

70-761.100q

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



Website: <https://vceplus.com>
VCE to PDF Converter: <https://vceplus.com/vce-to-pdf/>
Facebook: <https://www.facebook.com/VCE.For.All.VN/>
Twitter : https://twitter.com/VCE_Plus

<https://vceplus.com/>

70-761

Querying Data with Transact-SQL

Exam A

QUESTION 1

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

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

You create a table named 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
```



<https://vceplus.com/>

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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

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 ROLLBACK 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 and the 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 uncommittable transaction.

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

QUESTION 2

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution. After you answer a question in this section. You will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You 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
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products(ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
            VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
        IF @@ERROR = 51000
            THROW
    END CATCH
END
```



Does the solution meet the goal?

A. Yes

B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

A transaction is correctly defined for the INSERT INTO .VALUES statement, and if there is an error in the transaction it will be caught and the transaction will be rolled back. However, error number 51000 will not be returned, as it is only used in an IF @ERROR = 51000 statement.

Note: @@TRANSCOUNT returns the number of BEGIN TRANSACTION statements that have occurred on the current connection.

References: <https://msdn.microsoft.com/en-us/library/ms187967.aspx>

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 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
    BEGIN TRY
        INSERT INTO Products(ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
        VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
    END TRY
    BEGIN CATCH
        THROW 51000, 'The product could not be created.', 1
    END CATCH
END
```



Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

If the INSERT INTO statement raises an error, the statement will be caught and an error 51000 will be thrown. In this case no records will have been inserted.

Note:

You can implement error handling for the INSERT statement by specifying the statement in a TRY...CATCH construct.

If an INSERT statement violates a constraint or rule, or if it has a value incompatible with the data type of the column, the statement fails and an error message is returned.

References: <https://msdn.microsoft.com/en-us/library/ms174335.aspx>

QUESTION 4

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

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

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

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

You must insert the following data into the Customer table:

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

You need to ensure that both records are inserted or neither record is inserted.

Solution: You run the following Transact-SQL statement:

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



Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

QUESTION 5

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

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

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

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

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

QUESTION 6

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

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

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

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

You must insert the following data into the Customer table:

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

You need to ensure that both records are inserted or neither record is inserted.

Solution: You run the following Transact-SQL statement:

```
INSERT INTO dbo.Customer (FirstName, LastName, DateOfBirth, CreditLimit)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000), ('Jossef', 'Goldberg', '1995-06-03', 5500)
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Explanation:

With the INSERT INTO..VALUES statement we can insert both values with just one statement. This ensures that both records or neither is inserted.

References: <https://msdn.microsoft.com/en-us/library/ms174335.aspx>

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

Sales.Customers

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

Application.Cities

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

Sales.CustomerCategories

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

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

The main page of the application will be based on an indexed view that contains the area and phone number for all customers. You need to return the area code from the PhoneNumber field.

Solution: You run the following Transact-SQL statement:

```
CREATE FUNCTION AreaCode (  
    @phoneNumber nvarchar(20)  
)  
RETURNS  
TABLE  
WITH SCHEMABINDING  
AS  
RETURN (  
    SELECT TOP 1 @phoneNumber as PhoneNumber, VALUE as AreaCode  
    FROM STRING_SPLIT(@phoneNumber, '-')  
)
```



Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

The function should return nvarchar(10) and not a TABLE.

References: <https://sqlstudies.com/2014/08/06/schemabinding-what-why/>

QUESTION 8

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

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

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

Sales.Customers

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

Application.Cities

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

Sales.CustomerCategories

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

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

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

You need to return the area code from the PhoneNumber field.

Solution: You run the following Transact-SQL statement:

```
CREATE FUNCTION AreaCode (  
    @phoneNumber nvarchar(20)  
)  
{  
    RETURNS nvarchar(10)  
    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: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

As the result of the function will be used in an indexed view we should use schemabinding.

References: <https://sqlstudies.com/2014/08/06/schemabinding-what-why/>

QUESTION 9

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

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

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

Sales.Customers

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

Application.Cities

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

Sales.CustomerCategories

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

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

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

You need to return the area code from the PhoneNumber field.

Solution: You run the following Transact-SQL statement:

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



<https://vceplus.com/>

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

We need SELECT TOP 1 @areacode =.. to ensure that only one value is returned.

QUESTION 10

You need to create a database object that meets the following requirements:

- accepts a product identified 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. an extended stored procedure
- B. a user-defined scalar function
- C. a user-defined stored procedure that has an OUTPUT parameter
- D. a temporary table that has a columnstore index

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

User-defined scalar functions are execution plans that accept parameters, perform an action such as a complex calculation, and returns the result of that action as a value. The return value can either be a single scalar value or a result set. Furthermore the execution plan is cached and reusable.

User-defined scalar functions can also be called from within a SELECT statement and can be used in a JOIN clause.

Incorrect Answers:

- A: Using extended stored procedures is not recommended as they has been deprecated. CLR Integration should be used instead of extended stored procedures.
- C: Stored procedures cannot be used in a SELECT statement or in a JOIN clause.
- D: A temporary table is a result set and not a value.

References: <https://www.c-sharpcorner.com/UploadFile/996353/difference-between-stored-procedure-and-user-defined-functio/>

QUESTION 11

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 CLR function
- C. a stored procedure
- D. a user-defined scalar function

Correct Answer: AC

Section: (none)

Explanation

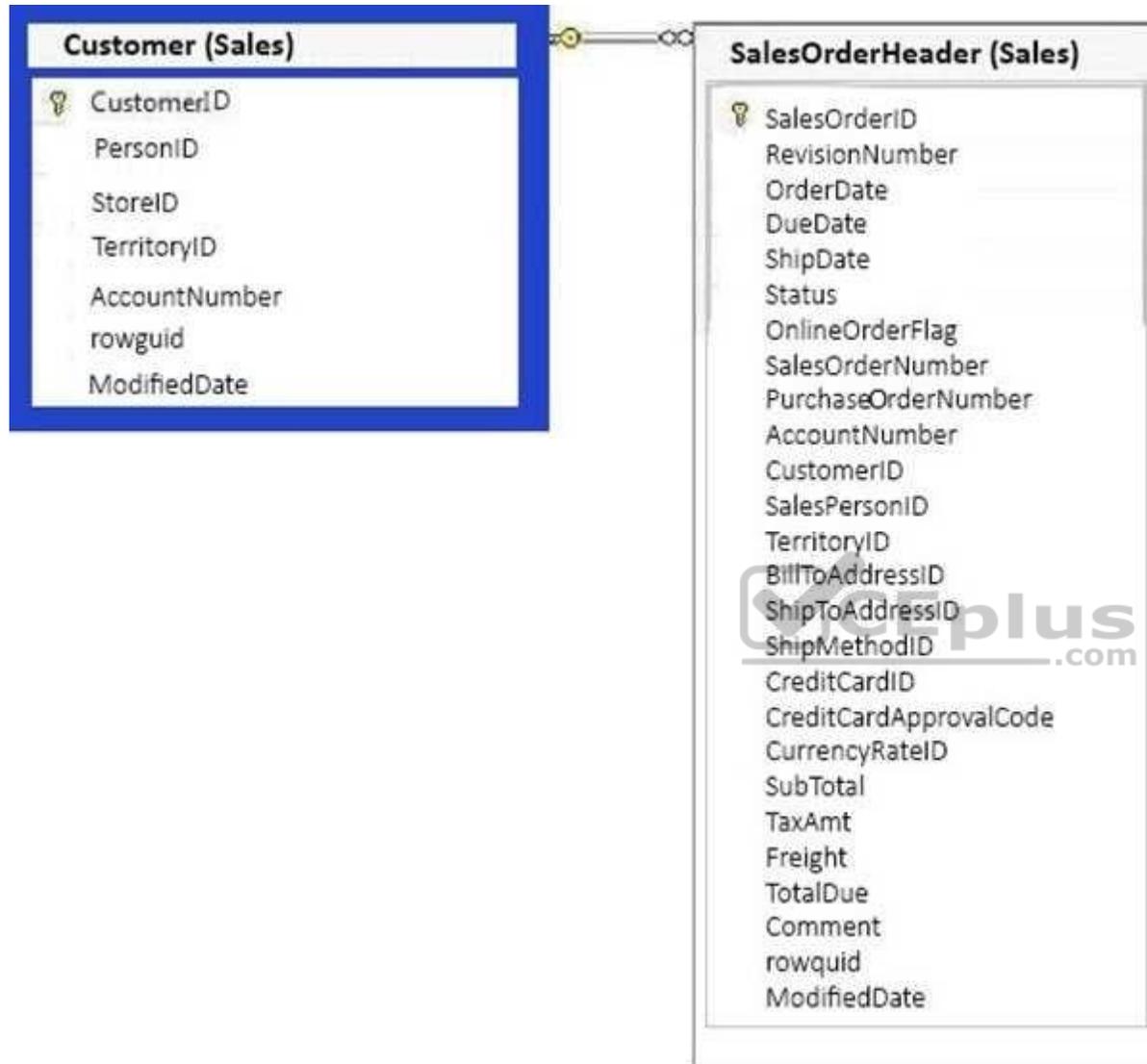
Explanation/Reference:

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 12**

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 13

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 14

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.

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

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

```
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
```

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

A.

B.

C.

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 15

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

```
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
```

```
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
```

A.

B.

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

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

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

C.



D.

E.

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 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 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 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 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

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 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 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 19

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?

```
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
)
```

```
SELECT DISTINCT ServerID, LogMessage FROM Errors AS e1
GROUP BY ServerID, LogMessage
HAVING MAX(Occurrences) = 1
```

```
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
)
```

```
SELECT ServerID, LogMessage FROM Errors AS e1
GROUP BY ServerID, LogMessage, Occurrences
HAVING COUNT(*) = 1
ORDER BY Occurrences
```

A.

B.

C.

D.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:



QUESTION 20

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.



<https://vceplus.com/>

Which Transact-SQL statement should you run?

```
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
```

```
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
```

A.

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
```

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

C.

D.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

QUESTION 21

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
```

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

```
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
```

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



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

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

B. C.

D. E. F.

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

G.



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 22

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
```

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

```
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
```

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



```
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
```

B.

C.

D.

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 23

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?

```
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
```

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

```
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
```

A.

B.

C.

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

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

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



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

D. E. F.

G.

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 24

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?

```
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
```

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

```
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
```

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

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

A.

B. C.

D. E.

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



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

F.

G.

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 25

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

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

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

You need to audit all customer data.

Which Transact-SQL statement should you run?



```
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
```

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

```
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')
```

A.

B. C.

```
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
```

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

```
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')
```

```
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'.com
```

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

D.

E.

F.

G.

H.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

Note: A system-versioned temporal table defined through is a new type of user table in SQL Server 2016, here defined on the last line WITH (SYSTEM_VERSIONING = ON..., is designed to keep a full history of data changes and allow easy point in time analysis.

To query temporal data, the SELECT statement FROM<table> clause has a new clause FOR SYSTEM_TIME with five temporal-specific sub-clauses to query data across the current and history tables.

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

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

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

Which Transact-SQL statement should you run?

```
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
```

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

```
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')
```

```
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
```

- A.

- B.

- C.

D.

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

```
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')
```

```
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'
```

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

E.

F.

G.

H.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Explanation:

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

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

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

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



QUESTION 27

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
```

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

```
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')
```

```
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
```

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

```
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')
```

```
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'
```

B. C.

D. E.

F. G.

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

H.

Correct Answer: G

Section: (none)

Explanation

Explanation/Reference:

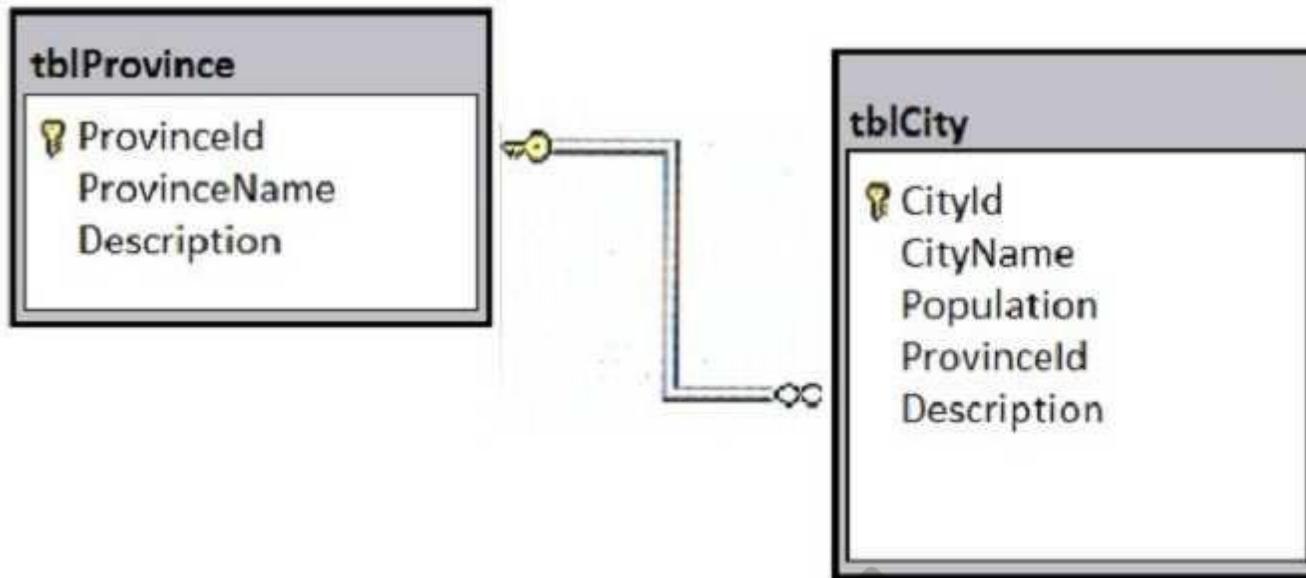
QUESTION 28

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.

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

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

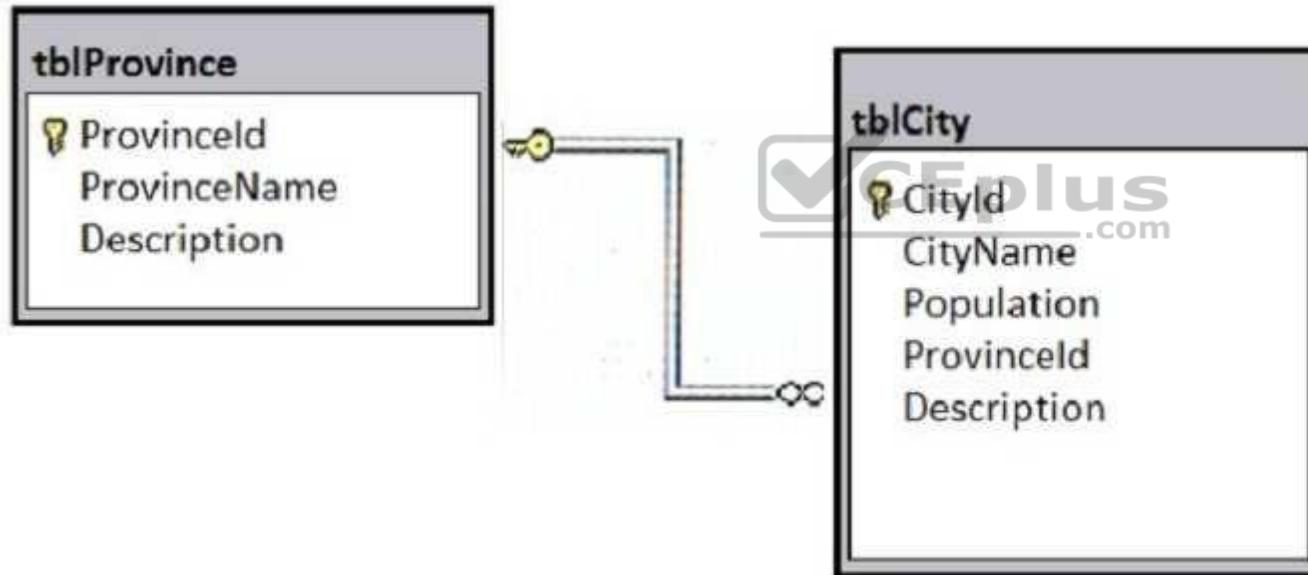
We need to list all provinces that have at least two large cities. There is no reference to this in the code.

QUESTION 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.

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 30

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

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

You 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/>

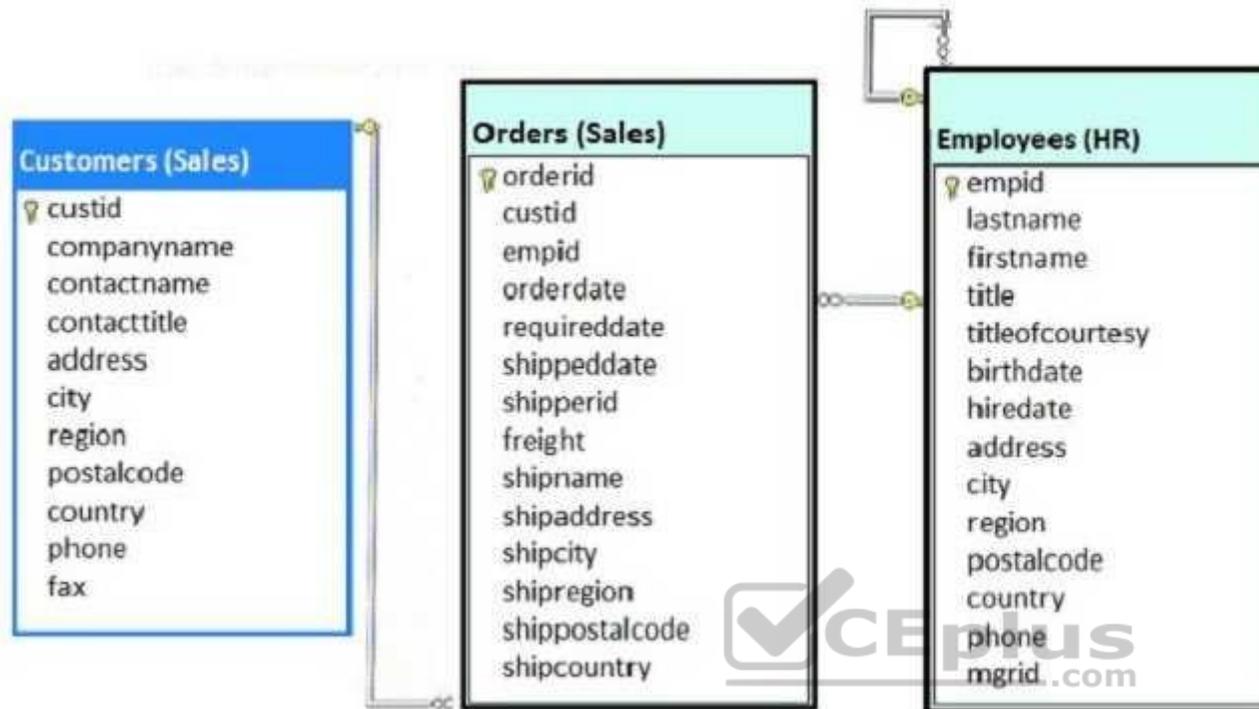
QUESTION 31

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.

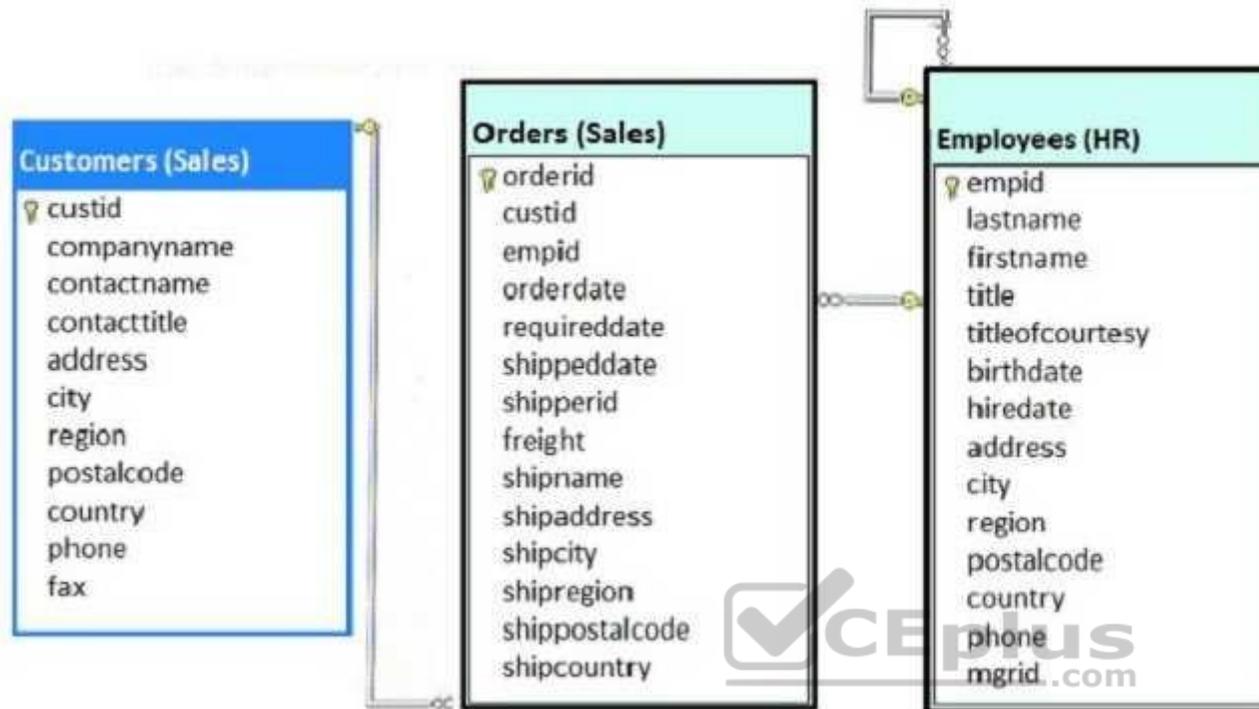
QUESTION 32

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

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

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





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

- the customer number ▪ the customer contact name
- the date the order was placed, with a name of DateofOrder
- a column named Salesperson, formatted with the employee first name, a space, and the employee last name
- orders for customers where the employee identifier equals 4 The output must be sorted by order date, with the newest orders first.

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

Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

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.

AWHERE o.empid =4 clause is correctly used.

GROUP BY is also required.

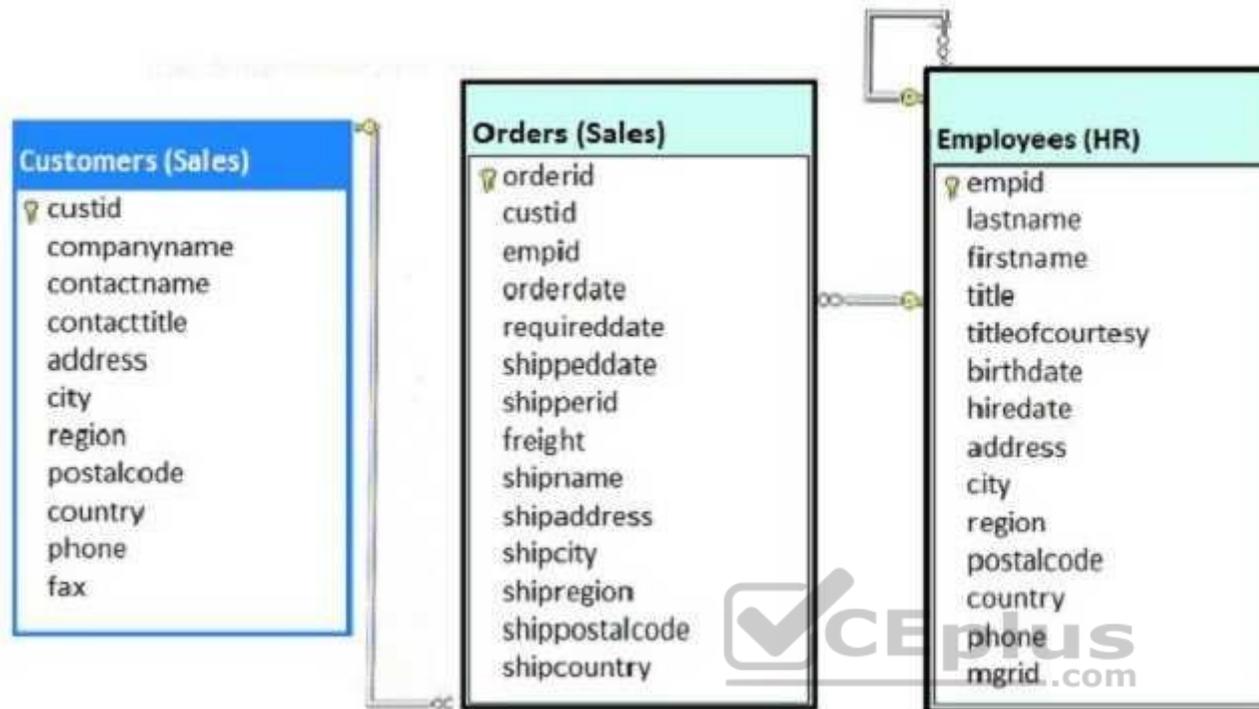
QUESTION 33

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 34

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?

```
CREATE TABLE MiscellaneousPayment (Id uniqueidentifier  
DEFAULT NEWSEQUENTIALID() PRIMARY KEY, Reason varchar(500),  
Amount money)
```

```
CREATE TABLE MiscellaneousPayment (Id  
int identity(1,1) PRIMARY KEY, Reason nvarchar(500), Amount  
numeric(19,4))
```

```
CREATE TABLE MiscellaneousPayment (Id uniqueidentifier  
DEFAULT NEWSEQUENTIALID() PRIMARY KEY, Reason varchar(500),  
Amount decimal(19,4))
```

```
CREATE TABLE MiscellaneousPayment (Id uniqueidentifier  
DEFAULT NEWID() PRIMARY KEY, Reason nvarchar(500), Amount  
decimal(19,4))
```

A.

B. C.

D.

```
CREATE TABLE MiscellaneousPayment (Id uniqueidentifier  
DEFAULT NEWSEQUENTIALID() PRIMARY KEY, Reason nvarchar(500),  
Amount decimal(19,4))
```

```
CREATE TABLE MiscellaneousPayment (Id uniqueidentifier  
DEFAULT NEWID() PRIMARY KEY, Reason nvarchar(500),  
Amount money)
```

E.

F.



Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Explanation:

Incorrect Answers:

A: For column Reason we must use nvarchar, not varchar, as multilingual values must be supported. NEWSEQUENTIALID cannot be referenced in queries. In addition, the money datatype uses rounding and will result in rounding errors.

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.

E: NEWSEQUENTIALID cannot be referenced in queries.

F: The money datatype uses rounding and will result in rounding errors. We should use decimal instead.

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 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 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 36

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
E. SELECT custid FROM
Sales.Customers
UNION ALL
SELECT custid FROM
Sales.Orders F. SELECT
custid FROM
Sales.Customers
UNION
SELECT custid FROM
Sales.Orders G. SELECT
c.custid FROM
Sales.Customers c
RIGHT OUTER JOIN Sales.Orders o
ON c.custid = o.custid
```



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 37

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

```
SELECT COUNT(*)  
FROM (SELECT AcctNo  
FROM tblDepositAcct  
INTERSECT  
SELECT AcctNo  
FROM tblLoanAcct) R
```

A.

```
SELECT COUNT(*)
FROM (SELECT CustNo
      FROM tblDepositAcct
      UNION
      SELECT CustNo
      FROM tblLoanAcct) R
```

```
SELECT COUNT(*)
FROM (SELECT CustNo
      FROMtblDepositAcct
      UNION ALL
      SELECT CustNo
      FROM tblLoanAcct) R
```

```
SELECT COUNT (DISTINCT D.CustNo)
FROM tblDepositAcct D, tblLoanAcct L
WHERE D.CustNo = L.CustNo
```



```
SELECT COUNT(DISTINCT L.CustNo)
FROM tblDepositAcct D
RIGHT JOIN tblLoanAcct L ON D.CustNo =L.CustNo
WHERE D.CustNo IS NULL
```

```
SELECT COUNT(*)
FROM (SELECT CustNo
      FROM tblDepositAcct
      EXCEPT
      SELECT CustNo
      FROM tblLoanAcct) R
```

B. C.

D. E. F.

```
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  
SELECT COUNT(*)  
FROM tblDepositAcct D  
FULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo
```

G.



H.

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 38

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



```
SELECT COUNT(*)  
FROM (SELECT AcctNo  
FROM tblDepositAcct  
INTER  
SECTSELECT Acct  
NoFROM tblLoanAcct) R
```

A.

```
SELECT COUNT(*)
  FROM (SELECT CustNo
        FROM tblDepositAcct
        UNION
        SELECT CustNo
        FROM tblLoanAcct) R
```

```
SELECT COUNT(*)
  FROM (SELECT CustNo
        FROM tblDepositAcct
        UNION ALL
        SELECT CustNo
        FROM tblLoanAcct) R
```

```
SELECT COUNT (DISTINCT D.CustNo)
  FROM tblDepositAcctD, tblLoanAcct L
  WHERE D.CustNo = L.CustNo
```

```
SELECT COUNT(DISTINCT L.CustNo)
  FROM tblDepositAcct D
  RIGHT JOIN tblLoanAcct L ON D.CustNo = L.CustNo
  WHERE D.CustNo IS NULL
```

```
SELECT COUNT(*)
  FROM (SELECT CustNo
        FROM tblDepositAcct
        EXCEPT
        SELECT CustNo
        FROM tblLoanAcct) R
```

B. C.



D. E. F.

```
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  
  
SELECT COUNT (*)  
FROM tblDepositAcct D  
FULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo
```

G.



H.

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 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 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 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 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
```

<https://vceplus.com/>



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 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 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 42

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 43

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

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

You have a 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+COALESCE(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:

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 44

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?

- 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 45

You are building a stored procedure that will update data in a table named Table1 by using a complex query as the data source.

You need to ensure that the SELECT statement in the stored procedure meets the following requirements:

- Data being processed must be usable in several statements in the stored procedure. ▪

Data being processed must contain statistics.

What should you do?

- A. Update Table1 by using a common table expression (CTE).
- B. Insert the data into a temporary table, and then update Table1 from the temporary table.
- C. Place the SELECT statement in a derived table, and then update Table1 by using a JOIN to the derived table.
- D. Insert the data into a table variable, and then update Table1 from the table variable.

Correct Answer: B

Section: (none)

Explanation**Explanation/Reference:**

Explanation:

Temp Tables...

Are real materialized tables that exist in tempdb

Have dedicated stats generated by the engine

Can be indexed

Can have constraints

Persist for the life of the current CONNECTION

Can be referenced by other queries or subproce

Incorrect Answers:

A: CTEs do not have dedicated stats. They rely on stats on the underlying objects

C: Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.



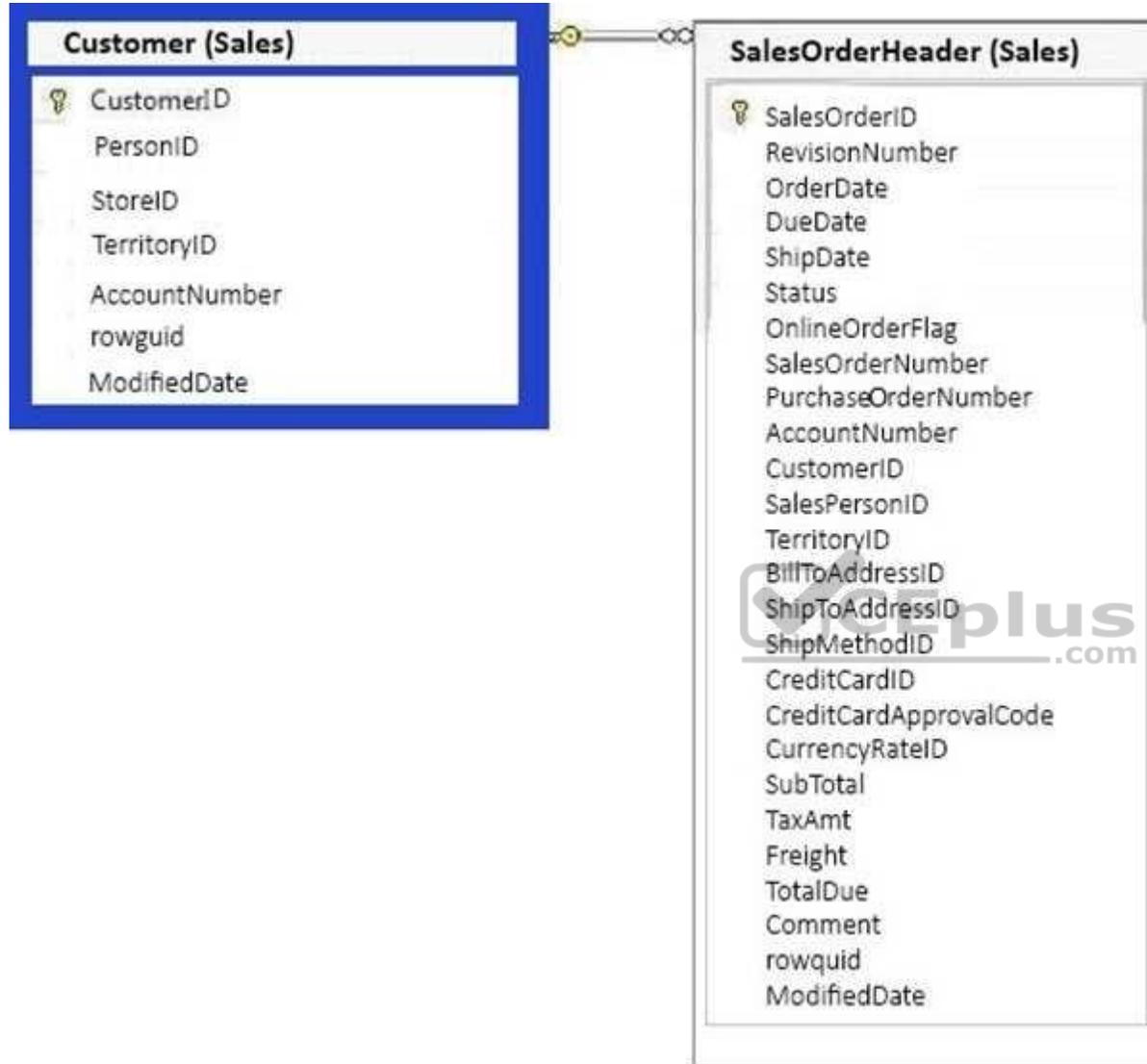
References: [https://technet.microsoft.com/en-us/library/ms190766\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190766(v=sql.105).aspx)

<https://dba.stackexchange.com/questions/13112/whats-the-difference-between-a-cte-and-a-temp-table>

QUESTION 46

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
```

```
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
```

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

```
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
```

B.

C.

D.

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 47

You have a disk-based table that contains 15 columns.



You query the table for the number of new rows created during the current day.

You need to create an index for the query. The solution must generate the smallest possible index.

Which type of index should you create?

- A. clustered
- B. filtered nonclustered with a getdate() predicate in the WHERE statement clause
- C. hash
- D. nonclustered with compression enabled

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Explanation:

A filtered index is an optimized nonclustered index especially suited to cover queries that select from a well-defined subset of data. It uses a filter predicate to index a portion of rows in the table. A well-designed filtered index can improve query performance as well as reduce index maintenance and storage costs compared with full-table indexes.

Creating a filtered index can reduce disk storage for nonclustered indexes when a full-table index is not necessary.

References: <https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-filtered-indexes>

QUESTION 48

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
Unites States	NULL	NULL	\$23395792.75
Unites States	Alabama	NULL	\$646508.75
Unites States	Alabama	Bazemore	\$34402.00
Unites States	Alabama	Belgreen	\$51714.65

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: E

Section: (none)

Explanation

Explanation/Reference:

Explanation:

Example of GROUP BY CUBE result set:

In the following example, the CUBE operator returns a result set that has one grouping for all possible combinations of columns in the CUBE list and a grand total grouping.



Region	Country	Store	SalesPersonID	Total Sales
NULL	NULL	NULL	NULL	254013.6014
NULL	NULL	NULL	287	28461.1854
NULL	NULL	NULL	288	17073.0655
NULL	NULL	NULL	290	208479.3505
NULL	NULL	Spa and Exercise Outfitters	NULL	236210.9015
NULL	NULL	Spa and Exercise Outfitters	287	27731.551
NULL	NULL	Spa and Exercise Outfitters	290	208479.3505
NULL	NULL	Versatile Sporting Goods Company	NULL	17802.6999
NULL	NULL	Versatile Sporting Goods Company	287	729.6344
NULL	NULL	Versatile Sporting Goods Company	288	17073.0655
NULL	DE	NULL	NULL	17802.6999
NULL	DE	NULL	287	729.6344
NULL	DE	NULL	288	17073.0655
NULL	DE	Versatile Sporting Goods Company	NULL	17802.6999
NULL	DE	Versatile Sporting Goods Company	287	729.6344

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 49**

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	Abbottsburg	\$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 50

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?

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

```
SELECT COUNT(*)
FROM (SELECT CustNo
 FROM tblDepositAcct
 UNION
 SELECT CustNo
 FROM tblLoanAcct) R
```

```
SELECT COUNT(*)
FROM (SELECT CustNo
 FROMtblDepositAcct
 UNION ALL
 SELECT CustNo
 FROM tblLoanAcct) R
```

```
SELECT COUNT (DISTINCT D.CustNo)
FROM tblDepositAcct D, tblLoanAcct L
WHERE D.CustNo = L.CustNo
```

```
SELECT COUNT(DISTINCT L.CustNo)
FROM tblDepositAcct D
RIGHT JOIN tblLoanAcct L ON D.CustNo =L.CustNo
WHERE D.CustNo IS NULL
```

```
SELECT COUNT(*)
FROM (SELECT CustNo
 FROM tblDepositAcct
 EXCEPT
 SELECT CustNo
 FROM tblLoanAcct) R
```

B. C.



D. E. F.

```
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
```

```
SELECT COUNT(*)
FROM tblDepositAcct D
FULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo
```

G.



H.

**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 51

**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                         |

#### 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:



```
WITH DIST_CTE (CustA, CustB, Dist)
AS (
 SELECT A.CustomerID AS CustA, B.CustomerID AS CustB,
 B.DeliveryLocation.ShortestLineTo(A.DeliveryLocation).STLength() AS Dist
 FROM Sales.Customers AS A
 CROSS JOIN Sales.Customers AS B
 WHERE A.CustomerID <> B.CustomerID
)
SELECT TOP 1 CustB, Dist
FROM DIST_CTE
WHERE CustA = @custID
ORDER BY Dist
```

The variable @custID is set to a valid customer.

Does the solution meet the goal?

- A. Yes
- B. No

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Explanation:

ShortestLineTo (geometry Data Type) Returns a LineString instance with two points that represent the shortest distance between the two geometry instances. The length of the LineString instance returned is the distance between the two geometry instances.

STLength (geometry Data Type) returns the total length of the elements in a geometry instance.

References: <https://docs.microsoft.com/en-us/sql/t-sql/spatial-geometry/shortestlineto-geometry-data-type>

#### **QUESTION 52**

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

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

You 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                         |

#### 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, B.DeliveryLocation ^ A.DeliveryLocation AS Dist
FROM Sales.Customers AS A
JOIN Sales.Customers AS B
ON A.DeliveryCityID = B.DeliveryCityID
WHERE A.CustomerID = @custID AND A.CustomerID <> B.CustomerID
```

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 53

**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                         |

#### 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 54

**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<br>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 55**

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

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

Which Transact-SQL statement should you run?

```
CREATE TABLE Sales (
 SalesID int IDENTITY(1,1),
 SalesDate DateTime NOT NULL,
 SalesAmount decimal(18,2) NULL
)
```

```
CREATE TABLE Sales (
 SalesID UNIQUEIDENTIFIER DEFAULT NEWSEQUENTIALID() PRIMARY KEY,
 SalesDate DateTime2 NOT NULL,
 SalesAmount money NULL
)
```

```
CREATE TABLE Sales (
 SalesID UNIQUEIDENTIFIER DEFAULT NEWSEQUENTIALID() PRIMARY KEY,
 SalesDate DateTime2 NOT NULL,
 SalesAmount decimal(18,2) NULL
)
```

```
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)
)
```

```
CREATE TABLE Sales (
 SalesID UNIQUEIDENTIFIER DEFAULT NEWID() PRIMARY KEY,
 SalesDate DateTime NOT NULL,
 SalesAmount money NULL
)
```

A.



B. C.

D.

E.



**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, E: 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 56**

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;
```

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

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



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

B.

C.

D.



**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 57**

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

You 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?

```
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
```

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

```
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
```

A.

B.

C.

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

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

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



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

D. E. F.

G.

H.

**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 58**

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');
```



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

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

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

B.



C.

D.



**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 59**

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?



<https://vceplus.com/>

- 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 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
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 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 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

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**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)
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 63**

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

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

You have a table 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(UnitsOnOrder,
NULL)) AS TotalUnitPrice FROM Products
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

#### QUESTION 64

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

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

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

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

You must insert the following data into the Customer table:

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

You need to ensure that both records are inserted or neither record is inserted.

Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Reference: <https://docs.microsoft.com/it-it/sql/t-sql/statements/insert-transact-sql?view=sql-server-2017>

#### QUESTION 65

**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 = CAST(20012 AS varchar(5))
AND RegistrationDate > '2016-01-01'
```

Does the solution meet the goal?

- A. Yes
- B. No

**Correct Answer:** B

**Section:** (none)

**Explanation**



**Explanation/Reference:**

<https://docs.microsoft.com/en-us/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-2017>

#### 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 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 67**

**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 68

**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 69**

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?

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

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

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

A.

B.

C.

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

D.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Answer: B

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 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 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 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 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 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 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 73

**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?

```
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
```

```
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
```

```
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
```

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

A.



B.

C.

D.

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

```
DECLARE @startedTasks TABLE(TaskId int, ProjectId int)
UPDATE Task SET StartTime = GETDATE() OUTPUT inserted.TaskId, deleted.ProjectId INTO @startedTasks
WHERE StartTime is NULL
SELECT COUNT(*) FROM @startedTasks WHERE ProjectId IS NULL
```

E.

F.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**QUESTION 74**

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');
```



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

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

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

B.

C.

D.

**Correct Answer:** B

**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 75**

**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
```

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

```
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')
```

```
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
```

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

```
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')
```

```
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'
```

B. C.

D.

E.

F. G.

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

H.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

#### QUESTION 76

**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?

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

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

```
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')
```

```
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
```

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

A.

B. C.

D. E.

```
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')
```

```
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'
```

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

F.

G.

H.

**Correct Answer:** F  
**Section:** (none)  
**Explanation**

**Explanation/Reference:**

#### QUESTION 77

**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 global temporary table in the stored procedure.

Does this meet the goal?

- A. Yes
- B. No

**Correct Answer:** A  
**Section:** (none)  
**Explanation**

**Explanation/Reference:**

#### QUESTION 78

**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

**Correct Answer: B**

**Section: (none)**

**Explanation**

**Explanation/Reference:**



#### **QUESTION 79**

**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 80

**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.



<https://vceplus.com/>

Does this meet the goal?

- A. Yes
- B. No

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**



#### QUESTION 81

**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 SalesOrders.

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

Solution: You recommend the following query:

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

Does this meet the goal?

- A. Yes
- B. No

**Correct Answer:** B  
**Section:** (none)  
**Explanation**



**Explanation/Reference:**

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

## QUESTION 82

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

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

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

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

Solution: You recommend the following query:

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

Does this meet the goal?

- A. Yes
- B. No

**Correct Answer:** A  
**Section:** (none)  
**Explanation**



**Explanation/Reference:**

### QUESTION 83

**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

**Correct Answer: B**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

#### **QUESTION 84**

You are developing a mobile app to manage meetups. The app allows for users to view the 25 closest people with similar interests. You have a table that contains records for approximately two million people. You create the table by running the following Transact-SQL statement:

```
CREATE TABLE Person (
 PersonID INT,
 Name NVARCHAR(155) NOT NULL,
 Location GEOGRAPHY,
 Interests NVARCHAR(MAX)
)
```



You create the following table valued function to generate lists of people:

```
CREATE FUNCTION dbo.nearby (@person AS INT)
 RETURNS @Res TABLE (
 PersonId INT NOT NULL,
 Location GEOGRAPHY
)
AS
BEGIN
 . . .
END
```

You need to build a report that shows meetings with at least two people only.

What should you use?

- A. OUTER APPLY
- B. CROSS APPLY
- C. PIVOT
- D. LEFT OUTER JOIN



**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

References: <https://www.sqlshack.com/the-difference-between-cross-apply-and-outer-apply-in-sql-server/> **QUESTION 85**

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

The add-on must meet the following requirements:

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

You run the following Transact-SQL statement:

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

You need to ensure that users can perform searches of descriptions.

Which Transact-SQL statement should you run? A.

```
DECLARE @term NVARCHAR(255)
```

```
. . .
```

```
SELECT Id, Description
```

```
FROM Bug
```

```
WHERE CHARINDEX(@term, Description) > 0
```



```
DECLARE @term NVARCHAR(255)
. . .
SELECT Id, Description
FROM Bug
WHERE DIFFERENCE(@term, Description) > 0
DECLARE @term NVARCHAR(255)
. . .
SELECT Id, Description
FROM Bug
WHERE CONTAINS (Description, '%@term%')
DECLARE @term NVARCHAR(255)
. . .
SELECT Id, Description
FROM Bug
WHERE CONTAINS (Description, @term)
```



B.

C.

D.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

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

#### **QUESTION 86**

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.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

#### **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 nonclustered index on the primary key column that does NOT include columns.

Does this meet the goal?

- A. Yes
- B. No

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

References: <https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-2017>

#### QUESTION 88

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

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

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

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

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

```
DECLARE cur CURSOR
FOR
SELECT LastName, CreditLimit
FROM Customer
```



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

If the credit limit is zero, you must delete the customer record while fetching data.

You need to add the `DELETE` statement.

Solution: You add the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Explanation:

CURRENT OF specifies that the DELETE is performed at the current position of the specified cursor.

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

#### QUESTION 89

You have a table named Table1 that contains 200 million rows. Table1 contains a column named SaleDate that has a data type of DateTime2(3).

Users report that the following query runs slowly.

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

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

What should you use to replace the WHERE statement?

- A. WHERE SaleDate >= '2017-01-01' AND SaleDate < '2018-01-01'

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

**Correct Answer: C**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

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

### QUESTION 90

You have a database that tracks customer complaints.

The database contains a table named Complaints that includes the following columns:

| Column name        | Column description                                                        |
|--------------------|---------------------------------------------------------------------------|
| ComplaintID        | This is a unique identifier for a complaint record.                       |
| CustomerTranscript | This column stores a transcribed verbatim record of a customer complaint. |

You need to create a query that lists complaints about defective products. The report must include complaints where the exact phrase “defective product” occurs, as well as complaints where similar phrases occur.

Which Transact-SQL statement should you run?

- A. `SELECT ComplaintID, ComplaintTranscript FROM Complaints  
WHERE CONTAINS(CustomerTranscript, 'defective')  
AND CONTAINS(CustomerTranscript, 'product')`
- B. `SELECT ComplaintID, CustomerTranscript FROM Complaints  
WHERE SOUNDEX('defective') = SOUNDEX('product')`
- C. `SELECT ComplaintID, CustomerTranscript FROM Complaints  
WHERE FREETEXT(CustomerTranscript, 'defective product')`

D. SELECT ComplaintID, Customer Transcript FROM Complaints  
WHERE CustomerTranscript like '%defective product%'

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

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

### QUESTION 91

You run the following Transact-SQL statements:

```
CREATE TABLE CourseParticipants
(
 CourseID INT NOT NULL,
 CourseDate DATE NOT NULL,
 LocationDescription VARCHAR(100) NOT NULL,
 NumParticipants INT NOT NULL
)
```

You need to create a query that returns the total number of attendees for each combination of CourseID, CourseDate, and the following locations: Lisbon, London, and Seattle. The result set should resemble the following:

|   | CourseID | CourseDate | Lisbon | London | Seattle |
|---|----------|------------|--------|--------|---------|
| 1 | 1        | 2018-02-01 | NULL   | NULL   | 15      |
| 2 | 2        | 2018-02-01 | 33     | NULL   | NULL    |
| 3 | 1        | 2018-02-02 | NULL   | 20     | NULL    |
| 4 | 1        | 2018-02-03 | 20     | 10     | NULL    |
| 5 | 2        | 2018-02-03 | NULL   | 20     | NULL    |

Which Transact-SQL code segment should you run?

A. SELECT \*

```
FROM CourseParticipants
PIVOT(SUM(NumParticipants) FOR LocationDescription
IN (Lisbon, London, Seattle))
```

- B. 

```
SELECT *
FROM CourseParticipants
PIVOT(SUM(NumParticipants) FOR LocationDescription
IN (Lisbon, London, Seattle)) as PVTTable
```
- C. 

```
SELECT *
FROM CourseParticipants
UNPIVOT(SUM(NumParticipants) FOR LocationDescription
IN (Lisbon, London, Seattle))
```
- D. 

```
SELECT *
FROM CourseParticipants
UNPIVOT(SUM(NumParticipants) FOR LocationDescription
IN (Lisbon, London, Seattle)) AS PVTTable
```

**Correct Answer: B**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

References: [https://www.techonthenet.com/sql\\_server/pivot.php](https://www.techonthenet.com/sql_server/pivot.php)



## QUESTION 92

**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 Person that contains information about employees. Users are requesting a way to access specific columns from the Person table without specifying the Person table in the query statement. The columns that users can access will be determined when the query is running against the data. There are some records that are restricted, and a trigger will evaluate whether the request is attempting to access a restricted record.

You need to ensure that users can access the needed columns while minimizing storage on the database server.

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:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

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

### QUESTION 93

A company's sales team is divided in two different regions, North and South. You create tables named SalesNorth and SalesSouth. The SalesNorth table stores sales data from the North region. The SalesSouth table stores sales data from the South region. Both tables use the following structure:

| Column name | Data type    | Allow nulls |
|-------------|--------------|-------------|
| region      | CHAR(1)      | Yes         |
| salesID     | INT          | Yes         |
| customer    | VARCHAR(150) | Yes         |
| amount      | MONEY        | Yes         |

You need to create a consolidated result set that includes all records from both tables.

Which Transact-SQL statement should you run?

- A. 

```
SELECT SalesNorth.salesID, SalesNorth.customer,
SalesNorth.amount, SalesSouth.SalesID, SalesSouth.customer,
SalesSouth.amount
FROM SalesNorth
```

```
JOIN SalesSouth ON SalesNorth.salesID = SalesSouth.salesID
```

```
B. SELECT SalesNorth.salesID, SalesNorth.customer,
SalesNorth.amount, SalesSouth.salesID, SalesSouth.customer,
SalesSouth.amount
FROM SalesNorth
LEFT JOIN SalesSouth
ON SalesNorth.salesID=SalesSouth.salesID
```

```
C. SELECT salesID, customer, amount
FROM SalesNorth
UNION ALL
SELECT salesID, customer, amount
FROM SalesSouth D. SELECT salesID,
customer, amount
FROM SalesNorth
UNION
SELECT salesID, customer, amount
FROM SalesSouth
```

**Correct Answer: C**

**Section: (none)**

**Explanation**



**Explanation/Reference:**

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

#### **QUESTION 94**

You are developing a database to track employee progress relative to training goals. You run the following Transact-SQL statements:

```
CREATE TABLE Employees(
 EmployeeID INT IDENTITY(1,1) NOT NULL,
 Name VARCHAR(150) NULL,
 CONSTRAINT PK_Employees PRIMARY KEY CLUSTERED (
 EmployeeID ASC
) WITH (STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF) ON PRIMARY
) ON PRIMARY

CREATE TABLE CoursesTaken(
 CourseID INT NOT NULL,
 EmployeeID INT NOT NULL,
 CourseTakenOn DATE NULL,
 CONSTRAINT PK_CoursesTaken PRIMARY KEY CLUSTERED (
 CourseID ASC, EmployeeID ASC
) WITH (STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF) ON PRIMARY

CREATE TABLE Courses(
 CourseID INT IDENTITY(1,1) NOT NULL,
 Course VARCHAR(50) NULL,
 CONSTRAINT PK_Courses PRIMARY KEY CLUSTERED (
 CourseID ASC
) WITH (STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF) ON PRIMARY
) ON PRIMARY
```

You must build a report that shows all Employees and the courses that they have taken. Employees that have not taken training courses must still appear in the report. The report must display NULL in the course column for these employees.

You need to create a query for the report.

```
SELECT e.Name, c.Course
FROM dbo.Courses c
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID
INNER JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
SELECT e.Name, c.Course
FROM dbo.Courses c
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID
JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
SELECT e.Name, c.Course
FROM dbo.Courses c
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID
LEFT JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
SELECT e.Name, c.Course
FROM dbo.Courses c
JOIN dbo.CoursesTaken ct ON c.CourseID = ct.CourseID
RIGHT JOIN dbo.Employees e ON ct.EmployeeID = e.EmployeeID
```

A.

B.

C.

D.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Explanation:

Incorrect Answers:

A, B: JOIN and INNER JOIN displays only the rows that have a match in both joined tables

References: <https://www.mssqltips.com/sqlservertip/1667/sql-server-join-example/>

### QUESTION 95

**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;
HAVING
 COUNT (Ord.OrderID) > 0;
```

Does this meet the goal?

- A. Yes
- B. No



**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

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

#### QUESTION 96

**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
 THROW 51000, 'The product could not be created,' 1
 END CATCH
END
```

Does this meet the goal?

- A. Yes
- B. No

**Correct Answer: B**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

Explanation:

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 uncommittable transaction.

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

Example of correct use:

```
BEGIN CATCH
```

```
-- Test XACT_STATE for 0, 1, or -1.
-- If 1, the transaction is committable.
-- If -1, the transaction is uncommittable and should
-- be rolled back.
-- XACT_STATE = 0 means there is no transaction and
- a commit or rollback operation would generate an error.
```



```
-- Test whether the transaction is uncommittable.
```

```
IF (XACT_STATE()) = -1
```

```
BEGIN
```

```
 PRINT 'The transaction is in an uncommittable state.' +
 ' Rolling back transaction.'
```

```
 ROLLBACK TRANSACTION;
```

```
END;
```

```
-- Test whether the transaction is active and valid.
```

```
IF (XACT_STATE()) = 1
```

```
BEGIN
```

```
 PRINT 'The transaction is committable.' +
 ' Committing transaction.'
```

```
 COMMIT TRANSACTION;
```

```
END;
END CATCH;
```

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

#### QUESTION 97

You are performing a code review of stored procedures. Code at line SP03 fails to run (Line numbers are included for reference only.)

```
SP01 BEGIN TRY
SP02 BEGIN TRANSACTION
SP03 . . .
SP04 COMMIT TRANSACTION
SP05 END TRY
SP06 BEGIN CATCH
SP07
SP08 ROLLBACK TRANSACTION
SP09 END CATCH
```



You need to ensure that transactions are rolled back when an error occurs.

Which Transact-SQL segment should you insert at line SP07?

- A. `If @@Error <> 0`
- B. `If @@ TRANCOUNT = 0`
- C. `If @@ TRANCOUNT > 0`
- D. `If @@ Error = 0`

**Correct Answer: C**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

Explanation:

Using TRY...CATCH in a transaction

The following example shows how a TRY...CATCH block works inside a transaction. The statement inside the TRY block generates a constraint violation error.

```
BEGIN TRANSACTION;

BEGIN TRY
 -- Generate a constraint violation error.
 DELETE FROM Production.Product
 WHERE ProductID = 980;
END TRY
BEGIN CATCH
 SELECT
 ERROR_NUMBER() AS ErrorNumber
 ,ERROR_SEVERITY() AS ErrorSeverity
 ,ERROR_STATE() AS ErrorState
 ,ERROR_PROCEDURE() AS ErrorProcedure
 ,ERROR_LINE() AS ErrorLine
 ,ERROR_MESSAGE() AS ErrorMessage;

 IF @@TRANCOUNT > 0
 ROLLBACK TRANSACTION;
END CATCH;

IF @@TRANCOUNT > 0
 COMMIT TRANSACTION;
GO
```



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

### QUESTION 98

You have a database that tracks customer complaints.

The database contains a table named Complaints that includes the following columns:

| Column name        | Column description                                                        |
|--------------------|---------------------------------------------------------------------------|
| ComplaintID        | This is a unique identifier for a complaint record.                       |
| CustomerTranscript | This column stores a transcribed verbatim record of a customer complaint. |

You need to create a query that lists complaints about defective products. The report must include complaints where the exact phrase “defective product” occurs, as well as complaints where similar phrases occur. A.

```
SELECT ComplaintID, CustomerTranscript FROM Complaints
INNER JOIN FREETEXTABLE (Complaints, CustomerTranscript, 'defective product') AS Matches
ON Complaints.ComplaintID = Matches.[KEY]
```

```
SELECT ComplaintID, CustomerTranscript FROM Complaints
INNER JOIN CONTAINSTABLE (Complaints, CustomerTranscript, '%defective% product%') AS Matches
ON Complaints.ComplaintID = Matches.[KEY]
```

```
SELECT ComplaintID, CustomerTranscript, FROM Complaints
WHERE CONTAINS (CustomerTranscript, 'defective product')
```

```
SELECT ComplaintID, ComplaintTranscript FROM Complaints
WHERE CONTAINS (CustomerTranscript, 'defective') AND CONTAINS (CustomerTranscript, 'product')
```

B.

C.

D.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

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

#### QUESTION 99

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

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

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

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

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

```
DECLARE cur CURSOR
FOR
SELECT LastName, CreditLimit
FROM Customer

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

If the credit limit is zero, you must delete the customer record while fetching data.

You need to add the DELETE statement.

Solution: You add the following Transact-SQL statement:

```
IF @CreditLimit = 0
 DELETE TOP (1) Customer
 WHERE LastName = @LastName
```

Does the solution meet the goal?

- A. Yes
- B. No

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Explanation:

Use a WHERE CURRENT OF clause, which deletes at the current position of the specified cursor.

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

#### QUESTION 100

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?

```
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
```

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

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

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

A.

B.

C.

D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

<https://vceplus.com/>

