

70-761 Dumps

Querying Data with Transact-SQL (beta)

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

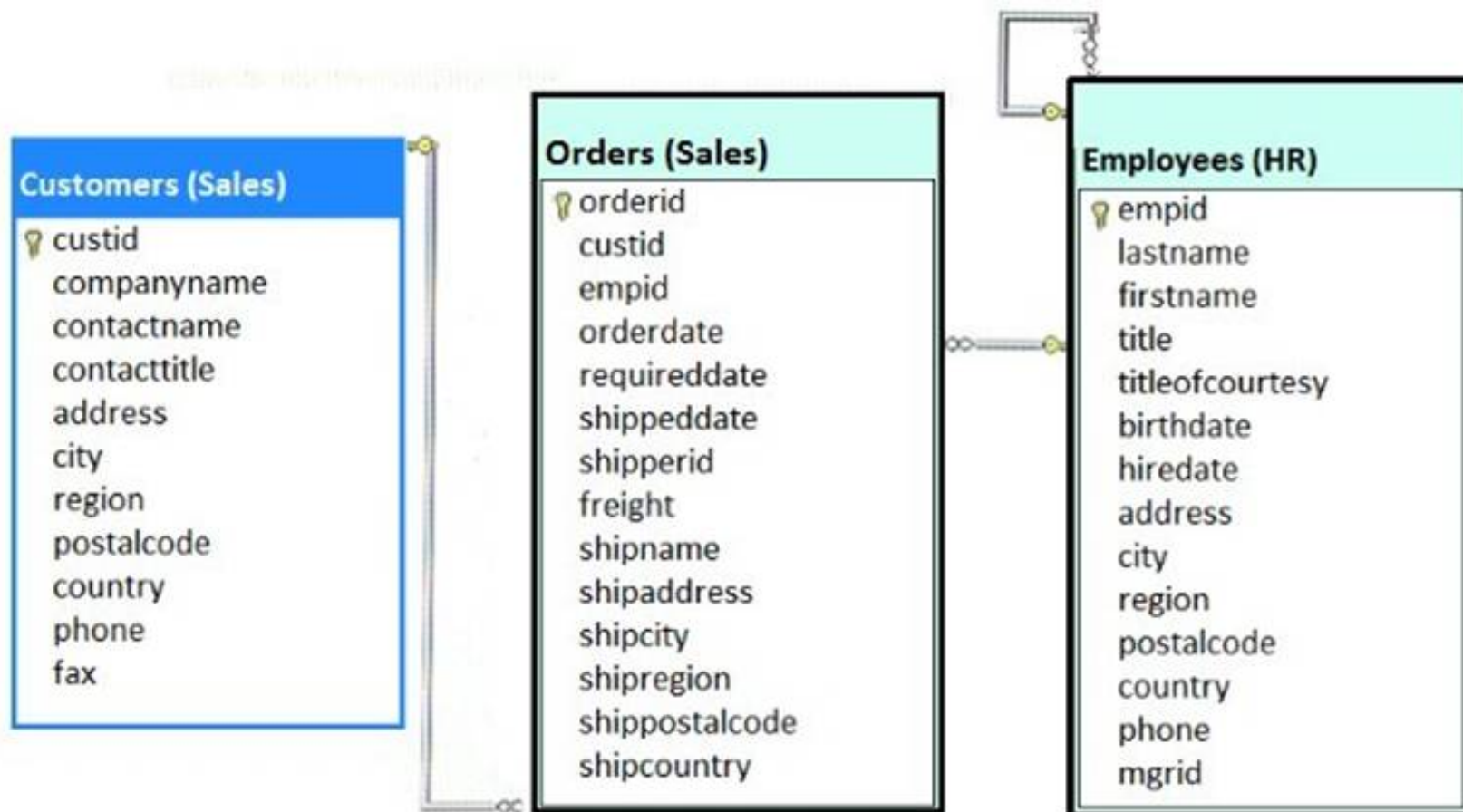


NEW QUESTION 1

Note: This question is part of a series of questions that 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
ORDER BY DateofOrder DESC
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

We need a GROUP BY statement as we want to return an order for each customer.

NEW QUESTION 2

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

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

You have a table named Products that stores information about products your company sells. The table has a column named ListPrice that stores retail pricing information for products.

Some products are used only internally by the company. Records for these products are maintained in the Products table for inventory purposes. The price for each of these products is \$0.00. Customers are not permitted to order these products.

You need to increase the list price for products that cost less than \$100 by 10 percent. You must only increase pricing for products that customers are permitted to order.

Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

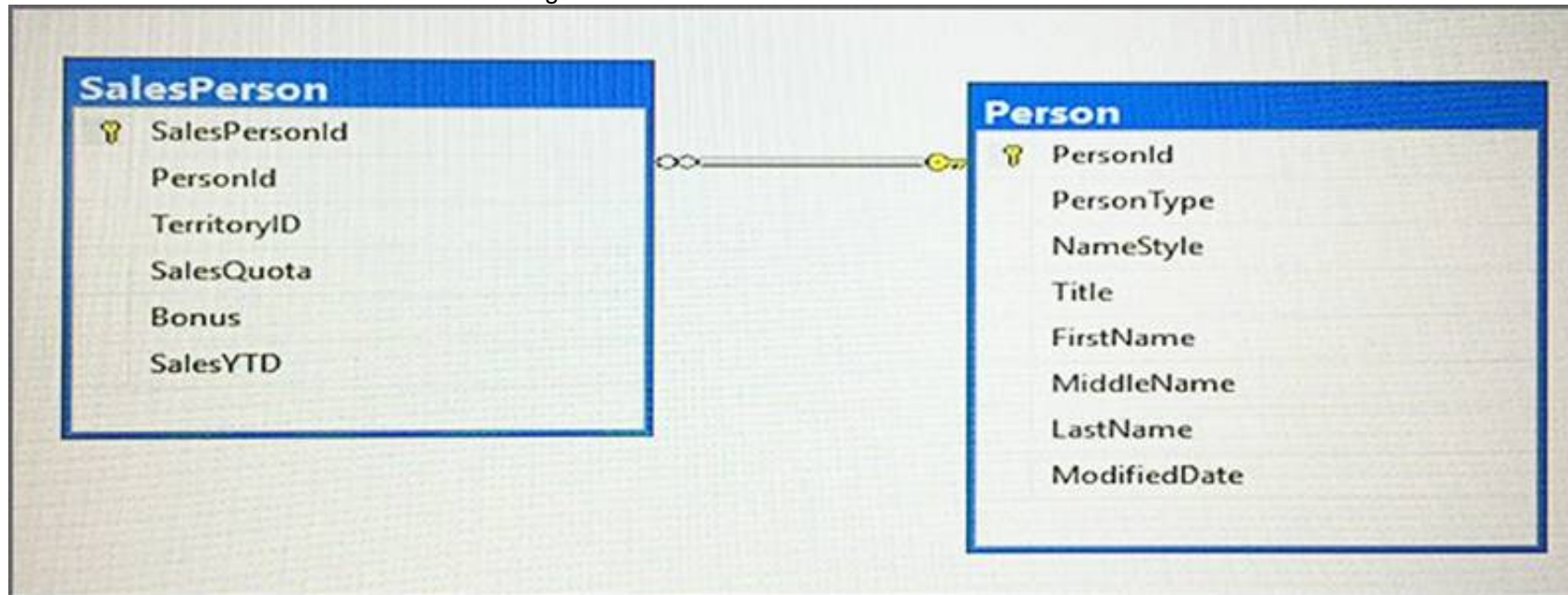
Answer: B

Explanation:

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

NEW QUESTION 3

You have a database that contains the following tables.



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

Construct the query using the following guidelines:

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
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 top 3 lastname,salesYTD
2 FROM Person AS p INNER JOIN SalesPerson AS s 3 ON p.PersonID = s.SalesPersonID
4 WHERE territoryid is null 5 order by salesytd dsec

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

1 SELECT top 3 lastname,salesYTD
2 FROM Person AS p INNER JOIN SalesPerson AS s 3 ON p.PersonID = s.SalesPersonID
4 WHERE territoryid is not null 5 order by salesytd desc

Note:
On line 4 add a not before null. On line 5 change dsec to desc.

NEW QUESTION 4

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:

The tables include the following records: Customer_CRMSystem

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

Customer_HRSystem

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

Records that contain null values for CustomerCode can be uniquely identified by CustomerName.
You need to create a list of all unique customers that appear in either table. 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: E

Explanation:

UNION combines the results of two or more queries into a single result set that includes all the rows that belong to all queries in the union. The UNION operation is different from using joins that combine columns from two tables.

NEW QUESTION 5

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

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

You are building a stored procedure that will be used by hundreds of users concurrently.

You need to store rows that will be processed later by the stored procedure. The object that stores the rows must meet the following requirements:

Be indexable

Contain up-to-date statistics

Be able to scale between 10 and 100,000 rows

The solution must prevent users from accessing one another's data. Solution: You create a table variable in the stored procedure.

Does this meet the goal?

A. Yes

B. No

Answer: B

NEW QUESTION 6

You have a database named DB1 that contains a table named HR.Employees. HR.Employees contains two columns named ManagerID and EmployeeID.

ManagerID refers to EmployeeID.

You need to create a query that returns a list of all employees, the manager of each employee, and the numerical level of each employee in your organization's hierarchy.

Which five statements should you add to the query in sequence? To answer, move the appropriate statements from the list of statements to the answer area and arrange them in the correct order.

Statements	Answer Area
<pre>SELECT Employees.ManagerId, Employees.EmployeeId, EmployeeLevel+1 FROM Employees JOIN Managers ON Employees.EmployeeId = Managers.ManagerId)</pre>	
<pre>WITH Managers AS (</pre>	
<pre>SELECT* FROM Managers ORDER BY ManagerID</pre>	
<pre>SELECT ManagerId, EmployeeId, 0 AS EmployeeLevel FROM Employees WHERE ManagerId IS NULL</pre>	
<pre>UNION ALL</pre>	
<pre>UNION</pre>	

A. Mastered

B. Not Mastered

Answer: A

Explanation:

References:

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

NEW QUESTION 7

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	Answer Area
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)	

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

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

NEW QUESTION 8

Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You 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 9

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 10

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

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

You must insert the following data into the Customer table:

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

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

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

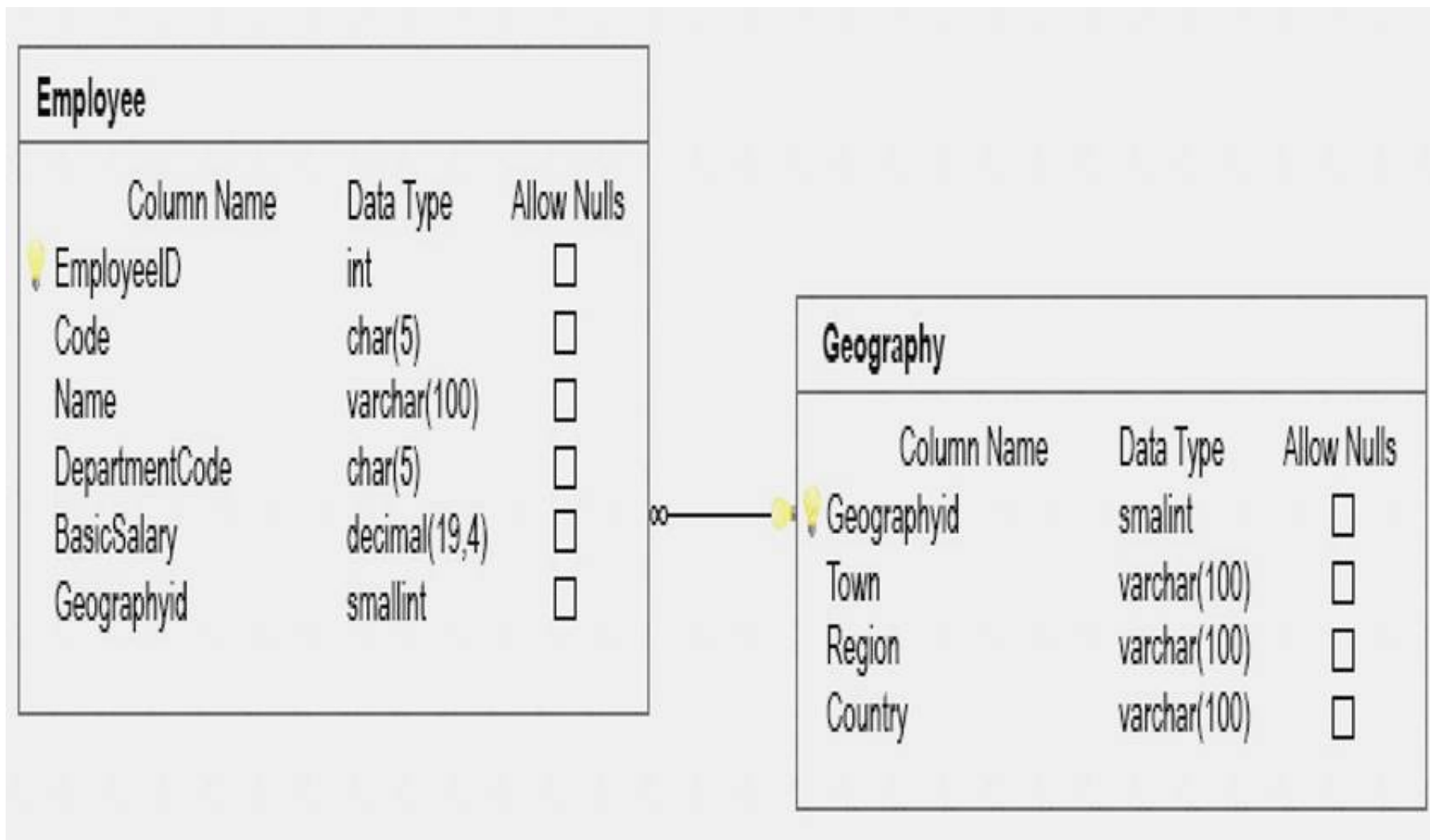
Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 10

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	

- A. Mastered
- B. Not Mastered

Answer: A

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 11

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

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

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

You need to recommend a query that returns all the customers. The query must also return the number of orders that each customer placed in 2016.

Solution: You recommend the following query:

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

Does this meet the goal?

- A. Yes
- B. No

Answer: A

NEW QUESTION 16

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

You have a database that stores sales and order information.

Users must be able to extract information from the tables on an ad hoc basis. They must also be able to reference the extracted information as a single table.

You need to implement a solution that allows users to retrieve the data required, based on variables defined at the time of the query.

What should you implement?

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

Answer: C

Explanation:

User-defined functions that return a table data type can be powerful alternatives to views. These functions are referred to as table-valued functions. A table-valued user-defined function can be used where table or view expressions are allowed in Transact-SQL queries. While views are limited to a single SELECT statement, user-defined functions can contain additional statements that allow more powerful logic than is possible in views.

A table-valued user-defined function can also replace stored procedures that return a single result set. References: [https://technet.microsoft.com/en-us/library/ms191165\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms191165(v=sql.105).aspx)

NEW QUESTION 18

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 22

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

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

You have a 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 26

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

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

A value of 1 in the IsActive column indicates that a user is active.

You need to create a count for active users in each role. If a role has no active users. You must display a zero as the active users count.

Which Transact-SQL statement should you run?

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

Answer: B

NEW QUESTION 30

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

Does the solution meet the goal?

- A. Yes
B. No

Answer: A

Explanation:

With the INSERT INTO..VALUES statement we can insert both values with just one statement. This ensures that both records or neither is inserted.

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

NEW QUESTION 32

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.

- A. Mastered
- B. Not Mastered

Answer: A

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 34

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
    ,
    FROM MSL.dbo.Bug
WHERE
```

Product
Description
DateCreated
ReportingUser

```
like @product
```

ASCII(Product)
CAST(Id AS TEXT)
TRANSLATED(Id,'CI','CS')
Product COLLATE SQL_Latin1_General_CP1_CS_AS

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

References:

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

NEW QUESTION 38

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 40

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:

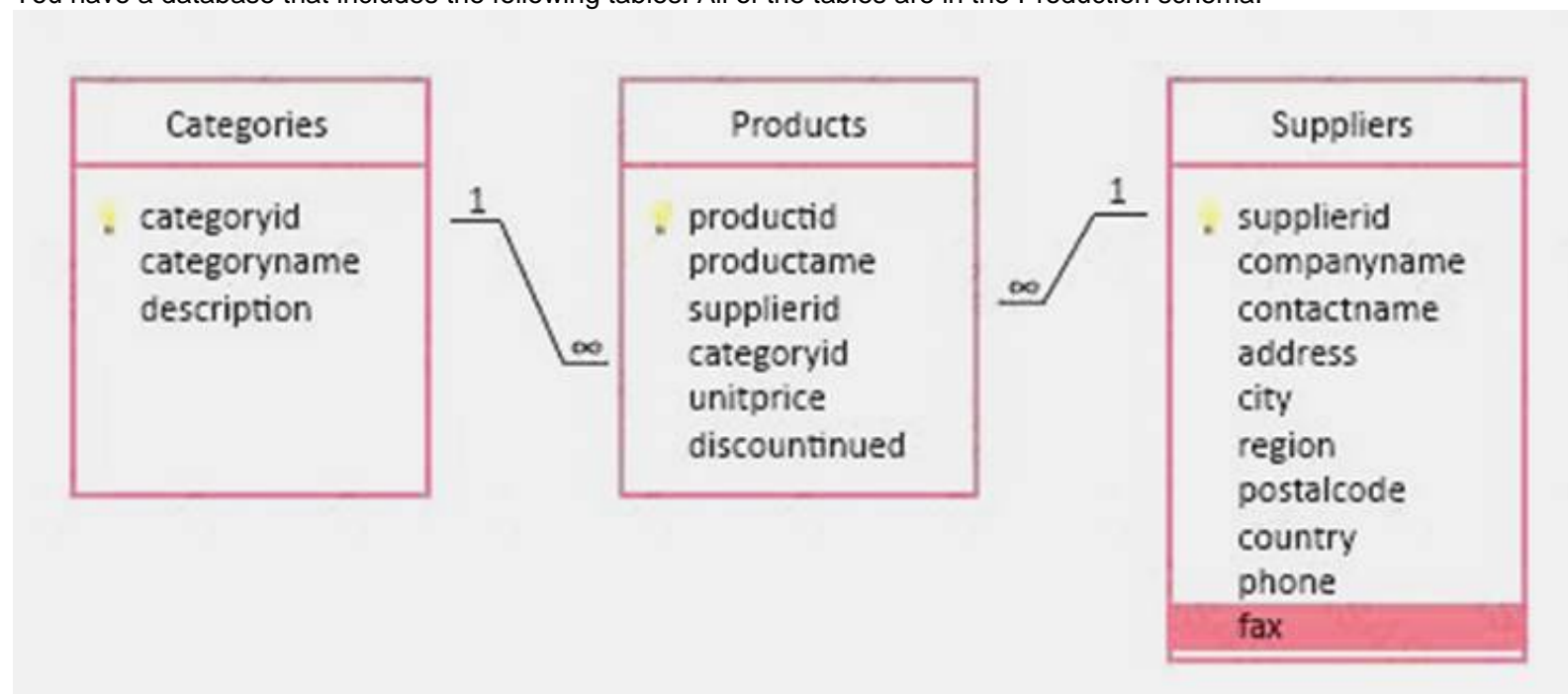
Using an Indexed View would allow you to keep your base data in properly normalized tables and maintain data-integrity while giving you the denormalized "view" of that data.

References:

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

NEW QUESTION 42

You have a database that includes the following tables. All of the tables are in the Production schema.



You need to create a query that returns a list of product names for all products in the Beverages category. Construct the query using the following guidelines:

Use the first letter of the table name as the table alias.

Use two-part column names.

Do not surround object names with square brackets.

Do not use implicit joins.

Do not use variables.

Use single quotes to surround literal values.

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 p.productname
2  FROM Production.Categories AS c
3  inner join production.products as p on c.categoryid*p.categoryid
4  WHERE c.categoryname = 'Beverages'
  
```

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

1 SELECT p.productname

2 FROM Production.categories AS c
3 inner join production.products as p on c.categoryid=p.categoryid 4 WHERE c.categoryname = 'Beverages'
Note: On line 3 change * to =

NEW QUESTION 45

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 < 100
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

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

NEW QUESTION 50

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 51

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 53

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	Answer Area
<div>RAISERROR (60000, 16, 1)</div>	CREATE PROCEDURE ADDEmployee
<div>THROW 60000, 'The record was not added.', 1</div>	@FirstName nvarchar(30),
<div>IF XACT_STATE () <> 0 ROLLBACK TRANSACTION</div>	@LastName nvarchar(40),
<div>IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION</div>	@Title nvarchar(50) = 'New Employee',
<div>SAVE TRANSACTION AddEmployee</div>	@DepartmentID smallint,
<div>COMMIT TRANSACTION</div>	@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

- A. Mastered
B. Not Mastered

Answer: A

Explanation:


Transact-SQL segments	Answer Area
RAISERROR (60000, 16, 1)	CREATE PROCEDURE ADDEmployee
THROW 60000, 'The record was not added.', 1	@FirstName nvarchar(30),
IF XACT_STATE () <> 0 ROLLBACK TRANSACTION	@LastName nvarchar(40),
IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION	@Title nvarchar(50) = 'New Employee',
SAVE TRANSACTION AddEmployee	@DepartmentID smallint,
COMMIT TRANSACTION	@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 57

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 need to create the query for the Sales by Region report.

Which function should you apply to each column? To answer, select the appropriate options in the answer area.

Answer area

Column

Function

MiddleName

	▼
NULLIF	
REPLACE	
COALESCE	

RegionCode

	▼
NULLIF	
REPLACE	
COALESCE	

- A. Mastered
B. Not Mastered

Answer: A

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.

If MiddleName is NULL, FirstName must be displayed. If both FirstName and MiddleName have null values, the word Unknown must be displayed.

The following example shows how COALESCE selects the data from the first column that has a nonnull value.

SELECT Name, Class, Color, ProductNumber, COALESCE(Class, Color, ProductNumber) AS FirstNotNull FROM Production.Product;

Not NULLIF: NULLIF returns the first expression if the two expressions are not equal. If the expressions are equal, NULLIF returns a null value of the type of the first expression.

Box 2: COALESCE

If RegionCode is NULL, the word Unknown must be displayed.

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

NEW QUESTION 59

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

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

A value of 1 in the IsActive column indicates that a user is active.

You need to create a count for active users in each role. If a role has no active users. you must display a zero as the active users count.

Which Transact-SQL statement should you run?

- A** SELECT R.RoleName, COUNT(U.UserId) AS ActiveUserCount FROM tblRoles R
LEFT JOIN (SELECT UserId, RoleId FROM tblUsers WHERE IsActive = 1) U ON U.RoleId = R.RoleId
GROUP BY R.RoleId, R.RoleName
- B** SELECT R.RoleName, U.ActiveUserCount FROM tblRoles R
INNER JOIN (SELECT RoleId, COUNT(*) AS ActiveUserCount FROM tblUsers WHERE IsActive = 1
GROUP BY RoleId) U ON R.RoleId = U.RoleId
- C** SELECT R.RoleName, COUNT(*) AS ActiveUserCount FROM tblRoles R
LEFT JOIN (SELECT UserId, RoleId FROM tblUsers WHERE IsActive = 1) U ON U.RoleId = R.RoleId
GROUP BY R.RoleId, R.RoleName
- D** SELECT R.RoleName, U.ActiveUserCount FROM tblRoles R CROSS JOIN
(SELECT COUNT(*) AS ActiveUserCount FROM tblUsers WHERE IsActive = 1) U

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

Answer: C

NEW QUESTION 60

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

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

You 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 61

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 66

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
NULL	NULL	Abbottsburg	\$45453.25
NULL	NULL	Absecon	\$33140.15
NULL	NULL	Accomac	\$43226.80
NULL	NULL	Aceitunas	\$23001.40

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: C

Explanation:

In the result sets that are generated by the GROUP BY operators, NULL has the following uses:

If a grouping column contains NULL, all null values are considered equal, and they are put into one NULL group.

When a column is aggregated in a row, the value of the column is shown as NULL. Example of GROUP BY ROLLUP result set:

Region	Country	Store	SalesPersonID	Total Sales
NULL	NULL	NULL	NULL	297597.8
NULL	NULL	NULL	284	33633.59
NULL	NULL	Spa and Exercise Outfitters	284	32774.36
NULL	FR	Spa and Exercise Outfitters	284	32774.36
Europe	FR	Spa and Exercise Outfitters	284	32774.36
NULL	NULL	Versatile Sporting Goods Company	284	859.232
NULL	DE	Versatile Sporting Goods Company	284	859.232
Europe	DE	Versatile Sporting Goods Company	284	859.232
NULL	NULL	NULL	286	246272.4
NULL	NULL	Spa and Exercise Outfitters	286	246272.4
NULL	FR	Spa and Exercise Outfitters	286	246272.4
Europe	FR	Spa and Exercise Outfitters	286	246272.4
NULL	NULL	NULL	289	17691.83
NULL	NULL	Versatile Sporting Goods Company	289	17691.83
NULL	DE	Versatile Sporting Goods Company	289	17691.83
Europe	DE	Versatile Sporting Goods Company	289	17691.83

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 70

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 71

You have a project management application. The application uses a Microsoft SQL Server database to store data. You are developing a software bug tracking add-on for the application.
The add-on must meet the following requirements:
Allow case sensitive searches for product.
Filter search results based on exact text in the description.
Support multibyte Unicode characters.
You run the following Transact-SQL statement:

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

You need to display a comma separated list of all product bugs filed by a user named User1.
How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct locations. Each Transact-SQL segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.
NOTE: Each correct selection is worth one point.

Transact-SQL segments

@List NVARCHAR(MAX) = ''

@List NVARCHAR(MAX)

@List TABLE

@List=Product+ ',' + @List

@List=@List+ ',' + Product

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

Answer Area

DECLARE

Transact-SQL segment

SELECT

Transact-SQL segment

From Bug WHERE ReportingUser = User1
SELECT @List

- A. Mastered
- B. Not Mastered

Answer: A

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

NEW QUESTION 74

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section. you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You 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 76

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

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

You need to write a query that will change the value of the job title column to Customer Representative for any employee who lives in Seattle and has a job title of Sales Representative. If the employee does not have a manager defined, you must not change the title.

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

Transact-SQL segments

SET title = 'Customer Representative'

WHERE title = 'Sales Representative'
AND city = 'Seattle' AND mgrid IS NOT NULL

UPDATE HR.Employees

SET city = 'Seattle' and mgrid = NULL

INSERT INTO HR.Employees

VALUES ('Customer Representative')

WHERE title = 'Sales Representative'

DELETE FROM HR.Employees

Answer Area

⏮
⏭
⏪
⏩

- A. Mastered
B. Not Mastered

Answer: A

Explanation:

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

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

You are creating a report to measure the impact of advertising efforts that were designed to attract new customers. The report must show the number of new customers per day for each customer category, but only if the number of new customers is greater than five.

You need to write the query to return data for the report.

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

Transact-SQL segments

CAST(Cust.AccountOpenedDate AS DATE)

DATEPART(day, Cust.AccountOpenedDate)

HAVING

WHERE

COUNT(Cust.CustomerId)

MAX(Cust.CustomerID)

RANK

GROUP BY

Answer Area

SELECT Count(Cust.CustomerId), CustCat.CustomerCategoryName,

Transact-SQL segment

FROM Sales.Customers AS Cust
INNER JOIN Sales.CustomerCategories AS CustCat
ON Cust.CustomerCategoryID = CustCat.CustomerCategoryID

Transact-SQL segment

CustCat.CustomerCategoryName,

Transact-SQL segment

Transact-SQL segment

Transact-SQL segment

> 5

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Transact-SQL segments

CAST(Cust.AccountOpenedDate AS DATE)

DATEPART(day, Cust.AccountOpenedDate)

HAVING

WHERE

COUNT(Cust.CustomerId)

MAX(Cust.CustomerID)

RANK

GROUP BY

Answer Area

SELECT Count(Cust.CustomerId), CustCat.CustomerCategoryName,

CAST(Cust.AccountOpenedDate AS DATE)

FROM Sales.Customers AS Cust
INNER JOIN Sales.CustomerCategories AS CustCat
ON Cust.CustomerCategoryID = CustCat.CustomerCategoryID

GROUP BY

CustCat.CustomerCategoryName,

CAST(Cust.AccountOpenedDate AS DATE)

WHERE

COUNT(Cust.CustomerId)

> 5

NEW QUESTION 85

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

Column name	Requirements
Id	<ul style="list-style-type: none"> primary key of the table value must be globally unique value must be automatically generated for INSERTs operations
Reason	<ul style="list-style-type: none"> stores reasons for the payment supports multilingual values supports values with 1 to 500 characters
Amount	<ul style="list-style-type: none"> stores monetary values must not produce rounding errors with calculations

Which Transact-SQL statement should you run?

- A. CREATE TABLE MiscellaneousPayment (Id uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY,Reason varchar(500),Amount money)
 B. CREATE TABLE MiscellaneousPayment (Id int identify(1,1)PRIMARY KEY,Reason nvarchar(500),Amount numeric(19,4))
 C. CREATE TABLE MiscellaneousPayment (Id uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY,Reason varchar(500),Amount decimal(19,4))
 D. CREATE TABLE MiscellaneousPayment (Id uniqueidentifier DEFAULT NEWID() PRIMARY KEY,Reason nvarchar(500),Amount decimal(19,4))

Answer: D

NEW QUESTION 87

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 run a query to find the total number of customers who have both 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 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: A

Explanation:

The SQL INTERSECT operator is used to return the results of 2 or more SELECT statements. However, it only returns the rows selected by all queries or data sets. If a record exists in one query and not in the other, it will be omitted from the INTERSECT results.

References: <https://www.techonthenet.com/sql/intersect.php>

NEW QUESTION 88

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

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

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

The following table describes the columns in Sales.Orders.

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

The following table describes the columns in Sales.OrderLines.

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

You need to create a function that accepts a CustomerID as a parameter and returns the following information:

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

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

Transact-SQL segments

- COUNT
- SUM
- AVG
- ORDER BY
- GROUP BY
- RETURNS INT
- RETURNS NULL ON NULL INPUT
- RETURNS TABLE

Answer Area

```
CREATE FUNCTION Sales.GetCustomerInformation(@CustomerID int)
```

Transact-SQL segment

```
AS  
RETURN
```

```
(  
    SELECT C.CustomerName, C.PhoneNumber, C.AccountOpenedDate,  
           C.StandardDiscountPercentage, C.CreditLimit, C.IsOnCreditHold,  
           (O.OrderID) AS TotalNumberOfOrders,  
           (OL.UnitPrice) AS TotalOrderPrice,  
           (OL.Quantity) AS AverageOrderQuantity  
    FROM Sales.Customers C  
    JOIN Sales.Orders AS O ON O.CustomerID = C.CustomerID  
    JOIN Sales.OrderLines AS OL ON OL.OrderID = O.OrderID  
    WHERE C.CustomerID = @CustomerID
```

Transact-SQL segment C.CustomerName, C.PhoneNumber, C.AccountOpenedDate,

C.StandardDiscountPercentage, C.CreditLimit, C.IsOnCreditHold

```
)
```


- A. Mastered
B. Not Mastered

Answer: A

Explanation:

Box1: RETURNS TABLE

The function should return the following information:

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

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

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

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

Need to use GROUP BY for the aggregate functions.

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

NEW QUESTION 90

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 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 H

Answer: G

NEW QUESTION 94

You have a table named HumanResources.Department that was created with the query shown in the exhibit. (Click the Exhibit button.)

```
CREATE TABLE HumanResources.Department  
(  
    DepID int IDENTITY(1,1) NOT NULL PRIMARY KEY CLUSTERED  
    , DeptName varchar(50) NOT NULL  
    , ManagerID INT NULL  
    , ParentDeptID int NULL  
    , SysStartTime datetime2 GENERATED ALWAYS AS ROW START NOT NULL  
    , SysEndTime datetime2 GENERATED ALWAYS AS ROW END NOT NULL  
    , PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime)  
)  
WITH (SYSTEM_VERSIONING = ON)  
;
```

You need to query temporal data in the table.

In the table below, identify the Transact-SQL segments that must be used to retrieve the appropriate data. NOTE: Make only one selection in each column.

Answer Area

Clause	At a particular point in time	Only from history table
All	<input type="radio"/>	<input type="radio"/>
FROM	<input type="radio"/>	<input type="radio"/>
AS OF	<input type="radio"/>	<input type="radio"/>
BETWEEN	<input type="radio"/>	<input type="radio"/>
CONTAINED IN	<input type="radio"/>	<input type="radio"/>

- A. Mastered
B. Not Mastered

Answer: A

Explanation:

AS OF: Returns a table with a rows containing the values that were actual (current) at the specified point in time in the past.

CONTAINED IN: If you search for non-current row versions only, we recommend you to use CONTAINED IN as it works only with the history table and will yield the best query performance.

NEW QUESTION 99

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 accepts a parameter of type geography and 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:

Return the average normalized readings named AverageReading.

Return the nearest mountain name named Mountain.

Do not return any other columns.

Exclude sensors for which no normalized reading exists. Construct the query using the following guidelines:

Use one part names to reference tables, columns and functions.

Do not use parentheses unless required.

Define column headings using the AS keyword.

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 avg (normalizedreading) as averagereading, location as mountain
2 FROM GroundSensors
3 WHERE normalizedreading is not null
```

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

1 SELECT avg (normalizedreading) as AverageReading, location as Mountain
2 FROM GroundSensors
3 WHERE normalizedreading is not null
Note: On line 1 change to AverageReading and change to Mountain.

NEW QUESTION 100

.....

Thank You for Trying Our Product

* 100% Pass or Money Back

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

* One year free update

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

* Trusted by Millions

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

* Shop Securely

All transactions are protected by VeriSign!

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

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