

70-761.exam.75q

Number: 70-761
Passing Score: 800
Time Limit: 120 min



<https://www.gratisexam.com/>

70-761

Querying Data with Transact-SQL

<https://www.gratisexam.com/>

Exam A

QUESTION 1

You need to create an indexed view that requires logic statements to manipulate the data that the view displays.

Which two database objects should you use? Each correct answer presents a complete solution.



<https://www.gratisexam.com/>

- A. a user-defined table-valued function
- B. a CLR function
- C. a stored procedure
- D. a user-defined scalar function

Correct Answer: AC

Section: (none)

Explanation

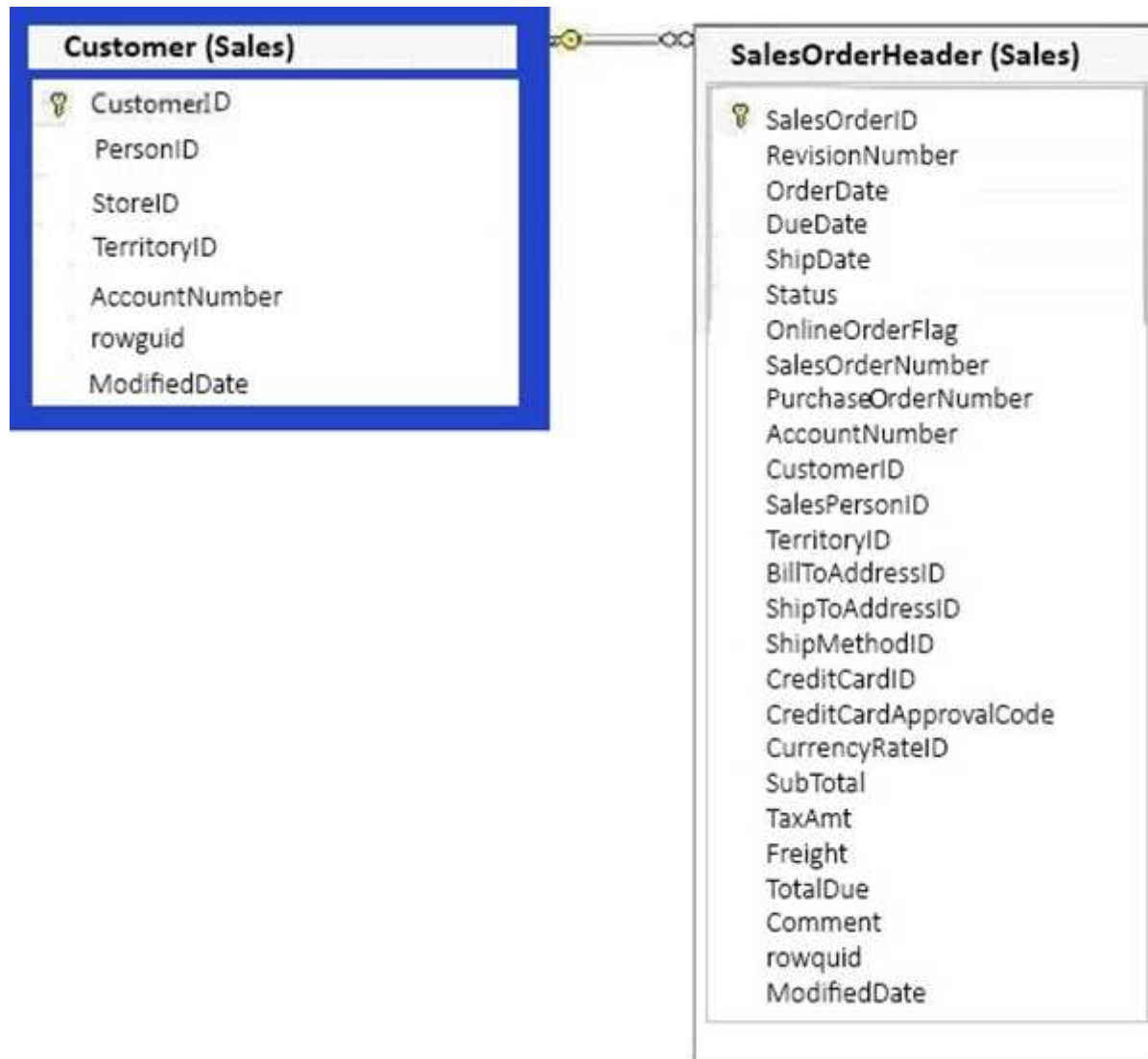
Explanation/Reference:

Explanation:

You can create a database object inside an instance of SQL Server that is programmed in an assembly created in the Microsoft .NET Framework common language runtime (CLR). Database objects that can leverage the rich programming model provided by the common language runtime include aggregate functions, functions, stored procedures, triggers, and types.

QUESTION 2

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



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

Which Transact-SQL statement should you run?

A.

```
SELECT C.CustomerID, ISNULL(SOH.SalesOrderID, 0) AS OrderID, ISNULL(MAX(OrderDate), '')
FROM Sales.Customer C LEFT OUTER JOIN Sales.SalesOrderHeader SOH
ON C.CustomerID = SOH.CustomerID
GROUP BY C.CustomerID, SOH.SalesOrderID
ORDER BY C.CustomerID
```

B.

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

C.

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

D.

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

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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

ISNULL Syntax: ISNULL (check_expression , replacement_value) author:"Luxemburg, Rosa"

The ISNULL function replaces NULL with the specified replacement value. The value of check_expression is returned if it is not NULL; otherwise, replacement_value is returned after it is implicitly converted to the type of check_expression.

References: <https://msdn.microsoft.com/en-us/library/ms184325.aspx>

QUESTION 3

You have a database that contains the following tables:

Customer

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

CustomerAudit

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

Where the value of the CustomerID column equals 3, you need to update the value of the CreditLimit column to 1000 for the customer. You must ensure that the change to the record in the Customer table is recorded on the CustomerAudit table.

Which Transact-SQL statement should you run?

A.

```
UPDATE Customer
SET CreditLimit = 1000
WHERE CustomerId = 3
INSERT INTO dbo.CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
SELECT CustomerId, CreditLimit, CreditLimit
FROM Customer
WHERE CustomerId = 3
```

B.

```
UPDATE Customer
SET CreditLimit = 1000
WHERE CustomerId = 3
INSERT INTO dbo.CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
SELECT CustomerId, CreditLimit, CreditLimit
FROM Customer
```

C.

```
UPDATE Customer
SET CreditLimit = 1000
OUTPUT inserted.CustomerId, inserted.CreditLimit, deleted.CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
WHERE CustomerId = 3
```

D.

```
UPDATE Customer
SET CreditLimit = 1000
OUTPUT inserted.CustomerId, deleted.CreditLimit, inserted.CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
WHERE CustomerId = 3
```

A. Option A

B. Option B

- C. Option C
- D. Option D

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

Incorrect Answers:

C: The deleted.Creditlimit should be inserted in the second column, the OldCreditLimit column, not the third column.

References: <https://msdn.microsoft.com/en-us/library/ms177564.aspx>

QUESTION 4

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

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

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

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

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

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

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

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

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

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

- For customers that are on a credit hold, return the CustomerID and the latest recorded population for the postal city that is associated with the customer.

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

A.

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

B.

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

C.

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

D.

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

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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

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>

QUESTION 5

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

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Explanation:

The datetime datatype 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.

Incorrect Answers:

A: ValidFrom should be less (<) than @sdate AND ValidTo should be greater (>) than @edate.

B: We should add a day with DATEADD, not subtract one day.

C: We cannot compare a date to an exact datetime.

References: <https://msdn.microsoft.com/en-us/library/ms187819.aspx>

QUESTION 6

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

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

Explanation:

The ISNULL function replaces NULL with the specified replacement value.

References: <https://msdn.microsoft.com/en-us/library/ms184325.aspx>

QUESTION 7

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



<https://www.gratisexam.com/>

- 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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

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 8

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

You 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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Explanation:

User-defined functions that return a table data type can be powerful alternatives to views. These functions are referred to as table-valued functions. A table-valued

user-defined function can be used where table or view expressions are allowed in Transact-SQL queries. While views are limited to a single SELECT statement, user-defined functions can contain additional statements that allow more powerful logic than is possible in views. A table-valued user-defined function can also replace stored procedures that return a single result set.

References: [https://technet.microsoft.com/en-us/library/ms191165\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms191165(v=sql.105).aspx)

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 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

Correct Answer: H

Section: (none)

Explanation

Explanation/Reference:

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 10

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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

QUESTION 11

DRAG DROP

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

Servers

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

Errors

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

You are building a webpage that shows the three most common errors for each server.

You need to return the data for the webpage.

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

NOTE: Each correct selection is worth one point.

Select and Place:

Transact-SQL segments

svr.ServerID

errs.ServerID

INNER JOIN

CROSS APPLY

WITHIN GROUP

WHERE ServerID =
svr.ServerID

WHERE ServerID =
errs.ErrorID



Answer Area

```
SELECT  Transact-SQL segment , errs.LogMessage
FROM Servers AS svr
Transact-SQL segment
(
    SELECT TOP 3 LogMessage
    FROM Errors
    Transact-SQL segment
    ORDER BY Occurrences
) AS errs
```

Correct Answer:

Transact-SQL segments

errs.ServerID
INNER JOIN
WITHIN GROUP
WHERE ServerID = errs.ErrorID



Answer Area

```
SELECT  svr.ServerID , errs.LogMessage
FROM Servers AS svr
CROSS APPLY
(
    SELECT TOP 3 LogMessage
    FROM Errors
    WHERE ServerID =
    svr.ServerID
    ORDER BY Occurrences
) AS errs
```

Section: (none)

Explanation

Explanation/Reference:

QUESTION 12

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

```

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

```

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

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

Which Transact-SQL statement should you run?

A.

```

SELECT R.RoleName, COUNT(U.UserId) AS ActiveUserCount FROM tblRoles R
LEFT JOIN (SELECT UserId, RoleId FROM tblUsers WHERE IsActive = 1) U ON U.RoleId = R.RoleId
GROUP BY R.RoleId, R.RoleName

```

B.

```

SELECT R.RoleName, U.ActiveUserCount FROM tblRoles R
INNER JOIN (SELECT RoleId, COUNT(*) AS ActiveUserCount FROM tblUsers WHERE IsActive = 1
GROUP BY RoleId) U ON R.RoleId = U.RoleId

```

C.

```

SELECT R.RoleName, COUNT(*) AS ActiveUserCount FROM tblRoles R
LEFT JOIN (SELECT UserId, RoleId FROM tblUsers WHERE IsActive = 1) U ON U.RoleId = R.RoleId
GROUP BY R.RoleId, R.RoleName

```

D.

```

SELECT R.RoleName, U.ActiveUserCount FROM tblRoles R CROSS JOIN
(SELECT COUNT(*) AS ActiveUserCount FROM tblUsers WHERE IsActive = 1) U

```

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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

QUESTION 13

SIMULATION

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

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.

Correct Answer: Please see explanation

Section: (none)

Explanation

Explanation/Reference:

Explanation:

```
1 SELECT Rawcount FROM (SELECT CityID, QuestionID, RawCount)
2 AS t1 UNPIVOT
3 (rawcount for questioned in (QuestionID)) AS t2
4 JOIN t2 ON t1.CityName = t2.cityName
```

UNPIVOT must be used to rotate columns of the Rawsurvey table into column values.

References: [https://technet.microsoft.com/en-us/library/ms177410\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx)

QUESTION 14

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 list of customers that do not appear in the Customer_HRSystem table.
Which Transact-SQL statement should you run?

A.

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

B.

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

C.

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

D.

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

E.

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

F.

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

G.

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

H.

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

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

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Explanation:

EXCEPT returns distinct rows from the left input query that aren't output by the right input query.

References: <https://msdn.microsoft.com/en-us/library/ms188055.aspx>

QUESTION 15

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 customers who appear in both tables and have a proper CustomerCode.

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 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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

When there are null values in the columns of the tables being joined, the null values do not match each other. The presence of null values in a column from one of the tables being joined can be returned only by using an outer join (unless the WHERE clause excludes null values).

References: [https://technet.microsoft.com/en-us/library/ms190409\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190409(v=sql.105).aspx)

QUESTION 16

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

You have a database that 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

Correct Answer: G

Section: (none)

Explanation

Explanation/Reference:

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)

QUESTION 17

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

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

The tables include the following records:

Customer_CRMSystem



<https://www.gratisexam.com/>

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

Customer_HRSystem

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

Records that contain null values for CustomerCode can be uniquely identified by CustomerName.

You need to create a list of all unique customers that appear in either table.

Which Transact-SQL statement should you run?

A.

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

B.

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

C.

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

D.

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

E.

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

F.

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

G.

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

H.

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

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

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

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

Incorrect Answers:

F: UNION ALL incorporates all rows into the results. This includes duplicates. If not specified, duplicate rows are removed.

References: <https://msdn.microsoft.com/en-us/library/ms180026.aspx>

QUESTION 18

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

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

```

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

```

You are developing a report that aggregates customer data only for the year 2014. The report requires that the data be denormalized.

You need to return the data for the report.

Which Transact-SQL statement should you run?

A.

```

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

```

B.

```

SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, AnnualRevenue, DateCreated, ValidFrom, ValidTo
FROM Customers
FOR SYSTEM_TIME ALL ORDER BY ValidFrom

```

C.


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

D.

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

E.

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

F.

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

G.

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

H.

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

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

Correct Answer: G

Section: (none)

Explanation

Explanation/Reference:

QUESTION 19

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

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

```

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

```

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

- Output data by using a tree-like structure.
- Allow mixed content types.
- Use custom metadata attributes.

Which Transact-SQL statement should you run?

A.

```

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

```

B.

```

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

```

C.

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

D.

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

E.

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

F.

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

G.

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

H.

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

A. Option A

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

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

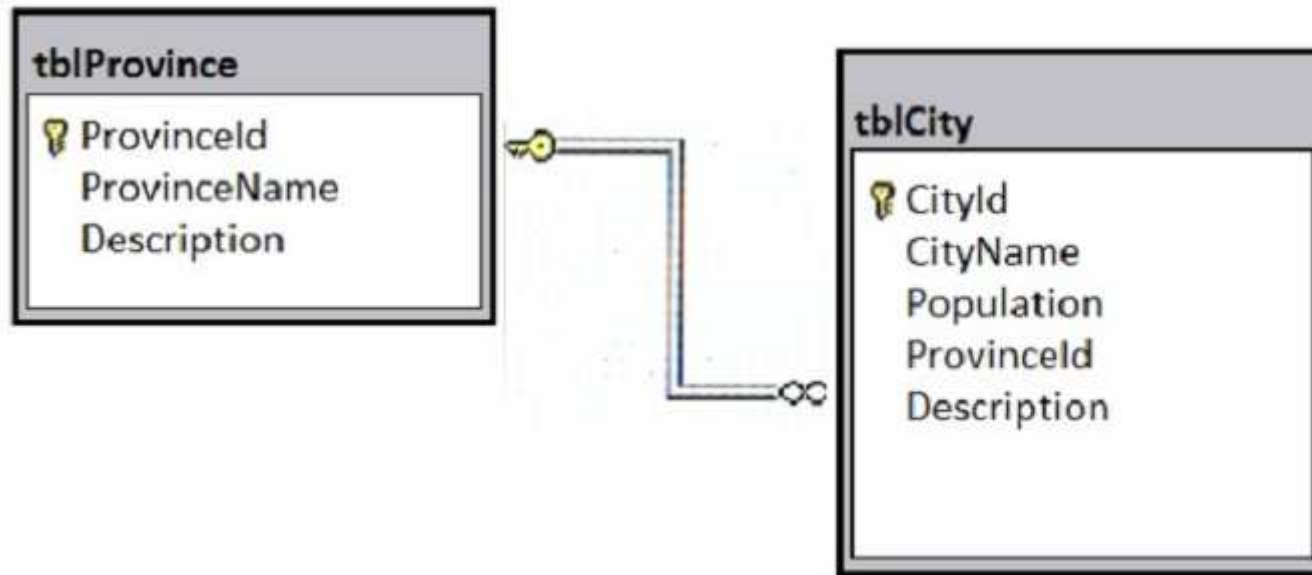
References: <https://msdn.microsoft.com/en-us/library/ms178107.aspx>

QUESTION 20

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

Solution: You run the following Transact-SQL statement:

```
SELECT P.ProvinceId, P.ProvinceName, CitySummary.LargeCityCount
FROM tblProvince P
OUTER APPLY (
    SELECT COUNT(*) AS LargeCityCount FROM tblCity C
    WHERE C.Population >= 1000000 AND C.ProvinceId = P.ProvinceId
) CitySummary
```

Does the solution meet the goal?

A. Yes

B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

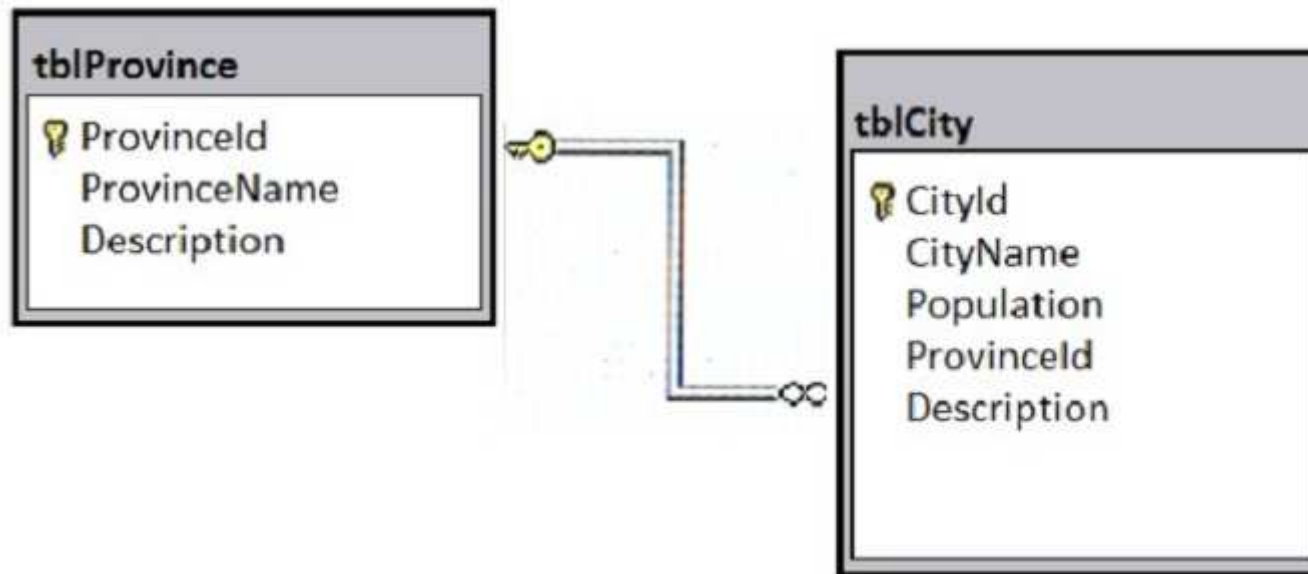
We need to list all provinces that have at least two large cities. There is no reference to this in the code.

QUESTION 21

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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

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.

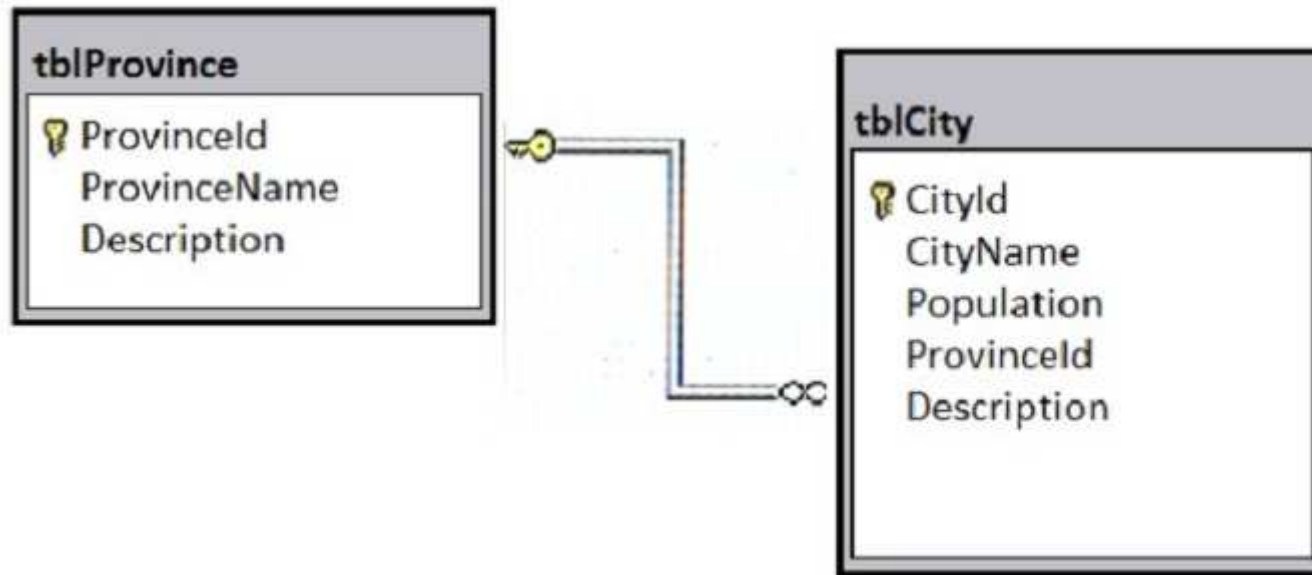
References: [https://technet.microsoft.com/en-us/library/ms190690\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190690(v=sql.105).aspx)

QUESTION 22

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

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

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 APPLY (
    SELECT COUNT(*) AS LargeCityCount FROM tblCity C
    WHERE C.Population>=1000000 AND C.ProvinceId = P.ProvinceId
) CitySummary
WHERE CitySummary.LargeCityCount >=2
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

The requirement to list all provinces that have at least two large cities is met by the WHERE CitySummary.LargeCityCount >=2 clause.

CROSS APPLY will work fine here.

Note:

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

There are two forms of APPLY: CROSS APPLY and OUTER APPLY. CROSS APPLY returns only rows from the outer table that produce a result set from the table-valued function. OUTER APPLY returns both rows that produce a result set, and rows that do not, with NULL values in the columns produced by the table-valued function.

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

QUESTION 23

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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

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 $2^{31}-1$ bytes (2 GB).

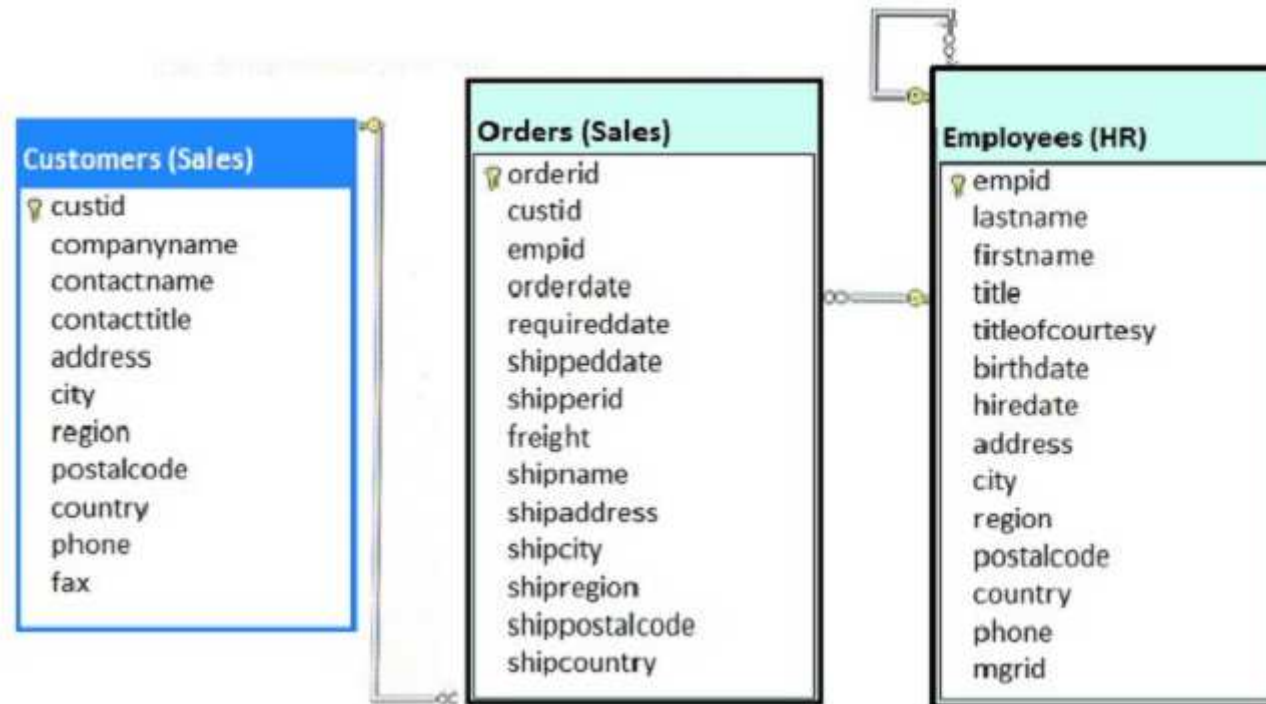
References:

<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/>

QUESTION 24

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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

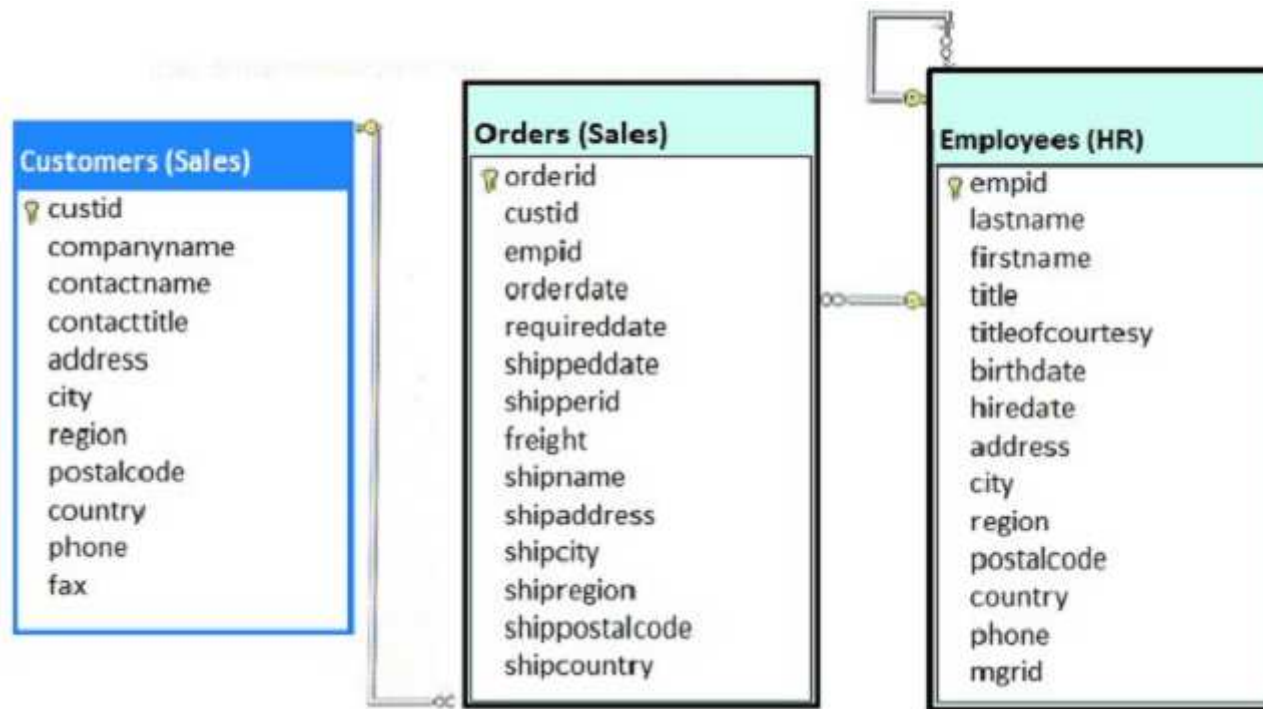
We should use a WHERE clause, not a HAVING clause. The HAVING clause would refer to aggregate data.

QUESTION 25

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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

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.

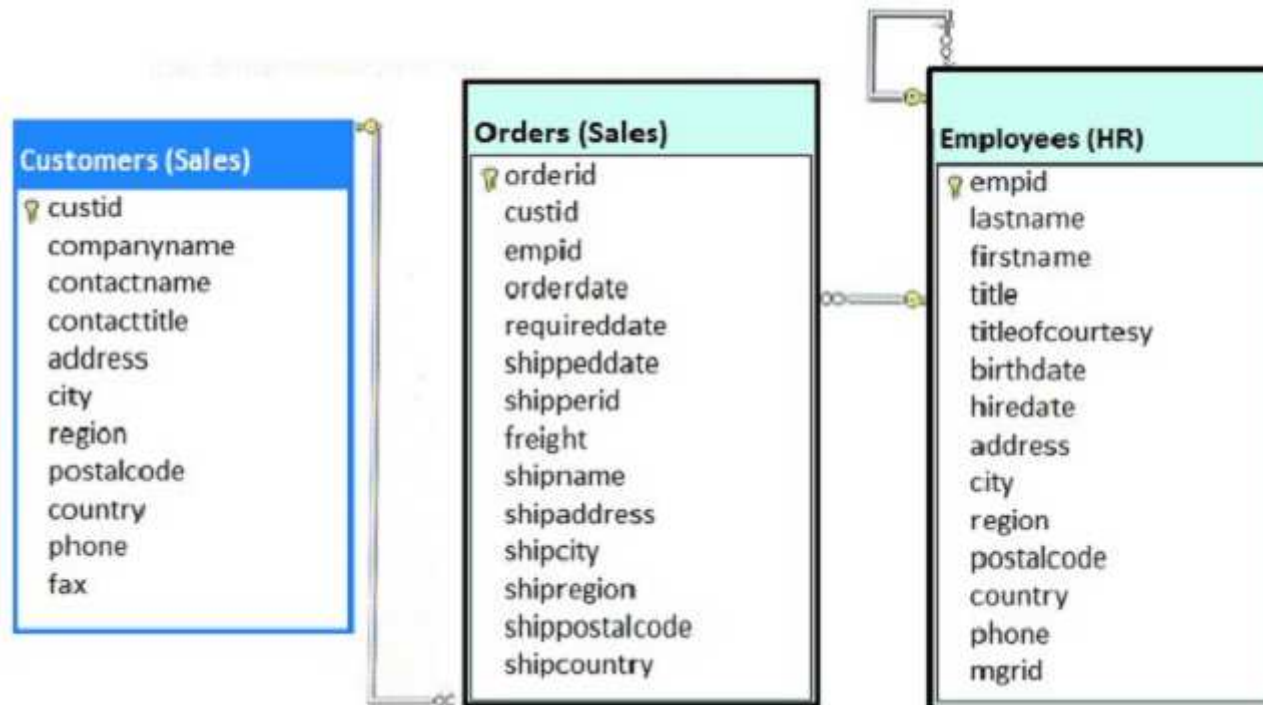
GROUP BY is also required.

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 includes the tables shown in the exhibit (Click the Exhibit button.)



You need to create a Transact-SQL query that returns the following information:

- the customer number
- the customer contact name
- the date the order was placed, with a name of DateofOrder
- a column named Salesperson, formatted with the employee first name, a space, and the employee last name
- orders for customers where the employee identifier equals 4

The output must be sorted by order date, with the newest orders first.

The solution must return only the most recent order for each customer.

Solution: You run the following Transact-SQL statement:

```
SELECT c.custid, contactname, MAX(orderdate) AS DateofOrder,  
e.firstname + ' ' + e.lastname AS Salesperson  
FROM Sales.Customers AS c  
INNER JOIN Sales.Orders AS o ON c.custid = o.custid  
INNER JOIN HR.Employees AS e ON o.empid = e.empid  
WHERE o.empid = 4  
ORDER BY DateofOrder DESC
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

QUESTION 27

DRAG DROP

You have two tables named UserLogin and Employee respectively.

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

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

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

Select and Place:

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

Correct Answer:

Code segments

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

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

Answer Area

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

```
BEGIN CATCH
```

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

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

```
END CATCH
```

Section: (none)

Explanation

Explanation/Reference:

QUESTION 28

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

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

Which Transact-SQL statement should you run?

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

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Incorrect Answers:

A: For column Reason we must use nvarchar, not varchar, as multilingual values must be supported.

B: We cannot use INT for the Id column as new values must be automatically generated.

C: For column Reason we must use nvarchar, not varchar, as multilingual values must be supported.

Note: Nvarchar stores UNICODE data. If you have requirements to store UNICODE or multilingual data, nvarchar is the choice. Varchar stores ASCII data and should be your data type of choice for normal use.

References: <https://docs.microsoft.com/en-us/sql/t-sql/data-types/nchar-and-nvarchar-transact-sql>

QUESTION 29

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

Multiple processes use the data from a table named Sales and place it in other databases across the organization. Some of the processes are not completely aware of the data types in the Sales table. This leads to data type conversion errors.

You need to implement a method that returns a NULL value if data conversion fails instead of throwing an error.

What should you implement?

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

Correct Answer: H

Section: (none)

Explanation

Explanation/Reference:

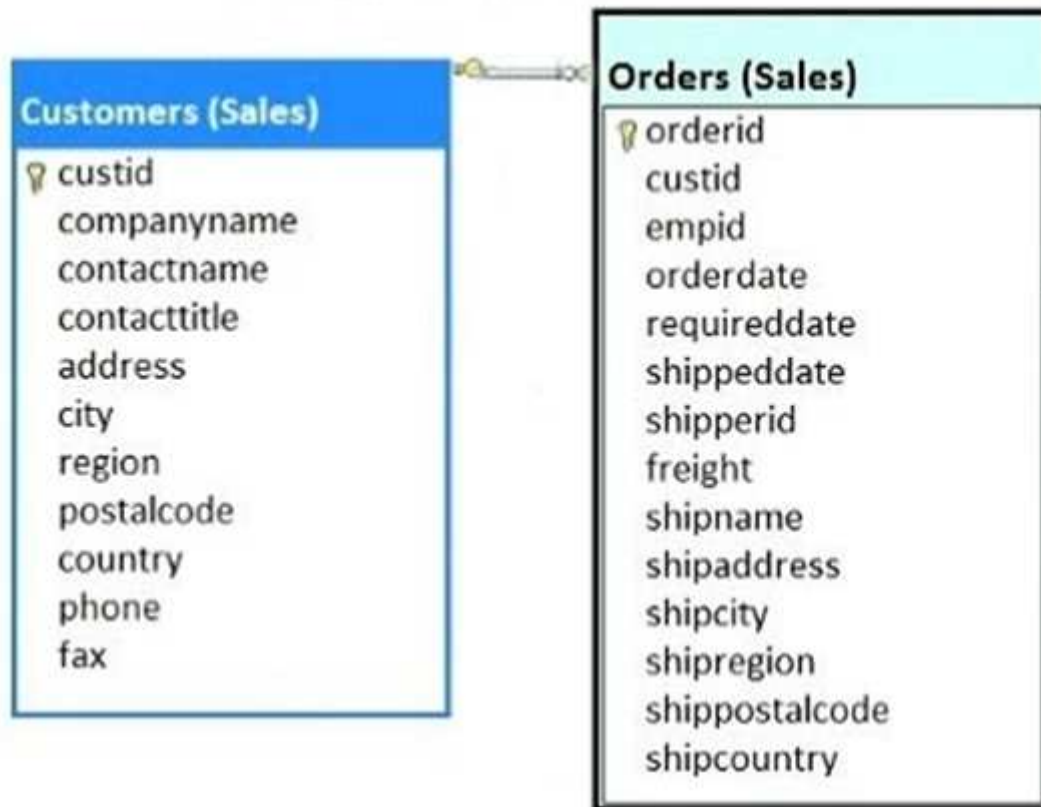
Explanation:

TRY_CONVERT returns a value cast to the specified data type if the cast succeeds; otherwise, returns null.

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

QUESTION 30

You have a database that contains the following tables:



You need to write a query that returns a list of all customers who have not placed orders.

Which Transact-SQL statement should you run?

- A. SELECT c.custid FROM Sales.Customers c INNER JOIN Sales.Order o ON c.custid = o.custid
- B. SELECT custid FROM Sales.Customers INTERSECT SELECT custid FROM Sales.Orders
- C. SELECT c.custid FROM Sales.Customers c LEFT OUTER Sales.Order o ON c.custid = o.custid
- D. SELECT c.custid FROM Sales.Customers c LEFT OUTER JOIN Sales.Order o ON c.custid = o.custid WHERE orderid IS NULL

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Explanation:

Inner joins return rows only when there is at least one row from both tables that matches the join condition. Inner joins eliminate the rows that do not match with a row from the other table. Outer joins, however, return all rows from at least one of the tables or views mentioned in the FROM clause, as long as those rows meet any WHERE or HAVING search conditions. All rows are retrieved from the left table referenced with a left outer join, and all rows from the right table referenced in a right outer join. All rows from both tables are returned in a full outer join.

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

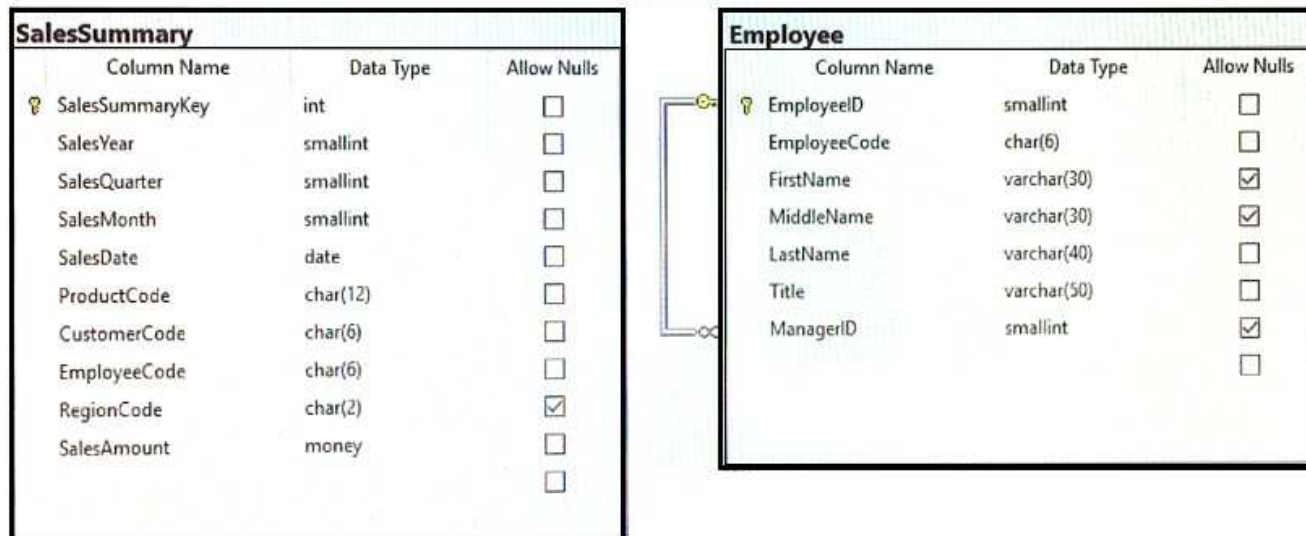
QUESTION 31

HOTSPOT

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.)



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:

- be joinable with the SELECT statement that supplies data for the report
- can be used multiple times for this report and other reports
- accept parameters
- be saved as a permanent object

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

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

End of Repeated Scenario

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

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

Hot Area:

Answer area

Column

Function

MiddleName

	▼
NULLIF	
REPLACE	
COALESCE	

RegionCode

	▼
NULLIF	
REPLACE	
COALESCE	

Correct Answer:

Answer area

Column	Function
MiddleName	<div><div></div><div>▼</div></div> <div>NULLIF</div> <div>REPLACE</div> <div>COALESCE</div>
RegionCode	<div><div></div><div>▼</div></div> <div>NULLIF</div> <div>REPLACE</div> <div>COALESCE</div>

Section: (none)

Explanation

Explanation/Reference:

Explanation:

Box 1: COALESCE

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

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

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

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

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

Box 2: COALESCE

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

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

QUESTION 32

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

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

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

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

Which Transact-SQL statement should you run?

A.

```
SELECT R.RoleName, COUNT(*) AS ActiveUserCount  
FROM tblRoles R  
CROSS JOIN (SELECT UserId, RoleId FROM tblUsers  
    WHERE IsActive = 1) U  
WHERE U.RoleId = R.RoleId  
GROUP BY R.RoleId, R.RoleName
```

- B. `SELECT R.RoleName, COUNT(*) AS ActiveUserCount
FROM tblRoles R
LEFT JOIN (SELECT UserId, RoleId FROM tblUsers
WHERE IsActive = 1) U
ON U.RoleId = R.RoleId
GROUP BY R.RoleId, R.RoleName`
- C. `SELECT R.RoleName, U.ActiveUserCount
FROM tblRoles R
CROSS JOIN (SELECT RoleId, COUNT(*) AS ActiveUserCount
FROM tblUsers WHERE IsActive = 1 GROUP BY RoleId) U`
- D. `SELECT R.RoleName, ISNULL (U.ActiveUserCount, 0)
AS ActiveUserCount
FROM tblRoles R
LEFT JOIN (SELECT RoleId, COUNT(*) AS ActiveUserCount
FROM tblUsers WHERE IsActive = 1 GROUP BY RoleId) U`

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 33

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

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

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

You need to determine the total number of customers who have either deposit accounts or loan accounts, but not both types of accounts.

Which Transact-SQL statement should you run?

- A. `SELECT COUNT(*)
FROM (SELECT AcctNo
FROM tblDepositAcct
INTERSECT
SELECT AcctNo
FROM tblLoanAcct) R`
- B. `SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
UNION
SELECT CustNo
FROM tblLoanAcct) R`

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

Correct Answer: G

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

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

```
USE AdventureWorks2008R2;
GO
-- The OUTER keyword following the FULL keyword is optional.
SELECT p.Name, sod.SalesOrderID
FROM Production.Product p
FULL OUTER JOIN Sales.SalesOrderDetail sod
ON p.ProductID = sod.ProductID
WHERE p.ProductID IS NULL
OR sod.ProductID IS NULL
ORDER BY p.Name
```

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

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 to that question.

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

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

You need to determine the total number of customers who have only loan accounts.

Which Transact-SQL statement should you run?

- A. `SELECT COUNT(*)
FROM (SELECT AcctNo
FROM tblDepositAcct
INTERSECT
SELECT AcctNo
FROM tblLoanAcct) R`
- B. `SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
UNION
SELECT CustNo
FROM tblLoanAcct) R`

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

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

QUESTION 35

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

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

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

You need to run a query to find the total number of customers who have both deposit and loan accounts.

Which Transact-SQL statement should you run?

- A. `SELECT COUNT(*)
FROM (SELECT AcctNo
FROM tblDepositAcct
INTER
SECTSELECT Acct
NoFROM tblLoanAcct) R`
- B. `SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
UNION
SELECT CustNo
FROM tblLoanAcct) R`
- C. `SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
UNION ALL
SELECT CustNo
FROM tblLoanAcct) R`
- D. `SELECT COUNT (DISTINCT D.CustNo)
FROM tblDepositAcctD, tblLoanAcct L
WHERE D.CustNo = L.CustNo`
- E. `SELECT COUNT(DISTINCT L.CustNo)
FROM tblDepositAcct D
RIGHT JOIN tblLoanAcct L ON D.CustNo = L.CustNo
WHERE D.CustNo IS NULL`

- F. `SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
EXCEPT
SELECT CustNo
FROM tblLoanAcct) R`
- G. `SELECT COUNT (DISTINCT COALESCE(D.CustNo, L.CustNo))
FROM tblDepositAcct D
FULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo
WHERE D.CustNo IS NULL OR L.CustNo IS NULL`
- H. `SELECT COUNT(*)
FROM tblDepositAcct D
FULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo`

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

QUESTION 36

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 < 100
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

QUESTION 37

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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

QUESTION 38

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

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

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

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

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

Solution: You run the following Transact-SQL statement:


```
UPDATE Production.Products
SET ListPrice = ListPrice + 1.1
WHERE ListPrice
BETWEEN .01 and 99.99
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

QUESTION 39

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

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

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

```
CREATE TABLE Products (
    ProductID int NOT NULL PRIMARY KEY,
    ProductName nvarchar(100) NULL,
    UnitPrice decimal(18, 2) NOT NULL,
    UnitsInStock int NOT NULL,
    UnitsOnOrder int NULL
)
```

The Products table includes the data shown in the following table:

ProductID	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	ProductA	10.00	10	15
2	ProductB	30.00	20	Null
3	ProductC	15.00	5	20

TotalUnitPrice is calculated by using the following formula:

$\text{TotalUnitPrice} = \text{UnitPrice} * (\text{UnitsInStock} + \text{UnitsOnOrder})$

You need to ensure that the value returned for TotalUnitPrice for ProductB is equal to 600.00.

Solution: You run the following Transact-SQL statement:

```
SELECT ProductName, UnitPrice*(UnitsInStock+ISNULL(UnitsOnOrder,0)) AS
TotalUnitPrice FROM Products
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

ISNULL (check_expression , replacement_value)

Arguments:

check_expression

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

replacement_value

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

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

QUESTION 40

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

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

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

```
CREATE TABLE Products (  
    ProductID int NOT NULL PRIMARY KEY,  
    ProductName nvarchar(100) NULL,  
    UnitPrice decimal(18, 2) NOT NULL,  
    UnitsInStock int NOT NULL,  
    UnitsOnOrder int NULL  
)
```

The Products table includes the data shown in the following table:

ProductID	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	ProductA	10.00	10	15
2	ProductB	30.00	20	Null
3	ProductC	15.00	5	20

TotalUnitPrice is calculated by using the following formula:

$$\text{TotalUnitPrice} = \text{UnitPrice} * (\text{UnitsInStock} + \text{UnitsOnOrder})$$

You need to ensure that the value returned for TotalUnitPrice for ProductB is equal to 600.00.

Solution: You run the following Transact-SQL statement:

```
SELECT ProductName, UnitPrice*(UnitsInStock+COALESCE(UnitsOnOnrder,0)) AS  
TotalUnitPrice FROM Products
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

QUESTION 41

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

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

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

```
CREATE TABLE Products (  
    ProductID int NOT NULL PRIMARY KEY,  
    ProductName nvarchar(100) NULL,  
    UnitPrice decimal(18, 2) NOT NULL,  
    UnitsInStock int NOT NULL,  
    UnitsOnOrder int NULL  
)
```

The Products table includes the data shown in the following table:

ProductID	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	ProductA	10.00	10	15
2	ProductB	30.00	20	Null
3	ProductC	15.00	5	20

TotalUnitPrice is calculated by using the following formula:

$\text{TotalUnitPrice} = \text{UnitPrice} * (\text{UnitsInStock} + \text{UnitsOnOrder})$

You need to ensure that the value returned for TotalUnitPrice for ProductB is equal to 600.00.

Solution: You run the following Transact-SQL statement:

```
SELECT ProductName, UnitPrice*(UnitsInStock+UnitsOnOrder) AS
TotalUnitPrice FROM Products
```

Does the solution meet the goal?



<https://www.gratisexam.com/>

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

The NULL value in the UnitsOnOrder field would cause a runtime error.

QUESTION 42

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

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

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

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

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

CountryName	StateProvinceName	CityName	TotalSales
NULL	NULL	NULL	\$23395792.75
NULL	NULL	Abbotsburg	\$45453.25
NULL	NULL	Absecon	\$33140.15
NULL	NULL	Accomac	\$43226.80
NULL	NULL	Aceitunas	\$23001.40

Which statement clause should you add at line 3?

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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

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

- If a grouping column contains NULL, all null values are considered equal, and they are put into one NULL group.
- When a column is aggregated in a row, the value of the column is shown as NULL.

Example of GROUP BY ROLLUP result set:

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

References: [https://technet.microsoft.com/en-us/library/bb522495\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/bb522495(v=sql.105).aspx)

QUESTION 43

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

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

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

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

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

CountryName	StateProvinceName	CityName	TotalSales
United States	Wyoming	Yoder	\$7638.11
United States	Wyoming	NULL	\$1983745.99
United States	NULL	NULL	\$2387435981.22
NULL	NULL	NULL	\$2387435981.22

Which statement clause should you add at line 3?

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

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

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)

QUESTION 44

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 datetime(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 = CustomerHistory))
```

You need to view all customer data.

Which Transact-SQL statement should you run?

- A.

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

- B. SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address,
AnnualRevenue, DateCreated, ValidFrom, ValidTo
 FROM Customers
 FOR SYS_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 PivorCustomers
 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 00:00:00.0000000' AND '2015-01-01 00:00:00.0000000'
- H. SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address,
ValidFrom, ValidTo
 FROM Customers
 WHERE DateCreated
 BETWEEN '20140101' and '20141231'

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

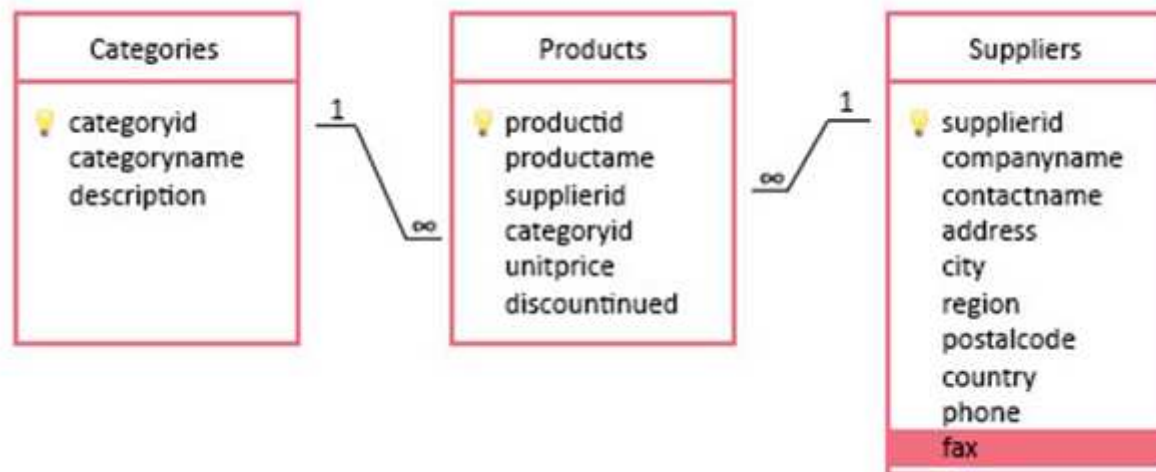
The FOR SYSTEM_TIME ALL clause returns all the row versions from both the Temporal and History table.

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

QUESTION 45

SIMULATION

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



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

Construct the query using the following guidelines:

- Use the first letter of the table name as the table alias.
- Use two-part column names.
- Do not surround object names with square brackets.
- Do not use implicit joins.
- Do not use variables.

- Use single quotes to surround literal values.

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

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

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

Correct Answer: Please see explanation

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

Note: On line 3 change * to =

QUESTION 46

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

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

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

You need to determine the total number of deposit and loan accounts.

Which Transact-SQL statement should you run?

- A. `SELECT COUNT(*)
FROM (SELECT AcctNo
FROM tblDepositAcct
INTERSECT
SELECT AcctNo
FROM tblLoanAcct) R`
- B. `SELECT COUNT(*)
FROM (SELECT CustNo
FROM tblDepositAcct
UNION
SELECT CustNo
FROM tblLoanAcct) R`

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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Explanation:

Would list the customers with duplicates, which would equal the number of accounts.

Incorrect Answers:

A: INTERSECT returns distinct rows that are output by both the left and right input queries operator.

B: Would list the customers without duplicates.

D: Number of customers.

F: EXCEPT returns distinct rows from the left input query that aren't output by the right input query.

References:

<https://msdn.microsoft.com/en-us/library/ms180026.aspx>

QUESTION 47

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

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

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

Sales.Customers

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

Application.Cities

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

Sales.CustomerCategories

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

Your company is developing a new social application that connects customers to each other based on the distance between their delivery locations.

You need to write a query that returns the nearest customer.

Solution: You run the following Transact-SQL statement:

```
SELECT TOP 1 B.CustomerID, A.DeliveryLocation.STDistance(B.DeliveryLocation) AS Dist
FROM Sales.Customers AS A
CROSS JOIN Sales.Customers AS B
WHERE A.CustomerID = @custID AND A.CustomerID <> B.CustomerID
ORDER BY Dist
```

The variable @custID is set to a valid customer.

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 48

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

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

Which Transact-SQL statement should you run?

- A. `CREATE TABLE Sales (
SalesID int IDENTITY(1,1),
SalesDate DateTime NOT NULL,
SalesAmount decimal(18,2) NULL
)`
- B. `CREATE TABLE Sales (
SalesID UNIQUEIDENTIFIER DEFAULT NEWSEQUENTIALID() PRIMARY KEY,
SalesDate DateTime2 NOT NULL,
SalesAmount money NULL
)`
- C. `CREATE TABLE Sales (
SalesID UNIQUEIDENTIFIER DEFAULT NEWSEQUENTIALID() PRIMARY KEY,
SalesDate DateTime2 NOT NULL,
SalesAmount decimal(18,2) NULL
)`
- D. `CREATE TABLE Sales (
SalesID int NOT NULL IDENTITY(1,1),
SalesDate DateTime2 NOT NULL,
SalesAmount decimal(18,4) NULL,
CONSTRAINT PK_SalesID PRIMARY KEY CLUSTERED (SalesID)
)`

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Explanation:

datetime2 Defines a date that is combined with a time of day that is based on 24-hour clock. datetime2 can be considered as an extension of the existing datetime type that has a larger date range, a larger default fractional precision, and optional user-specified precision.

Incorrect Answers:

B, C: NEWSEQUENTIALID creates a GUID that is greater than any GUID previously generated by this function on a specified computer since Windows was started. A GUID uses more space than IDENTITY value.

References:

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetime2-transact-sql>

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

QUESTION 49

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

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

Which query should you recommend?

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

C.

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

D.

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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

Incorrect Answers:

B: The count should be on the Cust instance of Sales.Customers as it is to the right side of the join.

C: Need a WHERE statement with an IS NULL clause.

D: Must use a LEFT JOIN to obtain the NULL values.

References: [https://technet.microsoft.com/en-us/library/ms190014\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190014(v=sql.105).aspx)

QUESTION 50

HOTSPOT

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

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

You query a database that includes two tables: Project and Task. The Project table includes the following columns:

Column name	Data type	Notes
ProjectId	int	This is a unique identifier for a project.
ProjectName	varchar(100)	
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the project is not finished yet.
UserId	int	Identifies the owner of the project.

The Task table includes the following columns:

Column name	Data type	Notes
TaskId	int	This is a unique identifier for a task.
TaskName	varchar(100)	A nonclustered index exists for this column.
ParentTaskId	int	Each task may or may not have a parent task.
ProjectId	int	A null value indicates the task is not assigned to a specific project.
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the task is not completed yet.
UserId	int	Identifies the owner of the task.

Users report performance issues when they run the following query:

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

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

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

Hot Area:

Answer Area

```
SELECT COUNT(*) AS TotalTestTasksCount FROM (
```

```
    SELECT T.TaskId, T.TaskName
```

```
    FROM Task T
```

```
    WHERE
```

	▼
T.TaskName	
LEFT(T.TaskName,4)	
RIGHT(T.TaskName,4)	
CHARINDEX('TEST', T.TaskName)	

```
    LIKE
```

	▼
'TEST'	
'TEST%'	
"%TEST'	
'%TEST%'	

```
    AND
```

	▼
T.EndTime IS NOT NULL	
T.StartTime = T.EndTime	

```
) AS R
```

Correct Answer:

Answer Area

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

|                               |   |
|-------------------------------|---|
|                               | ▼ |
| T.TaskName                    |   |
| LEFT(T.TaskName,4)            |   |
| RIGHT(T.TaskName,4)           |   |
| CHARINDEX('TEST', T.TaskName) |   |

 LIKE 

|          |   |
|----------|---|
|          | ▼ |
| 'TEST'   |   |
| 'TEST%'  |   |
| '%TEST'  |   |
| '%TEST%' |   |

  
    AND 

|                         |   |
|-------------------------|---|
|                         | ▼ |
| T.EndTime IS NOT NULL   |   |
| T.StartTime = T.EndTime |   |

  
    ) AS R
```

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

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

QUESTION 51

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 following records exist in the tables:

Customer_CRMSystem

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

Customer_HRSystem

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

Records that contain null values for CustomerCode can be uniquely identified by CustomerName.

You need to display distinct customers that appear in both tables.

Which Transact-SQL statement should you run?

- A.

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

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

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

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

- E. `SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
UNION
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem`
- F. `SELECT CustomerCode, CustomerName
FROM Customer_CRMSystem
UNION ALL
SELECT CustomerCode, CustomerName
FROM Customer_HRSystem`
- G. `SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
FROM Customer_CRMSystem c
CROSS JOIN Customer_HRSystem h`
- H. `SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
FROM Customer_CRMSystem c
FULL OUTER JOIN Customer_HRSystem h
ON c.CustomerCode = h.CustomerCode AND c.CustomerName = h.CustomerName`

Correct Answer: H

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

Incorrect Answers:

A: Inner joins return rows only when there is at least one row from both tables that matches the join condition. Inner joins eliminate the rows that do not match with a row from the other table.

B: INTERSECT returns distinct rows that are output by both the left and right input queries operator.

D: EXCEPT returns distinct rows from the left input query that aren't output by the right input query.

E: UNION specifies that multiple result sets are to be combined and returned as a single result set, but this will not work here as the CustomerID column values do not match.

F: UNION ALL incorporates all rows into the results. This includes duplicates. If not specified, duplicate rows are removed.

G: A cross join would produce the Cartesian product of the two tables.

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

QUESTION 52

DRAG DROP

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

You query a database that includes two tables: Project and Task. The Project table includes the following columns:

Column name	Data type	Notes
ProjectId	int	This is a unique identifier for a project.
ProjectName	varchar(100)	
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the project is not finished yet.
UserId	int	Identifies the owner of the project.

The Task table includes the following columns:

Column name	Data type	Notes
TaskId	int	This is a unique identifier for a task.
TaskName	varchar(100)	A nonclustered index exists for this column.
ParentTaskId	int	Each task may or may not have a parent task.
ProjectId	int	A null value indicates the task is not assigned to a specific project.
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the task is not completed yet.
UserId	int	Identifies the owner of the task.

You need to find all projects that have at least one task that took more than 50 hours to complete. You must also determine the average duration of the tasks that took more than 50 hours to complete for each project.

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.

Select and Place:

Transact-SQL segments

AVR(DATEDIFF(hh, T.StartTime, T.EndTime))

AVR(DATEDIFF(yy, T.StartTime, T.EndTime))

SUM(DATEDIFF(hh, T.StartTime, T.EndTime))/SU

DATEDIFF(hh, T.StartTime, T.EndTime)) > 50

DATEDADD(hh, 50, T.StartTime,) > T.EndTime

DATEADD(yy, -50, T.EndTime) <= T.StartTime

• • • •

Answer area

```
SELECT P.ProjectId, P.ProjectName, T.Summary.AvgDurationHours FROM Project P
OUTER APPLY
(
  SELECT  AS AvgDurationHours FROM Task T
  WHERE T.ProjectId = P.ProjectId
  AND 
) TSummary

WHERE T.Summary.AvgDurationHours IS NOT NULL
```

Correct Answer:

Transact-SQL segments

```
AVR(DATEDIFF(yy, T.StartTime, T.EndTime))
```

```
SUM(DATEDIFF(hh, T.StartTime, T.EndTime))/SU
```

```
DATEDIFF(hh, T.StartTime, T.EndTime)) > 50
```

```
DATEADD(yy, -50, T.EndTime) <= T.StartTime
```

....

Answer area

```
SELECT P.ProjectId, P.ProjectName, T.Summary.AvgDurationHours FROM Project P
OUTER APPLY
(
  SELECT AVR(DATEDIFF(hh, T.StartTime, T.EndTime)) AS AvgDurationHours FROM Task T
  WHERE T.ProjectId = P.ProjectId
  AND DATEDADD(hh, 50, T.StartTime, ) > T.EndTime
) TSummary

WHERE T.Summary.AvgDurationHours IS NOT NULL
```


Section: (none)

Explanation

Explanation/Reference:

QUESTION 53

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 CONTAINED IN ('2017-01-01');

C.

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

D.

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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

Incorrect Answers:

A, B: CONTAINED IN has two parameters: CONTAINED IN (<start_date_time> , <end_date_time>)

References: <https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-table>

QUESTION 54

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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

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)

QUESTION 55

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

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

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

```
CREATE TABLE Products (  
    ProductID int IDENTITY (1, 1), NOT NULL PRIMARY KEY,  
    ProductName nvarchar (100), NULL,  
    UnitPrice decimal (18, 2) NOT NULL,  
    UnitsInStock int NOT NULL,  
    UnitsOnOrder int NULL  
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct  
    @ProductName nvarchar(100),  
    @UnitPrice decimal (18, 2),  
    @UnitsInStock int,  
    @UnitsOnOrder int  
AS  
BEGIN  
    INSERT INTO Products (ProductName, UnitPrice, UnitsInStock, UnitsOnOrder)  
    VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)  
END
```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
@ProductName nvarchar (100),
@UnitPrice decimal (18, 2),
@UnitsInStock int,
@UnitsOnOrder int
AS
BEGIN
    SET XACT_ABORT ON
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products (ProductName, UnitPrice, UnitsInStock, UnitsOnOrder)
            VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF XACT_STATE () <> 0 ROLLBACK TRANSACTION
        THROW 51000, 'The product could not be created,' 1
    END CATCH
END
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 56

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

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

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

```
CREATE TABLE Products (  
    ProductID int IDENTITY (1, 1), NOT NULL PRIMARY KEY,  
    ProductName nvarchar (100), NULL,  
    UnitPrice decimal (18, 2) NOT NULL,  
    UnitsInStock int NOT NULL,  
    UnitsOnOrder int NULL  
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal (18, 2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    INSERT INTO Products (ProductName, UnitPrice, UnitsInStock, UnitsOnOrder)
    VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
@ProductName nvarchar (100),
@UnitPrice decimal (18, 2),
@UnitsInStock int,
@UnitsOnOrder int
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products (ProductName, UnitPrice, UnitsInStock, UnitsOnOrder)
            VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
        RAISERROR (51000,16, 1)
    END CATCH
END
```

Does the solution meet the goal?



<https://www.gratisexam.com/>

A. Yes

B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 57

You have a database that contains the following tables:

Customer

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

CustomerAudit

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

Where the value of the CustomerID column equals 3, you need to update the value of the CreditLimit column to 1000 for the customer. You must ensure that the change to the record in the Customer table is recorded on the CustomerAudit table.

Which Transact-SQL statement should you run?

A.

```
UPDATE Customer
SET CreditLimit= 1000
OUTPUT inserted. CustomerId, deleted. CreditLimit, deleted. CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit, ChangedBy)
WHERE CustomerId=3
```

B.

```
UPDATE Customer
SET CreditLimit= 1000
OUTPUT inserted. CustomerId, GETDATE (), deleted. CreditLimit, inserted. CreditLimit, SYSTEM_USER
INTO CustomerAudit (CustomerId, DateChanged, OldCreditLimit, NewCreditLimit, ChangedBy)
WHERE CustomerId=3
```

C.

```
UPDATE Customer
SET CreditLimit= 1000
WHERE CustomerId=3
INSERT INTO CustomerAudit (CustomerId, DateChanged, OldCreditLimit, NewCreditLimit,
ChangedBy)
SELECT CustomerId, GETDATE (), CreditLimit, CreditLimit, SYSTEM_USER
FROM Customer
WHERE CustomerID =3
```

D.

```
UPDATE Customer
SET CreditLimit= 1000
OUTPUT inserted. CustomerId, inserted. CreditLimit, inserted. CreditLimit
INTO CustomerAudit (CustomerId, OldCreditLimit, NewCreditLimit)
WHERE CustomerId=3
```

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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

QUESTION 58

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

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

You need to determine the total number of customers who have only deposit accounts.

Which Transact-SQL statement should you run?

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

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

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

References: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/set-operators-except-and-intersect-transact-sql?view=sql-server-2017>

QUESTION 59

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

Correct Answer: B
Section: (none)
Explanation

Explanation/Reference:

Explanation:

Mathematical equation will only return 10 % of the value.

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

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

You run the following query:

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

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

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

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

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 61

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

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

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

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

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

order.

Solution: You run the following Transact-SQL statement:

```
UPDATE Production.Products  
SET ListPrice = ListPrice * 1.1  
WHERE ListPrice  
BETWEEN .01 and 99.99
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

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

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 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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 63

You have a database named DB1 that contains a temporal table named Sales.Customers.

You need to create a query that returns the credit limit that was available to each customer in DB1 at the beginning of 2017.

Which query should you execute?

A.

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME CONTAINED IN ( '2017-01-01 ' );
```

B.

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

C.

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

D.

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME BETWEEN '2016-12-31' AND '2017-01-01');
```

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 64

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

that question.

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

You need to create a query that generates sample data for a sales table in the database. The query must include every product in the inventory for each customer.

Which statement clause should you use?

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

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

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

QUESTION 65

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

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

All the sales data is stored in a table named table1. You have a table named table2 that contains city names.

You need to create a query that lists only the cities that have no sales.

Which statement clause should you add to the query?

- A. GROUP BY
- B. MERGE

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

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

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

QUESTION 66

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

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

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

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

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

CountryName	StateProvinceName	CityName	TotalSales
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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

QUESTION 67

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

You query a database that includes two tables: Project and Task. The Project table includes the following columns:

Column name	Data type	Notes
ProjectId	int	This is a unique identifier for a project.
ProjectName	varchar(100)	
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the project is not finished yet.
UserId	int	Identifies the owner of the project.

The Task table includes the following columns:

Column name	Data type	Notes
TaskId	int	This is a unique identifier for a task.
TaskName	varchar(100)	A nonclustered index exists for this column.
ParentTaskId	int	Each task may or may not have a parent task.
ProjectId	int	A null value indicates the task is not assigned to a specific project.
StartTime	datetime2(7)	
EndTime	datetime2(7)	A null value indicates the task is not completed yet.
UserId	int	Identifies the owner of the task.

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

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

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

What set of Transact-SQL statements should you run?

A.

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

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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 68

You have a database named DB1 that contains a temporal table named Sales.Customers.

You need to create a query that returns the credit limit that was available to each customer in DB1 at the beginning of 2017.

Which query should you execute?

- A. `SELECT`
`CustomerID,`
`CustomerName,`
`CreditLimit`
`FROM`
`Sales.Customers`
`FOR SYSTEM_TIME CONTAINED IN ('2017-01-01 00:00:00');`

- B. `SELECT`
 CustomerID,
 CustomerName,
 CreditLimit
`FROM`
 Sales.Customers
 FOR SYSTEM_TIME AS OF '2017-01-01 00:00:00';
- C. `SELECT`
 CustomerID,
 CustomerName,
 CreditLimit
`FROM`
 Sales.Customers
 FOR SYSTEM_TIME CONTAINED IN ('2016-12-31', '2017-01-01');
- D. `SELECT`
 CustomerID,
 CustomerName,
 CreditLimit
`FROM`
 Sales.Customers
 FOR SYSTEM_TIME BETWEEN '2016-12-31' AND '2017-01-01');

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

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

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

You are developing a report that aggregates customer data only for the year 2014. The report requires that the data be denormalized.

You need to return the data for the report.

Which Transact-SQL statement should you run?

- A.

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

```
SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, AnnualRevenue, DateCreated, ValidFrom, ValidTo  
FROM Customers  
FOR SYSTEM_TIME ALL ORDER BY ValidFrom
```

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

Correct Answer: G
Section: (none)

Explanation

Explanation/Reference:

QUESTION 70

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

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

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

You need to 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'
```

Correct Answer: G

Section: (none)

Explanation

Explanation/Reference:

QUESTION 71

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 run 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 the total annual revenue for all customers, followed by a row for each customer that shows the customer's name and annual revenue.

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'`

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

QUESTION 72

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

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

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

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

- Be indexable
- Contain up-to-date statistics
- Be able to scale between 10 and 100,000 rows

The solution must prevent users from accessing one another's data.

Solution: You create a table variable in the stored procedure.

Does this meet the goal?

- A. Yes
B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

QUESTION 73

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 hash index on the primary key column.

Does this meet the goal?

A. Yes

B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Reference: <https://msdn.microsoft.com/en-us/library/dn133190.aspx>

QUESTION 74

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

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

You 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

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

QUESTION 75

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

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

You are creating indexes in a data warehouse.

You have a dimension table named Table1 that has 10,000 rows. The rows are used to generate several reports.

The reports join a column that is the primary key.

The execution plan contains bookmark lookups for Table1.

You discover that the reports run slower than expected.

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

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

Does this meet the goal?

A. Yes

B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:



<https://www.gratisexam.com/>