

# Microsoft

## Exam Questions 70-761

Querying Data with Transact-SQL (beta)



**NEW QUESTION 1**

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

Start of repeated scenario

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

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

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

You review the Employee table and make the following observations:

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

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

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

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

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

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

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

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

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

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

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

End of Repeated Scenario

You are creating the queries for Report1 and Report2.

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

Which object should you use to join the SalesSummary table with the other tables that each report uses? To answer, drag the appropriate objects to the correct reports. each object may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

Objects	Answer area	
<input type="text" value="view"/>	Report	Object
<input type="text" value="indexed view"/>	Report1	<input type="text" value="Object"/>
<input type="text" value="subquery"/>	Report2	<input type="text" value="Object"/>
<input type="text" value="scalar function"/>		
<input type="text" value="table-valued function"/>		
<input type="text" value="stored procedure"/>		
<input type="text" value="derived table"/>		
<input type="text" value="common table expression (CTE)"/>		

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Box 1: common table expression (CTE)

A common table expression (CTE) can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query. Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.

A CTE can be used to:

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

Box 2: view

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

References: [https://technet.microsoft.com/en-us/library/ms190766\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190766(v=sql.105).aspx)

**NEW QUESTION 2**

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

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

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

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

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

Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

**Answer:** B

**Explanation:**

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

**NEW QUESTION 3**

You have a table named Table1 that contains 200 million rows. Table1 contains a column named SaleDate that has a data type of DateTime2(3). Users report that the following query runs slowly.

```
Select SalesPerson, count(*)
FROM table1
Where year(SaleDate) = 2017
GROUP BY SalesPerson
```

You need to reduce the amount of time it takes to run the query. What should you use to replace the WHERE statement?

- A. WHERE SaleDate >= '2017-01-01' AND SaleDate < '2018-01-01'
- B. WHERE cast(SaleDate as varchar(10)) BETWEEN '2017-01-01' AND '2017-12-31'
- C. WHERE cast(SaleDate as date) BETWEEN '2017-01-01' AND '2017-12-31'
- D. WHERE 2017 = year(SaleDate)

**Answer:** C

**Explanation:**

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

**NEW QUESTION 4**

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

Servers

Column	Data type	Notes
ServerID	int	This is the primary key for the table.
DNS	nvarchar(100)	Null values are not permitted for this column.

Errors

Column	Data type	Notes
ErrorID	int	This is the primary key for the table.
ServerID	int	Null values are not permitted for this column. This column is a foreign key that is related to the ServerID column in the Servers table.
Occurrences	int	Null values are not permitted for this column.
LogMessage	nvarchar(max)	Null values are not permitted for this column.

You need to return all error log messages and the server where the error occurs most often. Which Transact-SQL statement should you run?

- A**
- ```
SELECT DISTINCT ServerID, LogMessage FROM Errors AS e1
WHERE Occurrences > ALL (
    SELECT e2.Occurrences FROM Errors AS e2
    WHERE e2.LogMessage = e1.LogMessage AND e2.ServerID <> e1.ServerID
)
```
- B**
- ```
SELECT DISTINCT ServerID, LogMessage FROM Errors AS e1
GROUP BY ServerID, LogMessage
HAVING MAX(Occurrences) = 1
```
- C**
- ```
SELECT DISTINCT ServerID, LogMessage FROM Errors AS e1
WHERE LogMessage IN (
    SELECT TOP 1 e2.LogMessage FROM Errors AS e2
    WHERE e2.LogMessage = e1.LogMessage AND e2.ServerID <> e1.ServerID
    ORDER BY e2.Occurrences
)
```
- D**
- ```
SELECT ServerID, LogMessage FROM Errors AS e1
GROUP BY ServerID, LogMessage, Occurrences
HAVING COUNT(*) = 1
ORDER BY Occurrences
```

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

**Answer: C**

#### 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 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 6**

You develop and deploy a project management application. The application uses a Microsoft SQL Server database to store data. You are developing a software bug tracking add-on for the application.

The add-on must meet the following requirements:

- Allow case sensitive searches for product.
- Filter search results based on exact text in the description.
- Support multibyte Unicode characters.

You run the following Transact-SQL statement:

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

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

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

NOTE: Each correct selection is worth one point.

```
DECLARE @product NVARCHAR(255)
. . .
SELECT
    Id
    ,
    FROM MS1.dbo.Bug
WHERE
```

	▼
Product	
Description	
DateCreated	
ReportingUser	

```
like @product
```

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

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

References:

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

**NEW QUESTION 7**

You create three tables by running the following Transact-SQL statements:

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

For reporting purposes, you need to find the active user count for each role, and the total active user count. The result must be ordered by active user count of each role. You must use common table expressions (CTEs). 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
<pre>Total AS (   SELECT COUNT(*) AS TotalCountInAllRoles   FROM ActiveUsers ) SELECT S.*, Total.TotalCountInAllRoles FROM RoleSummary S, Total ORDER BY S.ActiveUserCount</pre>	
<pre>WITH ActiveUsers AS (   SELECT UserId   FROM tblUsers   WHERE IsActive=1 ),</pre>	
<pre>RoleNCount AS (   SELECT RoleId, COUNT(*) AS ActiveUser-   Count   FROM tblUsersInRoles BRG   INNER JOIN ActiveUsers U ON BRG.UserId =   U.UserId   GROUP BY BRG.RoleId ),</pre>	
<pre>Total AS (   SELECT COUNT(*) AS TotalCountInAllRoles   FROM ActiveUsers ) SELECT S.*, Total.TotalCountInAllRoles FROM RoleSummary S, Total</pre>	
<pre>RoleSummary AS (   SELECT R.RoleName, ISNULL   (S.ActiveUserCount,0) AS ActiveUserCount   FROM tblRoles R   LEFT JOIN RoleNCount S ON R.RoleId =   S.RoleId   ORDER BY S.ActiveUserCount ),</pre>	
<pre>RoleSummary AS (   SELECT R.RoleName, ISNULL   (S.ActiveUserCount,0) AS ActiveUserCount   FROM tblRoles R   LEFT JOIN RoleNCount S ON R.RoleId =   S.RoleId ),</pre>	

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

### Transact-SQL segments

```
Total AS (
    SELECT COUNT(*) AS TotalCountInAllRoles
    FROM ActiveUsers
)
SELECT S.*, Total.TotalCountInAllRoles
FROM RoleSummary S, Total
ORDER BY S.ActiveUserCount
```

```
WITH ActiveUsers AS (
    SELECT UserId
    FROM tblUsers
    WHERE IsActive=1
),
```

```
RoleNCount AS (
    SELECT RoleId, COUNT(*) AS ActiveUser-
    Count
    FROM tblUsersInRoles BRG
    INNER JOIN ActiveUsers U ON BRG.UserId =
    U.UserId
    GROUP BY BRG.RoleId
),
```

```
Total AS (
    SELECT COUNT(*) AS TotalCountInAllRoles
    FROM ActiveUsers
)
SELECT S.*, Total.TotalCountInAllRoles
FROM RoleSummary S, Total
```

```
RoleSummary AS (
    SELECT R.RoleName, ISNULL
    (S.ActiveUserCount,0) AS ActiveUserCount
    FROM tblRoles R
    LEFT JOIN RoleNCount S ON R.RoleId =
    S.RoleId
    ORDER BY S.ActiveUserCount
),
```

```
RoleSummary AS (
    SELECT R.RoleName, ISNULL
    (S.ActiveUserCount,0) AS ActiveUserCount
    FROM tblRoles R
    LEFT JOIN RoleNCount S ON R.RoleId =
    S.RoleId
),
```

### Answer Area

```
RoleNCount AS (
    SELECT RoleId, COUNT(*) AS ActiveUser-
    Count
    FROM tblUsersInRoles BRG
    INNER JOIN ActiveUsers U ON BRG.UserId =
    U.UserId
    GROUP BY BRG.RoleId
),
WITH ActiveUsers AS (
    SELECT UserId
    FROM tblUsers
    WHERE IsActive=1
),
```

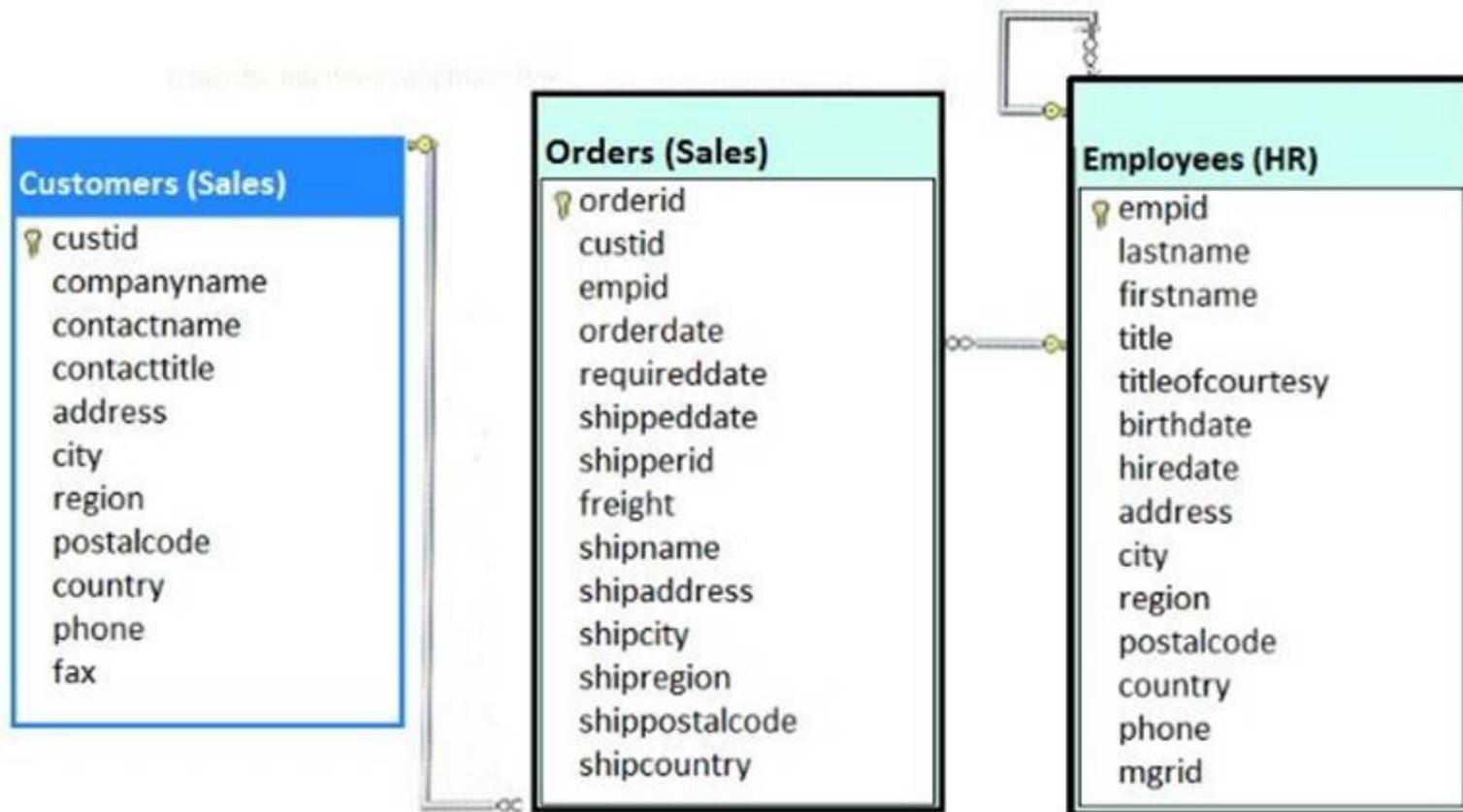
```
RoleSummary AS (
    SELECT R.RoleName, ISNULL
    (S.ActiveUserCount,0) AS ActiveUserCount
    FROM tblRoles R
    LEFT JOIN RoleNCount S ON R.RoleId =
    S.RoleId
    ORDER BY S.ActiveUserCount
```

```
Total AS (
    SELECT COUNT(*) AS TotalCountInAllRoles
    FROM ActiveUsers
)
SELECT S.*, Total.TotalCountInAllRoles
FROM RoleSummary S, Total
ORDER BY S.ActiveUserCount
```



**NEW QUESTION 8**

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
GROUP BY c.custid, contactname, firstname, lastname, o.empid
HAVING o.empid = 4
ORDER BY DateofOrder DESC
    
```

Does the solution meet the goal?

- A. Yes
- B. No

**Answer:** B

**Explanation:**

We should use a WHERE clause, not a HAVING clause. The HAVING clause would refer to aggregate data.

**NEW QUESTION 9**

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

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

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

Which Transact-SQL statement should you run?

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

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

**Answer:** C

**Explanation:**

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

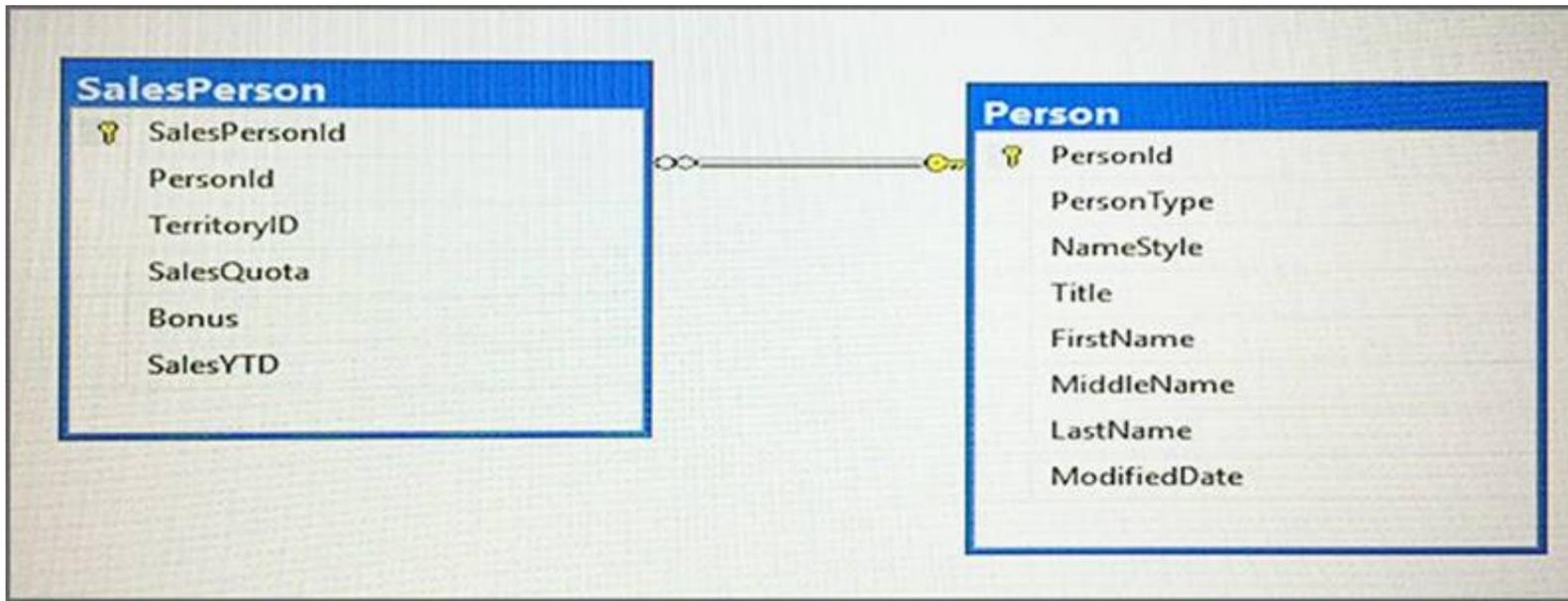
Export data from SQL Server as JSON, or format query results as JSON, by adding the FOR JSON clause to a SELECT statement.

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

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

**NEW QUESTION 10**

You have a database that contains the following tables.



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

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

## Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURRENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT

DEFAULT	ON	TSEQUAL
DELETE	OPEN	UNION
DENY	OPENDATASOURCE	UNIQUE
DESC	OPENQUERY	UNPIVOT
DISK	OPENROWSET	UPDATE
DISTINCT	OPENXML	UPDATETEXT
DISTRIBUTED	OPTION	USE
DOUBLE	OR	USER
DROP	ORDER	VALUES
DUMP	OUTER	VARYING
ELSE	OVER	VIEW
END	PERCENT	WAITFOR
ERRLVL	PIVOT	WHEN
ESCAPE	PLAN	WHERE
ESCEPT	PRECISION	WHILE
EXEC	PRIMARY	WITH
EXECUTE	PRINT	WITHIN GROUP
EXISTS		WRITETEXT

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

```

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

```

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

- A. SELECT TOP 3(p.FirstName + '&#39; &#39; + p.LastName) AS FullName, s.SalesYTD FROM Person AS p INNER JOIN SalesPerson AS s ON p.PersonID = s.PersonID WHERE
- B. TerritoryID IS NOT NULL ORDER BY
- C. SalesYTD DESC

**Answer:** A

**NEW QUESTION 10**

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

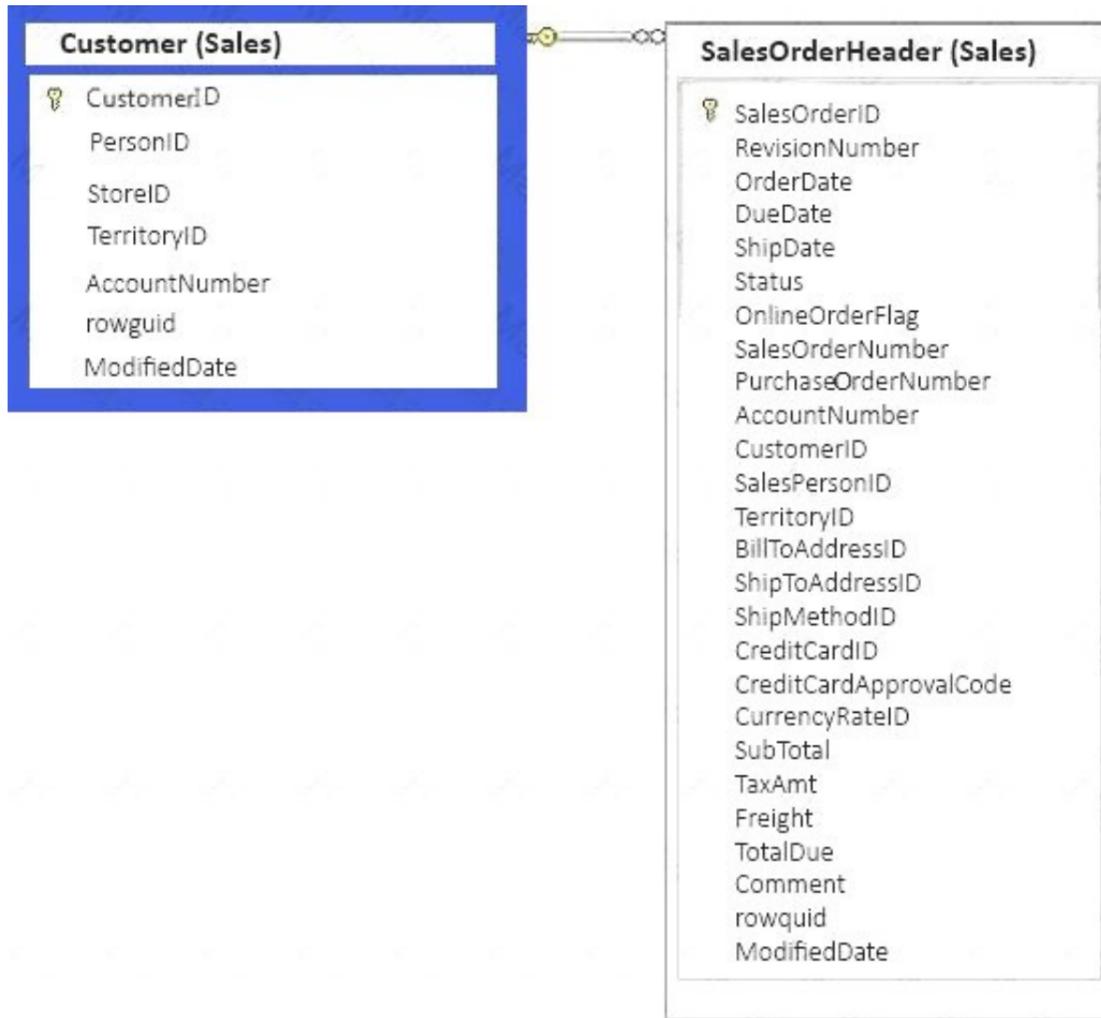
**Answer:** B

**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 14**

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



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

Which Transact-SQL statement should you run?

A

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

B

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

C

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

D

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

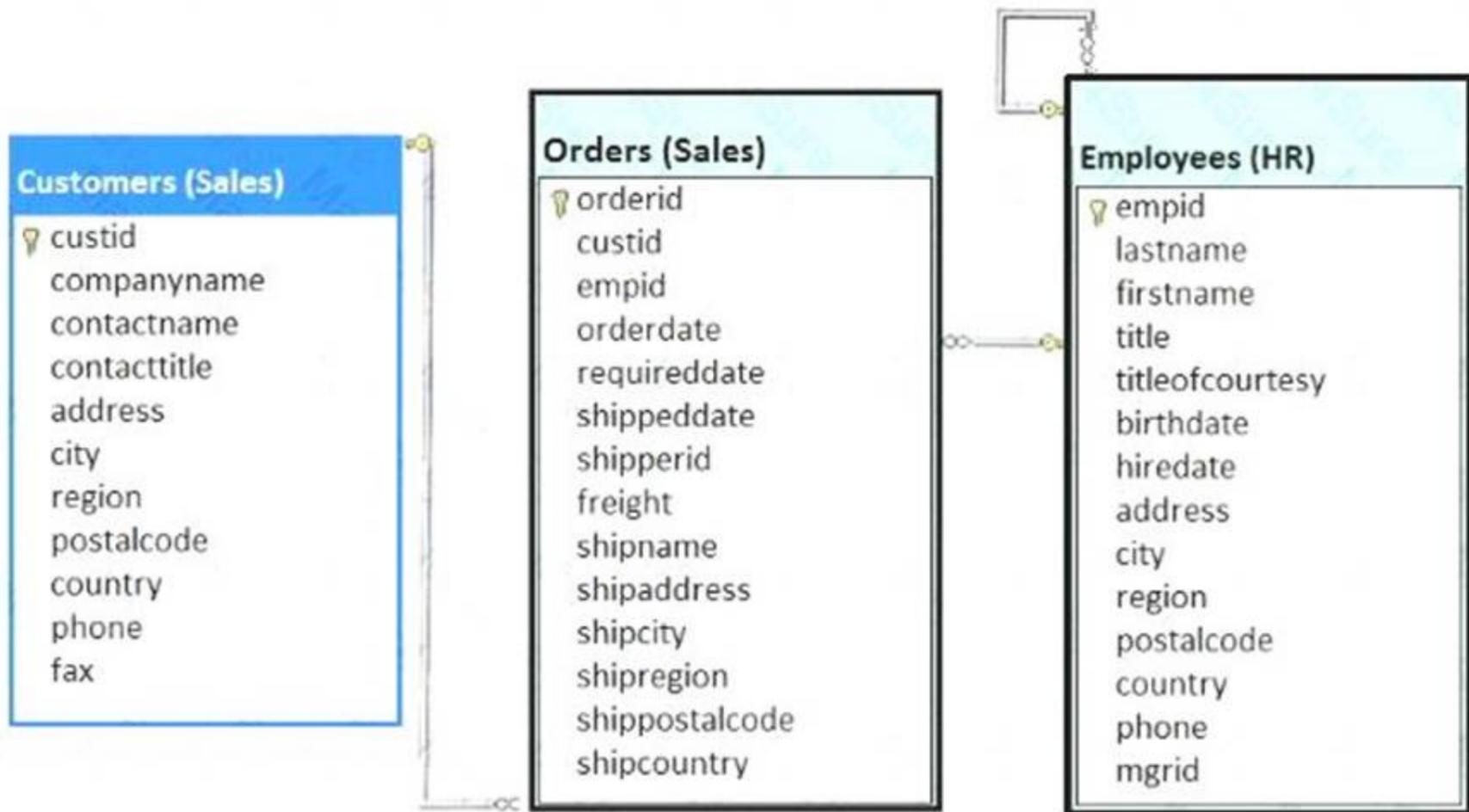
- A. Option A
- B. Option B
- C. Option C

D. Option D

**Answer:** A

**NEW QUESTION 19**

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 23**

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 26**

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 null values
StandardDiscountPercentage	int	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 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 TOP 1 @areaCode = value FROM STRING_SPLIT(@phoneNumber, '-')
    RETURN @areaCode
END
```

Does the solution meet the goal?

- A. Yes
- B. No

**Answer:** A

**Explanation:**

The following indicates a correct solution:

The function returns a nvarchar(10) value.

Schemabinding is used.

SELECT TOP 1 ... gives a single value Note: nvarchar(max) is correct statement. nvarchar [ ( n | max ) ]

Variable-length Unicode string data. n defines the string length and can be a value from 1 through 4,000. max indicates that the maximum storage size is 2<sup>31</sup>-1 bytes (2 GB).

References:

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/nchar-and-nvarchar-transact-sql> <https://sqlstudies.com/2014/08/06/schemabinding-what-why/>

**NEW QUESTION 31**

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

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

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

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

Which Transact-SQL statement should you run?

- A 

```
SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, AnnualRevenue, DateCreated
FROM Customers
GROUP BY GROUPING SETS((FirstName, LastName), (Address), (CustomerID, AnnualRevenue), (CustomerID), ())
ORDER BY CustomerID, FirstName, LastName, Address, AnnualRevenue
```
- B 

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

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

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

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

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

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

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

A. Option A

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

Answer: E

**Explanation:**

Calculate aggregate column through AVG function and GROUP BY clause.

**NEW QUESTION 32**

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

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

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

Output data by using a tree-like structure.

Allow mixed content types.

Use custom metadata attributes.

Which Transact-SQL statement should you run?

- A**

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

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

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

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

E

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

F

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

G

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

H

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

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

Answer: F

**NEW QUESTION 34**

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

You have a database that contains several connected tables. The tables contain sales data for customers in the United States only.

You have the following partial query for the database. (Line numbers are included for reference only.)

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

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

CountryName	StateProvinceName	CityName	TotalSales
United States	Alabama	Bazemore	\$34402.00
United States	Alabama	Belgreen	\$51714.65
United States	Alabama	Broomtown	\$59.349.20
United States	Alabama	Coker	\$26409.50
United States	Alabama	Eulaton	\$54225.35

Which statement clause should you add at line 3?

- A. GROUP BY
- B. MERGE
- C. GROUP BY ROLLUP

- D. LEFT JOIN
- E. GROUP BY CUBE
- F. CROSS JOIN
- G. PIVOT
- H. UNPIVOT

**Answer:** A

**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 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 40**

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

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

\*Propagates all unexpected errors to the calling process

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

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

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

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

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

THROW syntax:

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

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

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

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

On Severity level 16. Using THROW to raise an exception

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

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

Here is the result set.

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

Note: RAISERROR syntax:

RAISERROR ( { msg\_id | msg\_str | @local\_variable }

{ ,severity ,state }

[ ,argument [ ,...n ] ] )

[ WITH option [ ,...n ] ]

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

References:

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

#### NEW QUESTION 42

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://technet.microsoft.com/en-us/library/ms190690(v=sql.105).aspx) <https://msdn.microsoft.com/en-us/library/hh213574.aspx>

**NEW QUESTION 47**

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

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

You need to write a query that will change the value of the job title column to Customer Representative for any employee who lives in Seattle and has a job title of Sales Representative. If the employee does not have a manager defined, you must not change the title. Which three Transact-SQL segments should you use to develop the solution? To answer, move the appropriate Transact-SQL segments from the list of Transact-SQL segments to the answer area and arrange them in the correct order.

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

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

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

**NEW QUESTION 48**

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

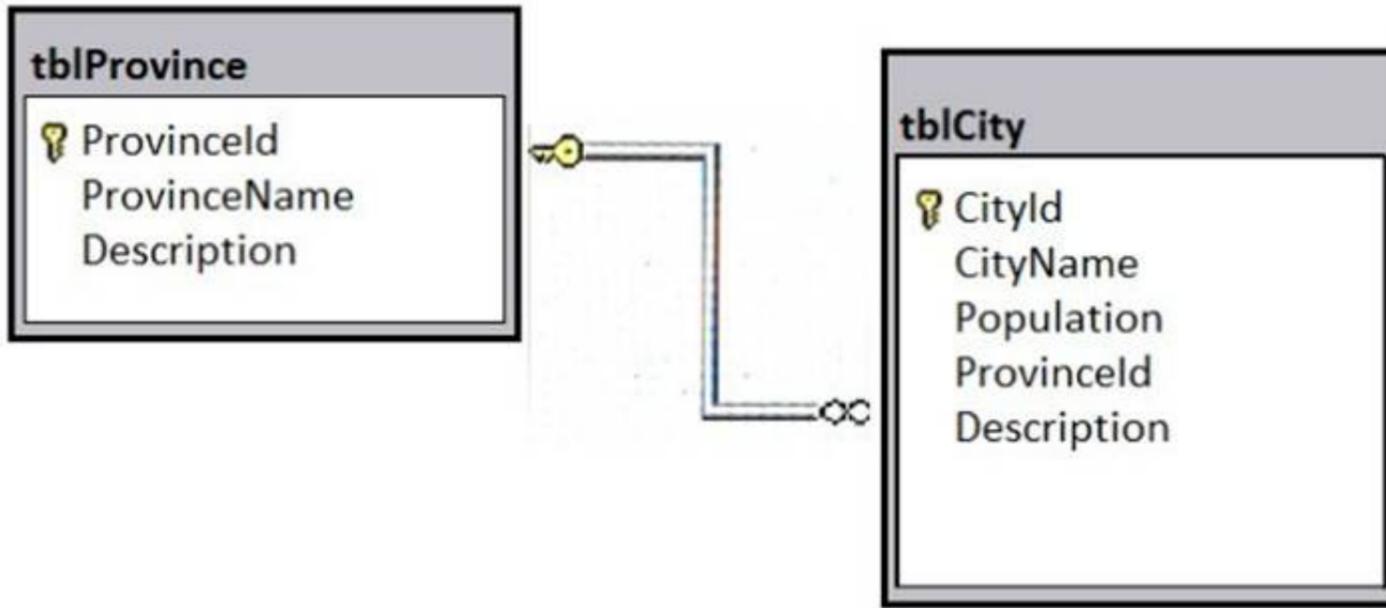
Answer: B

**NEW QUESTION 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.

A database has two tables as shown in the following database diagram:



You need to list all provinces that have at least two large cities. A large city is defined as having a population of at least one million residents. The query must return the following columns:

- tblProvince.Provinceld
- tblProvince.ProvinceName
- a derived column named LargeCityCount that presents the total count of large cities for the province

Solution: You run the following Transact-SQL statement:

```

SELECT P.ProvinceId, P.ProvinceName, CitySummary.LargeCityCount
FROM tblProvince P
CROSS JOIN (
    SELECT COUNT(*) AS LargeCityCount FROM tblCity C
    WHERE C.Population >= 1000000
) CitySummary
WHERE CitySummary.LargeCityCount >= 2
    
```

Does the solution meet the goal?

- A. Yes
- B. No

**Answer:** B

**Explanation:**

The SQL CROSS JOIN produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if no WHERE clause is used along with CROSS JOIN. This kind of result is called as Cartesian Product. This is not what is required in this scenario.

References: [https://technet.microsoft.com/en-us/library/ms190690\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190690(v=sql.105).aspx)

**NEW QUESTION 57**

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

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

```

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

You need to return normalized data for all customers that were added in the year 2014. Which Transact-SQL statement should you run?

A

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

B

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

C

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

D

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

E

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

F

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

G

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

H

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

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

Answer: G

**NEW QUESTION 58**

Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question. You have a database that contains tables named Customer\_CRMSystem and Customer\_HRSystem. Both tables use the following structure:

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

The tables include the following records: Customer\_CRMSystem

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

Customer\_HRSystem

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

Records that contain null values for CustomerCode can be uniquely identified by CustomerName. You need to display a Cartesian product, combining both tables. Which Transact-SQL statement should you run?

- A**

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

```
SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
INTERSECT
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem
```
- C**

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

```
SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
EXCEPT
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem
```
- E**

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

```

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

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

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

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

**Answer:** G

**Explanation:**

A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table.

References: [https://technet.microsoft.com/en-us/library/ms190690\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190690(v=sql.105).aspx)

**NEW QUESTION 59**

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 = CusCat.CustomerCategoryID

```

Transact-SQL segment

CustCat.CustomerCategoryName,

Transact-SQL segment

Transact-SQL segment

Transact-SQL segment

> 5

- A. Mastered
- B. Not Mastered

**Answer: A**

**Explanation:**

### Transact-SQL segments

- CAST (Cust.AccountOpenedDate AS DATE)
- DATEPART (day, Cust.AccountOpenedDate)
- HAVING
- WHERE
- COUNT (Cust.CustomerId)
- MAX (Cust.CustomerID)
- RANK
- GROUP BY

### Answer Area

```

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

**NEW QUESTION 60**

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

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

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

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

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

Solution: You run the following Transact-SQL statement:

```
UPDATE Production.Products
SET ListPrice = (ListPrice* .1)
WHERE ListPrice <100
```

Does the solution meet the goal?

- A. Yes
- B. No

**Answer: B**

**Explanation:**

Mathematical equation will only return 10 % of the value.

**NEW QUESTION 62**

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 63**

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

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

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

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

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

Solution: You run the following Transact-SQL statement:

```
UPDATE Production.Products
SET ListPrice = ListPrice * 1.1
WHERE ListPrice
BETWEEN 0 and 100
```

Does the solution meet the goal?

- A. Yes
- B. No

**Answer: B**

**NEW QUESTION 67**

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

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

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

A reporting table named SurveyReport has the following columns: CityID, QuestionID, and RawCount, where RawCount is the value from the RawSurvey table. You need to write a Transact-SQL query to meet the following requirements:

- Retrieve data from the RawSurvey table in the format of the SurveyReport table.
- The CityID must contain the CityID of the city that was surveyed.
- The order of cities in all SELECT queries must match the order in the RawSurvey table.
- The order of cities in all IN statements must match the order in the RawSurvey table. Construct the query using the following guidelines:
- Use one-part names to reference tables and columns, except where not possible.
- ALL SELECT statements must specify columns.
- Do not use column or table aliases, except those provided.
- Do not surround object names with square brackets.

## Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURRENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT

DEFAULT	ON	TSEQUAL
DELETE	OPEN	UNION
DENY	OPENDATASOURCE	UNIQUE
DESC	OPENQUERY	UNPIVOT
DISK	OPENROWSET	UPDATE
DISTINCT	OPENXML	UPDATETEXT
DISTRIBUTED	OPTION	USE
DOUBLE	OR	USER
DROP	ORDER	VALUES
DUMP	OUTER	VARYING
ELSE	OVER	VIEW
END	PERCENT	WAITFOR
ERRLVL	PIVOT	WHEN
ESCAPE	PLAN	WHERE
ESCEPT	PRECISION	WHILE
EXEC	PRIMARY	WITH
EXECUTE	PRINT	WITHIN GROUP
EXISTS		WRITETEXT

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

```

1 SELECT CityID, QuestionID, RawCount
2 AS t1
3 AS t2
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:**

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

**NEW QUESTION 69**

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>

**NEW QUESTION 73**

.....

## **Thank You for Trying Our Product**

### **We offer two products:**

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

2nd - Questions and Answers in PDF Format

### **70-761 Practice Exam Features:**

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

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