

Microsoft

Exam Questions 70-761

Querying Data with Transact-SQL (beta)



NEW QUESTION 1

Note: This question is part of a series of questions that use the same 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	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
```

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

Answer: H

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.

NEW QUESTION 2

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

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

The tables include the following records: Customer_CRMSystem

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

Customer_HRSystem

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

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

You need to create a list of all unique customers that appear in either table. Which Transact-SQL statement should you run?

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

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

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

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

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

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

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

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

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

H. Option H

Answer: E

Explanation:

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

NEW QUESTION 3

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

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

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

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

Be indexable

Contain up-to-date statistics

Be able to scale between 10 and 100,000 rows

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

Does this meet the goal?

A. Yes

B. No

Answer: B

NEW QUESTION 4

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

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

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

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

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

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

TotalUnitPrice is calculated by using the following formula: TotalUnitPrice = UnitPrice * (UnitsInStock + UnitsOnOrder)

You need to ensure that the value returned for TotalUnitPrice for ProductB is equal to 600.00. Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

A. Yes

B. No

Answer: B

Explanation:

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

NEW QUESTION 5

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

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

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

Statements

Answer Area

```
SELECT Employees.ManagerId, Employees.EmployeeId,
EmployeeLevel+1
FROM Employees
JOIN Managers ON Employees.EmployeeId =
Managers.ManagerId)
```

```
WITH Managers AS (
```

```
SELECT*
FROM Managers
ORDER BY ManagerID
```

```
SELECT ManagerId, EmployeeId, 0 AS EmployeeLevel
FROM Employees
WHERE ManagerId IS NULL
```

```
UNION ALL
```

```
UNION
```



- A. Mastered
B. Not Mastered

Answer: A

Explanation:

References:

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

NEW QUESTION 6

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

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

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

The following table describes the columns in Sales.Orders.

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

The following table describes the columns in Sales.OrderLines.

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

You need to create a function that calculates the highest tax rate charged for an item in a specific order. Which five Transact-SQL segments should you use to develop the solution? To answer, move the appropriate Transact-SQL segments from the list of Transact-SQL segments to the answer area and arrange them in the correct order.

Transact-SQL segments

RETURNS decimal(18,2)

CREATE FUNCTION Sales.CalculateTaxRate ()

CREATE FUNCTION Sales.CalculateTaxRate (
 @OrderID int
)

RETURN @CalculatedRate
END

SET @CalculatedTaxRate = (
 SELECT 1 + (MAX(TaxRate)
 / 100)
 FROM Sales.OrderLines
 WHERE OrderID = @OrderID

RETURNS Table
END

AS
BEGIN
declare @CalculatedTaxRate
decimal(18,2)

Answer Area

⏪

⏩

⏴

⏵

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

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

NEW QUESTION 7

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
 After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
 You are creating indexes in a data warehouse.
 You have a dimension table named Table1 that has 10,000 rows. The rows are used to generate several reports. The reports join a column that is the primary key. The execution plan contains bookmark lookups for Table1. You discover that the reports run slower than expected.
 You need to reduce the amount of time it takes to run the reports.
 Solution: You create a nonclustered index on the primary key column that includes the bookmark lookup columns.

Does this meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 8

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

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

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

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

- A**

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

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

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

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

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

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

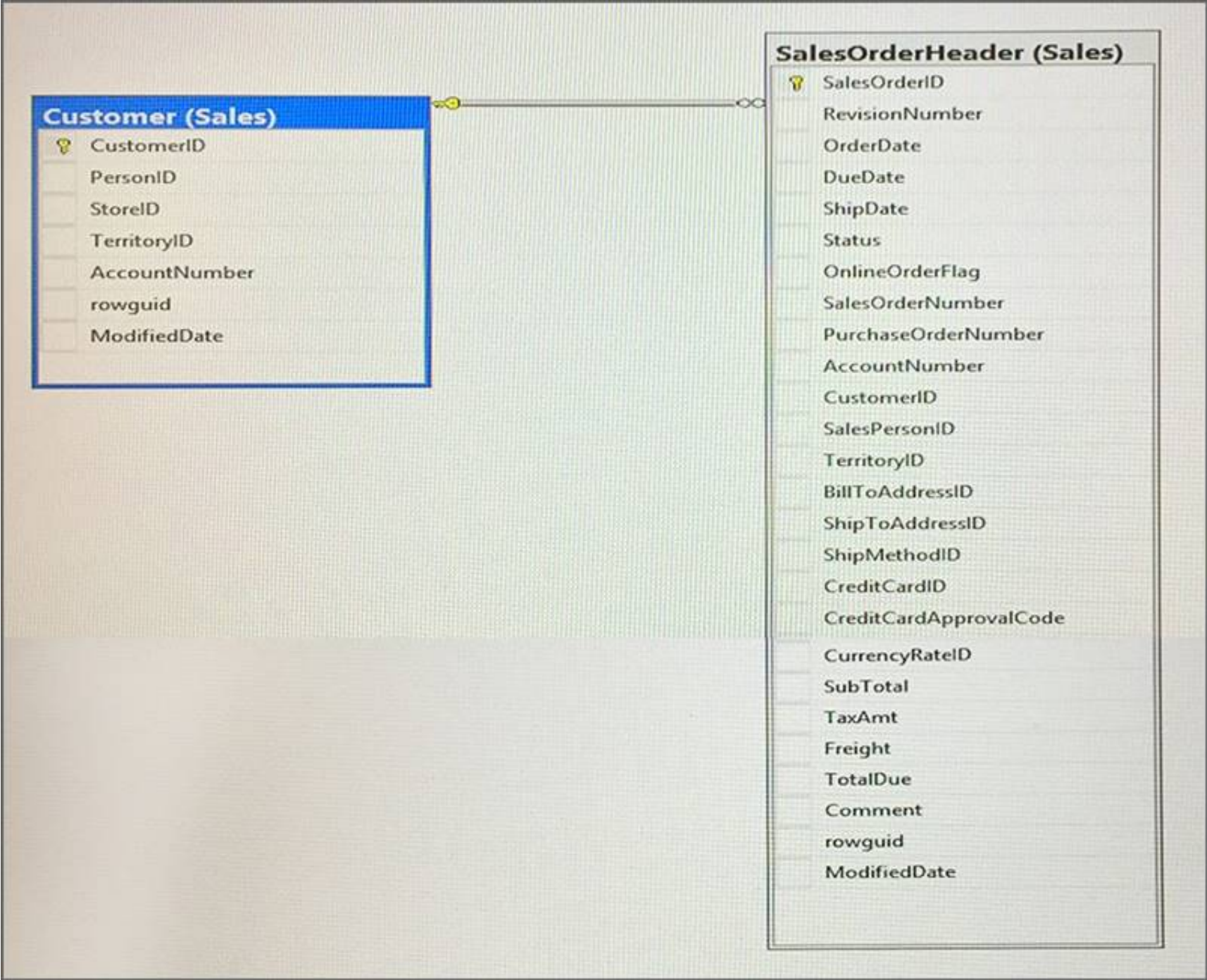
Answer: G

Explanation:

The following query searches for row versions for Employee row with EmployeeID = 1000 that were active at least for a portion of period between 1st January of

2014 and 1st January 2015 (including the upper boundary):
 SELECT * FROM Employee FOR SYSTEM_TIME
 BETWEEN '2014-01-01 00:00:00.0000000' AND '2015-01-01 00:00:00.0000000'
 WHERE EmployeeID = 1000 ORDER BY ValidFrom;
 References: <https://msdn.microsoft.com/en-us/library/dn935015.aspx>

NEW QUESTION 9
 You have a database that includes the tables shown in the exhibit. (Click the exhibit button.)



You need to create a list of all customers and the date that the customer placed their last order. For customers who have not placed orders, you must substitute a zero for the order ID and 01/01/1990 for the date.
 Which Transact-SQL statement should you run?

A

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

B

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

C

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

D

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

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

Answer: A**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>

NEW QUESTION 10

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

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

Answer: G

NEW QUESTION 10

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

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

You create a table named 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, ProductPrice, ProductsInStock, ProductsOnOrder)  
    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, ProductPrice, ProductsInStock, ProductsOnOrder)  
            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

Answer: B

Explanation:

With X_ABORT ON the INSERT INTO statement and the transaction will be rolled back when an error is raised, it would then not be possible to ROLLBACK it again in the IF XACT_STATE() <> 0 ROLLACK TRANSACTION statement.

Note: A transaction is correctly defined for the INSERT INTO ..VALUES statement, and if there is an error in the transaction it will be caught ant he transaction will be rolled back, finally an error 51000 will be raised.

Note: When SET XACT_ABORT is ON, if a Transact-SQL statement raises a run-time error, the entire transaction is terminated and rolled back.

XACT_STATE is a scalar function that reports the user transaction state of a current running request. XACT_STATE indicates whether the request has an active user transaction, and whether the transaction is capable of being committed.

The states of XACT_STATE are:

0 There is no active user transaction for the current request.

1 The current request has an active user transaction. The request can perform any actions, including writing data and committing the transaction.

2 The current request has an active user transaction, but an error has occurred that has caused the transaction to be classified as an committable transaction.

References:

<https://msdn.microsoft.com/en-us/library/ms188792.aspx> <https://msdn.microsoft.com/en-us/library/ms189797.aspx>

NEW QUESTION 15

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

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

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

Answer: AC

NEW QUESTION 18

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

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

For reporting purposes, you need to find the active user count for each role, and the total active user count. The result must be ordered by active user count of each role. You must use common table expressions (CTEs).

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

Transact-SQL segments

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

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

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

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

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

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

Answer Area



- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Transact-SQL segments

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

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

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

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

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

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

Answer Area

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

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

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



NEW QUESTION 21

You work for an organization that monitors seismic activity around volcanos. You have a table named GroundSensors. The table stored data collected from seismic sensors. It includes the columns describes in the following table:

Name	Data Type	Notes
SensorID	int	primary key
Location	geography	do not allow null values
Tremor	int	do not allow null values
NormalizedReading	float	allow null values

The database also contains a scalar value function named NearestMountain that returns the name of the mountain that is nearest to the sensor. You need to create a query that shows the average of the normalized readings from the sensors for each mountain. The query must meet the following requirements:

- * Include the average normalized readings and nearest mountain name.
- * Exclude sensors for which no normalized reading exists.
- * Exclude those sensors with value of zero for tremor. Construct the query using the following guidelines:
- * Use one part names to reference tables, columns and functions.
- * Do not use parentheses unless required.
- * Do not use aliases for column names and table names.
- * Do not surround object names with square brackets.

Keywords

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

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

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

```
1 select
```

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

GROUP BY is a SELECT statement clause that divides the query result into groups of rows, usually for the purpose of performing one or more aggregations on each group. The SELECT statement returns one row per group.

SELECT SensorID, NearestMountain(Location) FROM GroundSensors

WHERE TREMOR < 0 AND NormalizedReading IS NOT NULL

GROUP BY SensorID, NearestMountain(Location)

References: <https://msdn.microsoft.com/en-us/library/ms177673.aspx>

NEW QUESTION 26

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

You have a database that is denormalized. Users make frequent changes to data in a primary table.

You need to ensure that users cannot change the tables directly, and that changes made to the primary table also update any related tables.

What should you implement?

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

Answer: B

Explanation:

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

References:

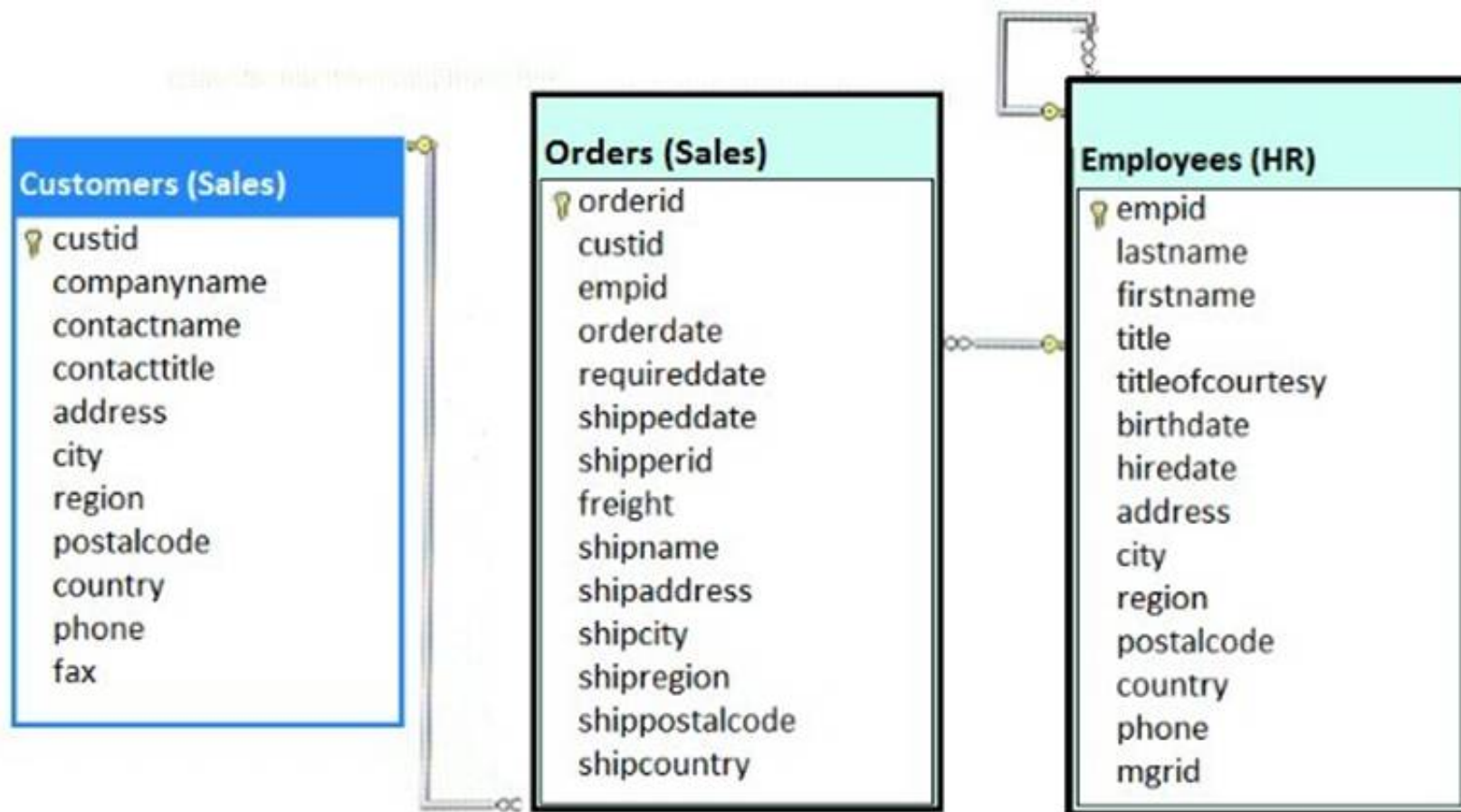
<http://stackoverflow.com/questions/4789091/updating-redundant-denormalized-data-automatically-in-sql-server>

NEW QUESTION 28

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

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

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



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

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

The output must be sorted by order date, with the newest orders first. The solution must return only the most recent order for each customer. Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

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

NEW QUESTION 29

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

Answer: B

NEW QUESTION 33

You need to create a stored procedure to update a table named Sales.Customers. The structure of the table is shown in the exhibit. (Click the exhibit button.)

Sales.Customers	
Columns	
	custid (PK, int, not null)
	companyname (nvarchar(40), not null)
	contactname (nvarchar(30), not null)
	contacttitle (nvarchar(30), not null)
	address (nvarchar(60), not null)
	city (nvarchar(15), not null)
	region (nvarchar(15), null)
	postalcode (nvarchar(10), null)
	country (nvarchar(15), not null)
	phone (nvarchar(24), not null)
	fax (nvarchar(24), null)

The stored procedure must meet the following requirements:

- Accept two input parameters.
- Update the company name if the customer exists.
- Return a custom error message if the customer does not exist.

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

NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.

Transact-SQL segments

CREATE PROCEDURE Sales.ModCompanyName
@custID int, @newname nvarchar(40) AS

IF NOT EXISTS (SELECT custid FROM
Sales.Customers WHERE custid = @custID)

UPDATE Sales.Customers
SET companyname = @newname
WHERE custid = @custID

BEGIN THROW 55555, 'The customer ID
does not exist', 1 END

UPDATE Sales.Customers
SET companyname = @custID
WHERE custid = @newname

IF EXISTS (SELECT custid FROM
Sales.Customers
WHERE custid = @custID)

ROLLBACK TRANSACTION

Answer Area

⏪

⏩

⏴

⏵

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Transact-SQL segments

CREATE PROCEDURE Sales.ModCompanyName
 @custID int, @newname nvarchar(40) AS

IF NOT EXISTS (SELECT custid FROM
 Sales.Customers WHERE custid = @custID)

UPDATE Sales.Customers
 SET companyname = @newname
 WHERE custid = @custID

BEGIN THROW 55555, 'The customer ID
 does not exist', 1 END

UPDATE Sales.Customers
 SET companyname = @custID
 WHERE custid = @newname

IF EXISTS (SELECT custid FROM
 Sales.Customers
 WHERE custid = @custID)

ROLLBACK TRANSACTION

Answer Area

CREATE PROCEDURE Sales.ModCompanyName
 @custID int, @newname nvarchar(40) AS

IF EXISTS (SELECT custid FROM
 Sales.Customers
 WHERE custid = @custID)

UPDATE Sales.Customers
 SET companyname = @newname
 WHERE custid = @custID

IF NOT EXISTS (SELECT custid FROM
 Sales.Customers WHERE custid = @custID)

BEGIN THROW 55555, 'The customer ID
 does not exist', 1 END

NEW QUESTION 38

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.

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.

When running an operation, you updated a column named EndTime for several records in the Project table, but updates to the corresponding task records in the Task table failed.

You need to synchronize the value of the EndTime column in the Task table with the value of the EndTime column in the project table. The solution must meet the following requirements:

* If the EndTime column has a value, make no changes to the record.

* If the value of the EndTime column is null and the corresponding project record is marked as completed, update the record with the project finish time.

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

Transact-SQL segments

FROM Project AS P

WHERE P.EndTime IS NOT NULL AND
T.EndTime is NULL

FROM Task AS T

WHERE P.EndTime IS NULL AND T.EndTime
IS NOT NULL

UPDATE T SET T.EndTime = P.EndTime

INNER JOIN Project AS P ON T.ProjectId
= P.ProjectId

INNER JOIN Task AS T ON T.UserId =
P.UserId

UPDATE P SET P.EndTime = T.EndTime

Answer Area

⏪

⏩

⏴

⏵

- A. Mastered
- B. Not Mastered


Answer: A


Explanation:


Box 1: UPDATE T SET T.EndTime = P.EndTime
We are updating the EndTime column in the Task table. Box 2: FROM Task AS T
Where are updating the task table.
Box 3: INNER JOIN Project AS P on T.ProjectID = P.ProjectID
We join with the Project table (on the ProjectID columnID column). Box 4: WHERE P.EndTime is NOT NULL AND T.EndTime is NULL
We select the columns in the Task Table where the EndTime column in the Project table has a value (NOT NULL), but where it is NULL in the Task Table.
References: <https://msdn.microsoft.com/en-us/library/ms177523.aspx>

NEW QUESTION 43

Note: This question is part of a series of questions that use the same scenario. For your convenience, the scenario is repeated in each question. Each question presents a different goal and answer choices, but the text of the scenario is exactly the same in each question in this series.
Start of repeated scenario
You have a database that contains the tables shown in the exhibit. (Click the Exhibit button.)

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



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

You review the Employee table and make the following observations:

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

- Sales data is only recorded for sales representatives.

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

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

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

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

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

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

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

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

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

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

End of Repeated Scenario

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

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

Answer area

Column

Function

MiddleName

▼

NULLIF

REPLACE

COALESCE

RegionCode

▼

NULLIF

REPLACE

COALESCE

A. Mastered

B. Not Mastered

Answer: A

Explanation:

Box 1: COALESCE

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

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

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

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

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

Box 2: COALESCE

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

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

NEW QUESTION 48

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

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

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

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

Be indexable

Contain up-to-date statistics

Be able to scale between 10 and 100,000 rows

The solution must prevent users from accessing one another's data. Solution: You create a local temporary table in the stored procedure. Does this meet the goal?

- A. Yes
- B. No

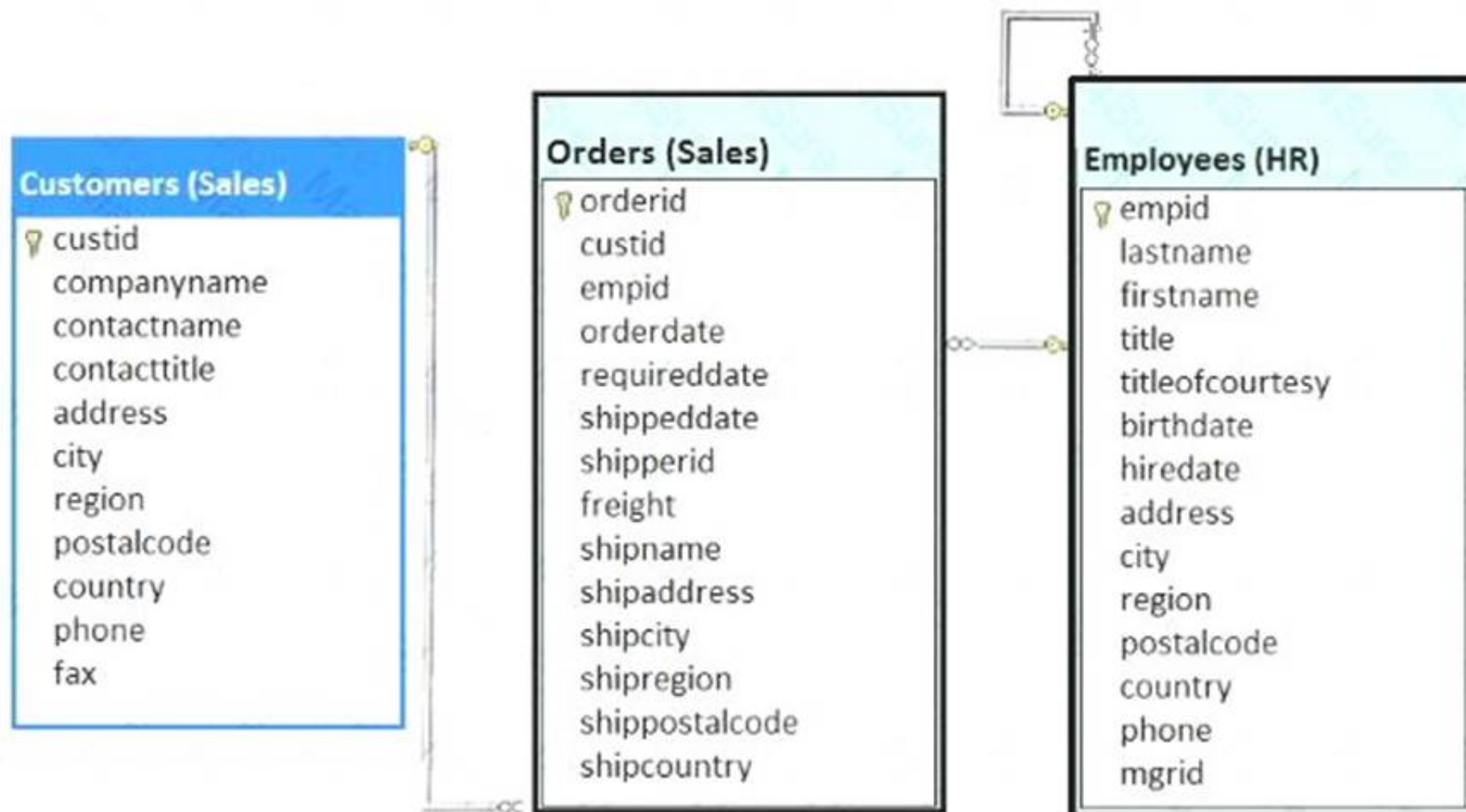
Answer: B

NEW QUESTION 50

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

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

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



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

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

The output must be sorted by order date, with the newest orders first. The solution must return only the most recent order for each customer. Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: A

Explanation:

The MAX(orderdate) in the SELECT statement makes sure we return only the most recent order. A WHERE o.empid =4 clause is correctly used. GROUP BY is also required.

NEW QUESTION 54

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

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

All the sales data is stored in a table named table1. You have a table named table2 that contains city names. You need to create a query that lists only the cities that have no sales.

Which statement clause should you add to the query?

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

Answer: D

NEW QUESTION 57

You create a table named Sales.Orders by running the following Transact-SQL statement:

```
CREATE TABLE Sales.Orders (  
    OrderID int NOT NULL,  
    OrderDate date NULL,  
    ShippedDate date NULL,  
    Status varchar(10),  
    CONSTRAINT PK_ORDERS PRIMARY KEY CLUSTERED OrderID  
)
```

You need to write a query that removes orders from the table that have a Status of Canceled. Construct the query using the following guidelines:

Keywords

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

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

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

```
1 DELETE from sales.orders where status='calceled'
```

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

1. DELETE from sales.orders where status='Canceled' Note: On line 1 change calceled to Canceled

Example: Using the WHERE clause to delete a set of rows

The following example deletes all rows from the ProductCostHistory table in the AdventureWorks2012 database in which the value in the StandardCost column is more than 1000.00.

DELETE FROM Production.ProductCostHistory WHERE StandardCost > 1000.00;

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

NEW QUESTION 58

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

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

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

The tables include the following records: Customer_CRMSystem

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

Customer_HRSystem

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

Records that contain null values for CustomerCode can be uniquely identified by CustomerName. You need to display a 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

Answer: D

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>

NEW QUESTION 60

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

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

You 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?

- A. Yes
- B. No

Answer: B

NEW QUESTION 62

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

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

Which query should you execute?

A

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

B

```
SELECT
    CustomerID,
    CustomerName,
    CreditLimit
FROM
    Sales.Customers
    FOR SYSTEM_TIME AS OF '2017-01-01 00:00:00';
```

C

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

D

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

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

Answer: B

NEW QUESTION 65

You have the following Transact-SQL statement: DELETE FROM Person
WHERE PersonID = 5

You need to implement error handling.

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

NOTE: Each correct selection is worth one point.

BEGIN TRANSACTION

▼
END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

DELETE FROM Person
WHERE PersonID = 5

▼
END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

▼
END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

IF @@TRANCOUNT > 0

▼
END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

▼
END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

IF @@TRANCOUNT > 0
COMMIT TRANSACTION

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

BEGIN TRANSACTION

END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

DELETE FROM Person
WHERE PersonID = 5

END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

IF @@TRANCOUNT > 0

END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

END TRY
COMMIT TRANSACTION
END CATCH
BEGIN TRY
ROLLBACK TRANSACTION
BEGIN CATCH

IF @@TRANCOUNT > 0
COMMIT TRANSACTION

NEW QUESTION 66

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

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

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

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

Solution: You recommend the following query:

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

Does this meet the goal?

- A. Yes
- B. No

Answer: B

NEW QUESTION 68

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

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

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

The tables include the following records: Customer_CRMSystem

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

Customer_HRSystem

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

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

- A SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
 FROM Customer_CRMSystem c
 INNER JOIN Customer_HRSystem h
 ON c.CustomerCode = h.CustomerCode AND c.CustomerName = h.CustomerName
- B SELECT CustomerCode, CustomerName
 FROM Customer_CRMSystem
 INTERSECT
 SELECT CustomerCode, CustomerName
 FROM Customer_HRSystem
- C SELECT c.CustomerCode, c.CustomerName
 FROM Customer_CRMSystem c
 LEFT OUTER JOIN Customer_HRSystem h
 ON c.CustomerCode = h.CustomerCode
 WHERE h.CustomerCode IS NULL AND c.CustomerCode IS NOT NULL
- D SELECT CustomerCode, CustomerName
 FROM Customer_CRMSystem
 EXCEPT
 SELECT CustomerCode, CustomerName
 FROM Customer_HRSystem
- E SELECT CustomerCode, CustomerName
 FROM Customer_CRMSystem
 UNION
 SELECT CustomerCode, CustomerName
 FROM Customer_HRSystem
- F SELECT CustomerCode, CustomerName
 FROM Customer_CRMSystem
 UNION ALL
 SELECT CustomerCode, CustomerName
 FROM Customer_HRSystem
- G SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
 FROM Customer_CRMSystem c
 CROSS JOIN Customer_HRSystem h
- H SELECT c.CustomerCode, c.CustomerName, h.CustomerCode, h.CustomerName
 FROM Customer_CRMSystem c
 FULL OUTER JOIN Customer_HRSystem h
 ON c.CustomerCode = h.CustomerCode AND c.CustomerName = h.CustomerName

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

Answer: G

Explanation:

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

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

NEW QUESTION 72

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

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

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

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

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

Which Transact-SQL statement should you run?

- A** `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, AnnualRevenue, DateCreated
FROM Customers
GROUP BY GROUPING SETS((FirstName, LastName), (Address), (CustomerID, AnnualRevenue), (CustomerID), ())
ORDER BY CustomerID, FirstName, LastName, Address, AnnualRevenue`
- B** `SELECT FirstName, LastName, Address
FROM Customers
FOR SYSTEM_TIME ALL ORDER BY ValidFrom`
- C** `SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
FROM Customers AS c
ORDER BY c.CustomerID
FOR JSON AUTO, ROOT('Customers')`
- D** `SELECT * FROM (SELECT CustomerID, FirstName, LastName, Address, AnnualRevenue, DateCreated
FROM Customers) AS Customers PIVOT(AVG(AnnualRevenue)
FOR DateCreated IN([2014])) AS PivotCustomers
ORDER BY LastName, FirstName`
- E** `SELECT CustomerID, AVG(AnnualRevenue)
AS AverageAnnualRevenue, FirstName, LastName, Address, DateCreated
FROM Customers WHERE YEAR(DateCreated) >= 2014
GROUP BY CustomerID, FirstName, LastName, Address, DateCreated`

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

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

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

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

Answer: F

Explanation:

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

* The EXPLICIT mode allows more control over the shape of the XML. You can mix attributes and elements at will in deciding the shape of the XML. It requires a specific format for the resulting rowset that is generated because of query execution. This rowset format is then mapped into XML shape. The power of EXPLICIT mode is to mix attributes and elements at will, create wrappers and nested complex properties, create

space-separated values (for example, OrderID attribute may have a list of order ID values), and mixed contents.

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

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

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

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

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

Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

Mathematical equation will only return 10 % of the value.

NEW QUESTION 76

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

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

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


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

You must insert the following data into the Customer table:

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

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

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

As there are two separate INSERT INTO statements we cannot ensure that both or neither records is inserted.

NEW QUESTION 78

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

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

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

The following table describes the columns in Sales.Orders.

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

The following table describes the columns in Sales.OrderLines.

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

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

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

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

Transact-SQL segments

COUNT

SUM

AVG

ORDER BY

GROUP BY

RETURNS INT

RETURNS NULL ON NULL INPUT

RETURNS TABLE

Answer Area

```

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

- A. Mastered
 B. Not Mastered

Answer: A

Explanation:

Box1: RETURNS TABLE

The function should return the following information:

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

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

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

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

Need to use GROUP BY for the aggregate functions.

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

NEW QUESTION 82

You have a date related query that would benefit from an indexed view. You need to create the indexed view.

Which two Transact-SQL functions can you use? Each correct answer presents a complete solution. NOTE: Each correct selection is worth one point

- A. DATEADD
 B. AT TIME ZONE
 C. GETUTCDATE

D. DATEDIFF

Answer: A

NEW QUESTION 87

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

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

You are creating indexes in a data warehouse.

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

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

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

A. Yes

B. No

Answer: A

NEW QUESTION 92

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

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

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

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

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

Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

A. Yes

B. No

Answer: B

NEW QUESTION 96

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.

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.

Task level is defined using the following rules:

Task category	Task level definition
Tasks that have no parent task	[Task Level] = 0
Tasks that have a parent task	[Task Level] = [Parent Task's Level] + 1

You need to determine the task level for each task in the hierarchy.
 Which five Transact-SQL segments should you use to develop the solution? To answer, move the appropriate Transact-SQL segments from the list of Transact-SQL segments to the answer area and arrange them in the correct order.

Transact-SQL segments

)
 SELECT * FROM TaskWithLevel

SELECT CAST(NULL AS int) AS ParentTaskId, T.TaskId, T.TaskName, 0 AS TaskLevel
FROM Task T WHERE T.ParentTaskId IS NULL

With TaskWithLevel (ParentTaskId, TaskId, TaskName, TaskLevel)
As (

UNION

SELECT R.TaskId AS ParentTaskId, T.TaskId, T.TaskName, R.TaskLevel+1 AS TaskLevel
FROM Task T INNER JOIN TaskWithLevel R ON T.ParentTaskId = R.TaskId

SELECT T.TaskId AS ParentTaskId, CAST(null AS int) AS TaskId, T.TaskName, 0 AS TaskLevel
FROM Task T WHERE T.ParentTaskId IS NULL

UNION ALL

Answer Area

⏪

⏩

⏴

⏵

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:
 Box 1: SELECT CAST (NULL AS INT) AS ParentTaskID, etc.
 This statement selects all tasks with task level 0.
 The ParentTaskID could be null so we should use CAST (NULL AS INT) AS ParentTaskID. Box 2: UNION
 We should use UNION and not UNION ALL as we do not want duplicate rows.
 UNION specifies that multiple result sets are to be combined and returned as a single result set.
 Incorrect: Not UNION ALL: ALL incorporates all rows into the results. This includes duplicates. If not specified, duplicate rows are removed.
 Box 3, Box 4, Box 5:
 These statements select all tasks with task level >0. References:
<https://msdn.microsoft.com/en-us/library/ms180026.aspx>

NEW QUESTION 100

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.

Passing Certification Exams Made Easy

visit - <https://www.surepassexam.com>

- ALL SELECT statements must specify columns.
- Do not use column or table aliases, except those provided.
- Do not surround object names with square brackets.

Keywords

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

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

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

```
1 SELECT CityID, QuestionID, RawCount
2 AS t1
3 AS t2
4 JOIN
```

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

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

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

NEW QUESTION 103

You are building a stored procedure named SP1 that calls a stored procedure named SP2. SP2 calls another stored procedure named SP3 that returns a Recordset. The Recordset is stored in a temporary table. You need to ensure that SP2 returns a text value to SP1. What should you do?

- A. Create a temporary table in SP2, and then insert the text value into the table.
- B. Return the text value by using the ReturnValue when SP2 is called.
- C. Add the txt value to an OUTPUT parameter of SP2.
- D. Create a table variable in SP2, and then insert the text value into the table.

Answer: C

NEW QUESTION 104

.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questions and Answers in PDF Format

70-761 Practice Exam Features:

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

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