

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

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

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

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

If the credit limit is zero, you must delete the customer record while fetching data. You need to add the DELETE statement.

Solution: You add the following Transact-SQL statement:

```
IF @CreditLimit = 0  
    DELETE Customer  
    WHERE CURRENT OF cur
```

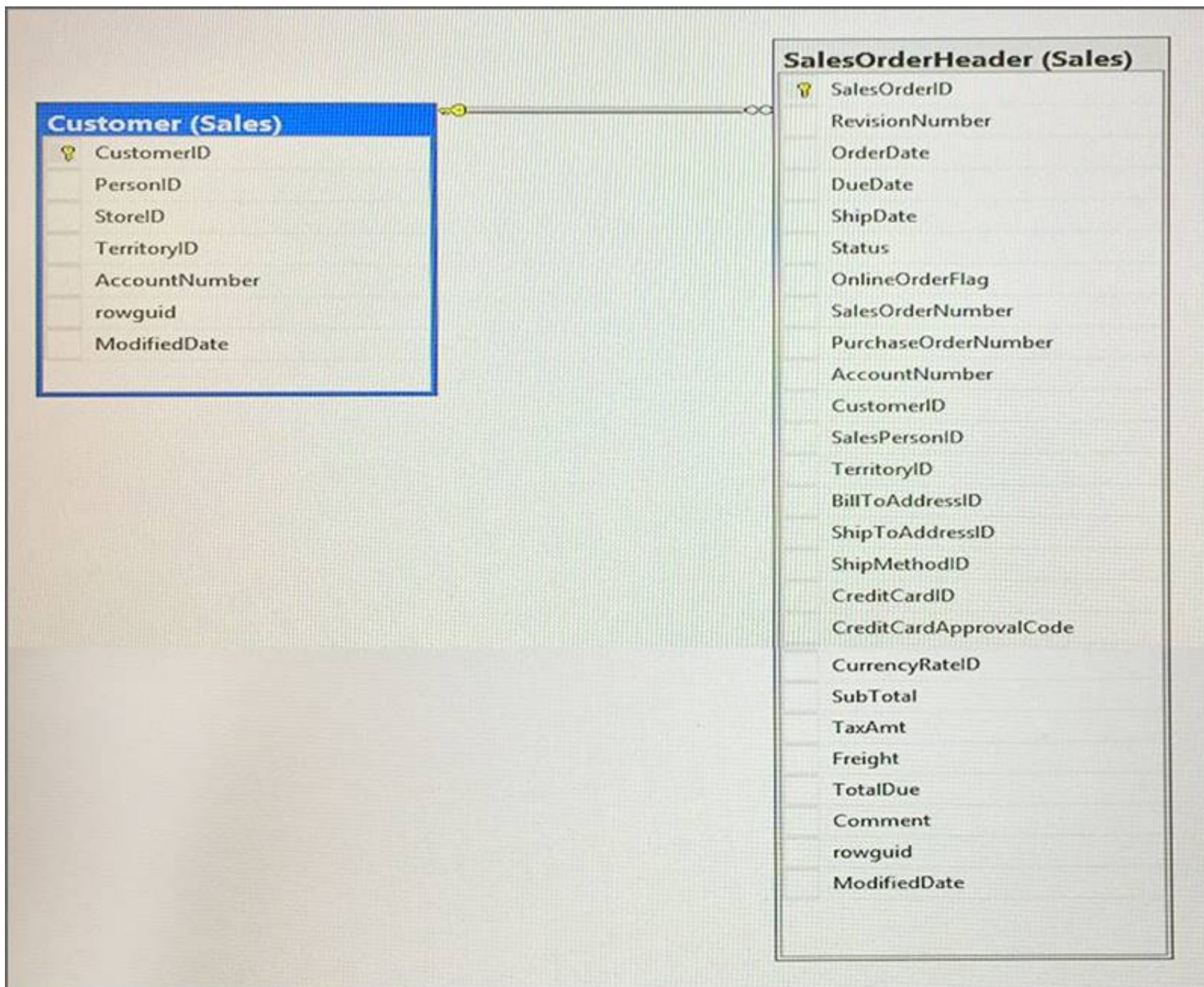
Does the solution meet the goal?

- A. YES
- B. NO

Answer: B

NEW QUESTION 2

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 3

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

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

You are 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 4

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


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

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

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

If the credit limit is zero, you must delete the customer record while fetching data. You need to add the DELETE statement.

Solution: You add the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. YES
- B. NO

Answer: B

Explanation:

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

NEW QUESTION 6

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

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

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

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

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

Answer: E

Explanation:

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

References: https://www.w3schools.com/sql/sql_join_right.asp

NEW QUESTION 7

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 8

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 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 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 10

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

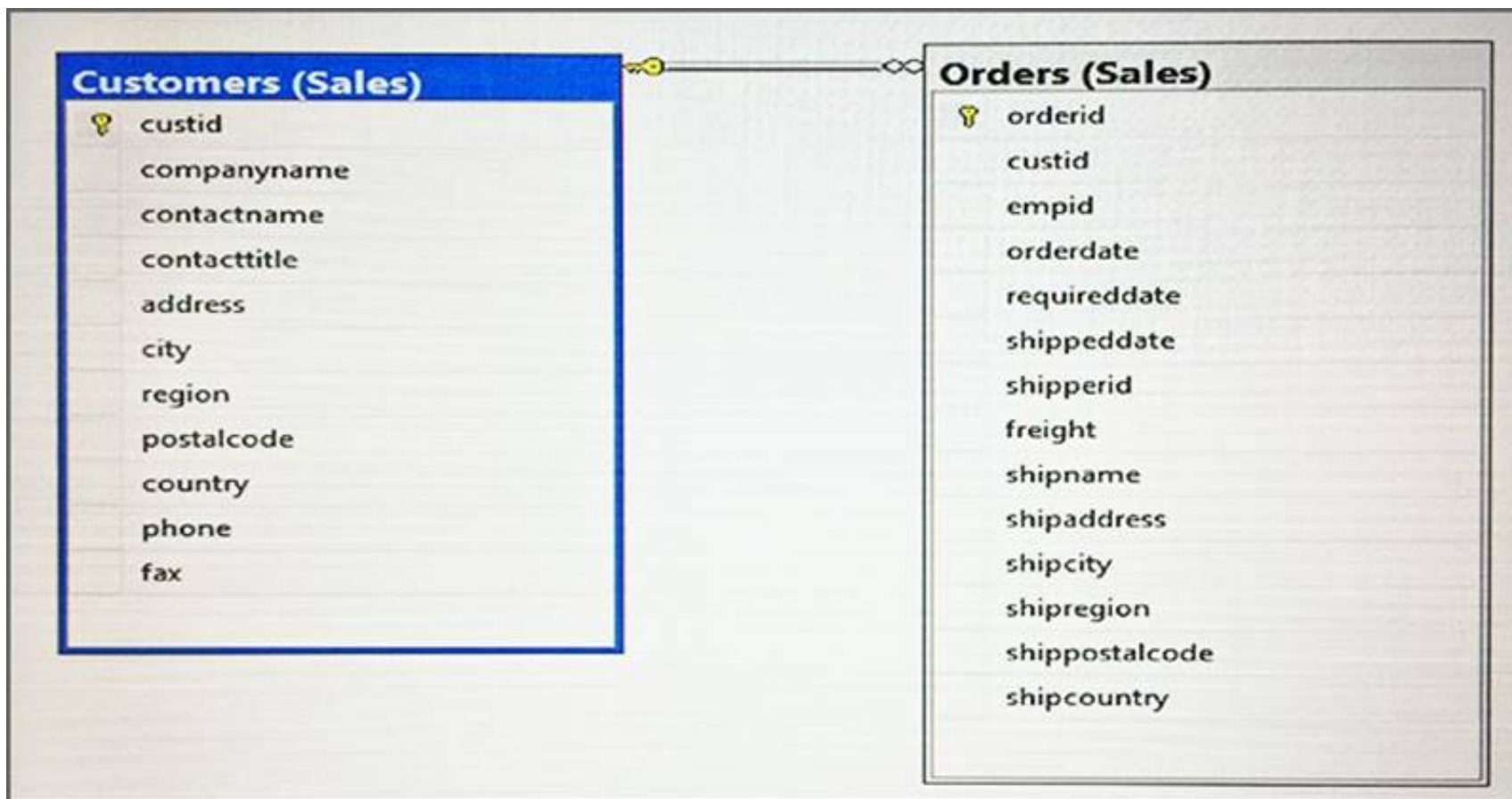
Explanation:

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 10

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	Answer Area
GROUP BY c.custid	
FROM sales.Customers AS c INNER JOIN sales.Orders AS o	
ON c.orderid = o.orderid	
SELECT c.custid AS CustomerID, MAX(o.orderdate) AS LastOrderDate	
FROM sales.Customers AS c LEFT OUTER JOIN sales.Orders AS o	
GROUP BY LastOrderDate	
ON c.custid = o.custid	
SELECT c.custid AS CustomerID, COALESCE (MAX(o.orderdate), '19000101') AS LastOrderDate	

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Box 1: SELECT..COALESCE...

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

Box 2: ..LEFT OUTER JOIN..

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

Box 3: ON c.custid = o.custid

We JOIN on the custID column, which is available in both tables. Box 4: GROUP BY c.custid

References:

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

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


NEW QUESTION 15


Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question

presents a different goal and answer choices, but the text of the scenario is exactly the same in each question in this series.

Start of repeated scenario

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

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

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

You review the Employee table and make the following observations:

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

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

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

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

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

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

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

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

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

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

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

End of Repeated Scenario

You need to create a query to return the data for the Sales Summary report.

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

Transact-SQL segments

Answer Area

```
SalesQuarter_cte (SalesYear, SalesQuarter,
QuarterSalesAmount)
AS
(
    SELECT SalesYear, SalesQuarter, SUM
(SalesAmount) QuarterSalesAmount
    FROM dbo.SalesSummary
    GROUP BY SalesYear, SalesQuarter
)
```

```
SELECT y.SalesYear, q.SalesQuarter,
y.YearSalesAmount, q.QuarterSalesAmount
FROM SalesYear_cte y
INNER JOIN SalesQuarter_cte q
ON y.SalesYear = q.SalesYear;
```

```
SELECT SalesYear, 0 AS SalesQuarter, SUM
(SalesAmount) YearSalesAmount, 0
QuarterSalesAmount
FROM dbo.SalesSummary
GROUP BY SalesYear
```

```
SELECT SalesYear, SalesQuarter, 0
YearSalesAmount, SUM(SalesAmount)
QuarterSalesAmount
FROM dbo.SalesSummary
GROUP BY SalesYear, SalesQuarter
```

```
WITH SalesYear_cte (SalesYear, SalesQuarter,
QuarterSalesAmount, YearSalesAmount )
AS
(
    SELECT SalesYear, SalesQuarter, 0
QuarterSalesAmount, SUM (SalesAmount)
YearSalesAmount
    FROM dbo.SalesSummary
    GROUP BY SalesYear, SalesQuarter
),
```

UNION ALL

```
WITH SalesYear_cte (SalesYear, YearSalesAmount)
AS
(
    SELECT SalesYear, SUM (SalesAmount)
YearSalesAmount
    FROM dbo.SalesSummary
    GROUP BY SalesYear
),
```



- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Use two CTE expressions, one for salesYear and one for SalesQuarter, and combine them with a SELECT statement.

Note: 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.

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 16

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 17

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 18

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 23

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.)





Sales.Customers	
Columns	
	custid (PK, int, 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		

- A. Mastered
- B. Not Mastered

Answer: A

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)	IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID	UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID
BEGIN THROW 55555, 'The customer ID does not exist', 1 END	IF NOT EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
UPDATE Sales.Customers SET companyname = @custID WHERE custid = @newname	BEGIN THROW 55555, 'The customer ID does not exist', 1 END
IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)	
ROLLBACK TRANSACTION	

NEW QUESTION 25

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 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 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

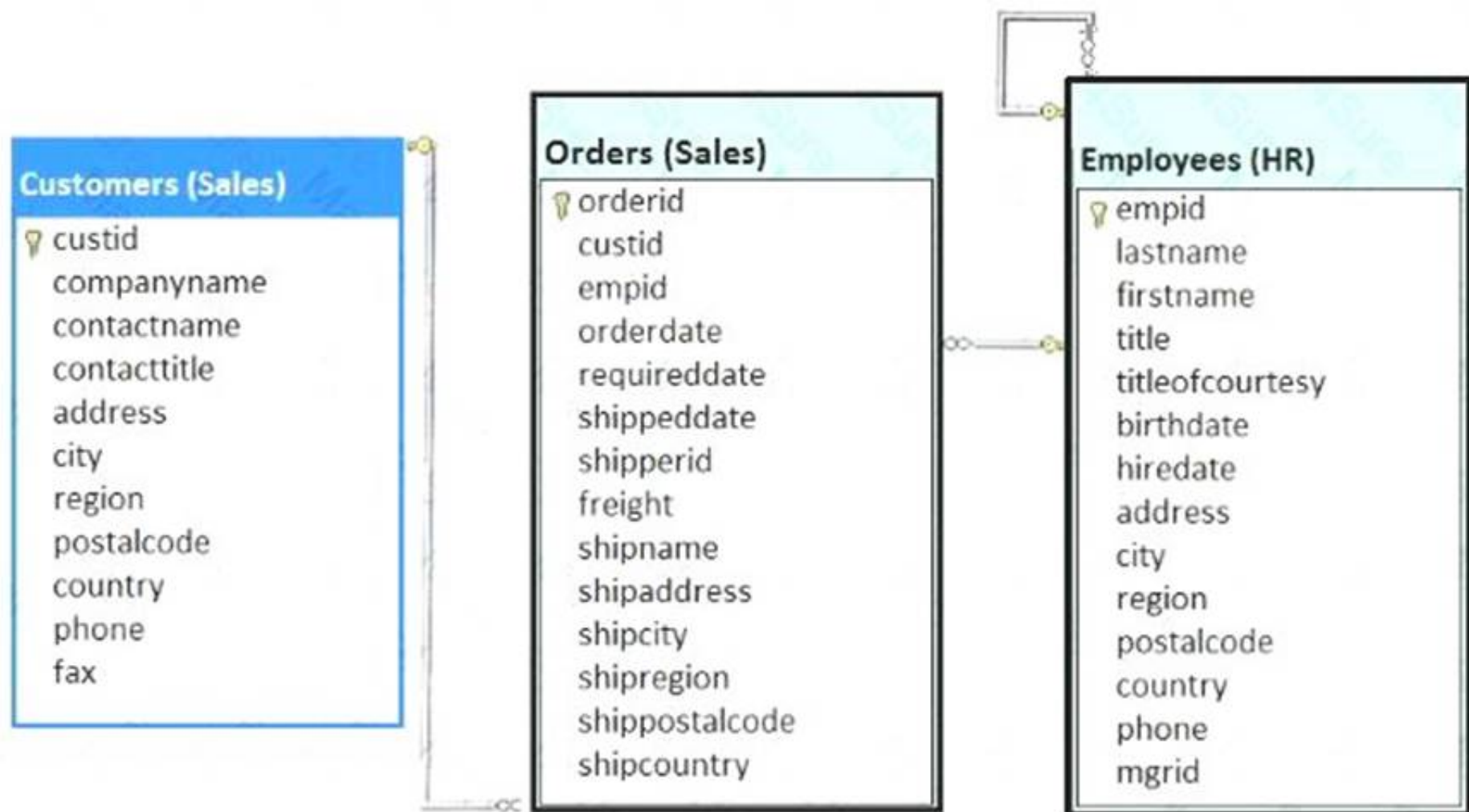
Answer: B

NEW QUESTION 32

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 33

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

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

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

The following table describes the columns in Sales.Orders.

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

The following table describes the columns in Sales.OrderLines.

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

You need to create a stored procedure that inserts data into the Customers table. The stored procedure must meet the following requirements:

- Data changes occur as a single unit of work.
- Data modifications that are successful are committed and a value of 0 is returned.
- Data modifications that are unsuccessful are rolled back. The exception severity level is set to 16 and a value of -1 is returned.
- The stored procedure uses a built-in scalar function to evaluate the current condition of data modifications.
- The entire unit of work is terminated and rolled back if a run-time error occurs during execution of the stored procedure.

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

NOTE: Each correct selection is worth one point.

Transact-SQL segments

Answer Area

RAISERROR

THROW

XACT_ABORT

XACT_STATE

@@TRANCOUNT

ROLLBACK

COMMIT

END

```
CREATE PROCEDURE Sales.InsertCustomer
    @CustomerName nvarchar(100),
    @PhoneNumber nvarchar(20),
    @AccountOpenedDate date,
    @StandardDiscountPercentage decimal(18,3),
    @CreditLimit decimal(18,2),
    @IsCreditOnHold bit,
    @DeliveryLongitude nvarchar(50),
    @DeliveryLatitude nvarchar(50)
AS
BEGIN
    SET NOCOUNT ON
    SET  ON

    BEGIN TRY
        BEGIN TRANSACTION
        INSERT INTO Sales.Customers (CustomerName, PhoneNumber, AccountOpenedDate,
            StandardDiscountPercentage, CreditLimit, IsOnCreditHold, DeliveryLocation)
        VALUES
            (@CustomerName, @PhoneNumber, @AccountOpenedDate, @StandardDiscountPercentage,
            @CreditLimit, @IsCreditOnHold, geography::Point(ISNULL(@DeliveryLongitude, ''),
            ISNULL(@DeliveryLatitude, ''), 4326))

         TRANSACTION
    END TRY
    BEGIN CATCH
        IF  () <> 0  TRANSACTION

        PRINT 'Unable to create the customer record.'
        

        RETURN -1
    END CATCH
    RETURN 0
END
```

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Box 1: XACT_ABORT

XACT_ABORT specifies whether SQL Server automatically rolls back the current transaction when a Transact-SQL statement raises a run-time error. When SET XACT_ABORT is ON, if a Transact-SQL statement raises a run-time error, the entire transaction is terminated and rolled back.

Box 2: COMMIT

Commit the transaction. Box 3: XACT_STATE

Box 4: ROLLBACK

Rollback the transaction Box 5: THROW

THROW raises an exception and the severity is set to 16.

Requirement: Data modifications that are unsuccessful are rolled back. The exception severity level is set to 16 and a value of -1 is returned.

References:

<https://msdn.microsoft.com/en-us/library/ms188792.aspx> <https://msdn.microsoft.com/en-us/library/ee677615.aspx>

NEW QUESTION 35

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 36

You have a database containing the following tables: Servers

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

Errors

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

You have a user-defined, scalar function named IPLookup that takes a DNS name as a parameter and returns

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

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

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

You need to insert data by using the view.

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

Transact-SQL segments

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

Answer Area

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

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

NEW QUESTION 39

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

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

You 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 42

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 43

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

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

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

Sales.Customers

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

Application.Cities

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

Sales.CustomerCategories

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

The company's development team is designing a customer directory application. The application must list customers by the area code of their phone number. The area code is defined as the first three characters of the phone number.

The main page of the application will be based on an indexed view that contains the area and phone number for all customers.

You need to return the area code from the PhoneNumber field. Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

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

NEW QUESTION 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.

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 49

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

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

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

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

Solution: You recommend the following query:

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

Does this meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 52

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

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

You are creating indexes in a data warehouse.

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

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

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

Solution: You create a nonclustered index on the primary key column that does NOT include columns. Does this meet the goal?

- A. YES
- B. NO

Answer: A

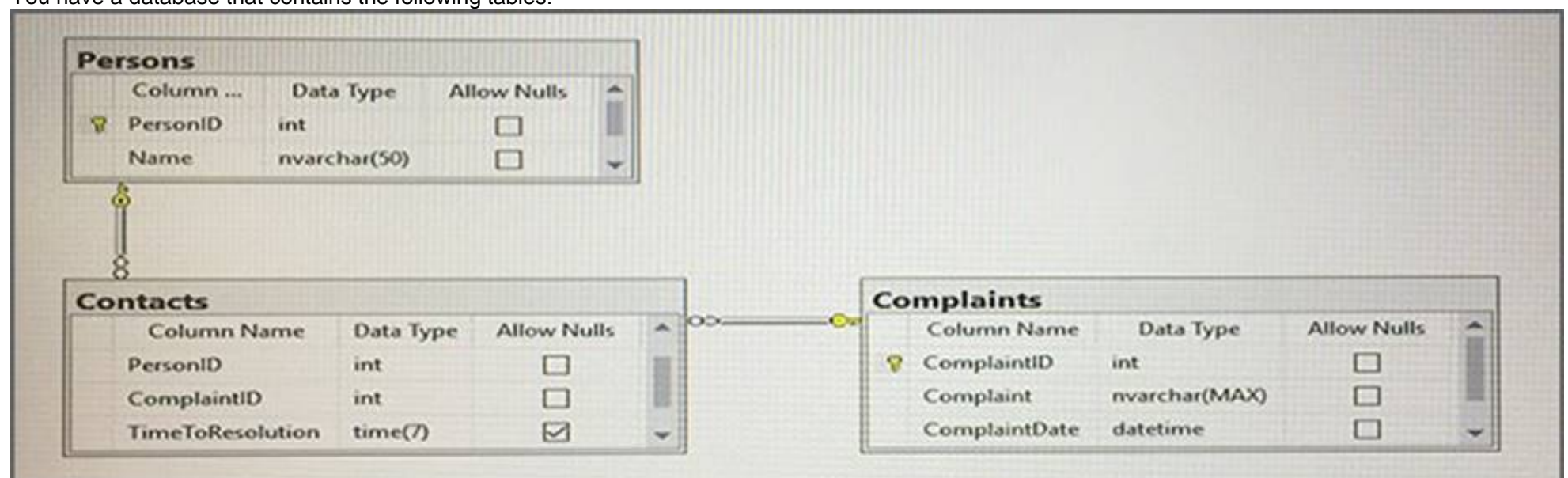
Explanation:

References:

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

NEW QUESTION 54

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.

- A. Mastered
- B. Not Mastered

Answer: A

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 58

You have a table that was created by running the following Transact-SQL statement:

```

CREATE TABLE Courses (
    CourseID INT IDENTITY(1,1) NOT NULL,
    Course VARCHAR(50) NULL)

```

You need to query the Courses table and return the result set as JSON. The output from the query must resemble the following format:

```

{
  "Courses":
  [
    {
      "Course ID":1,
      "Name":"Database Development"
    },
    {
      "Course ID":2,
      "Name":"Programming in C#"
    }
  ]
}

```

- A. SELECT CourseID AS [Course ID], Course as Name FROM CoursesFOR JSON PATH('Courses')
- B. SELECT CourseID AS 'Course ID', Course AS Name FROM CoursesFOR JSON ROOT('Courses')
- C. SELECT CourseID AS [Course ID], Course AS Name FROM CoursesFOR JSON AUTO, ROOT('Courses')

D. SELECT CourseID AS 'Course ID', Course AS Name FROM CoursesFOR JSON AUTO, INCLUDE_NULL_VALUES('Courses')

Answer: D

Explanation:

References:

<https://docs.microsoft.com/en-us/sql/relational-databases/json/include-null-values-in-json-include-null-values-op>

NEW QUESTION 60

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%'

- A. Mastered
- B. Not Mastered

Answer: A

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 63

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 if 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 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 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 73**

You have two tables named UserLogin and Employee respectively.

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

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

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

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Code segments

```
BEGIN TRY
    UPDATE dbo.Employee
    SET IsDeleted = 1
    WHERE Id = 1
END TRY
```

```
BEGIN CATCH
    RAISERROR ('No tables updated!',
16, 1)
END CATCH
```

```
UPDATE dbo.Employee
SET IsDeleted = 1
WHERE Id = 1
```

```
BEGIN CATCH
```

```
BEGIN TRY
    UPDATE dbo.UserLogin
    SET IsDeleted = 1
    WHERE Id = 1
END TRY
```

```
END CATCH
```

```
BEGIN TRY
    UPDATE dbo.UserLogin
    SET IsDeleted = 1
    WHERE Id = 1
    UPDATE dbo.Employee
    SET IsDeleted = 1
    WHERE Id = 1
END TRY
```

Answer Area

```
BEGIN TRY
    UPDATE dbo.UserLogin
    SET IsDeleted = 1
    WHERE Id = 1
END TRY
```

```
BEGIN CATCH
```

```
BEGIN TRY
    UPDATE dbo.Employee
    SET IsDeleted = 1
    WHERE Id = 1
END TRY
```

```
BEGIN CATCH
    RAISERROR ('No tables updated!',
16, 1)
END CATCH
```

```
END CATCH
```

NEW QUESTION 78

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
WHERE CustomerId = 3
INSERT INTO dbo.CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
SELECT CustomerId, CreditLimit, CreditLimit
FROM Customer
WHERE CustomerId = 3
```
- B**
- ```
UPDATE Customer
SET CreditLimit = 1000
WHERE CustomerId = 3
INSERT INTO dbo.CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
SELECT CustomerId, CreditLimit, CreditLimit
FROM Customer
```
- C**
- ```
UPDATE Customer
SET CreditLimit = 1000
OUTPUT inserted.CustomerId, inserted.CreditLimit, deleted.CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
WHERE CustomerId = 3
```
- D**
- ```
UPDATE Customer
SET CreditLimit = 1000
OUTPUT inserted.CustomerId, deleted.CreditLimit, inserted.CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
WHERE CustomerId = 3
```

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

Answer: D

Explanation:

The OUTPUT Clause returns information from, or expressions based on, each row affected by an INSERT, UPDATE, DELETE, or MERGE statement. These results can be returned to the processing application for use in such things as confirmation messages, archiving, and other such application requirements. The results can also be inserted into a table or table variable. Additionally, you can capture the results of an OUTPUT clause in a nested INSERT, UPDATE, DELETE, or MERGE statement, and insert those results into a target table or view.

Note: If the column modified by the .RITE clause is referenced in an OUTPUT clause, the complete value of the column, either the before image in deleted.column_name or the after image in inserted.column_name, is returned to the specified column in the table variable.

NEW QUESTION 81

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,
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	Answer Area
CAST(Cust.AccountOpenedDate AS DATE)	SELECT Count(Cust.CustomerId), CustCat.CustomerCategoryName, CAST(Cust.AccountOpenedDate AS DATE)
DATEPART(day, Cust.AccountOpenedDate)	FROM Sales.Customers AS Cust
HAVING	INNER JOIN Sales.CustomerCategories AS CustCat
WHERE	ON Cust.CustomerCategoryId = CustCat.CustomerCategoryId
COUNT(Cust.CustomerId)	GROUP BY CustCat.CustomerCategoryName, CAST(Cust.AccountOpenedDate AS DATE)
MAX(Cust.CustomerID)	WHERE COUNT(Cust.CustomerId) > 5
RANK	
GROUP BY	

NEW QUESTION 85

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 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: F

Explanation:

In a FOR XML clause, you specify one of these modes: RAW, AUTO, EXPLICIT, and PATH.

* The EXPLICIT mode allows more control over the shape of the XML. You can mix attributes and elements at will in deciding the shape of the XML. It requires a specific format for the resulting rowset that is generated because of query execution. This rowset format is then mapped into XML shape. The power of EXPLICIT mode is to mix attributes and elements at will, create wrappers and nested complex properties, create space-separated values (for example, OrderID attribute may have a list of order ID values), and mixed contents.

* The PATH mode together with the nested FOR XML query capability provides the flexibility of the EXPLICIT mode in a simpler manner.

References: <https://msdn.microsoft.com/en-us/library/ms178107.aspx>

NEW QUESTION 86

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, TownID, CreatedDate)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000, NULL, GETDATE())
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, TownID, CreatedDate)
VALUES ('Jossef', 'Goldberg', '1995-06-03', 5500, NULL, 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 89

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

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

The following table describes the columns in Sales.Orders.

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

The following table describes the columns in Sales.OrderLines.

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

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

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

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

Transact-SQL segments

COUNT

SUM

AVG

ORDER BY

GROUP BY

RETURNS INT

RETURNS NULL ON NULL INPUT

RETURNS TABLE

Answer Area

```

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

- A. Mastered
B. Not Mastered

Answer: A

Explanation:

Box1: RETURNS TABLE

The function should return the following information:

- all customer information for the customer
 - the total number of orders for the customer
 - the total price of all orders for the customer
 - the average quantity of items per order
- Box 2: COUNT

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

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

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

Need to use GROUP BY for the aggregate functions.

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

NEW QUESTION 94

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

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

You are creating indexes in a data warehouse.

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

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

You need to reduce the amount of time it takes to run the reports. Solution: You create a clustered index on the primary key column. Does this meet the goal?

- A. Yes
- B. No

Answer: A

NEW QUESTION 96

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 99

.....

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>