

- **Vendor: Microsoft**
- **Exam Code: 70-761**
- **Exam Name: Querying Data with Transact-SQL**
- **Question 41 – Question 50**

[Visit PassLeader and Download Full Version 70-761 Exam Dumps](#)

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	SELECT Transact-SQL segment , errs.LogMessage
errs.ServerID	FROM Servers AS svr
INNER JOIN	Transact-SQL segment
CROSS APPLY	(
WITHIN GROUP	SELECT TOP 3 LogMessage
WHERE ServerID = svr.ServerID	FROM Errors
WHERE ServerID = errs.ErrorID	Transact-SQL segment
	ORDER BY Occurrences
) AS errs

Answer:

Transact-SQL segments	Answer Area
svr.ServerID	SELECT svr.ServerID , errs.LogMessage
errs.ServerID	FROM Servers AS svr
INNER JOIN	CROSS APPLY
CROSS APPLY	(
WITHIN GROUP	SELECT TOP 3 LogMessage
WHERE ServerID = svr.ServerID	FROM Errors
WHERE ServerID = errs.ErrorID	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>

[**Visit PassLeader and Download Full Version 70-761 Exam Dumps**](#)