2018 NEW Microsoft 70-761: Querying Data with Transact-SQL Exam Questions and Answers RELEASED in Braindump2go.com Online IT Study Website Today!

2018 Braindump2go Valid Microsoft 70-761 Exam Preparation Materials:

2018 Latest 135Q&As 70-761 PDF Dumps and VCE Dumps:

https://www.braindump2go.com/70-761.html

QUESTION 106
Hotspot Question
You have a table named HumanResources.Department that was created with the query shown in the exhibit. (Click the Exhibit button.)

```
CREATE TABLE HumanResources.Department
(
    DepID int IDENTITY(1,1) NOT NULL PRIMARY KEY CLUSTERED
  , DeptName varchar(50) NOT NULL
  , ManagerID INT NULL
  , ParentDeptID int NULL
  , SysStartTime datetime2 GENERATED ALWAYS AS ROW START NOT TULL
  , SysEndTime datetime2 GENERATED ALWAYS AS ROW END NOT NULL
  , PERIOD FOR SYSTEM_TIME (SysStartTime,SysEndTime)
)
WITH (SYSTEM_VERSIONING = ON)
;
```

You need to query temporal data in the table.
In the table below, identify the Transact-SQL segments that must be used to retrieve the appropriate data. **NOTE:** Make only one selection in each column.

Free Download Braindump2go Latest 70-761 Exam PDF and VCE Dumps 135q from www.braindump2go.com

100% Pass Guaranteed! 100% Real Exam Questions!

https://www.braindump2go.com/70-761.html

Answer Area

Clause	At a particular point in time	Only from history table
All	○	○
FROM	○	○
AS OF	○	○
BETWEEN	○	○
CONTAINED IN	○	○

Answer:

Answer Area

Clause	At a particular point in time	Only from history table
All	○	○
FROM	○	○
AS OF	◉	○
BETWEEN	○	○
CONTAINED IN	○	◉

Explanation:
AS OF: Returns a table with a rows containing the values that were actual (current) at the specified point in time in the past.
CONTAINED IN: If you search for non-current row versions only, we recommend you to use CONTAINED IN as it works only with the history table and will yield the best query performance.
Incorrect Answers:
Not ALL: Returns the union of rows that belong to the current and the history table.
References: https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system- versioned-temporal-table

QUESTION 107
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:
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
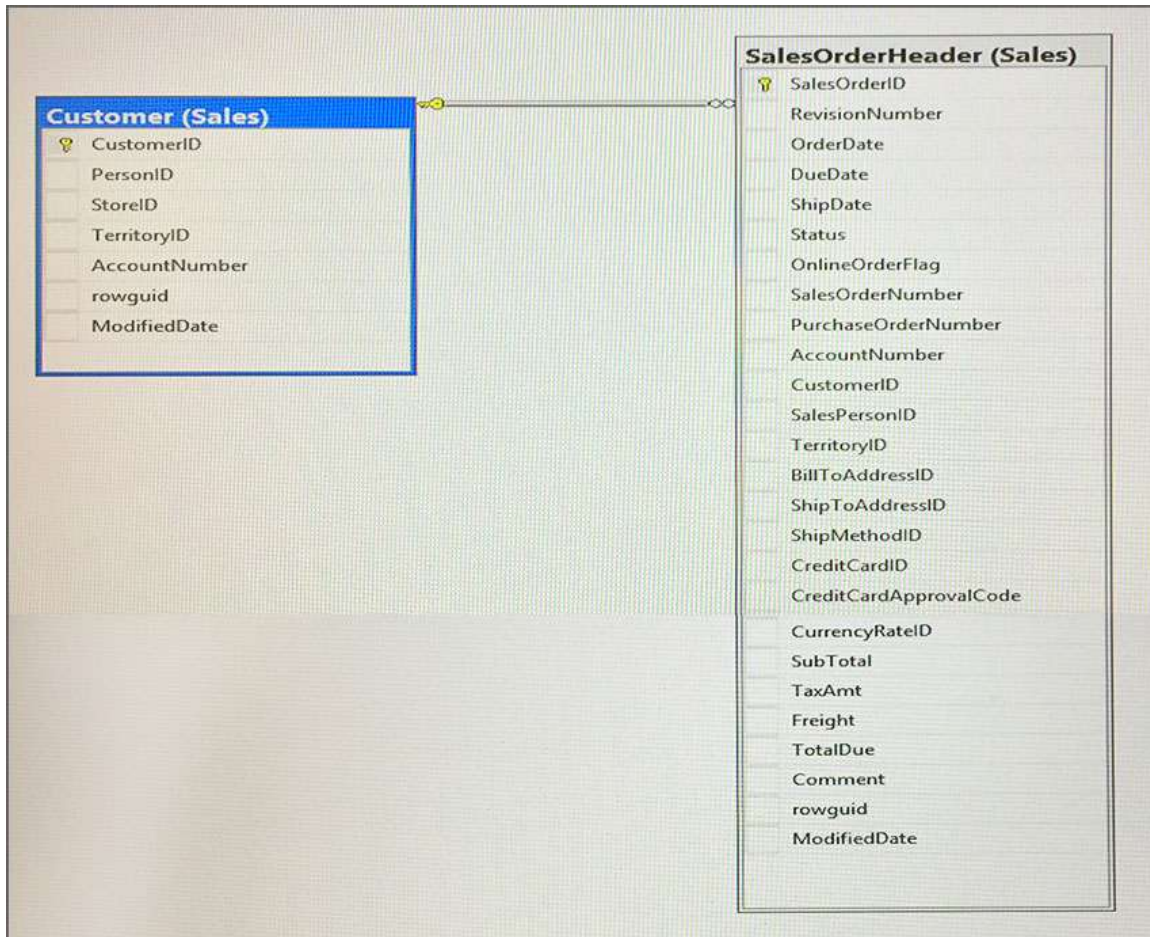
QUESTION 108
You have a database that includes the tables shown in the exhibit. (Click the exhibit button.)

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

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

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

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

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

Answer: A
Explanation:
COALESCE evaluates the arguments in order and returns the current value of the first expression that initially does not evaluate to NULL.
References: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/coalesce-transact-sql

QUESTION 109
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.

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

QUESTION 110
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You have a database that contains several connected tables. The tables contain sales data for customers in the United States only.
You have the following partial query for the database. (Line numbers are included for reference only.)

```
01  SELECT CountryName, StateProvinceName, CityName, Quantity*UnitPrice as TotalSales
02  FROM Sales
03
04  ORDER BY CountryName, StateProvinceName, CityName
```

You need to complete the query to generate the output shown in the following table.

CountryName	StateProvinceName	CityName	TotalSales
NULL	NULL	NULL	$23395792.75
Unites States	NULL	NULL	$23395792.75
Unites States	Alabama	NULL	$646508.75
Unites States	Alabama	Bazemore	$34402.00
Unites States	Alabama	Belgreen	$51714.65

Which statement clause should you add at line 3?

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

Answer: E
Explanation:
Example of GROUP BY CUBE result set:
In the following example, the CUBE operator returns a result set that has one grouping for all possible combinations of columns in the CUBE list and a grand total grouping.

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

References: https://technet.microsoft.com/en-us/library/bb522495(v=sql.105).aspx

QUESTION 111
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.
You have a database that contains several connected tables. The tables contain sales data for customers in the United States only.

You have the following partial query for the database. (Line numbers are included for reference only.)

```
01   SELECT CountryName, StateProvinceName, CityName, Quantity*UnitPrice as TotalSales
02   FROM Sales
03
04   ORDER BY CountryName, StateProvinceName, CityName
```

You need to complete the query to generate the output shown in the following table.

CountryName	StateProvinceName	CityName	TotalSales
NULL	NULL	NULL	$23395792.75
NULL	NULL	Abbottsburg	$45453.25
NULL	NULL	Absecon	$33140.15
NULL	NULL	Accomac	$43226.80
NULL	NULL	Aceitunas	$23001.40

Which statement clause should you add at line 3?

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

Answer: C
Explanation: In the result sets that are generated by the GROUP BY operators, NULL has the following uses:
If a grouping column contains NULL, all null values are considered equal, and they are put into one NULL group.
When a column is aggregated in a row, the value of the column is shown as NULL.
Example of GROUP BY ROLLUP result set:

Region	Country	Store	SalesPersonID	Total Sales
NULL	NULL	NULL	NULL	297597.8
NULL	NULL	NULL	284	33633.59
NULL	NULL	Spa and Exercise Outfitters	284	32774.36
NULL	FR	Spa and Exercise Outfitters	284	32774.36
Europe	FR	Spa and Exercise Outfitters	284	32774.36
NULL	NULL	Versatile Sporting Goods Company	284	859.232
NULL	DE	Versatile Sporting Goods Company	284	859.232
Europe	DE	Versatile Sporting Goods Company	284	859.232
NULL	NULL	NULL	286	246272.4
NULL	NULL	Spa and Exercise Outfitters	286	246272.4
NULL	FR	Spa and Exercise Outfitters	286	246272.4
Europe	FR	Spa and Exercise Outfitters	286	246272.4
NULL	NULL	NULL	289	17691.83
NULL	NULL	Versatile Sporting Goods Company	289	17691.83
NULL	DE	Versatile Sporting Goods Company	289	17691.83
Europe	DE	Versatile Sporting Goods Company	289	17691.83

References: https://technet.microsoft.com/en-us/library/bb522495(v=sql.105).aspx

QUESTION 112
Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.

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

You have the following partial query for the database. (Line numbers are included for reference only.)

```
01  SELECT CountryName, StateProvinceName, CityName, Quantity*UnitPrice as TotalSales
02  FROM Sales
03
```

You need to complete the query to generate the output shown in the following table.

CountryName	StateProvinceName	CityName	TotalSales
United States	Wyoming	Yoder	$7638.11
United States	Wyoming	NULL	$1983745.99
United States	NULL	NULL	$2387435981.22
NULL	NULL	NULL	$2387435981.22

Which statement clause should you add at line 3?

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

Answer: F
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

QUESTION 113
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 datetime(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 = CustomerHistory))
```

You need to view all customer data.
Which Transact-SQL statement should you run?

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

B.
```
    SELECT CustomerID, FirstName, LastName, TaxIdNumber, Addr
AnnualRevenue, DateCreated, ValidFrom, ValidTo
    FROM Customers
    FOR SYSTE_TIME ALL ORDER BY ValidFrom
```

C.
```
    SELECT c.CustomerId, c.FirstName, c.LastName, c.Address, c.Validf
c.ValidTo
    FROM Customers AS c
    ORDER BY c.CustomerID
    FOR JSON AUTO, ROOT('Customers')
```

D.
```
    SELECT * FROM (SELECT CustomerID, FirstName, LastName, Addre
AnnualRevenue, DateCreated
    FROM Customers) AS Customers PIVOT(AVG(AnnualRevenue)
    FOR DateCreated IN([2014])) AS PivorCustomers
    ORDER BY LastName, FirstName
```

E.
```
SELECT CustomerID, AVG(AnnualRevenue)
AS AverageAnnualRevenue, FirstName, LastName, Address, DateCreat
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 00:00:00.0000000' AND '2015-01-01 00:00:00.000

H.   SELECT CustomerID, FirstName, LastName, TaxIdNumber, Addre
    ValidFrom, ValidTo
        FROM Customers
        WHERE DateCreated
        BETWEEN '20140101' and '20141231'

Answer: B
Explanation:
The FOR SYSTEM_TIME ALL clause returns all the row versions from both the Temporal and History table.
References: https://msdn.microsoft.com/en-us/library/dn935015.aspx