to qualify column names.
• Define the alias for all fields by using the AS keyword.

• Use the first letter of the table name as the table alias.
• Do not use the Row_number function.
• Do not surround object names with square brackets.
• Do not use variables.

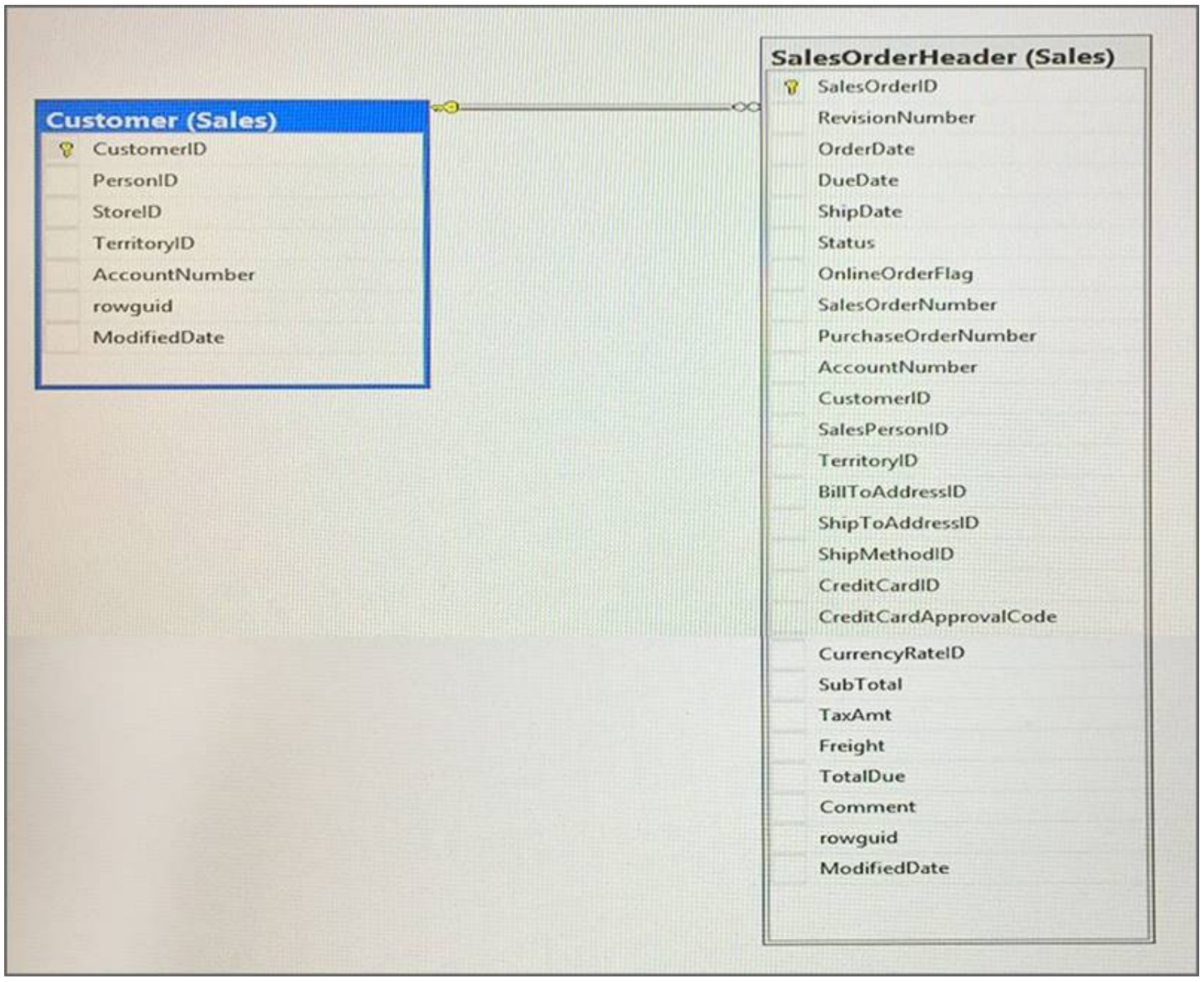
Part of the correct Transact-SQL has been provided in the answer area below. Enter the code in the answer area that resolves the problem and meets the stated goals or requirements. You can add code within the code that has been provided as well as below it.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
TRY_Convert

NEW QUESTION 8
You have a database that includes the tables shown in the exhibit. (Click the exhibit button.)



You need to create a list of all customers and the date that the customer placed their last order. For customers who have not placed orders, you must substitute a zero for the order ID and 01/01/1990 for the date.
Which Transact-SQL statement should you run?

A

```
SELECT C.CustomerID, COALESCE(MAX(OrderDate), '19000101')
FROM Sales.Customer C LEFT OUTER JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID
ORDER BY C.CustomerID
```

B

```
SELECT C.CustomerID, MAX(OrderDate)
FROM Sales.Customer C LEFT OUTER JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID
ORDER BY C.CustomerID
```

C

```
SELECT C.CustomerID, MAX(OrderDate)
FROM Sales.Customer C CROSS JOIN Sales.SalesOrderHeader SOH
GROUP BY C.CustomerID
ORDER BY C.CustomerID
```

D

```
SELECT C.CustomerID, SOH.SalesOrderID, MAX(OrderDate)
FROM Sales.Customer C LEFT OUTER JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID, SOH.SalesOrderID
ORDER BY C.CustomerID
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

Explanation:
COALESCE evaluates the arguments in order and returns the current value of the first expression that initially does not evaluate to NULL.
References: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/coalesce-transact-sql

NEW QUESTION 9
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.
You create a table named Customer by running the following Transact-SQL statement:

```
CREATE TABLE Customer (
     CustomerID int IDENTITY(1,1) PRIMARY KEY,
     FirstName varchar(50) NULL,
     LastName varchar(50) NOT NULL,
     DateOfBirth date NOT NULL,
     CreditLimit money CHECK (CreditLimit < 10000),
     TownID int NULL REFERENCES dbo.Town(TownID),
     CreatedDate datetime DEFAULT(Getdate())
)
```

You must insert the following data into the Customer table:

Record	First name	Last name	Date of Birth	Credit limit	Town ID	Created date
Record 1	Yvonne	McKay	1984-05-25	9,000	no town details	current date and time
Record 2	Jossef	Goldberg	1995-06-03	5,500	no town details	current date and time

You need to ensure that both records are inserted or neither record is inserted. Solution: You run the following Transact-SQL statement:

```
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000)
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit)
VALUES ('Jossef', 'Goldberg', '1995-06-03', 5500)
```

Does the solution meet the goal?

A. Yes
B. No

Answer: B

NEW QUESTION 10

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.
You create a table named Customer by running the following Transact-SQL statement:

```
CREATE TABLE Customer (
      CustomerID int IDENTITY(1,1) PRIMARY KEY,
      FirstName varchar(50) NULL,
      LastName varchar(50) NOT NULL,
      DateOfBirth date NOT NULL,
      CreditLimit money CHECK (CreditLimit < 10000),
      TownID int NULL REFERENCES Town(TownID),
      CreatedDate datetime DEFAULT(GETDATE())
)
```

You create a cursor by running the following Transact-SQL statement:

```
DECLARE cur CURSOR
FOR
SELECT LastName, CreditLimit
FROM Customer

DECLARE @LastName varchar(50), @CreditLimit money
OPEN cur
FETCH NEXT FROM cur INTO @LastName, @CreditLimit
WHILE (@@FETCH_STATUS = 0)
BEGIN
  FETCH NEXT FROM cur INTO @LastName, @CreditLimit
END
CLOSE cur
DEALLOCATE cur
```

If the credit limit is zero, you must delete the customer record while fetching data. You need to add the DELETE statement.
Solution: You add the following Transact-SQL statement:

```
IF @CreditLimit = 0
   DELETE Customer
   WHERE CustomerID IN (SELECT CustomerID)
   FROM Customer WHERE LastName = @LastName)
```

Does the solution meet the goal?

A. YES
B. NO

Answer: B

Explanation:
References: https://docs.microsoft.com/en-us/sql/t-sql/statements/delete-transact-sql?view=sql-server-2017

NEW QUESTION 10
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply to that question.
You have a database for a banking system. The database has two tables named tblDepositAcct and tblLoanAcct that store deposit and loan accounts, respectively. Both tables contain the following columns:

Column name	Data type	Primary key column	Description
CustNo	int	No	This column uniquely identifies a customer in the bank. A customer may have both deposit and loan accounts.
AcctNo	int	Yes	This column uniquely identifies a customer in the bank.
ProdCode	varchar(3)	No	This column identifies the product type of an account. A customer may have multiple accounts for the same product type.

You need to determine the total number of customers who have only loan accounts. Which Transact-SQL statement should you run?

A. SELECT COUNT(*)FROM (SELECT AcctNoFROM tblDepositAcctINTERSECTSELECTAcctNoFROM tblLoanAcct) R
B. SELECT COUNT(*)FROM (SELECT CustNoFROM tblDepositAcctUNIONSELECT CustNoFROMtblLoanAcct) R
C. SELECT COUNT(*)FROM (SELECT CustNoFROM tblDepositAcctUNION ALLSELECTCustNoFROM tblLoanAcct) R
D. SELECT COUNT (DISTINCT D.CustNo)FROM tblDepositAcct D, tblLoanAcct LWHERE D.CustNo= L.CustNo
E. SELECT COUNT(DISTINCT L.CustNo)FROM tblDepositAcct DRIGHT JOIN tblLoanAcct L ON D.CustNo = L.CustNoWHERE D.CustNo IS NULL
F. SELECT COUNT(*)FROM (SELECT CustNoFROM tblDepositAcctEXCEPTSELECT CustNoFROMtblLoanAcct) R
G. SELECT COUNT (DISTINCT COALESCE(D.CustNo, L.CustNo))FROM tblDepositAcct DFULLJOIN tblLoanAcct L ON D.CustNo = L.CustNoWHERE D.CustNo IS NULL OR L.CustNo IS NULL
H. SELECT COUNT(*)FROM tblDepositAcct DFULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo

Answer: E

Explanation:
The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.
References: https://www.w3schools.com/sql/sql_join_right.asp

NEW QUESTION 15
Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question presents a different goal and answer choices, but the text of the scenario is exactly the same in each question in this series.
You are developing a database to track customer orders. The database contains the following tables: Sales.Customers, Sales.Orders, and Sales.OrderLines.
The following table describes the columns in Sales.Customers.

Column name	Data type	Constraints
CustomerID	int	primary key
CustomerName	nvarchar(100)	does not allow null values
PhoneNumber	nvarchar(20)	does not allow null values
AccountOpenedDate	date	does not allow null values
StandardDiscountPercentage	decimal(18,3)	does not allow null values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow null values
DeliveryLocation	geography	does not allow null values
PhoneNumber	nvarchar(20)	does not allow null values

The following table describes the columns in Sales.Orders.

Column name	Data type	Constraints
OrderID	int	primary key
CustomerID	int	foreign key to the Sales.Customers table
OrderDate	date	does not allow null values

The following table describes the columns in Sales.OrderLines.

Column name	Data type	Constraints
OrderLineID	int	primary key
OrderID	int	foreign key to the Sales.Orders table
Quantity	int	does not allow null values
UnitPrice	decimal(18,2)	null values are permitted
TaxRate	decimal(18,3)	does not allow null values

You need to create a database object that calculates the total price of an order including the sales tax. The database object must meet the following requirements:
- Reduce the compilation cost of Transact-SQL code by caching the plans and reusing them for repeated execution.
- Return a value.
- Be callable from a SELECT statement.

How should you complete the Transact-SQL statements? To answer, select the appropriate Transact-SQL segments in the answer area.

Answer Area

```
CREATE [ PROCEDURE | VIEW | FUNCTION ]  Sales.CalculateOrderPrice
(
    @orderID int
)
[ WITH EXECUTE AS OWNER | RETURNS decimal(18,2) | RETURNS TABLE ]
AS
[ BEGIN TRAN | BEGIN | RETURN ]
    DECLARE @OrderPrice decimal(18,2)
    DECLARE @CalculatedTaxRate decimal(18,2)
    SET @OrderPrice = (SELECT SUM(Quantity * UnitPrice) FROM Sales.OrderLines WHERE OrderID = @OrderID
    SET @CalculatedTaxRate = (SELECT 1 + (MAX(TaxRate) / 100) FROM Sales.OrderLines WHERE OrderID = @OrderID)

RETURN ( [ @OrderPrice * @CalculatedTaxRate | SELECT (#OrderPrice * #CalculatedTaxRate) AS CalculatedOrderPrice | CalculateOrderPrice ] )

[ RETURN | COMMIT | END ]
```

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Box 1: FUNCTION
To be able to return a value we should use a scalar function.
CREATE FUNCTION creates a user-defined function in SQL Server and Azure SQL Database. The return value can either be a scalar (single) value or a table.
Box 2: RETURNS decimal(18,2)
Use the same data format as used in the UnitPrice column. Box 3: BEGIN
Transact-SQL Scalar Function Syntax include the BEGIN ..END construct.
CREATE [OR ALTER] FUNCTION [schema_name.] function_name
([{ @parameter_name [AS][type_schema_name.] parameter_data_type [= default] [READONLY] }
[,...n]
]
)
RETURNS return_data_type
[WITH <function_option> [,...n]] [AS]
BEGIN
function_body
RETURN scalar_expression END
[;]
Box 4: @OrderPrice * @CalculatedTaxRate Calculate the price including tax.
Box 5: END
Transact-SQL Scalar Function Syntax include the BEGIN ..END construct. References: https://msdn.microsoft.com/en-us/library/ms186755.aspx

NEW QUESTION 19
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You have a database that stores sales and order information.
Users must be able to extract information from the tables on an ad hoc basis. They must also be able to reference the extracted information as a single table.
You need to implement a solution that allows users to retrieve the data required, based on variables defined at the time of the query.
What should you implement?

A. the COALESCE function
B. a view
C. a table-valued function
D. the TRY_PARSE function
E. a stored procedure
F. the ISNULL function
G. a scalar function
H. the TRY_CONVERT function

Answer: C

Explanation:
User-defined functions that return a table data type can be powerful alternatives to views. These functions are referred to as table-valued functions. A table-valued user-defined function can be used where table or view expressions are allowed in Transact-SQL queries. While views are limited to a single SELECT statement, user-defined functions can contain additional statements that allow more powerful logic than is possible in views.
A table-valued user-defined function can also replace stored procedures that return a single result set. References: https://technet.microsoft.com/en-us/library/ms191165(v=sql.105).aspx

NEW QUESTION 22
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.
You have a database that tracks orders and deliveries for customers in North America. The database contains the following tables:
Sales.Customers

Column	Data type	Notes
CustomerID	int	primary key
CustomerCategoryID	int	foreign key to the Sales.CustomerCategories table
PostalCityID	int	foreign key to the Application.Cities table
DeliveryCityID	int	foreign key to the Application.Cities table
AccountOpenedDate	datetime	does not allow new values
StandardDiscountPercentage	int	does not allow new values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow new values
DeliveryLocation	geography	does not allow new values
PhoneNumber	nvarchar(20)	does not allow new values

Application.Cities

Column	Data type	Notes
CityID	int	primary key
LatestRecordedPopulation	bigint	null values are permitted

Sales.CustomerCategories

Column	Data type	Notes
CustomerCategoryID	int	primary key
CustomerCategoryName	nvarchar(50)	does not allow null values

Your company is developing a new social application that connects customers to each other based on the distance between their delivery locations.
You need to write a query that returns the nearest customer. Solution: You run the following Transact-SQL statement:

```
SELECT TOP 1 B.CustomerID, B.DeliveryLocation ^ A.DeliveryLocation AS Dist
FROM Sales.Customers AS A
JOIN Sales.Customers AS B
ON A.DeliveryCityID = B.DeliveryCityID
WHERE A.CustomerID = @custID AND A.CustomerID <> B.CustomerID
```

The variable @custID is set to a valid customer. Does the solution meet the goal?

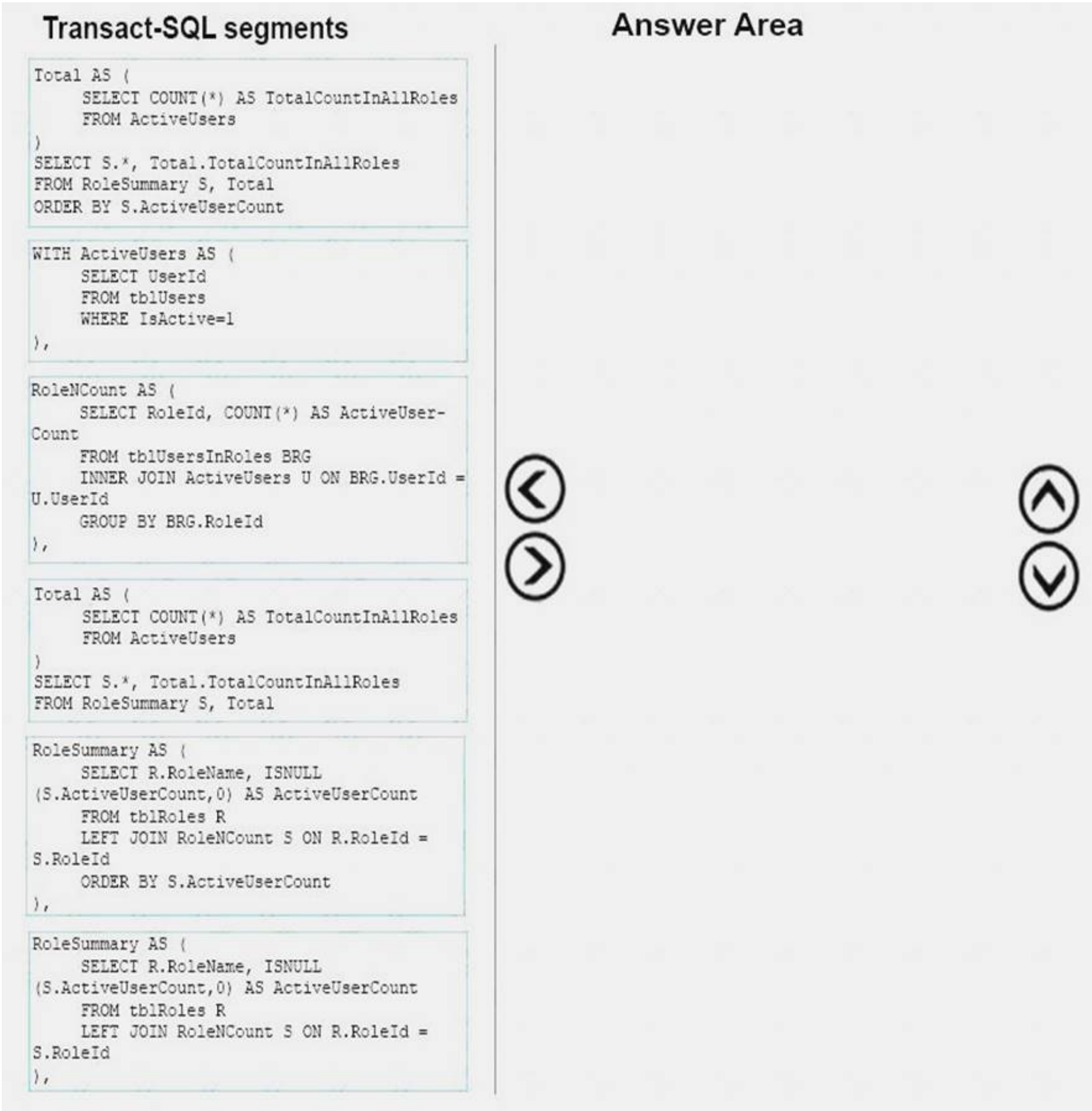
A. Yes
B. No

Answer: B

NEW QUESTION 24
You create three tables by running the following Transact-SQL statements:

```
CREATE TABLE tblRoles (
     RoleId int NOT NULL IDENTITY(1,1) PRIMARY KEY CLUSTERED,
     RoleName varchar(20) NOT NULL
)
CREATE TABLE tblUsers (
     UserId int NOT NULL IDENTITY(1,1) PRIMARY KEY CLUSTERED,
     UserName varchar(20) UNIQUE NOT NULL,
     IsActive bit NOT NULL DEFAULT(1)
)
CREATE TABLE tblUsersInRoles (
     UserId int NOT NULL FOREIGN KEY REFERENCES tblUsers(UserId),
     RoleId int NOT NULL FOREIGN KEY REFERENCES tblRoles(RolesId)
)
```

For reporting purposes, you need to find the active user count for each role, and the total active user count. The result must be ordered by active user count of each role. You must use common table expressions (CTEs).
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.



A. Mastered
B. Not Mastered

Answer: A

Explanation:

Transact-SQL segments

```
Total AS (
     SELECT COUNT(*) AS TotalCountInAllRoles
     FROM ActiveUsers
)
SELECT S.*, Total.TotalCountInAllRoles
FROM RoleSummary S, Total
ORDER BY S.ActiveUserCount
```

```
WITH ActiveUsers AS (
     SELECT UserId
     FROM tblUsers
     WHERE IsActive=1
),
```

```
RoleNCount AS (
     SELECT RoleId, COUNT(*) AS ActiveUser-
Count
     FROM tblUsersInRoles BRG
     INNER JOIN ActiveUsers U ON BRG.UserId =
U.UserId
     GROUP BY BRG.RoleId
),
```

```
Total AS (
     SELECT COUNT(*) AS TotalCountInAllRoles
     FROM ActiveUsers
)
SELECT S.*, Total.TotalCountInAllRoles
FROM RoleSummary S, Total
```

```
RoleSummary AS (
     SELECT R.RoleName, ISNULL
(S.ActiveUserCount,0) AS ActiveUserCount
     FROM tblRoles R
     LEFT JOIN RoleNCount S ON R.RoleId =
S.RoleId
     ORDER BY S.ActiveUserCount
),
```

```
RoleSummary AS (
     SELECT R.RoleName, ISNULL
(S.ActiveUserCount,0) AS ActiveUserCount
     FROM tblRoles R
     LEFT JOIN RoleNCount S ON R.RoleId =
S.RoleId
),
```

Answer Area

```
RoleNCount AS (
     SELECT RoleId, COUNT(*) AS ActiveUser-
Count
     FROM tblUsersInRoles BRG
     INNER JOIN ActiveUsers U ON BRG.UserId =
U.UserId
     GROUP BY BRG.RoleId
),
WITH ActiveUsers AS (
     SELECT UserId
     FROM tblUsers
     WHERE IsActive=1
),
```

```
RoleSummary AS (
     SELECT R.RoleName, ISNULL
(S.ActiveUserCount,0) AS ActiveUserCount
     FROM tblRoles R
     LEFT JOIN RoleNCount S ON R.RoleId =
S.RoleId
     ORDER BY S.ActiveUserCount
Total AS (
     SELECT COUNT(*) AS TotalCountInAllRoles
     FROM ActiveUsers
)
SELECT S.*, Total.TotalCountInAllRoles
FROM RoleSummary S, Total
ORDER BY S.ActiveUserCount
```

NEW QUESTION 29
You work for an organization that monitors seismic activity around volcanos. You have a table named GroundSensors. The table stored data collected from seismic sensors. It includes the columns describes in the following table:

Name	Data Type	Notes
SensorID	int	primary key
Location	geography	do not allow null values
Tremor	int	do not allow null values
NormalizedReading	float	allow null values

The database also contains a scalar value function named NearestMountain that returns the name of the mountain that is nearest to the sensor.
You need to create a query that shows the average of the normalized readings from the sensors for each mountain. The query must meet the following requirements:
* Include the average normalized readings and nearest mountain name.
* Exclude sensors for which no normalized reading exists.
* Exclude those sensors with value of zero for tremor. Construct the query using the following guidelines:
* Use one part names to reference tables, columns and functions.
* Do not use parentheses unless required.
* Do not use aliases for column names and table names.
* Do not surround object names with square brackets.

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT

```
DEFAULT                 ON                      TSEQUAL
DELETE                  OPEN                    UNION
DENY                    OPENDATASOURCE          UNIQUE
DESC                    OPENQUERY               UNPIVOT
DISK                    OPENROWSET              UPDATE
DISTINCT                OPENXML                 UPDATETEXT
DISTRIBUTED             OPTION                  USE
DOUBLE                  OR                      USER
DROP                    ORDER                   VALUES
DUMP                    OUTER                   VARYING
ELSE                    OVER                    VIEW
END                     PERCENT                 WAITFOR
ERRLVL                  PIVOT                   WHEN
ESCAPE                  PLAN                    WHERE
ESCEPT                  PRECISION               WHILE
EXEC                    PRIMARY                 WITH
EXECUTE                 PRINT                   WITHIN GROUP
EXISTS                                          WRITETEXT
```

Part of the correct Transact-SQL has been provided in the answer area below. Enter the code in the answer area that resolves the problem and meets the stated goals or requirements. You can add code within the code that has been provided as well as below it.

```
1 select
```

Use the Check Syntax button to verify your work. Any syntax or spelling errors will be reported by line and character position.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
GROUP BY is a SELECT statement clause that divides the query result into groups of rows, usually for the purpose of performing one or more aggregations on each group. The SELECT statement returns one row per group.
SELECT SensorID, NearestMountain(Location) FROM GroundSensors
WHERE TREMOR <> 0 AND NormalizedReading IS NOT NULL
GROUP BY SensorID, NearestMountain(Location)
References: https://msdn.microsoft.com/en-us/library/ms177673.aspx

NEW QUESTION 31
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.
You have a database that tracks orders and deliveries for customers in North America. The database contains the following tables:
Sales.Customers

Column	Data type	Notes
CustomerID	int	primary key
CustomerCategoryID	int	foreign key to the Sales.CustomerCategories table
PostalCityID	int	foreign key to the Application.Cities table
DeliveryCityID	int	foreign key to the Application.Cities table
AccountOpenedDate	datetime	does not allow new values
StandardDiscountPercentage	int	does not allow new values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow new values
DeliveryLocation	geography	does not allow new values
PhoneNumber	nvarchar(20)	does not allow new values

Application.Cities

Column	Data type	Notes
CityID	int	primary key
LatestRecordedPopulation	bigint	null values are permitted

Sales.CustomerCategories

Column	Data type	Notes
CustomerCategoryID	int	primary key
CustomerCategoryName	nvarchar(50)	does not allow null values

Your company is developing a new social application that connects customers to each other based on the distance between their delivery locations.
You need to write a query that returns the nearest customer. Solution: You run the following Transact-SQL statement:

```
WITH DIST_CTE (CustA, CustB, Dist)
AS (
      SELECT A.CustomerID AS CustA, B.CustomerID AS CustB,
B.DeliveryLocation.ShortestLineTo(A.DeliveryLocation).STLength() AS Dist
      FROM Sales.Customers AS A
      CROSS JOIN Sales.Customers AS B
      WHERE A.CustomerID <> B.CustomerID
)
SELECT TOP 1 CustB, Dist
FROM DIST_CTE
WHERE CustA = @custID
ORDER BY Dist
```

The variable @custID is set to a valid customer. Does the solution meet the goal?

A. Yes
B. No

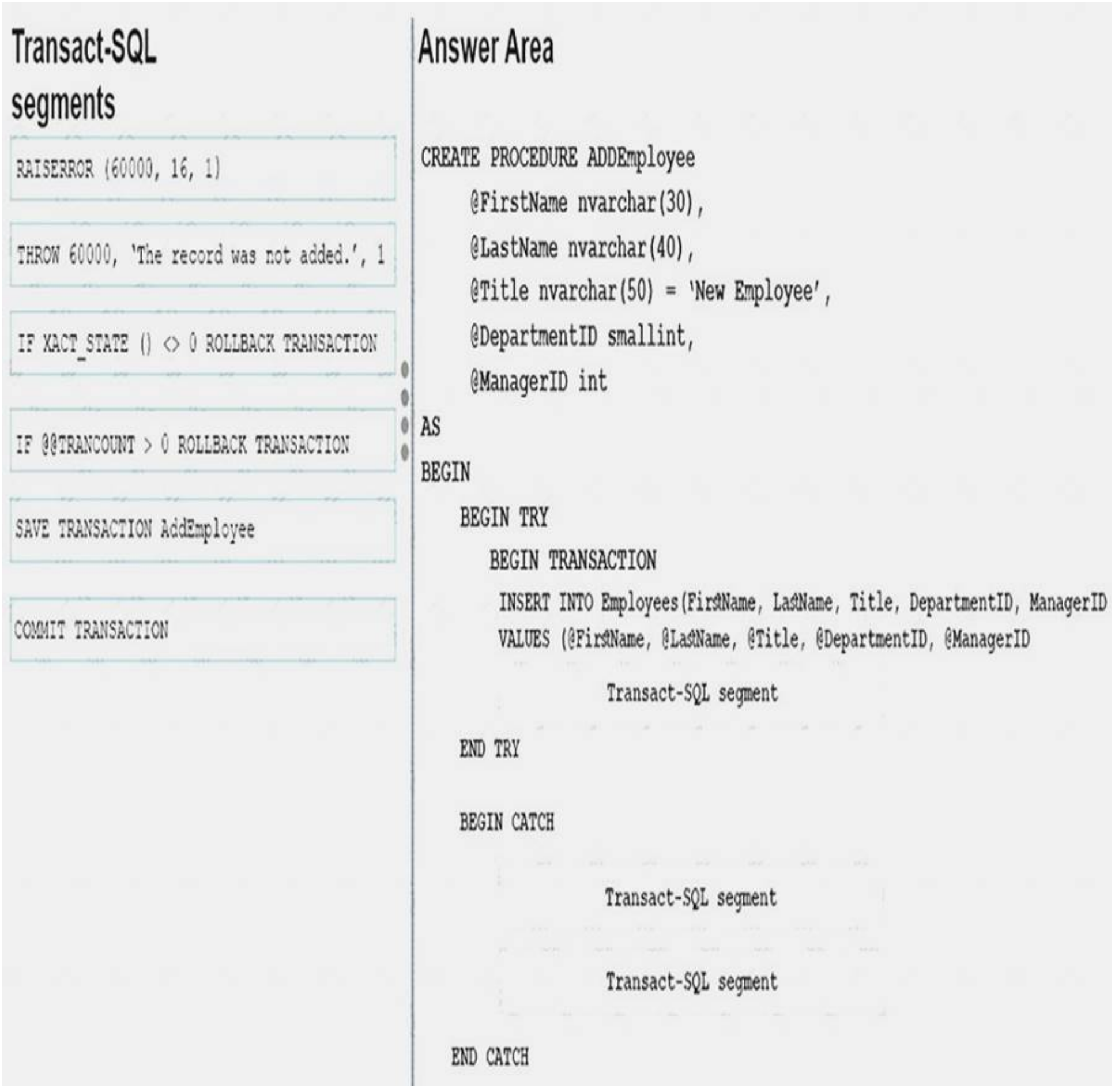
Answer: A

Explanation:
ShortestLineTo (geometry Data Type) Returns a LineString instance with two points that represent the shortest distance between the two geometry instances. The length of the LineString instance returned is the distance between the two geometry instances.
STLength (geometry Data Type) returns the total length of the elements in a geometry instance. References: https://docs.microsoft.com/en-us/sql/t-sql/spatial-geometry/shortestlineto-geometry-data-type

NEW QUESTION 36
You run the following Transact-SQL statement:

```
CREATE TABLE Employees (
      EmployeeID int IDENTITY(1, 1) PRIMARY KEY NOT NULL,
      FirstName nvarchar(30) NOT NULL,
      LastName nvarchar(40) NOT NULL,
      Title nvarchar(50) NOT NULL,
      DepartmentID smallint NOT NULL,
      ManagerID int NULL
)
```

You need to create a stored procedure that meets the following requirements:
Inserts data into the Employees table.
Processes all data changes as a single unit of work.
Sets the exception severity level to 16 and an error number of 60, 000 when any error occurs.
If a Transact-SQL statement raises a runtime error, terminates and reverts the entire unit of work, and indicates the line number in the statement where the error occurred.
Inserts the value New Employee for the Title column if no title is provided.
How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segment to the correct target. 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.



A. Mastered
B. Not Mastered

Answer: A

Explanation:

Transact-SQL segments

```
RAISERROR (60000, 16, 1)
```

```
THROW 60000, 'The record was not added.', 1
```

```
IF XACT_STATE () <> 0 ROLLBACK TRANSACTION
```

```
IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
```

```
SAVE TRANSACTION AddEmployee
```

```
COMMIT TRANSACTION
```

Answer Area

```
CREATE PROCEDURE ADDEmployee
     @FirstName nvarchar(30),
     @LastName nvarchar(40),
     @Title nvarchar(50) = 'New Employee',
     @DepartmentID smallint,
     @ManagerID int
AS
BEGIN
    BEGIN TRY
       BEGIN TRANSACTION
       INSERT INTO Employees(FirstName, LastName, Title, DepartmentID, ManagerID
       VALUES (@FirstName, @LastName, @Title, @DepartmentID, @ManagerID
       COMMIT TRANSACTION
    END TRY

    BEGIN CATCH
       IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
       RAISERROR (60000, 16, 1)
    END CATCH
```

NEW QUESTION 38
You have a database named MyDb. You run the following Transact-SQL statements:

```
CREATE TABLE tblRoles (
     RoleId int NOT NULL IDENTITY(1,1) PRIMARY KEY CLUSTERED,
     RoleName varchar(20) NOT NULL
)
CREATE TABLE tblUsers (
     UserId int NOT NULL IDENTITY(10000,1) PRIMARY KEY CLUSTERED,
     UserName varchar(20) UNIQUE NOT NULL,
     RoleId int NULL FOREIGN KEY REFERENCES tbRoles(RoleId),
     IsActive bit NOT NULL DEFAULT(1)
)
```

A value of 1 in the IsActive column indicates that a user is active.
You need to create a count for active users in each role. If a role has no active users. you must display a zero as the active users count.
Which Transact-SQL statement should you run?

A
```
SELECT R.RoleName, COUNT(U.UserId) AS ActiveUserCount FROM tblRoles R
LEFT JOIN (SELECT UserId, RoleId FROM tblUsers WHERE IsActive = 1) U ON U.RoleId = R.RoleId
GROUP BY R.RoleId, R.RoleName
```

B
```
SELECT R.RoleName, U.ActiveUserCount FROM tblRoles R
INNER JOIN (SELECT RoleId, COUNT(*) AS ActiveUserCount FROM tblUsers WHERE IsActive = 1
GROUP BY RoleId) U ON R.RoleId = U.RoleId
```

C
```
SELECT R.RoleName, COUNT(*) AS ActiveUserCount FROM tblRoles R
LEFT JOIN (SELECT UserId, RoleId FROM tblUsers WHERE IsActive = 1)U ON U.RoleId = R.RoleId
GROUP BY R.RoleId, R.RoleName
```

D
```
SELECT R.RoleName, U.ActiveUserCount FROM tblRoles R CROSS JOIN
(SELECT COUNT(*) AS ActiveUserCount FROM tblUsers WHERE IsActive = 1) U
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: C

NEW QUESTION 39
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.
You are building a stored procedure that will be used by hundreds of users concurrently.
You need to store rows that will be processed later by the stored procedure. The object that stores the rows must meet the following requirements:
Be indexable
Contain up-to-date statistics
Be able to scale between 10 and 100,000 rows
The solution must prevent users from accessing one another's data. Solution: You create a local temporary table in the stored procedure. Does this meet the goal?

A. Yes
B. No

Answer: B

NEW QUESTION 40
Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question presents a different goal and answer choices, but the text of the scenario is exactly the same in each question in this series.
You query a database that includes two tables: Project and Task. The Project table includes the following columns:

Column name	Data type	Notes
ProjectId	int	This is a unique identifier for a project.
ProjectName	varchar(100)	
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the project is not finished yet.
UserId	int	Identifies the owner of the project.

The Task table includes the following columns:

Column name	Data type	Notes
TaskId	int	This is a unique identifier for a task.
TaskName	varchar(100)	A nonclustered index exists for this column.
ParentTaskId	int	Each task may or may not have a parent task.
ProjectId	int	A null value indicates the task is not assigned to a specific project.
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the task is not completed yet.
UserId	int	Identifies the owner of the task.

You need to find all projects that have at least one task that took more than 50 hours to complete. You must also determine the average duration of the tasks that took more that took more than 50 hours to complete for each project.
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.

A. Mastered
B. Not Mastered

Answer: A

Explanation:

Transact-SQL segments

```
AVR(DATEDIFF(hh, T.StartTime, T.EndTime))

AVR(DATEDIFF(yy, T.StartTime, T.EndTime))

SUM(DATEDIFF(hh, T.StartTime, T.EndTime))/SU

DATEDIFF(hh, T.StartTime, T.EndTime)) > 50

DATEDADD(hh, 50, T.StartTime, ) > T.EndTime

DATEADD(yy, -50, T.EndTime) <= T.StartTime
```

●●●●

Answer area

```
SELECT P.ProjectId, P.ProjectName, T.Summary.AvgDurationHours FROM Project P
OUTER APPLY
(
   SELECT [ Transact-SQL segment ] AS AvgDurationHours FROM Task T

   WHERE T.ProjectId = P.ProjectId

   AND [ Transact-SQL segment ]

) TSummary

WHERE T.Summary.AvgDurationHours IS NOT NULL
```

NEW QUESTION 41
Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question presents a different goal and answer choices, but the text of the scenario is exactly the same in each question in this series.
You are developing a database to track customer orders. The database contains the following tables: Sales.Customers, Sales.Orders, and Sales.OrderLines. The following table describes the columns in Sales.Customers.

Column name	Data type	Constraints
CustomerID	int	primary key
CustomerName	nvarchar(100)	does not allow null values
PhoneNumber	nvarchar(20)	does not allow null values
AccountOpenedDate	date	does not allow null values
StandardDiscountPercentage	decimal(18,3)	does not allow null values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow null values
DeliveryLocation	geography	does not allow null values
PhoneNumber	nvarchar(20)	does not allow null values

The following table describes the columns in Sales.Orders.

Column name	Data type	Constraints
OrderID	int	primary key
CustomerID	int	foreign key to the Sales.Customers table
OrderDate	date	does not allow null values

The following table describes the columns in Sales.OrderLines.

Column name	Data type	Constraints
OrderLineID	int	primary key
OrderID	int	foreign key to the Sales.Orders table
Quantity	int	does not allow null values
UnitPrice	decimal(18,2)	null values are permitted
TaxRate	decimal(18,3)	does not allow null values

You need to create a stored procedure that inserts data into the Customers table. The stored procedure must meet the following requirements:
- Data changes occur as a single unit of work.
- Data modifications that are successful are committed and a value of 0 is returned.
- Data modifications that are unsuccessful are rolled back. The exception severity level is set to 16 and a value of -1 is returned.
- The stored procedure uses a built-it scalar function to evaluate the current condition of data modifications.
- The entire unit of work is terminated and rolled back if a run-time error occurs during execution of the stored procedure.

How should complete the stored procedure definition? To answer, drag the appropriate Transact-SQL segments to the correct targets. 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.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Box 1: XACT_ABORT
XACT_ABORT specifies whether SQL Server automatically rolls back the current transaction when a Transact-SQL statement raises a run-time error.
When SET XACT_ABORT is ON, if a Transact-SQL statement raises a run-time error, the entire transaction is terminated and rolled back.
Box 2: COMMIT
Commit the transaction. Box 3: XACT_STATE
Box 4: ROLLBACK
Rollback the transaction Box 5: THROW
THROW raises an exception and the severity is set to 16.
Requirement: Data modifications that are unsuccessful are rolled back. The exception severity level is set to 16 and a value of -1 is returned.
References:
https://msdn.microsoft.com/en-us/library/ms188792.aspx https://msdn.microsoft.com/en-us/library/ee677615.aspx

NEW QUESTION 44
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.
You have a database that tracks orders and deliveries for customers in North America. The database contains the following tables:
Sales.Customers

Column	Data type	Notes
CustomerID	int	primary key
CustomerCategoryID	int	foreign key to the Sales.CustomerCategories table
PostalCityID	int	foreign key to the Application.Cities table
DeliveryCityID	int	foreign key to the Application.Cities table
AccountOpenedDate	datetime	does not allow new values
StandardDiscountPercentage	int	does not allow new values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow new values
DeliveryLocation	geography	does not allow new values
PhoneNumber	nvarchar(20)	does not allow new values data is formatted as follows: 425-555-0187

Application.Cities

Column	Data type	Notes
CityID	int	primary key
LatestRecordedPopulation	bigint	null values are permitted

Sales.CustomerCategories

Column	Data type	Notes
CustomerCategoryID	int	primary key
CustomerCategoryName	nvarchar(50)	does not allow null values

The company's development team is designing a customer directory application. The application must list customers by the area code of their phone number. The area code is defined as the first three characters of the phone number.
The main page of the application will be based on an indexed view that contains the area and phone number for all customers.
You need to return the area code from the PhoneNumber field. Solution: You run the following Transact-SQL statement:

```
CREATE FUNCTION AreaCode (
      @phoneNumber nvarchar(20)
)
RETURNS nvarchar(10)
WITH SCHEMABINDING
AS
BEGIN
      DECLARE @areaCode nvarchar(max)
      SELECT @areaCode = value FROM STRING_SPLIT(@phoneNumber, '-')
      RETURN @areaCode
END
```

Does the solution meet the goal?

A. Yes
B. No

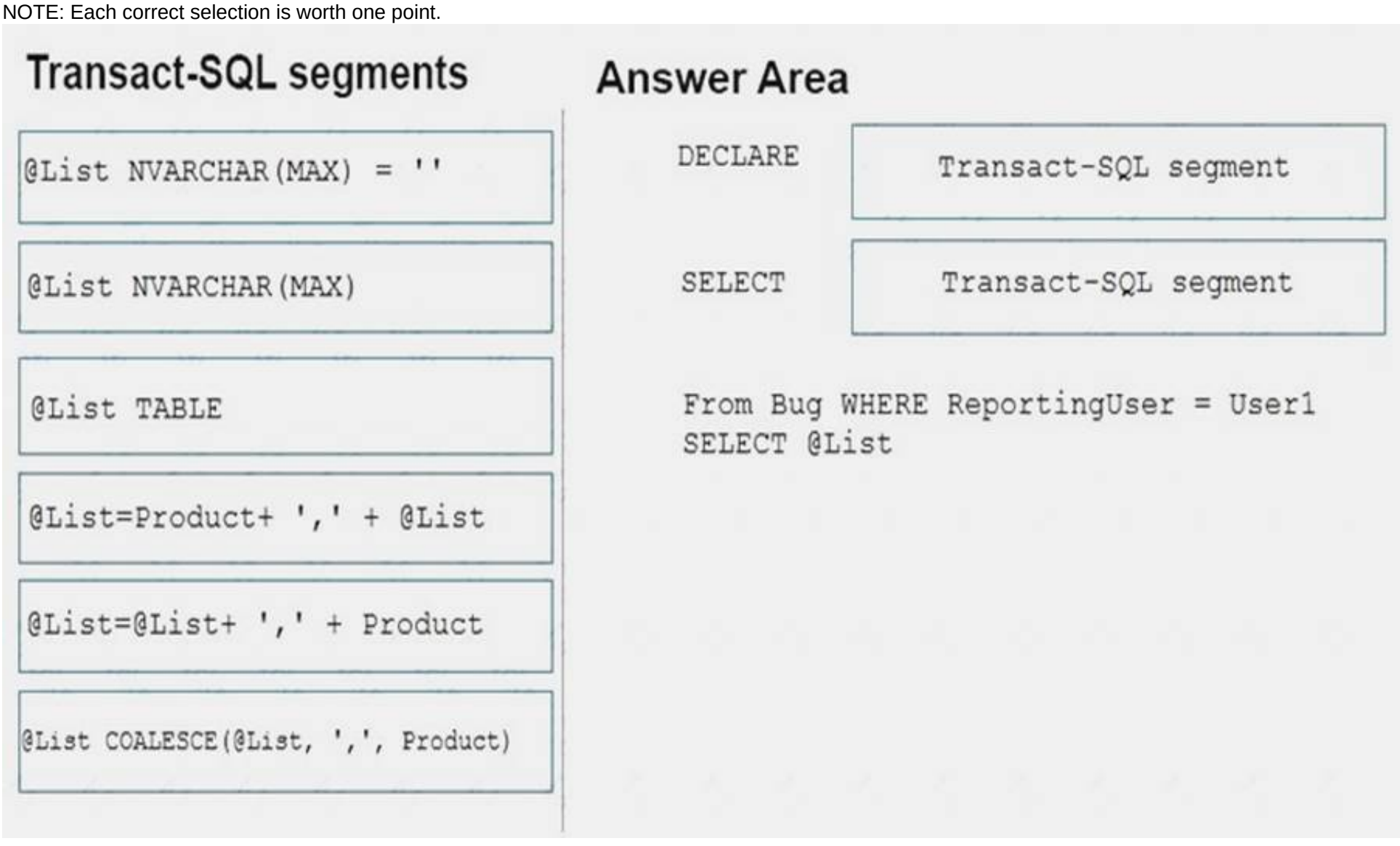
Answer: B

Explanation:
The variable max, in the line DECLARE @areaCode nvarchar(max), is not defined.

NEW QUESTION 45
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.
You run the following Transact-SQL statement:

```
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 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.



A. Mastered
B. Not Mastered

Answer: A

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

NEW QUESTION 49
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You have a table named Products that contains information about the products that your company sells. The table contains many columns that do not always contain values.
You need to implement an ANSI standard method to convert the NULL values in the query output to the phrase “Not Applicable”.
What should you implement?

A. the COALESCE function
B. a view
C. a table-valued function
D. the TRY_PARSE function
E. a stored procedure
F. the ISNULL function
G. a scalar function
H. the TRY_CONVERT function

Answer: F

Explanation:
The ISNULL function replaces NULL with the specified replacement value. References: https://msdn.microsoft.com/en-us/library/ms184325.aspx

NEW QUESTION 53
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.
You have a database named DB1 that contains two tables named Sales.Customers and Sales.Orders. Sales.Customers has a foreign key relationship to a column named CustomerID in SalesOrders.
You need to recommend a query that returns all the customers. The query must also return the number of orders that each customer placed in 2016.
Solution: You recommend the following query:

```
SELECT
      Cust.CustomerName,
      NumberOfOrders = COUNT(*)
FROM
      Sales.Customers Cust
LEFT JOIN
      Sales.Orders Ord
            ON Cust.CustomerID = Ord.OrderID
GROUP BY
      Cust.CustomerName;
```

Does this meet the goal?

A. Yes
B. No

Answer: B

NEW QUESTION 58
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.
You create a table named Customer by running the following Transact-SQL statement:

```
CREATE TABLE Customer (
      CustomerID int IDENTITY(1,1) PRIMARY KEY,
      FirstName varchar(50) NULL,
      LastName varchar(50) NOT NULL,
      DateOfBirth date NOT NULL,
      CreditLimit money CHECK (CreditLimit < 10000),
      TownID int NULL REFERENCES dbo.Town(TownID),
      CreatedDate datetime DEFAULT(Getdate())
)
```

You must insert the following data into the Customer table:

Record	First name	Last name	Date of Birth	Credit limit	Town ID	Created date
Record 1	Yvonne	McKay	1984-05-25	9,000	no town details	current date and time
Record 2	Jossef	Goldberg	1995-06-03	5,500	no town details	current date and time

You need to ensure that both records are inserted or neither record is inserted. Solution: You run the following Transact-SQL statement:

```
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000, GETDATE())
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Jossef', 'Goldberg', '1995-06-03', 5500, GETDATE())
GO
```

Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:

As there are two separate INSERT INTO statements we cannot ensure that both or neither records is inserted.

NEW QUESTION 62
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.
You are creating indexes in a data warehouse.
You have a dimension table named Table1 that has 10,000 rows. The rows are used to generate several reports. The reports join a column that is the primary key.
The execution plan contains bookmark lookups for Table1. You discover that the reports run slower than expected.
You need to reduce the amount of time it takes to run the reports.
Solution: You create a nonclustered index on the primary key column that does NOT include columns. Does this meet the goal?

A. YES
B. NO

Answer: A

Explanation:
References:
https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?

NEW QUESTION 63
You create a table named Sales.Orders by running the following Transact-SQL statement:

```
CREATE TABLE Sales.Orders (
      OrderID int NOT NULL,
      OrderDate date NULL,
      ShippedDate date NULL,
      Status varchar(10),
      CONSTRAINT PK_ORDERS PRIMARY KEY CLUSTERED OrderID
)
```

You need to write a query that removes orders from the table that have a Status of Canceled. Construct the query using the following guidelines:

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT

```
DEFAULT
DELETE
DENY
DESC
DISK
DISTINCT
DISTRIBUTED
DOUBLE
DROP
DUMP
ELSE
END
ERRLVL
ESCAPE
ESCEPT
EXEC
EXECUTE
EXISTS
ON
OPEN
OPENDATASOURCE
OPENQUERY
OPENROWSET
OPENXML
OPTION
OR
ORDER
OUTER
OVER
PERCENT
PIVOT
PLAN
PRECISION
PRIMARY
PRINT
TSEQUAL
UNION
UNIQUE
UNPIVOT
UPDATE
UPDATETEXT
USE
USER
VALUES
VARYING
VIEW
WAITFOR
WHEN
WHERE
WHILE
WITH
WITHIN GROUP
WRITETEXT
```

Part of the correct Transact-SQL has been provided in the answer area below. Enter the code in the answer area that resolves the problem and meets the stated goals or requirements. You can add code within the code that has been provided as well as below it.

```
1 DELETE from sales.orders where status='calceled'
```

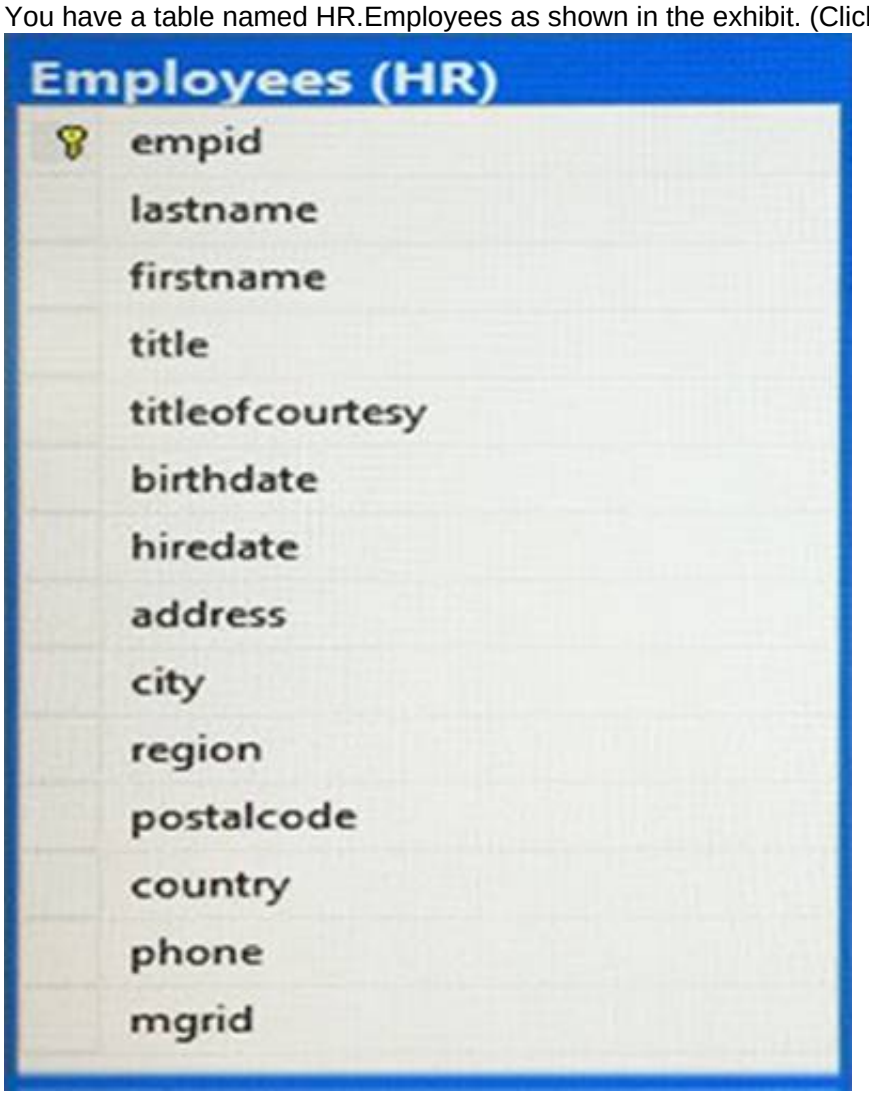
Use the Check Syntax button to verify your work. Any syntax or spelling errors will be reported by line and character position.

A. Mastered
B. Not Mastered

Answer: A

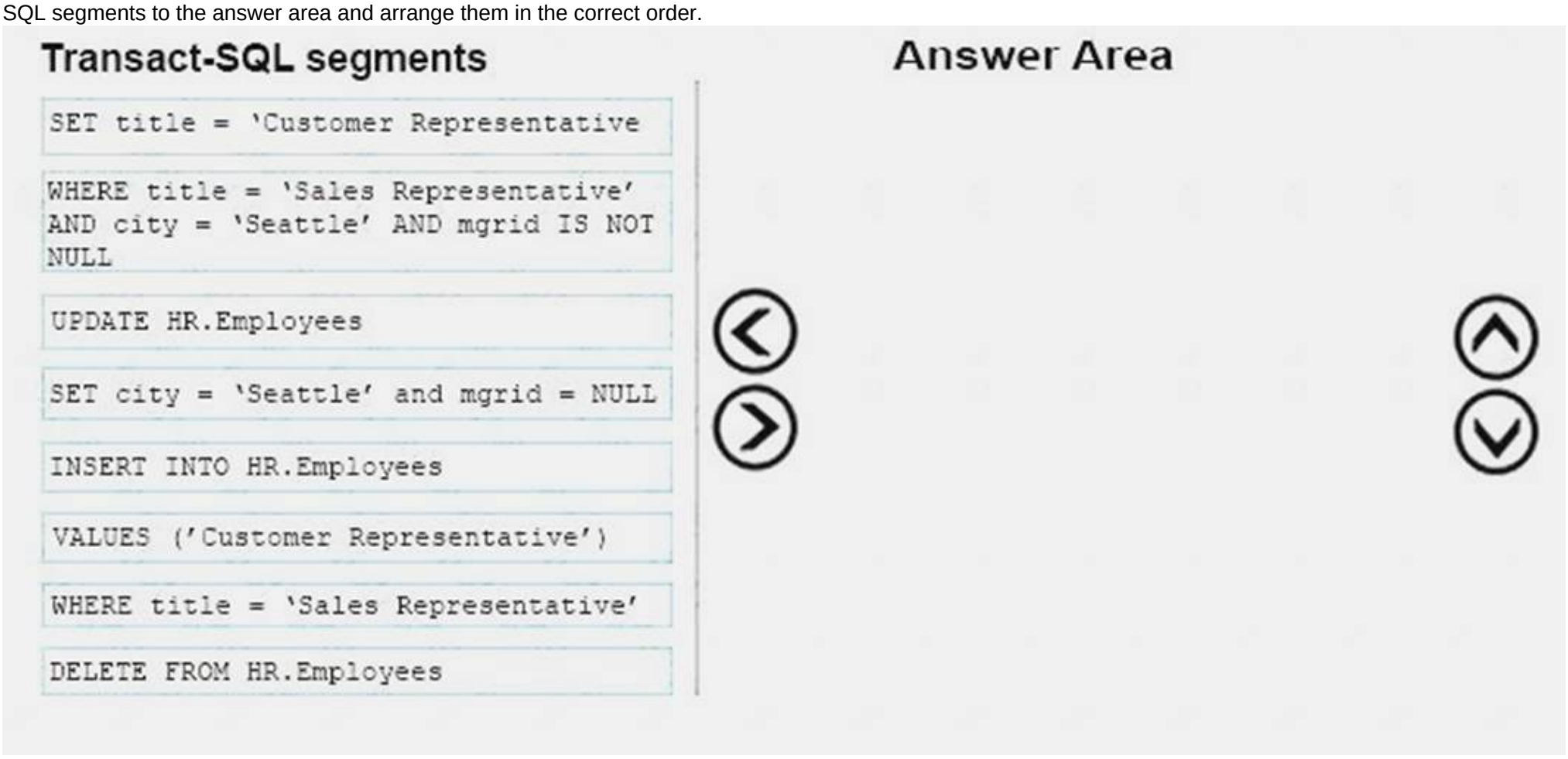
Explanation:
1. DELETE from sales.orders where status='Canceled' Note: On line 1 change calceled to Canceled
Example: Using the WHERE clause to delete a set of rows
The following example deletes all rows from the ProductCostHistory table in the AdventureWorks2012 database in which the value in the StandardCost column is more than 1000.00.
DELETE FROM Production.ProductCostHistory WHERE StandardCost > 1000.00;
References: https://docs.microsoft.com/en-us/sql/t-sql/statements/delete-transact-sql

NEW QUESTION 67
You have a table named HR.Employees as shown in the exhibit. (Click the exhibit button.)



You need to write a query that will change the value of the job title column to Customer Representative for any employee who lives in Seattle and has a job title of

Sales Representative. If the employee does not have a manager defined, you must not change the title.
Which three 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.



A. Mastered
B. Not Mastered

Answer: A

Explanation:
References: https://msdn.microsoft.com/en-us/library/ms177523.aspx

NEW QUESTION 71
You have the following Transact-SQL statement: DELETE FROM Person
WHERE PersonID = 5
You need to implement error handling.
How should you complete Transact-SQL statement? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

```
BEGIN TRANSACTION
```
[▼]
- END TRY
- COMMIT TRANSACTION
- END CATCH
- BEGIN TRY
- ROLLBACK TRANSACTION
- BEGIN CATCH

```
    DELETE FROM Person
    WHERE PersonID = 5
```
[▼]
- END TRY
- COMMIT TRANSACTION
- END CATCH
- BEGIN TRY
- ROLLBACK TRANSACTION
- BEGIN CATCH

[▼]
- END TRY
- COMMIT TRANSACTION
- END CATCH
- BEGIN TRY
- ROLLBACK TRANSACTION
- BEGIN CATCH

```
IF @@TRANCOUNT > 0
```
[▼]
- END TRY
- COMMIT TRANSACTION
- END CATCH
- BEGIN TRY
- ROLLBACK TRANSACTION
- BEGIN CATCH

[▼]
- END TRY
- COMMIT TRANSACTION
- END CATCH
- BEGIN TRY
- ROLLBACK TRANSACTION
- BEGIN CATCH

```
IF @@TRANCOUNT > 0
 COMMIT TRANSACTION
```

A. Mastered
B. Not Mastered

Answer: A

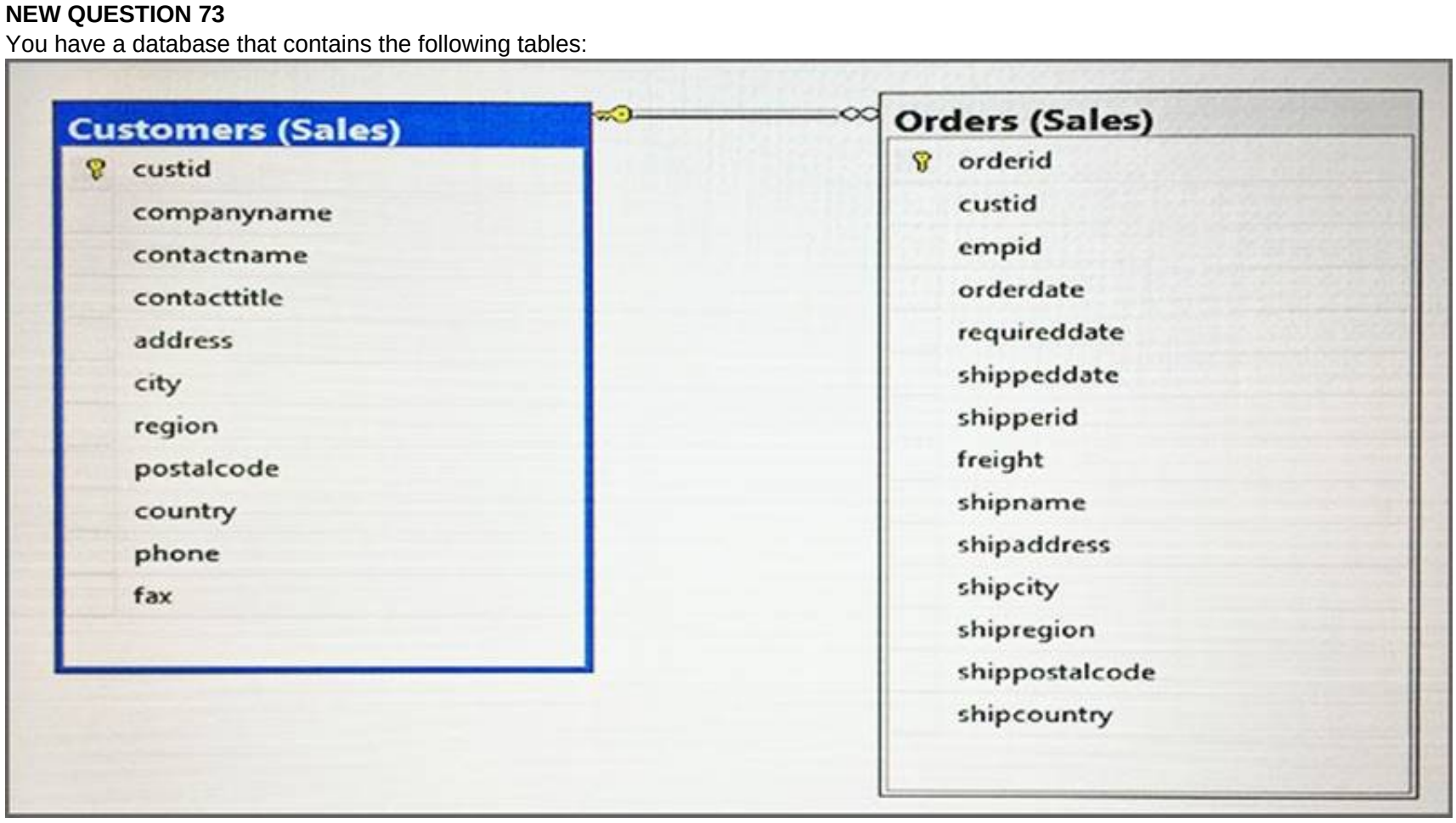
Explanation:

```
BEGIN TRANSACTION
[ END TRY | COMMIT TRANSACTION | END CATCH | BEGIN TRY | ROLLBACK TRANSACTION | BEGIN CATCH ]
    DELETE FROM Person
    WHERE PersonID = 5
[ END TRY | COMMIT TRANSACTION | END CATCH | BEGIN TRY | ROLLBACK TRANSACTION | BEGIN CATCH ]
[ END TRY | COMMIT TRANSACTION | END CATCH | BEGIN TRY | ROLLBACK TRANSACTION | BEGIN CATCH ]

IF @@TRANCOUNT > 0
    [ END TRY | COMMIT TRANSACTION | END CATCH | BEGIN TRY | ROLLBACK TRANSACTION | BEGIN CATCH ]
    [ END TRY | COMMIT TRANSACTION | END CATCH | BEGIN TRY | ROLLBACK TRANSACTION | BEGIN CATCH ]
IF @@TRANCOUNT > 0
 COMMIT TRANSACTION
```

NEW QUESTION 73

You have a database that contains the following tables:



You need to write a query that returns a list of all customers who have not placed orders. Which Transact-SQL statement should you run?

A. SELECT c.custidFROM Sales.Customers c INNER JOIN Sales.Order oON c.custid = o.custid
B. SELECT custid FROM Sales.CustomersINTERSECTSELECT custid FROM Sales.Orders
C. SELECT c.custidFROM Sales.Customers c LEFT OUTER Sales.Order oON c.custid = o.custid
D. SELECT c.custidFROM Sales.Customers c LEFT OUTER JOIN Sales.Order o ON c.custid = o.custidWHERE orderid IS NULL

Answer: D

Explanation:
Inner joins return rows only when there is at least one row from both tables that matches the join condition. Inner joins eliminate the rows that do not match with a row from the other table. Outer joins, however, return all rows from at least one of the tables or views mentioned in the FROM clause, as long as those rows meet any WHERE or HAVING search conditions. All rows are retrieved from the left table referenced with a left outer join, and all rows from the right table referenced in a right outer join. All rows from both tables are returned in a full outer join.
References: https://technet.microsoft.com/en-us/library/ms187518(v=sql.105).aspx

NEW QUESTION 74
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.
You have a table named Products that stores information about products your company sells. The table has a column named ListPrice that stores retail pricing information for products.
Some products are used only internally by the company. Records for these products are maintained in the Products table for inventory purposes. The price for each of these products is $0.00. Customers are not permitted to order these products.
You need to increase the list price for products that cost less than $100 by 10 percent. You must only increase pricing for products that customers are permitted to order.
Solution: You run the following Transact-SQL statement:

```
UPDATE Production. Products
SET ListPrice = (ListPrice* .1)
WHERE ListPrice <100
```

Does the solution meet the goal?

A. Yes
B. No

Answer: B

Explanation:
Mathematical equation will only return 10 % of the value.

NEW QUESTION 78
You are building a stored procedure named sp1 that calls a stored procedure named SP2.
SP2 calls another stored procedure named SP3 that returns a Recordset. The Recordset is stored in a temporary table.
You need to ensure that SP2 returns a text value to sp1. What should you do?

A. Return the text value by using the Retumvaiue when sp2 is called.
B. Create a temporary table in sp2, and then insert the text value into the table.
C. Create a table variable in SP2, and then insert the text value into the table.
D. Add the text value to an output parameter of SP2.

Answer: B

NEW QUESTION 81
You create a table named Sales.Orders by running the following Transact-SQL statement:

```
CREATE TABLE Sales.Orders (
      OrderID int NOT NULL,
      OrderDate date NULL,
      ShippedDate date NULL,
      Status varchar(10),
      CONSTRAINT PK_ORDERS PRIMARY KEY CLUSTERED
)
```

You need to write a query that meets the following requirements:
- removes orders from the table that were placed before January 1, 2012
- uses the date format of YYYYMMDD
- ensures that the order has been shipped before deleting the record Construct the query using the following guidelines:
- use one-part column names and two-part table names
- do not use functions

- do not surround object names with square brackets
- do not use variables
- do not use aliases for column names and table names

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT

```
DEFAULT
DELETE
DENY
DESC
DISK
DISTINCT
DISTRIBUTED
DOUBLE
DROP
DUMP
ELSE
END
ERRLVL
ESCAPE
ESCEPT
EXEC
EXECUTE
EXISTS
ON
OPEN
OPENDATASOURCE
OPENQUERY
OPENROWSET
OPENXML
OPTION
OR
ORDER
OUTER
OVER
PERCENT
PIVOT
PLAN
PRECISION
PRIMARY
PRINT
TSEQUAL
UNION
UNIQUE
UNPIVOT
UPDATE
UPDATETEXT
USE
USER
VALUES
VARYING
VIEW
WAITFOR
WHEN
WHERE
WHILE
WITH
WITHIN GROUP
WRITETEXT
```

Part of the correct Transact-SQL has been provided in the answer area below. Enter the code in the answer area that resolves the problem and meets the stated goals or requirements. You can add code within the code that has been provided as well as below it.

```
1  DELETE
```

Use the Check Syntax button to verify your work. Any syntax or spelling errors will be reported by line and character position.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
DELETE Sales.Orders FROM Sales.Orders
WHERE OrderDate < '20120101' AND ShippedDate IS NOT NULL

NEW QUESTION 84
.....

Thank You for Trying Our Product

* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

* One year free update

You can enjoy free update one year. 24x7 online support.

* Trusted by Millions

We currently serve more than 30,000,000 customers.

* Shop Securely

All transactions are protected by VeriSign!

100% Pass Your 70-761 Exam with Our Prep Materials Via below:

https://www.certleader.com/70-761-dumps.html