KTest

更に上のクオリティ　更に上のサービス

問題集

http://www.ktest.jp

1年で無料進級することに提供する

Exam : 70-461

Title : Querying Microsoft SQL Server 2012

Version : DEMO

1.You use a Microsoft SQL Server database that contains a table.
The table has records of web requests as shown in the exhibit. (Click the Exhibit button.)

HttpRequest
HttpRequestId
RequestDateTime
ClientIP
ClientUsername
ServerIP
ServerPort
HttpMethodId
UriStem
UriQuery
ServerStatus
ServerSubstatus
ServerWin32Status
BytesSent
BytesReceived
TimeTaken
ClientVersion
ClientHost
ClientUserAgentId
ClientId
SessionId
TimeSpent

Your network has three web servers that have the following IP addresses:
- 10.0.0.1
- 10.0.0.2
- 10.0.0.3

You need to create a query that displays the following information:
- The number of requests for each web page (UriStem) grouped by the web server (ServerIP) that served the request
- A column for each server

Which Transact-SQL query should you use?

A
```
SELECT
   UriStem,
   [10.0.0.1],
   [10.0.0.2],
   [10.0.0.3],
FROM
 (SELECT HttpRequestId, ServerIP, UriStem FROM HttpRequest) r
PIVOT (
COUNT (r.HttpRequestId)
FOR r.ServerIP IN ([10.0.0.1], [10.0.0.2], [10.0.0.3])
) AS pvt
ORDER BY
 pvt.UriStem
```

B
```
SELECT
  UriStem,
SUM(CASE WHEN ServerIP = '10.0.0.1' THEN 1 ELSE 0 END) AS
[10.0.0.1],
SUM(CASE WHEN ServerIP = '10.0.0.2' THEN 1 ELSE 0 END) AS
[10.0.0.2],
SUM(CASE WHEN ServerIP = '10.0.0.3' THEN 1 ELSE 0 END) AS
[10.0.0.3],
FROM
    HttpRequest
GROUP BY
    ServerIP
ORDER BY
   UriStem
```

C
```
SELECT
 UriStem,
Server,
Requests
FROM
 (SELECT HttpRequestId, ServerIP, UriStem FROM HttpRequest) r
UNPIVOT (
Requests FOR Server IN ([ServerIP])
) AS pvt
ORDER BY
Pvt.UriStem
```

D
```
DECLARE @Results TABLE (
 UriStem VARCHAR(255),
[10.0.0.1] INT,
[10.0.0.2] INT,
[10.0.0.3] INT)

INSERT INTO @Results (UriStem, [10.0.0.1])
SELECT UriStem COUNT(HttpRequestId)
FROM HttpRequest
WHERE ServerIP - '10.0.0.1'

UPDATE @Results
SET [10.0.0.2] = COUNT(HttpRequestId)
FROM HttpRequest h INNER JOIN @Results r ON h.UriStem =
r.UriStem
WHERE ServerIP = '10.0.0.2'

UPDATE @Results
SET [10.0.0.3] = COUNT(HttpRequestId)
FROM HttpRequest h INNER JOIN @Results r ON h.UriStem =
r.UriStem
WHERE ServerIP = '10.0.0.3'

SELECT
     UriStem,
   [10.0.0.1] ,
   [10.0.0.2] ,
   [10.0.0.3]
FROM
@Results
```

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

Answer: A

Explanation:

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.

References: https://docs.microsoft.com/en-us/sql/t-sql/queries/from-using-pivot-and-unpivot?view=sql-server-2017

2.DRAG DROP

You develop a Microsoft SQL Server database for a sales ordering application.

You want to create a report that displays the increase of order quantities over the previous year for each product.

You need to write a query that displays:

- Product name,
- Year of sales order,
- Sales order quantity, and
- Increase of order quantity over the previous year.

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.

Statements

```
FROM Sales.SalesOrderHeader SOH
INNER JOIN Sales.SalesOrderDetail SOD ON
SOH.SalesOrderID = SOD.SalesOrderID
INNER JOIN Production.Product PRO ON
SOD.ProductID = PRO.ProductID
```

```
GROUP BY PRO.Name, OrderDate
```

```
GROUP BY PRO.Name, YEAR(OrderDate)
```

```
SELECT Pro.Name, YEAR(OrderDate), SUM
(SOD.OrderQty), SUM(SOD.OrderQty) -
LEAD(SUM(SOD.OrderQty), 1, 0)
OVER (PARTITION BY PRO.Name ORDER BY YEAR
(OrderDate) DESC)
```

```
SELECT Pro.Name, YEAR(OrderDate), SUM
(SOD.OrderQty), SUM(SOD.OrderQty) -
LAG(SUM(SOD.OrderQty), 1, 0)
OVER (PARTITION BY PRO.Name ORDER BY YEAR
(OrderDate) DESC)
```

Answer Area

Answer:

Statements

```
FROM Sales.SalesOrderHeader SOH
INNER JOIN Sales.SalesOrderDetail SOD ON
SOH.SalesOrderID = SOD.SalesOrderID
INNER JOIN Production.Product PRO ON
SOD.ProductID = PRO.ProductID
```

```
GROUP BY PRO.Name, OrderDate
```

```
GROUP BY PRO.Name, YEAR(OrderDate)
```

```
SELECT Pro.Name, YEAR(OrderDate), SUM
(SOD.OrderQty), SUM(SOD.OrderQty) -
LEAD(SUM(SOD.OrderQty), 1, 0)
OVER (PARTITION BY PRO.Name ORDER BY YEAR
(OrderDate) DESC)
```

```
SELECT Pro.Name, YEAR(OrderDate), SUM
(SOD.OrderQty), SUM(SOD.OrderQty) -
LAG(SUM(SOD.OrderQty), 1, 0)
OVER (PARTITION BY PRO.Name ORDER BY YEAR
(OrderDate) DESC)
```

Answer Area

```
FROM Sales.SalesOrderHeader SOH
INNER JOIN Sales.SalesOrderDetail SOD ON
SOH.SalesOrderID = SOD.SalesOrderID
INNER JOIN Production.Product PRO ON
SOD.ProductID = PRO.ProductID
```

```
SELECT Pro.Name, YEAR(OrderDate), SUM
(SOD.OrderQty), SUM(SOD.OrderQty) -
LEAD(SUM(SOD.OrderQty), 1, 0)
OVER (PARTITION BY PRO.Name ORDER BY YEAR
(OrderDate) DESC)
```

```
GROUP BY PRO.Name, YEAR(OrderDate)
```

Explanation:

Box 1: FROM ..

Box 2: LAG (not LEAD)

Lag accesses data from a previous row in the same result set without the use of a self-join starting with SQL Server 2012 (11.x). LAG provides access to a row at a given physical offset that comes before the current row. Use this analytic function in a SELECT statement to compare values in the current row with values in a previous row.

Not lead: Lead accesses data from a subsequent row in the same result set without the use of a self-join starting with SQL Server 2012 (11.x). LEAD provides access to a row at a given physical offset that follows the current row.

Box 3: GROY BY PRO.NAME, YEAR (OrderDate)

References: https://docs.microsoft.com/en-us/sql/t-sql/functions/lag-transact-sql?view=sql-server2017

3.You develop a Microsoft SQL Server database that contains a table named Employee, defined as follows:

```
CREATE TABLE [dbo].[Employee]
(
[EmployeeID] int PRIMARY KEY
[Firstname] varchar(50) NOT NULL,
[LastName] varchar(50) NOT NULL,
[DepartmentID] int NOT NULL,
[HireDate] date NOT NULL
)
```

You need to create a view that contains two computed columns representing the month and the year of the [HireDate] of each Employee.
Which function should you use?
A. DATENAME( )
B. CONVERT( )
C. TRYDATEDIFF( )
D. MONTH( ) and YEAR( )
Answer: D
Explanation:
The Month function returns an integer that represents the month of the specified date.
The Year function an integer that represents the year of the specified date.
References:
https://docs.microsoft.com/en-us/sql/t-sql/functions/month-transact-sql?view=sql-server-2017
https://docs.microsoft.com/en-us/sql/t-sql/functions/year-transact-sql?view=sql-server-2017

4.You administer a Microsoft SQL Server database named ContosoDb.
The database has the following schema collection:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://tempuri.org/po.xsd"
xmlns="http://tempuri.org/po.xsd"
elementFormDefault="qualified">
<xs:element name="purchaseOrder" type="PurchaseOrderType"/>
<xs:complexType name="PuchaseOrderType">
<xs:sequence>
<xs:element name="items" type="Items"/>
</xs:sequence>
<xs:attribute name="orderDate" type="xs:date"/>
<xs:attribute name="requiresApproval" type="xs:boolean"/>
</xs:complexType>
<xs:complexType name ="Items">
<xs:sequence>
<xs:element name="item" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="productName" type="xs:string"/>
<xs:element name="quantity" type="xs:positiveInteger"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

The database has a table named ReceivedPurchaseOrders that includes an XML column named PurchaseOrder by using the above schema.

You need to set the requiresApproval attribute of the XML documents to false if they contain more than 50 items.

Which Transact-SQL query should you run?

A

```
UPDATE ReceivedPurchaseOrders SET PurchaseOrder.modify('
  declare namespace MI="http:/ tempuri.org/po.xsd";
  replace value of (/MI:purchaseOrder/MI:requiresApproval)
   with (
   if  (count(/MI:purchaseOrder/MI:items/MI:item)>50) then
          xs:boolean("true")
  else
          xs:boolean("false")
)');
```

B

```
UPDATE ReceivedPurchaseOrders SET PurchaseOrder.modify('
  declare namespace MI="http:/tempuri.org/po.xsd";
  replace value of (/MI:purchaseOrder/MI:requiresApproval)
   with (
   if  (count(/MI:purchaseOrder/MI:items)>50) then
          xs:boolean("true")
  else
          xs:boolean("false")
)');
```

C

```
UPDATE ReceivedPurchaseOrders SET PurchaseOrder.modify('
  declare namespace MI="http:/tempuri.org/po.xsd";
  replace value of (/MI:purchaseOrder/@requiresApproval)[1]
   with (
   if  (count(/MI:purchaseOrder/MI:items/MI:item)>50) then
          xs:boolean("true")
  else
          xs:boolean("false")
)');
```

D

```
UPDATE ReceivedPurchaseOrders SET PurchaseOrder.modify('
  declare namespace MI="http:/tempuri.org/po.xsd";
  replace value of (/MI:purchaseOrder/@requiresApproval)[1]
   with (
   if  (count(/MI:purchaseOrder/MI:items)>50) then
          xs:boolean("true")
  else
          xs:boolean("false")
)');
```

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

Answer: D

Explanation:

Replace value of (XML DML) updates the value of a node in the document.
Example: -- update text in the first manufacturing step
SET @myDoc.modify('
replace value of (/Root/Location/step[1]/text())[1]
with "new text describing the manu step"
');

5.DRAG DROP

Your Microsoft SQL Server database contains tables as shown below.
You have tables that were created by running the following Transact-SQL statements:

```
CREATE TABLE dbo.Category
(
CategoryID INT NOT NULL IDENTITY(1,1) CONSTRAINT PK_Category
PRIMARY KEY CLUSTERED
, CategoryName VARCHAR(200) NOT NULL
, ProductDescription VARCHAR(1000) NULL
, IsActive BIT DEFAULT (1)
)
GO

CREATE TABLE dbo.Product
(
ProductID INT NOT NULL IDENTITY(1,1) CONSTRAINT PK_Product
PRIMARY KEY CLUSTERED
, ProductName VARCHAR(200) NOT NULL
, CategoryID INT NOT NULL
, ProductDescription VARCHAR(1000) NULL
, ListPrice MONEY NOT NULL
, Quantity INT NOT NULL
, CONSTRAINT FK_Product_Category FOREIGN KEY (CategoryID)
REFERENCES Category(CategoryID)
)
GO
```

The Product table contains 10,000 records. The maximum ProductID is 11,000.
There are 12 rows in the Category table. The maximum CategoryID is 12.
The Product table contains at least one product in every category.
Data in the tables was accidently modified. To correct this, you need to make some updates directly to the tables. You issue several statements.
Which result or results will you obtain for each Transact-SQL statement? To answer, drag the appropriate results to the correct Transact-SQL statements. Each result 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.

SQL Statements

- The statement succeeds.
- The statement fails because the syntax is incorrect.
- The statement fails because the primary key constraint in the Product table is violated.
- The statement fails because the value for an identity column cannot be explicitly specified during an insert
- The statement fails because the foreign key constraint is violated.
- The statement fails because an identity column value cannot ba changed during an update statement.
- The statement fails because the data type is incorrect for one of the fields.

Answer area

SET IDENTITY_INSERT dbo.Product ON;INSERT dbo.Product (ProductID, ProductName, categoryID, ProductDescription, ListPrice, Quantity) VALUES (20000, 'Strawberry Yogurt', 9 '', 0.98*6, 57);SET IDENTITY_INSERT dbo.Product OFF;	Result
DELETE dbo.Category WHERE CategoryID = 11	Result
INSERT dbo.Product (ProductName, CategoryID, ListPrice, Quantity) VALUES ('Chocolate Cake', 25, 5, 100);	Result
UPDATE dbo.Category SET IsActive='-' WHERE CategoryID = 5	Result

Answer:

SQL Statements

- The statement succeeds.
- The statement fails because the syntax is incorrect.
- The statement fails because the primary key constraint in the Product table is violated.
- The statement fails because the value for an identity column cannot be explicitly specified during an insert
- The statement fails because the foreign key constraint is violated.
- The statement fails because an identity column value cannot ba changed during an update statement.
- The statement fails because the data type is incorrect for one of the fields.

Answer area

SET IDENTITY_INSERT dbo.Product ON;INSERT dbo.Product (ProductID, ProductName, categoryID, ProductDescription, ListPrice, Quantity) VALUES (20000, 'Strawberry Yogurt', 9 '', 0.98*6, 57);SET IDENTITY_INSERT dbo.Product OFF;	The statement succeeds.
DELETE dbo.Category WHERE CategoryID = 11	The statement fails because the foreign key constraint is violated.
INSERT dbo.Product (ProductName, CategoryID, ListPrice, Quantity) VALUES ('Chocolate Cake', 25, 5, 100);	The statement fails because the value for an identity column cannot be explicitly specified during an insert
UPDATE dbo.Category SET IsActive='-' WHERE CategoryID = 5	The statement fails because the data type is incorrect for one of the fields.

Explanation:

Box 1: The SET IDENTITY_INSERT command allows explicit values to be inserted into the identity column of a table.

Box 2: The Product table contains at least one product in every category.

Box 3:

Box 4: Bit is a data type that can take a value of 1, 0, or NULL.

References: https://docs.microsoft.com/en-us/sql/t-sql/data-types/bit-transact-sql?view=sql-server-2017 https://docs.microsoft.com/en-us/sql/t-sql/statements/set-identity-insert-transact-sql?view=sql-server-2017