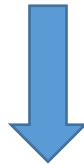


Microsoft MCSA Certification 70-761 Exam



- **Vendor: Microsoft**
- **Exam Code: 70-761**
- **Exam Name: Querying Data with Transact-SQL**

Get Complete Version Exam 70-761 Dumps with VCE and PDF Here



<https://www.passleader.com/70-761.html>

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
```

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

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

Solution: You run the following Transact-SQL statement:

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

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

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

- There is no active user transaction for the current request.
- The current request has an active user transaction. The request can perform any actions, including writing data and committing the 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.

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

Answer: B

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: @@TRANCOUNT 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

Answer: A

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

Answer: B

Explanation:

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

Answer: B

Explanation:

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

Answer: A

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

Answer: A

Explanation:

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

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

Answer: B

Explanation:

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

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
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

The variable max, in the line DECLARE @areaCode nvarchar(max), is not defined.

QUESTION 10

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

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

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

You need to return the total count of tasks that are impacted by this UPDATE operation, but are not associated with a project. What set of Transact-SQL statements should you run?

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

Answer: B

Explanation:

The WHERE clause of the third line should be WHERE ProjectID IS NULL, as we want to count the tasks that are not associated with a project.

QUESTION 11

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.

Hotspot Question

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

You need to identify the owner of each task by using the following rules:

- Return each task's owner if the task has an owner.
- If a task has no owner, but is associated with a project that has an owner, return the project's owner.
- Return the value -1 for all other cases.

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

Answer Area

SELECT T.TaskId, T.TaskName,

ISNULL COALESCE CHOOSE	(T.UserId, P.UserId, -1 P.UserId, T.UserId, -1 -1, P.UserId, T.UserId -1, T.UserId, P.UserId) AS OwnerUserId
------------------------------	---

FROM Task T

INNER JOIN LEFT JOIN RIGHT JOIN	Project P ON T.ProjectId = P.ProjectId
---------------------------------------	--

Answer:

Answer Area

SELECT T.TaskId, T.TaskName,

ISNULL	(T.UserId, P.UserId, -1) AS OwnerUserId
COALESCE		P.UserId, T.UserId, -1	
CHOOSE		-1, P.UserId, T.UserId	
		-1, T.UserId, P.UserId	

FROM Task T

INNER JOIN	Project P ON T.ProjectId = P.ProjectId
LEFT JOIN	
RIGHT JOIN	

Explanation:

Box 1: COALESCE

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

Box 2: T.UserId, p.UserId, -1

- Return each task's owner if the task has an owner.
- If a task has no owner, but is associated with a project that has an owner, return the project's owner.
- Return the value -1 for all other cases.

Box 3: RIGHT JOIN

The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match. Here the right side could be NULL as the projectID of the task could be NULL.

References:

<https://msdn.microsoft.com/en-us/library/ms190349.aspx>

http://www.w3schools.com/Sql/sql_join_right.asp

QUESTION 12

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.

Drag and Drop Question

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

Task level is defined using the following rules:

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

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

Transact-SQL segments		Answer Area
<code>) SELECT * FROM TaskWithLevel</code>		
<code>SELECT CAST(NULL AS int) AS ParentTaskId, T.TaskId, T.TaskName, 0 AS TaskLevel FROM Task T WHERE T.ParentTaskId IS NULL</code>		
<code>With TaskWithLevel (ParentTaskId, TaskId, TaskName, TaskLevel) As {</code>		
<code>UNION</code>		
<code>SELECT R.TaskId AS ParentTaskId, T.TaskId, T.TaskName, R.TaskLevel+1 AS TaskLevel FROM Task T INNER JOIN TaskWithLevel R ON T.ParentTaskId = R.TaskId</code>		
<code>SELECT T.TaskId AS ParentTaskId, CAST(null AS int) AS TaskId, T.TaskName, 0 AS TaskLevel FROM Task T WHERE T.ParentTaskId IS NULL</code>		
<code>UNION ALL</code>		

⏪ ⏩ ⏴ ⏵

Answer:

Transact-SQL segments	Answer Area
<pre> } SELECT * FROM TaskWithLevel </pre>	<pre> SELECT CAST(NULL AS int) AS ParentTaskId, T.TaskId, T.TaskName, 0 AS TaskLevel FROM Task T WHERE T.ParentTaskId IS NULL </pre>
<pre> With TaskWithLevel (ParentTaskId, TaskId, TaskName, TaskLevel) As { </pre>	<pre> UNION </pre>
<pre> UNION </pre>	<pre> SELECT R.TaskId AS ParentTaskId, T.TaskId, T.TaskName, R.TaskLevel+1 AS TaskLevel FROM Task T INNER JOIN TaskWithLevel R ON T.ParentTaskId = R.TaskId </pre>
<pre> SELECT R.TaskId AS ParentTaskId, T.TaskId, T.TaskName, R.TaskLevel+1 AS TaskLevel FROM Task T INNER JOIN TaskWithLevel R ON T.ParentTaskId = R.TaskId </pre>	<pre> With TaskWithLevel (ParentTaskId, TaskId, TaskName, TaskLevel) As { </pre>
<pre> SELECT T.TaskId AS ParentTaskId, CAST(null AS int) AS TaskId, T.TaskName, 0 AS TaskLevel FROM Task T WHERE T.ParentTaskId IS NULL </pre>	<pre> } SELECT * FROM TaskWithLevel </pre>
<pre> UNION ALL </pre>	

Explanation:

Box 1: SELECT CAST (NULL AS INT) AS ParentTaskID, etc.

This statement selects all tasks with task level 0. The ParentTaskID could be null so we should use CAST (NULL AS INT) AS ParentTaskID.

Box 2: UNION

We should use UNION and not UNION ALL as we do not want duplicate rows. UNION specifies that multiple result sets are to be combined and returned as a single result set.

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

Box 3, Box 4, Box 5:

These statements select all tasks with task level >0.

References:

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

QUESTION 13

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.

Drag and Drop Question

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

When running an operation, you updated a column named EndTime for several records in the Project table, but updates to the corresponding task records in the Task table failed. You need to synchronize the value of the End Time column in the Task table with the value of the EndTime column in the project table. The solution must meet the following requirements:

- If the End Time column has a value, make no changes to the record.
- If the value of the EndTime column is null and the corresponding project record is marked as completed, update the record with the project finish time.

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





Transact-SQL segments

FROM Project AS P
WHERE P.EndTime IS NOT NULL AND T.EndTime is NULL
FROM Task AS T
WHERE P.EndTime IS NULL AND T.EndTime IS NOT NULL
UPDATE T SET T.EndTime = P.EndTime
INNER JOIN Project AS P ON T.ProjectId = P.ProjectId
INNER JOIN Task AS T ON T.UserId = P.UserId
UPDATE P SET P.EndTime = T.EndTime

Answer Area



Answer:

Transact-SQL segments	Answer Area
FROM Project AS P	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">   </div> <div style="text-align: center;">   </div> </div>
WHERE P.EndTime IS NOT NULL AND T.EndTime is NULL	
FROM Task AS T	
WHERE P.EndTime IS NULL AND T.EndTime IS NOT NULL	
UPDATE T SET T.EndTime = P.EndTime	
INNER JOIN Project AS P ON T.ProjectId = P.ProjectId	
INNER JOIN Task AS T ON T.UserId = P.UserId	
UPDATE P SET P.EndTime = T.EndTime	

Explanation:

Box 1: UPDATE T SET T.EndTime = P.EndTime

We are updating the EndTime column in the Task table.

Box 2: FROM Task AS T

Where are updating the task table.

Box 3: INNER JOIN Project AS P on T.ProjectID = P.ProjectID We join with the Project table (on the ProjectID columnID column).

Box 4: WHERE P.EndTime is NOT NULL AND T.EndTime is NULL We select the columns in the Task Table where the EndTime column in the Project table has a value (NOT NULL), but where it is NULL in the Task Table.

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

QUESTION 14

Drag and Drop Question

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

- Produces a warning if the credit limit parameter is greater than 7,000
- Propagates all unexpected errors to the calling process

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

Transact-SQL segments	Answer Area
RAISERROR ('Warning: Credit limit is over 7,000!', 16, 1)	<pre> CREATE PROC dbo.UpdateCustomer @CustomerID int, @CreditLimit money AS BEGIN DECLARE @ErrorMessage varchar(1000) BEGIN TRY T UPDATE dbo.Customer SET CreditLimit = @CreditLimit WHERE CustomerID = @CustomerID END TRY BEGIN CATCH SET @ErrorMessage = ERROR_MESSAGE() INSERT INTO dbo.ErrorLog (ApplicationID, [Date], ErrorMessage) VALUES (1, GETDATE(), @ErrorMessage) END CATCH END </pre> <div style="border: 1px dashed gray; padding: 5px; margin: 10px auto; width: 200px; text-align: center;">Transact-SQL segment</div> <div style="border: 1px dashed gray; padding: 5px; margin: 10px auto; width: 200px; text-align: center;">Transact-SQL segment</div>
RAISERROR ('Warning: Credit limit is over 7,000!', 10, 1)	
THROW 51000, 'Warning: Credit limit is over 7,000!', 1	
THROW	
RAISERROR (@ErrorMessage, 16, 1)	
RAISERROR (@ErrorMessage, 10, 1)	
THROW 51000, @ErrorMessage, 1	
RAISERROR (@ErrorMessage, 20, 1) WITH LOG	

Answer:

Transact-SQL segments

```
RAISERROR ('Warning: Credit
limit is over 7,000!', 16, 1)
RAISERROR ('Warning: Credit
limit is over 7,000!', 10, 1)
THROW 51000, 'Warning: Credit
limit is over 7,000!', 1
THROW
RAISERROR (@ErrorMessage, 16, 1)
RAISERROR (@ErrorMessage, 10, 1)
THROW 51000, @ErrorMessage, 1
RAISERROR (@ErrorMessage, 20, 1)
WITH LOG
```

Answer Area

```
CREATE PROC dbo.UpdateCustomer @CustomerID int, @CreditLimit money
AS
BEGIN
    DECLARE @ErrorMessage varchar(1000)
    BEGIN TRY
        THROW 51000, 'Warning: Credit
        limit is over 7,000!', 1
        UPDATE dbo.Customer
        SET CreditLimit = @CreditLimit
        WHERE CustomerID = @CustomerID
    END TRY
    BEGIN CATCH
        SET @ErrorMessage = ERROR_MESSAGE()
        INSERT INTO dbo.ErrorLog (ApplicationID, [Date], ErrorMessage)
        VALUES (1, GETDATE(), @ErrorMessage)
        RAISERROR (@ErrorMessage, 16, 1)
    END CATCH
END
```

Explanation:

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

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

THROW syntax:

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

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

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

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

On Severity level 16. Using THROW to raise an exception The following example shows how to use the THROW statement to raise an exception.

Transact-SQL

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

Here is the result set.

Msg 51000, Level 16, State 1, Line 1

The record does not exist.

Note: RAISERROR syntax:

```
RAISERROR ( { msg_id | msg_str | @local_variable } { ,severity ,state }
[ ,argument [ ,...n ] ] )
[ WITH option [ ,...n ] ]
```

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

References:

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

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

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

QUESTION 15

[70-761 Exam Dumps](#) [70-761 Exam Questions](#) [70-761 PDF Dumps](#) [70-761 VCE Dumps](#)

[Back to the Source of this PDF & Get More Free Braindumps -- www.microsoftbraindumps.com](#)

Hotspot Question

You have the following stored procedure:

```
CREATE PROC dbo.UpdateLogs @Code char(5), @ApplicationId int, @Info varchar(1000)
AS
BEGIN
    BEGIN TRY
        BEGIN TRAN
            INSERT INTO dbo.Log1 VALUES (@Code, @ApplicationId, @Info)
            IF @Code = 'C2323' AND @ApplicationId = 1
                RAISERROR('C2323 code from HR application!', 16, 1)
            ELSE
                INSERT INTO dbo.Log2 VALUES (@Code, @ApplicationId, @Info)
                INSERT INTO dbo.Log3 VALUES (@Code, @ApplicationId, @Info)
            BEGIN TRAN
                IF @Code = 'C2323'
                    ROLLBACK TRAN
                ELSE
                    INSERT INTO dbo.Log4 VALUES (@Code, @ApplicationId, @Info)
                    IF @@TRANCOUNT > 0
                        COMMIT TRAN
            END TRY
        BEGIN CATCH
            IF XACT_STATE() != 0
                ROLLBACK TRAN
        END CATCH
    END
```

You run the following Transact-SQL statements:

```
EXEC dbo.UpdateLogs 'C2323', 1, 'Employee records are updated.'
EXEC dbo.UpdateLogs 'C2323', 10, 'Sales process started.'
```

What is the result of each Transact-SQL statement? To answer, select the appropriate options in the answer area.

Answer Area

Stored procedure execution

Result

First stored procedure execution

- ☐ All transactions are rolled back.
- ☐ Only the Log1 and Log3 tables are updated.
- ☐ Only the Log1 table is updated.
- ☐ All four tables are updated.

Second stored procedure execution

- ☐ Only the Log1, Log2, and Log3 tables are updated.
- ☐ All transactions are rolled back.
- ☐ Only the Log1 table is updated.
- ☐ Only the Log1 and Log3 tables are updated.

Answer:

Answer Area

Stored procedure execution

Result

First stored procedure execution

All transactions are rolled back.
Only the Log1 and Log3 tables are updated.
Only the Log1 table is updated.
All four tables are updated.

Second stored procedure execution

Only the Log1, Log2, and Log3 tables are updated.
All transactions are rolled back.
Only the Log1 table is updated.
Only the Log1 and Log3 tables are updated.

Explanation:

Box 1: All transactions are rolled back.

The first IF-statement, IF @CODE = 'C2323' AND @ApplicationID = 1, will be true, an error will be raised, the error will be caught in the CATCH block, and the only transaction that has been started will be rolled back.

Box 2: Only Log1, Log2, and Log3 tables are updated. The second IF-statement, IF @Code = 'C2323', will be true, so the second transaction will be rolled back, but log1, log2, and log3 was updated before the second transaction.

QUESTION 16

Hotspot Question

You need to develop a Transact-SQL statement that meets the following requirements:

- The statement must return a custom error when there are problems updating a table.
- The error number must be value 50555.
- The error severity level must be 14.
- A Microsoft SQL Server alert must be triggered when the error condition occurs.

Which Transact-SQL segment should you use for each requirement? To answer, select the appropriate Transact-SQL segments in the answer area.

Answer Area

Requirement

Transact-SQL segment

Check for error condition

BEGIN TRANSACTION...END TRANSACTION
TRY_PARSE
BEGIN...END
TRY...CATCH

Custom error implementation

THROW 50555, 'The update failed.', 1
RAISERROR (50555, 14,1, 'The update failed.') WITH LOG
RAISERROR (50555, 14,1 'The update failed.') WITH NOWAIT
RAISERROR (50555, 'The update failed.')

Answer:

Answer Area

Requirement	Transact-SQL segment
Check for error condition	<div> <div>▼</div> <div> BEGIN TRANSACTION...END TRANSACTION TRY_PARSE BEGIN...END TRY...CATCH </div> </div>
Custom error implementation	<div> <div>▼</div> <div> THROW 50555, 'The update failed.', 1 RAISERROR (50555, 14, 1, 'The update failed.') WITH LOG RAISERROR (50555, 14, 1 'The update failed.') WITH NOWAIT RAISERROR (50555, 'The update failed.') </div> </div>

Explanation:

Box 1: TRY...CATCH

The TRY...CATCH Transact-SQL construct implements error handling for Transact-SQL that is similar to the exception handling in the Microsoft Visual C# and Microsoft Visual C++ languages. A group of Transact-SQL statements can be enclosed in a TRY block. If an error occurs in the TRY block, control is passed to another group of statements that is enclosed in a CATCH block.

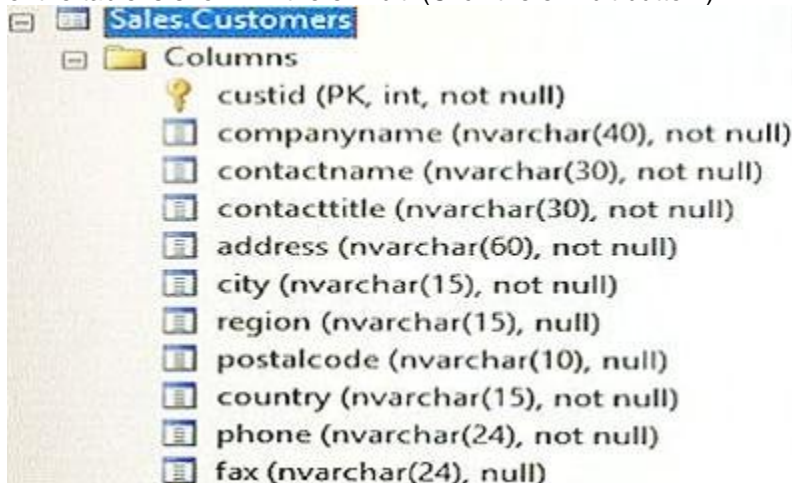
Box 2: RAISERROR(50555, 14, 1 'The update failed.') WITH LOG We must use RAISERROR to be able to specify the required severity level of 14, and we should also use the LOG option, which Logs the error in the error log and the application log for the instance of the Microsoft SQL Server Database Engine, as this enable a MS MS SQL SERVER alert to be triggered.

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

QUESTION 17

Drag and Drop Question

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



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

The stored procedure must meet the following requirements:

- Accept two input parameters.

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

Which five Transact-SQL segments should you use to develop the solution? To answer, move the appropriate Transact-SQL segments from the list of Transact-SQL segments to the answer area and arrange them in the correct order. NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.

Transact-SQL segments

CREATE PROCEDURE Sales.ModCompanyName @custID int, @newname nvarchar(40) AS
IF NOT EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID
BEGIN THROW 55555, 'The customer ID does not exist', 1 END
UPDATE Sales.Customers SET companyname = @custID WHERE custid = @newname
IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
ROLLBACK TRANSACTION

Answer Area



Answer:

Transact-SQL segments

CREATE PROCEDURE Sales.ModCompanyName @custID int, @newname nvarchar(40) AS
IF NOT EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID
BEGIN THROW 55555, 'The customer ID does not exist', 1 END
UPDATE Sales.Customers SET companyname = @custID WHERE custid = @newname
IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
ROLLBACK TRANSACTION

Answer Area

CREATE PROCEDURE Sales.ModCompanyName @custID int, @newname nvarchar(40) AS
IF EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
UPDATE Sales.Customers SET companyname = @newname WHERE custid = @custID
IF NOT EXISTS (SELECT custid FROM Sales.Customers WHERE custid = @custID)
BEGIN THROW 55555, 'The customer ID does not exist', 1 END

QUESTION 18

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

Answer: AC





QUESTION 19

Drag and Drop Question

You have two tables named UserLogin and Employee respectively. You need to create a Transact-SQL script that meets the following requirements:

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

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

Code segments	Answer Area
BEGIN CATCH RAISERROR ('No tables updated!', 16, 1) END CATCH	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">   </div> <div style="text-align: center;">   </div> </div>
UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1	
BEGIN TRY UPDATE dbo.UserLogin SET IsDeleted = 1 WHERE Id = 1	
BEGIN TRY UPDATE dbo.UserLogin SET IsDeleted = 1 WHERE Id = 1 UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1	
BEGIN CATCH	
BEGIN TRY UPDATE dbo.Employee SET IsDeleted = 1 WHERE Id = 1	
END CATCH	

Answer:

Code segments

```

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

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

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

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

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

Answer Area

```

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

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

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

Explanation:

A TRY block must be immediately followed by an associated CATCH block. Including any other statements between the END TRY and BEGIN CATCH statements generates a syntax error.

References: <https://msdn.microsoft.com/en-us/library/ms175976.aspx>

QUESTION 20

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

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

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

- Include the average normalized readings and nearest mountain name.
- Exclude sensors for which no normalized reading exists.
- Exclude those sensors with value of zero for tremor.

Construct the query using the following guidelines:

- Use one part names to reference tables, columns and functions.
- Do not use parentheses unless required.
- Do not use aliases for column names and table names.
- Do not surround object names with square brackets.

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURRENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT
DEFAULT	ON	TSEQUAL
DELETE	OPEN	UNION
DENY	OPENDATASOURCE	UNIQUE
DESC	OPENQUERY	UNPIVOT
DISK	OPENROWSET	UPDATE
DISTINCT	OPENXML	UPDATETEXT
DISTRIBUTED	OPTION	USE
DOUBLE	OR	USER
DROP	ORDER	VALUES
DUMP	OUTER	VARYING
ELSE	OVER	VIEW
END	PERCENT	WAITFOR
ERRLVL	PIVOT	WHEN
ESCAPE	PLAN	WHERE
ESCEPT	PRECISION	WHILE
EXEC	PRIMARY	WITH
EXECUTE	PRINT	WITHIN GROUP
EXISTS		WRITETEXT

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

1 select

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

Answer: GROUP BY

Explanation:

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

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

QUESTION 21

Drag and Drop Question

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

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

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

Transact-SQL segments

```
SET title = 'Customer Representative'

WHERE title = 'Sales Representative'
AND city = 'Seattle' AND mgrid IS NOT
NULL

UPDATE HR.Employees

SET city = 'Seattle' and mgrid = NULL

INSERT INTO HR.Employees

VALUES ('Customer Representative')

WHERE title = 'Sales Representative'

DELETE FROM HR.Employees
```

Answer Area



Answer:

Transact-SQL segments

```
SET title = 'Customer Representative'

WHERE title = 'Sales Representative'
AND city = 'Seattle' AND mgrid IS NOT
NULL

UPDATE HR.Employees

SET city = 'Seattle' and mgrid = NULL

INSERT INTO HR.Employees

VALUES ('Customer Representative')

WHERE title = 'Sales Representative'

DELETE FROM HR.Employees
```

Answer Area

UPDATE HR.Employees

SET title = 'Customer Representative'

WHERE title = 'Sales Representative'
AND city = 'Seattle' AND mgrid IS NOT
NULL



Explanation:

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

QUESTION 22

Hotspot Question

You have the following Transact-SQL query:

```
SELECT
    City.CityID,
    City.CityName,
    TranslateName(Nearby.CityName) AS NearbyCity
FROM Cities AS City
CROSS APPLY NearbyCities(City.CityID) AS Nearby
```

What type of functions are used in the query? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

Answer Area

Function	Type
TranslateName	<div>▼</div> <div>Scalar</div> <div>Table-Valued</div> <div>System</div> <div>Aggregate</div>
NearbyCities	<div>▼</div> <div>Scalar</div> <div>Table-Valued</div> <div>System</div> <div>Aggregate</div>

Answer:

Answer Area

Function	Type
TranslateName	<div>▼</div> <div>Scalar</div> <div>Table-Valued</div> <div>System</div> <div>Aggregate</div>
NearbyCities	<div>▼</div> <div>Scalar</div> <div>Table-Valued</div> <div>System</div> <div>Aggregate</div>

Explanation:

Box 1: Scalar

The return value of a function can either be a scalar (single) value or a table.

Box 2: Table-Valued

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.

References:

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

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

QUESTION 23

Drag and Drop Question

You have a database that includes the following tables:



You need to create a list of all customer IDs and the date of the last order that each customer placed. If the customer has not placed any orders, you must return the date January 1, 1900. The column names must be CustomerID and LastOrderDate. Which four Transact-SQL segments should you use to develop the solution? To answer, move the appropriate Transact-SQL segments from the list of Transact-SQL segments to the answer area and arrange them in the correct order.

Transact-SQL segments

GROUP BY c.custid

FROM sales.Customers AS c INNER JOIN sales.Orders AS o

ON c.orderid = o.orderid

SELECT c.custid AS CustomerID, MAX(o.orderdate) AS LastOrderDate

FROM sales.Customers AS c LEFT OUTER JOIN sales.Orders AS o

GROUP BY LastOrderDate

ON c.custid = o.custid

SELECT c.custid AS CustomerID, COALESCE (MAX(o.orderdate), '19000101') AS LastOrderDate

Answer Area



Answer:

Transact-SQL segments

```
GROUP BY c.custid

FROM sales.Customers AS c INNER
JOIN sales.Orders AS o

ON c.orderid = o.orderid

SELECT c.custid AS CustomerID,
MAX(o.orderdate) AS LastOrderDate

FROM sales.Customers AS c LEFT
OUTER JOIN sales.Orders AS o

GROUP BY LastOrderDate

ON c.custid = o.custid

SELECT c.custid AS CustomerID,
COALESCE (MAX(o.orderdate),
'19000101') AS LastOrderDate
```

Answer Area

```
SELECT c.custid AS CustomerID,
COALESCE (MAX(o.orderdate),
'19000101') AS LastOrderDate

FROM sales.Customers AS c LEFT
OUTER JOIN sales.Orders AS o

ON c.custid = o.custid

GROUP BY c.custid
```

Explanation:

Box 1: SELECT...COALESCE...

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

Box 2: LEFT OUTER JOIN...

The LEFT JOIN (LEFT OUTER JOIN) keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match. A customer might have no orders so the right table must be allowed have a NULL value.

Box 3: ON c.custid = o.custid

We JOIN on the custID column, which is available in both tables.

Box 4: GROUP BY c.custid

References:

[https://technet.microsoft.com/en-us/library/ms189499\(v=sql.110\).aspx](https://technet.microsoft.com/en-us/library/ms189499(v=sql.110).aspx)

http://www.w3schools.com/sql/sql_join_left.asp

QUESTION 24

Hotspot Question

You run the following Transact-SQL statement:

```
CREATE TABLE Sales.Customers (
    custid int IDENTITY(1,1) NOT NULL,
    companyname nvarchar(50) NULL,
    contacttitle nvarchar(30) NOT NULL,
    address nvarchar(60) NOT NULL,
    postalcode nvarchar(10) NOT NULL,
    region nvarchar(15) NULL,
    phone nvarchar(24) NOT NULL,
    fax nvarchar(24) NULL,
) ON PRIMARY
```

You need to ensure that you can insert data into the table. What are the characteristics of the data? To answer, select the appropriate options in the answer area.

Answer Area

Column input constraint

Values cannot be entered into this column

A value must be inserted into this column

Data entry into this column is optional

Column name

	▼
custid	
fax	
postalcode	
region	

	▼
custid	
fax	
postalcode	
region	

	▼
custid	
fax	
postalcode	
region	

Answer:

Answer Area

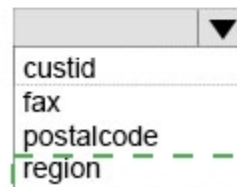
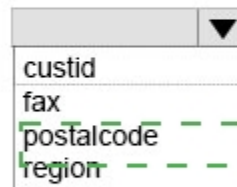
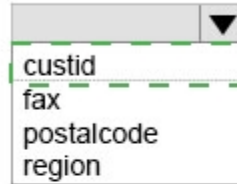
Column input constraint

Values cannot be entered into this column

A value must be inserted into this column

Data entry into this column is optional

Column name



Explanation:

Box 1: custid

IDENTITY indicates that the new column is an identity column. When a new row is added to the table, the Database Engine provides a unique, incremental value for the column. Identity columns are typically used with PRIMARY KEY constraints to serve as the unique row identifier for the table.

Box 2: postalcode

postalcode is declared as NOT NULL, which means that a value must be inserted.

Box 3: region

Fax is also a correct answer. Both these two columns are declared as NULL, which means that data entry is optional.

References: <https://msdn.microsoft.com/en-us/library/ms174979.aspx>

QUESTION 25

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

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

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

- removes orders from the table that were placed before January 1, 2012
- uses the date format of YYYYMMDD
- ensures that the order has been shipped before deleting the record

Construct the query using the following guidelines:

- use one-part column names and two-part table names

- do not use functions
- do not surround object names with square brackets
- do not use variables
- do not use aliases for column names and table names

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURRENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT
DEFAULT	ON	TSEQUAL
DELETE	OPEN	UNION
DENY	OPENDATASOURCE	UNIQUE
DESC	OPENQUERY	UNPIVOT
DISK	OPENROWSET	UPDATE
DISTINCT	OPENXML	UPDATETEXT
DISTRIBUTED	OPTION	USE
DOUBLE	OR	USER
DROP	ORDER	VALUES
DUMP	OUTER	VARYING
ELSE	OVER	VIEW
END	PERCENT	WAITFOR
ERRLVL	PIVOT	WHEN
ESCAPE	PLAN	WHERE
ESCEPT	PRECISION	WHILE
EXEC	PRIMARY	WITH
EXECUTE	PRINT	WITHIN GROUP
EXISTS		WRITETEXT

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

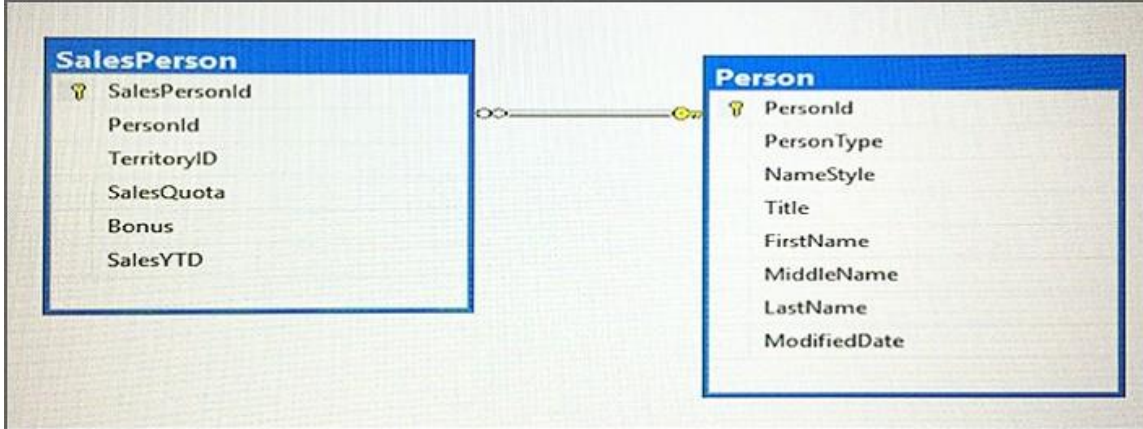


Use the Check Syntax button to verify your work. Any syntax or spelling errors will be reported by line and character position.

Answer: Pending

QUESTION 26

You have a database that contains the following tables.



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

- Return a column named Fullname that includes the salesperson FirstName, a space, and then LastName.
- Include the current year-to-date sales for each salesperson.
- Display only data for the three salespersons with the lowest year-to-year sales values.
- Exclude salespersons that have no value for TerritoryID.

Construct the query using the following guidelines:

- Use the first letter of a table name as the table alias.
- Use two-part column names.
- Do not surround object names with square brackets.
- Do not use implicit joins.
- Use only single quotes for literal text.
- Use aliases only if required.

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURRENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT
DEFAULT	ON	TSEQUAL
DELETE	OPEN	UNION
DENY	OPENDATASOURCE	UNIQUE
DESC	OPENQUERY	UNPIVOT
DISK	OPENROWSET	UPDATE
DISTINCT	OPENXML	UPDATETEXT
DISTRIBUTED	OPTION	USE
DOUBLE	OR	USER
DROP	ORDER	VALUES
DUMP	OUTER	VARYING
ELSE	OVER	VIEW
END	PERCENT	WAITFOR
ERRLVL	PIVOT	WHEN
ESCAPE	PLAN	WHERE
ESCEPT	PRECISION	WHILE
EXEC	PRIMARY	WITH
EXECUTE	PRINT	WITHIN GROUP
EXISTS		WRITETEXT

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

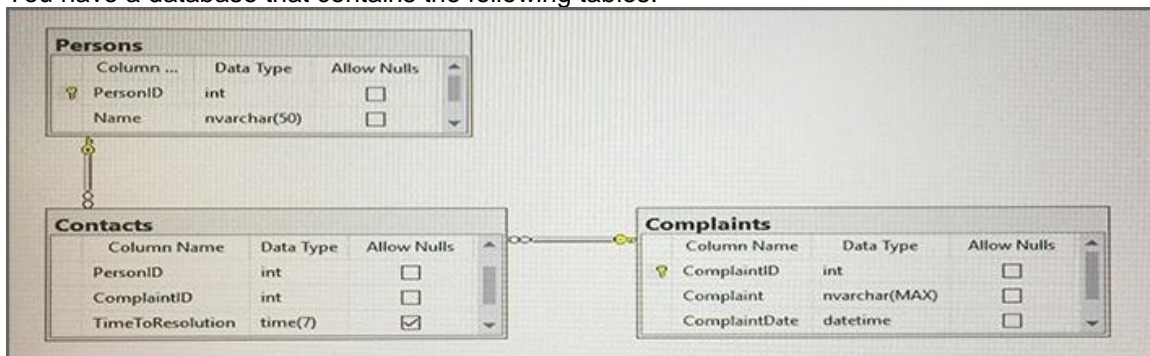
```
1 SELECT
2 FROM Person AS P INNER JOIN SalesPerson AS S
3 ON P.PersonID = S.SalesPersonID
4 WHERE
```

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

Answer: Pending

QUESTION 27

You have a database that contains the following tables.



You need to create a query that lists all complaints from the Complaints table, and the name of the person handling the complaints if a person is assigned. The ComplaintID must be displayed first, followed by the person name. Construct the query using the following guidelines:

- Use two-part column names.
- Use one-part table names.
- Do not use aliases for column names or table names.
- Do not use Transact-SQL functions.
- Do not use implicit joins.
- Do not surround object names with square brackets.

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

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURRENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT
DEFAULT	ON	TSEQUAL
DELETE	OPEN	UNION
DENY	OPENDATASOURCE	UNIQUE
DESC	OPENQUERY	UNPIVOT
DISK	OPENROWSET	UPDATE
DISTINCT	OPENXML	UPDATETEXT
DISTRIBUTED	OPTION	USE
DOUBLE	OR	USER
DROP	ORDER	VALUES
DUMP	OUTER	VARYING
ELSE	OVER	VIEW
END	PERCENT	WAITFOR
ERRLVL	PIVOT	WHEN
ESCAPE	PLAN	WHERE
ESCEPT	PRECISION	WHILE
EXEC	PRIMARY	WITH
EXECUTE	PRINT	WITHIN GROUP
EXISTS		WRITETEXT

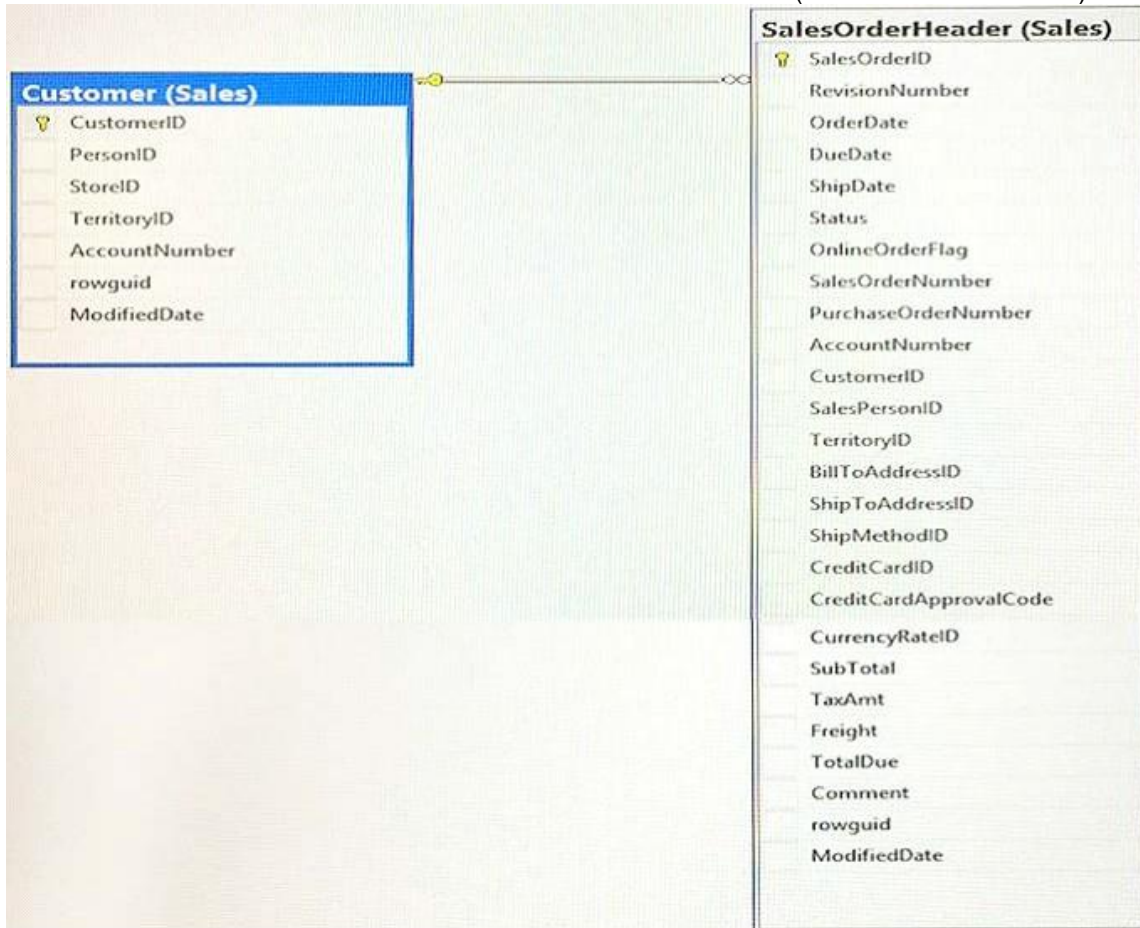

```
1 SELECT Complaints.ComplaintId,
2 FROM
3 JOIN
4 JOIN
```

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

Answer: Pending

QUESTION 28

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 0 1/01/1990 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

Answer: A

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 29

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	

Customer Audit:

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

Answer: D

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 table variable.

QUESTION 30

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

Drag and Drop Question

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

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

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

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

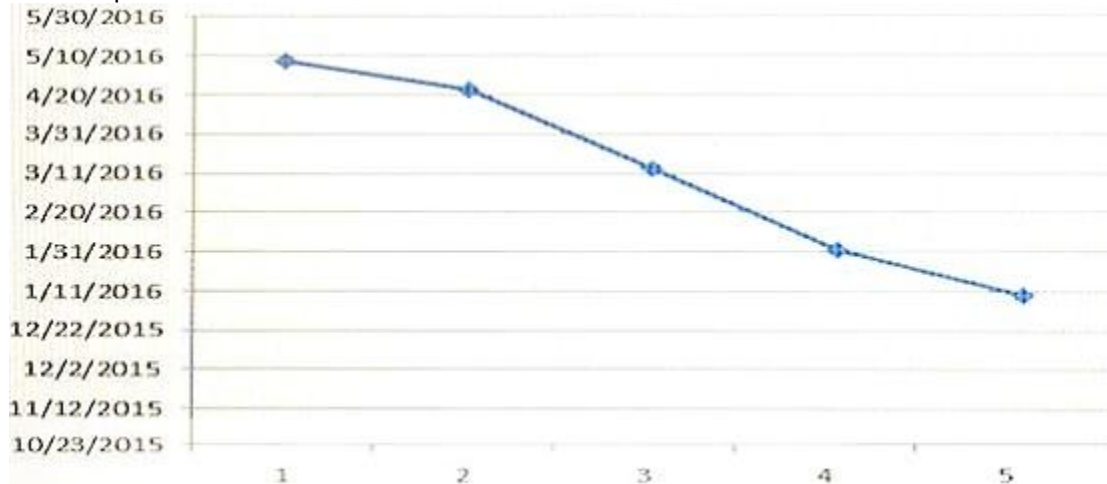
Details for the Sales.CustomerCategories table are shown in the following table:

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

You are creating a report to show when the first customer account was opened in each city. The report contains a line chart with the following characteristics:

- The chart contains a data point for each city, with lines connecting the points.
- The X axis contains the position that the city occupies relative to other cities.
- The Y axis contains the date that the first account in any city was opened.

An example chart is shown below for five cities:



During a sales promotion, customers from various cities open new accounts on the same date. You need to write a query that returns the data for the chart. How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct locations. Each Transact-SQL segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content. NOTE: Each correct selection is worth one point.

Transact-SQL segments

DENSE_RANK() OVER

RANK() OVER

(ORDER BY MIN(AccountOpenedDate) DESC)

(PARTITION BY CityID ORDER BY min(AccountOpenedDate) DESC)

(ORDER BY AccountOpenedDate DESC)

(PARTITION BY CityID ORDER BY AccountOpenedDate DESC)

GROUP BY CityID

GROUP BY PARTITION

Answer Area

SELECT

CityID,
MIN(AccountOpenedDate),

Transact-SQL segment

Transact-SQL segment

FROM Application.Citites
INNER JOIN Sales.Customers ON CityID = PostalCityID

Transact-SQL segment

ORDER BY MIN(AccountOpenedDate) DESC

Answer:

Transact-SQL segments

DENSE_RANK() OVER

RANK() OVER

(ORDER BY MIN(AccountOpenedDate)
DESC)

(PARTITION BY CityID ORDER BY
min(AccountOpenedDate) DESC)

(ORDER BY AccountOpenedDate DESC)

(PARTITION BY CityID ORDER BY
AccountOpenedDate DESC)

GROUP BY CityID

GROUP BY PARTITION

Answer Area

```
SELECT
    CityID,
    MIN(AccountOpenedDate),
    RANK() OVER
    (PARTITION BY CityID ORDER BY
    min(AccountOpenedDate) DESC)
FROM Application.Cities
INNER JOIN Sales.Customers ON CityID = PostalCityID
GROUP BY CityID
ORDER BY MIN(AccountOpenedDate) DESC
```

Explanation:

Box 1: RANK() OVER

RANK returns the rank of each row within the partition of a result set. The rank of a row is one plus the number of ranks that come before the row in question. ROW_NUMBER and RANK are similar. ROW_NUMBER numbers all rows sequentially (for example 1, 2, 3, 4, 5).

QUESTION 31

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

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

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

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

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

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

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

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

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

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

- A
- ```
SELECT CustomerID, LatestRecordedPopulation
FROM Sales.Customers
CROSS JOIN Application.Cities
WHERE (IsOnCreditHold = 0 AND DeliveryCityID = CityID)
OR (IsOnCreditHold = 1 AND PostalCityID = CityID)
```
- B
- ```
SELECT CustomerID, LatestRecordedPopulation
FROM Sales.Customers
INNER JOIN Application.Cities AS A
ON A.CityID = IIF(IsOnCreditHold = 0, DeliveryCityID, PostalCityID)
```
- C
- ```
SELECT CustomerID, ISNULL(A.LatestRecordedPopulation, B.LatestRecordedPopulation)
FROM Sales.Customers
INNER JOIN Application.Cities AS A ON A.CityID = DeliveryCityID
INNER JOIN Application.Cities AS B ON B.CityID = PostalCityID
WHERE IsOnCreditHold = 0
```
- D
- ```
SELECT CustomerID, LatestRecordedPopulation,
IIF(IsOnCreditHold = 0, DeliveryCityID, PostalCityID) As CityId
FROM Sales.Customers
INNER JOIN Application.Cities AS A ON A.CityID = CityId
```

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

Answer: AB

Explanation:

Using Cross Joins

A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table. However, if a WHERE clause is added, the cross join behaves as an inner join.

B: You can use the IIF in the ON-statement.

IIF returns one of two values, depending on whether the Boolean expression evaluates to true or false in SQL Server.

References:

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

<https://msdn.microsoft.com/en-us/library/hh213574.aspx>

QUESTION 32

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

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

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

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

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

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

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

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

- A
- ```
DECLARE @sdate DATETIME, @edate DATETIME
SET @sdate = DATEFROMPARTS (2013, 12, 31)
set @edate = DATEADD(d, 1, @sdate)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME ALL
WHERE ValidFrom > @sdate AND ValidTo < @edate
```
- B
- ```
DECLARE @sdate DATETIME, @edate DATETIME
SET @sdate = DATEFROMPARTS (2013, 12, 31)
set @edate = DATEADD(d, -1, @sdate)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME BETWEEN @sdate AND @edate
```
- C
- ```
DECLARE @date DATE
SET @date = DATEFROMPARTS (2013, 12, 31)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME AS OF @date
```
- D
- ```
DECLARE @date DATE
SET @date = DATEFROMPARTS (2013, 12, 31)
SELECT * FROM Sales.Customers WHERE @date BETWEEN ValidFrom AND ValidTo
```

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

Answer: D

Explanation:

The datetime 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.

QUESTION 33

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

Drag and Drop Question

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

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

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

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

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

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

You are creating a report to measure the impact of advertising efforts that were designed to attract new customers. The report must show the number of new customers per day for each customer category, but only if the number of new customers is greater than five. You need to write the query to return data for the report. How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct locations. Each Transact-SQL segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

Transact-SQL segments

CAST(Cust.AccountOpenedDate AS DATE)

DATEPART(day, Cust.AccountOpenedDate)

HAVING

WHERE

COUNT(Cust.CustomerId)

MAX(Cust.CustomerId)

RANK

GROUP BY

Answer Area

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

Transact-SQL segment

FROM Sales.Customers AS Cust
INNER JOIN Sales.CustomerCategories AS CustCat
ON Cust.CustomerCategoryID = CustCat.CustomerCategoryID

Transact-SQL segment

CustCat.CustomerCategoryName,

Transact-SQL segment

Transact-SQL segment

Transact-SQL segment

> 5

Answer:

Transact-SQL segments

CAST(Cust.AccountOpenedDate
AS DATE)

DATEPART(day,
Cust.AccountOpenedDate)

HAVING

WHERE

COUNT(Cust.CustomerId)

MAX(Cust.CustomerID)

RANK

GROUP BY

Answer Area

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

QUESTION 34

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

Drag and Drop Question

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

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

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

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

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

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

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

Transact-SQL segments

0, 1, 2, 3, 4

(0...4)

BETWEEN 0 AND 4

PIVOT

GROUP BY

[CreditLimit]

AVG(CreditLimit)

Answer Area

```

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

Answer:

Transact-SQL segments

0, 1, 2, 3, 4

(0...4)

BETWEEN 0 AND 4

PIVOT

GROUP BY

[CreditLimit]

AVG(CreditLimit)

Answer Area

```

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

Explanation:

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

Pivot example:

-- Pivot table with one row and five columns

SELECT 'AverageCost' AS Cost_Sorted_By_Production_Days, [0], [1], [2], [3], [4]

FROM

(SELECT DaysToManufacture, StandardCost

FROM Production.Product) AS SourceTable

PIVOT

(

AVG(StandardCost)

FOR DaysToManufacture IN ([0], [1], [2], [3], [4])) AS PivotTable;

Box 2: [CreditLimit]

Box 3: PIVOT

You can use the PIVOT and UNPIVOT relational operators to change a table-valued expression into another table. PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output, and performs aggregations where they are required on any remaining column values that are wanted in the final output.

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

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

Syntax: test_expression [NOT] IN (subquery | expression [,...n]) Where expression[,... n] is a list of expressions to test for a match. All expressions must be of the same type as test_expression.

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

QUESTION 35

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

Drag and Drop Question

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

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

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

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

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

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

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

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

Transact-SQL segments

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

Answer Area

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

SELECT CityID
FROM Application.Cities
[Transact-SQL segment] LatestRecordedPopulation > 10000
[Transact-SQL segment]
[Transact-SQL segment]
```

Answer:

Transact-SQL segments

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

Answer Area

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

SELECT CityID
FROM Application.Cities
[WHERE] LatestRecordedPopulation > 10000
[AND [IsOnCreditHold] = 0]
[)]
```

Explanation:

Box 1: IN (

The IN clause determines whether a specified value matches any value in a subquery or a list. Syntax: test_expression [NOT] IN (subquery | expression [,...n]) Where subquery is a subquery that has a result set of one column. This column must have the same data type as test_expression.

Box 2: WHERE

Box 3: AND [IsOnCreditHold] = 0

Box 4:)

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

QUESTION 36

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

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

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

Answer: F

Explanation:

The ISNULL function replaces NULL with the specified replacement value.

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

QUESTION 37

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

You have a database that is denormalized. Users make frequent changes to data in a primary table. You need to ensure that users cannot change the tables directly, and that changes made to the primary table also update any related tables. What should you implement?

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

Answer: B

Explanation:

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

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

QUESTION 38

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

You have a database that stores sales and order information. Users must be able to extract information from the tables on an ad hoc basis. They must also be able to reference the extracted information as a single table. You need to implement a solution that allows users to retrieve the data required, based on variables defined at the time of the query. What should you implement?

- A. the COALESCE function
- B. a view
- C. a table-valued function

- D. the TRY PARSE function
- E. a stored procedure
- F. the ISNULL function
- G. a scalar function
- H. the TRY CONVERT function

Answer: C

Explanation:

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

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

QUESTION 39

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

You have a table named AuditTrail that tracks modifications to data in other tables. The AuditTrail table is updated by many processes. Data input into AuditTrail may contain improperly formatted date time values. You implement a process that retrieves data from the various columns in AuditTrail, but sometimes the process throws an error when it is unable to convert the data into valid date time values. You need to convert the data into a valid date time value using the en-US format culture code. If the conversion fails, a null value must be returned in the column output. The conversion process must not throw an error. What should you implement?

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

Answer: H

Explanation:

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

References: <https://msdn.microsoft.com/en-us/library/hh230993.aspx>

QUESTION 40

Hotspot Question

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

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

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

Answer Area

Subquery name

Subquery replacement

Subquery1

▼

common table expression (CTE)

local temporary table

global temporary table

Subquery2

▼

common table expression (CTE)

local temporary table

global temporary table

Subquery3

▼

common table expression (CTE)

local temporary table

global temporary table

Answer:

Answer Area

Subquery name	Subquery replacement
Subquery1	<div>▼</div> <div>common table expression (CTE)</div> <div>local temporary table</div> <div>global temporary table</div>
Subquery2	<div>▼</div> <div>common table expression (CTE)</div> <div>local temporary table</div> <div>global temporary table</div>
Subquery3	<div>▼</div> <div>common table expression (CTE)</div> <div>local temporary table</div> <div>global temporary table</div>

Explanation:

Subquery1: common table expression (CTE)

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

Subquery2: global temporary table

Global temporary tables are visible to any user and any connection after they are created, and are deleted when all users that are referencing the table disconnect from the instance of SQL Server.

Subquery3: local temporary table

Local temporary tables are visible only to their creators during the same connection to an instance of SQL Server as when the tables were first created or referenced. Local temporary tables are deleted after the user disconnects from the instance of SQL Server.

References:

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

<https://technet.microsoft.com/en-us/library/ms186986.aspx>

QUESTION 41

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

Servers:

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

Errors:

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

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

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

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

Answer: C

QUESTION 42

Drag and Drop Question

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 are building a webpage that shows the three most common errors for each server. You need to return the data for the webpage. How should you complete the Transact-SQL statement? To answer, drag the appropriate Transact-SQL segments to the correct location. Each Transact-SQL segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content. NOTE: Each correct selection is worth one point.

Transact-SQL segments

Answer Area

svr.ServerID	<div> <div>SELECT Transact-SQL segment , errs.LogMessage</div> <div>FROM Servers AS svr</div> <div>Transact-SQL segment</div> <div>{</div> <div> SELECT TOP 3 LogMessage</div> <div> FROM Errors</div> <div> Transact-SQL segment</div> <div> ORDER BY Occurrences</div> <div>} AS errs</div> </div>
errs.ServerID	
INNER JOIN	
CROSS APPLY	
WITHIN GROUP	
WHERE ServerID = svr.ServerID	
WHERE ServerID = errs.ErrorID	

Answer:

Transact-SQL segments

svr.ServerID

errs.ServerID

INNER JOIN

CROSS APPLY

WITHIN GROUP

WHERE ServerID =
svr.ServerID

WHERE ServerID =
errs.ErrorID

Answer Area

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

QUESTION 43

You have a table named Cities that has the following two columns: CityID and CityName. The CityID column uses the int data type, and CityName uses nvarchar(max). You have a table named RawSurvey. Each row includes an identifier for a question and the number of persons that responded to that question from each of four cities. The table contains the following representative data:

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

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

- Retrieve data from the RawSurvey table in the format of the SurveyReport table.
- The CityID must contain the CityID of the city that was surveyed.
- The order of cities in all SELECT queries must match the order in the RawSurvey table.
- The order of cities in all IN statements must match the order in the RawSurvey table.

Construct the query using the following guidelines:

- Use one-part names to reference tables and columns, except where not possible.
- ALL SELECT statements must specify columns.
- Do not use column or table aliases, except those provided.
- Do not surround object names with square brackets.

Keywords

ADD	EXIT	PROC
ALL	EXTERNAL	PROCEDURE
ALTER	FETCH	PUBLIC
AND	FILE	RAISERROR
ANY	FILLFACTOR	READ
AS	FORFOREIGN	READTEXT
ASC	FREETEXT	RECONFIGURE
AUTHORIZATION	FREETEXTTABLE	REFERENCES
BACKUP	FROM	REPLICATION
BEGIN	FULL	RESTORE
BETWEEN	FUNCTION	RESTRICT
BREAK	GOTO	RETURN
BROWSE	GRANT	REVERT
BULK	GROUP	REVOKE
BY	HAVING	RIGHT
CASCADE	HOLDLOCK	ROLLBACK
CASE	IDENTITY	ROWCOUNT
CHECK	IDENTITY_INSERT	ROWGUIDCOL
CHECKPOINT	IDENTITYCOL	RULE
CLOSE	IF	SAVE
CLUSTERED	IN	SCHEMA
COALESCE	INDEX	SECURITYAUDIT
COLLATE	INNER	SELECT
COLUMN	INSERT	SEMANTICKEYPHRASETABLE
COMMIT	INTERSECT	SEMANTICSIMILARITYDETAILSTABLE
COMPUTE	INTO	SEMANTICSIMILARITYTABLE
CONCAT	IS	SESSION_USER
CONSTRAINT	JOIN	SET
CONTAINS	KEY	SETUSER
CONTAINSTABLE	KILL	SHUTDOWN
CONTINUE	LEFT	SOME
CONVERT	LIKE	STATISTICS
CREATE	LINENO	SYSTEM_USER
CROSS	LOAD	TABLE
CURRENT	MERGE	TABLESAMPLE
CURRENT_DATE	NATIONAL	TEXTSIZE
CURRENT_TIME	NOCHECK	THEN
CURRENT_TIMESTAMP	NONCLUSTERED	TO
CURRENT_USER	NOT	TOP
CURSOR	NULL	TRAN
DATABASE	NULLIF	TRANSACTION
DBCC	OF	TRIGGER
DEALLOCATE	OFF	TRUNCATE
DECLARE	OFFSETS	TRY_CONVERT
DEFAULT	ON	TSEQUAL
DELETE	OPEN	UNION
DENY	OPENDATASOURCE	UNIQUE
DESC	OPENQUERY	UNPIVOT
DISK	OPENROWSET	UPDATE
DISTINCT	OPENXML	UPDATETEXT
DISTRIBUTED	OPTION	USE
DOUBLE	OR	USER
DROP	ORDER	VALUES
DUMP	OUTER	VARYING
ELSE	OVER	VIEW
END	PERCENT	WAITFOR
ERRLVL	PIVOT	WHEN
ESCAPE	PLAN	WHERE
ESCEPT	PRECISION	WHILE
EXEC	PRIMARY	WITH
EXECUTE	PRINT	WITHIN GROUP
EXISTS		WRITETEXT

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

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

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

Answer: UNPIVOT

Explanation:

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

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

QUESTION 44

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 Is Active column indicates that a user is active. You need to create a count for active users in each role. If a role has no active users, you must display a zero as the active users count. Which Transact-SQL statement should you run?

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

A. Option A

B. Option B

- C. Option C
- D. Option D

Answer: C

QUESTION 45

Drag and Drop Question

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

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

For reporting purposes, you need to find the active user count for each role, and the total active user count. The result must be ordered by active user count of each role. You must use common table expressions (CTEs). Which four Transact-SQL segments should you use to develop the solution? To answer, move the appropriate Transact-SQL segments from the list of Transact-SQL segments to the answer area and arrange them in the correct order.

Transact-SQL segments

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

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

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

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

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

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

Answer Area



Answer:

Transact-SQL segments

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

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

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

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

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

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

Answer Area

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

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

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



QUESTION 46

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

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

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

The tables include the following records:
Customer_CRMSystem:

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

Customer_HRSystem:

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

Records that contain null values for CustomerCode can be uniquely identified by CustomerName. You need to display a list of customers that do not appear in the Customer_HRSystem table. Which Transact-SQL statement should you run?

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

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

Answer: D

Explanation:

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

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

QUESTION 47

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 Customer Name. You need to display customers who appear in both tables and have a proper CustomerCode. Which Transact-SQL statement should you run?

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

A. Option A

- B. Option B
- C. Option C
- D. Option D
- E. Option E

Answer: A

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 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 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 Customer Name. You need to display a Cartesian product, combining both tables. Which Transact-SQL statement should you run?

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

- A. Option A
B. Option B

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

Answer: G

Explanation:

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

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

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

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

A. Option A

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

Answer: E

Explanation:

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

QUESTION 50

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.

Drag and Drop Question

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

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

The following table describes the columns in Sales.Orders:

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

The following table describes the columns in Sales.OrderLines:

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

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

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

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

Transact-SQL segments

Answer Area

COUNT
SUM
AVG
ORDER BY
GROUP BY
RETURNS INT
RETURNS NULL ON NULL INPUT
RETURNS TABLE

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

Answer:

Transact-SQL segments

Answer Area

COUNT
SUM
AVG
ORDER BY
GROUP BY
RETURNS INT
RETURNS NULL ON NULL INPUT
RETURNS TABLE

```
CREATE FUNCTION Sales.GetCustomerInformation(@CustomerID int)
    RETURNS TABLE
AS
RETURN
(
    SELECT C.CustomerName, C.PhoneNumber, C.AccountOpenedDate,
    C.StandardDiscountPercentage, C.CreditLimit, C.IsOnCreditHold,
    COUNT (O.OrderID) AS TotalNumberOfOrders,
    SUM (OL.UnitPrice) AS TotalOrderPrice,
    AVG (OL.Quantity) AS AverageOrderQuantity
    FROM Sales.Customers C
    JOIN Sales.Orders AS O ON O.CustomerID = C.CustomerID
    JOIN Sales.OrderLines AS OL ON OL.OrderID = O.OrderID
    WHERE C.CustomerID = @CustomerID
    GROUP BY
    C.CustomerName, C.PhoneNumber, C.AccountOpenedDate,
    C.StandardDiscountPercentage, C.CreditLimit, C.IsOnCreditHold
)
```

Explanation:

Box1: RETURNS TABLE

The function should return the following information:

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

Box 2: COUNT

The function should return the total number of orders for the customer.

Box 3: SUM

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

Box 3: AVG

The function should return the average quantity of items per order.

Box 4: GROUP BY

Need to use GROUP BY for the aggregate functions.

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

QUESTION 51

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.

Hotspot Question

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

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

The following table describes the columns in Sales.Orders:

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

The following table describes the columns in Sales.OrderLines:

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

You need to create a database object that calculates the total price of an order including the sales tax. The database object must meet the following requirements:

- Reduce the compilation cost of Transact-SQL code by caching the plans and reusing them for repeated execution.
- Return a value.
- Be callable from a SELECT statement.

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

Answer Area

```
CREATE PROCEDURE Sales.CalculateOrderPrice

(
    @orderID int
)

WITH EXECUTE AS OWNER
RETURNS decimal(18,2)
RETURNS TABLE

AS

BEGIN TRAN
BEGIN
RETURN

    DECLARE @OrderPrice decimal(18,2)
    DECLARE @CalculatedTaxRate decimal(18,2)
    SET @OrderPrice = (SELECT SUM(Quantity * UnitPrice) FROM Sales.OrderLines WHERE OrderID = @OrderID)
    SET @CalculatedTaxRate = (SELECT 1 + (MAX(TaxRate) / 100) FROM Sales.OrderLines WHERE OrderID = @OrderID)

    RETURN ( @OrderPrice * @CalculatedTaxRate
              SELECT (#OrderPrice * #CalculatedTaxRate) AS CalculatedOrderPrice
              CalculateOrderPrice
            )

RETURN
COMMIT
END
```

Answer:

Answer Area

```
CREATE PROCEDURE Sales.CalculateOrderPrice

(
    @orderID int
)

WITH EXECUTE AS OWNER
RETURNS decimal(18,2)
RETURNS TABLE

AS

BEGIN TRAN
BEGIN
RETURN

    DECLARE @OrderPrice decimal(18,2)
    DECLARE @CalculatedTaxRate decimal(18,2)
    SET @OrderPrice = (SELECT SUM(Quantity * UnitPrice) FROM Sales.OrderLines WHERE OrderID = @OrderID)
    SET @CalculatedTaxRate = (SELECT 1 + (MAX(TaxRate) / 100) FROM Sales.OrderLines WHERE OrderID = @OrderID)

    RETURN ( @OrderPrice * @CalculatedTaxRate
              SELECT (#OrderPrice * #CalculatedTaxRate) AS CalculatedOrderPrice
              CalculateOrderPrice
            )

RETURN
COMMIT
END
```

Explanation:

Box 1: FUNCTION

To be able to return a value we should use a scalar function.

CREATE FUNCTION creates a user-defined function in SQL Server and Azure SQL Database. The return value can either be a scalar (single) value or a table.

Box 2: RETURNS decimal(18,2)

Use the same data format as used in the UnitPrice column.

Box 3: BEGIN

Transact-SQL Scalar Function Syntax include the BEGIN ..END construct.

```
CREATE [ OR ALTER ] FUNCTION [ schema_name. ] function_name ( [ { @parameter_name
[ AS ][ type_schema_name. ] parameter_data_type [ = default ] [ READONLY ] }
[ ,...n ]
]
)
```

RETURNS return_data_type

[WITH <function_option> [,...n]]

[AS]

BEGIN

function_body

RETURN scalar_expression

END

[;]

Box 4: @OrderPrice * @CalculatedTaxRate

Calculate the price including tax.

Box 5: END

Transact-SQL Scalar Function Syntax include the BEGIN ..END construct.

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

QUESTION 52

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.

Drag and Drop Question

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

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

The following table describes the columns in Sales.Orders:

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

The following table describes the columns in Sales.OrderLines:

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

You need to create a stored procedure that inserts data into the Customers table. The stored procedure must meet the following requirements:

- Data changes occur as a single unit of work.
- Data modifications that are successful are committed and a value of 0 is returned.
- Data modifications that are unsuccessful are rolled back. The exception severity level is set to 16 and a value of -1 is returned.
- The stored procedure uses a built-in scalar function to evaluate the current condition of data modifications.
- The entire unit of work is terminated and rolled back if a run-time error occurs during execution of the stored procedure.

How should complete the stored procedure definition? To answer, drag the appropriate Transact-SQL segments to the correct targets. Each Transact-SQL segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content. NOTE: Each correct selection is worth one point.

Transact-SQL segments

- RAISERROR
- THROW
- XACT_ABORT
- XACT_STATE
- @@TRANCOUNT
- ROLLBACK
- COMMIT
- END

Answer Area

```

CREATE PROCEDURE Sales.InsertCustomer
    @CustomerName nvarchar(100),
    @PhoneNumber nvarchar(20),
    @AccountOpenedDate date,
    @StandardDiscountPercentage decimal(18,3),
    @CreditLimit decimal(18,2),
    @IsCreditOnHold bit,
    @DeliveryLongitude nvarchar(50),
    @DeliveryLatitude nvarchar(50)
AS
BEGIN
    SET NOCOUNT ON
    SET  ON

    BEGIN TRY
        BEGIN TRANSACTION
        INSERT INTO Sales.Customers (CustomerName, PhoneNumber, AccountOpenedDate,
            StandardDiscountPercentage, CreditLimit, IsOnCreditHold, DeliveryLocation)
        VALUES
            (@CustomerName, @PhoneNumber, @AccountOpenedDate, @StandardDiscountPercentage,
            @CreditLimit, @IsCreditOnHold, geography::Point(ISNULL(@DeliveryLongitude, ''),
            ISNULL(@DeliveryLatitude, ''), 4326))
         TRANSACTION
    END TRY
    BEGIN CATCH
        IF  () <> 0  TRANSACTION
        PRINT 'Unable to create the customer record.'
        
        RETURN -1
    END CATCH
    RETURN 0
END

```

Answer:

Transact-SQL segments

RAISERROR
THROW
XACT_ABORT
XACT_STATE
@@TRANCOUNT
ROLLBACK
COMMIT
END

Answer Area

```
CREATE PROCEDURE Sales.InsertCustomer
    @CustomerName nvarchar(100),
    @PhoneNumber nvarchar(20),
    @AccountOpenedDate date,
    @StandardDiscountPercentage decimal(18,3),
    @CreditLimit decimal(18,2),
    @IsCreditOnHold bit,
    @DeliveryLongitude nvarchar(50),
    @DeliveryLatitude nvarchar(50)
AS
BEGIN
    SET NOCOUNT ON
    SET XACT_ABORT ON

    BEGIN TRY
        BEGIN TRANSACTION
        INSERT INTO Sales.Customers (CustomerName, PhoneNumber, AccountOpenedDate,
            StandardDiscountPercentage, CreditLimit, IsOnCreditHold, DeliveryLocation)
        VALUES
            (@CustomerName, @PhoneNumber, @AccountOpenedDate, @StandardDiscountPercentage,
            @CreditLimit, @IsCreditOnHold, geography::Point(ISNULL(@DeliveryLongitude, ''),
            ISNULL(@DeliveryLatitude, ''), 4326))
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF XACT_STATE () <> 0 ROLLBACK TRANSACTION
        PRINT 'Unable to create the customer record.'
        THROW
        RETURN -1
    END CATCH
    RETURN 0
END
```

Explanation:

Box 1: XACT_ABORT

XACT_ABORT specifies whether SQL Server automatically rolls back the current transaction when a Transact-SQL statement raises a run-time error. When SET XACT_ABORT is ON, if a Transact-SQL statement raises a run-time error, the entire transaction is terminated and rolled back.

Box 2: COMMIT

Commit the transaction.

Box 3: XACT_STATE

Box 4: ROLLBACK

Rollback the transaction

Box 5: THROW

THROW raises an exception and the severity is set to 16. Requirement: Data modifications that are unsuccessful are rolled back. The exception severity level is set to 16 and a value of -1 is returned.

References:

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

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

QUESTION 53

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.

Drag and Drop Question

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

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

The following table describes the columns in Sales.Orders:

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

The following table describes the columns in Sales.OrderLines:

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

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

Transact-SQL segments

```

RETURNS decimal(18,2)

CREATE FUNCTION Sales.CalculateTaxRate ()

CREATE FUNCTION Sales.CalculateTaxRate (
    @OrderID int
)

RETURN @CalculatedRate
END

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

RETURNS Table
END

AS
BEGIN
declare @CalculatedTaxRate
decimal(18,2)

```

Answer Area



Answer:

Transact-SQL segments

```

RETURNS decimal(18,2)

CREATE FUNCTION Sales.CalculateTaxRate ()

CREATE FUNCTION Sales.CalculateTaxRate (
    @OrderID int
)

RETURN @CalculatedRate
END

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

RETURNS Table
END

AS
BEGIN
declare @CalculatedTaxRate
decimal(18,2)

```

Answer Area

```

CREATE FUNCTION Sales.CalculateTaxRate ()

RETURNS decimal(18,2)

AS
BEGIN
declare @CalculatedTaxRate
(SET @CalculatedTaxRate = (
    SELECT 1 + (MAX(TaxRate)
        / 100)
    FROM Sales.OrderLines
    WHERE OrderID = @OrderID
RETURN @CalculatedRate
END

```



Explanation:

References:

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

QUESTION 54

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

You 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?

- A

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

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

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

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

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

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

- G `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers FOR SYSTEM_TIME
BETWEEN '2014-01-01 00:00:00.000000' AND '2015-01-01 00:00:00.000000'`
- H `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers
WHERE DateCreated
BETWEEN '20140101' AND '20141231'`
- A. Option A
B. Option B
C. Option C
D. Option D
E. Option E
F. Option F
G. Option G
H. Option H

Answer: B

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 55

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

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

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

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

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

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

Answer: G

Explanation:

The following query searches for row versions for Employee row with EmployeeID = 1000 that were active at least for a portion of period between 1st January of 2014 and 1st January 2015 (including the upper boundary):

```
SELECT * FROM Employee  
FOR SYSTEM_TIME
```

BETWEEN '2014-01-01 00:00:00.0000000' AND '2015-01-01 00:00:00.0000000' WHERE EmployeeID = 1000 ORDER BY ValidFrom;

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

QUESTION 56

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

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

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

You are developing a report that displays customer information. The report must contain a grand total column. You need to write a query that returns the data for the report. Which Transact-SQL statement should you run?

- A

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

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

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

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

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

- F `SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
FROM Customers AS c ORDER BY c.CustomerID
FOR XML PATH ('CustomerData'), root ('Customers')`
- G `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers FOR SYSTEM_TIME
BETWEEN '2014-01-01 00:00:00.000000' AND '2015-01-01 00:00:00.000000'`
- H `SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
FROM Customers
WHERE DateCreated
BETWEEN '20140101' AND '20141231'`
- A. Option A
B. Option B
C. Option C
D. Option D
E. Option E
F. Option F
G. Option G
H. Option H

Answer: E

Explanation:

Calculate aggregate column through AVG function and GROUP BY clause.

QUESTION 57

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

You create a table named Customers. Data stored in the table must be exchanged between web pages and web servers by using AJAX calls that use REST endpoint. You need to return all customer information by using a data exchange format that is text- based and lightweight. Which Transact-SQL statement should you run?

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

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

Answer: C

Explanation:

JSON can be used to pass AJAX updates between the client and the server. Export data from SQL Server as JSON, or format query results as JSON, by adding the FOR JSON clause to a SELECT statement. When you use the FOR JSON clause, you can specify the structure of the output explicitly, or let the structure of the SELECT statement determine the output.

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

QUESTION 58

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

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

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

You are developing a report that aggregates customer data only for the year 2014. The report requires that the data be denormalized. You need to return the data for the report. Which Transact-SQL statement should you run?

- A

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

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

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

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

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


- ' F SELECT c.CustomerID, c.FirstName, c.LastName, c.Address, c.ValidFrom, c.ValidTo
 FROM Customers AS c ORDER BY c.CustomerID
 FOR XML PATH ('CustomerData'), root ('Customers')
- ' G SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
 FROM Customers FOR SYSTEM_TIME
 BETWEEN '2014-01-01 00:00:00.000000' AND '2015-01-01 00:00:00.000000'
- ' H SELECT CustomerID, FirstName, LastName, TaxIdNumber, Address, ValidFrom, ValidTo
 FROM Customers
 WHERE DateCreated
 BETWEEN '20140101' AND '20141231'
- A. Option A
B. Option B
C. Option C
D. Option D
E. Option E
F. Option F
G. Option G
H. Option H

Answer: G

QUESTION 59

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

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

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

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

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

Which Transact-SQL statement should you run?

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

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

Answer: F

Explanation:

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

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

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

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

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

QUESTION 60

Your team is developing a database for a new online travel application. You need to design tables and other database objects to support the application. One particular table called Airline_Schedules needs to store the departure and arrival dates and times of flights along with time zone information. What should you do?

- A. Use the CAST function
- B. Use the DATETIMEOFFSET data type
- C. Use a user-defined table type
- D. Use the DATETIME2 data type

Answer: B

Explanation:

Datetimeoffset - defines a date that is combined with a time of a day that has time zone awareness and is based on a 24-hour clock.

QUESTION 61

As part of a new enterprise project, you're designing a new table to store financial transactions. This table could eventually store millions of rows and so storage space is very important. One of the columns in the table will store either a 1 or 0 value. Which data type would be most appropriate?

- A. tinyint
- B. bit
- C. float
- D. numeric

Answer: B

Explanation:

bit is an integer data type that can take a value of 1, 0, or NULL.

QUESTION 62

You're creating a new query that will select rows from a products tables. The query works out the count of products within each category by grouping on the category, filtering by categories that contain more than one product and then sorting the results in category order. In which order should these clauses be used in the query?

- A. GROUP BY, HAVING, ORDER BY
- B. HAVING, GROUP BY, ORDER BY
- C. ORDER BY, GROUP BY, HAVING
- D. GROUP BY, ORDER BY, HAVING

Answer: A

QUESTION 63

You are writing a set of queries against a FILESTREAM-enabled database. You create a stored procedure that will update multiple tables within a transaction. You need to ensure that if the stored procedure raises a run-time error, the entire transaction is terminated and rolled back. Which

Transact-SQL statement should you include at the beginning of the stored procedure?

- A. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
- B. SET XACT_ABORT ON
- C. SET IMPLICIT TRANSACTIONS OFF
- D. SET XACT_ABORT OFF

Answer: B

Explanation:

SET XACT_ABORT - Specifies whether SQL Server automatically rolls back the current transaction when a Transact-SQL statement raises a run-time error.

QUESTION 64

Which index type gives high performance gains for analytic queries that scan large amounts of data, especially on large tables?

- A. Rowstore
- B. Columnstore

Answer: B

Explanation:

Use columnstore indexes on data warehousing and analytics workloads, especially on fact tables, since they tend to require full table scans rather than table seeks.

QUESTION 65

You can use the PIVOT and UNPIVOT relational operators to change a table-valued expression into another table. Which clause rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output, and performs aggregations where they are required on any remaining column values that are wanted in the final output?

- A. PIVOT
- B. UNPIVOT

Answer: A

QUESTION 66

A DML trigger is an action programmed to execute when a data manipulation language (DML) event occurs in the database server. DML events include UPDATE, INSERT, or DELETE statements issued against a table or view. Which of the following is TRUE regarding INSTEAD OF triggers?

- A. All of these.
- B. INSTEAD OF triggers fire in place of the triggering action and before constraints are processed.
- C. If there are AFTER triggers on the table, they will fire after constraint processing.
- D. If the constraints are violated, the AFTER trigger is not executed.

Answer: A

QUESTION 67

Which Transact-SQL clause is described below? Generates totals that appear as additional summary columns at the end of the result set. When used with BY, the ____ clause generates control-breaks and subtotals in the result set.

- A. AVG
- B. COMPUTE
- C. None of these
- D. SUM

Answer: B

QUESTION 68

The transaction isolation level controls the locking and row versioning behavior of Transact-SQL statements issued by a connection to SQL Server. Which of the following transaction isolation levels is the default in SQL Server 2016?

- A. READ COMMITTED
- B. READ UNCOMMITTED
- C. REPEATABLE READ
- D. SERIALIZABLE

Answer: A

QUESTION 69

Study the JSON string shown in the image.

```
DECLARE @jsonInfo VARCHAR(MAX)
SET @jsonInfo = N'{
  "info":{
    "type":1,
    "address":{
      "town":"Bristol",
      "county":"Avon",
      "country":"England"
    },
    "tags":["Sport", "Water polo"]
  },
  "type":"Basic"
}'
```

Which Path parameter would NOT result in an error when used by JSON_VALUE in strict mode?

- A. \$.info."address"
- B. \$.info.address.town
- C. \$
- D. \$.info.type[0]

Answer: B

Explanation:

JSON_VALUE extracts a scalar value from a JSON string. In strict mode it will throw an error if the return result is a nested value/object. If you want to extract a nested value you must use JSON_QUERY instead.

QUESTION 70

As part of a global e-commerce business you are developing a Microsoft SQL Server database that supports the company's online website. The application contains a table that has the following definition:

```
CREATE TABLE Inventory  
(ItemID int NOT NULL PRIMARY KEY,  
ProductsInStore int NOT NULL,  
ProductsInWarehouse int NOT NULL)
```

You need to create a computed column that returns the sum total of the ProductsInStore and ProductsInWarehouse values for each row. Which T-SQL statement should you use?

- A. ALTER TABLE Inventory
ADD ProductsInStore - ProductsInWarehouse = TotalProducts
- B. ALTER TABLE Inventory
ADD TotalProducts AS ProductsInStore + ProductsInWarehouse
- C. ALTER TABLE Inventory
ADD TotalProducts AS SUM(ProductsInStore, ProductsInWarehouse)
- D. ALTER TABLE Inventory
ADD TotalProducts = ProductsInStore + ProductsInWarehouse

Answer: B

QUESTION 71

You are developing a Microsoft SQL Server database. You create a view that performs the following tasks: Joins 10 tables that contain up to 400,000 records each. Performs aggregations on 4 fields. The view is frequently used in several detailed reports. You need to improve the performance of the reports. What should you do?

- A. Convert the view into a table-valued function
- B. Convert the view into an indexed view
- C. Convert the view into a stored procedure and retrieve the result from the stored procedure into a temporary table
- D. Convert the view into a Common Table Expression (CTE)

Answer: B

Explanation:

The first index created on a view must be a unique clustered index. After the unique clustered index has been created, you can create more nonclustered indexes. Creating a unique clustered index on a view improves query performance because the view is stored in the database in the same way a table with a clustered index is stored. The query optimizer may use indexed views to speed up the query execution. The view does not have to be referenced in the query for the optimizer to consider that view for a substitution.

QUESTION 72

Is the following statement TRUE or FALSE? JSON is not a built-in data type in SQL Server 2016, and SQL Server 2016 does not have custom JSON indexes?

- A. TRUE
- B. FALSE

Answer: A

QUESTION 73

The !> operator is a comparison operator that compares two expressions. However, if ANSI_NULLS is set to ON and one of the operands is NULL, what will the result be?

expression !> expression (NULL)

- A. FALSE
- B. 0
- C. TRUE
- D. NULL

Answer: D

Explanation:

If either or both operands are NULL and SET ANSI_NULLS is set to ON, the result is NULL. If SET ANSI_NULLS is set to OFF, the result is FALSE if one of the operands is NULL, and TRUE if both operands are NULL.

QUESTION 74

You are developing a SQL Server database that contains a heap named OrdersArchive. You write the following Transact-SQL query:

```
INSERT INTO OrdersArchive  
SELECT * FROM CompletedOrders
```

You need to optimize transaction logging and locking for the statement. Which table hint should you use?

- A. HOLDLOCK
- B. TABLOCK
- C. XLOCK
- D. ROWLOCK

Answer: B

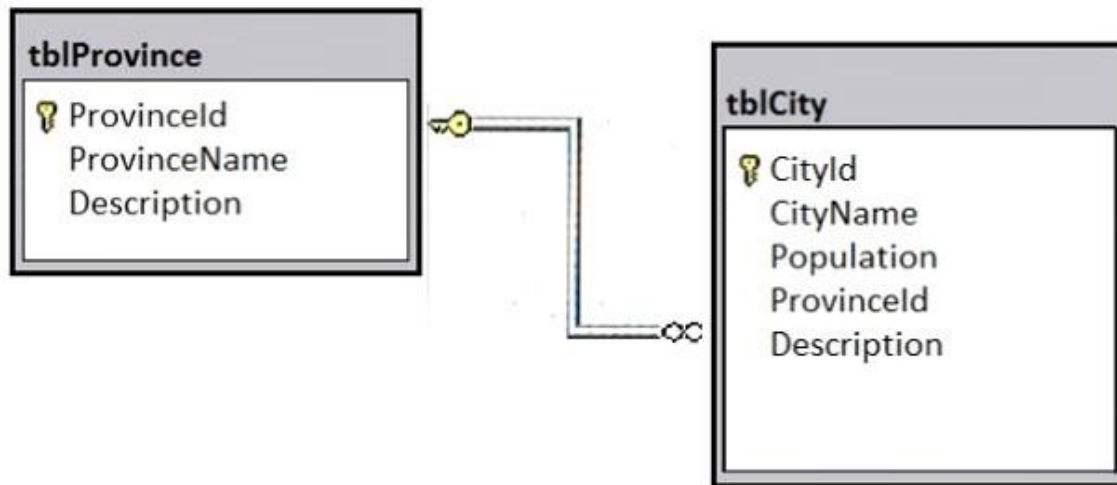
Explanation:

Table hints override the default behavior of the query optimizer for the duration of the data manipulation language (DML) statement by specifying a locking method, one or more indexes, a query-processing operation such as a table scan or index seek, or other options. Table hints are specified in the FROM clause of the DML statement and affect only the table or view referenced in that clause. When importing data into a heap by using the INSERT INTO SELECT FROM statement, you can enable optimized logging and locking for the statement by specifying the TABLOCK hint for the target table. In addition, the recovery model of the database must be set to simple or bulk-logged.

QUESTION 75

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

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

Answer: A

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.

[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 76

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

```
CREATE TABLE Products.Sales (
    SalesId int IDENTITY(1,1) PRIMARY KEY,
    SalesDate DateTime NOT NULL,
    SalesAmount decimal(18,2) NULL
)
```

You add the following data to the table:

<u>SalesID</u>	<u>SalesData</u>	<u>SalesAmount</u>
1	2015-03-05 16:37:23.630	65.00
2	2014-08-25 16:37:23.633	98.00
3	2014-10-15 16:37:23.633	39.00
4	2016-04-06 16:37:23.633	118.00
5	2014-08-29 16:37:23.633	79.00
6	2015-07-17 16:37:23.633	68.00
7	2016-01-03 16:37:23.637	115.00
8	2015-10-23 16:37:23.637	52.00
9	2014-12-07 16:37:23.637	109.00
10	2016-06-15 16:37:23.637	83.00

You are developing a report to display monthly sales data. You need to create a Transact-SQL query to meet the following requirements:

- Retrieve a column for the year followed by a column for each month from January through December.
- Include the total sales amount for each month.
- Aggregate columns by year, month, and then amount.

Construct the query using the following guidelines:

- Use the MONTH keyword as the interval when using the DATENAME function.
- Do not modify the provided IN clause.
- Do not surround object names with square brackets.
- Do not use implicit joins.
- Do not use the DATEPART function.

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

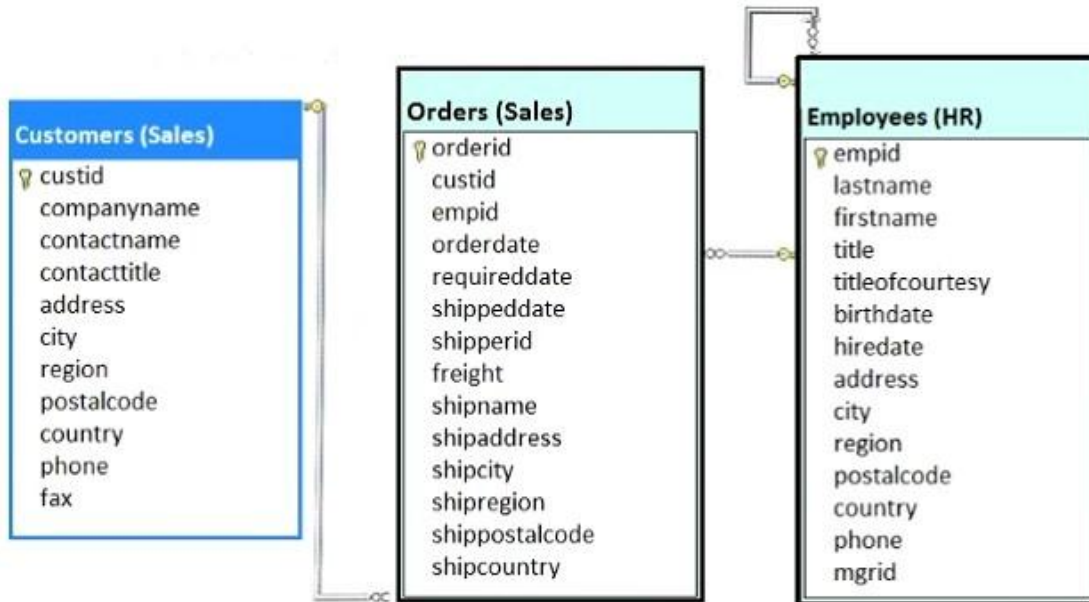
1. SELECT * FROM
2. (SELECT YEAR(SalesData) AS Year, DATENAME (MONTH, SalesDate) AS Month, SalesAmount AS Amount
- 3.
4.) AS MonthlySalesData
- 5.

6. FOR Month IN (January, February, March, April, May, June, July, August, September, October, November, December))
AS MonthNamePivot
.....

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 have a database that includes the tables shown in the exhibit:



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

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

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

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

Does the solution meet the goal?

A. Yes

B. No

Answer: B

Explanation:

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

QUESTION 78

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

Answer: H

Explanation:

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

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

QUESTION 79

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 tablevalue must be globally uniquevalue must be automatically generated for INSERTs operations
Reason	<ul style="list-style-type: none">stores reasons for the paymentsupports multilingual valuessupports values with 1 to 500 characters
Amount	<ul style="list-style-type: none">stores monetary valuesmust not produce rounding errors with calculations

Which Transact-SQL statement should you run?

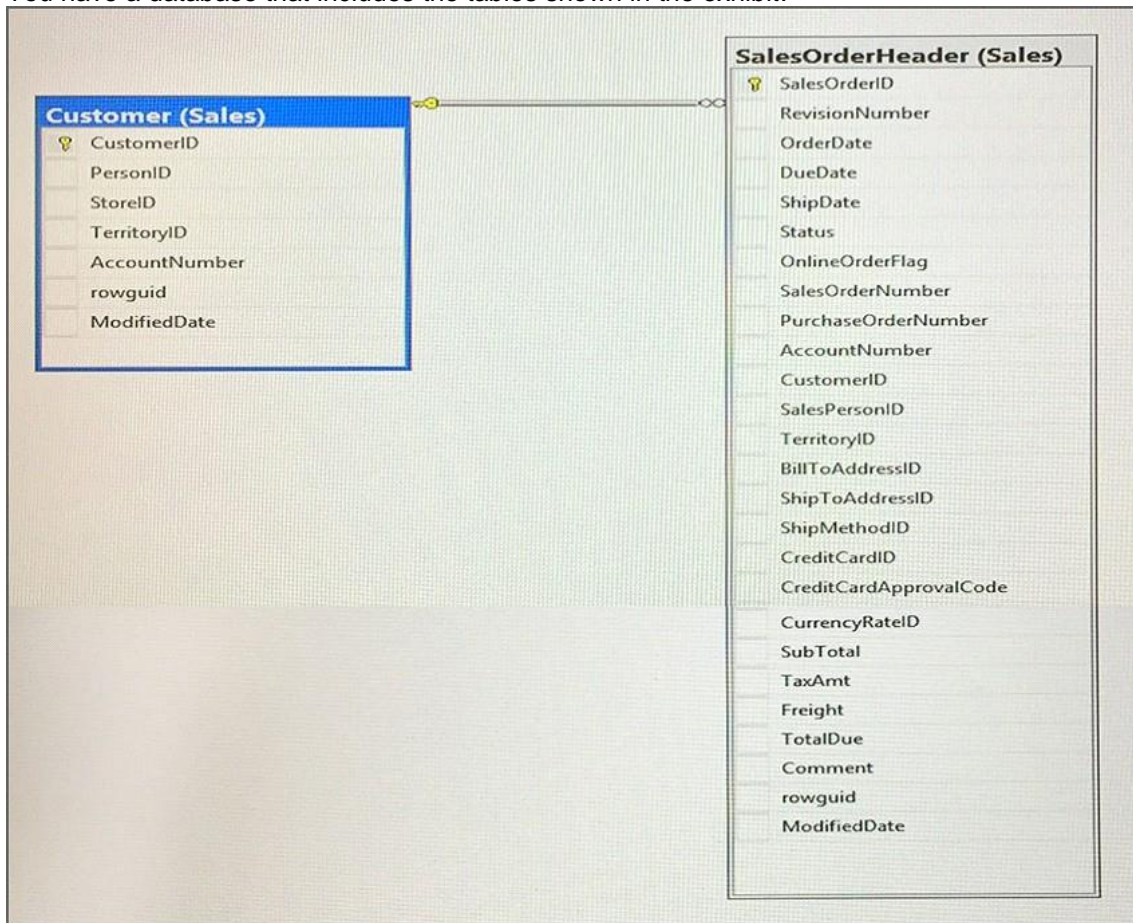
- A. `CREATE TABLE MiscellaneousPayment (Id uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY, Reason varchar(500), Amount money)`
- B. `CREATE TABLE MiscellaneousPayment (Id int identify(1,1) PRIMARY KEY, Reason nvarchar(500), Amount numeric(19,4))`

- C. CREATE TABLE MiscellaneousPayment (Id uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY, Reason varchar(500), Amount decimal(19,4))
- D. CREATE TABLE MiscellaneousPayment (Id uniqueidentifier DEFAULT NEWID() PRIMARY KEY, Reason nvarchar(500), Amount decimal(19,4))

Answer: D

QUESTION 80

You have a database that includes the tables shown in the exhibit:



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

Answer: A

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

QUESTION 81

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

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

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

You need to determine the total number of customers who have only loan accounts. Which Transact-SQL statement should you run?

- A. `SELECT COUNT(*)FROM (SELECT AcctNoFROM tblDepositAcctINTERSECTSELECT AcctNoFROM tblLoanAcct) R`
- B. `SELECT COUNT(*)FROM (SELECT CustNoFROM tblDepositAcctUNIONSELECT CustNoFROM tblLoanAcct) R`
- C. `SELECT COUNT(*)FROM (SELECT CustNoFROM tblDepositAcctUNION ALLSELECT CustNoFROM tblLoanAcct) R`
- D. `SELECT COUNT (DISTINCT D.CustNo)FROM tblDepositAcct D, tblLoanAcct LWHERE D.CustNo = L.CustNo`
- E. `SELECT COUNT(DISTINCT L.CustNo)FROM tblDepositAcct DRIGHT JOIN tblLoanAcct L ON D.CustNo = L.CustNoWHERE D.CustNo IS NULL`
- F. `SELECT COUNT(*)FROM (SELECT CustNoFROM tblDepositAcctEXCEPTSELECT CustNoFROM tblLoanAcct) R`
- G. `SELECT COUNT (DISTINCT COALESCE(D.CustNo, L.CustNo))FROM tblDepositAcct DFULL JOIN tblLoanAcct L ON D.CustNo = L.CustNoWHERE D.CustNo IS NULL OR L.CustNo IS NULL`
- H. `SELECT COUNT(*)FROM tblDepositAcct DFULL JOIN tblLoanAcct L ON D.CustNo = L.CustNo`

Answer: E

Explanation:

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

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

QUESTION 82

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

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

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

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

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

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

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

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

- A
- ```
DECLARE @sdate DATETIME, @edate DATETIME
SET @sdate = DATEFROMPARTS (2013, 12, 31)
set @edate = DATEADD(d, 1, @sdate)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME ALL
WHERE ValidFrom > @sdate AND ValidTo < @edate
```
- B
- ```
DECLARE @sdate DATETIME, @edate DATETIME
SET @sdate = DATEFROMPARTS (2013, 12, 31)
set @edate = DATEADD(d, -1, @sdate)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME BETWEEN @sdate AND @edate
```
- C
- ```
DECLARE @date DATE
SET @date = DATEFROMPARTS (2013, 12, 31)
SELECT * FROM Sales.Customers FOR SYSTEM_TIME AS OF @date
```
- D
- ```
DECLARE @date DATE
SET @date = DATEFROMPARTS (2013, 12, 31)
SELECT * FROM Sales.Customers WHERE @date BETWEEN ValidFrom AND ValidTo
```

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

Answer: D

Explanation:

The datetime 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.

QUESTION 83

.....

QUESTION 125

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

- accepts a product identifies as input
- calculates the total quantity of a specific product, including quantity on hand and quantity on - order
- caches and reuses execution plan
- returns a value
- can be called from within a SELECT statement
- can be used in a JOIN clause

What should you create?

- A. a temporary table that has a columnstore index
B. a user-defined table-valued function
C. a memory-optimized table that has updated statistics
D. a natively-compiled stored procedure that has an OUTPUT parameter

Answer: B

Explanation:

A table-valued user-defined function can also replace stored procedures that return a single result set. The table returned by a user-defined function can be referenced in the FROM clause of a Transact-SQL statement, but stored procedures that return result sets cannot.

[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 126

Drag and Drop

You have a database containing the following tables:

Servers

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

Errors

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

You have a user-defined, scalar function named IPLookup that takes a DNS name as a parameter and returns the IP address of the server. You have an additional user-defined, scalar function named DNSLookup, that takes an IP address as a parameter and returns a DNS name. You create a view named vwErrors by running the following Transact-SQL statement:

```
CREATE VIEW vwErrors
AS
SELECT ErrorID, IPLookup(DNS) as IP, LogMessage
FROM Errors
INNER JOIN Servers ON Errors.ServerID = Servers.ServerID
```

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

Transact-SQL segments

WITH APPEND
AFTER INSERT
INSTEAD OF INSERT
FROM inserted
FROM vwErrors
dbo.DNSLookup(IP)
Servers.IP

Answer Area

```
CREATE TRIGGER newErrorTrg on vwErrors
AS
BEGIN
    INSERT INTO Errors
        SELECT ErrorID, Servers.ServerID, LogMessage
        FROM
            INNER JOIN Servers on Servers.DNS =
END
```

Answer:

Transact-SQL segments	Answer Area
WITH APPEND	CREATE TRIGGER newErrorTrg on vwErrors
AFTER INSERT	INSTEAD OF INSERT
	AS
	BEGIN
	INSERT INTO Errors
	SELECT ErrorID, Servers.ServerID, LogMessage
FROM vwErrors	FROM inserted
	INNER JOIN Servers on Servers.DNS = dbo.DNSLookup(IP)
Servers.IP	END

Explanation:

<https://docs.microsoft.com/en-us/sql/t-sql/queries/output-clause-transact-sql>

QUESTION 127

Hotspot

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

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

The Task table includes the following columns:

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

Users report performance issues when they run the following query:

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

You need to improve query performance and limit results to projects that specify an end date. How should you complete the Transact-SQL statement? (To answer, select the appropriate Transact-SQL segments in the answer area.)

Answer Area

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

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

 LIKE 

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


    AND 

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


) AS R
```

Answer:

Answer Area

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

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

 LIKE 

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


    AND 

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


) AS R
```

Explanation:

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

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

QUESTION 128

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.

Answer: B

Explanation:

Incorrect:

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

Not C: Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.

[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 129

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

Answer: B

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.

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

QUESTION 130

Hotspot

You need to develop a Transact-SQL statement that meets the following requirements:

- The statement must return a custom error when there are problems updating a table.
- The error number must be the value 50555.
- The error severity level must be 14.
- A Microsoft SQL Server alert must be triggered when the error condition occurs.

Which Transact-SQL segment should you use for each requirement? (To answer, select the appropriate Transact-SQL segments in the answer area.)

Answer Area

Requirement

Check for error condition

Transact-SQL segment

```
BEGIN TRANSACTION...END TRANSACTIONS
TRY_PARSE
BEGIN...END
BEGIN CATCH...END CATCH
```

Custom error implementation

```
THROW 50555, 'The update failed.', 1
RAISERROR (50555,14,1 'The update failed.') WITH LOG
RAISERROR (50555,14,1 'The update failed.') WITH NOWAIT
RAISERROR (50555, 'The update failed.')
```

Answer:

Answer Area

Requirement

Check for error condition

Transact-SQL segment

```
BEGIN TRANSACTION...END TRANSACTIONS
TRY_PARSE
BEGIN...END
BEGIN CATCH...END CATCH
```

Custom error implementation

```
THROW 50555, 'The update failed.', 1
RAISERROR (50555,14,1 'The update failed.') WITH LOG
RAISERROR (50555,14,1 'The update failed.') WITH NOWAIT
RAISERROR (50555, 'The update failed.')
```

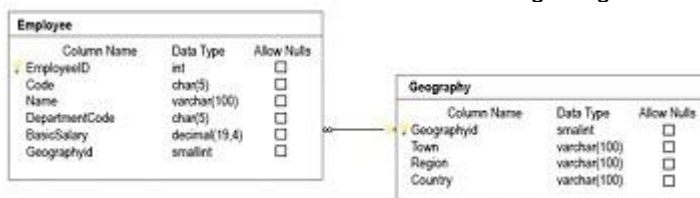
Explanation:

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

QUESTION 131

Hotspot

You have two tables as shown in the following image:



You need to analyze the following query. (Line numbers are included for reference only.)

```

01 DECLARE @DepartmentCode nchar(5) = N'DEP01'
02 DECLARE @RoundedUpSalary int
03 DECLARE @EmployeeName nvarchar(100)
04 SELECT
05     Name,
06     CONVERT(int, Code) EmployeeCode,
07     BasicSalary
08 FROM dbo.Employee e
09 INNER JOIN dbo.Geography g
10 ON e.GeographyId = g.GeographyId
11 WHERE DepartmentCode = @DepartmentCode

```

Use the drop-down menus to select the answer choice that completes each statement based on the information presented in the graphic.

Answer Area

Statements

An implicit conversion exists at [answer choice].

Answer choices

	▼
line number 6	
line number 10	
line number 11	

An explicit conversion exists at [answer choice].

	▼
line number 6	
line number 10	
line number 11	

Answer:

Answer Area

Statements

An implicit conversion exists at [answer choice].

Answer choices

	▼
line number 6	
line number 10	
line number 11	

An explicit conversion exists at [answer choice].

	▼
line number 6	
line number 10	
line number 11	

Explanation:

To compare char(5) and nchar(5) an implicit conversion has to take place. Explicit conversions use the CAST or CONVERT functions, as in line number 6.

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-type-conversion-database-engine#implicit-and-explicit-conversion>

QUESTION 132

Drag and Drop

You have a database containing the following tables:

Servers

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

Errors

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

You have a user-defined, scalar function named IPLookup that takes a DNS name as a parameter and returns the IP address of the server. You have an additional user-defined, scalar function named DNSLookup, that takes an IP address as a parameter and returns a DNS name. You create a view named vwErrors by running the following Transact-SQL statement:

```
CREATE VIEW vwErrors
AS
    SELECT ErrorID, IPLookup(DNS) as IP, LogMessage
    FROM Errors
    INNER JOIN Servers ON Errors.ServerID = Servers.ServerID
```

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

Transact-SQL segments

WITH APPEND
AFTER INSERT
INSTEAD OF INSERT
FROM inserted
FROM vwErrors
dbo.DNSLookup(IP)
Servers.IP

Answer Area

```
CREATE TRIGGER newErrorTrg on vwErrors
AS
BEGIN
    INSERT INTO Errors
        SELECT ErrorID, Servers.ServerID, LogMessage
        FROM vwErrors
        INNER JOIN Servers on Servers.DNS = 
END
```

Answer:

Transact-SQL segments	Answer Area
WITH APPEND	CREATE TRIGGER newErrorTrg on vwErrors
AFTER INSERT	INSTEAD OF INSERT
	AS
	BEGIN
	INSERT INTO Errors
	SELECT ErrorID, Servers.ServerID, LogMessage
FROM vwErrors	FROM inserted
	INNER JOIN Servers on Servers.DNS = dbo.DNSLookup(IP)
Servers.IP	END

Explanation:

<https://docs.microsoft.com/en-us/sql/t-sql/queries/output-clause-transact-sql>

QUESTION 133

.....

Get Complete Version Exam 70-761 Dumps with VCE and PDF Here



<https://www.passleader.com/70-761.html>