BrainDumps Collection

# Microsoft

## 70-461 Exam

**Microsoft Querying Microsoft SQL Server 2012/2014 Exam**

Thank you for Downloading 70-461 exam PDF Demo

You can also try our 70-461 practice exam software

**Download Free Demo**

https://www. braindumpscollection.com/70-461.html

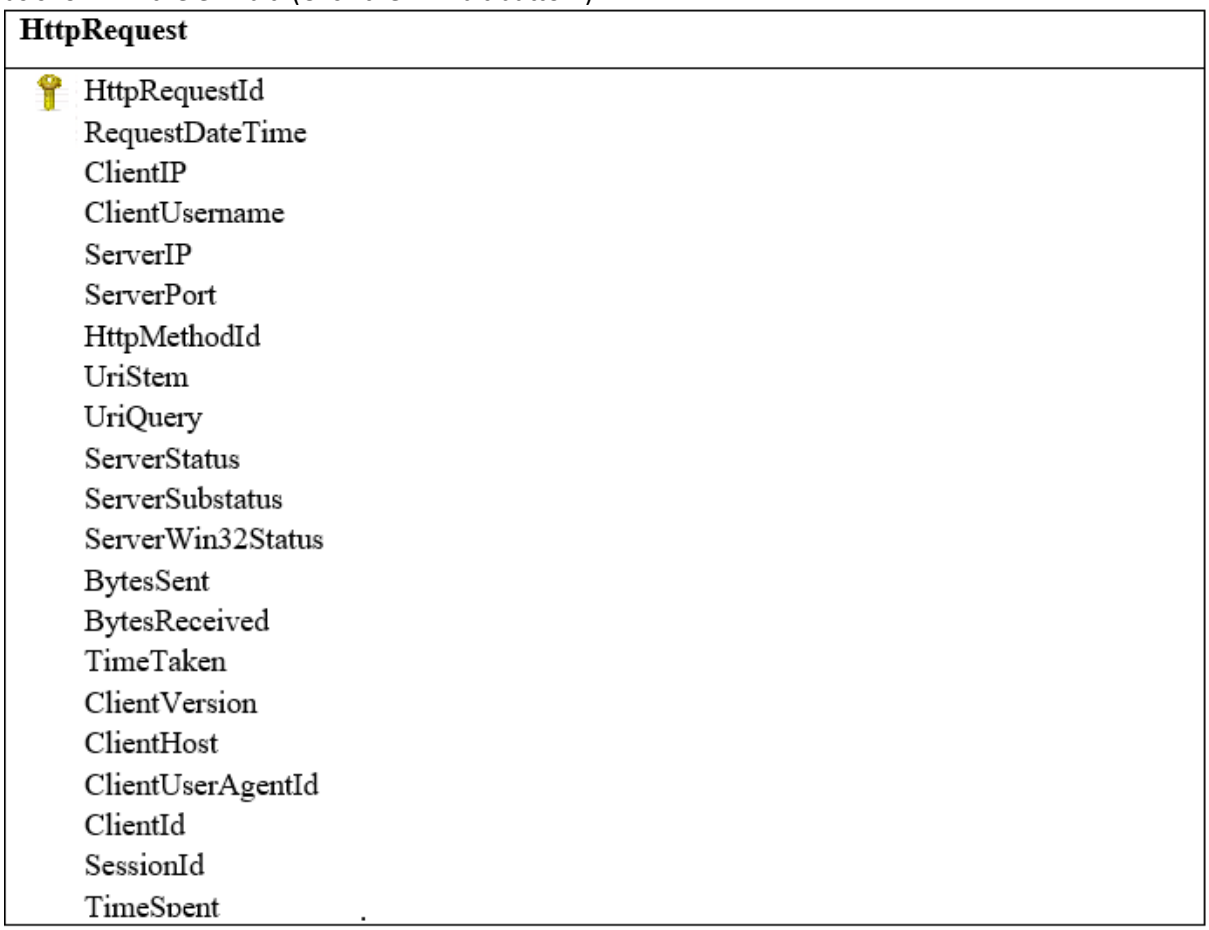
DEMO VERSION (LIMITED CONTENT)

Questions & Answers

# Version: 23.0

## Question: 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.)




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

## Question: 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:**

## 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) -
LAG(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-server-2017

## Question: 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

## Question: 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"
');

## Question: 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:**

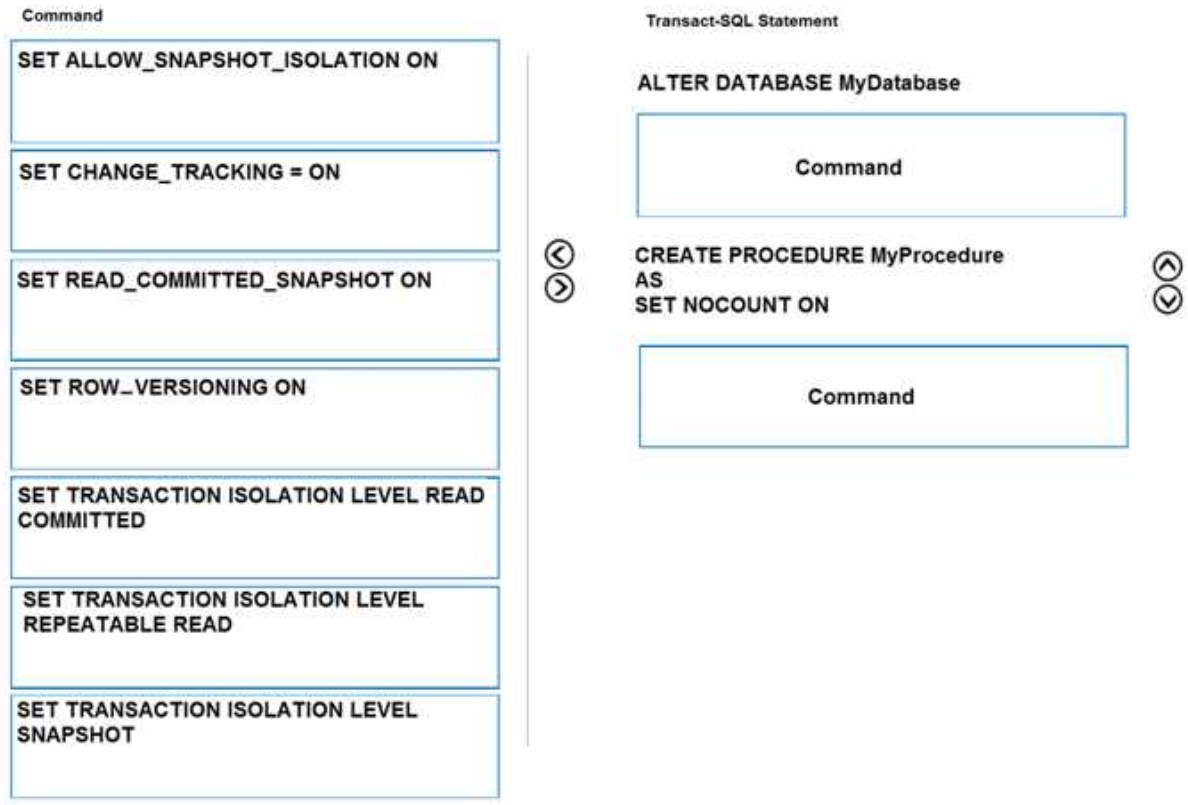
| Statement | |
|---|---|
| SET IDENTITY_INSERT dbo.Product ON;INSERT dbo.Product (ProductID, ProductName, CategoryID, ProductDescription, ListPrice, Quantity) VALUES (20000, 'Strawberry Yogurt', 9, '', 0.98*6, 57);SET IDENTITY_INSERT dbo.Product OFF; | The statement succeeds. |
| DELETE dbo.Category 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 = '-1' 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.

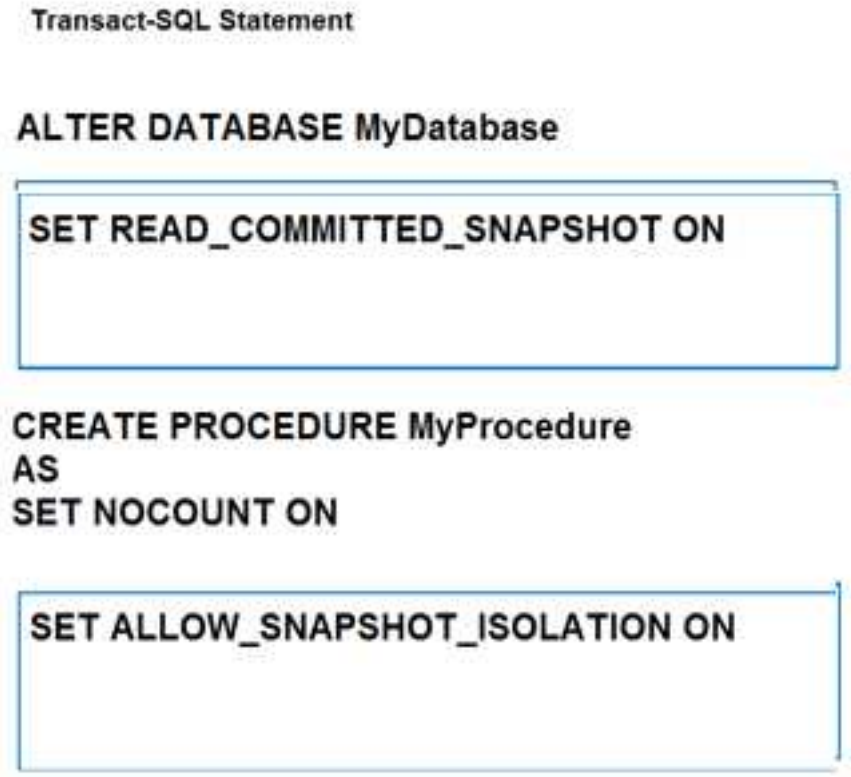
Box 2:
The Product table contains at least one product in every category.
Box 3:
Box 4:
Bit is a data type that can take a value of 1, 0, or NULL.
References:
https://docs.microsoft.com/en-us/sql/t-sql/data-types/bit-transact-sql?view=sql-server-2017
https://docs.microsoft.com/en-us/sql/t-sql/statements/set-identity-insert-transact-sql?view=sql-server-2017

## Question: 6

DRAG DROP
You develop an application that uses data from a Microsoft SQL Server database.
Your application experiences blocking problems.
You need to enable row versioning and you want connections to have row versioning enabled by default.
How should you complete the Transact-SQL statement? To answer, drag the appropriate command to the correct positions. Each command 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.



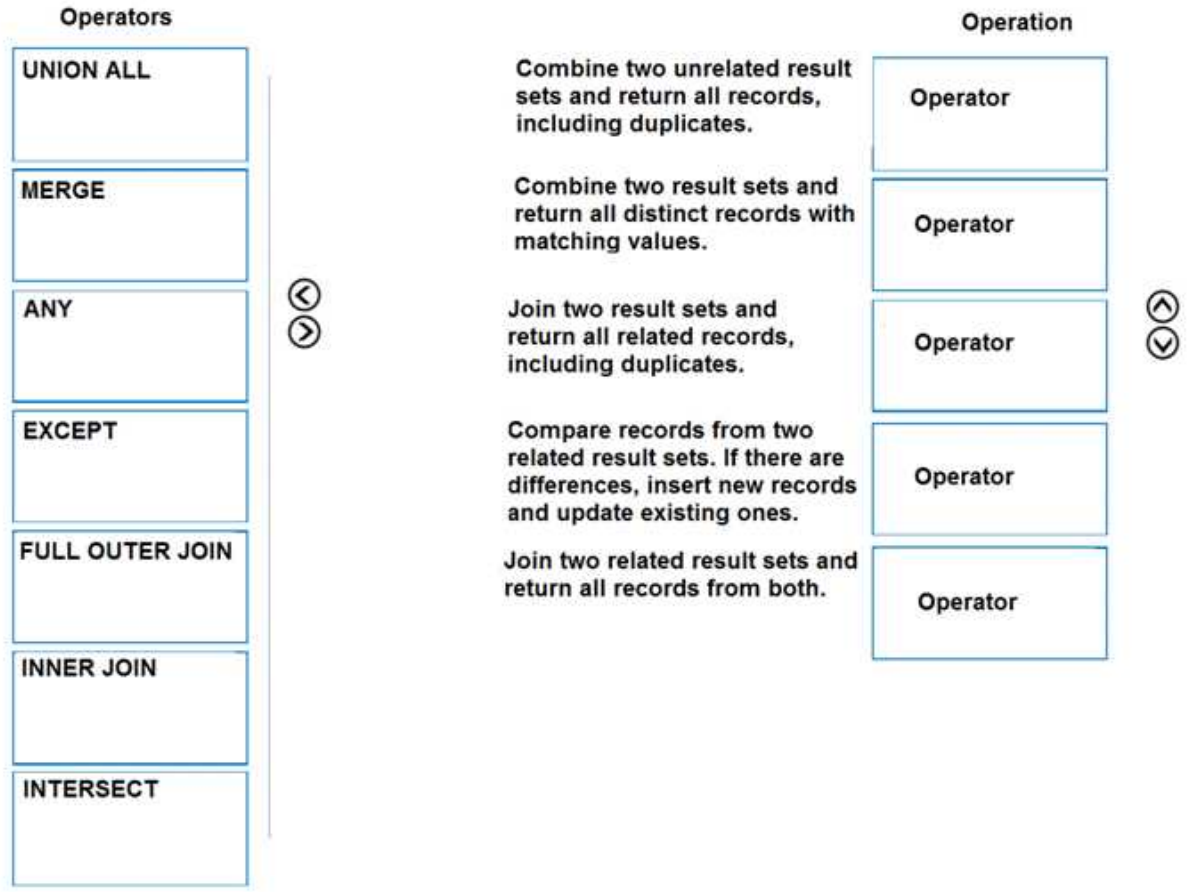

## Answer:



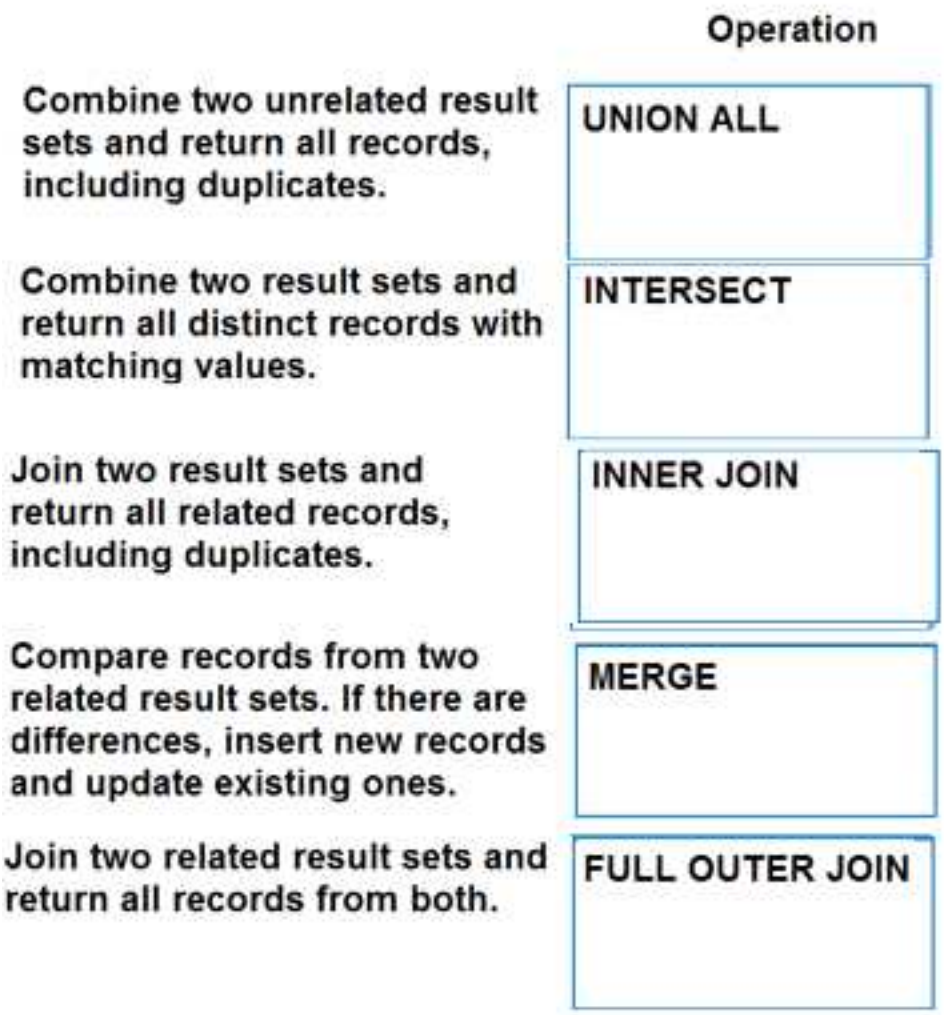

Explanation:
You can use a row versioning-based isolation level.
Set READ_COMMITTED_SNAPSHOT database option ON to enable read-committed transactions to use row versioning, and use snapshot isolation.
References: https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-2017

## Question: 7

DRAG DROP
You are a developer for a Microsoft SQL Server database. You need to write a stored procedure that performs several operations in the most efficient way possible.
Which operator or operators should you use? To answer, drag the appropriate operators to the correct operations. Each operator 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.


Operators
UNION ALL
MERGE
ANY
EXCEPT
FULL OUTER JOIN
INNER JOIN
INTERSECT
Operation
Combine two unrelated result sets and return all records, including duplicates.
Operator
Combine two result sets and return all distinct records with matching values.
Operator
Join two result sets and return all related records, including duplicates.
Operator
Compare records from two related result sets. If there are differences, insert new records and update existing ones.
Operator
Join two related result sets and return all records from both.
Operator

**Answer:**




Explanation:
Box 1: UNION ALL
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.
UNION ALL Incorporates all rows into the results. This includes duplicates. If ALL is not specified, duplicate rows are removed.
Box 2: INTERSECT
INTERSECT returns distinct rows that are output by both the left and right input queries operator.
Box 3: INNER JOIN
The INNER JOIN keyword selects records that have matching values in both tables.
Box 4: MERGE
Merge performs insert, update, or delete operations on a target table based on the results of a join with a source table. For example, you can synchronize two tables by inserting, updating, or deleting rows in one table based on differences found in the other table.
Box 5: FULL OUTER JOIN
The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.
Note: FULL OUTER JOIN can potentially return very large result-sets!

## Question: 8

You develop a database application for Microsoft SQL Server and Microsoft Azure SQL Database.
You need to raise an exception and transfer execution to a CATCH block.
You need to ensure that the exception returns output in the following format:
Msg 51000, Level 16, State 1, Line 1
The record does not exist.
Which Transact-SQL statement should you run?

A

```
DECLARE @Message NVARCHAR(2048);
SELECT @Message = FORMATMESSAGE('The record does not exist.');
THROW 51000
, 1
, @Message
```

B

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

C

```
THROW ERROR_MESSAGE
( 'The record does not exist.'), 1
```

D

```
THROW 51000
, FORMATMESSAGE('The record does not exist.')
, 1
```

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

**Answer: B**

Explanation:
The following example shows how to use the THROW statement to raise an exception.
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.
References: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/throw-transact-sql?view=sql-server-2017

## Question: 9

DRAG DROP
You develop a database application for Microsoft SQL Server 2012 and Microsoft Azure SQL Database.
You create a table named Purchasing.vVendorWithAddresses as shown in the following table.

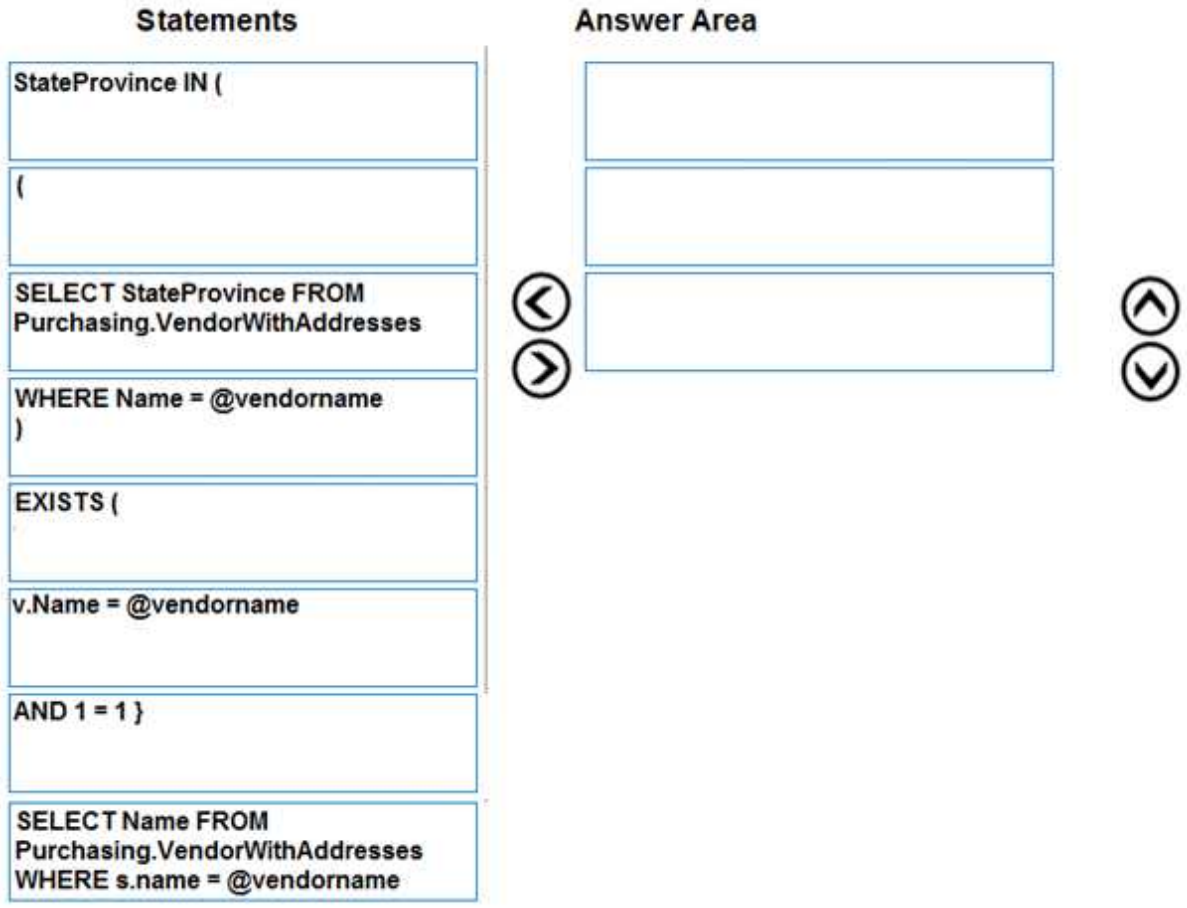
| | BusinessEntityID | Name | Address | City | StateProvinceName | PostalCode |
|---|---|---|---|---|---|---|
| 1 | 1492 | Australia Bike Retailer | 28 San Marino Ct. | Bellingham | Washington | 98225 |
| 2 | 1494 | Allenson Cycles | 4659 Montoya | Altadena | California | 91001 |
| 3 | 1496 | Advanced Bicycles | 7995 Edwards Ave. | Lynnwood | Washington | 98036 |
| 4 | 1498 | Trikes, Inc. | 90 Sunny Ave | Berkley | California | 94704 |
| 5 | 1500 | Morgan Bike Accessories | 9098 Story Lane | Albany | New York | 12210 |
| 6 | 1502 | Cycling Master | 4823 Stonewood Ct. | Walla Walla | Washington | 99362 |
| 7 | 1504 | Chicago Rent-All | 15 Pear Dr. | Newport Beach | California | 92625 |
| 8 | 1506 | Greenwood Athletic Company | 6441 Co Road | Lemon Grove | Arizona | 85252 |
| 9 | 1508 | Compete Enterprises, Inc. | 50 Via Del Sol | Lynnwood | Washington | 98036 |
| 10 | 1510 | International | 683 Larch Ct. | Salt Lake City | Utah | 84101 |

You write the following Transact-SQL (Line numbers are included for reference only.)
01 CREATE PROCEDURE

```
02 usp_GetVendorNeighbors
03 @vendorname nvarchar(50)
04 AS
05 SELECT name FROM
06 Purchasing.vVendorWithAddresses t
07 WHERE
08 . . .
```

You need to add Transact-SQL statements at line 08 to ensure that GetVendorInStateNeighbors

returns the names of vendors that are located in all states where the vendor specified in the @vendorname parameter has a location.
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.




**Answer:**

**Answer Area**

Explanation:
The IN statement determines whether a specified value matches any value in a subquery or a list.
Incorrect:
The EXISTS command specifies a subquery to test for the existence of rows.
References: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/in-transact-sql?view=sql-server-2017

## Question: 10

A local bank uses a SQL Server database to manage accounts. You are developing a stored procedure that contains multiple Transact-SQL INSERT statements.
The stored procedure must use transaction management to handle errors.
You need to ensure that the stored procedure rolls back the entire transaction if a run-time occurs.
Which Transact-SQL statement should you add to the stored procedure?

A. SET ARITHABORT ON
B. SET NOEXEC ON
C. SET TRANSACTION ISOLATION LEVEL ON
D. SET XACT_ABORT ON

**Answer: D**

Explanation:
SET 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.

References: https://docs.microsoft.com/en-us/sql/t-sql/statements/set-xact-abort-transact-sql?view=sql-server-2017

## Thank You for trying 70-461 PDF Demo

To try our 70-461 practice exam software visit link below

https://www. braindumpscollection.com/70-461.html

# Start Your 70-461 Preparation

Use Coupon “20OFF” for extra 20% discount on the purchase of Practice Test Software. Test your 70-461 preparation with actual exam questions.