

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 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
SET XACT_ABORT ON
BEGIN TRY
BEGIN TRANSACTION
INSERT INTO Products (ProductName, UnitPrice, UnitsInStock, UnitsOnOrder)
VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
COMMIT TRANSACTION
END TRY
BEGIN CATCH
IF XACT_STATE () <> 0 ROLLBACK TRANSACTION
THROW 51000, 'The product could not be created,' 1
END CATCH
END
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 2

Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.

You have a database that contains tables named Customer_CRMSystem and Customer_HRSystem. Both tables use the following structure:

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

The tables include the following records: Customer_CRMSystem

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

Customer_HRSystem

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

Records that contain null values for CustomerCode can be uniquely identified by CustomerName. You need to display distinct customers that appear in both

tables.

Which Transact-SQL statement should you run?

A

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

B

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

C

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

D

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

E

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

F

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

G

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

H

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

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

Answer: H

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

NEW QUESTION 3

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 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

End of repeated scenario

You need to create a common table expression (CTE) that returns the total number of orders per year for each customer.

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

```
SELECT CustomerID, COUNT(OrderID) AS  
TotalOrders, OrderYear  
FROM Orders_CTE
```

```
WITH Orders_CTE (CustomerID, OrderID,  
OrderYear)
```

```
GROUP BY OrderYear, CustomerID
```

```
AS  
(  
    SELECT c.CustomerID, o.OrderID,  
    YEAR(o.OrderDate) AS OrderYear  
    FROM Sales.Customers c, Sales.Orders o  
    WHERE o.CustomerID = c.CustomerID  
)
```

```
ORDER BY CustomerID, OrderYear
```

```
MERGE Sales.Customers
```

Answer Area



Answer:

Explanation:

Transact-SQL segments

SELECT CustomerID, COUNT(OrderID) AS
TotalOrders, OrderYear
FROM Orders_CTE

WITH Orders_CTE (CustomerID, OrderID,
OrderYear)

GROUP BY OrderYear, CustomerID

AS
(
SELECT c.CustomerID, o.OrderID,
YEAR(o.OrderDate) AS OrderYear
FROM Sales.Customers c, Sales.Orders o
WHERE o.CustomerID = c.CustomerID
)

ORDER BY CustomerID, OrderYear

MERGE Sales.Customers

Answer Area

WITH Orders_CTE (CustomerID, OrderID,
OrderYear)

AS
(
SELECT c.CustomerID, o.OrderID,
YEAR(o.OrderDate) AS OrderYear
FROM Sales.Customers c, Sales.Orders o
WHERE o.CustomerID = c.CustomerID
)

SELECT CustomerID, COUNT(OrderID) AS
TotalOrders, OrderYear
FROM Orders_CTE

GROUP BY OrderYear, CustomerID


ORDER BY CustomerID, OrderYear


NEW QUESTION 4


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"/>	



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"/>	

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
view	Report
indexed view	Report1
subquery	Report2
scalar function	Object
table-valued function	Object
stored procedure	
derived table	
common table expression (CTE)	

Answer:

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 5

You are developing a training management application. You run the following Transact-SQL statement:

```
CREATE TABLE Evaluations(
    EvaluationID INT NOT NULL,
    EmployeeID INT NULL,
    CourseID INT NULL,
    EvalScore INT NULL)
```

You must create a report that returns course identifiers and the average evaluation score for each course. The result set must include only one score for each employee for each course.

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.

Answer Area

```
WITH Evals_CTE (CourseID, EmployeeID, EvalScore)
AS
(
    SELECT
        FROM Evaluations

)
SELECT
FROM Evals_CTE
GROUP BY CourseID
```

▼

DISTINCT CourseID, EmployeeID

DISTINCT CourseID, EmployeeID, EvalScore

(DISTINCT (CourseID, EmployeeID, EvalScore

CourseID, EmployeeID, EvalScore

▼

CourseID, AVG(DISTINCT EvalScore

CourseID, SUM(EvalScore

CourseID, AVG(EvalScore

CourseID, EmployeeID, EvalScore

Answer:

Explanation: **Answer Area**

```
WITH Evals_CTE (CourseID, EmployeeID, EvalScore)
AS
(
    SELECT
        FROM Evaluations

)
SELECT
FROM Evals_CTE
GROUP BY CourseID
```

▼

DISTINCT CourseID, EmployeeID

DISTINCT CourseID, EmployeeID, EvalScore

(DISTINCT (CourseID, EmployeeID, EvalScore

CourseID, EmployeeID, EvalScore

▼

CourseID, AVG(DISTINCT EvalScore

CourseID, SUM(EvalScore

CourseID, AVG(EvalScore

CourseID, EmployeeID, EvalScore

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 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.

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 identify the owner of each task by using the following rules:

- Return each task's owner if the task has an owner.
- If a task has no owner, but is associated with a project that has an owner, return the project's owner.
- Return the value -1 for all other cases.

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

Answer Area

SELECT T.TaskId, T.TaskName,

▼

ISNULL
COALESCE
CHOOSE

▼

T.UserId, P.UserId, -1
P.UserId, T.UserId, -1
-1, P.UserId, T.UserId
-1, T.UserId, P.UserId

) AS OwnerUserId

FROM Task T

▼

INNER JOIN
LEFT JOIN
RIGHT JOIN

Project P ON T.ProjectId = P.ProjectId

Answer:

Explanation: Box 1: COALESCE

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

Box 2: T.UserId, p.UserId, -1

- Return each task's owner if the task has an owner.
- If a task has no owner, but is associated with a project that has an owner, return the project's owner.
- Return the value -1 for all other cases.

Box 3: LEFT JOIN

The LEFT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match. Here the right side could be NULL as the projectID of the task could be NULL.

References:

<https://msdn.microsoft.com/en-us/library/ms190349.aspx>

http://www.w3schools.com/Sql/sql_join_right.asp

NEW QUESTION 7

You must create a report that shows the regions that have a factory but do not have a shipping center. You need to create the query for the report.

Which two Transact-SQL statements can you use? Each correct answer presents a complete solution.

A)

```
SELECT Factory.Region
FROM Factory
LEFT JOIN
ShippingCenter ON ShippingCenter.Region = Factory.Region
```

B)

```
SELECT Region
FROM Factory
WHERE Region NOT IN
(SELECT Region FROM ShippingCenter)
```

C)


```
SELECT Region
FROM Factory
WHERE NOT EXISTS
(SELECT * FROM ShippingCenter WHERE ShippingCenter.Region = Factory.Region)
```

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

Answer: BC

NEW QUESTION 8

You have a database for a banking system. The database has two tables named tblDepositAcct and tblLoanAcct that store deposit and loan accounts, respectively. Both tables contain the following columns:

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

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

- A. SELECT COUNT(*)
FROM (SELECT AcctNo
FROM tblDepositAcct
INTERSECT
SELECT AcctNo
FROM tblLoanAcct) R
- B. SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
UNION
SELECT CustNo
FROM tblLoanAcct) R
- C. SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
UNION ALL
SELECT CustNo
FROM tblLoanAcct) R
- D. SELECT COUNT (DISTINCT D.CustNo)
FROM tblDepositAcct D, tblLoanAcct L
WHERE D.CustNo = L.CustNo
- E. SELECT COUNT(DISTINCT L.CustNo)
FROM tblDepositAcct D
RIGHT JOIN tblLoanAcct L ON D.CustNo = L.CustNo
WHERE D.CustNo IS NULL
- F. SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
EXCEPT
SELECT CustNo
FROM tblLoanAcct) R
- G. SELECT COUNT (DISTINCT COALESCE(D.CustNo, L.CustNo))
FROM tblDepositAcct D
FULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo
WHERE D.CustNo IS NULL OR L.CustNo IS NULL
- H. SELECT COUNT(*)
FROM tblDepositAcct D
FULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo

- 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: F

Explanation: References:
<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/set-operators-except-and-intersect-transact-sql?vie>

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 have a database that contains a single table named tblVehicleRegistration. The table is defined as follows:

Column name	Data type	Description
VehicleId	int	the primary key for the table
RegistrationNumber	varchar(5)	a vehicle registration number that contains only letters and numbers
RegistrationDate	date	the vehicle registration date
UserId	int	an identifier for the vehicle owner

You run the following query:

```
SELECT UserId FROM tblVehicleRegistration
WHERE RegistrationNumber = 20012
AND RegistrationDate > '2016-01-01'
```

The query output window displays the following error message: “Conversion failed when converting the varchar value ‘AB012’ to data type int.” You need to resolve the error.

Solution: You modify the Transact-SQL statement as follows:

```
SELECT UserId FROM tblVehicleRegistration
WHERE RegistrationNumber = 20012
AND RegistrationDate > CONVERT(DATE, '2016-01-01', 120)
```

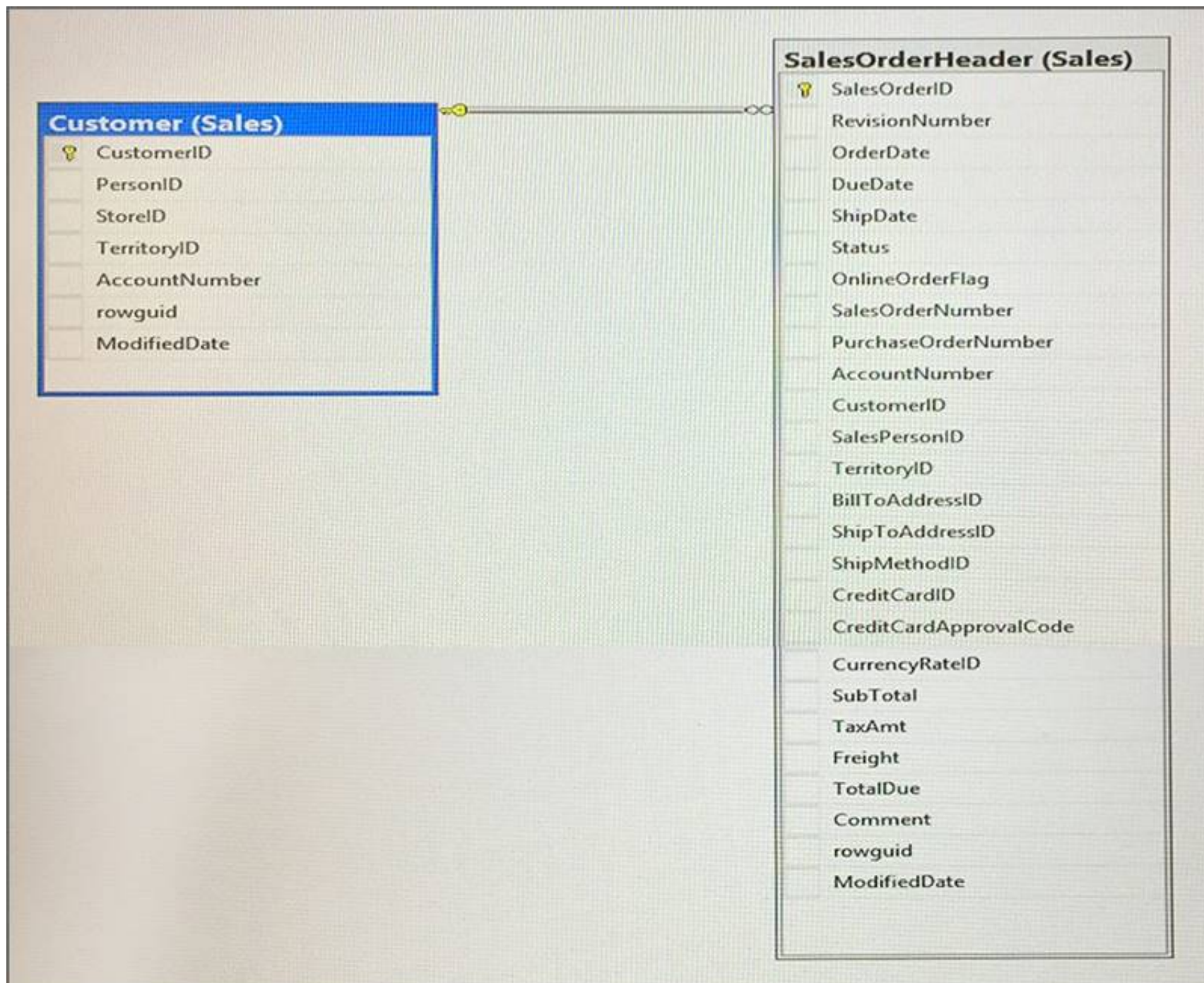
Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 10

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 10

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



Answer:

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 13

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

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

You are creating indexes in a data warehouse.

You have a dimension table named Table1 that has 10,000 rows. The rows are used to generate several reports. The reports join a column that is the primary key.

The execution plan contains bookmark lookups for Table1. You discover that the reports run slower than expected.

You need to reduce the amount of time it takes to run the reports.

Solution: You create a nonclustered index on the primary key column that includes the bookmark lookup columns.

Does this meet the goal?

A. Yes

B. No

Answer: B

NEW QUESTION 18

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.

Users report performance issues when they run the following query:

```
SELECT COUNT(*) AS TotalTestTasksCount FROM
(
    SELECT T.TaskId,T.TaskName FROM Task T
    WHERE SUBSTRING(T.TaskName,1,4) = 'TEST'
) AS R
```

You need to improve query performance and limit results to projects that specify an end date.

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

Answer Area

```
SELECT COUNT(*) AS TotalTestTasksCount FROM (
    SELECT T.TaskId,T.TaskName
    FROM Task T
    WHERE  LIKE 
    AND 
) AS R
```

Answer:

Explanation: **Answer Area**

```
SELECT COUNT(*) AS TotalTestTasksCount FROM (
    SELECT T.TaskId,T.TaskName
    FROM Task T
    WHERE  LIKE 
    AND 
) AS R
```

Wildcard character %: Any string of zero or more characters.

For example: If the LIKE '5%' symbol is specified, the Database Engine searches for the number 5 followed by any string of zero or more characters.

References: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/like-transact-sql>

NEW QUESTION 20

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 return normalized data for all customers that were added in the year 2014. 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`

- 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: G

Explanation: The following query searches for row versions for Employee row with EmployeeID = 1000 that were active at least for a portion of period between 1st January of 2014 and 1st January 2015 (including the upper boundary):

```
SELECT * FROM Employee FOR SYSTEM_TIME  
BETWEEN '2014-01-01 00:00:00.0000000' AND '2015-01-01 00:00:00.0000000'  
WHERE EmployeeID = 1000 ORDER BY ValidFrom;
```

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

NEW QUESTION 25

You create a table to store sales information for an online sales application by running the following Transact-SQL statement:

```
CREATE TABLE Sales (
    SalesOrderID INT NOT NULL,
    OrderDate DATETIME NOT NULL,
    Total MONEY NULL
)
```

You have a historical report that summarizes the sales for each quarter and year. The query that generated the data for the report is no longer available. A representative report contains the following data:

Total	Quarter	Year	G1	G2
\$8,771,886.36	1	2013	0	0
\$14,373,277.48	1	2014	0	0
\$23,145,163.83	1	NULL	0	1
\$12,225,061.38	2	2013	0	0
\$8,046,220.84	2	2014	0	0
\$20,271,282.22	2	NULL	0	1
\$14,339,319.19	3	2013	0	0
\$14,339,319.19	3	NULL	0	1
\$13,629,621.04	4	2013	0	0
\$13,629,621.04	4	NULL	0	1
\$71,385,386.28	NULL	NULL	1	1

You need to recreate the query for the report.

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.

SELECT

```
SUM(Total) AS Total,
DATEPART(quarter, OrderDate) AS Quarter,
YEAR(OrderDate) AS Year,
```

▼ AS G1,

DATEPART(quarter, OrderDate) % 1
MAX (DATEPART(quarter, OrderDate))
GROUPING (DATEPART(quarter, OrderDate))
CASE WHEN DATEPART(quarter, OrderDate)=NULL THEN 1 ELSE 0 END

▼ AS G2

GROUPING (YEAR (OrderDate))
DATEPART (YEAR, OrderDate) % 1
MAX (DATEPART (YEAR, OrderDate))
CASE WHEN DATEPART (YEAR, OrderDate)=NULL THEN 1 ELSE 0 END

FROM Sales

GROUP BY

▼

DATEPART(quarter, OrderDate)
DATEPART(quarter, OrderDate) WITH ROLLUP
DATEPART(quarter, OrderDate), YEAR(OrderDate)
DATEPART(quarter, OrderDate), YEAR(OrderDate) WITH ROLLUP

Answer:

Explanation:

```
SELECT
    SUM(Total) AS Total,
    DATEPART(quarter, OrderDate) AS Quarter,
    YEAR(OrderDate) AS Year,
    DATEPART(quarter, OrderDate) % 1
    MAX(DATEPART(quarter, OrderDate))
    GROUPING(DATEPART(quarter, OrderDate))
    CASE WHEN DATEPART(quarter, OrderDate) = NULL THEN 1 ELSE 0 END
AS G1,
    GROUPING(YEAR(OrderDate))
    DATEPART(YEAR, OrderDate) % 1
    MAX(DATEPART(YEAR, OrderDate))
    CASE WHEN DATEPART(YEAR, OrderDate) = NULL THEN 1 ELSE 0 END
AS G2
FROM Sales
GROUP BY
    DATEPART(quarter, OrderDate)
    DATEPART(quarter, OrderDate) WITH ROLLUP
    DATEPART(quarter, OrderDate), YEAR(OrderDate)
    DATEPART(quarter, OrderDate), YEAR(OrderDate) WITH ROLLUP
```

NEW QUESTION 26

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 29

You need to create a database object that meets the following requirements:

- accepts a product identifies as input
- calculates the total quantity of a specific product, including quantity on hand and quantity on order
- caches and reuses execution plan

returns a value
can be called from within a SELECT statement
can be used in a JOIN clause
What should you create?

- A. an extended stored procedure
- B. a user-defined table-valued function
- C. a user-defined stored procedure that has an OUTPUT parameter
- D. a memory-optimized table that has updated statistics

Answer: B

NEW QUESTION 32

You have the following Transact-SQL query:

```
SELECT
    City.CityID,
    City.CityName,
    TranslateName(Nearby.CityName) AS NearbyCity
FROM Cities AS City
CROSS APPLY NearbyCities(City.CityID) AS Nearby
```

What type of functions are used in the query? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

Answer Area

Function	Type
TranslateName	<div><div></div><div>▼</div><div>Scalar</div><div>Table-Valued</div><div>System</div><div>Aggregate</div></div>
NearbyCities	<div><div></div><div>▼</div><div>Scalar</div><div>Table-Valued</div><div>System</div><div>Aggregate</div></div>

Answer:

Explanation: Box 1: Scalar

The return value of a function can either be a scalar (single) value or a table. Box 2: Table-Valued

The APPLY operator allows you to invoke a table-valued function for each row returned by an outer table expression of a query. The table-valued function acts as the right input and the outer table expression acts as the left input. The right input is evaluated for each row from the left input and the rows produced are combined for the final output. The list of columns produced by the APPLY operator is the set of columns in the left input followed by the list of columns returned by the right input.

References:

<https://msdn.microsoft.com/en-us/library/ms186755.aspx> [https://technet.microsoft.com/en-us/library/ms175156\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms175156(v=sql.105).aspx)

NEW QUESTION 37

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

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

The table includes the data shown below:

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

You are developing a report that displays the following values and column headers in the order listed below:

- average price of a product named Average
- the smallest number of products in stock named LowestNumber
- the highest product price named HighestPrice

You need to write a query to return the results for the report. The query must meet the following requirements: You need to write a query to return the results for the report. The query must meet the following requirements:

- Use built-in, aggregate mathematical functions.
- Use two-part names and tables.
- Use the table alias to qualify column names.
- Define the alias for all fields by using the AS keyword.
- Use the first letter of the table name as the table alias.
- Do not use the Row_number function.
- Do not surround object names with square brackets.
- Do not use variables.

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

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.

Check Syntax

CHECK

FREETEXT

OPEN

CHECKPOINT

FREETEXTTABLE

OPENDATASOURCE

CLOSE

FROM

OPENQUERY

CLUSTERED

FULL

OPENROWSET

COALESCE

FUNCTION

OPENXML

COLLATE

GOTO

OPTION

COLUMN

GRANT

OR

COMMIT

GROUP

ORDER

COMPUTE

HAVING

OUTER

CONCAT

HOLDLOCK

OVER

CONSTRAINT

IDENTITY

PERCENT

CONTAINS

IDENTITY_INSERT

PIVOT

CONTAINSTABLE

IDENTITYCOL

PLAN

CONTINUE

IF

PRECISION

CONVERT

IN

PRIMARY

CREATE

INDEX

PRINT

CROSS

INNER

PROC

CURRENT

INSERT

PROCEDURE

CURRENT_DATE

INTERSECT

PUBLIC

CURRENT_TIME

INTO

RAISERROR

CURRENT_TIMESTAMP

IS

READ

CURRENT_USER

JOIN

READTEXT

CURSOR

KEY

SYSTEM_USER

DROP

RULE

TABLE

DUMP

SAVE

TABLESAMPLE

ELSE

SCHEMA

TEXTSIZE

END

SECURITYAUDIT

THEN

ERRLVL

SELECT

TO

ESCAPE

SEMANTICKEYPHRASETABLE

TOP

EXCEPT

SEMANTICSIMILARITYDETAILSTABLE

TRAN

EXEC

SEMANTICSIMILARITYTABLE

TRANSACTION

EXECUTE

SESSION_USER

TRIGGER

EXISTS

SET

TRUNCATE

EXIT

SETUSER

TRY_CONVERT

EXTERNAL

SHUTDOWN

TSEQUAL

FETCH

SOME

UNION

FILE

STATISTICS

UNIQUE

FILLFACTOR

SYSTEM_USER

UNPIVOT

FOR

TABLE

UPDATE

FOREIGN

TABLESAMPLE

UPDATETEXT

FREETEXT

TEXTSIZE

USE

FREETEXTTABLE

THEN

USER

FROM

TO

VALUES

FULL

TOP

VARYING

FUNCTION

TRAN

VIEW

GOTO

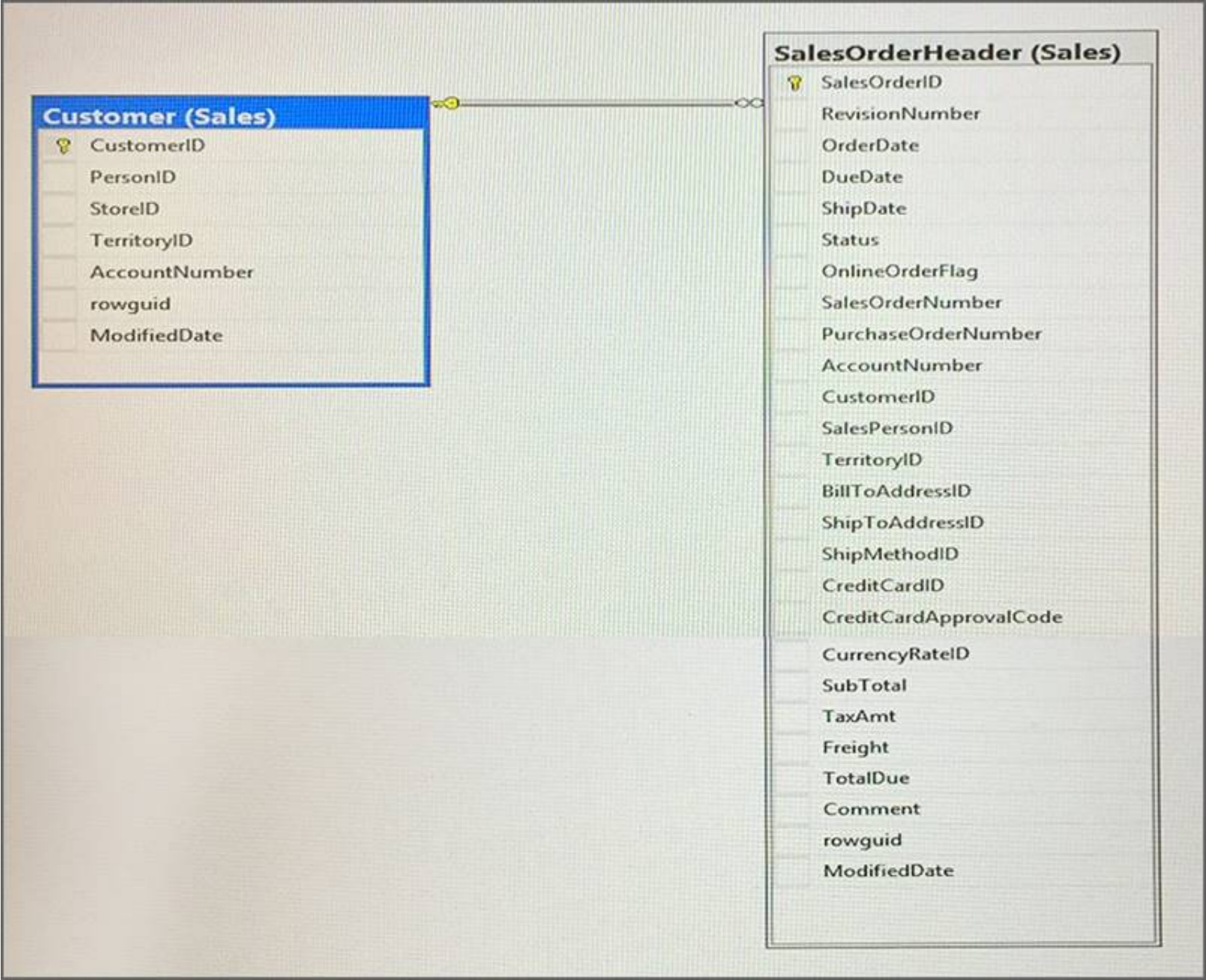
TRANSACTION

Answer:

Explanation: TRY_Convert

NEW QUESTION 41

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



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

A

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

B

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

C

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

D

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

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

Answer: A

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

NEW QUESTION 46

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 need to create a query that generates sample data for a sales table in the database. The query must include every product in the inventory for each customer.
Which statement clause should you use?

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

Answer: C

NEW QUESTION 50

You have a database that contains the following tables: Customer

Column name	Data type	Nullable	Default value
CustomerId	int	No	Identity property
FirstName	varchar(30)	Yes	
LastName	varchar(30)	No	
CreditLimit	money	No	

CustomerAudit

Column name	Data type	Nullable	Default value
CustomerId	int	No	
DateChanged	datetime	No	GETDATE()
OldCreditLimit	money	No	
NewCreditLimit	money	No	
ChangedBy	varchar(100)	No	SYSTEM USER

Where the value of the CustomerID column equals 3, you need to update the value of the CreditLimit column to 1000 for the customer. You must ensure that the change to the record in the Customer table is recorded on the CustomerAudit table.
Which Transact-SQL statement should you run?

- A.
UPDATE Customer
SET CreditLimit= 1000
OUTPUT inserted. CustomerId, deleted. CreditLimit, deleted. CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit, ChangedBy)
WHERE CustomerId=3
- B.
UPDATE Customer
SET CreditLimit= 1000
OUTPUT inserted. CustomerId, GETDATE (), deleted. CreditLimit, inserted. CreditLimit, SYSTEM_USER
INTO CustomerAudit (CustomerId, DateChanged, OldCreditLimit, NewCreditLimit, ChangedBy)
WHERE CustomerId=3
- C.
UPDATE Customer
SET CreditLimit= 1000
WHERE CustomerId=3
INSERT INTO CustomerAudit (CustomerId, DateChanged, OldCreditLimit, NewCreditLimit,
ChangedBy)
SELECT CustomerId, GETDATE (), CreditLimit, CreditLimit, SYSTEM_USER
FROM Customer
WHERE CustomerID =3
- D.
UPDATE Customer
SET CreditLimit= 1000
OUTPUT inserted. CustomerId, inserted. CreditLimit, inserted. CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
WHERE CustomerId=3

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

Answer: C

NEW QUESTION 53

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

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

You 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)
```

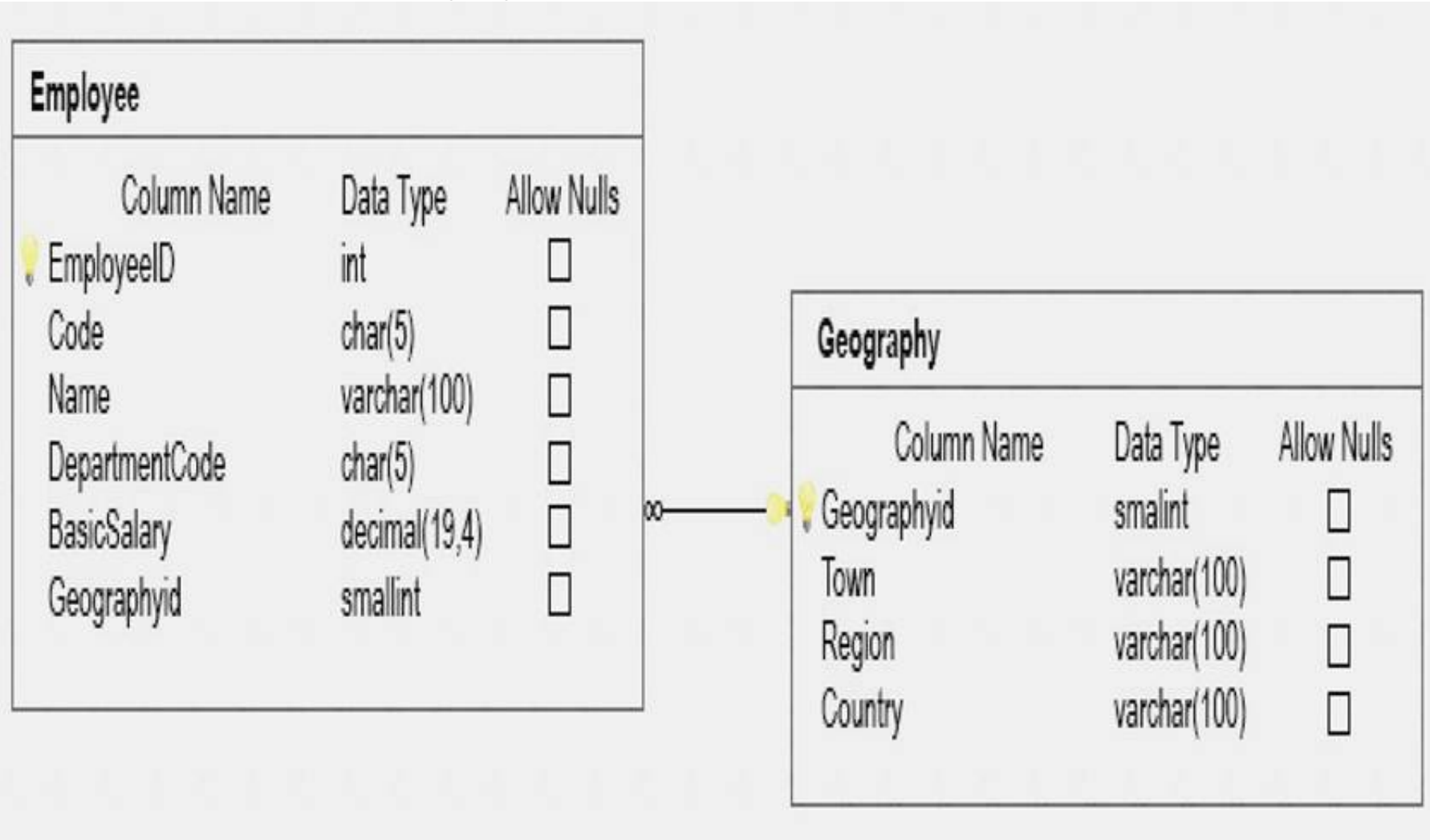
Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 57

You have two tables as shown in the following image:



You need to analyze the following query. (Line numbers are included for reference only.)

```
01 DECLARE @DepartmentCode nchar(5) = N'DEP01'
02 DECLARE @RoundedUpSalary int
03 DECLARE @EmployeeName nvarchar(100)
04 SELECT
05     Name,
06     CONVERT(int, Code) EmployeeCode,
07     BasicSalary
08 FROM dbo.Employee e
09 INNER JOIN dbo.Geography g
10 ON e.GeographyId = g.GeographyId
11 WHERE DepartmentCode = @DepartmentCode
```

Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic.
NOTE: Each correct selection is worth one point.

Answer Area

Statements

An implicit conversion exists at [answer choice].

Answer choices

	▼
line number 6	
line number 10	
line number 11	

An explicit conversion exists at [answer choice].

	▼
line number 6	
line number 10	
line number 11	

Answer:

Explanation: To compare char(5) and nchar(5) an implicit conversion has to take place. Explicit conversions use the CAST or CONVERT functions, as in line number 6.

References:

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-type-conversion-database-engine#implicit-and-explici>

NEW QUESTION 60

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 aggregates customer data only for the year 2014. The report requires that the data be denormalized.

You need to return the data for the report.

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 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 H

Answer: G**NEW QUESTION 65**

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 68

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 ');
```
- B**
- ```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME AS OF '2017-01-01';
```

C

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME ALL;
```

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 71

Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question presents a different goal and answer choices, but the text of the scenario is exactly the same in each question in this series.

You are developing a database to track customer orders. The database contains the following tables: Sales.Customers, Sales.Orders, and Sales.OrderLines. The following table describes the columns in Sales.Customers.

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

The following table describes the columns in Sales.Orders.

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

The following table describes the columns in Sales.OrderLines.

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

You need to create a database object that calculates the total price of an order including the sales tax. The database object must meet the following requirements:

- Reduce the compilation cost of Transact-SQL code by caching the plans and reusing them for repeated execution.
- Return a value.
- Be callable from a SELECT statement.

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

Answer Area

CREATE ▼ Sales.CalculateOrderPrice

PROCEDURE
VIEW
FUNCTION

(
@orderID int
)

▼
WITH EXECUTE AS OWNER
RETURNS decimal(18,2)
RETURNS TABLE

AS

▼
BEGIN TRAN
BEGIN
RETURN

DECLARE @OrderPrice decimal(18,2)
 DECLARE @CalculatedTaxRate decimal(18,2)
 SET @OrderPrice = (SELECT SUM(Quantity * UnitPrice) FROM Sales.OrderLines WHERE OrderID = @OrderID)
 SET @CalculatedTaxRate = (SELECT 1 + (MAX(TaxRate) / 100) FROM Sales.OrderLines WHERE OrderID = @OrderID)

RETURN (▼)

@OrderPrice * @CalculatedTaxRate
 SELECT (#OrderPrice * #CalculatedTaxRate) AS CalculatedOrderPrice
 CalculateOrderPrice

▼
RETURN
COMMIT
END

Answer:

Explanation: Box 1: FUNCTION

To be able to return a value we should use a scalar function.

CREATE FUNCTION creates a user-defined function in SQL Server and Azure SQL Database. The return value can either be a scalar (single) value or a table.

Box 2: RETURNS decimal(18,2)

Use the same data format as used in the UnitPrice column. Box 3: BEGIN

Transact-SQL Scalar Function Syntax include the BEGIN ..END construct.

CREATE [OR ALTER] FUNCTION [schema_name.] function_name

([{ @parameter_name [AS][type_schema_name.] parameter_data_type [= default] [READONLY] }

[,...n]

]

)

RETURNS return_data_type

[WITH <function_option> [,...n]] [AS]

BEGIN

function_body

RETURN scalar_expression END

[;]

Box 4: @OrderPrice * @CalculatedTaxRate Calculate the price including tax.

Box 5: END

Transact-SQL Scalar Function Syntax include the BEGIN ..END construct. References: <https://msdn.microsoft.com/en-us/library/ms186755.aspx>

NEW QUESTION 74

You run the following Transact-SQL statement:


```
CREATE TABLE Sales.Customers(  
    custid int IDENTITY(1,1) NOT NULL,  
    companyname nvarchar(50) NULL,  
    contacttitle nvarchar(30) NOT NULL,  
    address nvarchar(60) NOT NULL,  
    postalcode nvarchar(10) NOT NULL,  
    region nvarchar(15) NULL,  
    phone nvarchar(24) NOT NULL,  
    fax nvarchar(24) NULL,  
    ) ON PPRIMARY
```

You need to ensure that you can insert data into the table.
What are the characteristics of the data? To answer, select the appropriate options in the answer area.

Column input constraint

Values cannot be entered into this column

A value must be inserted into this column

Data entry into this column is optional

Column name

▼

custid

fax

postalcode

region

▼

custid

fax

postalcode

region

▼

custid

fax

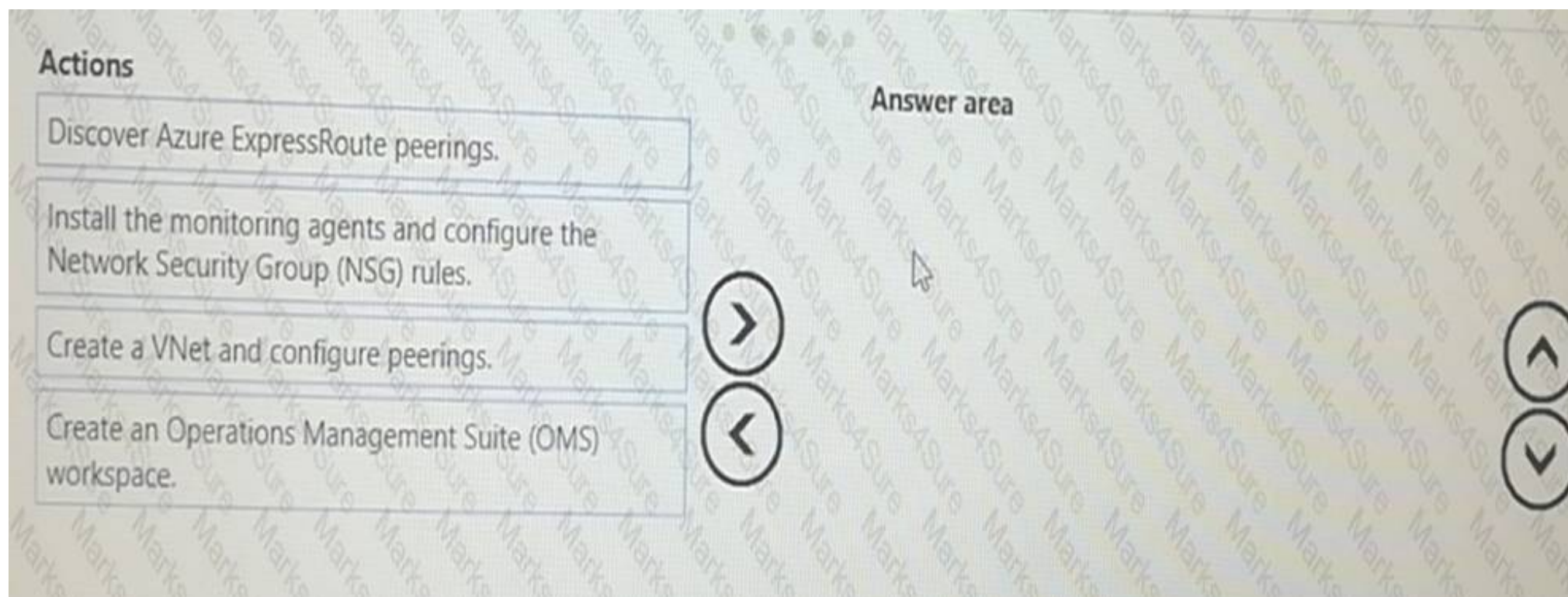
postalcode

region

Answer:

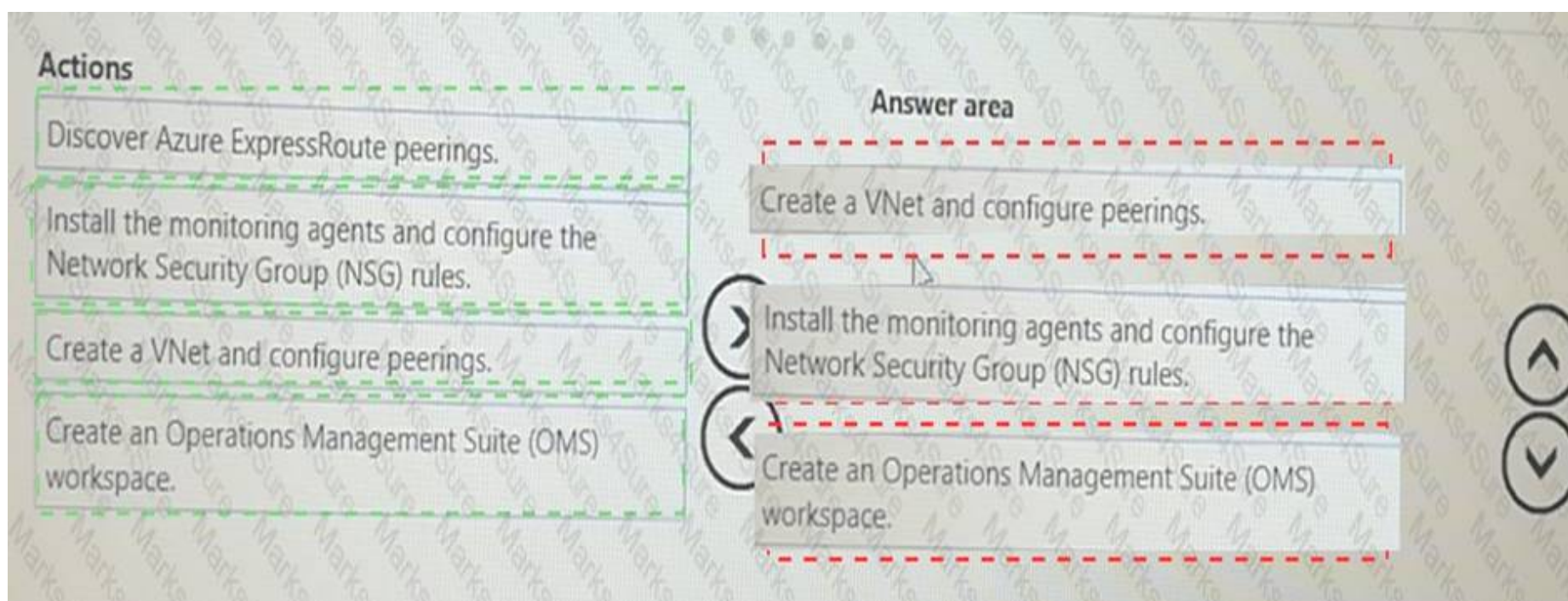
Explanation: Box 1: custid
IDENTITY indicates that the new column is an identity column. When a new row is added to the table, the Database Engine provides a unique, incremental value for the column. Identity columns are typically used with PRIMARY KEY constraints to serve as the unique row identifier for the table.
Box 2: postalcode
postalcode is declared as NOT NULL, which means that a value must be inserted. Box 3: region
Fax is also a correct answer. Both these two columns are declared as NULL, which means that data entry is optional.
References: <https://msdn.microsoft.com/en-us/library/ms174979.aspx>

NEW QUESTION 77
Your company plans to use Network Performance Monitor (NPM) on an existing Azure ExpressRoute connection.
You need to configure NPM.
Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



Answer:

Explanation:



NEW QUESTION 80

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 on this series.

You have a database that tracks orders and deliveries for customers in North America. System versioning is enabled for all tables. The database contains the Sales.Customers, Application.Cities, and Sales.CustomerCategories tables.

Details for the Sales.Customers table are shown in the following table:

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 values
StandardDiscountPercentage	int	does not allow values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow values
DeliveryLocation	geography	does not allow values
PhoneNumber	nvarchar(20)	does not allow values
ValidFrom	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW START
ValidTo	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW END

Details for the Application.Cities table are shown in the following table:

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

Details for the Sales.CustomerCategories table are shown in the following table:

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

The marketing department is performing an analysis of how discount affect credit limits. They need to know the average credit limit per standard discount percentage for customers whose standard discount percentage is between zero and four.

You need to create a query that returns the data for the analysis.

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

0, 1, 2, 3, 4

(0...4)

BETWEEN 0 AND 4

PIVOT

GROUP BY

[CreditLimit]

AVG(CreditLimit)

Answer Area

SELECT

Transact-SQL segment

FROM (

SELECT

StandardDiscountPercentage,

Transact-SQL segment

FROM Sales.Customers

) AS SourceTable

Transact-SQL segment

(

AVG(CreditLimit)

FOR StandardDiscountPercentage IN (

Transact-SQL segment

) AS CreditLimitTable

Answer:

Explanation: Box 1: 0, 1, 2, 3, 4

Pivot example:

-- Pivot table with one row and five columns

SELECT 'AverageCost' AS Cost_Sorted_By_Production_Days, [0], [1], [2], [3], [4]

FROM

(SELECT DaysToManufacture, StandardCost FROM Production.Product) AS SourceTable PIVOT

(

AVG(StandardCost)

FOR DaysToManufacture IN ([0], [1], [2], [3], [4])

) AS PivotTable; Box 2: [CreditLimit]

Box 3: PIVOT

You can use the PIVOT and UNPIVOT relational operators to change a table-valued expression into another table. PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output, and performs aggregations where they are required on any remaining column values that are wanted in the final output.

Box 4: 0, 1, 2, 3, 4

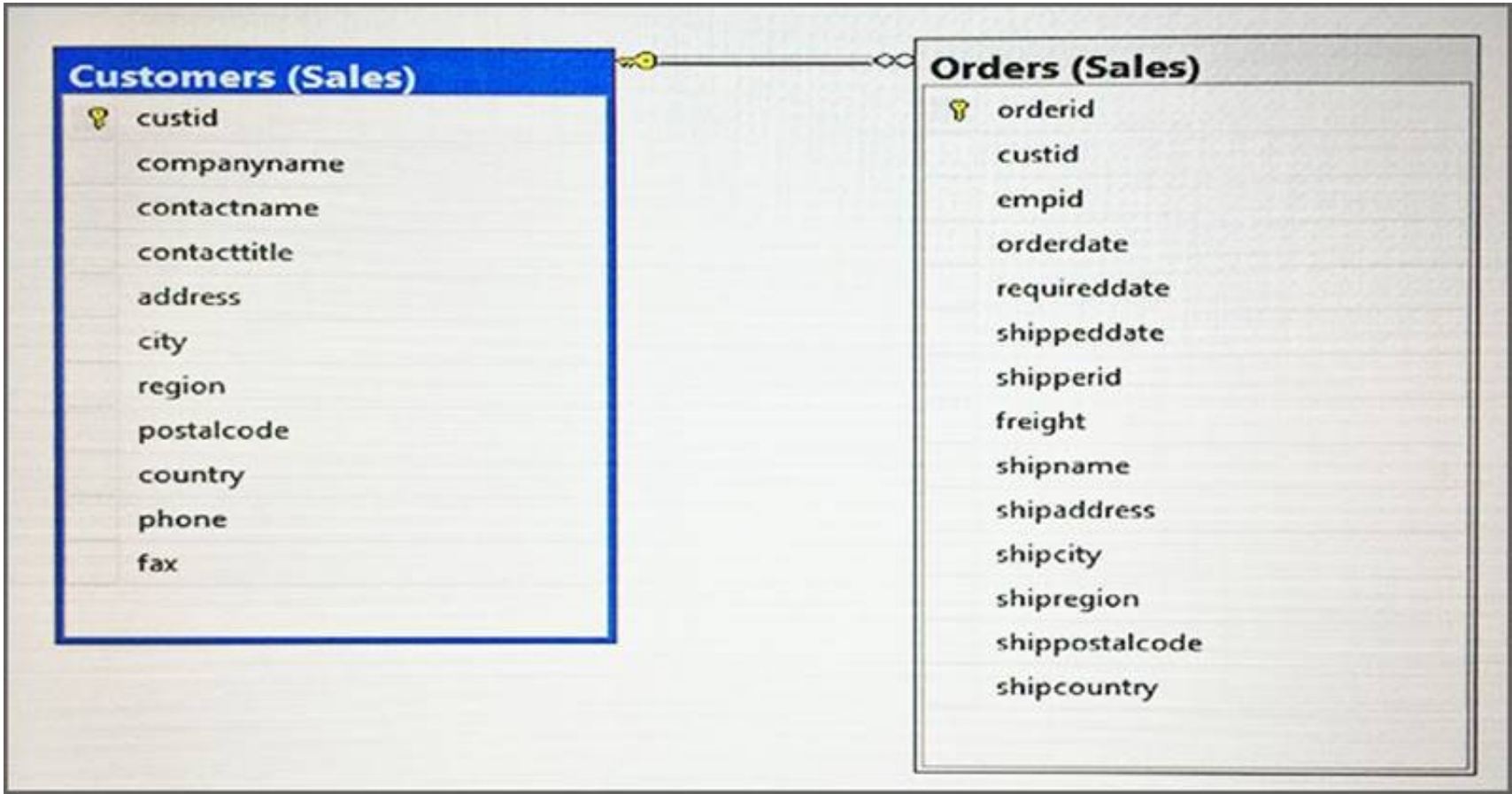
The IN clause determines whether a specified value matches any value in a subquery or a list. Syntax: test_expression [NOT] IN (subquery | expression [,...n])

Where expression[,... n]

is a list of expressions to test for a match. All expressions must be of the same type as test_expression. References: [https://technet.microsoft.com/en-us/library/ms177410\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx)

NEW QUESTION 81

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 LasOrderDate

ON c.custid = o.custid

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

Answer Area

⬅

➡

⬆

⬇

Answer:

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 82

You are developing a database to track employee progress relative to training goals. You run the following Transact-SQL statements:

```
CREATE TABLE Employees(  
    EmployeeID INT IDENTITY(1,1) NOT NULL,  
    Name VARCHAR(150) NULL,  
    CONSTRAINT PK_Employees PRIMARY KEY CLUSTERED (  
        EmployeeID ASC  
    ) WITH (STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF) ON PRIMARY  
    ) ON PRIMARY  
  
CREATE TABLE CoursesTaken(  
    CourseID INT NOT NULL,  
    EmployeeID INT NOT NULL,  
    CourseTakenOn DATE NULL,  
    CONSTRAINT PK_CoursesTaken PRIMARY KEY CLUSTERED (  
        CourseID ASC, EmployeeID ASC  
    ) WITH (STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF) ON PRIMARY  
    ) ON PRIMARY
```

You must build a report that shows all Employees and the courses that they have taken. Employees that have not taken training courses must still appear in the report. The report must display NULL in the course column for these employees.

You need to create a query for the report.

A)

```
SELECT e.Name, c.Course  
FROM dbo.Courses c  
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID  
INNER JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
```

B)

```
SELECT e.Name, c.Course  
FROM dbo.Courses c  
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID  
JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
```

C)

```
SELECT e.Name, c.Course  
FROM dbo.Courses c  
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID  
LEFT JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
```

D)

```
SELECT e.Name, c.Course  
FROM dbo.Courses c  
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID  
RIGHT JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
```

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

Answer: A

NEW QUESTION 84

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 .01 and 99.99
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation: Products with a price between \$0.00 and \$100 will be increased, while products with a price of \$0.00 would not be increased.

NEW QUESTION 87

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 88

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


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

DELETE from sales.orders
where orderdate <'2012-01-01' and shippeddate is not null

Answer:

Explanation: DELETE FROM Sales.Orders WHERE OrderDate < '2012-01-01' AND ShippedDate NOT NULL <https://msdn.microsoft.com/en-us/library/ms189835.aspx>
<https://msdn.microsoft.com/en-us/library/bb630352.aspx>

NEW QUESTION 92

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

```
CREATE TABLE Sales.Categories (
  CategoryID smallint NOT NULL PRIMARY KEY,
  Name nvarchar(50) NOT NULL,
  ParentCategoryID int NULL
)
```

You add the following data to the table.

CategoryID	Name	ParentCategoryID
1	Electronics	NULL
2	Cameras and photography	1
3	Computers and tablets	1
4	Cell phones and accessories	1
5	TV and audio	1
6	Digital cameras	2
9	laptops	3
13	Household goods	NULL
14	Bathroom items	13
15	Shower curtains	14

You need to create a query that uses a common table expression (CTE) to show the parent category of each category. The query must meet the following requirements:

- Return all columns from the Categories table in the order shown.
 - Exclude all categories that do not have a parent category.
- Construct the query using the following guidelines:
- Name the expression ParentCategories.
 - Use PC as the alias for the expression.
 - Use C as the alias for the Categories table.
 - Use the AS keyword for all table aliases.
 - Use individual column names for each column that the query returns.
 - Do not use a prefix for any column name.
 - 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      c(SELECT c.categoryid,c.name,c.parentcategoryid
2          FROM sales.categories c
3          WHERE parentcategoryid is not null
4      )
5      SELECT * FROM parentcategories

```

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.

Answer:

Explanation: 1 WITH ParentCategories pc (CategoryID, Name, PatentCategoryID) AS (SELECT c.categoryID,c.name,c.parentcategoryid
2 FROM sales.categories c
3 WHERE parentcategoryid is not null
4)
5 SELECT * FROM parentcategories

Note: On Line 1 replace c with WITH ParentCategories pc (CategoryID, Name, PatentCategoryID) AS Note: The basic syntax structure for a CTE is:
WITH expression_name [(column_name [...n])] AS
(CTE_query_definition)

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 94

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		

Answer:

Explanation: References:

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

NEW QUESTION 96

You are developing a mobile app to manage meetups. The app allows for users to view the 25 closest people with similar interests. You have a table that contains records for approximately two million people. You create the table by running the following Transact-SQL statement:

```
CREATE TABLE Person (
    PersonID INT,
    Name NVARCHAR(155) NOT NULL,
    Location GEOGRAPHY,
    Interests NVARCHAR(MAX)
)
```

You create the following table valued function to generate lists of people:

```
CREATE FUNCTION dbo.nearby (@person AS INT)
RETURNS @Res TABLE (
    PersonId INT NOT NULL,
    Location GEOGRAPHY
)
AS
BEGIN
    . . .
END
```

You need to build a report that shows meetings with at least two people only. What should you use?

- A. OUTER APPLY
- B. CROSS APPLY
- C. PIVOT
- D. LEFT OUTER JOIN

Answer: B

Explanation: References: <https://www.sqlshack.com/the-difference-between-cross-apply-and-outer-apply-in-sql-server/>

NEW QUESTION 99

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 100

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.

Answer:

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 102

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 is denormalized. Users make frequent changes to data in a primary table.

You need to ensure that users cannot change the tables directly, and that changes made to the primary table also update any related tables.

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: B

Explanation: Using an Indexed View would allow you to keep your base data in properly normalized tables and maintain data-integrity while giving you the denormalized "view" of that data.

References:

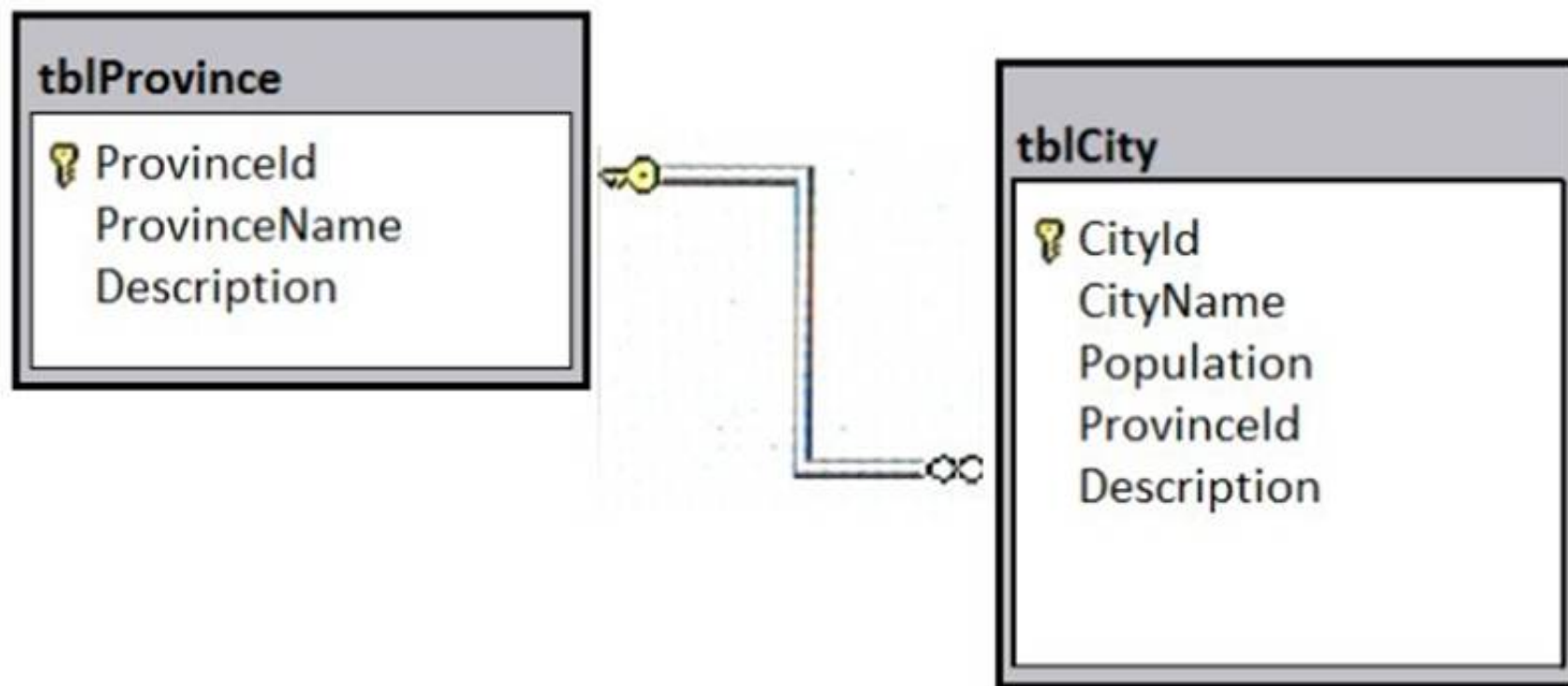
<http://stackoverflow.com/questions/4789091/updating-redundant-denormalized-data-automatically-in-sql-server>

NEW QUESTION 107

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.Provinceld`
- `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 109

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 111

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

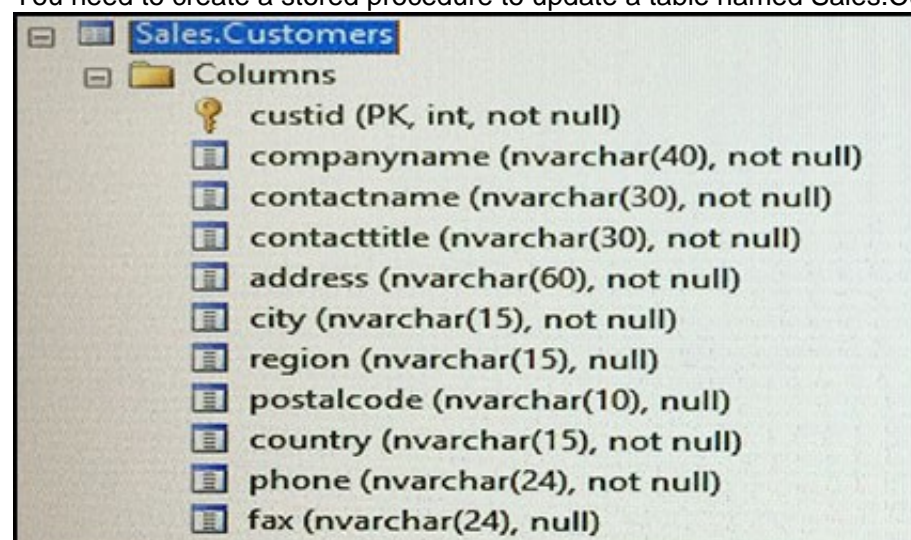
Answer: A

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 116

You need to create a stored procedure to update a table named Sales.Customers. The structure of the table is shown in the exhibit. (Click the exhibit button.)



Column	Data type	Notes
custid	int	primary key, not null
companyname	nvarchar(40)	not null
contactname	nvarchar(30)	not null
contacttitle	nvarchar(30)	not null
address	nvarchar(60)	not null
city	nvarchar(15)	not null
region	nvarchar(15)	null
postalcode	nvarchar(10)	null
country	nvarchar(15)	not null
phone	nvarchar(24)	not null
fax	nvarchar(24)	null

The stored procedure must meet the following requirements:

- Accept two input parameters.
- Update the company name if the customer exists.
- Return a custom error message if the customer does not exist.

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.

NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.

Transact-SQL segments	Answer Area
CREATE PROCEDURE Sales.ModCompanyName @custID int, @newname nvarchar(40) AS	
IF NOT EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)	
UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID	
BEGIN THROW 55555, 'The customer ID does not exist', 1 END	
UPDATE Sales.Customers SET companyname = @custID WHERE custid = @newname	
IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)	
ROLLBACK TRANSACTION	

Answer:

Explanation:

Transact-SQL segments	Answer Area
CREATE PROCEDURE Sales.ModCompanyName @custID int, @newname nvarchar(40) AS	CREATE PROCEDURE Sales.ModCompanyName @custID int, @newname nvarchar(40) AS
IF NOT EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)	
UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID	IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
BEGIN THROW 55555, 'The customer ID does not exist', 1 END	UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID
UPDATE Sales.Customers SET companyname = @custID WHERE custid = @newname	
IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)	IF NOT EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
ROLLBACK TRANSACTION	BEGIN THROW 55555, 'The customer ID does not exist', 1 END

NEW QUESTION 117

You have a database that contains the following tables: tblEmployees and tblSalesSummary. Each record contains approximately one million records. You use Microsoft SQL Server Management Studio (SSMS) to run two queries. The Include Actual Execution Plan option is enabled.

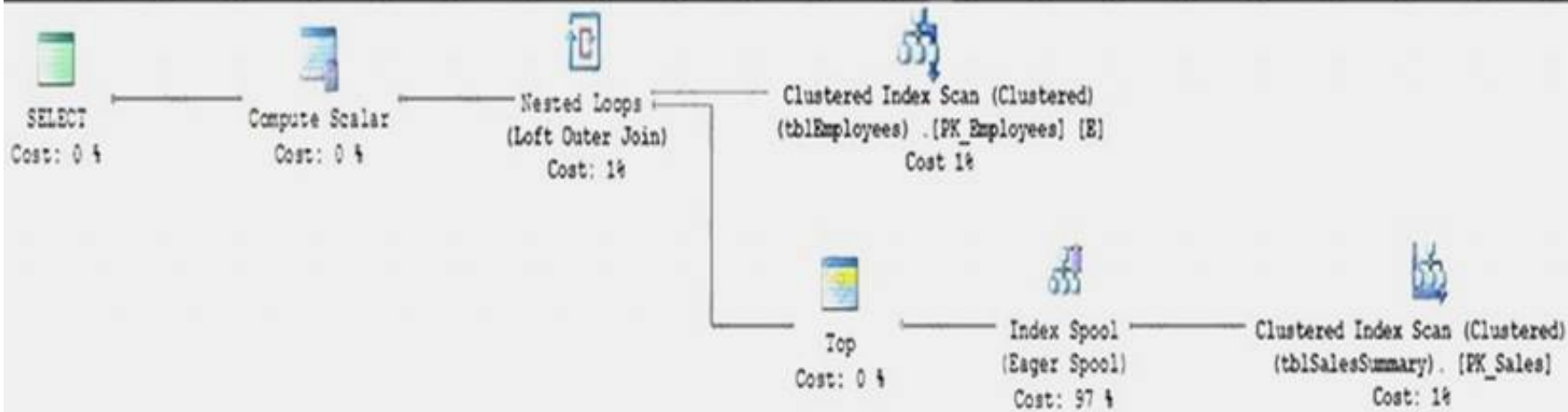
Both queries return the same results. SSMS generates the execution plans shown in the exhibit. (Click the Exhibit button.)

You need to troubleshoot the queries.

How should you interpret the execution plans? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

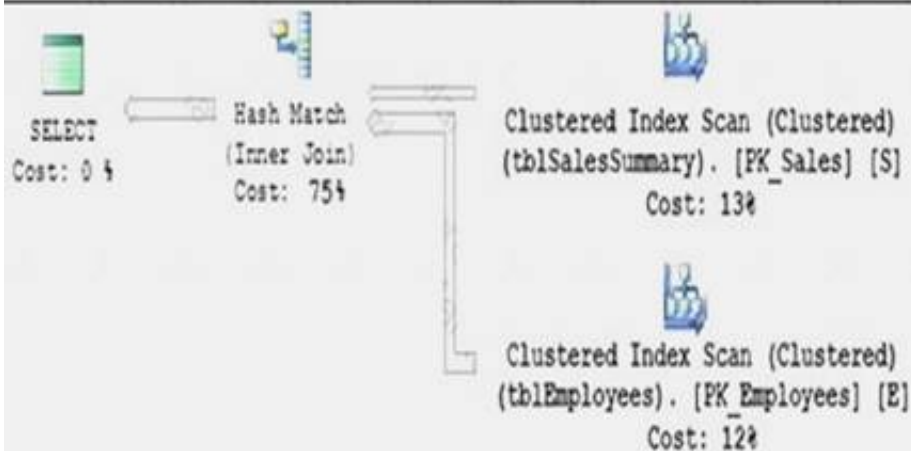
Query 1: Query cost (relative to the batch) : 93%

Select E.EmployeeNo, E.Name, SalesCount = (SELECT TOP 1 SalesCount FROM tblSalesSummary WHERE EmployeeNo = E.EmployeeNo) FROM tblEmployees



Query 2: Query cost (relative to the batch): 7%

Select E.EmployeeNo, E.Name, SalesCount = S.SalesCount FROM tblEmployees E INNER JOIN tblSalesSummary S ON S.EmployeeNo = E.EmployeeNo



Questions

Answer choices

Which of the two queries is more efficient?

- Query 1 is more efficient.
- Query 2 is more efficient.
- Both queries are equally efficient.

Why is the cost of the Index Spool operator highest in the first execution plan?

- There is no index in the tblSalesSummary table.
- The Index Scan on the tblSalesSummary table is executed many times.
- The Index Spool operator is executed many times.

Answer:

Explanation: References:

<https://docs.microsoft.com/en-us/sql/relational-databases/showplan-logical-and-physical-operators-reference?vie>

NEW QUESTION 120

You need to create a table named Sales that meets the following requirements:

Column name	Requirements
SalesID	<ul style="list-style-type: none"> - uniquely identify the row of data - automatically generate when data is inserted - use the least amount of storage space
SalesDate	<ul style="list-style-type: none"> - store the date and time of the sale based on 24-hour clock - use an ANSI SQL compliant data type
SalesAmount	<ul style="list-style-type: none"> - store the amount of the sale - avoid rounding errors when used in arithmetic calculations

Which Transact-SQL statement should you run?

A

```
CREATE TABLE Sales (  
    SalesID int IDENTITY(1,1) PRIMARY KEY,  
    SalesDate DateTime2 NOT NULL,  
    SalesAmount float NULL  
)
```

B

```
CREATE TABLE Sales (  
    SalesID int IDENTITY(1,1) PRIMARY KEY,  
    SalesDate DateTime2 NOT NULL,  
    SalesAmount decimal(18, 2) NULL  
)
```

C

```
CREATE TABLE Sales (  
    SalesID UNIQUEIDENTIFIER DEFAULT NEWSEQUENTIALID() PRIMARY KEY,  
    SalesDate DateTime2 NOT NULL,  
    SalesAmount decimal(18,2) NULL  
)
```

D

```
CREATE TABLE Sales (  
    SalesID int IDENTITY(1,1),  
    SalesDate DateTime NOT NULL,  
    SalesAmount decimal(18,2) NULL  
)
```

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

Answer: B

Explanation: References:
<https://docs.microsoft.com/en-us/sql/t-sql/data-types/decimal-and-numeric-transact-sql?view=sql-server-2017> <https://docs.microsoft.com/en-us/sql/t-sql/data-types/float-and-real-transact-sql?view=sql-server-2017>

NEW QUESTION 124

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.

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.

When running an operation, you updated a column named EndTime for several records in the Project table, but updates to the corresponding task records in the Task table failed.

You need to synchronize the value of the EndTime column in the Task table with the value of the EndTime column in the project table. The solution must meet the following requirements:

- * If the EndTime column has a value, make no changes to the record.
- * If the value of the EndTime column is null and the corresponding project record is marked as completed, update the record with the project finish time.

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

FROM Project AS P

WHERE P.EndTime IS NOT NULL AND T.EndTime is NULL

FROM Task AS T

WHERE P.EndTime IS NULL AND T.EndTime IS NOT NULL

UPDATE T SET T.EndTime = P.EndTime

INNER JOIN Project AS P ON T.ProjectId = P.ProjectId

INNER JOIN Task AS T ON T.UserId = P.UserId

UPDATE P SET P.EndTime = T.EndTime

Answer Area

Answer:

Explanation: Box 1: UPDATE T SET T.EndTime = P.EndTime
 We are updating the EndTime column in the Task table. Box 2: FROM Task AS T
 Where are updating the task table.
 Box 3: INNER JOIN Project AS P on T.ProjectID = P.ProjectID
 We join with the Project table (on the ProjectID columnID column). Box 4: WHERE P.EndTime is NOT NULL AND T.EndTime is NULL
 We select the columns in the Task Table where the EndTime column in the Project table has a value (NOT NULL), but where it is NULL in the Task Table.
 References: <https://msdn.microsoft.com/en-us/library/ms177523.aspx>

NEW QUESTION 125

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 named Customers. Data stored in the table must be exchanged between web pages and web servers by using AJAX calls that use REST endpoint.

You need to return all customer information by using a data exchange format that is text-based and lightweight.

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: C

Explanation: JSON can be used to pass AJAX updates between the client and the server.

Export data from SQL Server as JSON, or format query results as JSON, by adding the FOR JSON clause to a SELECT statement.

When you use the FOR JSON clause, you can specify the structure of the output explicitly, or let the structure of the SELECT statement determine the output.

References: <https://msdn.microsoft.com/en-us/library/dn921882.aspx>

NEW QUESTION 129

You have a database that contains the following tables: tblRoles, tblUsers, and tblUsersInRoles. The table tblRoles is defined as follows.

Column name	Data type	Nullable	Primary key
RoleID	int	No	Yes
RoleName	varchar(20)	No	No

You have a function named ufnGetRoleActiveUsers that was created by running the following Transact-SQL statement:

```
CREATE FUNCTION ufnGetRoleActiveUsers(@RoleId AS int)
RETURNS @roleSummary TABLE(UserName varchar (20))
AS
BEGIN
    INSERT INTO @roleSummary
    SELECT U.UserName FROM tblUsersInRoles BRG
    INNER JOIN tblUsers U
    ON U.UserId = BRG.UserId
    WHERE BRG.RoleId = @RoleId AND U.IsActive = 1
    RETURN
END
```

You need to list all roles and their corresponding active users. The query must return the RoleId, RoleName, and UserName columns. If a role has no active users, a NULL value should be returned as the UserName for that role. How should you complete the Transact-SQL statement? To answer, select the appropriate Transact-SQL segments in the answer area.

Answer area

SELECT *

FROM

tblRoles

tblUsersInRoles

tblUsers

CROSS JOIN

OUTER APPLY

CROSS APPLY

 ufnGetRoleActiveUsers(

RoleId

UserId

RoleName

)

Answer:

Explanation: Answer area

SELECT *

FROM

tblRoles

tblUsersInRoles

tblUsers

CROSS JOIN

OUTER APPLY

CROSS APPLY

 ufnGetRoleActiveUsers(

RoleId

UserId

RoleName

)

NEW QUESTION 130

You need to create a database object that meets the following requirements:

accepts a product identifies as input
calculates the total quantity of a specific product, including quantity on hand and quantity on order
caches and reuses execution plan
returns a value
can be called from within a SELECT statement
can be used in a JOIN clause
What should you create?

- A. a temporary table that has a columnstore index
- B. a user-defined table-valued function
- C. a memory-optimized table that has updated statistics
- D. a natively-compiled stored procedure that has an OUTPUT parameter

Answer: B

Explanation: A table-valued user-defined function can also replace stored procedures that return a single result set. The table returned by a user-defined function can be referenced in the FROM clause of a Transact-SQL statement, but stored procedures that return result sets cannot.
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 132

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:

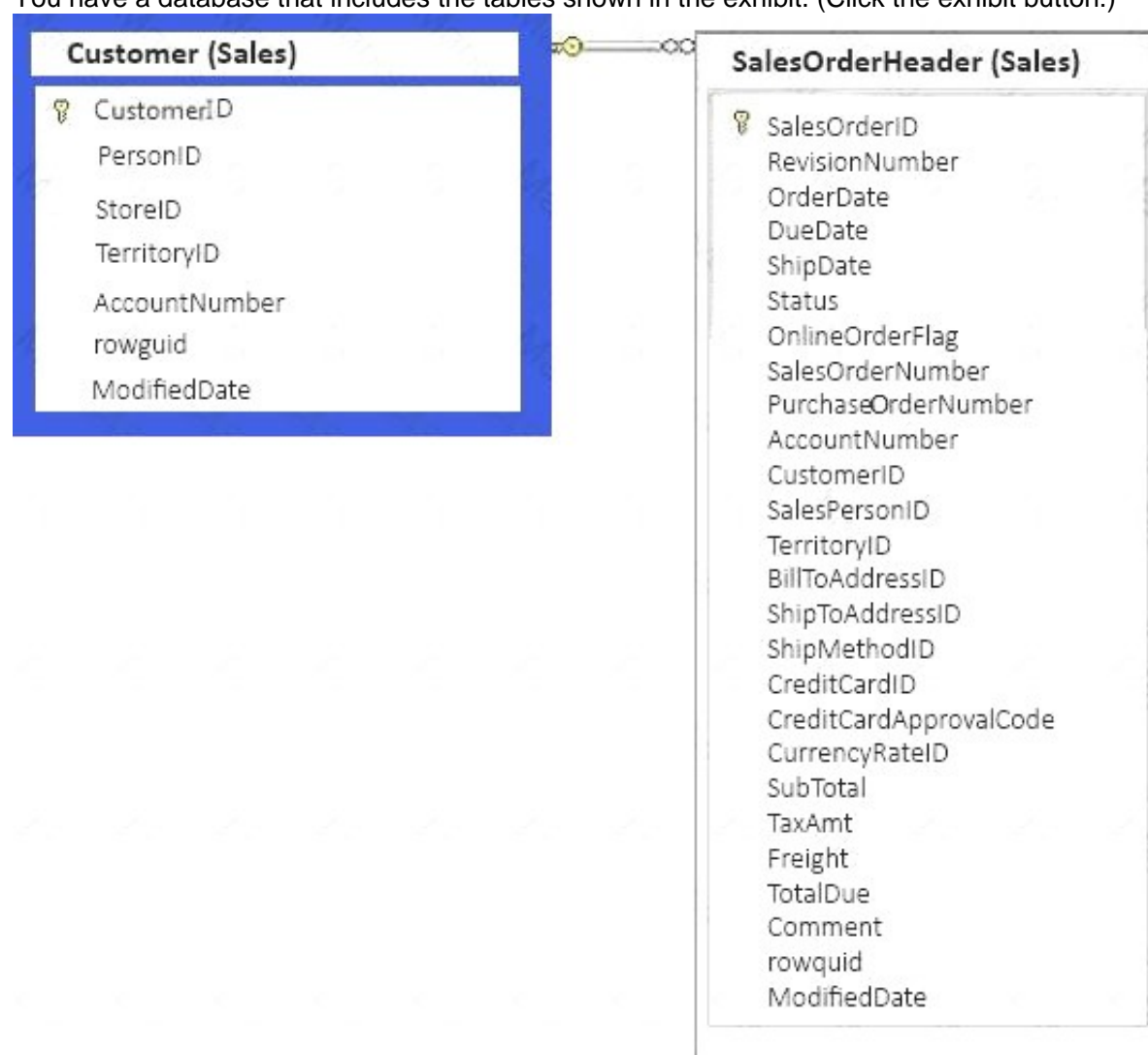
In the following example, the CUBE operator returns a result set that has one grouping for all possible combinations of columns in the CUBE list and a grand total grouping.

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 136

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



You need to create a list of all customers and the date that the customer placed their last order. For customers who have not placed orders, you must substitute 01/01/1990 for the date.

Which Transact-SQL statement should you run?

A

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

B

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

C

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

D

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

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

Answer: A

NEW QUESTION 141

You run the following Transact-SQL statement:

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

You need to create a stored procedure that meets the following requirements:

Inserts data into the Employees table.

Processes all data changes as a single unit of work.

Sets the exception severity level to 16 and an error number of 60, 000 when any error occurs.

If a Transact-SQL statement raises a runtime error, terminates and reverts the entire unit of work, and indicates the line number in the statement where the error occurred.

Inserts the value New Employee for the Title column if no title is provided.

How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segment to the correct target. Each Transact-SQL segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

Transact-SQL segments

RAISERROR (60000, 16, 1)

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

IF XACT_STATE () <> 0 ROLLBACK TRANSACTION

IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION

SAVE TRANSACTION AddEmployee

COMMIT TRANSACTION

Answer Area

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

    BEGIN CATCH
        Transact-SQL segment
        Transact-SQL segment
    END CATCH
```

Answer:

Explanation:

Transact-SQL segments

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

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

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

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

```
SAVE TRANSACTION AddEmployee
```

```
COMMIT TRANSACTION
```

Answer Area

```
CREATE PROCEDURE ADDEmployee
```

```
    @FirstName nvarchar(30),
```

```
    @LastName nvarchar(40),
```

```
    @Title nvarchar(50) = 'New Employee',
```

```
    @DepartmentID smallint,
```

```
    @ManagerID int
```

```
AS
```

```
BEGIN
```

```
    BEGIN TRY
```

```
        BEGIN TRANSACTION
```

```
        INSERT INTO Employees (FirstName, LastName, Title, DepartmentID, ManagerID
```

```
        VALUES (@FirstName, @LastName, @Title, @DepartmentID, @ManagerID
```

```
        COMMIT TRANSACTION
```

```
    END TRY
```

```
    BEGIN CATCH
```

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

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

```
    END CATCH
```

NEW QUESTION 142

You run the following Transact-SQL statements:

```
CREATE TABLE CourseParticipants
(
    CourseID INT NOT NULL,
    CourseDate DATE NOT NULL,
    LocationDescription VARCHAR(100) NOT NULL,
    NumParticipants INT NOT NULL
)
```

You need to create a query that returns the total number of attendees for each combination of CourseID, CourseDate, and the following locations: Lisbon, London, and Seattle. The result set should resemble the following:

	CourseID	CourseDate	Lisbon	London	Seattle
1	1	2018-02-01	NULL	NULL	15
2	2	2018-02-01	33	NULL	NULL
3	1	2018-02-02	NULL	20	NULL
4	1	2018-02-03	20	10	NULL
5	2	2018-02-03	NULL	20	NULL

Which Transact-SQL code segment should you run?

- A. SELECT *FROM CourseParticipants PIVOT(SUM(NumParticipants) FOR LocationDescription IN (Lisbon, London, Seattle))
- B. SELECT *FROM CourseParticipants PIVOT(SUM(NumParticipants) FOR LocationDescription IN (Lisbon, London, Seattle)) as PVTTable
- C. SELECT *FROM CourseParticipants UNPIVOT(SUM(NumParticipants) FOR LocationDescription IN (Lisbon, London, Seattle))
- D. SELECT *FROM CourseParticipants UNPIVOT(SUM(NumParticipants) FOR LocationDescription IN (Lisbon, London, Seattle) AS PVTTable

Answer: B

Explanation: References: https://www.techonthenet.com/sql_server/pivot.php

NEW QUESTION 143

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 144

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 145

You are building a stored procedure that will update data in a table named Table1 by using a complex query as the data source.

You need to ensure that the SELECT statement in the stored procedure meets the following requirements:

- Data being processed must be usable in several statements in the stored procedure.
- Data being processed must contain statistics. What should you do?

- A. Update Table1 by using a common table expression (CTE).
- B. Insert the data into a temporary table, and then update Table1 from the temporary table.
- C. Place the SELECT statement in a derived table, and then update Table1 by using a JOIN to the derived table.
- D. Insert the data into a table variable, and then update Table1 from the table variable.

Answer: B

Explanation: Temp Tables...

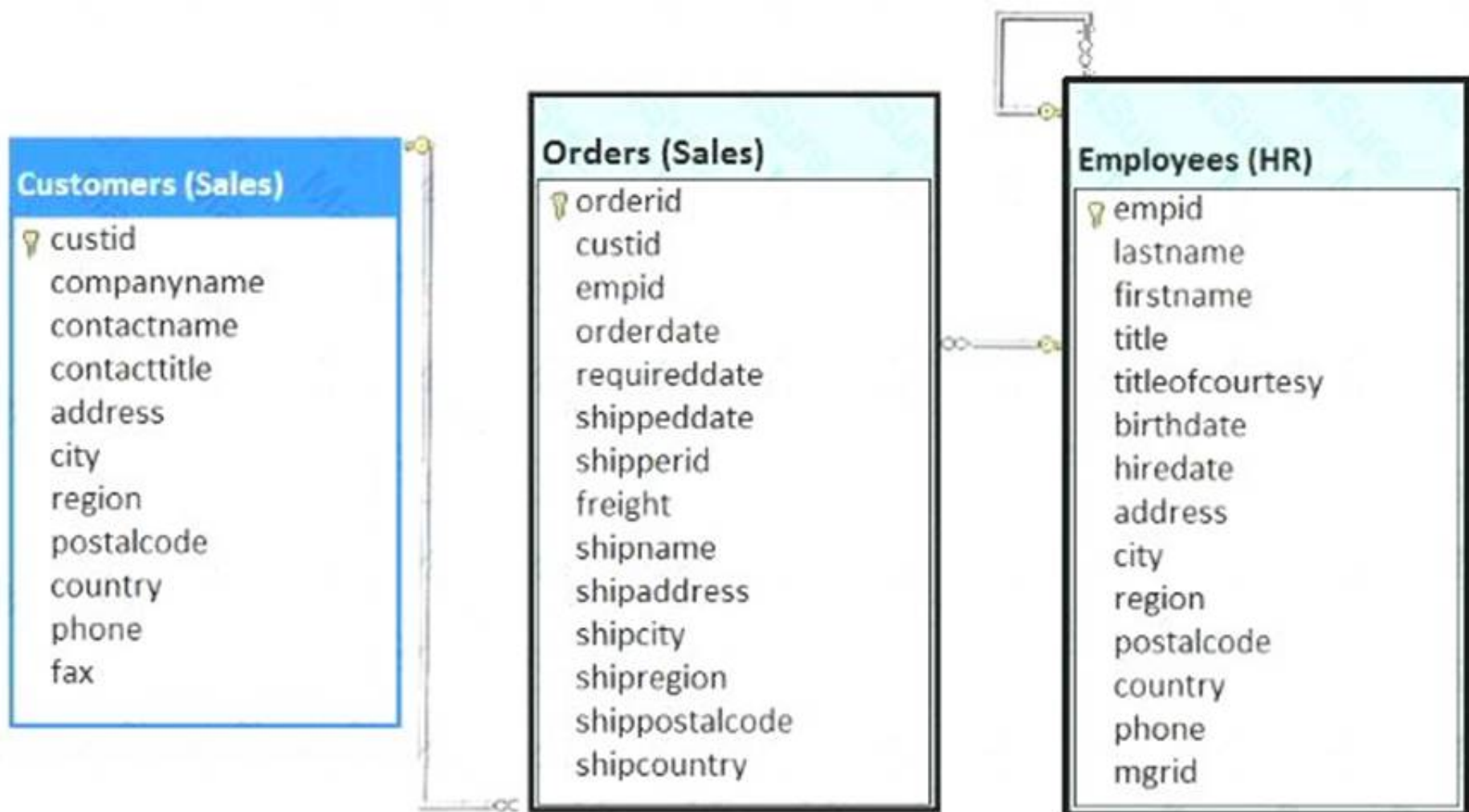
Are real materialized tables that exist in tempdb Have dedicated stats generated by the engine Can be indexed

Can have constraints

Persist for the life of the current CONNECTION Can be referenced by other queries or subproce

NEW QUESTION 147

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 152

You have a disk-based table that contains 15 columns.

You query the table for the number of new rows created during the current day.

You need to create an index for the query. The solution must generate the smallest possible index. Which type of index should you create?

- A. clustered
- B. filtered nonclustered with a getdate() predicate in the WHERE statement clause
- C. hash
- D. nonclustered with compression enabled

Answer: B

Explanation: A filtered index is an optimized nonclustered index especially suited to cover queries that select from a well-defined subset of data. It uses a filter predicate to index a portion of rows in the table. A well-designed filtered index can improve query performance as well as reduce index maintenance and storage costs compared with full-table indexes.

Creating a filtered index can reduce disk storage for nonclustered indexes when a full-table index is not necessary.

References: <https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-filtered-indexes>

NEW QUESTION 154

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

Answer:

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 158

A company's sales team is divided in two different regions, North and South. You create tables named SalesNorth and SalesSouth. The SalesNorth table stores sales data from the North region. The SalesSouth table stores sales data from the South region. Both tables use the following structure:

Column name	Data type	Allow nulls
region	CHAR(1)	Yes
salesID	INT	Yes
customer	VARCHAR(150)	Yes
amount	MONEY	Yes

You need to create a consolidated result set that includes all records from both tables. Which Transact-SQL statement should you run?

- A. SELECT SalesNorth.salesID, SalesNorth.customer, SalesNorth.amount, SalesSouth.SalesID, SalesSouth.customer, SalesSouth.amountFROM SalesNorthJOIN SalesSouth ON SalesNorth.salesID = SalesSouth.salesID
- B. SELECT SalesNorth.salesID, SalesNorth.customer, SalesNorth.amount, SalesSouth.salesID, SalesSouth.customer, SalesSouth.amountFROM SalesNorth LEFT JOIN SalesSouthON SalesNorth.salesID=SalesSouth.salesID
- C. SELECT salesID, customer, amount FROM SalesNorthUNION ALLSELECT salesID, customer, amount FROM SalesSouth
- D. SELECT salesID, customer, amount FROM SalesNorthUNIONSELECT salesID, customer, amountFROM SalesSouth

Answer: C

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

NEW QUESTION 162

You have the following stored procedure:


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

You run the following Transact-SQL statements:

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

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

Answer Area

Stored procedure execution	Result
First stored procedure execution	<div>▼</div> <div> All transactions are rolled back. Only the Log1 and Log3 tables are updated. Only the Log1 table is updated. All four tables are updated. </div>
Second stored procedure execution	<div>▼</div> <div> Only the Log1, Log2, and Log3 tables are updated. All transactions are rolled back. Only the Log1 table is updated. Only the Log1 and Log3 tables are updated. </div>

Answer:

Explanation: Box 1: All transactions are rolled back.

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

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

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

NEW QUESTION 167

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+ISNULL(UnitsOnOnrder,0)) AS  
TotalUnitPrice FROM Products
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: A

Explanation: ISNULL (check_expression , replacement_value) Arguments:

check_expression

Is the expression to be checked for NULL. check_expression can be of any type. replacement_value

Is the expression to be returned if check_expression is NULL. replacement_value must be of a type that is implicitly convertible to the type of check_expression.

References: <https://docs.microsoft.com/en-us/sql/t-sql/functions/isnull-transact-sql>

NEW QUESTION 172

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 173

You have a database named DB1 that contains two tables named Sales.Customers and Sales.CustomerTransaction. Sales.CustomerTransactions has a foreign key relationship to column named CustomerID in Sales.Customers.

You need to recommend a query that returns the number of customers who never completed a transaction. Which query should you recommend?

A

```
SELECT
    COUNT(Cust.CustomerID)
FROM
    Sales.Customers Cust
    LEFT JOIN
    Sales.CustomerTransactions Trans
    ON Cust.CustomerID = Trans.CustomerID
WHERE
    Trans.CustomerTransactionID IS NULL;
```

B

```
SELECT
    COUNT(CustomerID)
FROM
    Sales.Customers Cust
    LEFT JOIN
    Sales.CustomerTransactions Trans
    ON Cust.CustomerID = Trans.CustomerID
WHERE
    Trans.CustomerTransactionID IS NULL;
```

C

```
SELECT
    COUNT(Cust.CustomerID)
FROM
    Sales.Customers Cust
    LEFT JOIN
    Sales.CustomerTransactions Trans
    ON Cust.CustomerID = Trans.CustomerID
```

D

```
SELECT
    COUNT(Cust.CustomerID)
FROM
    Sales.Customers Cust
    INNER JOIN
    Sales.CustomerTransactions Trans
    ON Cust.CustomerID = Trans.CustomerID
WHERE
    Trans.CustomerTransactionID IS NULL;
```

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

Answer: A

NEW QUESTION 175

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.

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 plan to run the following query to update tasks that are not yet started:

```
UPDATE Task SET StartTime = GETDATE() WHERE StartTime IS NULL
```

You need to return the total count of tasks that are impacted by this UPDATE operation, but are not associated with a project. What set of Transact-SQL statements should you run?

- ☐ A
- ```
DECLARE @startedTasks TABLE(ProjectId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT deleted.ProjectId INTO @startedTasks WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE ProjectId IS NOT NULL
```
- ☐ B
- ```
DECLARE @startedTasks TABLE(TaskId int, ProjectId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT deleted.TaskId, deleted.ProjectId INTO @startedTasks
WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE ProjectId IS NULL
```
- ☐ C
- ```
DECLARE @startedTasks TABLE(TaskId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT inserted.TaskId, INTO @startedTasks WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE TaskId IS NOT NULL
```
- ☐ D
- ```
DECLARE @startedTasks TABLE(TaskId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT deleted.TaskId, INTO @startedTasks WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE TaskId IS NOT NULL
```

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

Answer: B

Explanation: The WHERE clause of the third line should be WHERE ProjectID IS NULL, as we want to count the tasks that are not associated with a project.

NEW QUESTION 176

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 develop a query that meets the following requirements:

Output data by using a tree-like structure.

Allow mixed content types.

Use custom metadata attributes.

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: F**NEW QUESTION 181**

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

Answer:

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

NEW QUESTION 184

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

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

You need to determine the total number of customers who have either deposit accounts or loan accounts, but not both types of accounts. Which Transact-SQL statement should you run?

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

Answer: G

Explanation: SQL Server provides the full outer join operator, FULL OUTER JOIN, which includes all rows from both tables, regardless of whether or not the other table has a matching value.

Consider a join of the Product table and the SalesOrderDetail table on their ProductID columns. The results show only the Products that have sales orders on them. The ISO FULL OUTER JOIN operator indicates that all rows from both tables are to be included in the results, regardless of whether there is matching data in the tables.

You can include a WHERE clause with a full outer join to return only the rows where there is no matching data between the tables. The following query returns only those products that have no matching sales orders, as well as those sales orders that are not matched to a product.

USE AdventureWorks2008R2; GO

-- The OUTER keyword following the FULL keyword is optional. SELECT p.Name, sod.SalesOrderID

FROM Production.Product p

FULL OUTER JOIN Sales.SalesOrderDetail sod ON p.ProductID = sod.ProductID

WHERE p.ProductID IS NULL OR sod.ProductID IS NULL ORDER BY p.Name ;

References: [https://technet.microsoft.com/en-us/library/ms187518\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms187518(v=sql.105).aspx)

NEW QUESTION 187

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.

All the sales data is stored in a table named table1. You have a table named table2 that contains city names. You need to create a query that lists only the cities that have no sales.

Which statement clause should you add to the query?

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

Answer: D

NEW QUESTION 192

Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.

You have a table named Products that contains information about the products that your company sells. The table contains many columns that do not always contain values.

You need to implement an ANSI standard method to convert the NULL values in the query output to the phrase "Not Applicable".

What should you implement?

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

Answer: F

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

NEW QUESTION 196

You have the following subqueries: Subquery1, Subquery2, and Subquery3.

You need to replace the three subqueries with named result sets or temporary tables. The following requirements must be met:

Subquery name	Requirements
Subquery1	The result set of this subquery must use the execution scope of a SELECT statement.
Subquery2	The result set of this subquery must be visible to other session users before disconnected.
Subquery3	The result set of this subquery must be accessible to other statements in the same session but must not be visible to other sessions.

Which replacement techniques should you use? To answer, select the appropriate options in the answer area.

Answer Area

Subquery name	Subquery replacement
Subquery1	<div><div>▼</div><div>common table expression (CTE) local temporary table global temporary table</div></div>
Subquery2	<div><div>▼</div><div>common table expression (CTE) local temporary table global temporary table</div></div>
Subquery3	<div><div>▼</div><div>common table expression (CTE) local temporary table global temporary table</div></div>

Answer:

Explanation: Subquery1: 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.

Subquery2: global temporary table
Global temporary tables are visible to any user and any connection after they are created, and are deleted when all users that are referencing the table disconnect from the instance of SQL Server.

Subquery3: local temporary table
Local temporary tables are visible only to their creators during the same connection to an instance of SQL Server as when the tables were first created or referenced. Local temporary tables are deleted after the user disconnects from the instance of SQL Server.

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) <https://technet.microsoft.com/en-us/library/ms186986.aspx>

NEW QUESTION 201

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You create a table named Customer by running the following Transact-SQL statement:

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

You must insert the following data into the Customer table:

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

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

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

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

NEW QUESTION 202

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 HistoryData IS NOT NULL

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

Answer Area

⏪

⏩

⏴

⏵

Answer:

Explanation: References:

[https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-t](https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-table)

NEW QUESTION 203

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 204

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 206

You need to create a stored procedure that meets the following requirements:

*Produces a warning if the credit limit parameter is greater than 7,000

*Propagates all unexpected errors to the calling process

How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQP segments to the correct locations. 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	Answer Area
<div>RAISERROR ('Warning: Credit limit is over 7,000!', 16, 1)</div>	<pre>CREATE PROC dbo.UpdateCustomer @CustomerID int, @CreditLimit money AS BEGIN DECLARE @ErrorMessage varchar(1000) BEGIN TRY</pre>
<div>RAISERROR ('Warning: Credit limit is over 7,000!', 10, 1)</div>	<div>T<div>Transact-SQL segment</div></div>
<div>THROW 51000, 'Warning: Credit limit is over 7,000!', 1</div>	<pre> UPDATE dbo.Customer SET CreditLimit = @CreditLimit WHERE CustomerID = @CustomerID END TRY</pre>
<div>THROW</div>	<pre> BEGIN CATCH SET @ErrorMessage = ERROR_MESSAGE() INSERT INTO dbo.ErrorLog (ApplicationID, [Date], ErrorMessage) VALUES (1, GETDATE(), @ErrorMessage)</pre>
<div>RAISERROR (@ErrorMessage, 16, 1)</div>	<div><div>Transact-SQL segment</div></div>
<div>RAISERROR (@ErrorMessage, 10, 1)</div>	<pre> END CATCH END</pre>
<div>THROW 51000, @ErrorMessage, 1</div>	
<div>RAISERROR (@ErrorMessage, 20, 1) WITH LOG</div>	

Answer:

Explanation: Box 1: THROW 51000, 'Warning: Credit limit is over 7,000!', 1

THROW raises an exception and transfers execution to a CATCH block of a TRY...CATCH construct in SQL Server.

THROW syntax:

THROW [{ error_number | @local_variable },

```
{ message | @local_variable },
{ state | @local_variable } ] [ ; ]
```

Box 2: RAISERROR (@ErrorMessage, 16,1)

RAISERROR generates an error message and initiates error processing for the session. RAISERROR can either reference a user-defined message stored in the sys.messages catalog view or build a message dynamically. The message is returned as a server error message to the calling application or to an associated CATCH block of a TRY...CATCH construct. New applications should use THROW instead.

Severity levels from 0 through 18 can be specified by any user. Severity levels from 19 through 25 can only be specified by members of the sysadmin fixed server role or users with ALTER TRACE permissions. For severity levels from 19 through 25, the WITH LOG option is required.

On Severity level 16. Using THROW to raise an exception

The following example shows how to use the THROW statement to raise an exception. Transact-SQL

```
THROW 51000, 'The record does not exist.', 1;
```

Here is the result set.

Msg 51000, Level 16, State 1, Line 1 The record does not exist.

Note: RAISERROR syntax:

```
RAISERROR ( { msg_id | msg_str | @local_variable }
{ ,severity ,state }
[ ,argument [ ,...n ] ] )
[ WITH option [ ,...n ] ]
```

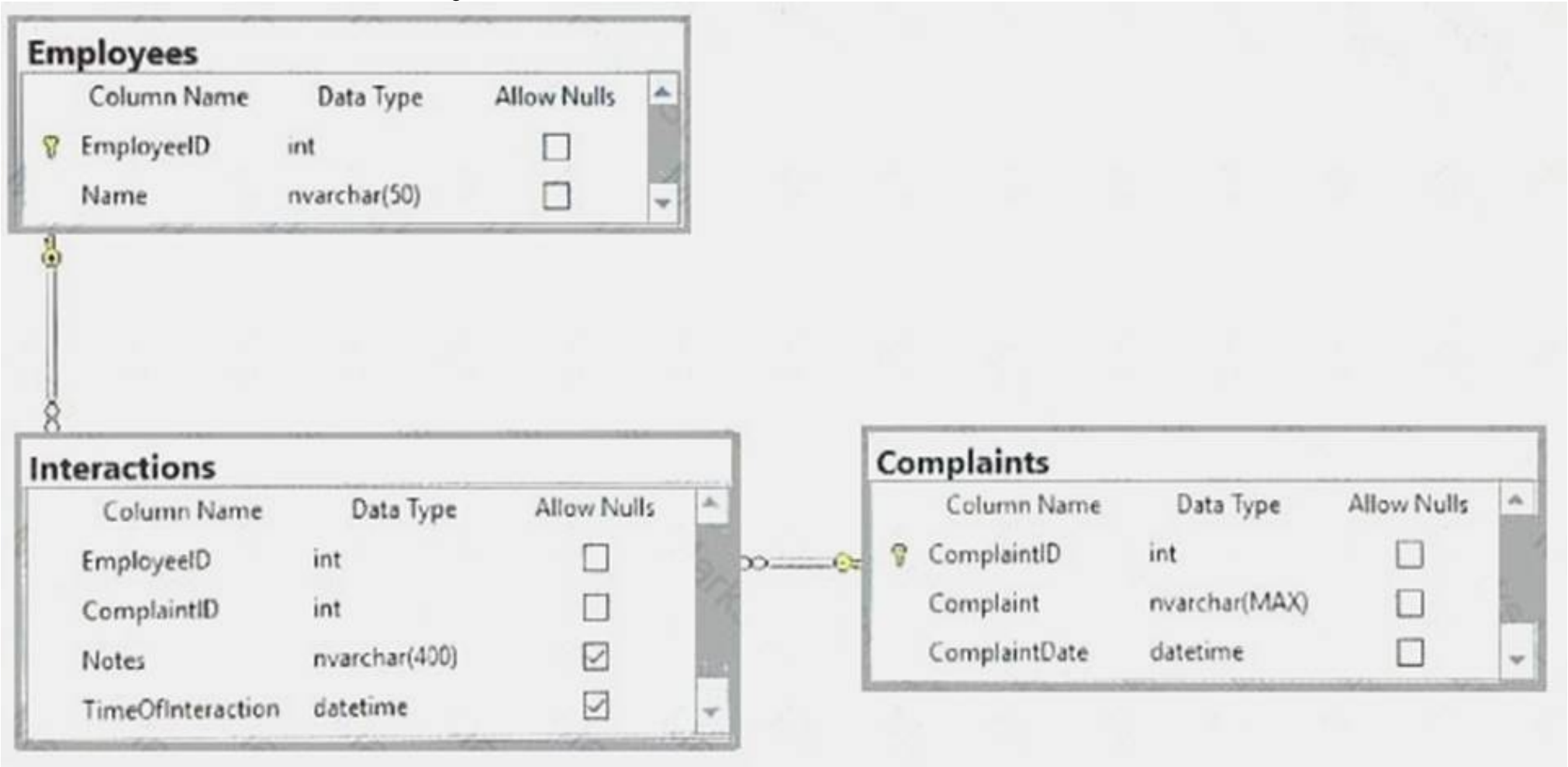
Note: The ERROR_MESSAGE function returns the message text of the error that caused the CATCH block of a TRY...CATCH construct to be run.

References:

<https://msdn.microsoft.com/en-us/library/ms178592.aspx> <https://msdn.microsoft.com/en-us/library/ms190358.aspx> <https://msdn.microsoft.com/en-us/library/ee677615.aspx>

NEW QUESTION 209

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

Answer:

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 212

You need to develop a function that returns a list of courses grouped by the total number of students in a course. The function must list only courses that have more than a specific number of students. The specific number of students is defined as an input variable for the function.
How should you complete the function? To answer, select the appropriate Transact-SQL segments in the answer area.
NOTE: Each correct selection is worth one point.

CREATE FUNCTION CoursesWithMoreThan (@totalStudents INT)
 RETURNS

	▼
TABLE	
HAVING	
WHERE	
INT	
SUM(cp.NumStudents) AS NumStudents	
SUM(cp.NumStudents)	

 AS
 RETURN
 SELECT c.Course,

	▼
TABLE	
HAVING	
WHERE	
INT	
SUM(cp.NumStudents) AS NumStudents	
SUM(cp.NumStudents)	

 FROM dbo.Courses c
 INNER JOIN dbo.CourseStudents cp ON c.CourseID = cp.CourseID

	▼	SUM(cp.NumStudents) > @totalStudents
TABLE		
HAVING		
WHERE		
INT		
SUM(cp.NumStudents) AS NumStudents		
SUM(cp.NumStudents)		

Answer:

Explanation:

CREATE FUNCTION CoursesWithMoreThan (@totalStudents INT)
 RETURNS

	▼
TABLE	
HAVING	
WHERE	
INT	
SUM(cp.NumStudents) AS NumStudents	
SUM(cp.NumStudents)	

 AS
 RETURN
 SELECT c.Course,

	▼
TABLE	
HAVING	
WHERE	
INT	
SUM(cp.NumStudents) AS NumStudents	
SUM(cp.NumStudents)	

 FROM dbo.Courses c
 INNER JOIN dbo.CourseStudents cp ON c.CourseID = cp.CourseID

	▼	SUM(cp.NumStudents) > @totalStudents
TABLE		
HAVING		
WHERE		
INT		
SUM(cp.NumStudents) AS NumStudents		
SUM(cp.NumStudents)		

NEW QUESTION 213

You have a table named Cities that has the following two columns: CityID and CityName. The CityID column uses the int data type, and CityName uses nvarchar(max).

You have a table named RawSurvey. Each row includes an identifier for a question and the number of persons that responded to that question from each of four cities. The table contains the following representative data:

QuestionID	Tokyo	Boston	London	New York
Q1	1	42	48	51
Q2	22	39	58	42
Q3	29	41	61	33
Q4	62	70	60	50
Q5	63	31	41	21
Q6	32	1	16	34

A reporting table named SurveyReport has the following columns: CityID, QuestionID, and RawCount, where RawCount is the value from the RawSurvey table.

You need to write a Transact-SQL query to meet the following requirements:

Retrieve data from the RawSurvey table in the format of the SurveyReport table.

The CityID must contain the CityID of the city that was surveyed.

The order of cities in all SELECT queries must match the order in the RawSurvey table.

The order of cities in all IN statements must match the order in the RawSurvey table.

Construct the query using the following guidelines:

Use one-part names to reference tables and columns, except where not possible.

ALL SELECT statements must specify columns.

Do not use column or table aliases, except those provided.

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 Rawcount
2      from (select cityid,questionid,rawcount) AS t1
3      unpivot
4      (rawcount for questionid in (QuestionID)) AS t2

```

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

Answer:

Explanation: 1 SELECT Rawcount

2 from (select cityid,questioned,rawcount) AS t1

3 unpivot

4 (rawcount for questioned in (QuestionID)) AS t2

5 JOIN t2

6. ON t1.CityName = t2.cityName

UNPIVOT must be used to rotate columns of the Rawsurvey table into column values. References: [https://technet.microsoft.com/en-us/library/ms177410\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx)

NEW QUESTION 216

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 on this series.

You have a database that tracks orders and deliveries for customers in North America. System versioning is enabled for all tables. The database contains the Sales.Customers, Application.Cities, and Sales.CustomerCategories tables.

Details for the Sales.Customers table are shown in the following table:

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 values
StandardDiscountPercentage	int	does not allow values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow values
DeliveryLocation	geography	does not allow values
PhoneNumber	nvarchar(20)	does not allow values
ValidFrom	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW START
ValidTo	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW END

Details for the Application.Cities table are shown in the following table:

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

Details for the Sales.CustomerCategories table are shown in the following table:

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

You need to create a query that meets the following requirements:

- For customers that are not on a credit hold, return the CustomerID and the latest recorded population for the delivery city that is associated with the customer.
- For customers that are on a credit hold, return the CustomerID and the latest recorded population for the postal city that is associated with the customer.

Which two Transact-SQL queries will achieve the goal? Each correct answer presents a complete solution.

- A**
- ```
SELECT CustomerID, LatestRecordedPopulation
FROM Sales.Customers
CROSS JOIN Application.Cities
WHERE (IsOnCreditHold = 0 AND DeliveryCityID = CityID)
OR (IsOnCreditHold = 1 AND PostalCityID = CityID)
```
- B**
- ```
SELECT CustomerID, LatestRecordedPopulation
FROM Sales.Customers
INNER JOIN Application.Cities AS A
ON A.CityID = IIF(IsOnCreditHold = 0, DeliveryCityID, PostalCityID)
```
- C**
- ```
SELECT CustomerID, ISNULL(A.LatestRecordedPopulation, B.LatestRecorded Population)
FROM Sales.Customers
INNER JOIN Application.Cities AS A ON A.CityID = DeliveryCityID
INNER JOIN Application.Cities AS B ON B.CityID = PostalCityID
WHERE IsOnCreditHold = 0
```
- D**
- ```
SELECT CustomerID, LatestRecordedPopulation,
IIF(IsOnCreditHold = 0, DeliveryCityID, PostalCityID) As CityId
FROM Sales.Customers
INNER JOIN Application.Cities AS A ON A.CityID = CityId
```


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

Answer: AB

Explanation: Using Cross Joins

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.

However, if a WHERE clause is added, the cross join behaves as an inner join. B: You can use the IIF in the ON-statement.

IIF returns one of two values, depending on whether the Boolean expression evaluates to true or false in SQL Server.

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) <https://msdn.microsoft.com/en-us/library/hh213574.aspx>

NEW QUESTION 217

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 contains a single table named tblVehicleRegistration. The table is defined as follows:

Column name	Data type	Description
VehicleId	int	the primary key for the table
RegistrationNumber	varchar(5)	a vehicle registration number that contains only letters and numbers
RegistrationDate	date	the vehicle registration date
UserId	int	an identifier for the vehicle owner

You run the following query:

```
SELECT UserId FROM tblVehicleRegistration
WHERE RegistrationNumber = 20012
AND RegistrationDate > '2016-01-01'
```

The query output window displays the following error message: "Conversion failed when converting the varchar value 'AB012' to data type int."

You need to resolve the error.

Solution: You modify the Transact-SQL statement as follows:

```
SELECT UserId FROM tblVehicleRegistration
WHERE RegistrationNumber = '20012'
AND RegistrationDate > '2016-01-01'
```

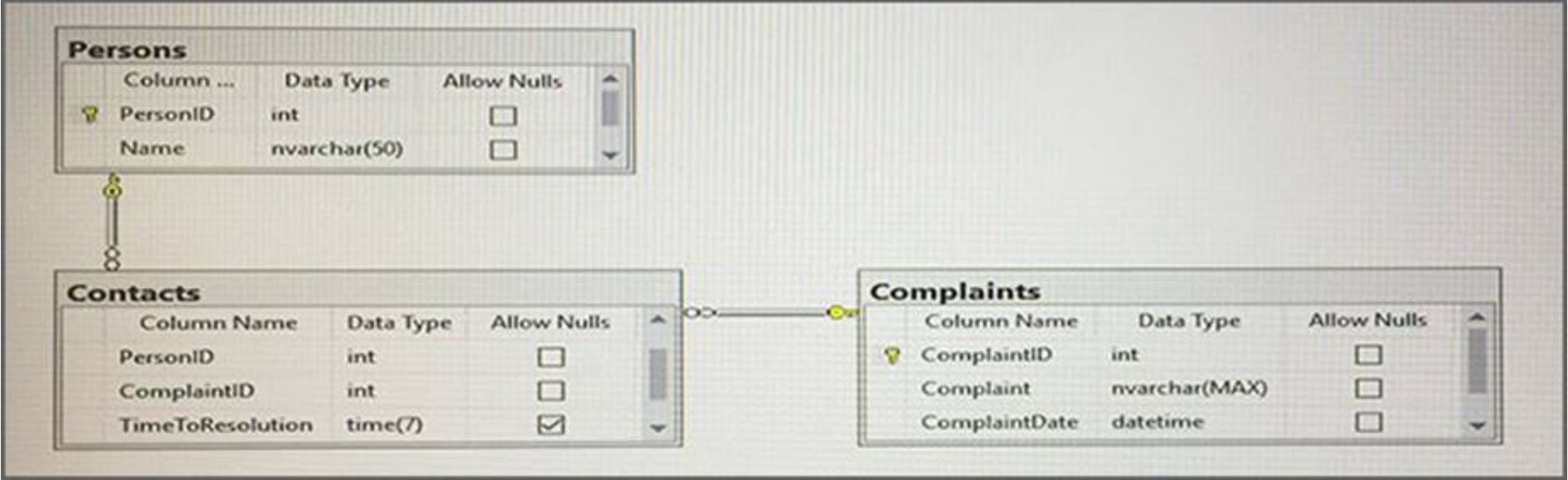
Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 221

You have a database that contains the following tables.



You need to create a query that lists all complaints from the Complaints table, and the name of the person handling the complaints if a person is assigned. The ComplaintID must be displayed first, followed by the person name.

Construct the query using the following guidelines:

- Use two-part column names.
- Use one-part table names.

- Do not use aliases for column names or table names.
- Do not use 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 Complaints.ComplaintId,
2 FROM
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.

Answer:

Explanation: SELECT

Complaints.ComplaintID, Persons.Name FROM

Complaints LEFT OUTER JOIN Contacts ON Complaints.ComplaintID = Contacts.ComplaintID

LEFT OUTER JOIN Persons ON Contacts.PersonID = Persons.PersonID

NEW QUESTION 225

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 plan to run the following query to update tasks that are not yet started: UPDATE Task SET StartTime = GETDATE() WHERE StartTime IS NULL

You need to return the total count of tasks that are impacted by this UPDATE operation, but are not associated with a project.

What set of Transact-SQL statements should you run?

A

```
DECLARE @startedTasks TABLE(ProjectId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT inserted.ProjectId INTO @startedTasks WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE ProjectId IS NOT NULL
```

B

```
DECLARE @startedTasks TABLE(TaskId int, ProjectId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT deleted.TaskId, deleted.ProjectId INTO @startedTasks
WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE ProjectId IS NULL
```

C

```
DECLARE @startedTasks TABLE(TaskId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT deleted.TaskId, INTO @startedTasks WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE TaskId IS NOT NULL
```

D

```
UPDATE Task SET StartTime = GETDATE() WHERE StartTime IS NULL
SELECT @@ROWCOUNT
```

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

Answer: B

NEW QUESTION 226

You have a database that contains the following tables:

Table	Columns
Sales.Customers	CustomerID, CustomerName
Sales.Invoices	CustomerID, ConfirmedReceivedBy

A delivery person enters an incorrect value for the CustomerID column in the Invoices table and enters the following text in the ConfirmedReceivedBy column:
“Package signed for by the owner Tim.”

You need to find the records in the Invoices table that contain the word Tim in the CustomerName field. 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

SELECT CustomerID FROM Sales.Customers

SELECT CustomerID FROM Sales.Invoices

INNER JOIN Sales.Customers
ON Sales.Customers.CustomerID = Sales.Invoices.CustomerID

FULL JOIN Sales.Customers
ON Sales.Customers.CustomerID = Sales.Invoices.CustomerID

WHERE CustomerName LIKE '%tim%'

WHERE ConfirmedReceivedBy IN (SELECT CustomerName
FROM Sales.Customers)

UNION

UNION ALL

Answer Area

Transact-SQL segment

Transact-SQL segment

Transact-SQL segment

Transact-SQL segment

WHERE ConfirmedReceivedBy LIKE '%tim%'

Answer:

Explanation: Box 1: SELECT CustomerID FROM Sales.Invoices

Box 2: INNER JOIN Sales.Customers.CustomerID = Sales.Invoices.CustomerID Box 3: WHERE CustomerName LIKE '%tim%'

Box 4: WHERE ConfirmedReceiveBy IN (SELECT CustomerName FROM Sales.Customers)

NEW QUESTION 231

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 236

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 237

You have a database that stored information about servers and application errors. The database contains the following tables.
Servers

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

Errors

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

You are building a webpage that shows the three most common errors for each server. You need to return the data for the webpage. How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct location. 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

svr.ServerID

errs.ServerID

INNER JOIN

CROSS APPLY

WITHIN GROUP

WHERE ServerID = svr.ServerID

WHERE ServerID = errs.ErrorID

Answer Area

SELECT

Transact-SQL segment

, errs.LogMessage

FROM Servers AS svr

Transact-SQL segment

(

SELECT TOP 3 LogMessage

FROM Errors

Transact-SQL segment

ORDER BY Occurrences

) AS errs

Answer:

Explanation:

Transact-SQL segments

svr.ServerID

errs.ServerID

INNER JOIN

CROSS APPLY

WITHIN GROUP

WHERE ServerID = svr.ServerID

WHERE ServerID = errs.ErrorID

Answer Area

SELECT

svr.ServerID

, errs.LogMessage

FROM Servers AS svr

CROSS APPLY

(

SELECT TOP 3 LogMessage

FROM Errors

WHERE ServerID = svr.ServerID

ORDER BY Occurrences

) AS errs

NEW QUESTION 240

.....

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](#)