➢ Vendor: Microsoft

➢ Exam Code: 70-761

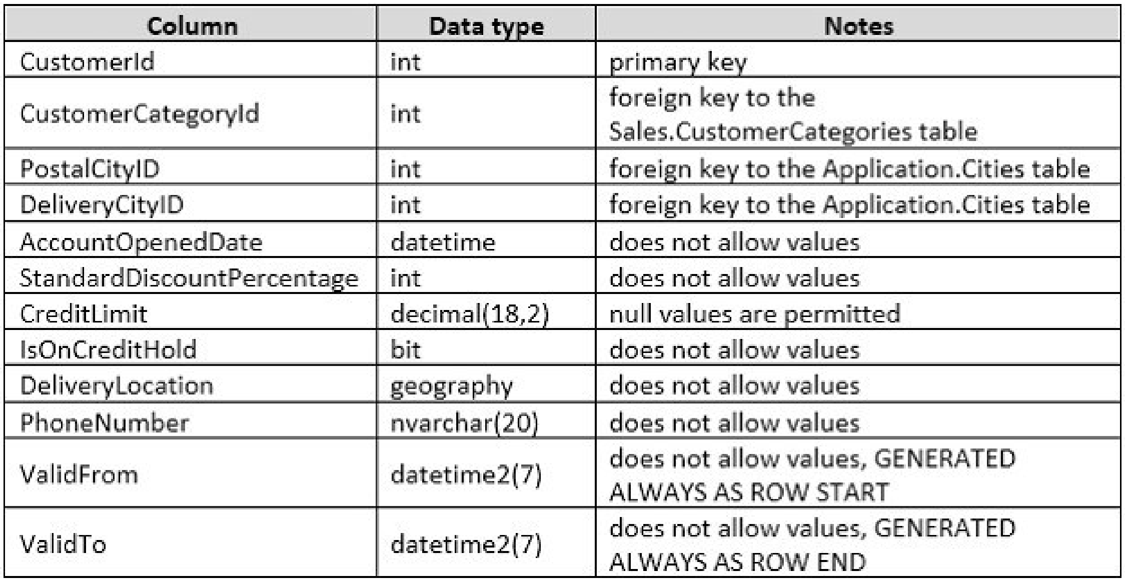
➢ Exam Name: Querying Data with Transact-SQL

➢ Question 31 – Question 40

QUESTION 31

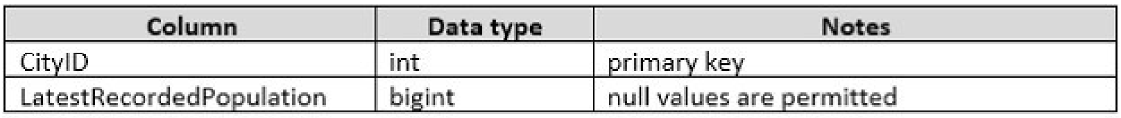
Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question presents a different goal and answer choices, but the text of the scenario is exactly the same in each question on this series.

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



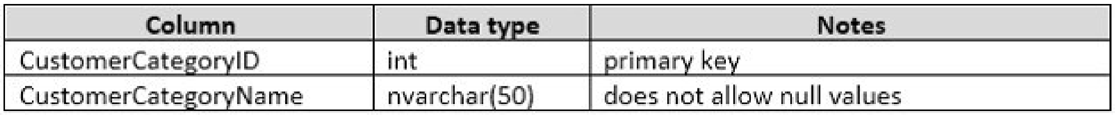
Column	Data type	Notes
CustomerId	int	primary key
CustomerCategoryId	int	foreign key to the Sales.CustomerCategories table
PostalCityID	int	foreign key to the Application.Cities table
DeliveryCityID	int	foreign key to the Application.Cities table
AccountOpenedDate	datetime	does not allow values
StandardDiscountPercentage	int	does not allow values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow values
DeliveryLocation	geography	does not allow values
PhoneNumber	nvarchar(20)	does not allow values
ValidFrom	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW START
ValidTo	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW END

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



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

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

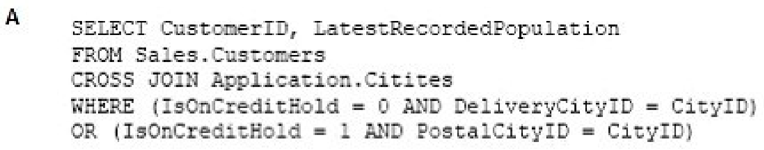


Column	Data type	Notes
CustomerCategoryID	int	primary key
CustomerCategoryName	nvarchar(50)	does not allow null values

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

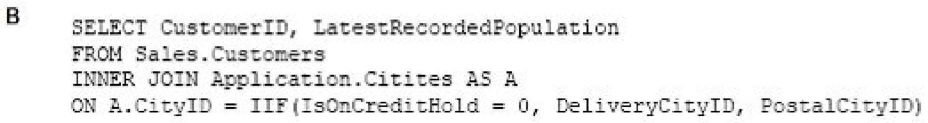
```
- For customers that are not on a credit hold, return the CustomerID and
the latest recorded population for the delivery city that is associated
with the customer.
- For customers that are on a credit hold, return the CustomerID and the
latest recorded population for the postal city that is associated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

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



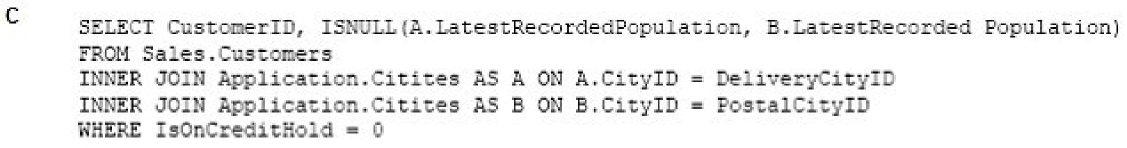
A

```
SELECT CustomerID, LatestRecordedPopulation
FROM Sales.Customers
CROSS JOIN Application.Citites
WHERE (IsOnCreditHold = 0 AND DeliveryCityID = CityID)
OR (IsOnCreditHold = 1 AND PostalCityID = CityID)
```



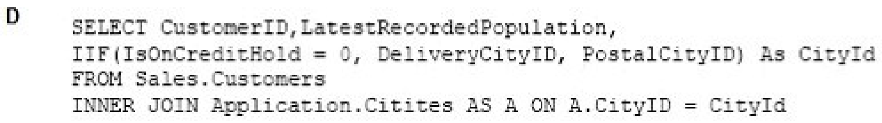
B

```
SELECT CustomerID, LatestRecordedPopulation
FROM Sales.Customers
INNER JOIN Application.Citites AS A
ON A.CityID = IIF(IsOnCreditHold = 0, DeliveryCityID, PostalCityID)
```



C

```
SELECT CustomerID, ISNULL(A.LatestRecordedPopulation, B.LatestRecorded Population)
FROM Sales.Customers
INNER JOIN Application.Citites AS A ON A.CityID = DeliveryCityID
INNER JOIN Application.Citites AS B ON B.CityID = PostalCityID
WHERE IsOnCreditHold = 0
```



D

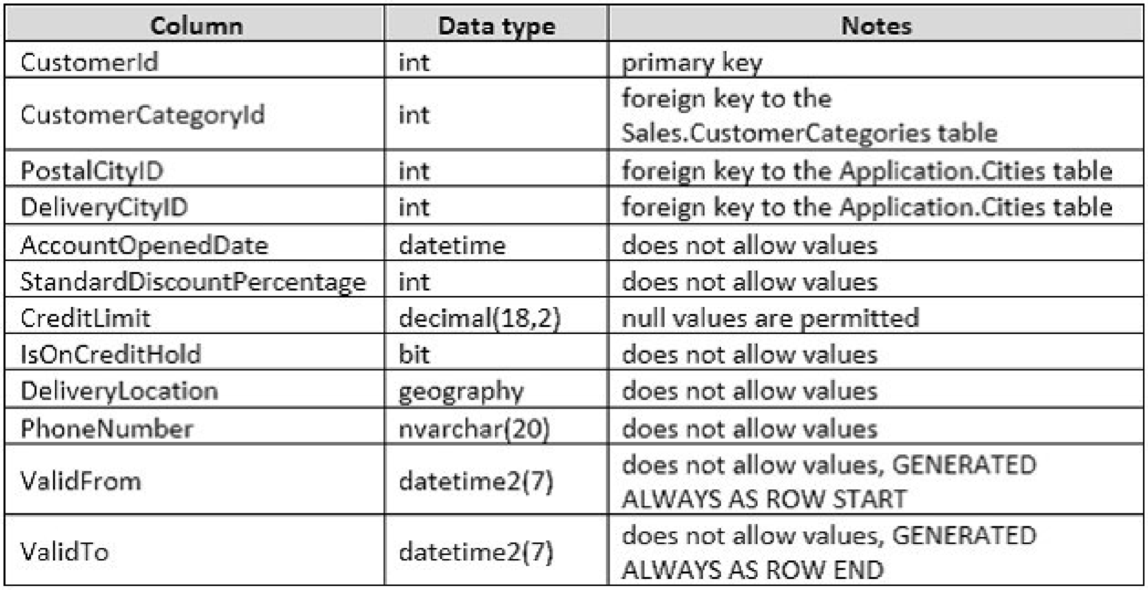
```
SELECT CustomerID,LatestRecordedPopulation,
IIF(IsOnCreditHold = 0, DeliveryCityID, PostalCityID) As CityId
FROM Sales.Customers
INNER JOIN Application.Citites AS A ON A.CityID = CityId
```

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

Answer: AB
Explanation:
Using Cross Joins

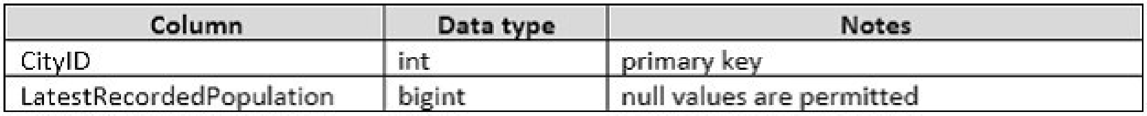
A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table. However, if a WHERE clause is added, the cross join behaves as an inner join.
B: You can use the IIF in the ON-statement.
IIF returns one of two values, depending on whether the Boolean expression evaluates to true or false in SQL Server.
References:
https://technet.microsoft.com/en-us/library/ms190690(v=sql.105).aspx
https://msdn.microsoft.com/en-us/library/hh213574.aspx

QUESTION 32
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:



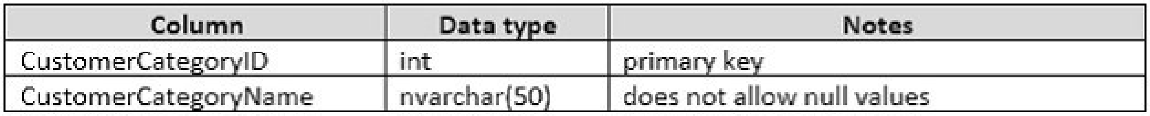
Column	Data type	Notes
CustomerId	int	primary key
CustomerCategoryId	int	foreign key to the Sales.CustomerCategories table
PostalCityID	int	foreign key to the Application.Cities table
DeliveryCityID	int	foreign key to the Application.Cities table
AccountOpenedDate	datetime	does not allow values
StandardDiscountPercentage	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:



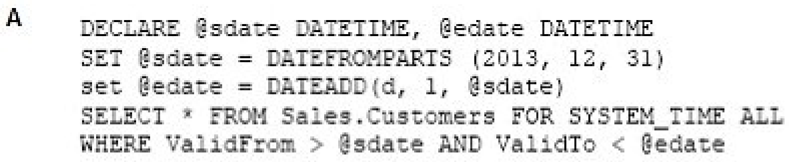
Column	Data type	Notes
CityID	int	primary key
LatestRecordedPopulation	bigint	null values are permitted

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



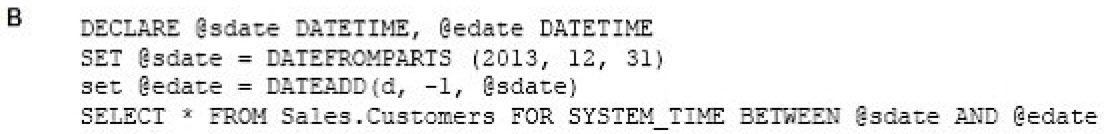
Column	Data type	Notes
CustomerCategoryID	int	primary key
CustomerCategoryName	nvarchar(50)	does not allow null values

You discover an application bug that impacts customer data for records created on or after January 1, 2014. In order to fix the data impacted by the bug, application programmers require a report that contains customer data as it existed on December 31 , 2013. You need to provide the query for the report. Which Transact-SQL statement should you use?



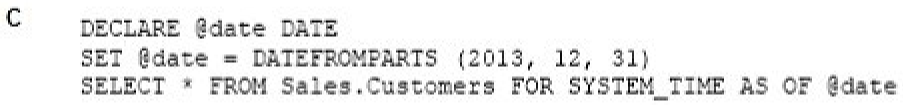
A

```
DECLARE @sdate DATETIME, @edate DATETIME
SET @sdate = DATEFROMPARTS (2013, 12, 31)
set @edate = DATEADD(d, 1, @sdate)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME ALL
WHERE ValidFrom > @sdate AND ValidTo < @edate
```



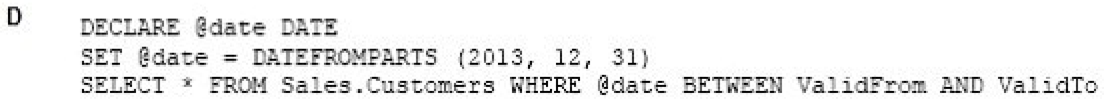
B

```
DECLARE @sdate DATETIME, @edate DATETIME
SET @sdate = DATEFROMPARTS (2013, 12, 31)
set @edate = DATEADD(d, -1, @sdate)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME BETWEEN @sdate AND @edate
```



C

```
DECLARE @date DATE
SET @date = DATEFROMPARTS (2013, 12, 31)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME AS OF @date
```



D

```
DECLARE @date DATE
SET @date = DATEFROMPARTS (2013, 12, 31)
SELECT * FROM Sales.Customers WHERE @date BETWEEN ValidFrom AND ValidTo
```

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

Answer: D
Explanation:
The datetime datetype defines a date that is combined with a time of day with fractional seconds that is based on a 24-hour clock. The DATEFROMPARTS function returns a date value for the specified year, month, and day.

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 on this series.
Drag and Drop Question
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

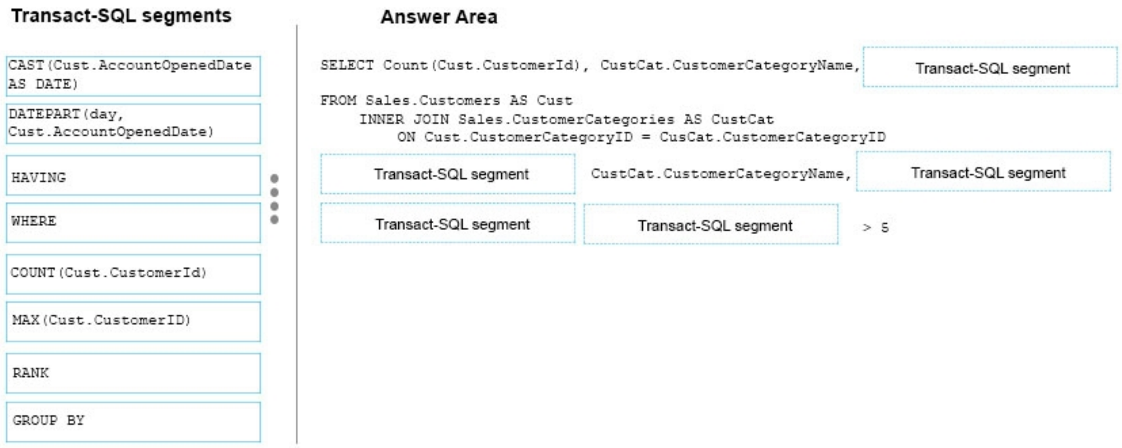
Details for the Application.Cities table are shown in the following table:

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.



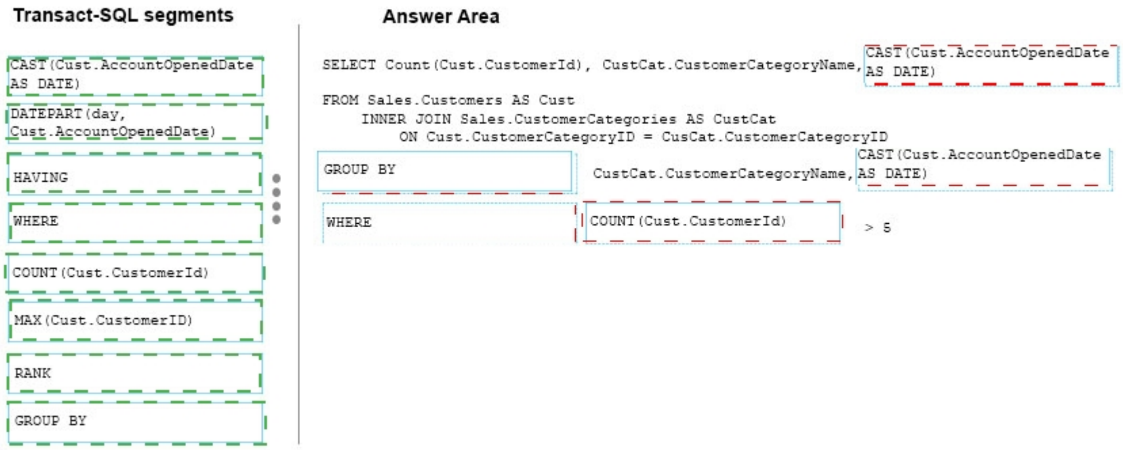
Transact-SQL segments

```
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 = CusCat.CustomerCategoryID
[Transact-SQL segment] CustCat.CustomerCategoryName, [Transact-SQL segment]
[Transact-SQL segment] [Transact-SQL segment] > 5
```

Answer:



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

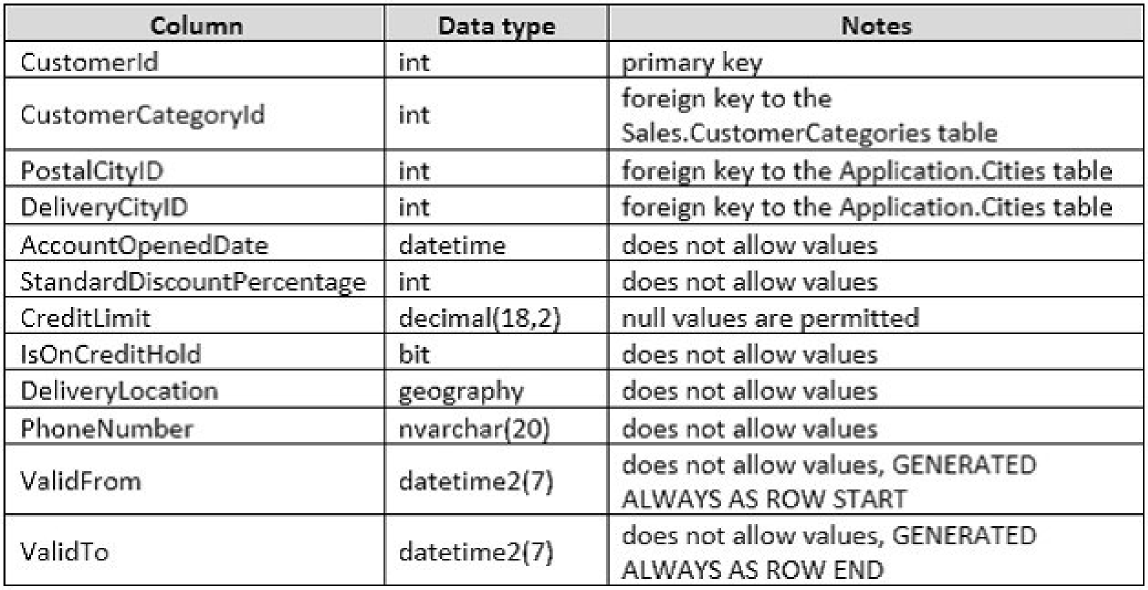
```
SELECT Count(Cust.CustomerId), CustCat.CustomerCategoryName, CAST(Cust.AccountOpenedDate
                                                              AS DATE)
FROM Sales.Customers AS Cust
     INNER JOIN Sales.CustomerCategories AS CustCat
          ON Cust.CustomerCategoryID = CusCat.CustomerCategoryID
GROUP BY        CustCat.CustomerCategoryName, CAST(Cust.AccountOpenedDate
                                              AS DATE)
WHERE           COUNT(Cust.CustomerId)    > 5
```

QUESTION 34

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.

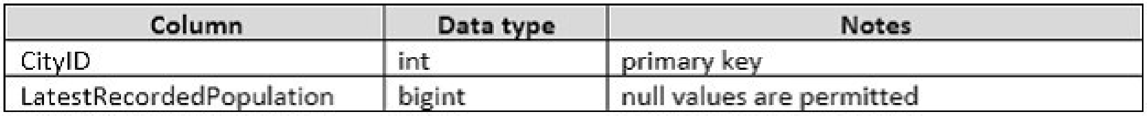
Drag and Drop Question

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



Column	Data type	Notes
CustomerId	int	primary key
CustomerCategoryId	int	foreign key to the Sales.CustomerCategories table
PostalCityID	int	foreign key to the Application.Cities table
DeliveryCityID	int	foreign key to the Application.Cities table
AccountOpenedDate	datetime	does not allow values
StandardDiscountPercentage	int	does not allow values
CreditLimit	decimal(18,2)	null values are permitted
IsOnCreditHold	bit	does not allow values
DeliveryLocation	geography	does not allow values
PhoneNumber	nvarchar(20)	does not allow values
ValidFrom	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW START
ValidTo	datetime2(7)	does not allow values, GENERATED ALWAYS AS ROW END

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

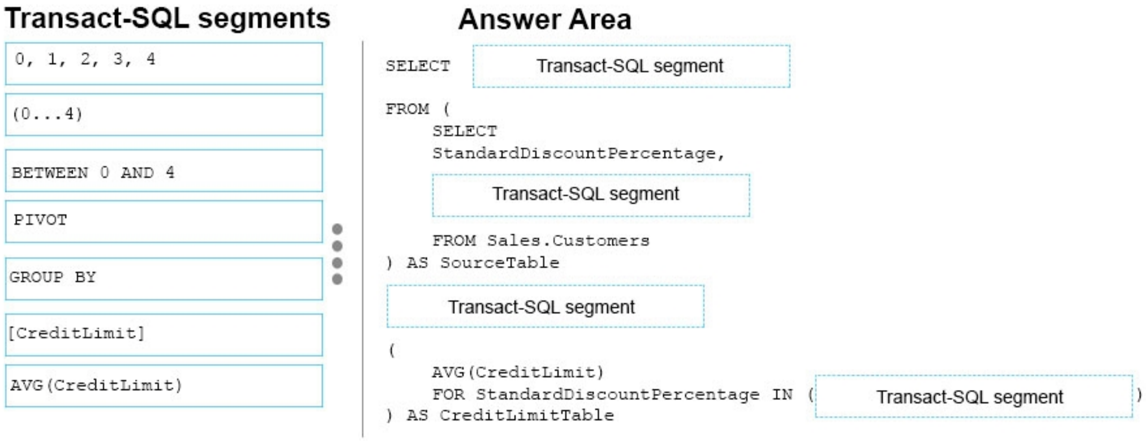


Column	Data type	Notes
CityID	int	primary key
LatestRecordedPopulation	bigint	null values are permitted

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

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

The marketing department is performing an analysis of how discount affect credit limits. They need to know the average credit limit per standard discount percentage for customers whose standard discount percentage is between zero and four. You need to create a query that returns the data for the analysis. How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct locations. Each Transact-SQL segments may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.



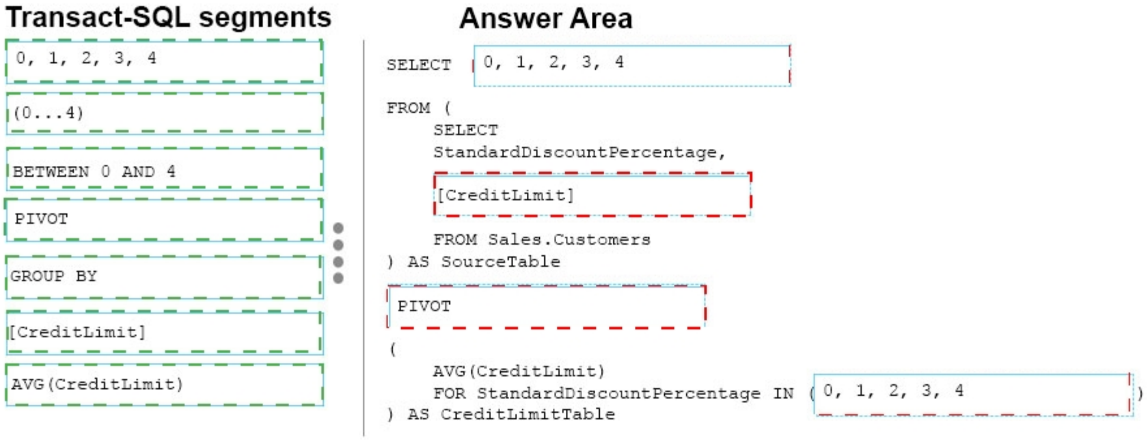
Transact-SQL segments

```
0, 1, 2, 3, 4
(0...4)
BETWEEN 0 AND 4
PIVOT
GROUP BY
[CreditLimit]
AVG(CreditLimit)
```

Answer Area

```
SELECT   Transact-SQL segment
FROM (
      SELECT
      StandardDiscountPercentage,
        Transact-SQL segment
      FROM Sales.Customers
) AS SourceTable
  Transact-SQL segment
(
      AVG(CreditLimit)
      FOR StandardDiscountPercentage IN ( Transact-SQL segment )
) AS CreditLimitTable
```

Answer:



Transact-SQL segments

```
0, 1, 2, 3, 4
(0...4)
BETWEEN 0 AND 4
PIVOT
GROUP BY
[CreditLimit]
AVG(CreditLimit)
```

Answer Area

```
SELECT  0, 1, 2, 3, 4
FROM (
      SELECT
      StandardDiscountPercentage,
      [CreditLimit]
      FROM Sales.Customers
) AS SourceTable
PIVOT
(
      AVG(CreditLimit)
      FOR StandardDiscountPercentage IN ( 0, 1, 2, 3, 4 )
) AS CreditLimitTable
```

Explanation:

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

Pivot example:

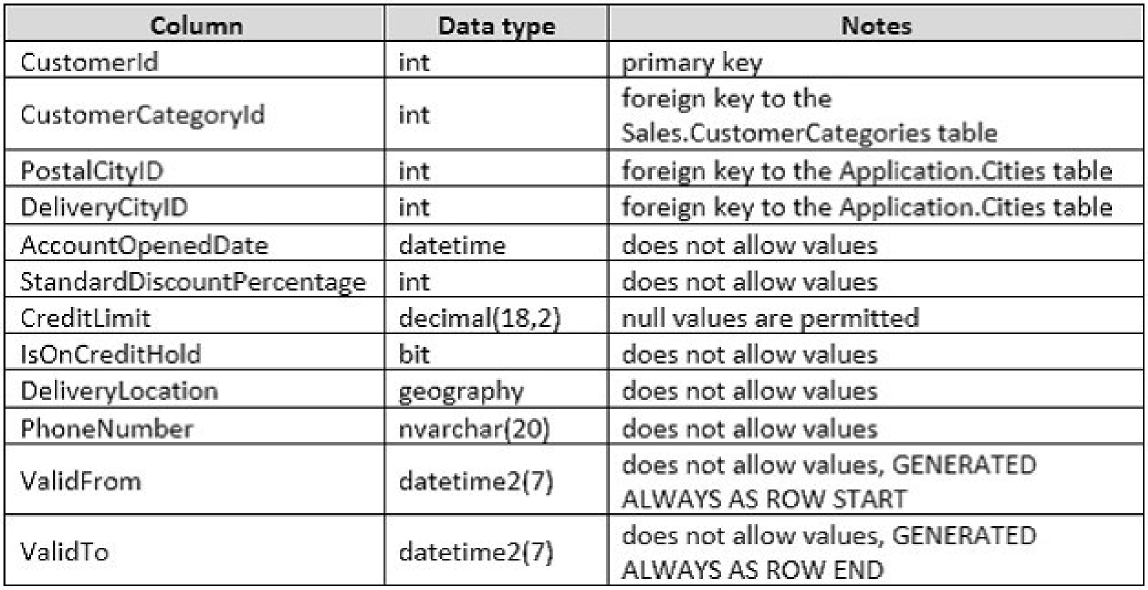
```
-- Pivot table with one row and five columns
SELECT 'AverageCost' AS Cost_Sorted_By_Production_Days, [0], [1], [2], [3], [4]
FROM
(SELECT DaysToManufacture, StandardCost
FROM Production.Product) AS SourceTable
PIVOT
(
AVG(StandardCost)
FOR DaysToManufacture IN ([0], [1], [2], [3], [4]) ) AS PivotTable;
```

Box 2: [CreditLimit]
Box 3: PIVOT
You can use the PIVOT and UNPIVOT relational operators to change a table-valued expression into another table. PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output, and performs aggregations where they are required on any remaining column values that are wanted in the final output.
Box 4: 0, 1, 2, 3, 4
The IN clause determines whether a specified value matches any value in a subquery or a list.
Syntax: test_expression [NOT] IN (subquery | expression [,...n]) Where expression[,... n] is a list of expressions to test for a match. All expressions must be of the same type as test_expression.
References: https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx

QUESTION 35

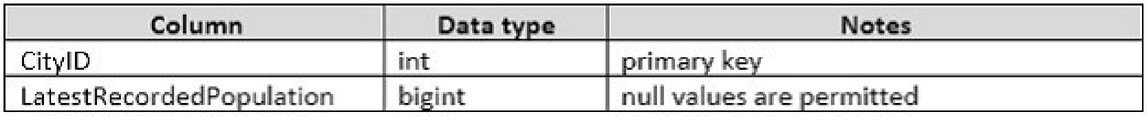
Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question presents a different goal and answer choices, but the text of the scenario is exactly the same in each question on this series.

Drag and Drop Question
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:



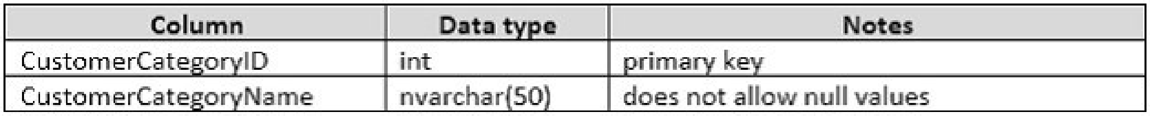
Column	Data type	Notes
CustomerId	int	primary key
CustomerCategoryId	int	foreign key to the Sales.CustomerCategories table
PostalCityID	int	foreign key to the Application.Cities table
DeliveryCityID	int	foreign key to the Application.Cities table
AccountOpenedDate	datetime	does not allow values
StandardDiscountPercentage	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:



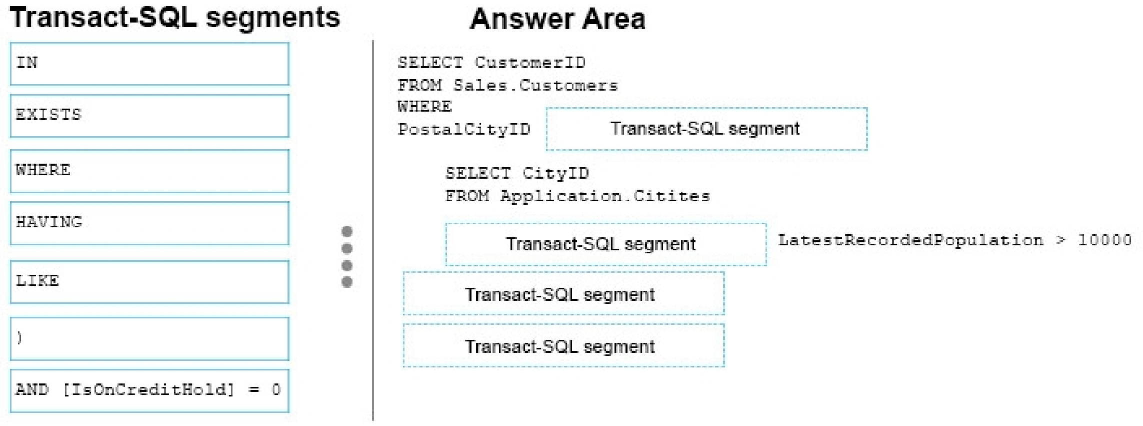
Column	Data type	Notes
CityID	int	primary key
LatestRecordedPopulation	bigint	null values are permitted

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 preparing a promotional mailing. The mailing must only be sent to customers in good standing that live in medium and large cities. You need to write a query that returns all customers that are not on credit hold who live in cities with a population greater than 10,000. 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.



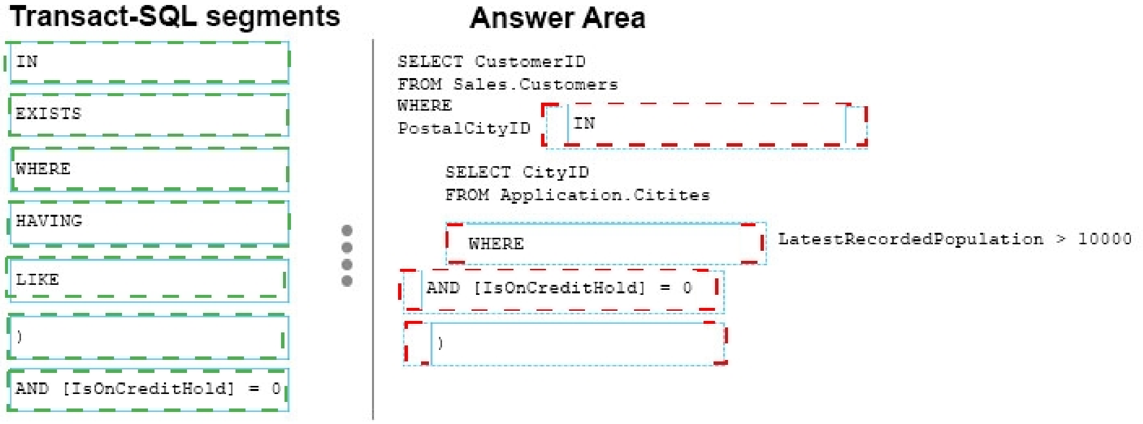
Transact-SQL segments

- IN
- EXISTS
- WHERE
- HAVING
- LIKE
-)
- AND [IsOnCreditHold] = 0

Answer Area

```
SELECT CustomerID
FROM Sales.Customers
WHERE
PostalCityID [Transact-SQL segment]

    SELECT CityID
    FROM Application.Citites

    [Transact-SQL segment] LatestRecordedPopulation > 10000
  [Transact-SQL segment]
  [Transact-SQL segment]
```

Answer:



Transact-SQL segments

- IN
- EXISTS
- WHERE
- HAVING
- LIKE
-)
- AND [IsOnCreditHold] = 0

Answer Area

```
SELECT CustomerID
FROM Sales.Customers
WHERE
PostalCityID [IN]

    SELECT CityID
    FROM Application.Citites

    [WHERE] LatestRecordedPopulation > 10000
  [AND [IsOnCreditHold] = 0]
  [)]
```

Explanation:

Box 1: IN (

The IN clause determines whether a specified value matches any value in a subquery or a list.

Syntax: test_expression [NOT] IN (subquery | expression [,...n]) Where subquery

Is a subquery that has a result set of one column. This column must have the same data type as test_expression.

Box 2: WHERE

Box 3: AND [IsOnCreditHold] = 0

Box 4:)

References: https://msdn.microsoft.com/en-us/library/ms177682.aspx

QUESTION 36

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

You have a table named Products that contains information about the products that your company sells. The table contains many columns that do not always contain values. You need to implement an ANSI standard method to convert the NULL values in the query output to the phrase "Not Applicable". What should you implement?

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

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

QUESTION 37
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

QUESTION 38
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

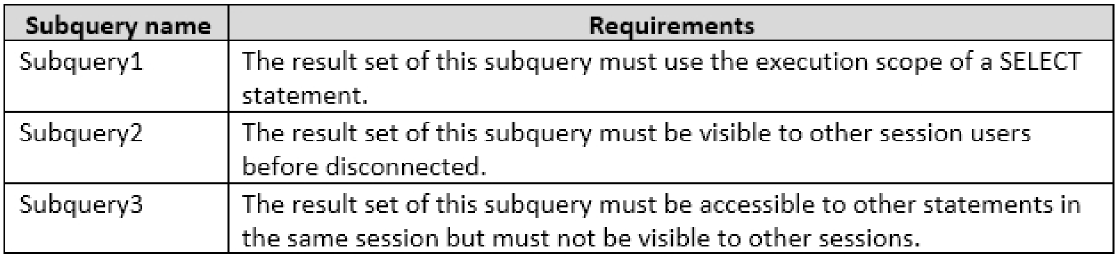
Answer: C
Explanation:
User-defined functions that return a table data type can be powerful alternatives to views. These functions are referred to as table-valued functions. A table-valued user-defined function can be used where table or view expressions are allowed in Transact-SQL queries. While views 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

QUESTION 39
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You have a table named AuditTrail that tracks modifications to data in other tables. The AuditTrail table is updated by many processes. Data input into AuditTrail may contain improperly formatted date time values. You implement a process that retrieves data from the various columns in AuditTrail, but sometimes the process throws an error when it is unable to convert the data into valid date time values. You need to convert the data into a valid date time value using the en-US format culture code. If the conversion fails, a null value must be returned in the column output. The conversion process must not throw 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

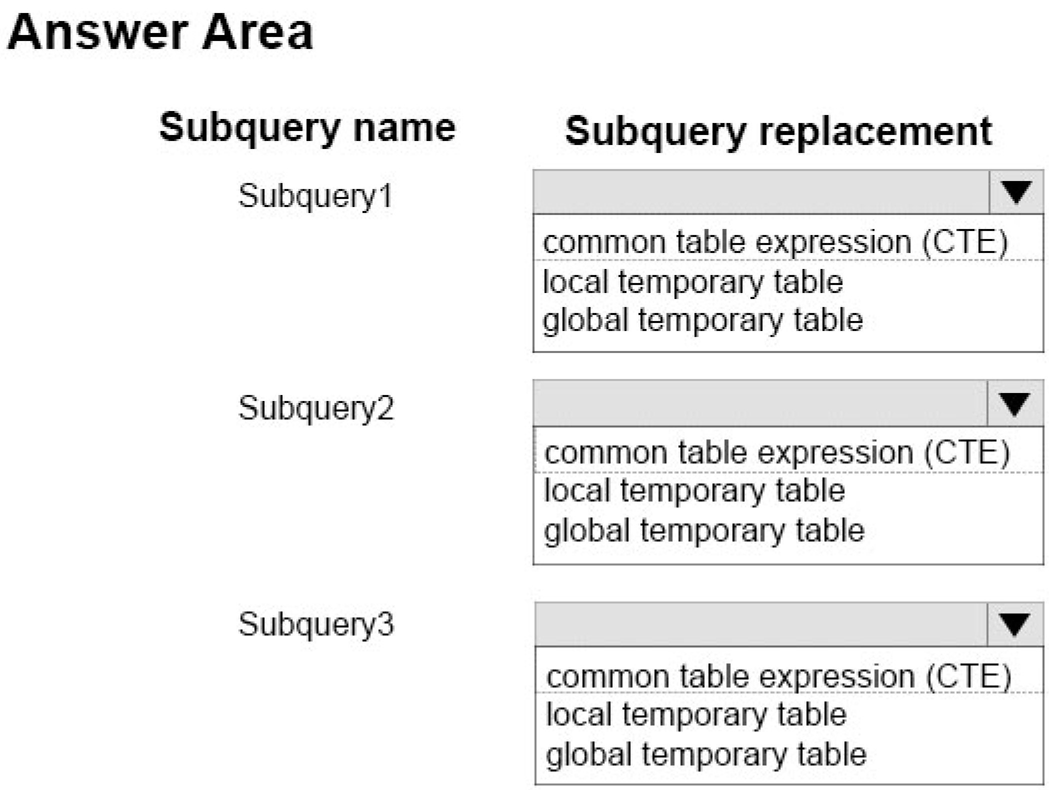
Answer: H
Explanation:
A TRY_CONVERT function returns a value cast to the specified data type if the cast succeeds; otherwise, returns null.
References: https://msdn.microsoft.com/en-us/library/hh230993.aspx

QUESTION 40
Hotspot Question
You have the following subqueries: Subquery1 , Subquery2, and Subquery3. You need to replace the three subqueries with named result sets or temporary tables. The following requirements must be met:



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

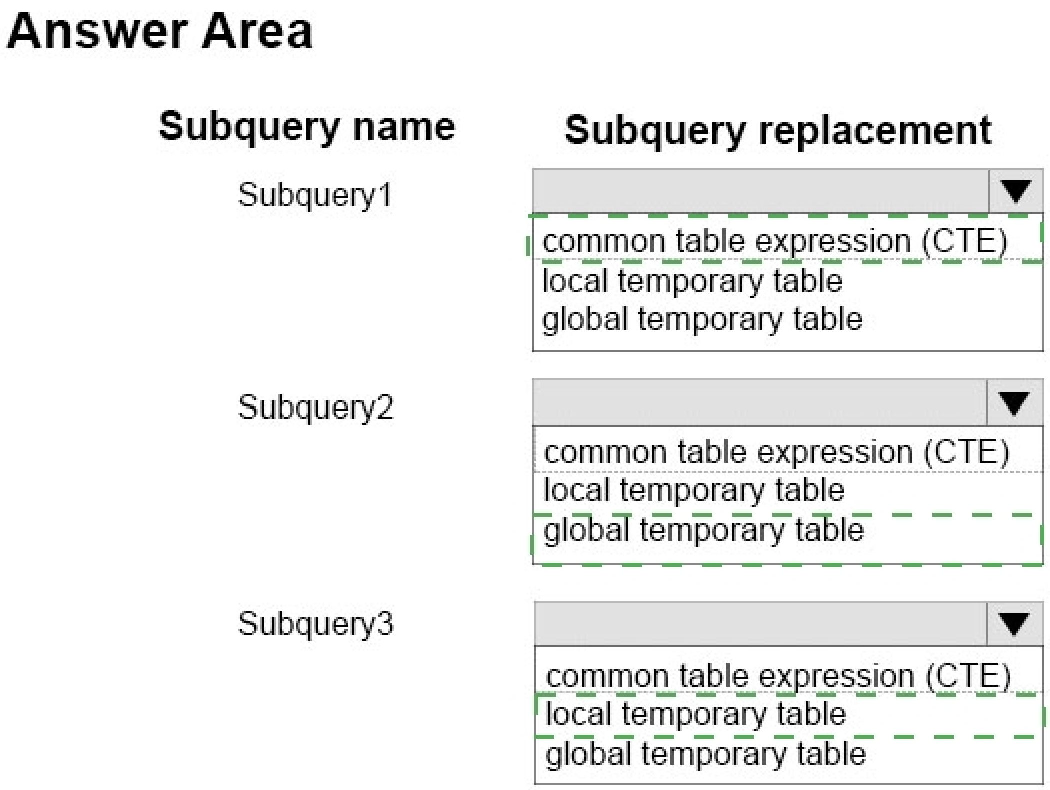
Which replacement techniques should you use? To answer, select the appropriate options in the answer area.



Answer Area

Subquery name	Subquery replacement
Subquery1	common table expression (CTE) local temporary table global temporary table
Subquery2	common table expression (CTE) local temporary table global temporary table
Subquery3	common table expression (CTE) local temporary table global temporary table

Answer:



Answer Area

Subquery name	Subquery replacement
Subquery1	[✓ common table expression (CTE)] local temporary table global temporary table
Subquery2	common table expression (CTE) local temporary table [✓ global temporary table]
Subquery3	common table expression (CTE) [✓ local temporary table] global temporary table

Explanation:
Subquery1: common table expression (CTE)
A common table expression (CTE) can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query. Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.
Subquery2: global temporary table
Global temporary tables are visible to any user and any connection after they are created, and are deleted when all users that are referencing the table disconnect from the instance of SQL Server.
Subquery3: local temporary table
Local temporary tables are visible only to their creators during the same connection to an instance of SQL Server as when the tables were first created or referenced. Local temporary tables are deleted after the user disconnects from the instance of SQL Server.
References:
https://technet.microsoft.com/en-us/library/ms190766(v=sql.105).aspx
https://technet.microsoft.com/en-us/library/ms186986.aspx