70-761$^{Q\&As}$

Querying Data with Transact-SQL

Pass Microsoft 70-761 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/70-761.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Microsoft Official Exam Center

QUESTION 1

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.

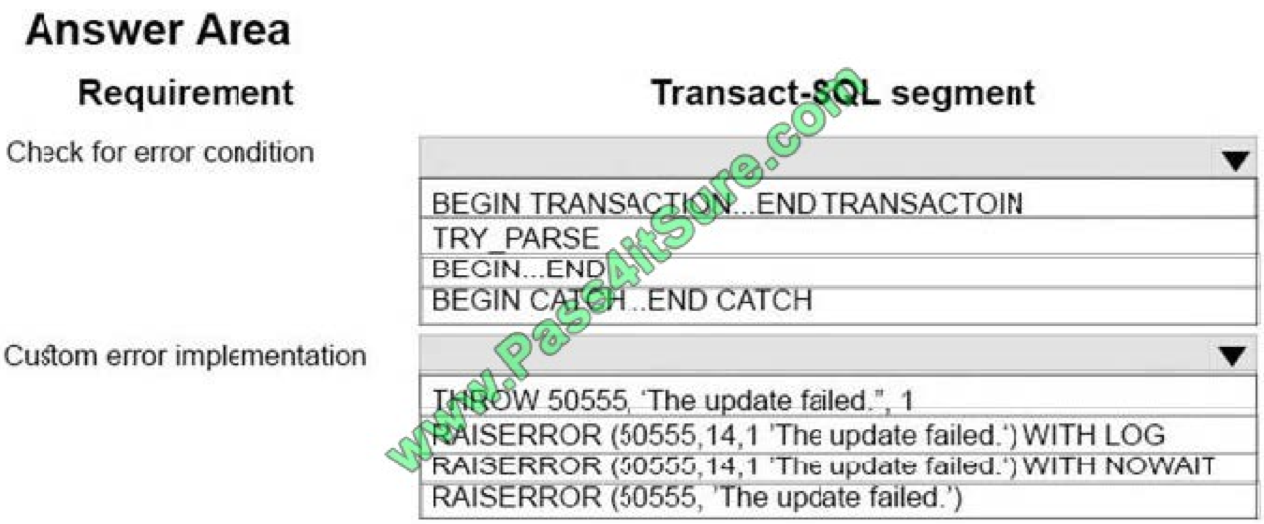
The error number must be the value 50555.

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.

Hot Area:



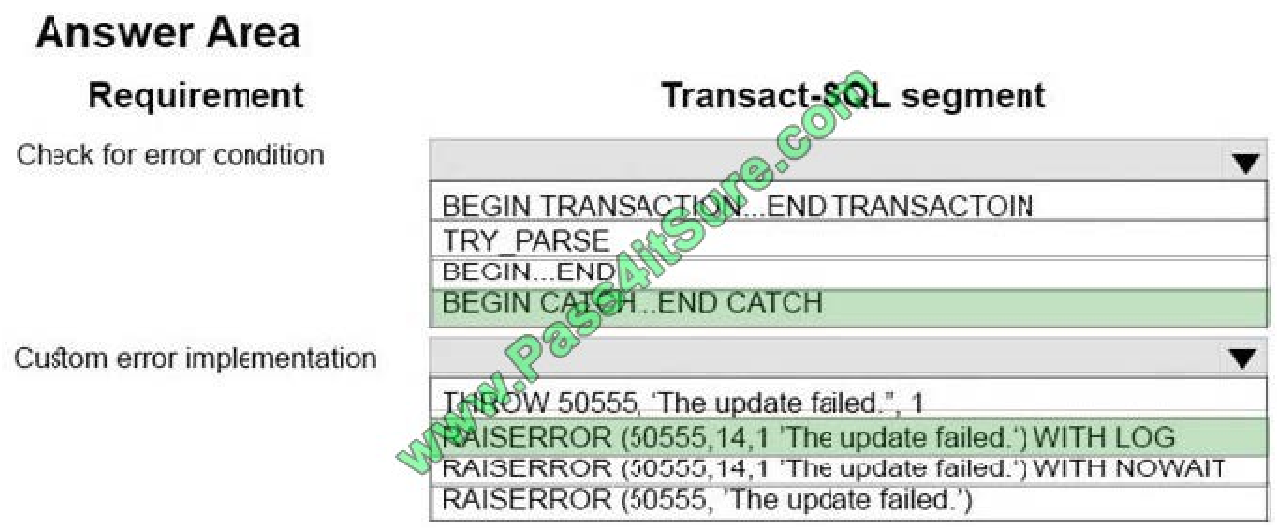
Answer Area

Requirement	Transact-SQL segment

Check for error condition	BEGIN TRANSACTION...END TRANSACTOIN
	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.')

Correct Answer:



Answer Area

Requirement	Transact-SQL segment

Check for error condition	BEGIN TRANSACTION...END TRANSACTOIN
	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.')

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.

Note: RAISERROR syntax:

RAISERROR({ msg_id | msg_str | @local_variable }

{ ,severity ,state }

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

[WITH option [,...n]]

The LOG option logs the error in the error log and the application log for the instance of the Microsoft SQL Server Database Engine.

References:

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

QUESTION 2

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

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

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

Data being processed must contain statistics.

What should you do?

A. Update Table1 by using a common table expression (CTE).

B. Insert the data into a temporary table, and then update Table1 from the temporary table.

C. Place the SELECT statement in a derived table, and then update Table1 by using a JOIN to the derived table.

D. Insert the data into a table variable, and then update Table1 from the table variable.

Correct Answer: B

Temp Tables... Are real materialized tables that exist in tempdb Have dedicated stats generated by the engine Can be indexed Can have constraints Persist for the life of the current CONNECTION Can be referenced by other queries or subproce Incorrect Answers:

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

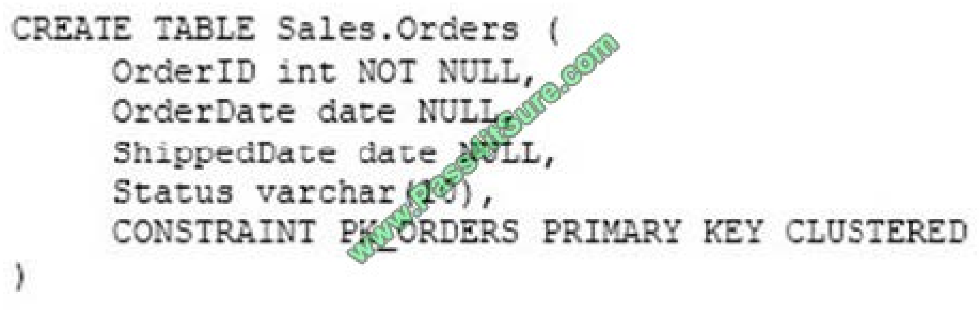
C: Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.

References: https://technet.microsoft.com/en-us/library/ms190766(v=sql.105).aspx
https://dba.stackexchange.com/questions/13112/whats-the-difference-between-a-cte-and-a-temp-table

QUESTION 3

SIMULATION

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:

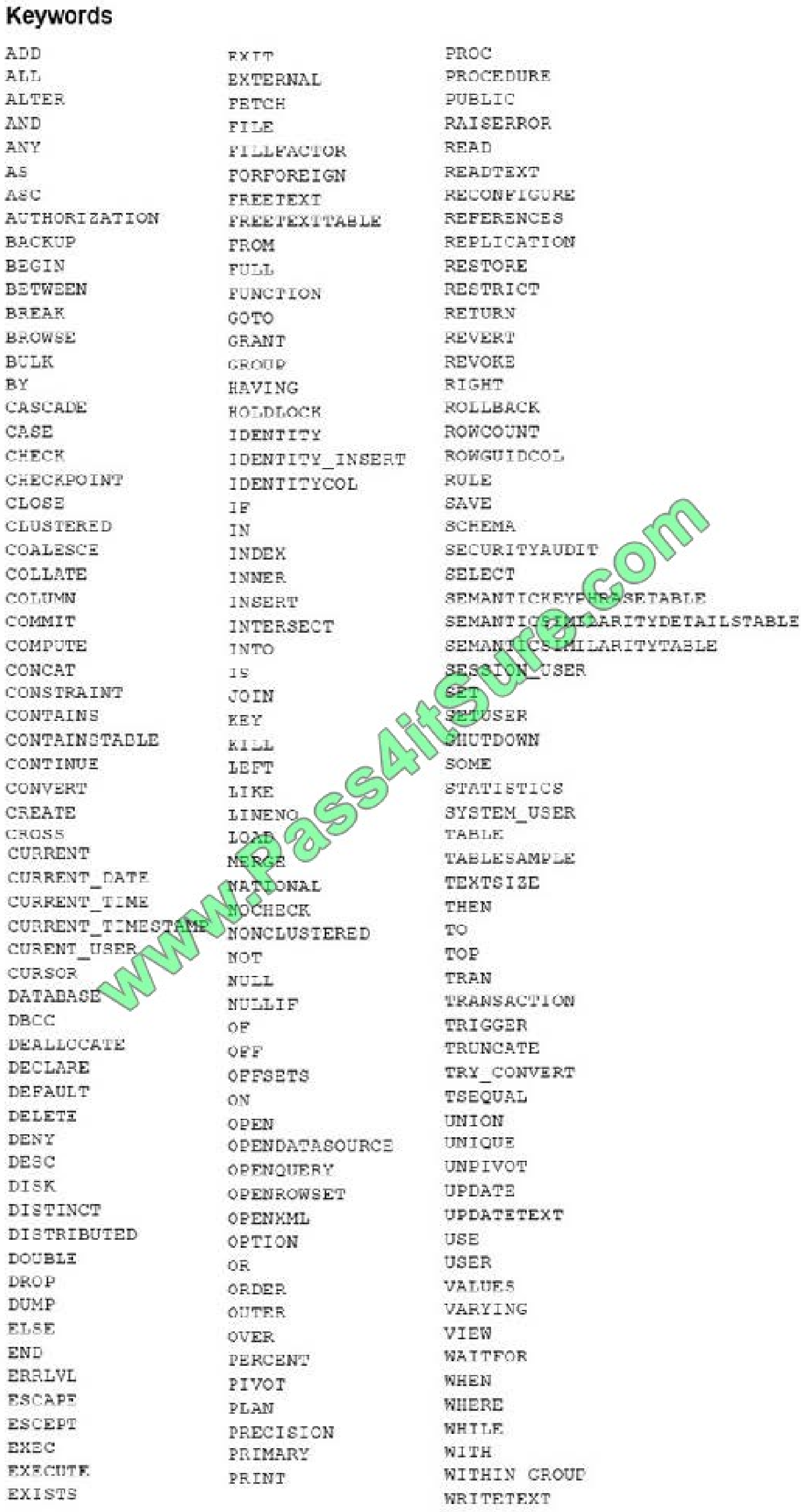
use one-part column names and two-part table names

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



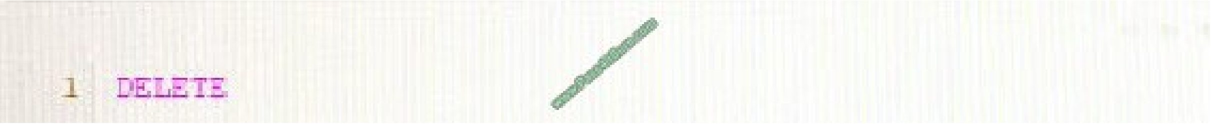
Keywords

ADD
ALL
ALTER
AND
ANY
AS
ASC
AUTHORIZATION
BACKUP
BEGIN
BETWEEN
BREAK
BROWSE
BULK
BY
CASCADE
CASE
CHECK
CHECKPOINT
CLOSE
CLUSTERED
COALESCE
COLLATE
COLUMN
COMMIT
COMPUTE
CONCAT
CONSTRAINT
CONTAINS
CONTAINSTABLE
CONTINUE
CONVERT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURENT_USER
CURSOR
DATABASE
DBCC
DEALLOCATE
DECLARE
DEFAULT
DELETE
DENY
DESC
DISK
DISTINCT
DISTRIBUTED
DOUBLE
DROP
DUMP
ELSE
END
ERRLVL
ESCAPE
ESCEPT
EXEC
EXECUTE
EXISTS

EXIT
EXTERNAL
FETCH
FILE
FILLFACTOR
FORFOREIGN
FREETEXT
FREETEXTTABLE
FROM
FULL
FUNCTION
GOTO
GRANT
GROUP
HAVING
HOLDLOCK
IDENTITY
IDENTITY_INSERT
IDENTITYCOL
IF
IN
INDEX
INNER
INSERT
INTERSECT
INTO
IS
JOIN
KEY
KILL
LEFT
LIKE
LINENO
LOAD
MERGE
NATIONAL
NOCHECK
NONCLUSTERED
NOT
NULL
NULLIF
OF
OFF
OFFSETS
ON
OPEN
OPENDATASOURCE
OPENQUERY
OPENROWSET
OPENXML
OPTION
OR
ORDER
OUTER
OVER
PERCENT
PIVOT
PLAN
PRECISION
PRIMARY
PRINT

PROC
PROCEDURE
PUBLIC
RAISERROR
READ
READTEXT
RECONFIGURE
REFERENCES
REPLICATION
RESTORE
RESTRICT
RETURN
REVERT
REVOKE
RIGHT
ROLLBACK
ROWCOUNT
ROWGUIDCOL
RULE
SAVE
SCHEMA
SECURITYAUDIT
SELECT
SEMANTICKEYPHRASETABLE
SEMANTICSIMILARITYDETAILSTABLE
SEMANTICSIMILARITYTABLE
SESSION_USER
SET
SETUSER
SHUTDOWN
SOME
STATISTICS
SYSTEM_USER
TABLE
TABLESAMPLE
TEXTSIZE
THEN
TO
TOP
TRAN
TRANSACTION
TRIGGER
TRUNCATE
TRY_CONVERT
TSEQUAL
UNION
UNIQUE
UNPIVOT
UPDATE
UPDATETEXT
USE
USER
VALUES
VARYING
VIEW
WAITFOR
WHEN
WHERE
WHILE
WITH
WITHIN GROUP
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.



```
DELETE
```

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

Correct Answer: See the solution below

DELETE FROM Sales.Orders WHERE OrderDate

References: https://msdn.microsoft.com/en-us/library/ms189835.aspx https://msdn.microsoft.com/en-us/library/bb630352.aspx

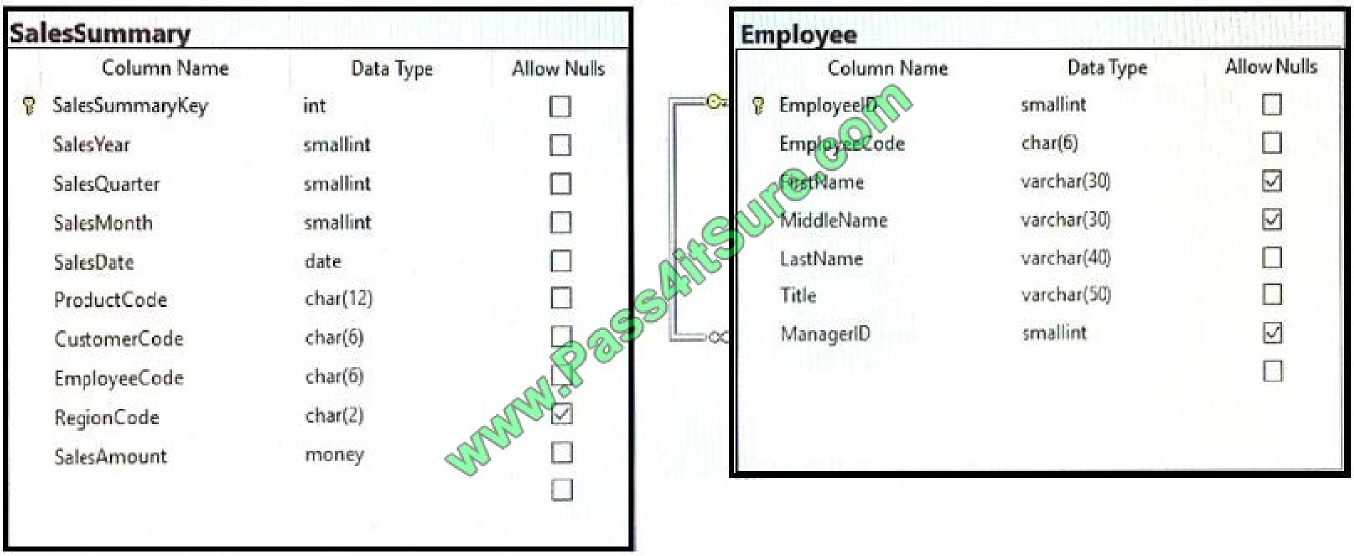
QUESTION 4

DRAG DROP

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

Start of repeated scenario

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



SalesSummary

	Column Name	Data Type	Allow Nulls
🔑	SalesSummaryKey	int	□
	SalesYear	smallint	□
	SalesQuarter	smallint	□
	SalesMonth	smallint	□
	SalesDate	date	□
	ProductCode	char(12)	□
	CustomerCode	char(6)	□
	EmployeeCode	char(6)	□
	RegionCode	char(2)	☑
	SalesAmount	money	□
			□

Employee

	Column Name	Data Type	Allow Nulls
🔑	EmployeeID	smallint	□
	EmployeeCode	char(6)	□
	FirstName	varchar(30)	☑
	MiddleName	varchar(30)	☑
	LastName	varchar(40)	□
	Title	varchar(50)	□
	ManagerID	smallint	☑
			□

You review the Employee table and make the following observations:

Every record has a value in the ManagerID except for the Chief Executive Officer (CEO).

The FirstName and MiddleName columns contain null values for some records.

The valid values for the Title column are Sales Representative manager, and CEO.

You review the SalesSummary table and make the following observations:

The ProductCode column contains two parts: The first five digits represent a product code, and the last seven digits represent the unit price. The unit price uses the following pattern: ####.##.

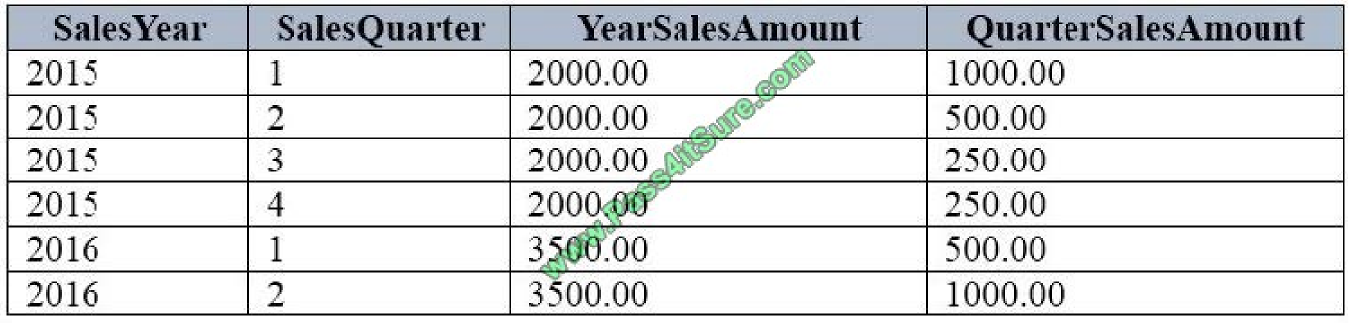
You observe that for many records, the unit price portion of the ProductCode column contains values.

The RegionCode column contains NULL for some records.

Sales data is only recorded for sales representatives.

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

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



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

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

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

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

be joinable with the SELECT statement that supplies data for the report

can be used multiple times with the SELECT statement for the report

be usable only with the SELECT statement for the report

not be saved as a permanent object

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

be joinable with the SELECT statement that supplies data for the report

can be used multiple times for this report and other reports

accept parameters

be saved as a permanent object

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

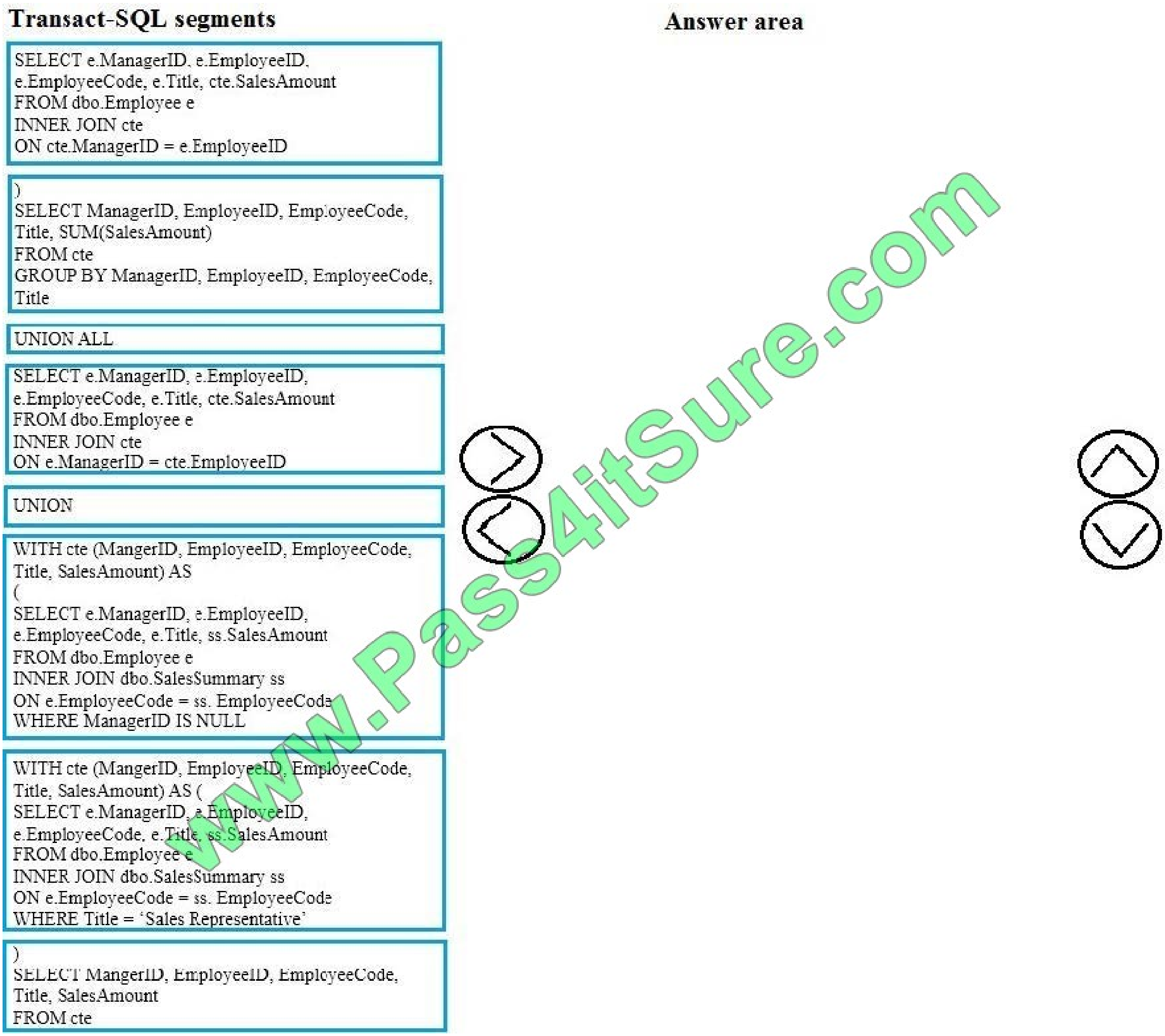
Current Price Stored Procedure: This stored procedure must return the unit price for a product when a product code is supplied. The unit price must include a dollar sign at the beginning. In addition, the unit price must contain a comma every three digits to the left of the decimal point, and must display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

End of Repeated Scenario

You need to create the query for the Sales Managers report.

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.

Select and Place:



```
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, cte.SalesAmount
FROM dbo.Employee e
INNER JOIN cte
ON cte.ManagerID = e.EmployeeID
```

```
)
SELECT ManagerID, EmployeeID, EmployeeCode,
Title, SUM(SalesAmount)
FROM cte
GROUP BY ManagerID, EmployeeID, EmployeeCode,
Title
```

```
UNION ALL
```

```
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, cte.SalesAmount
FROM dbo.Employee e
INNER JOIN cte
ON e.ManagerID = cte.EmployeeID
```

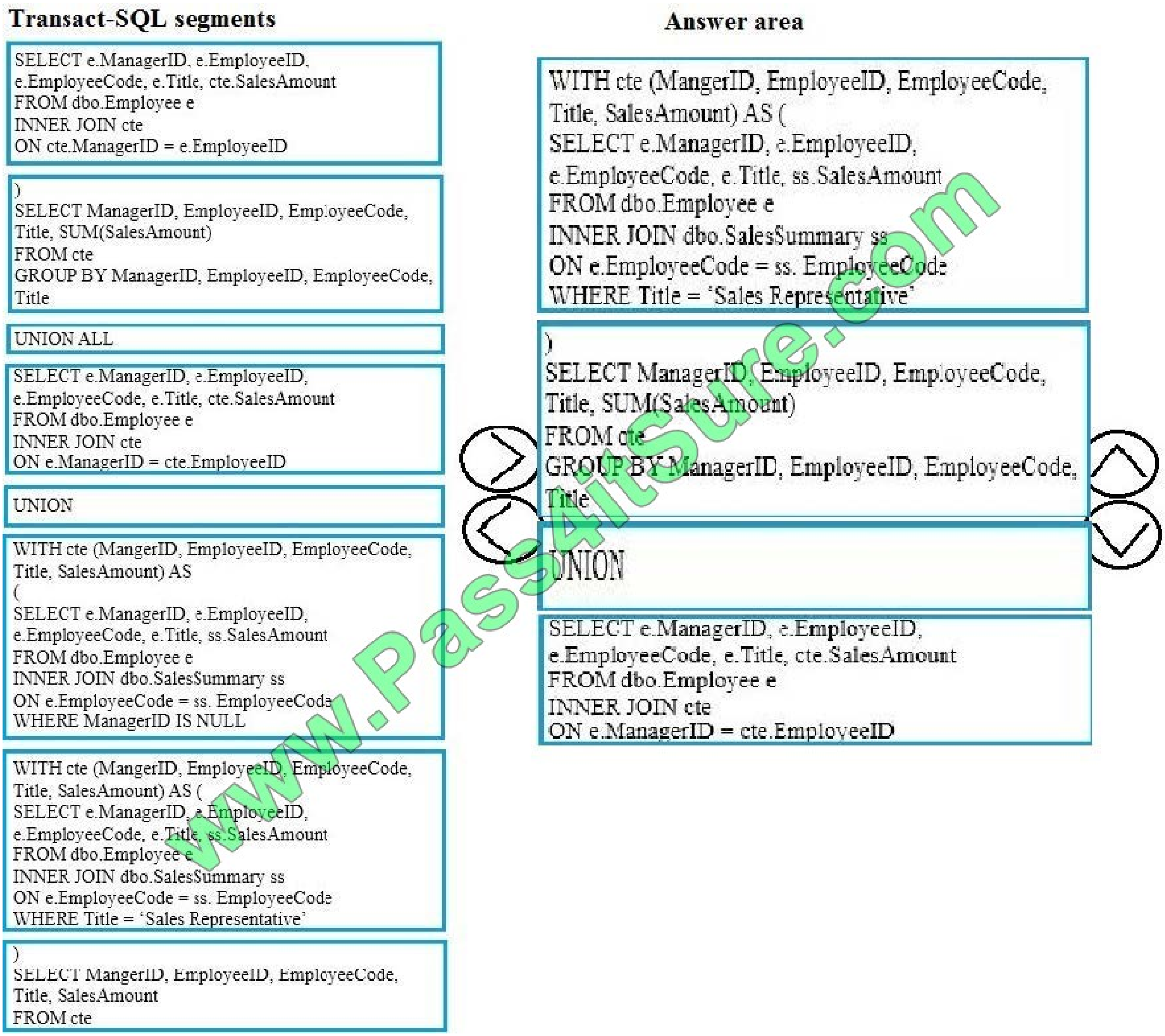
```
UNION
```

```
WITH cte (MangerID, EmployeeID, EmployeeCode,
Title, SalesAmount) AS
(
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, ss.SalesAmount
FROM dbo.Employee e
INNER JOIN dbo.SalesSummary ss
ON e.EmployeeCode = ss. EmployeeCode
WHERE ManagerID IS NULL
```

```
WITH cte (MangerID, EmployeeID, EmployeeCode,
Title, SalesAmount) AS (
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, ss.SalesAmount
FROM dbo.Employee e
INNER JOIN dbo.SalesSummary ss
ON e.EmployeeCode = ss. EmployeeCode
WHERE Title = 'Sales Representative'
```

```
)
SELECT MangerID, EmployeeID, EmployeeCode,
Title, SalesAmount
FROM cte
```

Correct Answer:



Transact-SQL segments

```
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, cte.SalesAmount
FROM dbo.Employee e
INNER JOIN cte
ON cte.ManagerID = e.EmployeeID
```

```
)
SELECT ManagerID, EmployeeID, EmployeeCode,
Title, SUM(SalesAmount)
FROM cte
GROUP BY ManagerID, EmployeeID, EmployeeCode,
Title
```

```
UNION ALL
```

```
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, cte.SalesAmount
FROM dbo.Employee e
INNER JOIN cte
ON e.ManagerID = cte.EmployeeID
```

```
UNION
```

```
WITH cte (MangerID, EmployeeID, EmployeeCode,
Title, SalesAmount) AS
(
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, ss.SalesAmount
FROM dbo.Employee e
INNER JOIN dbo.SalesSummary ss
ON e.EmployeeCode = ss. EmployeeCode
WHERE ManagerID IS NULL
```

```
WITH cte (MangerID, EmployeeID, EmployeeCode,
Title, SalesAmount) AS (
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, ss.SalesAmount
FROM dbo.Employee e
INNER JOIN dbo.SalesSummary ss
ON e.EmployeeCode = ss. EmployeeCode
WHERE Title = 'Sales Representative'
```

```
)
SELECT MangerID, EmployeeID, EmployeeCode,
Title, SalesAmount
FROM cte
```

Answer area

```
WITH cte (MangerID, EmployeeID, EmployeeCode,
Title, SalesAmount) AS (
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, ss.SalesAmount
FROM dbo.Employee e
INNER JOIN dbo.SalesSummary ss
ON e.EmployeeCode = ss. EmployeeCode
WHERE Title = 'Sales Representative'
```

```
)
SELECT ManagerID, EmployeeID, EmployeeCode,
Title, SUM(SalesAmount)
FROM cte
GROUP BY ManagerID, EmployeeID, EmployeeCode,
Title
```

```
UNION
```

```
SELECT e.ManagerID, e.EmployeeID,
e.EmployeeCode, e.Title, cte.SalesAmount
FROM dbo.Employee e
INNER JOIN cte
ON e.ManagerID = cte.EmployeeID
```

From scenario: Sales Manager report: This report lists each sales manager and the total sales amount for all employees that report to the sales manager. Box 1:..WHERE Title=\\'Sales representative\\'

The valid values for the Title column are Sales Representative manager, and CEO.

First we define the CTE expression.

Note: 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.

Box 2:

Use the CTE expression one time.

Box 3: UNION

Box 4:

Use the CTE expression a second time.

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

QUESTION 5

HOTSPOT

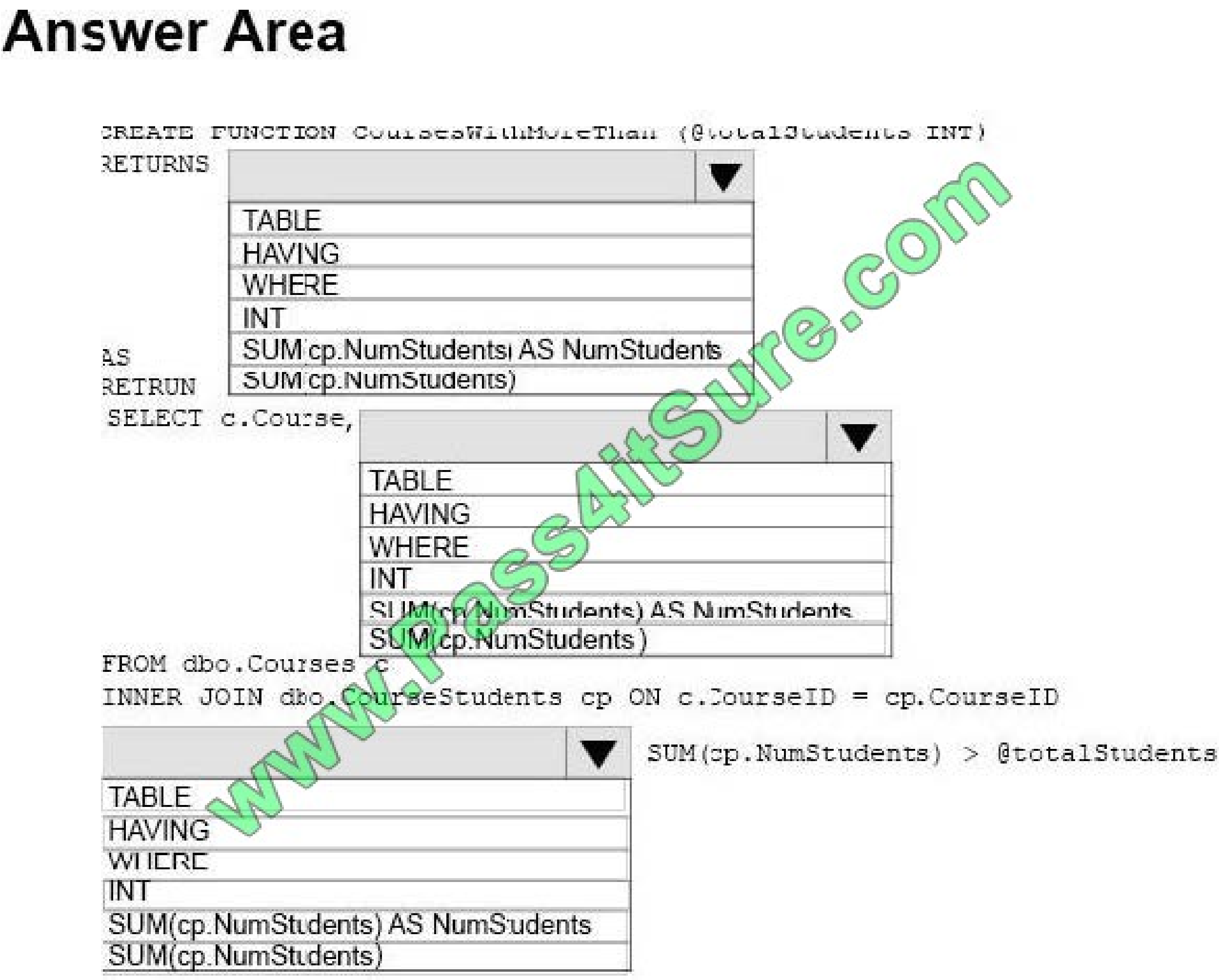
You need to develop a function that returns a list of courses grouped by the total number of students in a course.

The function must list only courses that have more than a specific number of students. The specific number of students is defined as an input variable for the function.

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

NOTE: Each correct selection is worth one point.

Hot Area:

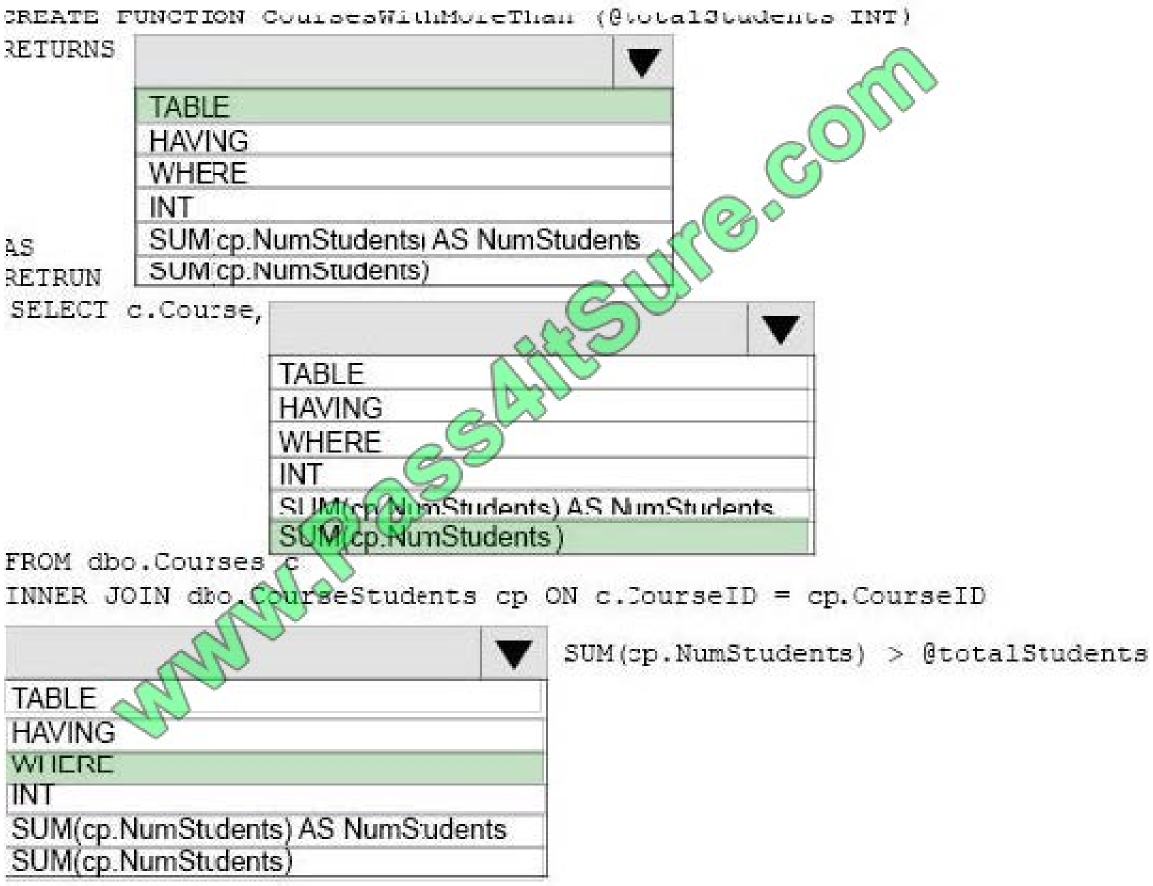


Answer Area

```
CREATE FUNCTION CoursesWithMoreThan (@totalStudents INT)
RETURNS
AS
RETRUN
 SELECT c.Course,
FROM dbo.Courses c
INNER JOIN dbo.CourseStudents cp ON c.CourseID = cp.CourseID
                                SUM(cp.NumStudents) > @totalStudents
```

Correct Answer:

Answer Area



```
CREATE FUNCTION CoursesWithMoreThan (@totalStudents INT)
RETURNS
```

- [✓ TABLE]
- HAVING
- WHERE
- INT
- SUM(cp.NumStudents) AS NumStudents
- SUM(cp.NumStudents)

```
AS
RETRUN
 SELECT c.Course,
```

- TABLE
- HAVING
- WHERE
- INT
- SUM(cp.NumStudents) AS NumStudents
- [✓ SUM(cp.NumStudents)]

```
FROM dbo.Courses c
INNER JOIN dbo.CourseStudents cp ON c.CourseID = cp.CourseID
```

- TABLE
- HAVING
- [✓ WHERE]
- INT
- SUM(cp.NumStudents) AS NumSudents
- SUM(cp.NumStudents)

```
SUM(cp.NumStudents) > @totalStudents
```

To Read the Whole Q&As, please purchase the Complete Version from Our website.

Try our product !

100% Guaranteed Success
100% Money Back Guarantee
365 Days Free Update
Instant Download After Purchase
24x7 Customer Support
Average 99.9% Success Rate
More than 800,000 Satisfied Customers Worldwide
Multi-Platform capabilities - Windows, Mac, Android, iPhone, iPod, iPad, Kindle

We provide exam PDF and VCE of Cisco, Microsoft, IBM, CompTIA, Oracle and other IT Certifications.
You can view Vendor list of All Certification Exams offered:

https://www.pass4itsure.com/allproducts

Need Help

Please provide as much detail as possible so we can best assist you.
To update a previously submitted ticket:

One Year Free Update

Free update is available within One Year after your purchase. After One Year, you will get 50% discounts for updating. And we are proud to boast a 24/7 efficient Customer Support system via Email.

Money Back Guarantee

To ensure that you are spending on quality products, we provide 100% money back guarantee for 30 days from the date of purchase.

Security & Privacy

We respect customer privacy. We use McAfee's security service to provide you with utmost security for your personal information & peace of mind.

Any charges made through this site will appear as Global Simulators Limited.