

Microsoft

Exam Questions 70-761

Querying Data with Transact-SQL (beta)





NEW QUESTION 1

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.

Start of repeated scenario

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

SalesSummary			
Column Name	Data Type	Allow Nulls	
 SalesSummaryKey	int	<input type="checkbox"/>	
SalesYear	smallint	<input type="checkbox"/>	
SalesQuarter	smallint	<input type="checkbox"/>	
SalesMonth	smallint	<input type="checkbox"/>	
SalesDate	date	<input type="checkbox"/>	
ProductCode	char(12)	<input type="checkbox"/>	
CustomerCode	char(6)	<input type="checkbox"/>	
EmployeeCode	char(6)	<input type="checkbox"/>	
RegionCode	char(2)	<input checked="" type="checkbox"/>	
SalesAmount	money	<input type="checkbox"/>	
		<input type="checkbox"/>	

Employee			
Column Name	Data Type	Allow Nulls	
 EmployeeID	smallint	<input type="checkbox"/>	
EmployeeCode	char(6)	<input type="checkbox"/>	
FirstName	varchar(30)	<input checked="" type="checkbox"/>	
MiddleName	varchar(30)	<input checked="" type="checkbox"/>	
LastName	varchar(40)	<input type="checkbox"/>	
Title	varchar(50)	<input type="checkbox"/>	
ManagerID	smallint	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

You review the Employee table and make the following observations:

- Every record has a value in the ManagerID except for the Chief Executive Officer (CEO).
- The FirstName and MiddleName columns contain null values for some records.
- The valid values for the Title column are Sales Representative manager, and CEO. You review the SalesSummary table and make the following observations:
- The ProductCode column contains two parts: The first five digits represent a product code, and the last seven digits represent the unit price. The unit price uses the following pattern: #####.##.
- You observe that for many records, the unit price portion of the ProductCode column contains values.
- The RegionCode column contains NULL for some records.
- Sales data is only recorded for sales representatives.

You are developing a series of reports and procedures to support the business. Details for each report or procedure follow.

Sales Summary report: This report aggregates data by year and quarter. The report must resemble the following table.

SalesYear	SalesQuarter	YearSalesAmount	QuarterSalesAmount
2015	1	2000.00	1000.00
2015	2	2000.00	500.00
2015	3	2000.00	250.00
2015	4	2000.00	250.00
2016	1	3500.00	500.00
2016	2	3500.00	1000.00

Sales Manager report: This report lists each sales manager and the total sales amount for all employees that report to the sales manager.

Sales by Region report: This report lists the total sales amount by employee and by region. The report must include the following columns: EmployeeCode, MiddleName, LastName, RegionCode, and SalesAmount. If MiddleName is NULL, FirstName must be displayed. If both FirstName and MiddleName have null values, the word Unknown must be displayed/ If RegionCode is NULL, the word Unknown must be displayed.

Report1: This report joins data from SalesSummary with the Employee table and other tables. You plan to create an object to support Report1. The object has the following requirements:

- be joinable with the SELECT statement that supplies data for the report
- can be used multiple times with the SELECT statement for the report
- be usable only with the SELECT statement for the report
- not be saved as a permanent object

Report2: This report joins data from SalesSummary with the Employee table and other tables. You plan to create an object to support Report1. The object has the following requirements:

Sales Hierarchy report. This report aggregates rows, creates subtotal rows, and super-aggregates rows over the SalesAmount column in a single result-set. The report uses SaleYear, SaleQuarter, and SaleMonth as a hierarchy. The result set must not contain a grand total or cross-tabulation aggregate rows.

Current Price Stored Procedure: This stored procedure must return the unit price for a product when a product code is supplied. The unit price must include a dollar sign at the beginning. In addition, the unit price must contain a comma every three digits to the left of the decimal point, and must display two digits to the left of the decimal point. The stored procedure must not throw errors, even if the product code contains invalid data.

End of Repeated Scenario

You are creating the queries for Report1 and Report2.

You need to create the objects necessary to support the queries.

Which object should you use to join the SalesSummary table with the other tables that each report uses? To answer, drag the appropriate objects to the correct reports. each object 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.

Objects	Answer area						
<div>view</div>	<table><thead><tr><th>Report</th><th>Object</th></tr></thead><tbody><tr><td>Report1</td><td><div>Object</div></td></tr><tr><td>Report2</td><td><div>Object</div></td></tr></tbody></table>	Report	Object	Report1	<div>Object</div>	Report2	<div>Object</div>
Report	Object						
Report1	<div>Object</div>						
Report2	<div>Object</div>						
<div>indexed view</div>							
<div>subquery</div>							
<div>scalar function</div>							
<div>table-valued function</div>							
<div>stored procedure</div>							
<div>derived table</div>							
<div>common table expression (CTE)</div>							

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Box 1: common table expression (CTE)
A common table expression (CTE) can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query. Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.
A CTE can be used to:
From Scenario: Report1: This report joins data from SalesSummary with the Employee table and other tables. You plan to create an object to support Report1. The object has the following requirements:
Box 2: view
From scenario: Report2: This report joins data from SalesSummary with the Employee table and other tables. You plan to create an object to support Report1. The object has the following requirements:
References: [https://technet.microsoft.com/en-us/library/ms190766\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190766(v=sql.105).aspx)

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

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 table variable in the stored procedure. Does this meet the goal?

- A. Yes
- B. No

Answer: B

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.

Statements

SELECT Employees.ManagerId, Employees.EmployeeId, EmployeeLevel+1
FROM Employees
JOIN Managers ON Employees.EmployeeId = Managers.ManagerId

WITH Managers AS (
SELECT*
FROM Managers
ORDER BY ManagerID

SELECT ManagerId, EmployeeId, 0 AS EmployeeLevel
FROM Employees
WHERE ManagerId IS NULL

UNION ALL

UNION

<

>

Answer Area

>

<

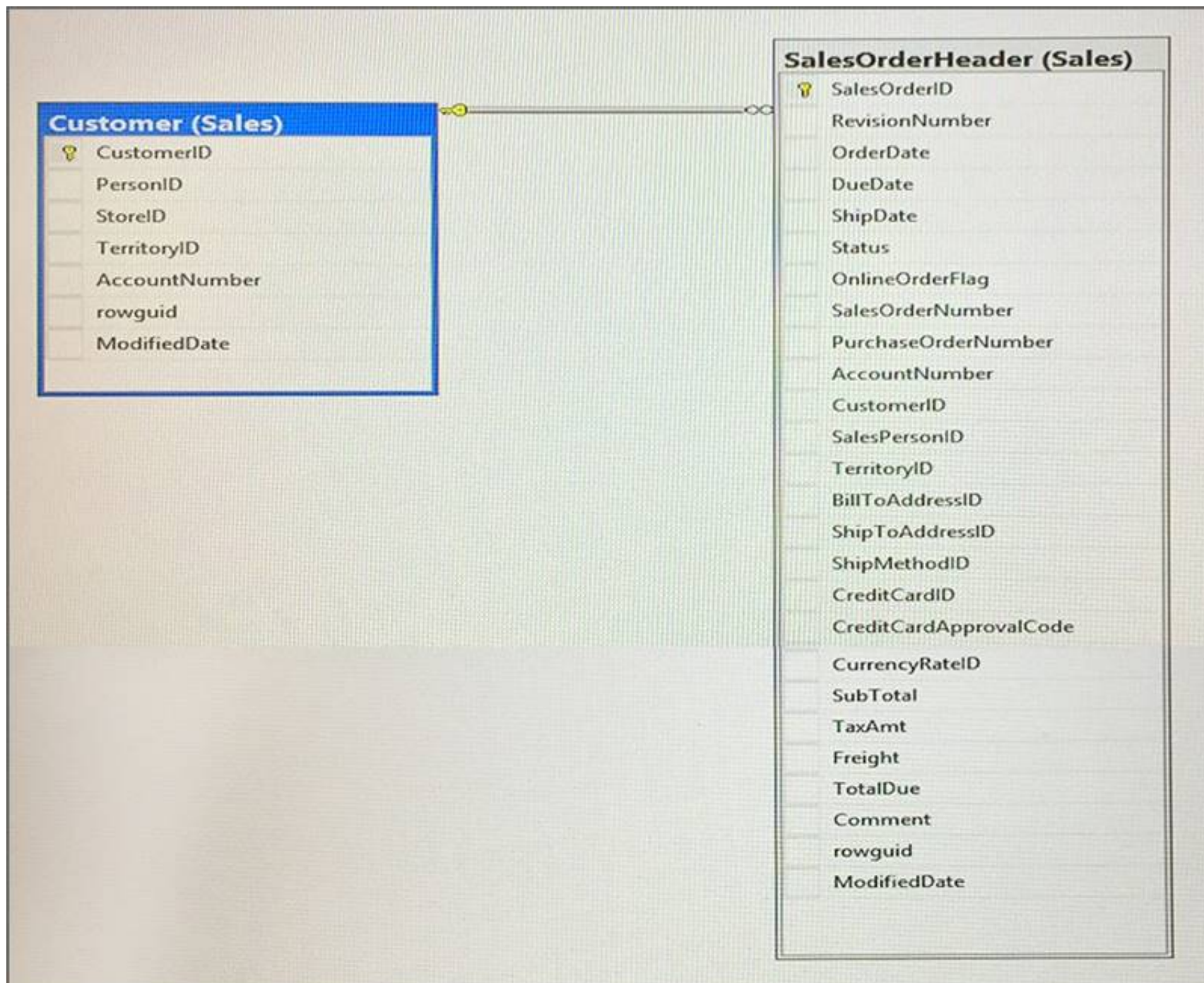
- A. Mastered
- B. Not Mastered

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
```

- 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

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 function that calculates the highest tax rate charged for an item in a specific order. Which five 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.

Transact-SQL segments

RETURNS decimal(18,2)

CREATE FUNCTION Sales.CalculateTaxRate ()

CREATE FUNCTION Sales.CalculateTaxRate (
 @OrderID int
)

RETURN @CalculatedRate
 END

SET @CalculatedTaxRate = (
 SELECT 1 + (MAX(TaxRate)
 / 100)
 FROM Sales.OrderLines
 WHERE OrderID = @OrderID
)

RETURNS Table
 END

AS
 BEGIN
 declare @CalculatedTaxRate
 decimal(18,2)

Answer Area

⏪

⏩

⏴

⏵

- A. Mastered
 B. Not Mastered

Answer: A

Explanation:

Box 1: CREATE FUNCTION...@OrderID
 Include definition for the ...@OrderID parameter. Box 2: RETURNS decimal(18,2)
 The function is defined to return a scalar value. Box 3: AS BEGIN ...
 Declare the local variables of the function. Box 4: SET @CalculatedTaxRate = (.. Calculate the tax rate.
 Box 5: RETURN @CalculatedRate END Return a scalar value.
 References: <https://msdn.microsoft.com/en-us/library/ms186755.aspx>

NEW QUESTION 7

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 8

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 run the following Transact-SQL statement:

```
CREATE TABLE Customers (  
    CustomerID int NOT NULL PRIMARY KEY CLUSTERED,  
    FirstName nvarchar(100) NOT NULL,  
    LastName nvarchar(100) NOT NULL,  
    TaxIdNumber varchar(20) NOT NULL,  
    Address nvarchar(1024) NOT NULL,  
    AnnualRevenue decimal(19,2) NOT NULL,  
    DateCreated datetime2(2) NOT NULL,  
    ValidFrom datetime2(2) GENERATED ALWAYS AS ROW START NOT NULL,  
    ValidTo datetime2(2) GENERATED ALWAYS AS ROW END NOT NULL,  
    PERIOD FOR SYSTEM_TIME(ValidFrom, ValidTo)  
)  
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = CustomersHistory))
```

You need to return the total annual revenue for all customers, followed by a row for each customer that shows the customer's name and annual revenue. Which Transact-SQL statement should you run?

A

```
SELECT FirstName, LastName, SUM(AnnualRevenue)  
FROM Customers  
GROUP BY GROUPING SETS((FirstName, LastName, AnnualRevenue), ())  
ORDER BY FirstName, LastName, AnnualRevenue
```

B

```
SELECT FirstName, LastName, Address  
FROM Customers  
FOR SYSTEM_TIME ALL ORDER BY ValidFrom
```

C

```
SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo  
FROM Customers AS c  
ORDER BY c.CustomerID  
FOR JSON AUTO, ROOT('Customers')
```

D

```
SELECT * FROM (SELECT CustomerID, FirstName, LastName, Address, AnnualRevenue, DateCreated  
FROM Customers) AS Customers PIVOT(AVG(AnnualRevenue)  
FOR DateCreated IN([2014])) AS PivotCustomers  
ORDER BY LastName, FirstName
```


E

```
SELECT CustomerID, AVG(AnnualRevenue)
AS AverageAnnualRevenue, FirstName, LastName, Address, DateCreated
FROM Customers WHERE YEAR(DateCreated) >= 2014
GROUP BY CustomerID, FirstName, LastName, Address, DateCreated
```

F

```
SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
FROM Customers AS c ORDER BY c.CustomerID
FOR XML PATH ('CustomerData'), root ('Customers')
```

G

```
SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers FOR SYSTEM_TIME
BETWEEN '2014-01-01 00:00:00.000000' AND '2015-01-01 00:00:00.000000'
```

H

```
SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers
WHERE DateCreated
BETWEEN '20140101' AND '20141231'
```

- 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: A

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 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 deposit and 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 CustNoFROMtblDepositAcctUNION 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: C

Explanation:

Would list the customers with duplicates, which would equal the number of accounts.

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 Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (
    ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    ProductName nvarchar(100) NULL,
    UnitPrice decimal(18, 2) NOT NULL,
    UnitsInStock int NOT NULL,
    UnitsOnOrder int NULL
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    INSERT INTO Products (ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
    VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products (ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
            VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
        IF @@ERROR = 51000
            THROW
    END CATCH
END
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

A transaction is correctly defined for the INSERT INTO ..VALUES statement, and if there is an error in the transaction it will be caught and the transaction will be rolled back. However, error number 51000 will not be returned, as it is only used in an IF @@ERROR = 51000 statement.

Note: @@TRANCOUNT returns the number of BEGIN TRANSACTION statements that have occurred on the current connection.

References: <https://msdn.microsoft.com/en-us/library/ms187967.aspx>

NEW QUESTION 14

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 Sales.Orders.

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(Cust.CustomerID)
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: A

NEW QUESTION 15

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

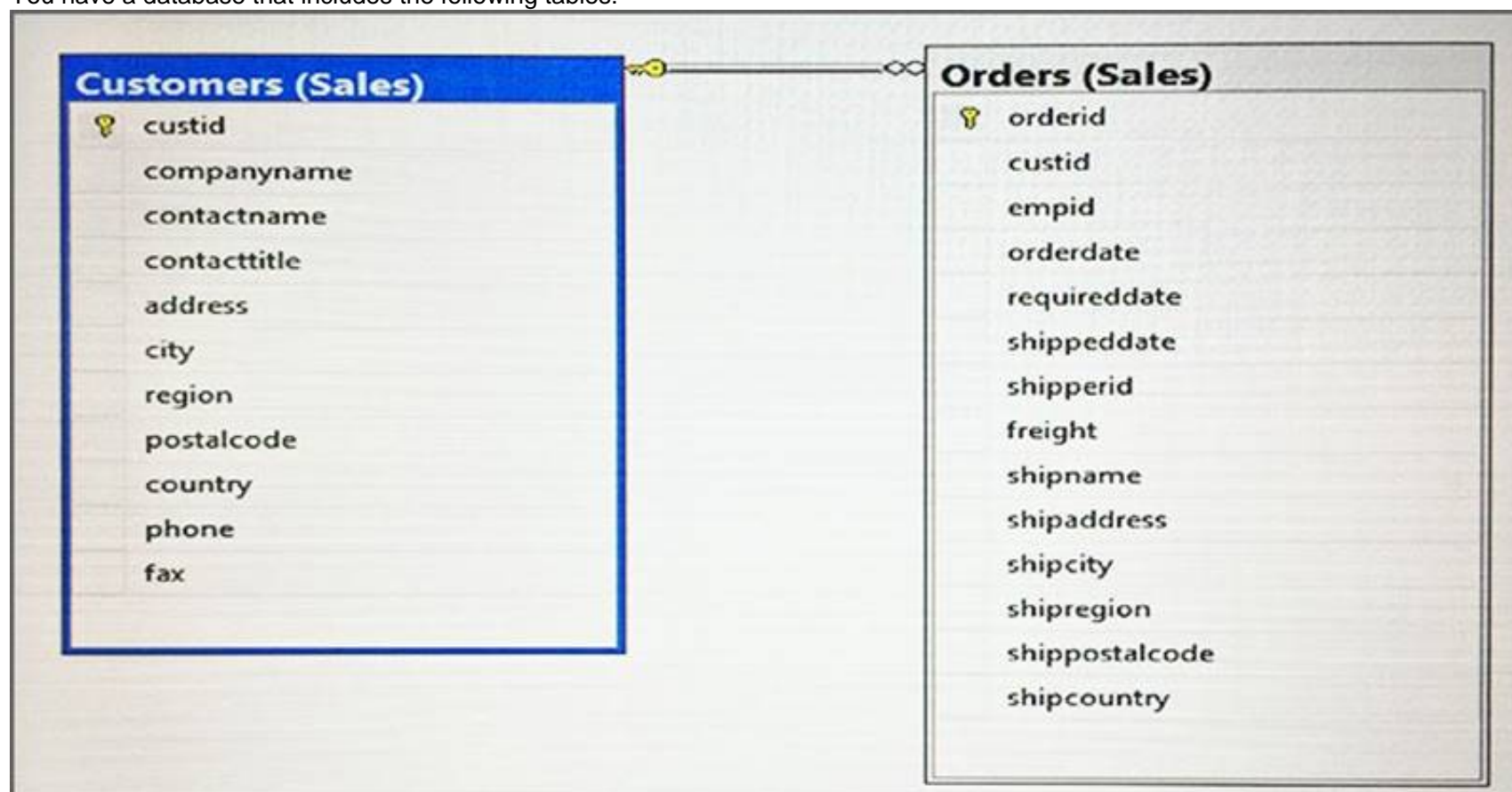
Explanation:

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](https://technet.microsoft.com/en-us/library/ms191165(v=sql.105).aspx)

NEW QUESTION 16

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.

Transact-SQL segments

GROUP BY c.custid

FROM sales.Customers AS c INNER JOIN sales.Orders AS o

ON c.orderid = o.orderid

SELECT c.custid AS CustomerID, MAX(o.orderdate) AS LastOrderDate

FROM sales.Customers AS c LEFT OUTER JOIN sales.Orders AS o

GROUP BY LastOrderDate

ON c.custid = o.custid

SELECT c.custid AS CustomerID, COALESCE (MAX(o.orderdate), '19000101') AS LastOrderDate

Answer Area

⬅

➡

⬆

⬇

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

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

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

NEW QUESTION 20

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 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 tblRoles(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(*) AS ActiveUserCount FROM tblRoles RCROSS JOIN (SELECT UserId, RoleId FROM tblUsers WHERE IsActive = 1) UWHERE U.RoleId = R.RoleIdGROUP BY R.RoleId, R.RoleName
- B. SELECT R.RoleName, COUNT(*) AS ActiveUserCount FROM tblRoles RLEFT JOIN (SELECTUserId, RoleId FROM tblUsers WHERE IsActive = 1) UON U.RoleId = R.RoleIdGROUP BY R.RoleId, R.RoleName
- C. SELECT R.RoleName, U.ActiveUserCount FROM tblRoles R CROSS JOIN(SELECT RoleId, COUNT(*) AS ActiveUserCountFROM tblUsers WHERE IsActive = 1 GROUP BY R.RoleId) U
- D. SELECT R.RoleName, ISNULL (U.ActiveUserCount,0) AS ActiveUserCountFROM tblRoles R LEFT JOIN (SELECT RoleId, COUNT(*) AS ActiveUserCountFROM tblUsers WHERE IsActive = 1 GROUP BY R.RoleId) U

Answer: B

NEW QUESTION 28

You develop and deploy 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 NULL,
    ReportingUser VARCHAR(50) NULL
)
```

Users report that searches for the product Salt also return results for the product salt. You need to ensure that the query returns the correct results.

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

NOTE: Each correct selection is worth one point.


```
DECLARE @product NVARCHAR(255)
. . .
SELECT
    Id
    ,
    Product
    Description
    DateCreated
    ReportingUser
FROM MSL.dbo.Bug
WHERE
```

▼ like @product

ASCII(Product)

CAST(Id AS TEXT)

TRANSLATED(Id,'CI','CS')

Product COLLATE SQL_Latin1_General_CP1_CS_AS

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

References:
<https://stackoverflow.com/questions/1831105/how-to-do-a-case-sensitive-search-in-where-clause-im-using-sql-s>

NEW QUESTION 30

You need to create an indexed view that requires logic statements to manipulate the data that the view displays. Which two database objects should you use? Each correct answer presents a complete solution.

- A. a user-defined table-valued function
- B. a CRL function
- C. a stored procedure
- D. a user-defined scalar function

Answer: AC

NEW QUESTION 32

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
CURRENT_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

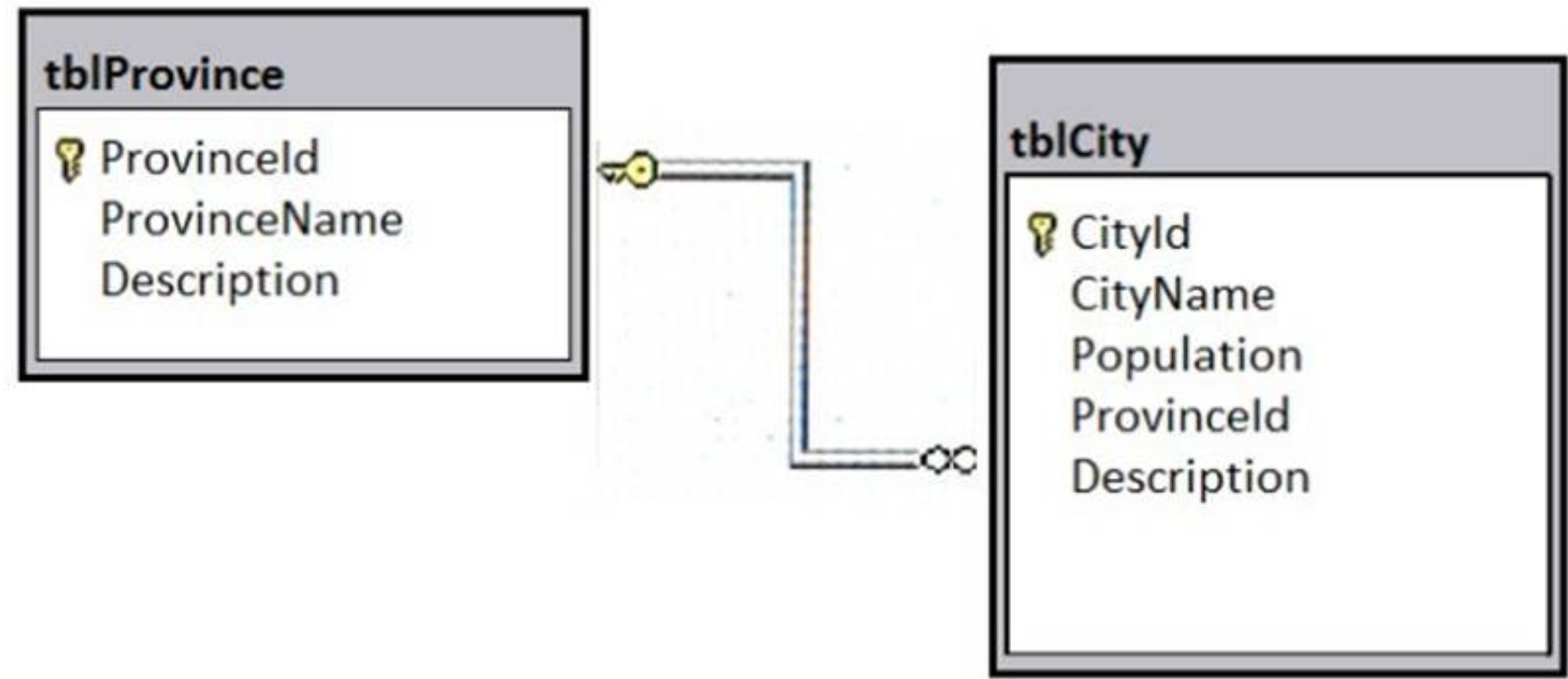
Answer: A

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 37

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

Answer: A

Explanation:

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

NEW QUESTION 38

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

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, A.DeliveryLocation.STDistance(B.DeliveryLocation) AS Dist FROM Sales.Customers AS A
CROSS JOIN Sales.Customers AS B
```

WHERE A.CustomerID = @custID AND A.CustomerID <> B.CustomerID ORDER BY Dist

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

- A. Yes
 B. No

Answer: B

NEW QUESTION 43

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 that was created by running the following Transact-SQL statement:

```
CREATE TABLE Products (
    ProductID int NOT NULL PRIMARY KEY,
    ProductName nvarchar(100) NULL,
    UnitPrice decimal(18, 2) NOT NULL,
    UnitsInStock int NOT NULL,
    UnitsOnOrder int NULL
)
```

The Products table includes the data shown in the following table:

ProductID	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	ProductA	10.00	10	15
2	ProductB	30.00	20	Null
3	ProductC	15.00	5	20

TotalUnitPrice is calculated by using the following formula: TotalUnitPrice = UnitPrice * (UnitsInStock + UnitsOnOrder)

You need to ensure that the value returned for TotalUnitPrice for ProductB is equal to 600.00. Solution: You run the following Transact-SQL statement:

```
SELECT ProductName, UnitPrice*(UnitsInStock+COALESCE(UnitsOnOnrder,0)) AS
TotalUnitPrice FROM Products
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: A

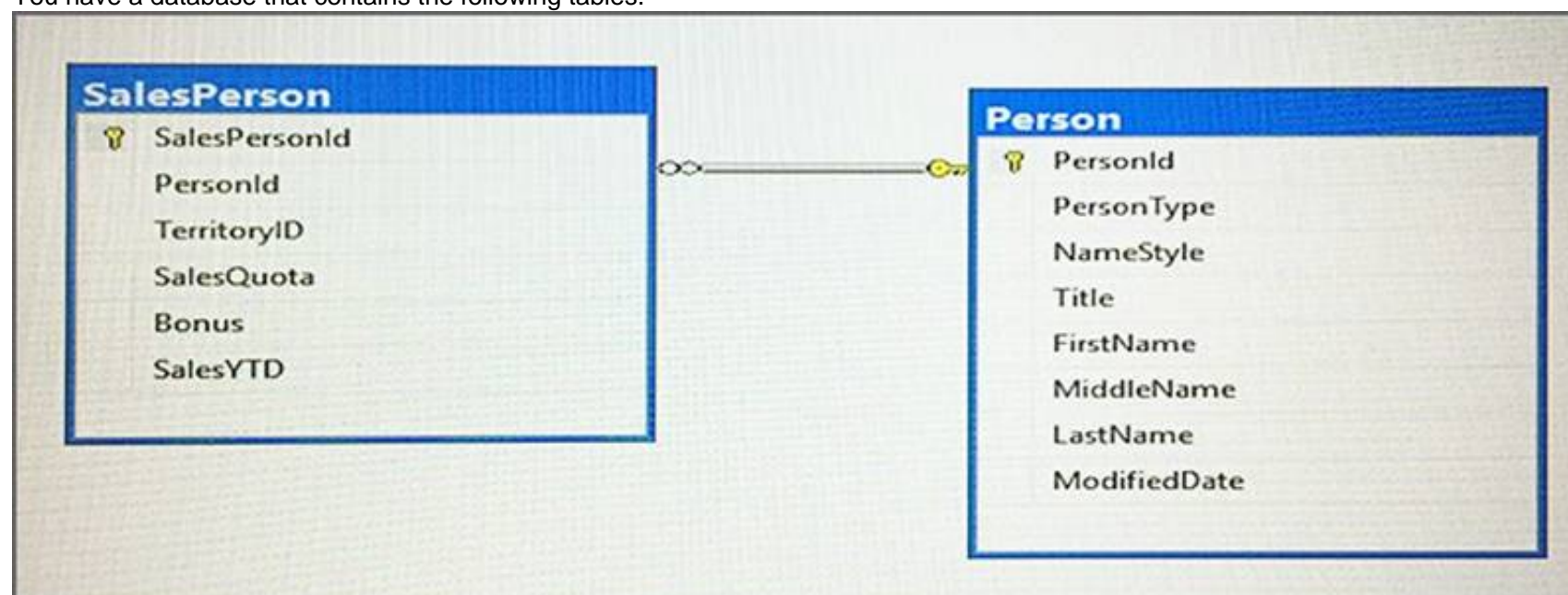
Explanation:

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 48

You have a database that contains the following tables.



You need to create a query that lists the lowest-performing salespersons based on the current year-to-date sales period. The query must meet the following requirements:

- Return a column named Fullname that includes the salesperson FirstName, a space, and then LastName.
- Include the current year-to-date sales for each salesperson.
- Display only data for the three salespersons with the lowest year-to-year sales values.
- Exclude salespersons that have no value for TerritoryID. Construct the query using the following guidelines:
- Use the first letter of a table name as the table alias.
- Use two-part column names.
- Do not surround object names with square brackets.
- Do not use implicit joins.
- Use only single quotes for literal text.
- Use aliases only if required.

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
CURRENT_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
2 FROM Person AS P INNER JOIN SalesPerson AS S
3 ON P.PersonID = S.SalesPersonID
4 WHERE
```

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

- A. SELECT TOP 3(p.FirstName + '' ' + p.LastName) AS FullName, s.SalesYTD FROM Person AS p INNER JOIN SalesPerson AS s ON p.PersonID = s.PersonID WHERE
B. TerritoryID IS NOT NULL ORDER BY
C. SalesYTD DESC

Answer: A

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 database that contains several connected tables. The tables contain sales data for customers in the United States only. You have the following partial query for the database. (Line numbers are included for reference only.)

```
01 SELECT CountryName, StateProvinceName, CityName, Quantity*UnitPrice as TotalSales
02 FROM Sales
03
04 ORDER BY CountryName, StateProvinceName, CityName
```

You need to complete the query to generate the output shown in the following table.

CountryName	StateProvinceName	CityName	TotalSales
NULL	NULL	NULL	\$23395792.75
Unites States	NULL	NULL	\$23395792.75
Unites States	Alabama	NULL	\$646508.75
Unites States	Alabama	Bazemore	\$34402.00
Unites States	Alabama	Belgreen	\$51714.65

Which statement clause should you add at line 3?

- A. GROUP BY
B. MERGE
C. GROUP BY ROLLUP
D. LEFT JOIN
E. GROUP BY CUBE
F. CROSS JOIN
G. PIVOT
H. UNPIVOT

Answer: E

Explanation:

Example of GROUP BY CUBE result set:

In the following example, the CUBE operator returns a result set that has one grouping for all possible combinations of columns in the CUBE list and a grand total grouping.

Region	Country	Store	SalesPersonID	Total Sales
NULL	NULL	NULL	NULL	254013.6014
NULL	NULL	NULL	287	28461.1854
NULL	NULL	NULL	288	17073.0655
NULL	NULL	NULL	290	208479.3505
NULL	NULL	Spa and Exercise Outfitters	NULL	236210.9015
NULL	NULL	Spa and Exercise Outfitters	287	27731.551
NULL	NULL	Spa and Exercise Outfitters	290	208479.3505
NULL	NULL	Versatile Sporting Goods Company	NULL	17802.6999
NULL	NULL	Versatile Sporting Goods Company	287	729.6344
NULL	NULL	Versatile Sporting Goods Company	288	17073.0655
NULL	DE	NULL	NULL	17802.6999
NULL	DE	NULL	287	729.6344
NULL	DE	NULL	288	17073.0655
NULL	DE	Versatile Sporting Goods Company	NULL	17802.6999
NULL	DE	Versatile Sporting Goods Company	287	729.6344

References: [https://technet.microsoft.com/en-us/library/bb522495\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/bb522495(v=sql.105).aspx)

NEW QUESTION 52

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 tblRoles(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 55

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 57

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 global temporary table in the stored procedure. Does this meet the goal?

- A. Yes
B. No

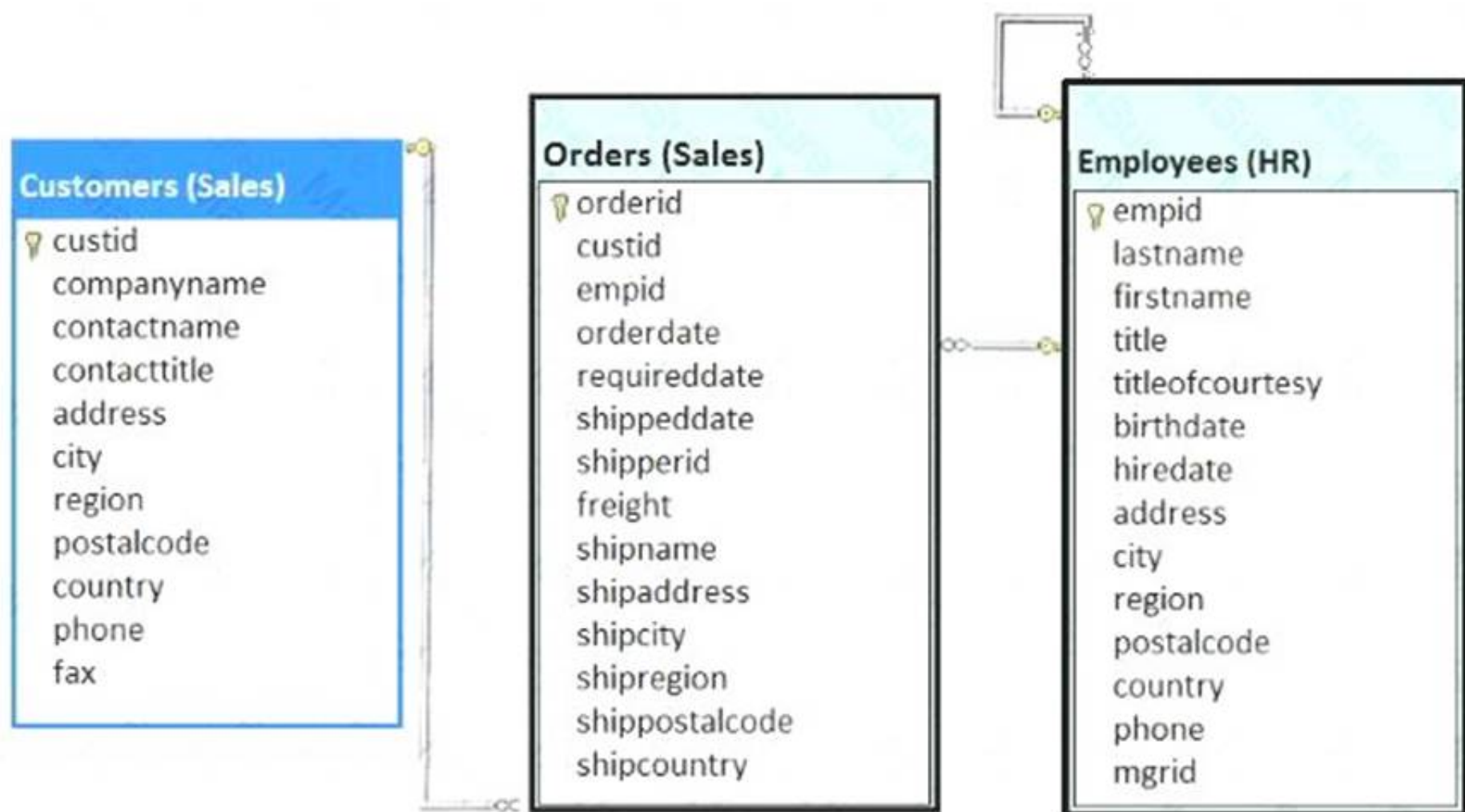
Answer: A

NEW QUESTION 60

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 includes the tables shown in the exhibit (Click the Exhibit button.)



You need to create a Transact-SQL query that returns the following information:

- the customer number
- the customer contact name
- the date the order was placed, with a name of DateofOrder
- a column named Salesperson, formatted with the employee first name, a space, and the employee last name
- orders for customers where the employee identifier equals 4

The output must be sorted by order date, with the newest orders first. The solution must return only the most recent order for each customer. Solution: You run the following Transact-SQL statement:

```
SELECT c.custid, contactname, MAX(orderdate) AS DateofOrder,
e.firstname + ' ' + e.lastname AS Salesperson
FROM Sales.Customers AS c
INNER JOIN Sales.Orders AS o ON c.custid = o.custid
INNER JOIN HR.Employees AS e ON o.empid = e.empid
WHERE o.empid = 4
GROUP BY c.custid, contactname, firstname, lastname
ORDER BY DateofOrder DESC
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: A

Explanation:

The MAX(orderdate) in the SELECT statement makes sure we return only the most recent order. A WHERE o.empid =4 clause is correctly used. GROUP BY is also required.

NEW QUESTION 65

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.

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
    
```

- 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 68

You create a table to track sales persons by running the following Transact-SQL statement:

```
CREATE TABLE SalesPerson(
  ID INT NOT NULL,
  TerritoryID INTNULL,
  Sales MONEY NOT NULL,
  EntryDate DATETIME NOT NULL
)
```

You need to create a report that shows the sales people within each territory for each year. The report must display sales people in order by highest sales amount. 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.

Transact-SQL segments

Sales

TerritoryID

RANK() OVER

GROUP BY

EntryDate

RANKING

PARTITION BY

Answer Area

SELECT

ID

TerritoryID,

Sales,

YEAR(EntryDate),

Segment

(

Segment

Segment

,

YEAR(

Segment

) ORDER BY

Segment

)

FROM SalesPerson

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Transact-SQL segments

Sales

TerritoryID

RANK() OVER

GROUP BY

EntryDate

RANKING

PARTITION BY

Answer Area

SELECT

ID

TerritoryID,

Sales,

YEAR(EntryDate),

RANK() OVER

(

PARTITION BY

TerritoryID

,

YEAR(

EntryDate

) ORDER BY

Sales

)

FROM SalesPerson

NEW QUESTION 69

You have a database containing the following tables: Servers

Column	Data type	Notes
ServerID	int	primary key
DNS	nvarchar(100)	does not allow null values

Errors

Column	Data type	Notes
ErrorID	int	primary key
ServerID	int	does not allow null values, foreign key to Servers table
LogMessage	nvarchar(max)	does not allow null values

You have a user-defined, scalar function named IPLookup that takes a DNS name as a parameter and returns the IP address of the server. You have an additional user-defined, scalar function named DNSLookup, that takes an IP address as a parameter and returns a DNS name.

You create a view named vwErrors by running the following Transact-SQL statement:

```
CREATE VIEW vwErrors
AS
    SELECT ErrorID, IPLookup(DNS) as IP, LogMessage
    FROM Errors
    INNER JOIN Servers ON Errors.ServerID = Servers.ServerID
```

You need to insert data by using the view.

How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct location. Each Transact-SQL segments 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.

Transact-SQL segments

- WITH APPEND
- AFTER INSERT
- INSTEAD OF INSERT
- FROM inserted
- FROM vwErrors
- dbo.DNSLookup(IP)
- Servers.IP

Answer Area

```
CREATE TRIGGER newErrorTrg on vwErrors
AS
BEGIN
    INSERT INTO Errors
        SELECT ErrorID, Servers.ServerID, LogMessage
        INNER JOIN Servers on Servers.DNS =
END
```

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

References: <https://docs.microsoft.com/en-us/sql/t-sql/queries/output-clause-transact-sql>

NEW QUESTION 71

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
CURRENT_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
2. FROM Sales.Products AS P

Use the Check Syntax button to verify your work. Any syntax or spelling errors will be reported by line and character position. You may check syntax as many times as needed.

- A. Mastered
B. Not Mastered

Answer: A

Explanation:

1. SELECT avg(P.ProductPrice) AS Average, min(P.ProductsInStock) AS LowestNumber, max(P.ProductPrice) AS HighestPrice
2. FROM Sales.Products AS P Make the additions to line 1.

References: <https://www.mssqltips.com/sqlservertip/4424/max-min-and-avg-sql-server-functions/>

NEW QUESTION 76

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 create a table by running the following Transact-SQL statement:

```
CREATE TABLE Customers (  
    CustomerID int NOT NULL PRIMARY KEY CLUSTERED,  
    FirstName nvarchar(100) NOT NULL,  
    LastName nvarchar(100) NOT NULL,  
    TaxIdNumber varchar(20) NOT NULL,  
    Address nvarchar(1024) NOT NULL,  
    AnnualRevenue decimal(19,2) NOT NULL,  
    DateCreated datetime2(2) NOT NULL,  
    ValidFrom datetime2(2) GENERATED ALWAYS AS ROW START NOT NULL,  
    ValidTo datetime2(2) GENERATED ALWAYS AS ROW END NOT NULL,  
    PERIOD FOR SYSTEM_TIME(ValidFrom, ValidTo)  
)  
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = CustomersHistory))
```

You need to audit all customer data.

Which Transact-SQL statement should you run?

- A** `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, AnnualRevenue, DateCreated
FROM Customers
GROUP BY GROUPING SETS(FirstName, LastName), (Address), (CustomerID, AnnualRevenue), (CustomerID), (())
ORDER BY CustomerID, FirstName, LastName, Address, AnnualRevenue`
- B** `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, AnnualRevenue, DateCreated, ValidFrom, ValidTo
FROM Customers
FOR SYSTEM_TIME ALL ORDER BY ValidFrom`
- C** `SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
FROM Customers AS c
ORDER BY c.CustomerID
FOR JSON AUTO, ROOT('Customers')`
- D** `SELECT * FROM (SELECT CustomerID, FirstName, LastName, Address, AnnualRevenue, DateCreated
FROM Customers) AS Customers PIVOT(AVG(AnnualRevenue)
FOR DateCreated IN([2014])) AS PivotCustomers
ORDER BY LastName, FirstName`
- E** `SELECT CustomerID, AVG(AnnualRevenue)
AS AverageAnnualRevenue, FirstName, LastName, Address, DateCreated
FROM Customers WHERE YEAR(DateCreated) >= 2014
GROUP BY CustomerID, FirstName, LastName, Address, DateCreated`
- F** `SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
FROM Customers AS c ORDER BY c.CustomerID
FOR XML PATH ('CustomerData'), root ('Customers')`
- G** `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers FOR SYSTEM_TIME
BETWEEN '2014-01-01 00:00:00.000000' AND '2015-01-01 00:00:00.000000'`
- H** `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers
WHERE DateCreated
BETWEEN '20140101' AND '20141231'`

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

Answer: B

Explanation:

The FOR SYSTEM_TIME ALL clause returns all the row versions from both the Temporal and History table. Note: A system-versioned temporal table defined through is a new type of user table in SQL Server 2016, here defined on the last line WITH (SYSTEM_VERSIONING = ON..., is designed to keep a full history of data changes and allow easy point in time analysis.

To query temporal data, the SELECT statement FROM<table> clause has a new clause FOR SYSTEM_TIME with five temporal-specific sub-clauses to query data across the current and history tables.

References: <https://msdn.microsoft.com/en-us/library/dn935015.aspx>

NEW QUESTION 78

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 create a table by running the following Transact-SQL statement:


```
CREATE TABLE Customers (  
    CustomerID int NOT NULL PRIMARY KEY CLUSTERED,  
    FirstName nvarchar(100) NOT NULL,  
    LastName nvarchar(100) NOT NULL,  
    TaxIdNumber varchar(20) NOT NULL,  
    Address nvarchar(1024) NOT NULL,  
    AnnualRevenue decimal(19,2) NOT NULL,  
    DateCreated datetime2(2) NOT NULL,  
    ValidFrom datetime2(2) GENERATED ALWAYS AS ROW START NOT NULL,  
    ValidTo datetime2(2) GENERATED ALWAYS AS ROW END NOT NULL,  
    PERIOD FOR SYSTEM_TIME(ValidFrom, ValidTo)  
)  
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = CustomersHistory))
```

You are developing a report that displays customer information. The report must contain a grand total column. You need to write a query that returns the data for the report.

Which Transact-SQL statement should you run?

- A `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, AnnualRevenue, DateCreated
FROM Customers
GROUP BY GROUPING SETS((FirstName, LastName), (Address), (CustomerID, AnnualRevenue), (CustomerID), ())
ORDER BY CustomerID, FirstName, LastName, Address, AnnualRevenue`
- B `SELECT FirstName, LastName, Address
FROM Customers
FOR SYSTEM_TIME ALL ORDER BY ValidFrom`
- C `SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
FROM Customers AS c
ORDER BY c.CustomerID
FOR JSON AUTO, ROOT('Customers')`
- D `SELECT * FROM (SELECT CustomerID, FirstName, LastName, Address, AnnualRevenue, DateCreated
FROM Customers) AS Customers PIVOT(AVG(AnnualRevenue)
FOR DateCreated IN([2014])) AS PivotCustomers
ORDER BY LastName, FirstName`
- E `SELECT CustomerID, AVG(AnnualRevenue)
AS AverageAnnualRevenue, FirstName, LastName, Address, DateCreated
FROM Customers WHERE YEAR(DateCreated) >= 2014
GROUP BY CustomerID, FirstName, LastName, Address, DateCreated`
- F `SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
FROM Customers AS c ORDER BY c.CustomerID
FOR XML PATH ('CustomerData'), root ('Customers')`
- G `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers FOR SYSTEM_TIME
BETWEEN '2014-01-01 00:00:00.000000' AND '2015-01-01 00:00:00.000000'`
- H `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers
WHERE DateCreated
BETWEEN '20140101' AND '20141231'`

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: E

Explanation:

Calculate aggregate column through AVG function and GROUP BY clause.

NEW QUESTION 83

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 85

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.

Transact-SQL segments

@List NVARCHAR(MAX) = ''

@List NVARCHAR(MAX)

@List TABLE

@List=Product+ ',' + @List

@List=@List+ ',' + Product

@List COALESCE(@List, ',', Product)

Answer Area

DECLARE

Transact-SQL segment

SELECT

Transact-SQL segment

From Bug WHERE ReportingUser = User1
SELECT @List

- 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 89

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 Sales.Orders.

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 90

You have a table named HumanResources.Employee. You configure the table to use a default history table that contains 10 years of data.

You need to write a query that retrieves the values of the BusinessEntityID and JobTitle fields. You must retrieve all historical data up to January 1, 2017 where the value of the BusinessEntityID column equals 4.

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

Transact-SQL segments

SELECT TOP 4 BusinessEntityID,
JobTitle

FOR SYSTEM_TIME BETWEEN
('2016-01-01' and '2017-01-01')

SELECT BusinessEntityID, JobTitle

FROM HumanResources.Employee.History

FROM HumanResources.Employee

WHERE BusinessEntityID = 4

WHERE BusinessEntityID = 4 and His-
toryData IS NOT NULL

FOR SYSTEM_TIME CONTAINED IN (' ',
'2017-01-01')

Answer Area



- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

References:
<https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-t>

NEW QUESTION 94

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 several connected tables. The tables contain sales data for customers in the United States only. You have the following partial query for the database. (Line numbers are included for reference only.)

```
01 SELECT CountryName, StateProvinceName, CityName, Quantity*UnitPrice as TotalSales
02 FROM Sales
03
```

You need to complete the query to generate the output shown in the following table.

CountryName	StateProvinceName	CityName	TotalSales
United States	Wyoming	Yoder	\$7638.11
United States	Wyoming	NULL	\$1983745.99
United States	NULL	NULL	\$2387435981.22
NULL	NULL	NULL	\$2387435981.22

Which statement clause should you add at line 3?

- A. GROUP BY
- B. MERGE
- C. GROUP BY ROLLUP
- D. LEFT JOIN
- E. GROUP BY CUBE
- F. CROSS JOIN
- G. PIVOT
- H. UNPIVOT

Answer: F

Explanation:

A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table. References: [https://technet.microsoft.com/en-us/library/ms190690\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190690(v=sql.105).aspx)

NEW QUESTION 97

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 * 1.1
WHERE ListPrice
BETWEEN .01 and 99.99
```

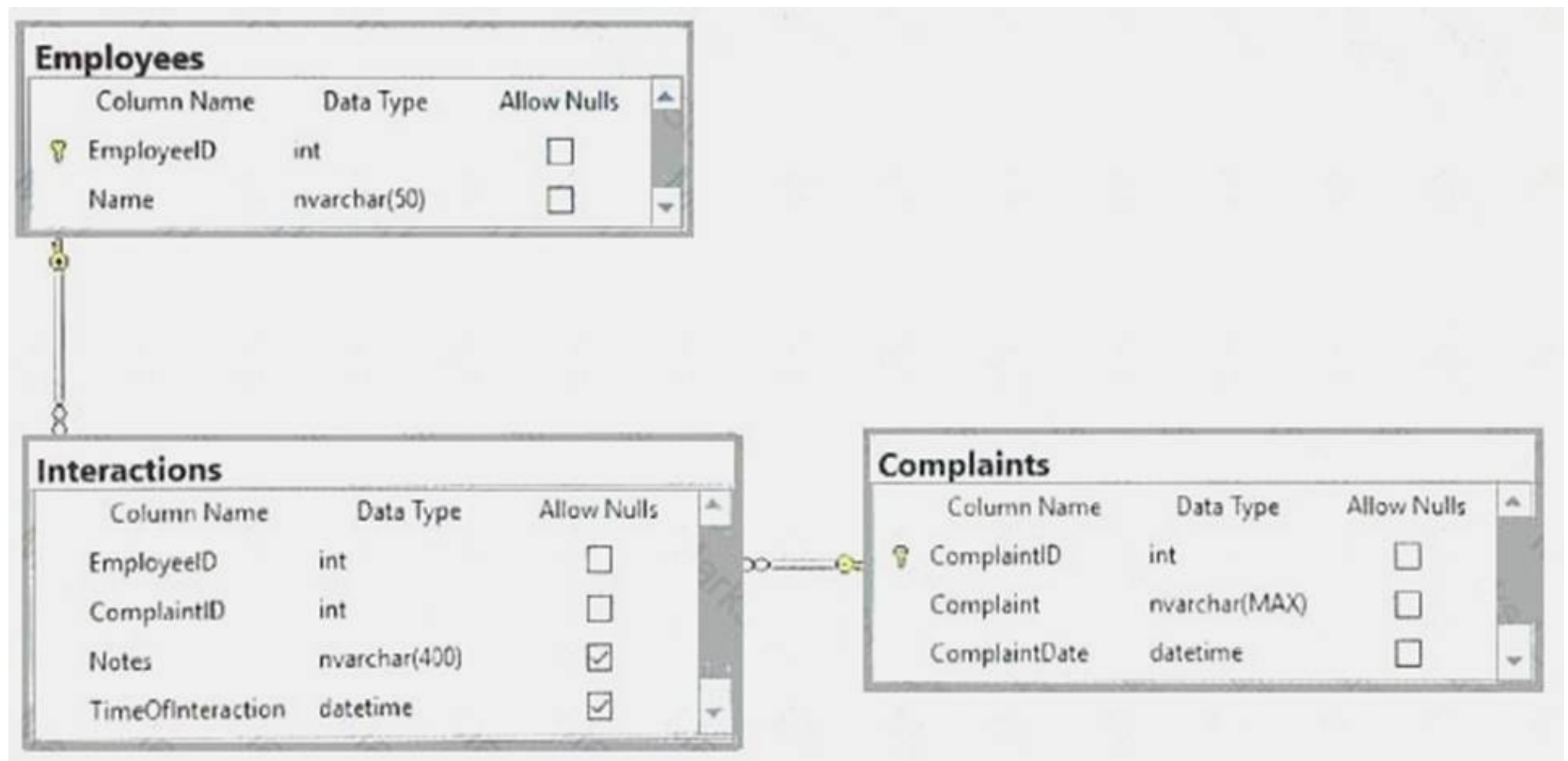
Does the solution meet the goal?

- A. Yes
- B. No

Answer: A

NEW QUESTION 100

You have a database that contains the following tables.



You need to create a query that returns each complaint, the names of the employees handling the complaint, and the notes on each interaction. The Complaint field must be displayed first, followed by the employee's name and the notes. Complaints must be returned even if no interaction has occurred.

Construct the query using the following guidelines:

- Use two-part column names.
- Use one-part table names.
- Use the first letter of the table name as its alias.
- Do not Transact-SQL functions.
- Do not use implicit joins.
- Do not surround object names with square brackets.

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.

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
CURRENT_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

1 SELECT c.Complaint, e.Name, i.Notes 2 FROM Complaints c
3 JOIN
4 JOIN

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

Check Syntax

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

1 SELECT c.Complaint, e.Name, i.Notes
2 FROM Complaints c
3 JOIN Interactions i ON c.ComplaintID = i.ComplaintID
4 JOIN Employees e ON i.EmployeeID = E.EmployeeID

NEW QUESTION 103

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 Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (  
    ProductID int IDENTITY (1, 1), NOT NULL PRIMARY KEY,  
    ProductName nvarchar (100), NULL,  
    UnitPrice decimal (18, 2) NOT NULL,  
    UnitsInStock int NOT NULL,  
    UnitsOnOrder int NULL  
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal (18, 2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    INSERT INTO Products (ProductName, UnitPrice, UnitsInStock, UnitsOnOrder)
    VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:

Insert product records as a single unit of work.

Return error number 51000 when a product fails to insert into the database.

If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
    @ProductName nvarchar (100),
    @UnitPrice decimal (18, 2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products (ProductName, UnitPrice, UnitsInStock, UnitsOnOrder)
            VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
        RAISERROR (51000,16, 1)
    END CATCH
END
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B





NEW QUESTION 106

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

Employees (HR)	
	empid
	lastname
	firstname
	title
	titleofcourtesy
	birthdate
	hiredate
	address
	city
	region
	postalcode
	country
	phone
	mgrid

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.

Transact-SQL segments		Answer Area
SET title = 'Customer Representative		
WHERE title = 'Sales Representative' AND city = 'Seattle' AND mgrid IS NOT NULL		
UPDATE HR.Employees	 	
SET city = 'Seattle' and mgrid = NULL		
INSERT INTO HR.Employees		
VALUES ('Customer Representative')		
WHERE title = 'Sales Representative'		
DELETE FROM HR.Employees		
		 

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

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

NEW QUESTION 110

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.

Multiple processes use the data from a table named Sales and place it in other databases across the organization. Some of the processes are not completely aware of the data types in the Sales table. This leads to data type conversion errors.

You need to implement a method that returns a NULL value id data conversion fails instead of throwing an error.

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: H

Explanation:

TRY_CONVERT returns a value cast to the specified data type if the cast succeeds; otherwise, returns null. References: <https://docs.microsoft.com/en-us/sql/t-sql/functions/try-convert-transact-sql>

NEW QUESTION 113

You have a database named DB1 that contains a temporal table named Sales.Customers.

You need to create a query that returns the credit limit that was available to each customer in DB1 at the beginning of 2017.

Which query should you execute?

A

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME CONTAINED IN ('2017-01-01 00:00:00');
```

B

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME AS OF '2017-01-01 00:00:00';
```

C

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME CONTAINED IN ('2016-12-31', '2017-01-01');
```

D

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME BETWEEN '2016-12-31' AND '2017-01-01');
```

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

Answer: B

NEW QUESTION 114

You have two tables named UserLogin and Employee respectively.

You need to create a Transact-SQL script that meets the following requirements:

- The script must update the value of the IsDeleted column for the UserLogin table to 1 if the value of the Id column for the UserLogin table is equal to 1.
- The script must update the value of the IsDeleted column of the Employee table to 1 if the value of the Id column is equal to 1 for the Employee table when an update to the UserLogin table throws an error.
- The error message "No tables updated!" must be produced when an update to the Employee table throws an error.

Which five 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.

Code segments	Answer Area
<pre>BEGIN TRY UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>	<div><div><div>⏮</div><div>⏭</div></div><div><div>⏮</div><div>⏭</div></div></div>
<pre>BEGIN CATCH RAISERROR ('No tables updated!', 16, 1) END CATCH</pre>	
<pre>UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1</pre>	
<pre>BEGIN CATCH</pre>	
<pre>BEGIN TRY UPDATE dbo.UserLogin SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>	
<pre>END CATCH</pre>	
<pre>BEGIN TRY UPDATE dbo.UserLogin SET IsDeleted = 1 WHERE Id = 1 UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>	

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Code segments	Answer Area
<pre>BEGIN TRY UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>	<pre>BEGIN TRY UPDATE dbo.UserLogin SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>
<pre>BEGIN CATCH RAISERROR ('No tables updated!', 16, 1) END CATCH</pre>	<pre>BEGIN CATCH .</pre>
<pre>UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1</pre>	<pre>BEGIN TRY UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>
<pre>BEGIN CATCH</pre>	<pre>BEGIN CATCH RAISERROR ('No tables updated!', 16, 1) END CATCH</pre>
<pre>BEGIN TRY UPDATE dbo.UserLogin SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>	<pre>END CATCH</pre>
<pre>END CATCH</pre>	
<pre>BEGIN TRY UPDATE dbo.UserLogin SET IsDeleted = 1 WHERE Id = 1 UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1 END TRY</pre>	

NEW QUESTION 115

You develop and deploy 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 NULL,
    ReportingUser VARCHAR(50) NULL
)
```

You need to ensure that users can perform searches of descriptions. Which Transact-SQL statement should you run?

- A. DECLARE @term NVARCHAR(255)
.
.
.
SELECT Id, Description
FROM Bug
WHERE CHARINDEX(@term, Description) > 0
- B. DECLARE @term NVARCHAR(255)
.
.
.
SELECT Id, Description
FROM Bug
WHERE DIFFERENCE(@term, Description) > 0
- C. DECLARE @term NVARCHAR(255)
.
.
.
SELECT Id, Description
FROM Bug
WHERE CONTAINS (Description, '%@term%')
- D. DECLARE @term NVARCHAR(255)
.
.
.
SELECT Id, Description
FROM Bug
WHERE CONTAINS (Description, @term)

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

Answer: D

Explanation:

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

NEW QUESTION 118

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 Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (  
    ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,  
    ProductName nvarchar(100) NULL,  
    UnitPrice decimal(18, 2) NOT NULL,  
    UnitsInStock int NOT NULL,  
    UnitsOnOrder int NULL  
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    INSERT INTO Products (ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
    VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    BEGIN TRY
        INSERT INTO Products (ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
        VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
    END TRY
    BEGIN CATCH
        THROW 51000, 'The product could not be created.', 1
    END CATCH
END
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: A

Explanation:

If the INSERT INTO statement raises an error, the statement will be caught and an error 51000 will be thrown. In this case no records will have been inserted.

Note:

You can implement error handling for the INSERT statement by specifying the statement in a TRY...CATCH construct.

If an INSERT statement violates a constraint or rule, or if it has a value incompatible with the data type of the column, the statement fails and an error message is returned.

References: <https://msdn.microsoft.com/en-us/library/ms174335.aspx>

NEW QUESTION 120

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 122

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 Returnvalue 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 125

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
CURRENT_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 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 129

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 NULL,
    ReportingUser VARCHAR(50) NULL
)
```

Users connect to an instance of the bug tracking application that is hosted in New York City. Users in Seattle must be able to display the local date and time for any bugs that they create.

You need to ensure that the DateCreated column displays correctly. Which Transact-SQL statement should you run?

- A. SELECT Id,Product,DateCreated AT TIME ZONE 'Pacific Standard Time' FROM Bug
- B. SELECT Id,Product, DATEADD(hh, -8, DateCreated) FROM Bug
- C. SELECT Id,Product, TODATETIMEOFFSET(DateCreated, -8) FROM Bug
- D. SELECT Id,Product,CAST(DateCreated AS DATETIMEOFFSET) FROM Bug

Answer: C

Explanation:

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

NEW QUESTION 133

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 function that accepts a CustomerID as a parameter and returns the following information:

- all customer information for the customer
- the total number of orders for the customer
- the total price of all orders for the customer
- the average quantity of items per order

How should you complete the function definition? To answer, drag the appropriate Transact-SQL segment to the correct locations. 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.

Transact-SQL segments

COUNT

SUM

AVG

ORDER BY

GROUP BY

RETURNS INT

RETURNS NULL ON NULL INPUT

RETURNS TABLE

Answer Area

```

CREATE FUNCTION Sales.GetCustomerInformation(@CustomerID int)
    Transact-SQL segment
AS
RETURN
(
    SELECT C.CustomerName, C.PhoneNumber, C.AccountOpenedDate,
    C.StandardDiscountPercentage, C.CreditLimit, C.IsOnCreditHold,
    Transact-SQL segment (O.OrderID) AS TotalNumberOfOrders,
    Transact-SQL segment (OL.UnitPrice) AS TotalOrderPrice,
    Transact-SQL segment (OL.Quantity) AS AverageOrderQuantity
    FROM Sales.Customers C
    JOIN Sales.Orders AS O ON O.CustomerID = C.CustomerID
    JOIN Sales.OrderLines AS OL ON OL.OrderID = O.OrderID
    WHERE C.CustomerID = @CustomerID
    Transact-SQL segment C.CustomerName, C.PhoneNumber, C.AccountOpenedDate,
    C.StandardDiscountPercentage, C.CreditLimit, C.IsOnCreditHold
)
        
```


- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Box1: RETURNS TABLE

The function should return the following information:

- all customer information for the customer
- the total number of orders for the customer
- the total price of all orders for the customer
- the average quantity of items per order

Box 2: COUNT
The function should return the total number of orders for the customer.

Box 3: SUM
The function should return the total price of all orders for the customer.

Box 4: AVG
The function should return the average quantity of items per order.

Need to use GROUP BY for the aggregate functions.

References: <https://msdn.microsoft.com/en-us/library/ms186755.aspx>

NEW QUESTION 136

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 clustered index on the primary key column. Does this meet the goal?

- A. Yes
- B. No

Answer: A

NEW QUESTION 141

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 * 1.1
WHERE ListPrice
BETWEEN 0 and 100
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 144

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

70-761 Practice Exam Features:

- * 70-761 Questions and Answers Updated Frequently
- * 70-761 Practice Questions Verified by Expert Senior Certified Staff
- * 70-761 Most Realistic Questions that Guarantee you a Pass on Your First Try
- * 70-761 Practice Test Questions in Multiple Choice Formats and Updates for 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The 70-761 Practice Test Here](#)