TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 1 (461P_4.2_13)

You are configuring an appropriate transaction isolation level for a session. You want to ensure the following:

*Statements in a particular transaction cannot read data that has been altered but not committed.
*Other transactions cannot modify data read by the current transaction until the transaction commits.
*Other transactions cannot insert rows with key values that would fall into the range read by statements in the current transaction until that transaction completes.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

2. SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

3. SET TRANSACTION ISOLATION LEVEL SNAPSHOT

4. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE <Correct>

5. >>SET TRANSACTION ISOLATION LEVEL READ COMMITTED

Explanation:
You use the SET TRANSACTION ISOLATION LEVEL SERIALIZABLE statement when you want to ensure that Transact-SQL statements cannot read data that has been altered but not committed, that other transactions cannot modify data read by the current transaction until the transaction commits, and that other transactions cannot insert rows with key values that would fall into the range read by statements in the current transaction until that transaction completes.

You use the SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED statement when you want to allow statements to read rows altered by other transactions, but those transactions have not yet been committed.

You use the SET TRANSACTION ISOLATION LEVEL READ COMMITTED statement when you want to ensure that statements cannot read data altered by other transactions when those transactions have not yet been committed. When you set the READ COMMITTED transaction level, data can be altered by other transactions between individual statements within the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL REPEATABLE READ statement when you want to ensure that statements cannot read data altered by other transactions not yet committed and that other transaction cannot modify data read by the current transaction until the current transaction commits. When the REPEATABLE READ isolation level is set, it is possible for other transactions to insert new rows that meet the search conditions of statements included in the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL SNAPSHOT statement when you want to ensure that data read by any statement in a transaction will be transactionally consistent with how that data existed at the start of the transaction. SNAPSHOT transactions do not block other transactions from writing data.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SET TRANSACTION ISOLATION LEVEL (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173763.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 2 (461P_4.1_07)

You are optimizing several Transact-SQL statements. You have determined that a particular statement will perform more efficiently if rows in a table locked by another transaction are skipped rather than waiting for the locks to be released.

Which of the following table hints will you use in your Transact-SQL statements when performing optimization?

1. ROWLOCK

2. TABLOCK

3. PAGLOCK

4. >>READPAST <Correct>

Explanation:
When you use the READPAST table query hint, rows in the table locked by other transactions are skipped rather than blocking the transaction.

When you use the ROWLOCK table query hint, row locks will be used when page or table locks are likely to be used.

When you use the TABLOCK table query hint, a lock is applied at the table level rather than at the row or page level.

When you use the PAGLOCK table query hint, a lock is placed at the page level rather than at the row or table level.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Table Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187373.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

Question Number (ID) : 3 (461P_4.4_06)

You have noticed that a particular block of Transact-SQL code triggers Error 1105. You want to determine the error state number of that particular error. You configure the appropriate TRY . . . CATCH construct, using the code so that you can further diagnose the error.

Which of the following system functions can you use to determine this information?

1. >>ERROR_STATE() <Correct>
2. ERROR_LINE()
3. ERROR_NUMBER()
4. ERROR_PROCEDURE()
5. ERROR_MESSAGE()
6. ERROR_SEVERITY()

Explanation:
You can use the ERROR_STATE() system function to determine the error state number. An error state number differs from an error number. For example, the 1105 error can have a state of 0, 1, 2, or 3.

You can use the ERROR_LINE() system function to determine the line number at which the error occurred.

You can use the ERROR_MESSAGE() system function to determine the text of the message that is returned to the application.

You can use the ERROR_NUMBER() system function to determine the error number.

You can use the ERROR_PROCEDURE() system function to determine the name of the trigger of stored procedure in which the error occurred.

You can use the ERROR_SEVERITY() system function to determine the severity of the error.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Implement error handling.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 2

ERROR_STATE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180031.aspx

Retrieving Error Information in Transact-SQL
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179495(SQL.105).aspx

Question Number (ID) : 4 (461P_1.2_07)

A view that you have created for the Hovercraft rentals database currently returns information on the Hovercraft model and the renter name. You have been asked to modify the view so that it also returns the Hovercraft's registration data.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER SCHEMA

2. ALTER INDEX

3. >>ALTER VIEW <Correct>

4. ALTER TABLE

Explanation:
You use the ALTER VIEW statement to modify the properties of a view.

You use the ALTER TABLE statement to modify the properties of a table. You cannot use this statement to modify the properties of a view.

You use the ALTER SCHEMA statement to modify the properties of a schema. You cannot use this statement to modify the properties of a view.

You use the ALTER INDEX statement to modify the properties of an index. You cannot use this statement to modify the properties of a view.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

ALTER VIEW (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173846.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 5 (461P_3.3_01)

Your database has two tables related to product models, named Production.ProductModel and Production.NewProductModel. You want to view all products that are listed in both tables but do not want to view duplicate entries.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT ProductModelID, Name FROM Production.ProductModel INTERSECT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name

2. >>SELECT ProductModelID, Name FROM Production.ProductModel UNION SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name <Correct>

3. SELECT ProductModelID, Name FROM Production.NewProductModel EXCEPT SELECT ProductModelID, Name FROM Production.ProductModel ORDER BY Name

4. SELECT ProductModelID, Name FROM Production.ProductModel EXCEPT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name

5. SELECT ProductModelID, Name FROM Production.ProductModel UNION ALL SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name

Explanation:
You should use the SELECT ProductModelID, Name FROM Production.ProductModel UNION SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce a list of all products in both tables without duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel INTERSECT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce only those entries that are in the output of both queries. You want to produce a list of all products in both tables without duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel EXCEPT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce a list of products from the ProductModel table as long as those products do not have duplicate entries in the NewProductModel table. You want to produce a list of all products in both tables without duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel UNION ALL SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce a list of all products in both tables but will output duplicate entries. You want to produce a list of all products in both tables without duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Production.NewProductModel EXCEPT SELECT ProductModelID, Name FROM Production.ProductModel ORDER BY Name statement. This will produce a list of products from the NewProductModel table as long as those products do not have duplicate entries in the ProductModel table. You want to produce a list of all products in both tables without duplicate entries.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

Set Operators (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ff848745.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 6 (461P_3.3_06)

The Hovercraft database has three tables related to Hovercraft design, OldDesign, CurrentDesign, and FutureDesign. Some designs are present in more than one table. You want to generate a list of designs that are present in all three tables.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

2. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

3. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

4. >>SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign
<Correct>

Explanation:
You should use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in all three tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in the OldDesign and CurrentDesign table but not present in the FutureDesign table. You want to generate a list of designs that are present in all three tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the FutureDesign table as well as those that are present in both the OldDesign and CurrentDesign tables. You want to generate a list of designs that are present in all three tables, not all items in all tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the OldDesign table as well as those that are present in both the CurrentDesign and FutureDesign tables. You want to generate a list of designs that are present in all three tables, not all items in one table and those shared across the two others.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

EXCEPT and INTERSECT (Transact-SQL)
MSDN

Link: http://msdn.microsoft.com/en-us/library/ms188055.aspx

Question Number (ID) : 7 (461P_4.4_03)

An error occurs when you execute a block of Transact-SQL code, causing a transaction to roll back. You want to determine the error number associated with this error. You configure the appropriate TRY . . . CATCH construct, using the code so that you can further diagnose the error.

Which of the following system functions can you use to determine this information?

1. ERROR_STATE()
2. ERROR_PROCEDURE()
3. >>ERROR_NUMBER() <Correct>
4. ERROR_SEVERITY()
5. ERROR_LINE()
6. ERROR_MESSAGE()

Explanation:
You can use the ERROR_NUMBER() system function to determine the error number.

You can use the ERROR_LINE() system function to determine the line number at which the error occurred.

You can use the ERROR_MESSAGE() system function to determine the text of the message that is returned to the application.

You can use the ERROR_PROCEDURE() system function to determine the name of the trigger of stored procedure in which the error occurred.

You can use the ERROR_SEVERITY() system function to determine the severity of the error.

You can use the ERROR_STATE() system function to determine the error state number. An error state number differs from an error number. For example, the 1105 error can have a state of 0, 1, 2 or 3.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Implement error handling.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 2

ERROR_NUMBER (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175069.aspx

Retrieving Error Information in Transact-SQL
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179495(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 8 (461P_2.5_02)

You execute the following statement:

```
declare @tempDoc xml
set @tempDoc = '<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>'
SELECT @tempDoc.query('/Root/ProductDescription/Features');
```

Which of the following correctly represents the output of this query?

1. <Passengers>60 passenger capacity</Passengers>

2. <Maintenance>5 year parts and labor extended maintenance </Maintenance>

3.
```
<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>
```

4.
```
>><Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>                  <Correct>
```

Explanation:
The query SELECT @tempDoc.query('/Root/ProductDescription/Features') produces the result:

```
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
```

The query SELECT @tempDoc.query('/Root') produces the result:

```
<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers') produces the result:

<Passengers>60 passenger capacity</Passengers>

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance') produces the result:

<Maintenance>5 year parts and labor extended maintenance </Maintenance>

Objective:
Work with Data

Sub Objective(s):
Query and manage XML data.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 7 - Lesson 1

Query() Method (xml Data Type)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191474.aspx

XML Data (SQL Server)
MSDN
Link: http://msdn.microsoft.com/en-us/library/bb522446.aspx

Question Number (ID) : 9 (461P_1.2_01)

You want to rename the Jobseekers view associated with the HumanResources schema so that it is named Candidates.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER VIEW [HumanResources].[Jobseekers] Candidates

2. ALTER VIEW [HumanResources].[Candidates] Jobseekers

3. >>EXEC sp_rename 'HumanResources.Jobseekers', 'Candidates' <Correct>

4. EXEC sp_rename 'HumanResources.Candidates', 'Jobseekers'

Explanation:
You should use the EXEC sp_rename 'HumanResources.Jobseekers', 'Candidates' statement to rename the Jobseekers view associated with the HumanResources schema to Candidates.

You should not use the EXEC sp_rename 'HumanResources.Candidates', 'Jobseekers' statement. This statement renames the Candidates view (or table) associated with the HumanResources schema to Jobseekers.

You should not use the ALTER VIEW [HumanResources].[Jobseekers] Candidates statement. You cannot use the ALTER VIEW statement to change the name of a view.

You should not use the ALTER VIEW [HumanResources].[Candidates] Jobseekers statement. You cannot use the ALTER VIEW statement to change the name of a view.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

Rename Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187586.aspx

Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190174.aspx

Question Number (ID) : 10 (461P_3.1_14)

You want to create a stored procedure that returns an employee's email address when partial values for first name and last name are provided as input.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50) = N'%',
  @FirstName nvarchar(50) = N'%'
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

```
2.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50) = N'%',
  @FirstName nvarchar(50) = N'%'
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName LIKE @FirstName AND LastName LIKE @LastName;
```

```
3.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50) = N'%',
  @FirstName nvarchar(50) = N'%'
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

```
4.  >>CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50) = N'%',
  @FirstName nvarchar(50) = N'%'
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName LIKE @FirstName AND LastName LIKE @LastName;                <Correct>
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50) = N'%',
  @FirstName nvarchar(50) = N'%'
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName LIKE @FirstName AND LastName LIKE @LastName;
```

Executing this statement creates a stored procedure that returns an employee's email address when partial values for first name and last name are provided as input.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50) = N'%',
   @FirstName nvarchar(50) = N'%'
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName LIKE @FirstName AND LastName LIKE @LastName;
```

This stored procedure outputs an employee's phone number rather than the first name, last name, and email address. You want to create a stored procedure that returns an employee's email address when partial values for first name and last name are provided as input.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50) = N'%',
   @FirstName nvarchar(50) = N'%'
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This stored procedure works only if correct values are passed to the first name and last name parameters. You want to create a stored procedure that returns an employee's email address when partial values for first name and last name are provided as input.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50) = N'%',
   @FirstName nvarchar(50) = N'%'
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This stored procedure works only if correct values are passed to the first name and last name parameters. You want to create a stored procedure that returns an employee's email address when partial values for first name and last name are provided as input.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

Question Number (ID) : 11 (461P_1.4_10)

You have an existing unique constraint named UQ_Password on the PasswordHash column of the Password table located in the Person schema. You want to extend this unique constraint to include the PasswordSalt column. The table has an existing primary key constraint.

Which of the following statements would you use to accomplish this goal? (Each correct answer presents part of a complete solution. Choose two.)

1. ALTER TABLE Person.Password ADD CONSTRAINT UQ_Password CHECK (PasswordSalt,PasswordHash)

2. >>ALTER TABLE Person.Password DROP CONSTRAINT UQ_Password <Correct>

3. ALTER TABLE Person.Password ADD CONSTRAINT UQ_Password PRIMARY KEY (PasswordSalt,PasswordHash)

4. >>ALTER TABLE Person.Password ADD CONSTRAINT UQ_Password UNIQUE (PasswordSalt,PasswordHash) <Correct>

Explanation:
You should use the ALTER TABLE Person.Password DROP CONSTRAINT UQ_Password statement. To modify an existing unique constraint, you need to drop the original and then create a new constraint with the same name. This statement removes the original constraint.

You should use the ALTER TABLE Person.Password ADD CONSTRAINT UQ_Password UNIQUE (PasswordSalt,PasswordHash) statement. After you have dropped the original constraint, you can create a new unique constraint on the appropriate columns by executing this statement.

You should not use the ALTER TABLE Person.Password ADD CONSTRAINT UQ_Password PRIMARY KEY (PasswordSalt,PasswordHash) statement. This statement attempts to create a primary key constraint. The table has an existing primary key constraint.

You should not use the ALTER TABLE Person.Password ADD CONSTRAINT UQ_Password CHECK (PasswordSalt,PasswordHash) statement. This statement attempts to create a check constraint rather than a unique constraint.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Modify Unique Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191535.aspx

Unique Constraints and Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187550.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 12 (461P_3.4_08)

You want to generate a list of integers that includes the integer raised to the power of 5 where the integers have a value between 1 and 20.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
   SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Power';
   SET @testvalue = @testvalue +1;
 END;
```

```
2.  >>DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
   SELECT @testvalue as 'Value', Power(@testvalue,5) as 'Power';
   SET @testvalue = @testvalue +1;
 END;                        <Correct>
```

```
3.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
   SELECT @testvalue as 'Value', LOG(@testvalue) as 'Power';
   SET @testvalue = @testvalue +1;
 END;
```

```
4.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
   SELECT @testvalue as 'Value', EXP(@testvalue) as 'Power';
   SET @testvalue = @testvalue +1;
 END;
```

Explanation:
You should use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
   SELECT @testvalue as 'Value', Power(@testvalue,5) as 'Power';
   SET @testvalue = @testvalue +1;
 END;
```

This statement will generate a list of integers and the integer raised to the power of 5 where the integers have a value between one 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
   SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Power';
```

```
    SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and their squares between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Power';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and the natural exponent (the constant e raised to the power of the number) between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Power';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and their natural logarithms between 1 and 20.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

POWER (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174276.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 13 (461P_1.1_05)

You want to add a new column to an existing table in one of your organization's databases. This column should have the name Donut and use the nvarchar 200 type.

Which of the following Transact-SQL statements should you use to accomplish this goal?

1. >>ALTER TABLE <Correct>
2. ALTER DATABASE
3. ALTER SCHEMA
4. ALTER COLUMN

Explanation:
You use the ALTER TABLE statement to add a column to an existing table.

You use the ALTER COLUMN statement as part of the ALTER TABLE statement to modify an existing column. You cannot use this statement to add a new column to an existing table.

You use the ALTER DATABASE statement to modify a database. You cannot use this statement to add a new column to an existing table.

You use the ALTER SCHEMA statement to modify an existing schema. You cannot use this statement to add a new column to an existing table.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

ALTER TABLE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190273.aspx

ALTER Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/cc879314.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

Question Number (ID) : 14 (461P_2.4_13)

You want to generate a ranked list of pay rates for employees who are paid every two weeks by querying the HumanResources.EmployeePayHistory table of the AdventureWorks2012 database. If more than one employee has the same rate, they should all be assigned the same rank. The employee who has the rate that is next below this rate should be ranked from the beginning as shown in the exhibit between rows 7 and 8.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;            <Correct>
```

```
2.  >>SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
3.  SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
4.  SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

Explanation:
You should use the statement:

```
SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use DENSE_RANK, the statement ranks output, and the same results share the same rank; however, the results do not appear in the same way as they do with the RANK statement. The next result when you use DENSE_RANK sequentially follows the last RANK. For example, 1,2,2,2,3 rather than 1,2,2,2,5.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement outputs results using the RANK function. When you use the RANK function and multiple results share the same rank, those results will be assigned that rank. The next result will be assigned the rank based on how many rows there are after the first row. Hence, ranking results can be of the sequence 1,2,2,2,5 rather than 1,2,2,2,3.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

The NTILE() function partitions the results into the equal sections based on the number you use with the NTILE function. This query would partition the results into five ranks rather than the ranking order shown in the exhibit.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use the ROW_NUMBER function, ranks are assigned sequentially, even when the result is the same. Instead of 1,2,2,2,5, in which the second, third, and fourth results are the same, the result would be 1,2,3,4,5.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

DENSE_RANK (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173825.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 15 (461P_3.4_20)

You want to use the appropriate function to convert a binary value to the datetime2 data type. The conversion should return a null value if this is not possible.

Which of the following built-in functions can you use to accomplish this goal?

1. PARSE

2. TRY_PARSE

3. CONVERT

4. >>TRY_CONVERT <Correct>

Explanation:
TRY_CONVERT enables you to attempt to convert data stored in one data type to another. If this conversion fails, a NULL value is written.

The PARSE function enables you to change a string written in one cultural format (such as en-AU) to data types such as datetime and datetime2. If you do not use PARSE, strings that are not in the appropriate format will not convert.

TRY_PARSE enables you to change a string written in one cultural format (such as en-AU) to other data types such as datetime and datetime2. When you use TRY_PARSE, if the operation fails, the conversion occurs with the value being set to NULL.

CONVERT enables you to convert data stored in one data type to another. The conversion might fail if you attempt to convert between data types where conversion is not possible.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

TRY_CONVERT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh230993.aspx

Question Number (ID) : 16 (461P_1.4_06)

You have created a new table named Maintenance in the Hovercraft Rentals database. This table is located in the Chassis schema and has a HovercraftID column that is set to disallow null values. You want to create a relationship between the HovercraftID column in this new table and the HovercraftID column in the Inventory table, also located in the Chassis schema.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER TABLE Maintenance.Chassis ADD CONSTRAINT FK_HovercraftID PRIMARY (HovercraftID) REFERENCES Inventory.Chassis (HovercraftID)

2. ALTER TABLE Maintenance.Chassis ADD CONSTRAINT FK_HovercraftID FOREIGN KEY (HovercraftID) REFERENCES Inventory.Chassis (HovercraftID)

3. ALTER TABLE Chassis.Maintenance ADD CONSTRAINT FK_HovercraftID PRIMARY (HovercraftID) REFERENCES Chassis.Inventory (HovercraftID)

4. >>ALTER TABLE Chassis.Maintenance ADD CONSTRAINT FK_HovercraftID FOREIGN KEY (HovercraftID) REFERENCES Chassis.Inventory (HovercraftID) <Correct>

Explanation:
You should use the ALTER TABLE Chassis.Maintenance ADD CONSTRAINT FK_HovercraftID FOREIGN KEY (HovercraftID) REFERENCES Chassis.Inventory (HovercraftID) statement. This statement creates the correct foreign key constraint between the HovercraftID column in the Chassis.Maintenance table and the HovercraftID column in the Chassis.Inventory table.

You should not use the ALTER TABLE Maintenance.Chassis ADD CONSTRAINT FK_HovercraftID FOREIGN KEY (HovercraftID) REFERENCES Inventory.Chassis (HovercraftID) statement. This statement references tables in the Maintenance and Inventory schema.

You should not use the ALTER TABLE Chassis.Maintenance ADD CONSTRAINT FK_HovercraftID PRIMARY (HovercraftID) REFERENCES Chassis.Inventory (HovercraftID) statement. This statement attempts to create a primary key.

You should not use the ALTER TABLE Maintenance.Chassis ADD CONSTRAINT FK_HovercraftID PRIMARY (HovercraftID) REFERENCES Inventory.Chassis (HovercraftID) statement. This statement attempts to create a primary key.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Create Foreign Key Relationships
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189049.aspx

Primary and Foreign Key Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179610.aspx

Question Number (ID) : 17 (461P_3.2_08)

You run the following query against a table in the Hovercraft database:

```
SELECT *
FROM Hovercraft.Sales
WHERE orderid = 3150;
```

You get the output shown in the exhibit

The discount on the product with unitid 234 is 12.5%. You want to update this table so that you decrease the discount assigned to each item in this order by 0.5%. When you do this, the new discount applied to unitid 234 for this item should be 12.0%.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>UPDATE Hovercraft.Sales SET discount -= 0.05 WHERE orderid = 3150 <Correct>

2. UPDATE Hovercraft.Sales SET discount *= 0.05 WHERE orderid = 3150

3. UPDATE Hovercraft.Sales SET discount /= 0.05 WHERE orderid = 3150

4. UPDATE Hovercraft.Sales SET discount += 0.05 WHERE orderid = 3150

Explanation:
You should use the UPDATE Hovercraft.Sales SET discount -= 0.05 WHERE orderid = 3150 statement. This statement will decrease the discount by 0.05 where the orderid = 3150.

You should not use the UPDATE Hovercraft.Sales SET discount += 0.05 WHERE orderid = 3150 statement. This statement will increase rather than decrease the discount by 5%.

You should not use the UPDATE Hovercraft.Sales SET discount *= 0.05 WHERE orderid = 3150 statement. This statement will multiply the existing discount by 5% rather than decreasing it by 5%.

You should not use the UPDATE Hovercraft.Sales SET discount /= 0.05 WHERE orderid = 3150 statement. This statement will divide the existing discount by 5% rather than decreasing it by 5%.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

UPDATE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177523.aspx

WHERE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188047.aspx

Question Number (ID) : 18 (461P_1.2_06)

You are creating several views for a Hovercraft rental database. You need to ensure that data within the Hovercraft rental database is modified in authorized ways only. When creating the view, you specify the limits to the ways that data can be modified. You want to ensure that all statements that modify data executed against the view follow these criteria.

Which of the following options would you use when creating this view?

1. ENCRYPTION
2. >>SCHEMABINDING
3. VIEW_METADATA
4. CHECK OPTION <Correct>

Explanation:
You would use CHECK OPTION. This option forces all statements that modify data executed against the view to follow criteria set within the statement that defines the view.

You would not use ENCRYPTION. This option encrypts the text of the CREATE VIEW statement. A side effect of using the ENCRYPTION option is that the view cannot be published as part of SQL Server replication.

You would not use SCHEMABINDING. This option ensures that the base table or tables cannot be modified in any way that would affect the view definition.

You would not use VIEW_METADATA. When you use this option, browse mode metadata will return the view name and not the underlying table names when it lists columns from the view in the result set.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187956.aspx

Question Number (ID) : 19 (461P_3.4_04)

You want to generate a list of integers and their natural logarithms between 1 and 20.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
2.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
3.  >>DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;                      <Correct>
```

```
4.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

Explanation:
You should use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and their natural logarithms between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and their squares between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
    SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
    SET @testvalue = @testvalue +1;
 END;
```

This statement outputs a list of integers and the natural exponent (the constant e raised to the power of the number) between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
  BEGIN
    SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
    SET @testvalue = @testvalue +1;
 END;
```

This statement outputs a list of integers and their square roots between 1 and 20.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

LOG (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190319.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 20 (461P_2.4_04)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued sick leave hours. You want to determine the maximum sick leave hours accrued by any single employee who has a job title like Marketing Manager.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1. >>SELECT MAX(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';          <Correct>
```

```
2. SELECT AVG(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

```
3. SELECT MIN(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

```
4. SELECT VAR(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

Explanation:
You should use the statement:

```
SELECT MAX(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the maximum value for sick leave hours for employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT MIN(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the minimum value for sick leave hours for employees who have the job title Marketing Manager. You want to find the maximum value for sick leave hours for employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT AVG(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the average value for SickLeave hours for employees with the job title Marketing Manager. You want to find the maximum value for SickLeave hours for employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT VAR(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the statistical variance in SickLeave hours for employees with the job title Marketing Manager. You want to find the maximum value for SickLeave hours for employees who have the job title Marketing Manager.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

MAX (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187751.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 21 (461P_2.4_07)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued vacation hours. You want to determine the total number of vacation hours belonging to employees who have ta job title like Design Engineer.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT AVG(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
2.  SELECT MAX(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
3.  SELECT COUNT(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
4.  >>SELECT SUM(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';            <Correct>
```

Explanation:
You should use the statement:

```
SELECT SUM(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement determines the total sum of vacation hours of all employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT COUNT(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This query finds the number of employees with the title Design Engineer who have values set for vacation hours other than NULL. You want to determine the total sum of vacation hours of all employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT AVG(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This query finds the average number of vacation hours for employees who have the job title Design Engineer. You want to determine the total sum of vacation hours of all employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT MAX(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This query finds the maximum number of vacation hours that any employee who has the job title Design Engineer has. You want to determine the total sum of vacation hours of all employees who have the job title Design Engineer.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

SUM (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187810.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 22 (461P_1.3_10)

You want to update Kevin Brown's job title from Marketing Assistant to Marketing Specialist by using an UPDATE statement against the HumanResources.vEmployee view.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Assistant'
WHERE LastName = N'Brown' AND FirstName = 'Kevin';
```

```
2.  >>UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Specialist'
WHERE LastName = N'Brown' AND FirstName = 'Kevin';                    <Correct>
```

```
3.  UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Assistant'
WHERE LastName = N'Kevin' AND FirstName = 'Brown';
```

```
4.  UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Specialist'
WHERE LastName = N'Kevin' AND FirstName = 'Brown';
```

Explanation:

You should use the statement:

```
UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Specialist'
WHERE LastName = N'Brown' AND FirstName = 'Kevin';
```

This statement uses the HumanResources.vEmployee view to update Kevin Brown's job title to Marketing Specialist.

You should not use the statement:

```
UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Assistant'
WHERE LastName = N'Brown' AND FirstName = 'Kevin';
```

This statement uses the HumanResources.vEmployee view to update Kevin Brown's job title to Marketing Assistant. You want to update it to Marketing Specialist.

You should not use the statement:

```
UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Specialist'
WHERE LastName = N'Kevin' AND FirstName = 'Brown';
```

This statement updates Brown Kevin's job title to Marketing Specialist, but this statement incorrectly uses Kevin as the last name and Brown as the first name.

You should not use the statement:

```
UPDATE HumanResources.vEmployee
SET JobTitle = N'Marketing Assistant'
WHERE LastName = N'Kevin' AND FirstName = 'Brown';
```

This statement updates Brown Kevin's job title to Marketing Assistant. You want to update Kevin Brown's job title to Marketing Specialist. Also, this statement incorrectly uses Kevin as the last name and Brown as the first name.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 23 (461P_2.4_20)

You want to generate a ranked list of employees by accrued sick leave hours by querying the HumanResources.Employee table of the AdventureWorks2012 database. The ranking should be from most accrued hours to least. If more than one employee has the same number of hours, they should be assigned the same rank. Ranking should be of the format 1,2,2,2,5 rather than of 1,2,2,2,3.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. SELECT BusinessEntityID, SickLeaveHours, RANK() OVER (ORDER BY SickLeaveHours ASC)
FROM Adventureworks2012.HumanResources.Employee;

2. SELECT BusinessEntityID, SickLeaveHours, DENSE_RANK() OVER (ORDER BY SickLeaveHours DESC)
FROM Adventureworks2012.HumanResources.Employee;

3. >>SELECT BusinessEntityID, SickLeaveHours, RANK() OVER (ORDER BY SickLeaveHours DESC)
FROM Adventureworks2012.HumanResources.Employee; <Correct>

4. SELECT BusinessEntityID, SickLeaveHours, DENSE_RANK() OVER (ORDER BY SickLeaveHours ASC)
FROM Adventureworks2012.HumanResources.Employee;

Explanation:
You should use the statement:

SELECT BusinessEntityID, SickLeaveHours, RANK() OVER (ORDER BY SickLeaveHours DESC)
FROM Adventureworks2012.HumanResources.Employee;

This statement ranks employees by the number of sick leave hours accrued, from most hours accrued to least. Because this query uses the RANK function, ranking will be of the format 1,2,2,2,5 rather than of 1,2,2,2,3 if more than one employee has the same number of accrued hours.

You should not use the statement:

SELECT BusinessEntityID, SickLeaveHours, DENSE_RANK() OVER (ORDER BY SickLeaveHours DESC)
FROM Adventureworks2012.HumanResources.Employee;

This statement ranks employees by the number of sick leave hours accrued, from most hours accrued to least. However, because this query uses the DENSE_RANK function, ranking is of the format 1,2,2,2,3 rather than of 1,2,2,2,5 if more than one employee has the same number of accrued hours. The RANK function provides the desired ranking.

You should not use the statement:

SELECT BusinessEntityID, SickLeaveHours, RANK() OVER (ORDER BY SickLeaveHours ASC)
FROM Adventureworks2012.HumanResources.Employee;

This statement ranks employee sick leave hours from least to most rather than from most to least.

You should not use the statement:

SELECT BusinessEntityID, SickLeaveHours, DENSE_RANK() OVER (ORDER BY SickLeaveHours ASC)
FROM Adventureworks2012.HumanResources.Employee;

This statement ranks employee sick leave hours from least to most rather than from most to least.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

RANK (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176102.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 24 (461P_2.2_05)

You want to generate a list of Hovercraft names and prices for Hovercraft built in Tonga by querying the Hovercraft.Products and Hovercraft.Suppliers tables in the Hovercraft database.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Fiji');

2. Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country N'Fiji');

3. Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Tonga');

4. >>Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country = N'Tonga'); <Correct>

Explanation:
You should use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country = N'Tonga'); statement. This query will output a list of Hovercraft names and prices for Hovercraft suppliers in Tonga.

You should not use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Tonga'); statement. This query will output a list of Hovercraft names and prices for all Hovercraft suppliers except Tonga. You want to output a list of Hovercraft names and prices for Hovercraft suppliers in Tonga.

You should not use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Fiji'); statement. This query will output a list of Hovercraft names and prices for Hovercraft suppliers for all suppliers not based in Fiji. You want to output a list of Hovercraft names and prices for Hovercraft suppliers in Tonga.

You should not use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country N'Fiji'); statement. This query will output a list of Hovercraft names and prices for Hovercraft suppliers for all suppliers based in Fiji. You want to output a list of Hovercraft names and prices for Hovercraft suppliers in Tonga.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

Question Number (ID) : 25 (461P_3.2_01)

You are adding tables to the Hovercraft database. You create a table by using the following statement:

```
Create Table Hovercraft.Testing
(
TestIdentifier uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY,
HullId tinyint NOT NULL,
TestDate date NOT NULL,
Registration nvarchar(8) NOT NULL,
Notes nvarchar(max)
)
```

Which of the following statements can you use to insert data into this table? (Each correct answer presents a complete solution. Choose two.)

1. >>INSERT INTO Hovercraft.Testing (HullID, TestDate, Registration, Notes) VALUES (124, '20120722', N'AAA321', N'Hovercraft in good condition. Next service in six months' time') <Correct>

2. INSERT INTO Hovercraft.Testing (HullID, TestDate, Registration, Notes) VALUES (247, '20120722', N'ABCD54321', N'Hovercraft in good condition. Next service in three months' time')

3. >>INSERT INTO Hovercraft.Testing (Notes, HullID, TestDate, Registration,) VALUES (N'Hovercraft requires maintenance.', 142, '20120715', N'AAA320') <Correct>

4. INSERT INTO Hovercraft.Testing (Notes, HullID, TestDate, Registration,) VALUES (N'Hovercraft requires maintenance.', 842, '20120709', N'AAB121')

Explanation:
You can use the INSERT INTO Hovercraft.Testing (HullID, TestDate, Registration, Notes) VALUES (124, '20120722', N'AAA321', N'Hovercraft in good condition. Next service in six months' time') statement. This statement correctly inserts the appropriate data into columns that use the defined data type.

You can use the INSERT INTO Hovercraft.Testing (Notes, HullID, TestDate, Registration,) VALUES (N'Hovercraft requires maintenance.', 142, '20120715', N'AAA320') statement. This statement correctly inserts the appropriate data into columns that use the defined data type.

You should not use the INSERT INTO Hovercraft.Testing (Notes, HullID, TestDate, Registration,) VALUES (N'Hovercraft requires maintenance.', 842, '20120709', N'AAB121') statement. This statement does not insert the appropriate data into columns that use the defined data type. The HullID column uses tinyint, which is restricted to integer values of 255 or less. This statement attempts to insert the value of 842.

You should not use the INSERT INTO Hovercraft.Testing (HullID, TestDate, Registration, Notes) VALUES (247, '20120722', N'ABCD54321', N'Hovercraft in good condition. Next service in three months' time') statement. This statement does not insert the appropriate data into columns that use the defined data type. This statement attempts to insert the nine-character ABCD54321 entry in the column that has the nvarchar(8) type.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

INSERT (Transact-SQL)

MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174335.aspx

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 26 (461P_2.2_06)

You want to generate a list of Hovercraft names and prices for Hovercraft built by suppliers in all countries except Singapore by querying the Hovercraft.Products and Hovercraft.Suppliers tables in the Hovercraft database.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country = N'Singapore');

2. Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country N'Tonga');

3. >>Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Singapore'); <Correct>

4. Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Tonga');

Explanation:
You should use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Singapore'); statement. This statement will output the list of Hovercraft names and prices not supplied from Singapore.

You should not use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country = N'Singapore'); statement. This statement will generate a list of Hovercraft names and prices supplied from Singapore. You want the list of Hovercraft names and prices not supplied from Singapore.

You should not use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE NOT country = N'Tonga'); statement. This statement will generate a list of Hovercraft names and prices not supplied from Tonga. You want the list of Hovercraft names and prices not supplied from Singapore.

You should not use the Select HovercraftName, HovercraftPrice from Hovercraft.Products WHERE supplierID IN (SELECT supplierID FROM Hovercraft.Suppliers WHERE country N'Tonga'); statement. This statement will generate a list of Hovercraft names and prices supplied from Tonga.You want the list of Hovercraft names and prices not supplied from Singapore.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

Question Number (ID) : 27 (461P_4.3_02)

Which of the following Transact-SQL statements do you use if you want to release the current result set from a cursor?

1. >>CLOSE <Correct>

2. DEALLOCATE

3. OPEN

4. FETCH

Explanation:
You use the CLOSE statement to release the current result set. This releases cursor locks on rows on which the cursor is positioned.

You use the DEALLOCATE statement to remove a cursor reference.

You use the OPEN statement to open a Transact-SQL cursor. The cursor is populated by the results of the Transact-SQL statement used when the cursor is created.

You use the FETCH statement to retrieve a specific row from a CURSOR.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Evaluate the use of row-based operations vs. set-based operations.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 16 - Lesson 1

CLOSE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175035.aspx

Cursors (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181441.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 28 (461P_3.1_09)

You want to create a stored procedure that lists the maximum and minimum list prices for each product model ID by querying the Production.Product table of the AdventureWorks database. This stored procedure should be associated with the production schema.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, AVG(Listprice), STDEV(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP  BY  ProductModelID;
```

```
2.  CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, AVG(Listprice), MAX(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP  BY  ProductModelID;
```

```
3.  CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MIN(Listprice), AVG(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
    GROUP  BY  ProductModelID;
```

```
4.  >>CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MIN(Listprice), MAX(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP  BY  ProductModelID;        <Correct>
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MIN(Listprice), MAX(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP BY ProductModelID;
```

This statement creates a stored procedure that, when executed, lists the maximum and minimum list prices for each product model ID by querying the Production.Product table of the AdventureWorks database.

You should not use the statement:

```
CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MIN(Listprice), AVG(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
    GROUP BY ProductModelID;
```

This statement creates a stored procedure that outputs a list of minimum and average prices for each product model by querying the Production.Product database. You want to create a stored procedure that, when executed, lists the maximum and minimum list prices for each product model ID by querying the Production.Product table of the AdventureWorks database.

You should not use the statement:

```
CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, AVG(Listprice), MAX(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP BY ProductModelID;
```

This statement creates a stored procedure that outputs a list of average and maximum prices for each product model by querying the Production.Product database. You want to create a stored procedure that, when executed, lists the maximum and minimum list prices for each product model ID by querying the Production.Product table of the AdventureWorks database.

You should not use the statement:

```
CREATE PROCEDURE Production.uspProductMaxMin
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, AVG(Listprice), STDEV(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP BY ProductModelID;
```

This statement creates a stored procedure that outputs the average product model price and the standard deviation for a product model's price. You want to create a stored procedure that, when executed, lists the maximum and minimum list prices for each product model ID by querying the Production.Product table of the AdventureWorks database.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 29 (461P_1.5_10)

You want to view the definition of a DML trigger named Hovercraft.RentalTrigger.

Which of the following Transact-SQL statements can you use to accomplish this goal? (Each correct answer presents a complete solution. Choose two.)

1. EXEC sp_rename 'Hovercraft.RentalTrigger'

2. >>SELECT definition from sys.sql_modules where object_id = OBJECT_ID(N'Hovercraft.RentalTrigger') <Correct>

3. >>EXEC sp_helptext 'Hovercraft.RentalTrigger' <Correct>

4. SELECT object_id from sys.sql_modules where object_id=OBJECT_ID(N'Hovercraft.RentalTrigger')

Explanation:
Executing the SELECT definition from sys.sql_modules where object_id = OBJECT_ID(N'Hovercraft.RentalTrigger') statement will provide the definition of the Hovercraft.RentalTrigger trigger.

Executing the EXEC sp_helptext 'Hovercraft.RentalTrigger' statement will provide the definition of the Hovercraft.RentalTrigger trigger.

You should not execute the EXEC sp_rename 'Hovercraft.RentalTrigger' statement. This statement will not provide you with the definition of the Hovercraft.RentalTrigger trigger. This statement can be used to rename tables, indexes, columns, alias data types or .NET Framework CLS user defined types. It cannot be used to rename triggers.

You should not execute the SELECT object_id from sys.sql_modules where object_id=OBJECT_ID(N'Hovercraft.RentalTrigger') statement. This will return the objectID of the trigger but not the trigger definition.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Get Information about DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179309.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Question Number (ID) : 30 (461P_3.1_16)

You want to create a stored procedure that outputs an employee's first name, last name, phone number, and email address. You want to ensure that the definition of this procedure is not available to anyone who can query the system catalog.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE HumanResources.uspNamePhoneEmail
WITH ENCRYPTION
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress, PhoneNumber
   FROM  HumanResources.vEmployee;                    <Correct>
```

```
2.  >>CREATE PROCEDURE HumanResources.uspNamePhoneEmail
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress, PhoneNumber
   FROM  HumanResources.vEmployee;
```

```
3.  CREATE PROCEDURE HumanResources.uspNamePhoneEmail
WITH ENCRYPTION
AS
   SET NOCOUNT ON;
   SELECT MiddleName, LastName, City, PhoneNumber
   FROM  HumanResources.vEmployee;
```

```
4.  CREATE PROCEDURE HumanResources.uspNamePhoneEmail
AS
   SET NOCOUNT ON;
   SELECT MiddleName, LastName, City, PhoneNumber
   FROM  HumanResources.vEmployee;
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspNamePhoneEmail
WITH ENCRYPTION
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress, PhoneNumber
   FROM HumanResources.vEmployee;
```

Executing this statement creates a stored procedure that outputs employee first name, last name, email address, and phone number and has a definition that cannot be extracted by querying the system catalog. Using the WITH ENCRYPTION option enables you to obfuscate the stored procedure's definition.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspNamePhoneEmail
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress, PhoneNumber
   FROM HumanResources.vEmployee;
```

You want to create a stored procedure that outputs employee first name, last name, email address, and phone number that also has a definition that cannot be extracted by querying the system catalog. The definition for this stored procedure will be visible to other users who can query the system catalog.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspNamePhoneEmail
WITH ENCRYPTION
AS
   SET NOCOUNT ON;
   SELECT MiddleName, LastName, City, PhoneNumber
   FROM HumanResources.vEmployee;
```

The stored procedure created with this statement outputs an employee's middle name, last name, city, and phone number. You want to create a stored procedure that outputs employee first name, last name, email address, and phone number.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspNamePhoneEmail
AS
   SET NOCOUNT ON;
   SELECT MiddleName, LastName, City, PhoneNumber
   FROM HumanResources.vEmployee;
```

The stored procedure created with this statement outputs an employee's middle name, last name, city, and phone number. You want to create a stored procedure that outputs employee first name, last name, email address, and phone number and has a definition that cannot be extracted by querying the system catalog.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

Question Number (ID) : 31 (461P_2.4_02)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued sick leave hours. You want to determine the average sick leave hours accrued by employees who have a job title like Marketing Manager.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1. SELECT MAX(SickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager %';

2. SELECT VAR(SickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager %';

3. >>SELECT AVG(SickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';          <Correct>

4. SELECT MIN(VSickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager %';
```

Explanation:
You should use the statement:

```
SELECT AVG(SickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager %';
```

This statement returns the average sick leave hours assigned to employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT MIN(SickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager %';
```

This statement returns the minimum sick leave hours assigned to any employee who has the job title Marketing Manager. You want to have the statement return the average sick leave hours assigned to employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT MAX(SickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager %';
```

This statement returns the maximum sick leave hours assigned to any employee who has the job title Marketing Manager. You want to have the statement return the average sick leave hours assigned to employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT VAR(SickLeaveHours) AS 'Average Sick Leave Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager %';
```

This statement returns the statistical variance across sick leave hours assigned to employees who have the job title Marketing Manager. You want to have the statement return the sick leave vacation hours assigned to employees who have the job title Marketing Manager.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

AVG (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177677.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 32 (461P_2.4_14)

By querying the HumanResources.EmployeePayHistory table of the AdventureWorks2012 database, you want to generate a ranked list of pay rates for employees who are paid every two weeks. Employees should be ranked sequentially even if they have the same rate.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
2.  >>SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;            <Correct>
```

```
3.  SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
4.  SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

Explanation:
You should use the statement:

```
SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use the ROW_NUMBER function, ranks are assigned sequentially, even when the result is the same. Instead of 1,2,2,2,5 where the second, third, and fourth result are the same, the result would be 1,2,3,4,5.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement will output results using the RANK function. When you use the RANK function and multiple results share the same rank, those results will be assigned that rank. The next result will be assigned the rank based on how many rows there are after the first row. Hence, ranking results can be of the sequence 1,2,2,2,5 rather than 1,2,2,2,3.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use DENSE_RANK, the statement ranks output, and the same results share the same rank; however, the results do not appear in the same way as they do with the RANK statement. The next result when you use DENSE_RANK sequentially follows the last RANK, for example, 1,2,2,2,3 rather than 1,2,2,2,5.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

The NTILE() function partitions the results into equal sections based on the number you use with the NTILE function. This query would partition the results into five ranks.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

ROW_NUMBER (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms186734.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 33 (461P_3.1_08)

You want to create a stored procedure that returns the average list price for each product model ID stored in the Production.Product table.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MAX(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP  BY  ProductModelID;
```

```
2.  >>CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, AVG(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP  BY  ProductModelID;        <Correct>
```

```
3.  CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, STDEV(Listprice) FROM Production.Product
  WHERE ProductModelID IS NOT NULL
  GROUP  BY  ProductModelID;
```

```
4.  CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MIN(Listprice) FROM Production.Product
   WHERE ProductModelID IS NOT NULL
    GROUP  BY  ProductModelID;
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, AVG(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP BY ProductModelID;
```

This statement creates a stored procedure that returns the average list price for each product model ID stored in the Production.Product table.

You should not use the statement:

```
CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MAX(Listprice)
   FROM Production.Product
   WHERE ProductModelID IS NOT NULL
   GROUP BY ProductModelID;
```

This statement creates a stored procedure that generates a list of maximum prices for each product model ID from the Production.Product table. You want to create a stored procedure that returns the average list price for each product model ID stored in the Production.Product table.

You should not use the statement:

```
CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, MIN(Listprice) FROM Production.Product
   WHERE ProductModelID IS NOT NULL
    GROUP BY ProductModelID;
```

This statement creates a stored procedure that generates a list of minimum prices for specific product model IDs from the Production.Product table. You want to create a stored procedure that returns the average list price for each product model ID stored in the Production.Product table.

You should not use the statement:

```
CREATE PROCEDURE Production.uspProductAvgData
AS
   SET NOCOUNT ON;
   SELECT ProductModelID, STDEV(Listprice) FROM Production.Product
  WHERE ProductModelID IS NOT NULL
  GROUP BY ProductModelID;
```

This statement creates a stored procedure that generates a list of the standard deviation of prices listed for a specific product model ID from the Production.Product table. You want to create a stored procedure that returns the average list price for each product model ID stored in the Production.Product table.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 34 (461P_4.3_01)

Which of the following Transact-SQL statements do you use if you want to remove a cursor reference?

1. >>DEALLOCATE <Correct>
2. FETCH
3. OPEN
4. CLOSE

Explanation:
You use the DEALLOCATE statement to remove a cursor reference.

You use the OPEN statement to open a Transact-SQL cursor. The cursor is populated by the results of the Transact-SQL statement used when the cursor is created.

You use the FETCH statement to retrieve a specific row from a CURSOR.

You use the CLOSE statement to release the current result set. This releases cursor locks on rows on which the cursor is positioned.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Evaluate the use of row-based operations vs. set-based operations.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 16 - Lesson 1

DEALLOCATE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188782.aspx

Cursors (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181441.aspx

Question Number (ID) : 35 (461P_3.1_17)

You want to create a stored procedure that returns an employee's first name, last name, and email address when executed given the employee's first and last names. The stored procedure should recompile each time it is executed.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
WITH RECOMPILE
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;    <Correct>
```

```
2.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
WITH RECOMPILE
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @LastName AND LastName = @FirstName;
```

```
3.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
WITH ENCRYPTION
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

```
4.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
WITH ENCRYPTION
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @LastName AND LastName = @FirstName;
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
WITH RECOMPILE
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement creates a stored procedure that returns an employee's first name, last name, and email address when executed given the employee's first and last names and recompiles each time it is executed.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
WITH ENCRYPTION
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement creates a stored procedure that returns an employee's first name, last name, and email address when executed given the employee' first and last names. It will not recompile each time it is executed. The definition will be obfuscated.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
WITH RECOMPILE
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @LastName AND LastName = @FirstName;
```

This statement creates a stored procedure that does not work properly because in the last line, the LastName parameter is incorrectly mapped to FirstName, and the FirstName parameter is incorrectly mapped to LastName.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
WITH ENCRYPTION
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @LastName AND LastName = @FirstName;
```

This statement creates a stored procedure that does not work properly because in the last line, the LastName parameter is incorrectly mapped to FirstName, and the FirstName parameter is incorrectly mapped to LastName. This stored procedure will also not recompile each time it is executed.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)

MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 36 (461P_1.4_01)

You want to add a primary key constraint to the existing Rentals table in the Hovercraft schema of your company's database.

Which of the following statements would you use to accomplish this goal?

1. CREATE TABLE
2. ALTER COLUMN
3. ALTER SCHEMA
4. >>ALTER TABLE <Correct>

Explanation:
You use the ALTER TABLE statement with the ADD CONSTRAINT and PRIMARY KEY keywords to add a primary key constraint to an existing table.

You use the CREATE TABLE statement to create a table. Although it is possible to add a primary key constraint to a table during table creation, in this question, you are asked which statement you use to add a primary key constraint to an existing table.

You use the ALTER SCHEMA statement to modify an existing schema. You cannot use this statement to add a constraint to a table.

You use the ALTER COLUMN statement with the ALTER TABLE statement to modify the properties of a column. You cannot use this statement to add a primary key constraint to an existing table.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Primary and Foreign Key Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179610.aspx

Create Primary Keys
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189039.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 37 (461P_4.2_10)

You are configuring an appropriate transaction isolation level for a session. You want to ensure that statements in the transaction cannot read data altered by other transactions if those transactions have not yet committed. Data selected during the transaction should be able to be modified during the current transaction by update statements issued later in the transaction. Other transactions should be able to insert data into rows with key values that would fall into the range read by statements in the current transaction.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
2. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
3. >>SET TRANSACTION ISOLATION LEVEL READ COMMITTED <Correct>
4. SET TRANSACTION ISOLATION LEVEL SNAPSHOT

Explanation:
You use the SET TRANSACTION ISOLATION LEVEL READ COMMITTED statement when you want to ensure that statements cannot read data altered by other transactions when those transactions have not yet been committed. When you set the READ COMMITTED transaction level, data can be altered by other transactions between individual statements within the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED statement when you want to allow statements to read rows altered by other transactions before those transactions have been committed.

You use the SET TRANSACTION ISOLATION LEVEL SNAPSHOT statement when you want to ensure that data read by any statement in a transaction will be transactionally consistent with how that data existed at the start of the transaction. SNAPSHOT transactions do not block other transactions from writing data.

You use the SET TRANSACTION ISOLATION LEVEL SERIALIZABLE statement when you want to ensure that Transact-SQL statements cannot read data that has been altered but not committed, that other transactions cannot modify data read by the current transaction until the transaction commits, and other transactions cannot insert rows with key values that would fall into the range read by statements in the current transaction until that transaction completes.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SET TRANSACTION ISOLATION LEVEL (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173763.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

Question Number (ID) : 38 (461P_3.4_12)

Which of the following queries that use string functions will return the result 'ailartsua'?

```
1.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  RIGHT(@testvalue,  3);
```

```
2.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  LEFT(@testvalue,  3);
```

```
3.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  UPPER(@testvalue);
```

```
4.  >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  REVERSE(@testvalue);    <Correct>
```

Explanation:
The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REVERSE(@testvalue)
```

will return the result 'ailartsua' because it will reverse all the characters from the string 'australia'.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEFT(@testvalue, 3)
```

will return the result 'aus' because it will return the first three characters from the left of the string 'australia'.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT UPPER(@testvalue)
```

will return the result 'AUSTRALIA' because it will convert all characters from the string 'australia' to uppercase.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RIGHT(@testvalue, 3)
```

will return the result 'lia' because it will return the three characters from the string from the right.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

REVERSE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180040.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 39 (461P_3.1_12)

You want to create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and phone number as output.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;    <Correct>
```

```
2.  CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName = @LastName AND LastName = @FirstName;
```

```
3.  CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @LastName AND LastName = @FirstName;
```

```
4.  CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement creates a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and phone number as output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This stored procedure returns the employee's email address rather than phone number. You want to create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and phone number as output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName = @LastName AND LastName = @FirstName;
```

The last line of this statement maps the @LastName parameter with FirstName and the @FirstName parameter with LastName. This stored procedure will not function properly.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeePhone
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @LastName AND LastName = @FirstName;
```

The last line of this statement maps the @LastName parameter with FirstName and the @FirstName parameter with LastName. This stored procedure will not function properly. Even if this problem did not exist, the stored procedure would output email address rather than phone number.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 40 (461P_4.1_01)

You are optimizing several complex queries. You have determined that you can improve performance by concatenating UNION sets.

Which of the following query hints should you use to implement this optimization?

1. >>CONCAT UNION <Correct>
2. HASH UNION
3. MERGE UNION
4. HASH JOIN

Explanation:
The CONCAT UNION query hint option specifies that all UNION operations will be performed by concatenating UNION sets.

The MERGE UNION query hint option specifies that all UNION operations will be performed by merging UNION sets.

The HASH UNION query hint option specifies that all UNION operations will be performed by hashing UNION sets.

The HASH JOIN query hint option specifies that all JOIN operations in the query use HASH JOINs.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Query Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181714.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 41 (461P_3.4_09)

Which of the following queries that use string functions will return the result 'aus'?

```
1.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  REVERSE(@testvalue);
```

```
2.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  UPPER(@testvalue);
```

```
3.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  RIGHT(@testvalue,  3);
```

```
4.  >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  LEFT(@testvalue,  3);        <Correct>
```

Explanation:
The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEFT(@testvalue, 3);
```

will return the result 'aus' because it will return the first three characters from the left of the string 'australia'.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT UPPER(@testvalue);
```

will return the result 'AUSTRALIA' because it will convert all characters from the string 'australia' to uppercase.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REVERSE(@testvalue);
```

will return the result 'ailartsua' because it will reverse all the characters from the string 'australia'.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RIGHT(@testvalue, 3);
```

will return the result 'lia' because it will return the three characters from the string from the right.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

LEFT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177601.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 42 (461P_1.3_01)

The Sales table of the Hovercraft database has the following columns:

CustomerID
ContactName
Address
Email
Phone
OrderNumber
Country

Your company has offices in 20 countries. You want to simplify the process of querying data so that sales people in Australia are returned results from the Sales table only where the value in the Country column is Australia. Sales people in Australia must be able to query data in all other columns.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>CREATE VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
WHERE  Country  =  N'Australia';        <Correct>
```

```
2.  ALTER VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
WHERE  Country  NOT  IN(N'Australia');
```

```
3.  CREATE VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
WHERE  Country  NOT  IN(N'Australia');
```

```
4.  ALTER VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
WHERE  Country  =  N'Australia';
```

Explanation:
You should use the statement:

```
CREATE VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
WHERE Country = N'Australia';
```

This statement creates a view that sales people in Australia can query. It only returns results from the Sales table where Country is set to Australia.

You should not use the statement:

```
CREATE VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
```

```
WHERE Country NOT IN(N'Australia');
```

This statement creates a view that lists all data in the dbo.Sales table except those where the country is set to Australia.

You should not use the statement:

```
ALTER VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
WHERE Country = N'Australia';
```

You use the ALTER VIEW statement to modify an existing view. You cannot use this statement to create a new view.

You should not use the statement:

```
ALTER VIEW dbo.vAustraliaCustomers
AS
   SELECT CustomerID, ContactName, Address, Email, Phone, OrderNumber, Country
FROM dbo.Sales
WHERE Country NOT IN(N'Australia');
```

You use the ALTER VIEW statement to modify an existing view. You cannot use this statement to create a new view.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

ALTER VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms173846.aspx

Question Number (ID) : 43 (461P_4.2_09)

You are working on some errors raised by a complex set of transactions. You want to ensure that transactions are not rolled back when low-severity errors are raised by the database engine.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. SET XACT_ABORT ON
2. SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
3. >>SET XACT_ABORT OFF <Correct>
4. SET TRANSACTION ISOLATION LEVEL READ COMMITTED

Explanation:
You use the SET XACT_ABORT OFF statement to allow a Transact-SQL statement to cause low-severity errors without rolling back the entire transaction.

You use the SET XACT_ABORT ON statement to ensure that when a Transact-SQL statement run-time error occurs, the entire transaction is terminated and rolled back.

You use the SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED statement when you want to allow statements to read rows that have been changed by other transactions but have yet to be committed.

You use the SET TRANSACTION ISOLATION LEVEL READ COMMITTED statement when you want to ensure that statements cannot read data that has been changed but not committed by other transactions.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SET XACT_ABORT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188792.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 44 (461P_1.1_03)

Which of the following Transact-SQL statements moves the Categories table from the Production schema to the Sales schema?

1. ALTER SCHEMA [Production] TRANSFER [Sales].[Categories]
2. >>ALTER SCHEMA [Sales] TRANSFER [Production].[Categories] <Correct>
3. ALTER SCHEMA [Sales] TRANSFER [Categories].[Production]
4. ALTER SCHEMA [Production] TRANSFER [Categories].[Sales]

Explanation:
You should use the ALTER SCHEMA [Sales] TRANSFER [Production].[Categories] statement. This statement will transfer the Categories table from the Production schema to the Sales schema.

You should not use the ALTER SCHEMA [Production] TRANSFER [Sales].[Categories] statement. This statement will transfer the Categories table from the Sale schema to the Production schema.

You should not use the ALTER SCHEMA [Production] TRANSFER [Categories].[Sales] statement. This statement will transfer the Sales table from the Categories schema to the Production schema.

You should not use the ALTER SCHEMA [Sales] TRANSFER [Categories].[Production] statement. This statement will move the Production table from the Categories schema to the Sales schema.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

ALTER SCHEMA (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173423.aspx

ALTER Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/cc879314.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

Question Number (ID) : 45 (461P_2.4_19)

You want to determine the top pay rate for employees who are paid every two weeks.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2);

2. SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;

3. >>SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2; <Correct>

4. SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2;

Explanation:
You should use the statement:

SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2;

This statement lists only the top rate for employees who are paid every two weeks.

You should not use the statement:

SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2;

This statement generates a list of pay rates for employees who are paid every two weeks, comparing each employee's pay rate with that of the top-paid employee who is paid every two weeks.

You should not use the statement:

SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2);

This statement lists the lowest pay rate for employees paid every two weeks.

You should not use the statement:

SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;

This statement ranks employee pay in terms of percentage rank within the organization.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

FIRST_VALUE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh213018.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 46 (461P_2.4_17)

You want to determine the lowest rate of pay for employees who are paid every two weeks.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2)
AND PayFrequency = 2;                <Correct>
```

```
2.  SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE  PayFrequency=2;
```

```
3.  SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
4.  SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2;
```

Explanation:
You should use the statement:

```
SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2);
```

This statement lists the lowest pay rate for employees paid every two weeks.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2;
```

This statement generates a list of pay rates for employees who are paid every two weeks, comparing each employee's pay rate with that of the top-paid employee who is paid every two weeks.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement ranks employee pay in terms of their percentage rank within the organization.

You should not use the statement:

```
SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2;
```

This statement lists only the top rate for employees who are paid every two weeks.

Objective:

Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

LAST_VALUE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh231517.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 47 (461P_3.3_02)

Your database has two tables related to product models, named Production.ProductModel and Production.NewProductModel. You want to produce a list of those products and productIDs that are present in both tables.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT ProductModelID, Name FROM Production.ProductModel EXCEPT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name

2. >>SELECT ProductModelID, Name FROM Production.ProductModel INTERSECT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name <Correct>

3. SELECT ProductModelID, Name FROM Production.ProductModel UNION SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name

4. SELECT ProductModelID, Name FROM Production.NewProductModel EXCEPT SELECT ProductModelID, Name FROM Production.ProductModel ORDER BY Name

5. SELECT ProductModelID, Name FROM Production.ProductModel UNION ALL SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name

Explanation:
You should use the SELECT ProductModelID, Name FROM Production.ProductModel INTERSECT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce only those products and productIDs that are present in both tables.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel UNION SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce a list of all products in both tables without duplicate entries. You want to produce a list of those products and productIDs that are present in both tables.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel EXCEPT SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce a list of products from the ProductModel table as long as those products do not have duplicate entries in the NewProductModel table. You want to produce a list of those products and productIDs that are present in both tables.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel UNION ALL SELECT ProductModelID, Name FROM Production.NewProductModel ORDER BY Name statement. This will produce a list of all products in both tables but will output duplicate entries. You want to produce a list of those products and productIDs that are present in both tables.

You should not use the SELECT ProductModelID, Name FROM Production.NewProductModel EXCEPT SELECT ProductModelID, Name FROM Production.ProductModel ORDER BY Name statement. This will produce a list of products from the NewProductModel table as long as those products do not have duplicate entries in the ProductModel table. You want to produce a list of those products and productIDs that are present in both tables.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

Set Operators (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ff848745.aspx

Question Number (ID) : 48 (461P_1.5_03)

You are creating a trigger that will report the number of rows in the Deleted table during its operation.

Which of the following statements would you include when defining this trigger to accomplish this goal?

1. >>SELECT COUNT(*) FROM Deleted <Correct>
2. SELECT COUNT(*) FROM Inserted
3. SELECT * FROM Deleted
4. SELECT * FROM Inserted

Explanation:
You should use the SELECT COUNT(*) FROM Deleted statement. This statement counts the number of rows in the Deleted table.

You should not use the SELECT COUNT(*) FROM Inserted statement. This statement counts the number of rows in the Inserted table. You are interested in the number of rows in the Deleted table.

You should not use the SELECT * FROM Inserted statement. This statement selects all content in the Inserted table, but it does not count the rows. You are interested in the number of rows in the Deleted table.

You should not use the SELECT * FROM Deleted statement. This statement selects all content from the Deleted table, but it does not count the rows in the Deleted table.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Use the inserted and deleted Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191300.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Question Number (ID) : 49 (461P_3.4_01)

You want to generate a list of the square roots of all integers between 1 and 20.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
2.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
3.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;                          <Correct>
```

```
4.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

Explanation:
You should use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and their square roots between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and their squares between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and the natural exponent (the constant e raised to the power of the number) between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of integers and their natural logarithms between 1 and 20.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

SQRT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176108.aspx

Question Number (ID) : 50 (461P_1.2_10)

You want to modify the properties of a view and move it to another schema.

Which of the following Transact-SQL statements would you use to accomplish this goal? (Each correct answer presents part of a complete solution. Choose two.)

1. >>ALTER SCHEMA <Correct>
2. CREATE VIEW
3. >>ALTER VIEW <Correct>
4. DROP VIEW

Explanation:
You use the ALTER VIEW statement to modify the properties of an existing view.

You use the ALTER SCHEMA statement to move a view from one schema to another.

You should not use the CREATE VIEW statement. This statement enables you to create a view. In this question, you are asked to move a view to a different schema and alter its properties.

You should not use the DROP VIEW statement. In this question, you are asked to move a view to a different schema and alter its properties.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

ALTER SCHEMA (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173423.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 51 (461P_2.1_10)

You want to query the Sales.SalesOrderDetail to generate a list of product IDs for products that cost less than $100.00 that have an average order quantity greater than 10.

Which of the following Transact-SQL statements can you use to accomplish this goal?

```
1.  SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice > 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) > 10
ORDER BY ProductID;
```

```
2.  SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice < 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) < 10
ORDER BY ProductID;
```

```
3.  SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice > 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) < 10
ORDER BY ProductID;
```

```
4.  >>SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice < 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) > 10
ORDER BY ProductID;
```

<Correct>

Explanation:
You should use the following statement:

```
SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice < 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) > 10
ORDER BY ProductID;
```

This statement will generate a list of product IDs of products that cost less than $100.00 and have an average order quantity exceeding 10 units.

You should not use the following statement:

```
SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice > 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) > 10
ORDER BY ProductID;
```

This statement will generate a list of product IDs of products that cost more than $100.00 and have an average order quantity exceeding 10 units. You are interested in products that cost less than $100.00.

You should not use the following statement:

```
SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice < 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) < 10
ORDER BY ProductID;
```

This statement will generate a list of product IDs of products that cost less than $100.00 and have an average order quantity of fewer than 10 units. You are interested in products that have an average order quantity of more than 10 units.

You should not use the following statement:

```
SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice > 100.00
GROUP BY ProductID
HAVING AVG(OrderQty) < 10
ORDER BY ProductID;
```

This statement will generate a list of product IDs of products that cost more than $100.00 and have an average order quantity of fewer than 10 units. You are interested in products that cost less than $100.00 and have an average order quantity of more than 10 units.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

Question Number (ID) : 52 (461P_2.4_10)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued sick leave hours. When an employee is off sick and uses all his or her hours, the hours are reset to NULL. You want to determine the total number of employees who have a job title like Marketing Manager who have outstanding sick leave hours.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT MAX(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

```
2.  SELECT AVG(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

```
3.  >>SELECT COUNT(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';          <Correct>
```

```
4.  SELECT SUM(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

Explanation:
You should use the statement:

```
SELECT COUNT(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This query finds the number of employees with the title Marketing Manager who have values set for sick leave hours other than NULL.

You should not use the statement:

```
SELECT SUM(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement determines the total sum of sick leave hours of all employees who have the job title Marketing Manager. You want to find the number of employees with the title Marketing Manager who have values set for sick leave hours other than NULL.

You should not use the statement:

```
SELECT AVG(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This query finds the average number of sick leave hours for employees who have the job title Marketing Manager. You want to find the number of employees with the title Marketing Manager who have values set for sick leave hours other than NULL.

You should not use the statement:

```
SELECT MAX(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This query finds the maximum number of sick leave hours that any employee who has the job title Marketing Manager has. You want to find the number of employees with the title Marketing Manager who have values set for sick leave hours other than NULL.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

COUNT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175997.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 53 (461P_3.2_05)

You create the table Hovercraft.Chassis by executing the following statement:

```
CREATE TABLE Hovercraft.Chassis
(
HovercraftID uniqueidentifier DEFAULT NEWID() PRIMARY KEY,
hullid INT NOT NULL,
registration nvarchar(8) NOT NULL,
maintenancedate DATE NOT NULL
CONSTRAINT DFT_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)), homebase NVARCHAR(15) NOT NULL,
fuelcapacity decimal(8,4) NOT NULL
);
```

Which of the following statements can you use to insert data into this table? (Each correct answer presents a complete solution. Choose two.)

1. >>INSERT INTO Hovercraft.Chassis (maintenancedate, hullid, registration, fuelcapacity, homebase) VALUES ('20120727', 26, N'AAA128', 800.75, N'Brisbane') <Correct>

2. >>INSERT INTO Hovercraft.Chassis (hullid, registration, maintenancedate, homebase, fuelcapacity) VALUES (24, N'AAA123', '20120727', N'Melbourne', 900.50) <Correct>

3. INSERT INTO Hovercraft.Chassis (hullid, registration, maintenancedate, homebase, fuelcapacity) VALUES (25, N'AAA124', 20120728, N'Sydney', 950.50)

4. INSERT INTO Hovercraft.Chassis (maintenancedate, hullid, registration, fuelcapacity, homebase) VALUES (20120719, 22, N'AAA119', 800.75, N'Hobart')

Explanation:
You can use the INSERT INTO Hovercraft.Chassis (hullid, registration, maintenancedate, homebase, fuelcapacity) VALUES (24, N'AAA123', '20120727', N'Melbourne', 900.50) statement. This statement correctly pairs column data types with column data as defined in the table creation statement.

You can use the INSERT INTO Hovercraft.Chassis (maintenancedate, hullid, registration, fuelcapacity, homebase) VALUES ('20120727', 26, N'AAA128', 800.75, N'Brisbane') statement. This statement correctly pairs column data types with column data as defined in the table creation statement.

You should not use the INSERT INTO Hovercraft.Chassis (hullid, registration, maintenancedate, homebase, fuelcapacity) VALUES (25, N'AAA124', 20120728, N'Sydney', 950.50) statement. This statement does not correctly pair column data types with column data as defined in the table creation statement. The problematic data is the 20120728 string, which, when used without the appropriate ' delimiter, will be treated as an integer rather than as a date string.

You should not use the INSERT INTO Hovercraft.Chassis (maintenancedate, hullid, registration, fuelcapacity, homebase) VALUES (20120719, 22, N'AAA119', 800.75, N'Hobart') statement. This statement does not correctly pair column data types with column data as defined in the table creation statement. The problematic data is the 20120719 string, which, when used without the appropriate ' delimiter, will be treated as an integer rather than as a date string.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

INSERT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174335.aspx

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 54 (461P_2.4_05)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued vacation hours. You want to determine the minimum vacation hours accrued by any single employee who has a job title like Design Engineer.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT VAR(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';

2.  SELECT AVG(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';

3.  >>SELECT MIN(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';              <Correct>

4.  SELECT MAX(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

Explanation:
You should use the statement:

```
SELECT MIN(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the minimum value for vacation hours for employees who have the job title Design Engineer.

You not should use the statement:

```
SELECT MAX(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the maximum value for vacation hours for employees who have the job title Design Engineer. You want to find the minimum value for vacation hours for employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT AVG(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the average value for vacation hours for employees with the job title Design Engineer. You want to find the minimum value for vacation hours for employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT VAR(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the statistical variance in vacation hours for employees with the job title Design Engineer. You want to find the minimum value for vacation hours for employees who have the job title Design Engineer.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

MIN (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179916.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 55 (461P_2.1_07)

You want to create a list of maximum and minimum ListPrices for each ProductModelID by querying the Production.Product database.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT ProductModelID, AVG(Listprice), STDEV(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID;

2. >>SELECT ProductModelID, MIN(Listprice), MAX(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID; <Correct>

3. SELECT ProductModelID, MIN(Listprice), AVG(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID;

4. SELECT ProductModelID, AVG(Listprice), MAX(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID;

Explanation:
You should use the SELECT ProductModelID, MIN(Listprice), MAX(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID; statement. This query will output a list of maximum and minimum prices for each product model by querying the Production.Product database.

You should not use the SELECT ProductModelID, MIN(Listprice), AVG(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID; statement. This query will output a list of minimum and average prices for each product model by querying the Production.Product database.

You should not use the SELECT ProductModelID, AVG(Listprice), MAX(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID; statement. This query will output a list of average and maximum prices for each product model by querying the Production.Product database.

You should not use the SELECT ProductModelID, AVG(Listprice), STDEV(Listprice) from Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductionModelID; statement. This query will output the average product model price and the standard deviation for a product model's price. The query should output only the minimum and maximum prices for each product model.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 56 (461P_2.2_10)

You want to generate a list of customers who placed an order for a hovercraft after 1 Jan 2012.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate < '20120101');

2. SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate BETWEEN '20100101' AND '20120101');

3. >>SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate > '20120101'); <Correct>

4. SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate = '20120101');

Explanation:
You should use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate > '20120101'); statement. This statement will generate a list of orders that were made after 1 Jan 2012.

You should not use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate < '20120101'); statement. This statement will generate a list of orders made prior to 1 Jan 2012. You want to generate a list of orders that were made after 1 Jan 2012.

You should not use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate = '20120101'); statement. This statement will generate a list of orders made on 1 Jan 2012. You want to generate a list of orders that were made after 1 Jan 2012.

You should not use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate BETWEEN '20100101' AND '20120101'); statement. This statement will generate a list of orders made between 1 Jan 2010 and 1 Jan 2012. You want to generate a list of orders that were made after 1 Jan 2012.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

Question Number (ID) : 57 (461P_4.1_03)

You are optimizing several complex queries. You have determined that you can improve performance by always merging UNION sets.

Which of the following query hints should you use to implement this optimization?

1. HASH JOIN

2. >>MERGE UNION <Correct>

3. HASH UNION

4. CONCAT UNION

Explanation:
The MERGE UNION query hint option specifies that all UNION operations will be performed by merging UNION sets.

The CONCAT UNION query hint option specifies that all UNION operations will be performed by concatenating UNION sets.

The HASH UNION query hint option specifies that all UNION operations will be performed by hashing UNION sets.

The HASH JOIN query hint option specifies that all JOIN operations in the query use HASH JOINs.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Query Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181714.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

Question Number (ID) : 58 (461P_3.4_02)

You want to generate a list of the squares of all integers between 1 and 20.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
2.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
3.  >>DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;                     <Correct>
```

```
4.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

Explanation:
You should use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and their squares between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and their square roots between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and the natural exponent (the constant e raised to the power of the number) between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and their natural logarithms between 1 and 20.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

SQUARE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173569.aspx

Question Number (ID) : 59 (461P_2.4_11)

The HumanResources.EmployeePayHistory table records employee pay and pay frequency. You are interested in determining the average pay and its standard deviation for employees who are paid every two weeks.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1. >>Select AVG(Rate) AS 'Average Rate', STDEV(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;          <Correct>
```

```
2. Select SUM(Rate) AS 'Average Rate', MIN(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
3. Select MAX(Rate) AS 'Average Rate', AVG(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
4. Select AVG(Rate) AS 'Average Rate', SUM(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

Explanation:

You should use the statement:

```
Select AVG(Rate) AS 'Average Rate', STDEV(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement calculates the average rate and standard deviation for all employees who are paid every two weeks.

You should not use the statement:

```
Select MAX(Rate) AS 'Average Rate', AVG(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement calculates the maximum rate and average rate for all employees who are paid every two weeks. You want to calculate the average rate and standard deviation for all employees who are paid every two weeks.

You should not use the statement:

```
Select AVG(Rate) AS 'Average Rate', SUM(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement calculates the average rate and total paid for all employees who are paid every two weeks. You want to calculate the average rate and standard deviation for all employees who are paid every two weeks.

You should not use the statement:

```
Select SUM(Rate) AS 'Average Rate', MIN(Rate) AS 'Standard Deviation'
FROM HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement calculates the total sum and average rate for all employees who are paid every two weeks. You want to calculate the average rate and standard deviation for all employees who are paid every two weeks.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

STDEV
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190474.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 60 (461P_3.1_15)

You want to create a stored procedure that enables you to update the vacation hours of employees. Employees who are salaried (SalariedFlag column set to 1) should receive these new vacation hours as an addition to their existing vacation hours. Employees who are not salaried (SalariedFlag column set to 0) should have their existing vacation hours set to the new vacation hours value.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>CREATE PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 1 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);                     <Correct>
```

```
2.  ALTER PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 1 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);
```

```
3.  CREATE PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 0 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);
```

```
4.  ALTER PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 0 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
```

```
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 1 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);
```

Executing this statement creates a stored procedure that updates the vacation hours of employees who are salaried (SalariedFlag column set to 1) so that they receive these new vacation hours as an addition to their existing vacation hours. Employees who are not salaried (SalariedFlag column set to 0) will have their existing vacation hours set to the new vacation hours value.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 0 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);
```

Instead of creating a stored procedure that updates the vacation hours of employees who are salaried so that they receive these new vacation hours as an addition to their existing vacation hours and employees who are not salaried have their existing vacation hours set to the new vacation hours value, this stored procedure does the opposite. Salaried employees get their vacation hours reset to the new value, and nonsalaried employees get the new value in addition to their existing vacation hours.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 1 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);
```

You use the CREATE PROCEDURE statement to create a new stored procedure. You use the ALTER PROCEDURE statement to modify an existing stored procedure.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.Update_VacationHours
@NewHours smallint
AS
SET NOCOUNT ON;
UPDATE HumanResources.Employee
SET VacationHours =
 ( CASE
    WHEN SalariedFlag = 0 THEN VacationHours + @NewHours
    ELSE @NewHours
  END);
```

You use the CREATE PROCEDURE statement to create a new stored procedure. You use the ALTER PROCEDURE statement to modify an existing stored procedure.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 61 (461P_2.5_01)

Which of the following options is correctly set to support querying XML-type columns?

1. ANSI_NULLS OFF

2. ANSI_WARNINGS OFF

3. >>ANSI_PADDING ON <Correct>

4. QUOTED_IDENTIFIER OFF

Explanation:
When preparing to query XML-type columns or variables using XML data type methods, you must ensure that the following options are set:

ANSI_PADDING ON
ANSI_NULLS ON
ANSI_WARNINGS ON
QUOTED_IDENTIFIER ON

Other options that should be configured are as follows:

ARITHABORT ON
CONTACT_NULL_YIELDS_NULL ON
NUMERIC_ROUNDABORT OFF

If these options are not set in this manner, queries and modifications using XML data type methods fail.

Objective:
Work with Data

Sub Objective(s):
Query and manage XML data.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 7 - Lesson 1

Retrieve and Query XML Data
MSDN
Link: http://msdn.microsoft.com/en-us/library/bb510442.aspx

XML Data (SQL Server)
MSDN
Link: http://msdn.microsoft.com/en-us/library/bb522446.aspx

Question Number (ID) : 62 (461P_4.2_11)

You are configuring an appropriate transaction isolation level for a session. You want to ensure the following:

*Statements in the transaction should not be able to read data altered by other transactions if those transactions have not yet been committed.
*Statements in other transactions should not be able to read data altered by the current transaction until the current transaction commits.
*Other transactions should be able to insert new rows that meet the search conditions of statements in the current transaction.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. SET TRANSACTION ISOLATION LEVEL REPEATABLE READ <Correct>

2. >>SET TRANSACTION ISOLATION LEVEL READ COMMITTED

3. SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

4. SET TRANSACTION ISOLATION LEVEL SNAPSHOT

5. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

Explanation:
You use the SET TRANSACTION ISOLATION LEVEL REPEATABLE READ statement when you want to ensure that statements cannot read data altered by other transactions not yet committed and that other transactions cannot modify data read by the current transactions until the current transaction commits. When the REPEATABLE READ isolation level is set, it is possible for other transactions to insert new rows that meet the search conditions of statements included in the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED statement when you want to allow statements to read rows altered by other transactions, but those transactions have not yet been committed.

You use the SET TRANSACTION ISOLATION LEVEL READ COMMITTED statement when you want to ensure that statements cannot read data altered by other transactions when those transactions have not yet been committed. When you set the READ COMMITTED transaction level, data can be altered by other transactions between individual statements within the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL SNAPSHOT statement when you want to ensure that data read by any statement in a transaction will be transactionally consistent with how that data existed at the start of the transaction. SNAPSHOT transactions do not block other transactions from writing data.

You use the SET TRANSACTION ISOLATION LEVEL SERIALIZABLE statement when you want to ensure that Transact-SQL statements cannot read data that has been altered but not committed, that other transactions cannot modify data read by the current transaction until the transaction commits, and when other transactions cannot insert new rows with key values that would fall into the range read by statements in the current transaction until that transaction completes.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SET TRANSACTION ISOLATION LEVEL (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173763.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 63 (461P_2.3_10)

You are creating the Passengers column for a table named Capacity for the Hovercraft database. None of the hovercraft that your company owns can hold more than 200 passengers. You will be storing each Hovercraft's maximum passenger capacity in this column.

Which of the following statements would you use when creating this column if you wanted to minimize the number of bytes consumed by the column when stored in the database?

1. CREATE TABLE Capacity (Passengers bigint)
2. CREATE TABLE Capacity (Passengers int)
3. >>CREATE TABLE Capacity (Passengers tinyint) <Correct>
4. CREATE TABLE Capacity (Passengers smallint)

Explanation:
You should use the CREATE TABLE Capacity (Passengers tinyint) statement. The tinyint data type enables you to store integer values between 0 and 255.

You should not use the CREATE TABLE Capacity (Passengers int) statement. Although the int data type could store this value, it uses 4 bytes as opposed to the tinyint data type's 1 byte.

You should not use the CREATE TABLE Capacity (Passengers smallint) statement. Although the smallint data type could store this value, it uses 2 bytes as opposed to the tinyint data type's 1 byte.

You should not use the CREATE TABLE Capacity (Passengers bigint) statement. Although the bigint data type could store this value, it uses 8 bytes as opposed to the tinyint data type's 1 byte.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 2 - Lesson 2

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

int, bigint, smallint, and tinyint (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187745.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 64 (461P_2.4_18)

You want to generate a list that ranks employee pay in terms of their percentage rank within the organization.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE  PayFrequency=2;
```

```
2.  >>SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;            <Correct>
```

```
3.  SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE  PayFrequency=2);
```

```
4.  SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2;
```

Explanation:
You should use the statement:

```
SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement ranks employee pay in terms of percentage rank within the organization.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2;
```

This statement generates a list of pay rates for employees who are paid every two weeks, comparing each employee's pay rate with that of the top-paid employee who is paid every two weeks.

You should not use the statement:

```
SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2);
```

This statement lists the lowest pay rate for employees paid every two weeks.

You should not use the statement:

```
SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2;
```

This statement lists only the top rate for employees who are paid every two weeks.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

PERCENT_RANK (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh213573.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 65 (461P_1.4_09)

You created a check constraint on the Score column of the Results table that incorrectly limits the values that can be inserted in that column. This check constraint is named CK_Results. You want to modify this check constraint so that it allows values between 1 and 20, inclusive.

Which of the following Transact-SQL statements can you use to accomplish this goal? (Each correct answer presents part of a complete solution. Choose two.)

1. ALTER TABLE dbo.Results ADD CONSTRAINT PK_Results PRIMARY KEY (Results)

2. ALTER TABLE dbo.Results ADD CONSTRAINT UN_Results UNIQUE (Score)

3. >>ALTER TABLE dbo.Results DROP CONSTRAINT CK_Results <Correct>

4. >>ALTER TABLE dbo.Results ADD CONSTRAINT CK_Results CHECK (Score > 0 and Score < 21) <Correct>

Explanation:
You should use the ALTER TABLE dbo.Results DROP CONSTRAINT CK_Results statement. To modify an existing check constraint, you need to drop the existing constraint and create a new one. This statement removes the existing check constraint.

You should use the ALTER TABLE dbo.Results ADD CONSTRAINT CK_Results CHECK (Score > 0 and Score < 21) statement. This statement creates the new check constraint. It must be executed after you remove the existing constraint.

You should not use the ALTER TABLE dbo.Results ADD CONSTRAINT UN_Results UNIQUE (Score) statement. This statement creates a new unique constraint. In this case, you need a new check constraint, just one with different properties.

You should not use the ALTER TABLE dbo.Results ADD CONSTRAINT CK_Results PRIMARY KEY (Results) statement. This statement creates a new primary key constraint. In this case, you need a new check constraint, just one with different properties.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Delete Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187626.aspx

Unique Constraints and Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187550.aspx

Question Number (ID) : 66 (461P_3.2_02)

You create the Hovercraft.OrdersAPAC table by using the following statement:

```
CREATE TABLE Hovercraft.OrdersAPAC
(
orderid INT NOT NULL IDENTITY(1, 1)
CONSTRAINT PK_Orders_orderid PRIMARY KEY,
custid INT NOT NULL,
empid INT NOT NULL,
orderdate DATE NOT NULL
CONSTRAINT DFX_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)), shipcountry NVARCHAR(15) NOT
NULL,
freight MONEY NOT NULL
);
```

The Hovercraft.Orders statement already exists and was created using the following statement:

```
CREATE TABLE Hovercraft.Orders
(
orderid INT NOT NULL IDENTITY(1, 1)
CONSTRAINT PK_Orders_orderid PRIMARY KEY,
custid INT NOT NULL,
empid INT NOT NULL,
orderdate DATE NOT NULL
CONSTRAINT DFZ_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)), shipcountry NVARCHAR(15) NOT
NULL,
freight MONEY NOT NULL
);
```

You want to populate the Hovercraft.OrdersAPAC table with all the order data from the Hovercraft.Orders table related to countries in the Asia/Pacific region such as Australia and New Zealand.

Which of the following statements can you use to accomplish this goal of inserting order data where Australia is the ship country? (Each correct answer presents a complete solution. Choose two.)

1. >>INSERT INTO Hovercraft.OrdersAPAC(shipcountry, orderid, custid, empid, orderdate, freight) SELECT shipcountry, orderid, custid, empid, orderdate, freight FROM Hovercraft.Orders WHERE shipcountry = N'Australia'; <Correct>

2. >>INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, shipcountry, freight) SELECT orderid, custid, empid, orderdate, shipcountry, freight FROM Hovercraft.Orders WHERE shipcountry = N'Australia'; <Correct>

3. INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, shipcountry, freight) SELECT orderid, custid, empid, orderdate, shipcountry, freight FROM Hovercraft.Orders;

4. INSERT INTO Hovercraft.OrdersAPAC(shipcountry, orderid, custid, empid, orderdate, freight) SELECT shipcountry, orderid, custid, empid, orderdate, freight FROM Hovercraft.Orders;

Explanation:
You can use the INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, shipcountry, freight) SELECT orderid, custid, empid, orderdate, shipcountry, freight FROM Hovercraft.Orders WHERE shipcountry = N'Australia'; statement. This statement correctly matches the data types of both the source and destination tables. It will insert data where Australia is listed in the shipcountry column.

You can use the INSERT INTO Hovercraft.OrdersAPAC(shipcountry, orderid, custid, empid, orderdate, freight) SELECT shipcountry, orderid, custid, empid, orderdate, freight FROM Hovercraft.Orders WHERE shipcountry = N'Australia'; statement. This statement correctly matches the data types of both the source and destination tables. It will insert data where Australia is listed in the shipcountry column.

You cannot use the INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, shipcountry, freight) SELECT orderid, custid, empid, orderdate, shipcountry, freight FROM Hovercraft.Orders; statement. Although this statement correctly matches the data types of both the source and destination tables, it inserts all data from the source table, not just data where Australia is listed in the shipcountry column.

You cannot use the INSERT INTO Hovercraft.OrdersAPAC(shipcountry, orderid, custid, empid, orderdate, freight) SELECT shipcountry, orderid, custid, empid, orderdate, freight FROM Hovercraft.Orders; statement. Although this statement correctly matches the data types of both the source and destination tables, it inserts all data from the source table, not just data where Australia is listed in the shipcountry column.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

INSERT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174335.aspx

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

Question Number (ID) : 67 (461P_1.5_07)

You want to re-enable the existing trigger named Hovercraft.service_reminder that applies to the Hovercraft.Logbook table. You might need to disable this trigger in the future.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. DISABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook

2. DROP TRIGGER Hovercraft.service_reminder

3. ALTER TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook

4. >>ENABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook <Correct>

Explanation:
You should use the ENABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook statement. This statement will enable the disabled trigger.

You should not use the DISABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook statement. This statement disables the trigger and will not enable a disabled trigger.

You should not use the DROP TRIGGER Hovercraft.service_reminder statement. This statement deletes the trigger rather than re-enabling the trigger.

You should not use the ALTER TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook statement. The ALTER TRIGGER statement modifies the definition of a trigger; you cannot use this statement to enable a disabled trigger.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Delete or Disable DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175506.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 68 (461P_2.2_09)

You want a list of customers who placed an order between 1 Jan 2010 and 1 Jan 2012.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate BETWEEN '20100101' AND '20120101'); <Correct>

2. SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate > '20100101');

3. SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate = '20100101');

4. SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate < '20100101');

Explanation:
You should use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate BETWEEN '20100101' AND '20120101'); statement. This statement will return the orders placed between 1 Jan 2010 and 1 Jan 2012.

You should not use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate = '20100101'); statement. This statement will only return orders placed on 1 Jan 2010. You want a statement that returns orders placed between 1 Jan 2010 and 1 Jan 2012.

You should not use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate < '20100101'); statement. This statement will only return orders placed before 1 Jan 2010. You want a statement that returns orders placed between 1 Jan 2010 and 1 Jan 2012.

You should not use the SELECT custid, companyname FROM Hovercraft.Customers AS C WHERE EXISTS (SELECT * FROM Hovercraft.Orders AS O WHERE O.custid = C.custid AND O.orderdate > '20100101'); statement. This statement will only return orders placed after 1 Jan 2010. You want a statement that returns orders placed between 1 Jan 2010 and 1 Jan 2012.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

Question Number (ID) : 69 (461P_3.1_04)

You want to create a stored procedure associated with the HumanResources schema that queries the HumanResources.vEmployeeDepartmentHistory view to generate a list of employees that includes their first names, last names, and start dates of employment.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, EndDate
   FROM  HumanResources.vEmployeeDepartmentHistory;
```

```
2.  CREATE PROCEDURE Person.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, EndDate
   FROM  HumanResources.vEmployeeDepartmentHistory;
```

```
3.  CREATE PROCEDURE Person.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM  HumanResources.vEmployeeDepartmentHistory;
```

```
4.  >>CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM  HumanResources.vEmployeeDepartmentHistory;                    <Correct>
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

This statement creates a stored procedure that is associated with the HumanResources schema and will output the last name, first name, and start date information.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

This statement creates a stored procedure that is associated with the HumanResources schema and will output the last name, first name, and end date information. You want to create a stored procedure that is associated with the HumanResources schema and will output the last name, first name, and start date information.

You should not use the statement:

```
CREATE PROCEDURE Person.uspEmployeeEndDates
AS
```

```
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

This statement creates a stored procedure that is associated with the Person schema and will output the last name, first name, and start date information. You want to create a stored procedure that is associated with the HumanResources schema and will output the last name, first name, and start date information.

You should not use the statement:

```
CREATE PROCEDURE Person.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

This statement creates a stored procedure that is associated with the Person schema and will output the last name, first name, and end date information. You want to create a stored procedure that is associated with the HumanResources schema and will output the last name, first name, and start date information.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 70 (461P_3.4_18)

You want to use the appropriate function to change a string that has the date written in Australian format so that it can be written to a column that uses the datetime2 data type. The conversion should return a null value if this is not possible.

Which of the following built-in functions can you use to accomplish this goal?

1. >>TRY_PARSE <Correct>

2. PARSE

3. TRY_CONVERT

4. CONVERT

Explanation:
TRY_PARSE enables you to change a string written in one cultural format (such as en-AU) to other data types such as datetime and datetime2. When you use TRY_PARSE, if the operation fails, the conversion occurs with the value being set to NULL.

The PARSE function enables you to change a string written in one cultural format (such as en-AU) to data types such as datetime and datetime2. If you do not use PARSE, strings that are not in the appropriate format will not convert.

TRY_CONVERT enables you to attempt to convert data stored in one data type to another. If this conversion fails, a NULL value is written.

CONVERT enables you to convert data stored in one data type to another. The conversion might fail if you attempt to convert between data types where conversion is not possible.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

TRY_PARSE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh213126.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 71 (461P_1.5_04)

You want to delete a trigger named Hovercraft.Data_Protect from the Hovercraft.Chassis table.

Which of the following Transact-SQL statements would you use to accomplish this goal?

1. CREATE TRIGGER Hovercraft.Chassis

2. CREATE TRIGGER Hovercraft.Data_Protect

3. >>DROP TRIGGER Hovercraft.Data_Protect <Correct>

4. DROP TRIGGER Hovercraft.Chassis

Explanation:
You should use the DROP TRIGGER Hovercraft.Data_Protect statement. This statement deletes the trigger named Hovercraft.Data_Protect.

You should not use the DROP TRIGGER Hovercraft.Chassis statement. This statement deletes a trigger named Hovercraft.Chassis. The name of the trigger that you want to delete is Hovercraft.Data_Protect.

You should not use the CREATE TRIGGER Hovercraft.Data_Protect statement. This statement is part of a statement that creates a trigger rather than deleting it.

You should not use the CREATE TRIGGER Hovercraft.Chassis statement. This statement refers to the Hovercraft.Chassis table and does not provide enough information to create a trigger. In this question, you are asked to delete a trigger.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Delete or Disable DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175506.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 72 (461P_1.5_09)

Which of the following types of DML triggers can you create using the CREATE TRIGGER statement? (Each correct answer presents a complete solution. Choose two.)

1. Transact-SQL

2. CLR DDL

3. >>AFTER <Correct>

4. >>INSTEAD OF <Correct>

Explanation:
There are two types of the DML trigger, the AFTER and INSTEAD OF triggers.

CLR DDL and Transact-SQL triggers are DDL rather than DML triggers.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Create DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190227.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Question Number (ID) : 73 (461P_2.2_02)

You are writing a query against tables in the AdventureWorks2012 database. You want to use information in the Person.Person, HumanResources.Employee, and Sales.SalesPerson tables to generate a list of the first and last names of each employee who received a bonus of $3500.00. The names output by the query should not be duplicated.

Which statement should you use?

```
1.  SELECT DISTINCT p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 3500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
```

```
2.  SELECT DISTINCT p.LastName, p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 4500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
```

```
3.  >>SELECT DISTINCT p.LastName, p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 3500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
          <Correct>
```

```
4.  SELECT DISTINCT p.LastName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 3500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
```

Explanation:
You should use the following statement:

```
SELECT DISTINCT p.LastName, p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 3500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
```

This statement will use a subquery against the contents of multiple tables to produce output of the first name and last name of all salespeople that received a bonus of $3500.00. No duplicates will be output.

You should not use the following statement:

```
SELECT DISTINCT p.LastName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 3500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
```

This statement will output the salesperson's last name only.

You should not use the following statement:

```
SELECT DISTINCT p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 3500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
```

This statement will output the salesperson's first name only.

You should not use the following statement:

```
SELECT DISTINCT p.LastName, p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
    ON e.BusinessEntityID = p.BusinessEntityID WHERE 4500.00 IN
    (SELECT Bonus
     FROM Sales.SalesPerson AS s
     WHERE e.BusinessEntityID = s.BusinessEntityID);
```

This statement will output the first name and last name of salespeople who earn a bonus of $4500.00 rather than $3500.00.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 74 (461P_1.3_06)

Which of the following clauses or keywords can you include when creating a view?

1. COMPUTE BY
2. COMPUTE
3. INTO
4. >>ENCRYPTION <Correct>

Explanation:
You can use the ENCRYPTION keyword when creating a view. This stops other users from being able to see the view definition.

You cannot use the COMPUTE clause when creating a view.

You cannot use the COMPUTE BY clause when creating a view.

You cannot use the INTO keyword when creating a view.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 75 (461P_3.1_03)

You want to create a stored procedure associated with the HumanResources schema that queries the vEmployeeDepartmentHistory view to generate a list of employees that includes their first names, last names, and end dates of employment.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT MiddleName, FirstName, StartDate
   FROM  HumanResources.vEmployeeDepartmentHistory;
```

```
2.  CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM  HumanResources.vEmployeeDepartmentHistory;
```

```
3.  CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT MiddleName, FirstName, EndDate
   FROM  HumanResources.vEmployeeDepartmentHistory;
```

```
4.  >>CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, EndDate
   FROM  HumanResources.vEmployeeDepartmentHistory;                    <Correct>
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

This statement enables you to create a stored procedure named uspEmployeeEndDates associated with the HumanResources schema that outputs employee first names, last names, and end dates.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT MiddleName, FirstName, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

You use this statement to create a stored procedure named uspEmployeeEndDates associated with the HumanResources schema that outputs employee first names, middle names, and end dates. You want the last name rather than the middle name output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

You use this statement to create a stored procedure named uspEmployeeEndDates associated with the HumanResources schema that outputs employee last names, first names, and start dates. You want the end dates rather than the start dates.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEndDates
AS
   SET NOCOUNT ON;
   SELECT MiddleName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory;
```

You use this statement to create a stored procedure named uspEmployeeEndDates associated with the HumanResources schema that outputs employee middle names, first names, and start dates. You want the end dates and last names rather than the start dates and middle names.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

Question Number (ID) : 76 (461P_1.1_07)

You want to update the properties of the Suburb column in the Address table of the TailspinToys2012 database. This table is located in the Orders schema. The column should now use the nvarchar(150) data type.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER TABLE [Address].[Orders] ALTER COLUMN Suburb nvarchar(150)

2. >>ALTER TABLE [Orders].[Address] ALTER COLUMN Suburb nvarchar(150) <Correct>

3. ALTER COLUMN [Orders].[Address].[Suburb] nvarchar(150)

4. ALTER COLUMN [Tailspintoys2012].[Orders] suburb nvarchar(150)

Explanation:
You should use the ALTER TABLE [Orders].[Address] ALTER COLUMN Suburb nvarchar(150) statement. This statement will update the Suburb column in the Address table that is located in the Orders schema so that it is of data type nvarchar(150).

You should not use the ALTER TABLE [Address].[Orders] ALTER COLUMN Suburb nvarchar(150) statement. This statement will update the Suburb column in the Orders table that is located in the Address schema rather than the Address table in the Orders schema.

You should not use the ALTER COLUMN [Tailspintoys2012].[Orders] suburb nvarchar(150) statement. You can only use the ALTER COLUMN statement from within the ALTER TABLE statement.

You should not use the ALTER COLUMN [Orders].[Address].[Suburb] nvarchar(150) statement. You can only use the ALTER COLUMN statement from within the ALTER TABLE statement.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

Modify Columns (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190259.aspx

ALTER Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/cc879314.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

Question Number (ID) : 77 (461P_2.1_06)

You want to create a list of average Listprice for each ProductModelID by querying the Production.Product database.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>SELECT ProductModelID, AVG(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID; <Correct>

2. SELECT ProductModelID, MIN(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID;

3. SELECT ProductModelID, MAX(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID;

4. SELECT ProductModelID, STDEV(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID;

Explanation:
You should use the SELECT ProductModelID, AVG(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID; statement. This statement will generate a list of average prices for a specific product model ID from the Production.Product table in the AdventureWorks2012 database.

You should not use the SELECT ProductModelID, MAX(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID; statement. This statement will generate a list of maximum prices for a specific product model ID from the Production.Product table in the AdventureWorks2012 database.

You should not use the SELECT ProductModelID, MIN(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID; statement. This statement will generate a list of minimum prices for a specific product model ID from the Production.Product table in the AdventureWorks2012 database.

You should not use the SELECT ProductModelID, STDEV(Listprice) FROM Production.Product WHERE ProductModelID IS NOT NULL GROUP BY ProductModelID; statement. This statement will generate a list of the standard deviation of prices listed for a specific product model ID from the Production.Product table in the AdventureWorks2012 database.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 78 (461P_3.4_03)

You want to generate a list of integers and the natural exponent (the constant e raised to the power of the number) between 1 and 20.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;                            <Correct>
```

```
2.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
3.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

```
4.  DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

Explanation:
You should use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', EXP(@testvalue) as 'Result';
     SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and the natural exponent (the constant e raised to the power of the number) between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
     SELECT @testvalue as 'Value', SQRT(@testvalue) as 'Result';
```

```
    SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and their square roots between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
    SELECT @testvalue as 'Value', SQUARE(@testvalue) as 'Result';
    SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and their squares between 1 and 20.

You should not use the statement:

```
DECLARE @testvalue float;
SET @testvalue = 1.00;
WHILE @testvalue < 21.00
   BEGIN
    SELECT @testvalue as 'Value', LOG(@testvalue) as 'Result';
    SET @testvalue = @testvalue +1;
  END;
```

This statement outputs a list of numbers and their natural logarithms between 1 and 20.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

EXP (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179857.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 79 (461P_1.2_04)

You are creating an important view that a web application will use to extract data from a SQL Server 2012 database. You want to ensure that the underlying table structure referenced by the view cannot be altered in a way that would affect the view definition without someone first dropping the view.

Which of the following options would you use with the CREATE VIEW statement?

1. ENCRYPTION
2. >>SCHEMABINDING <Correct>
3. VIEW_METADATA
4. CHECK OPTION

Explanation:
SCHEMABINDING ensures that the base table or tables cannot be modified in any way that would affect the view definition.

You would not use CHECK OPTION. This option forces all statements that modify data executed against the view to follow criteria set within the statement that defines the view.

You would not use ENCRYPTION. This option encrypts the text of the CREATE VIEW statement. A side effect of using ENCRYPTION is that the view cannot be published as part of SQL Server replication.

You would not use VIEW_METADATA. When you use this option, browse mode metadata will return the view name and not the underlying table names when it lists columns from the view in the result set.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187956.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 80 (461P_3.3_09)

The Hovercraft database has three tables related to Hovercraft design, OldDesign, CurrentDesign, and FutureDesign. Some designs are present in more than one table. You want to find items that are present in the OldDesign and CurrentDesign tables but not present in the FutureDesign table.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

2. >>SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign <Correct>

3. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

4. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

Explanation:
You should use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in the OldDesign and CurrentDesign tables but not present in the FutureDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in all three tables. You want to find items that are present in the OldDesign and CurrentDesign tables but not present in the FutureDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the FutureDesign table as well as FutureDesign items that are present in both the OldDesign and CurrentDesign tables. You want to find items that are present in the OldDesign and CurrentDesign table but not present in the FutureDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the OldDesign table as well as OldDesign items that are present in both the CurrentDesign and FutureDesign tables. You want to find items that are present in the OldDesign and CurrentDesign tables but not present in the FutureDesign table.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

EXCEPT and INTERSECT (Transact-SQL)

MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188055.aspx

Question Number (ID) : 81 (461P_2.1_08)

You have been examining the contents of the Sales.SpecialOffer table in the AdventureWorks2012 database. You have noticed that many values in the MaxQty column are set to NULL. You want to output some data from this table. Rather than outputting NULL values, you want to replace any value that is set to NULL with 0.

Which of the following Transact-SQL statements can you use to accomplish this goal? (Each correct answer presents a complete solution. Choose two.)

1. SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, MaxQty FROM Sales.SpecialOffer where MaxQty IS NOT NULL;

2. SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, MaxQty FROM Sales.SpecialOffer where MaxQty IS NULL;

3. >>SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, COALESCE(MaxQty, 0) as 'MaxQty' FROM Sales.SpecialOffer; <Correct>

4. >>SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, ISNULL(MaxQty, 0) as 'MaxQty' FROM Sales.SpecialOffer; <Correct>

Explanation:
You can use the SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, ISNULL(MaxQty, 0) as 'MaxQty' FROM Sales.SpecialOffer; statement. The output of this statement will replace all NULL values in the MaxQty column with the value 0.

You can use the SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, COALESCE(MaxQty, 0) as 'MaxQty' FROM Sales.SpecialOffer; statement. The output of this statement will replace all NULL values in the MaxQty column with the value 0.

You should not use the SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, MaxQty FROM Sales.SpecialOffer where MaxQty IS NULL; statement. This statement will list items where the MaxQty is NULL rather than replacing the output of null values with 0.

You should not use the SELECT SpecialOfferID, Description, Category, DiscountPct, MinQty, MaxQty FROM Sales.SpecialOffer where MaxQty IS NOT NULL; statement. This statement will list items where the MaxQty is not NULL rather than replacing the output of null values with 0.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

COALESCE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190349.aspx

ISNULL (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms184325.aspx

Question Number (ID) : 82 (461P_1.4_07)

You are modifying a table. The Inventory.Chassis table in the Hovercraft database has a primary key constraint defined on the ID column. You want to ensure that values in the Registration column contain no duplicates.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER TABLE Inventory.Chassis ADD CONSTRAINT PK_Registration PRIMARY KEY (Registration)
2. >>ALTER TABLE Inventory.Chassis ADD CONSTRAINT UNQ_Registration UNIQUE (Registration) <Correct>
3. ALTER TABLE Inventory.Chassis ADD CONSTRAINT PK_Registration FOREIGN KEY (Registration)
4. ALTER TABLE Inventory.Chassis ADD CONSTRAINT PK_Registration CHECK (Registration)

Explanation:
You should use the ALTER TABLE Inventory.Chassis ADD CONSTRAINT UNQ_Registration UNIQUE (Registration) statement. This statement creates a UNIQUE constraint that ensures that values in the Registration column contain no duplicates.

You should not use the ALTER TABLE Inventory.Chassis ADD CONSTRAINT PK_Registration PRIMARY KEY (Registration) statement. A primary key constraint already exists on the table, and it is only possible to have one foreign key constraint on a table.

You should not use the ALTER TABLE Inventory.Chassis ADD CONSTRAINT FK_Registration FOREIGN KEY (Registration) statement. This statement creates a foreign key constraint when the scenario in the question indicates a unique constraint is necessary.

You should not use the ALTER TABLE Inventory.Chassis ADD CONSTRAINT CK_Registration CHECK (Registration) statement. This statement creates a check constraint when the scenario in the question indicates a unique constraint is necessary.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Unique Constraints and Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187550.aspx

Create Unique Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190024.aspx

Question Number (ID) :   83 (461P_3.4_16)

Which of the following queries will return the result 'australia'?

```
1.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  REPLACE(@testvalue,'alia','ia');
```

```
2.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  LEN(@testvalue);
```

```
3.  >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  RTRIM(@testvalue);          <Correct>
```

```
4.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  STUFF(@testvalue,4,7,'picious');
```

Explanation:
The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RTRIM(@testvalue);
```

will remove all trailing spaces from the string in @testvalue. This will return the result 'australia' because @testvalue is set to a value with no trailing spaces.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEN(@testvalue);
```

will measure the length of the string in @testvalue, which is nine characters in length. It will return the result '9'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REPLACE(@testvalue,'alia','ia');
```

will replace the string 'alia' in the string in @testvalue with 'ia'. This will return the result 'austria'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT STUFF(@testvalue,4,7,'picious');
```

will insert the string 'picious' into the string in @testvalue, replacing seven characters, starting with the fourth character. This will return the result 'auspicious'.

Objective:
Modify Data

Sub Objective(s):

Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

RTRIM (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178660.aspx

Question Number (ID) : 84 (461P_4.1_04)

You have determined that a particular JOIN will work more efficiently if you use a MERGE join.

Which of the following Transact-SQL statements contains the appropriate query hint to accomplish this goal?

```
1.  >>SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER MERGE JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;     <Correct>
```

```
2.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

```
3.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER HASH JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

```
4.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER LOOP JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

Explanation:
The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER MERGE JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the MERGE JOIN hint and will use the MERGE join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

uses no join hint. You need to specify the hint if you want to force the use of a particular join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER HASH JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the HASH JOIN hint and will use the HASH join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER LOOP JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the LOOP JOIN hint and will use the LOOP join method.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Join Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173815.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

Question Number (ID) : 85 (461P_4.1_06)

You have determined that a particular JOIN will work more efficiently if you use a LOOP join.

Which of the following Transact-SQL statements contains the appropriate query hint to accomplish this goal?

```
1.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER HASH JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

```
2.  >>SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER LOOP JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;    <Correct>
```

```
3.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER MERGE JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

```
4.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

Explanation:
The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER LOOP JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the LOOP JOIN hint and will use the LOOP join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER MERGE JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the MERGE JOIN hint and will use the MERGE join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

uses no join hint. You need to specify the hint if you want to force the use of a particular join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER HASH JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the HASH JOIN hint and will use the HASH join method.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Join Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173815.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 86 (461P_2.5_06)

You execute the following statement:

```
declare @tempDoc xml
set @tempDoc = '<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>';
```

Which of the following queries would you have to append to this statement to generate the following result?

```
<Maintenance>5 year parts and labor extended maintenance </Maintenance>
```

1. >>SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance'); <Correct>

2. SELECT @tempDoc.query('/Root/ProductDescription/Features');

3. SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers');

4. SELECT @tempDoc.query('/Root');

Explanation:
The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance') produces the result:

```
<Maintenance>5 year parts and labor extended maintenance </Maintenance>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features') produces the result:

```
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
```

The query SELECT @tempDoc.query('/Root') produces the result:

```
<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers') produces the result:

```
<Passengers>60 passenger capacity</Passengers>
```

Objective:
Work with Data

Sub Objective(s):
Query and manage XML data.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 7 - Lesson 1

Query() Method (xml Data Type)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191474.aspx

XML Data (SQL Server)
MSDN
Link: http://msdn.microsoft.com/en-us/library/bb522446.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 87 (461P_1.1_01)

You are creating a number of tables for your company's Hovercraft Rental database. You want to create a table based on the properties of the columns and the data contained in rows output by a particular query. This new table should be automatically populated with the data that is generated by the query.

Which of the following T-SQL statements would you use to accomplish this goal?

1. ALTER TABLE

2. CREATE TABLE

3. DROP TABLE

4. >>SELECT . . . INTO <Correct>

Explanation:
You can use the SELECT . . . INTO statement to use the output from a select query to form the basis of creating a new table. When you do this, the new table's properties inherit the properties of the columns selected using the query.

Although you can use the CREATE TABLE statement to create a table, this will not automatically populate the table with data from a select query. This would require a separate operation.

You use the ALTER TABLE statement to modify the properties of a table. You cannot use the ALTER TABLE statement to create a new table.

You use the DROP TABLE statement to drop an existing table. You cannot use the DROP TABLE statement to create a new table.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements)

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

INTO Clause (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188029.aspx

CREATE TABLE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174979.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 88 (461P_3.4_06)

Which of the following mathematical functions would you use to return the smallest integer less than or equal to a specific numeric value?

1. CEILING

2. >>FLOOR <Correct>

3. ABS

4. POWER

Explanation:
You use the FLOOR function to return the smallest integer less than or equal to a specific numeric value.

You use the CEILING function to return the smallest integer greater than or equal to a specific numeric value.

You use the POWER function to raise a numeric value to a specific exponent.

You use the ABS function to determine the absolute value of a specific numeric value.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

FLOOR (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178531.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 89 (461P_1.3_08)

Which of the following statements about modifying data through a view are true? (Each correct answer presents a complete solution. Choose three.)

1. DELETE statements must reference columns from only one base table. <Correct>
2. Columns that use the COUNT aggregate function can be updated.
3. >>UPDATE statements must reference columns from only one base table. <Correct>
4. >>INSERT statements must reference columns from only one base table. <Correct>

Explanation:
You can use UPDATE statements against views only if they reference columns from one base table.

You can use INSERT statements against views only if they reference columns from one base table.

You can use DELETE statements against views only if they reference columns from one base table.

You cannot update columns that use derived values such as count.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 90 (461P_3.4_07)

Which of the following mathematical functions would you use to calculate the absolute numeric value of a numeric value?

1. POWER
2. FLOOR
3. CEILING
4. >>ABS <Correct>

Explanation:
You use the ABS function to determine the absolute value of a specific numeric value.

You use the CEILING function to return the smallest integer greater than or equal to a specific numeric value.

You use the FLOOR function to return the smallest integer less than or equal to a specific numeric value.

You use the POWER function to raise a numeric value to a specific exponent.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

ABS (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189800.aspx

Question Number (ID) : 91 (461P_3.2_09)

Which of the following statements can you use to delete all orders from the Hovercraft.Orders table where the productID is 343 and the shipping country is Australia?

1. DELETE FROM Hovercraft.Orders WHERE productID = 343 OR NOT shippingcountry = N'Australia'
2. DELETE FROM Hovercraft.Orders WHERE productID = 343 AND NOT shippingcountry = N'Australia'
3. DELETE FROM Hovercraft.Orders WHERE productID = 343 OR shippingcountry = N'Australia'
4. >>DELETE FROM Hovercraft.Orders WHERE productID = 343 AND shippingcountry = N'Australia' <Correct>

Explanation:
You should use the DELETE FROM Hovercraft.Orders WHERE productID = 343 AND shippingcountry = N'Australia' statement. This statement will delete data when the product ID is set to 343 and the shipping country is set to Australia.

You should not use the DELETE FROM Hovercraft.Orders WHERE productID = 343 OR shippingcountry = N'Australia' statement. This will delete data where the product ID is 343, independent of what the shipping country is. This statement will also delete data when the shipping country = Australia, independent of the product id.

You should not use the DELETE FROM Hovercraft.Orders WHERE productID = 343 AND NOT shippingcountry = N'Australia' statement. This statement will delete all data where the product ID is set to 343 except when the shipping country is Australia.

You should not use the DELETE FROM Hovercraft.Orders WHERE productID = 343 OR NOT shippingcountry = N'Australia' statement. This statement will delete all data where the product ID is 343. It will also delete all data where the shipping country is not Australia.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

DELETE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189835.aspx

NOT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189455.aspx

Question Number (ID) : 92 (461P_1.3_02)

There are several very large tables in the Hovercraft database. You want to create a database object that users in the Australian office can issue select queries against that will return only results that have the Country column of the Sales table set to the value Australia.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. CREATE PROCEDURE
2. >>CREATE VIEW <Correct>
3. DROP VIEW
4. DROP PROCEDURE

Explanation:
You should use the CREATE VIEW statement. Users can execute select queries against views. You could create a view on the Sales table that was defined so that the Country column equaled the value Australia.

You should not use the CREATE PROCEDURE statement. Users cannot execute select queries against stored procedures.

You should not use the DROP VIEW statement. The DROP VIEW statement allows you to remove an existing view. You want to create a new view.

You should not use the DROP PROCEDURE statement. This statement allows you to remove a stored procedure. You want to create a new view.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 93 (461P_3.1_18)

The stored procedure HumanResources.uspEmployeeEmail currently returns an employee's email address, first name, and last name. You want to modify this stored procedure so that it also outputs an employee's phone number.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>ALTER PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;    <Correct>
```

```
2.  ALTER PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

```
3.  ALTER PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

```
4.  ALTER PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @LastName AND LastName = @FirstName;
```

Explanation:
You should use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement modifies the HumanResources.uspEmployeeEmail stored procedure so that it outputs an employee's first name, last name, email address, and phone number.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement modifies the stored procedure, but the modified stored procedure outputs only first name, last name, and email address. You want to modify the HumanResources.uspEmployeeEmail stored procedure so that it outputs an employee's first name, last name, email address, and phone number.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement modifies the stored procedure, but the modified stored procedure outputs only the first name, last name, and phone number. You want to modify the HumanResources.uspEmployeeEmail stored procedure so that it outputs an employee's first name, last name, email address, and phone number.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeEmail
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EmailAddress, PhoneNumber
   FROM HumanResources.vEmployee
   WHERE FirstName = @LastName AND LastName = @FirstName;
```

This statement has an error in the last line that will block the stored procedure from functioning properly. This line mixes the FirstName field with the LastName parameter and the LastName field with the FirstName parameter.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

ALTER PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms189762.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

Question Number (ID) : 94 (461P_4.4_01)

You want to determine the specific line in a block of Transact-SQL code that caused an error. You configure an appropriate TRY . . . CATCH construct, using the code so that you can further diagnose the error.

Which of the following system functions can you use to determine this information?

1. ERROR_PROCEDURE()
2. ERROR_STATE()
3. >>ERROR_LINE() <Correct>
4. ERROR_MESSAGE()
5. ERROR_NUMBER()
6. ERROR_SEVERITY()

Explanation:
You can use the ERROR_LINE() system function to determine the line number at which the error occurred.

You can use the ERROR_MESSAGE() system function to determine the text of the message that is returned to the application.

You can use the ERROR_NUMBER() system function to determine the error number.

You can use the ERROR_PROCEDURE() system function to determine the name of the trigger of stored procedure in which the error occurred.

You can use the ERROR_SEVERITY() system function to determine the severity of the error.

You can use the ERROR_STATE() system function to determine the error state number. An error state number differs from an error number. For example, the 1105 error can have a state of 0, 1, 2 or 3.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Implement error handling.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 2

ERROR_LINE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178600.aspx

Retrieving Error Information in Transact-SQL
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179495(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 95 (461P_4.1_02)

You are optimizing several complex queries. You have determined that you can improve performance by hashing UNION sets.

Which of the following query hints should you use to implement this optimization?

1. CONCAT UNION
2. HASH JOIN
3. >>HASH UNION <Correct>
4. MERGE UNION

Explanation:
The HASH UNION query hint option specifies that all UNION operations will be performed by hashing UNION sets.

The CONCAT UNION query hint option specifies that all UNION operations will be performed by concatenating UNION sets.

The MERGE UNION query hint option specifies that all UNION operations will be performed by merging UNION sets.

The HASH JOIN query hint option specifies that all JOIN operations in the query use HASH JOINs.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Query Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181714.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 96 (461P_4.1_10)

You are optimizing several Transact-SQL statements. You have determined that a particular statement will perform more efficiently if a lock is applied at the page level rather than at the table or row level.

Which of the following table hints will you use in your Transact-SQL statements when performing optimization?

1. ROWLOCK
2. >>PAGLOCK <Correct>
3. READPAST
4. TABLOCK

Explanation:
When you use the PAGLOCK table query hint, a lock is placed at the page level rather than at the row or table level.

When you use the READPAST table query hint, rows in the table locked by other transactions are skipped rather than blocking the transaction.

When you use the ROWLOCK table query hint, row locks will be used when page or table locks are likely to be used.

When you use the TABLOCK table query hint, a lock is applied at the table level rather than at the row or page level.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Table Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187373.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 97 (461P_4.2_08)

Which of the following statements would you use to ensure that the SQL Server database engine automatically rolls back the entire current transaction if a Transact-SQL statement raises a run-time error?

1. >>SET XACT_ABORT ON <Correct>
2. SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
3. SET TRANSACTION ISOLATION LEVEL READ COMMITTED
4. SET XACT_ABORT OFF

Explanation:
You use the SET XACT_ABORT ON statement to ensure that when a Transact-SQL statement run-time error occurs, the entire transaction is terminated and rolled back.

You use the SET XACT_ABORT OFF to allow a Transact-SQL statement to cause low-severity errors without rolling back the entire transaction.

You use the SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED statement when you want to allow statements to read rows that have been changed by other transactions but have yet to be committed.

You use the SET TRANSACTION ISOLATION LEVEL READ COMMITTED statement when you want to ensure that statements cannot read data that has been changed but not committed by other transactions.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SET XACT_ABORT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188792.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

Question Number (ID) : 98 (461P_3.2_07)

You run the following query against a table in the Hovercraft database:

```
SELECT *
FROM Hovercraft.Sales
WHERE orderid = 3802;
```

You get the output shown in the exhibit.

The discount on the product with unitid 1 is 12.5%. You want to update this table so that you increase the discount assigned to each item in this order by an additional 5%.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. UPDATE Hovercraft.Sales SET discount /= 0.05 WHERE orderid = 3802

2. UPDATE Hovercraft.Sales SET discount = 0.05 WHERE orderid = 3802

3. >>UPDATE Hovercraft.Sales SET discount += 0.05 WHERE orderid = 3802 <Correct>

4. UPDATE Hovercraft.Sales SET discount *= 0.05 WHERE orderid = 3802

Explanation:
You should use the UPDATE Hovercraft.Sales SET discount += 0.05 WHERE orderid = 3802 statement. This will add an additional 5% to each discount for this orderid.

You should not use the UPDATE Hovercraft.Sales SET discount = 0.05 WHERE orderid = 3802 statement. This will set the discount for each item for this orderid to 5% rather than adding an additional discount to each item.

You should not use the UPDATE Hovercraft.Sales SET discount *= 0.05 WHERE orderid = 3802 statement. This will multiply the existing discount by 5% rather than adding 5% to the original discount.

You should not use the UPDATE Hovercraft.Sales SET discount /= 0.05 WHERE orderid = 3802 statement. This will divide the existing discount by 5% rather than adding 5% to the original discount.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

UPDATE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177523.aspx

WHERE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188047.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 99 (461P_1.3_09)

You want to use the HumanResources.vEmployeeDepartmentHistory view to update the end date of employee Terri Duffy to 10/10/2012.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>UPDATE HumanResources.vEmployeeDepartmentHistory
SET EndDate = '20121010'
WHERE LastName = N'Duffy' AND FirstName = 'Terri';                        <Correct>
```

```
2.  UPDATE HumanResources.vEmployeeDepartmentHistory
SET EndDate = '20121010'
WHERE LastName = N'Terri' AND FirstName = 'Duffy';
```

```
3.  UPDATE HumanResources.vEmployeeDepartmentHistory
SET StartDate = '20121010'
WHERE LastName = N'Duffy' AND FirstName = 'Terri';
```

```
4.  UPDATE HumanResources.vEmployeeDepartmentHistory
SET StartDate = '20121010'
WHERE LastName = N'Terri' AND FirstName = 'Duffy';
```

Explanation:
You should use the statement:

```
UPDATE HumanResources.vEmployeeDepartmentHistory
SET EndDate = '20121010'
WHERE LastName = N'Duffy' AND FirstName = 'Terri';
```

This uses the HumanResources.vEmployeeDepartmentHistory view to update the end date associated with Terri Duffy's record.

You should not use the statement:

```
UPDATE HumanResources.vEmployeeDepartmentHistory
SET StartDate = '20121010'
WHERE LastName = N'Duffy' AND FirstName = 'Terri';
```

This sets the start date rather than the end date associated with Terri Duffy's record.

You should not use the statement:

```
UPDATE HumanResources.vEmployeeDepartmentHistory
SET EndDate = '20121010'
WHERE LastName = N'Terri' AND FirstName = 'Duffy';
```

This statement incorrectly uses Terri as the last name and Duffy as the first name.

You should not use the statement:

```
UPDATE HumanResources.vEmployeeDepartmentHistory
SET StartDate = '20121010'
WHERE LastName = N'Terri' AND FirstName = 'Duffy';
```

This statement incorrectly uses Terri as the last name and Duffy as the first name. It also sets the start date rather than the end date.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx

Modify Data Through a View
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms180800.aspx

Question Number (ID) : 100 (461P_3.3_05)

Your database has two tables related to product models, named Hovercraft.OldDesign and Hovercraft.NewDesign. You want to produce a list of Hovercraft models from the NewDesign table as long as those Hovercraft models do not have duplicate entries in the OldDesign table, and you want the list to be ordered by Name.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

2. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION ALL SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

3. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

4. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

5. >>SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign ORDER BY Name <Correct>

Explanation:
You should use the SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign ORDER BY Name statement. This will produce a list of Hovercraft models by name from the NewDesign table as long as those Hovercraft models do not have duplicate entries in the OldDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of all Hovercraft models in both tables without duplicate entries. You want to produce a list of Hovercraft models by name from the NewDesign table as long as those Hovercraft models do not have duplicate entries in the OldDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce only those entries that are in the output of both queries. You want to produce a list of Hovercraft models by name from the NewDesign table as long as those Hovercraft models do not have duplicate entries in the OldDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of Hovercraft models from the OldDesign table as long as those Hovercraft models do not have duplicate entries in the NewDesign table. You want to produce a list of Hovercraft models by name from the NewDesign table as long as those Hovercraft models do not have duplicate entries in the OldDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION ALL SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of all Hovercraft models in both tables but will output duplicate entries. You want to produce a list of Hovercraft models by name from the NewDesign table as long as those Hovercraft models do not have duplicate entries in the OldDesign table.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

Set Operators (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ff848745.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 101 (461P_1.4_04)

You want to remove the unique constraint named UNQ_Alpha from the Hovercraft table. This table is associated with the Rentals schema.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>ALTER TABLE Rentals.Hovercraft DROP CONSTRAINT UNQ_Alpha <Correct>
2. ALTER SCHEMA Hovercraft.Rentals DROP CONSTRAINT UNQ_Alpha
3. ALTER TABLE Hovercraft.Rentals DROP CONSTRAINT UNQ_Alpha
4. ALTER SCHEMA Rentals.Hovercraft DROP CONSTRAINT UNQ_Alpha

Explanation:
You should use the ALTER TABLE Rentals.Hovercraft DROP CONSTRAINT UNQ_Alpha statement. This statement drops the UNQ_Alpha unique constraint from the Hovercraft table in the Rentals schema.

You should not use the ALTER TABLE Hovercraft.Rentals DROP CONSTRAINT UNQ_Alpha statement. This statement drops the unique constraint from the Rentals table in the Hovercraft schema.

You should not use the ALTER SCHEMA Rentals.Hovercraft DROP CONSTRAINT UNQ_Alpha statement. You cannot use the ALTER SCHEMA statement to drop a unique constraint from a table.

You should not use the ALTER SCHEMA Hovercraft.Rentals DROP CONSTRAINT UNQ_Alpha statement. You cannot use the ALTER SCHEMA statement to drop a unique constraint from a table.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements)

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Delete Unique Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188260.aspx

Unique Constraints and Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187550.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 102 (461P_4.2_03)

You want to create several markers within a transaction that a user can return to without necessarily rolling back the entire transaction.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. ROLLBACK TRANSACTION
2. >>SAVE TRANSACTION <Correct>
3. COMMIT TRANSACTION
4. BEGIN TRANSACTION

Explanation:
You use the SAVE TRANSACTION statement to create a savepoint within a transaction. You can use the ROLLBACK TRANSACTION statement to roll back to a savepoint.

You use the BEGIN TRANSACTION statement to mark the start point of an explicit local transaction.

You use the COMMIT TRANSACTION statement to mark the end of a successful implicit or explicit transaction.

You use the ROLLBACK TRANSACTION statement to roll back an implicit or explicit transaction. You can use this statement to roll back to a savepoint or to the beginning of the transaction.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SAVE TRANSACTION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188378.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

Question Number (ID) : 103 (461P_4.2_12)

You are configuring an appropriate transaction isolation level for a session. You want to ensure the following:

*Data read by any statement in the transaction must be transactionally consistent with how that data existed at the start of the transaction.
*Other transactions should be able to write data to rows selected by the transaction even when the transaction has yet to commit.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. SET TRANSACTION ISOLATION LEVEL SNAPSHOT <Correct>
2. SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
3. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
4. >>SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
5. SET TRANSACTION ISOLATION LEVEL READ COMMITTED

Explanation:
You use the SET TRANSACTION ISOLATION LEVEL SNAPSHOT statement when you want to ensure that data read by any statement in a transaction will be transactionally consistent with how that data existed at the start of the transaction. SNAPSHOT transactions do not block other transactions from writing data.

You use the SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED statement when you want to allow statements to read rows altered by other transactions, but those transactions have not yet been committed.

You use the SET TRANSACTION ISOLATION LEVEL READ COMMITTED statement when you want to ensure that statements cannot read data altered by other transactions when those transactions have not yet been committed. When you set the READ COMMITTED transaction level, data can be altered by other transactions between individual statements within the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL REPEATABLE READ statement when you want to ensure that statements cannot read data altered by other transactions but not yet committed and that other transactions cannot modify data read by the current transaction until the current transaction commits. When the REPEATABLE READ isolation level is set, it is possible for other transactions to insert new rows that meet the search conditions of statements included in the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL SERIALIZABLE statement when you want to ensure that Transact-SQL statements cannot read data that has been altered but not committed, that other transactions cannot modify data read by the current transaction until the transaction commits, and that other transactions cannot insert new rows with key values that would fall into the range read by statements in the current transaction until that transaction completes.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SET TRANSACTION ISOLATION LEVEL (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173763.aspx

Transaction Statements (Transact-SQL)

MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 104 (461P_2.3_02)

Several columns of the Sales.Customer table of the AdventureWorks2012 database have the following properties:

CustomerID: int, not null.
StoreID: int, not null.
ModifiedDate: Datetime, not null

A query against the table generates the output shown in the exhibit under Before. You need to modify this query so that the output appears as shown in the exhibit under After.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>SELECT CustomerID, StoreID, CAST(ModifiedDate as nvarchar(4)) as ModifiedDate from Sales.Customer

2. SELECT CustomerID, StoreID, CAST(ModifiedDate as date) as ModifiedDate from Sales.Customer

3. SELECT CustomerID, StoreID, CAST(ModifiedDate as datetime) as ModifiedDate from Sales.Customer

4. SELECT CustomerID, StoreID, CAST(ModifiedDate as nvarchar(12)) as ModifiedDate from Sales.Customer
<Correct>

Explanation:
You should use the SELECT CustomerID, StoreID, CAST(ModifiedDate as nvarchar(12)) as ModifiedDate from Sales.Customer statement. This statement correctly outputs the ModifiedDate column in the appropriate format.

You should not use the SELECT CustomerID, StoreID, CAST(ModifiedDate as nvarchar(4)) as ModifiedDate from Sales.Customer statement. This statement will incorrectly output the ModifiedDate column so that only the month will be displayed.

You should not use the SELECT CustomerID, StoreID, CAST(ModifiedDate as datetime) as ModifiedDate from Sales.Customer statement. This statement will output the ModifiedDate column as shown in the before exhibit rather than as shown in the after exhibit.

You should not use the SELECT CustomerID, StoreID, CAST(ModifiedDate as date) as ModifiedDate from Sales.Customer statement. This statement will output the ModifiedDate column in the 2008-10-13 format rather than Oct 13 2008 as shown in the exhibit.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:
Create DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190227.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 105 (461P_4.2_01)

Which of the following statements would you use to mark the start point of an explicit local transaction?

1. >>BEGIN TRANSACTION <Correct>
2. ROLLBACK TRANSACTION
3. COMMIT TRANSACTION
4. SAVE TRANSACTION

Explanation:
You use the BEGIN TRANSACTION statement to mark the start point of an explicit local transaction.

You use the COMMIT TRANSACTION statement to mark the end of a successful implicit or explicit transaction.

You use the SAVE TRANSACTION statement to create a savepoint within a transaction.

You use the ROLLBACK TRANSACTION statement to roll back an implicit or explicit transaction. You can use this statement to roll back to a savepoint or to the beginning of the transaction.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

BEGIN TRANSACTION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188929.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 106 (461P_4.4_04)

You are working with some Transact-SQL code that causes an error. You want to determine the trigger or stored procedure that caused a specific error. You configure the appropriate TRY . . . CATCH construct, using the code so that you can further diagnose the error.

Which of the following system functions can you use to determine this information?

1. ERROR_STATE()
2. >>ERROR_PROCEDURE() <Correct>
3. ERROR_NUMBER()
4. ERROR_MESSAGE()
5. ERROR_SEVERITY()
6. ERROR_LINE()

Explanation:
You can use the ERROR_PROCEDURE() system function to determine the name of the trigger or stored procedure in which the error occurred.

You can use the ERROR_LINE() system function to determine the line number at which the error occurred.

You can use the ERROR_MESSAGE() system function to determine the text of the message that is returned to the application.

You can use the ERROR_NUMBER() system function to determine the error number.

You can use the ERROR_SEVERITY() system function to determine the severity of the error.

You can use the ERROR_STATE() system function to determine the error state number. An error state number differs from an error number. For example, the 1105 error can have a state of 0, 1, 2 or 3.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Implement error handling.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 2

ERROR_PROCEDURE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188398.aspx

Retrieving Error Information in Transact-SQL
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179495(SQL.105).aspx

Question Number (ID) : 107 (461P_1.4_03)

You want to create a unique constraint on the Alpha column of the existing Exemplar table. This table is associated with the Person schema. The Alpha column is configured to disallow nulls.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. CREATE TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha UNIQUE (Alpha)
2. ALTER TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha UNIQUE (Alpha) <Correct>
3. CREATE TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha PRIMARY KEY (Alpha)
4. >>ALTER TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha PRIMARY KEY (Alpha)

Explanation:
You should use the ALTER TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha UNIQUE (Alpha) statement. This statement creates a unique constraint named AK_Alpha on the Alpha column of the Person.Exemplar table.

You should not use the ALTER TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha PRIMARY KEY (Alpha) statement. This statement creates a primary key constraint rather than a unique constraint.

You should not use the CREATE TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha PRIMARY KEY (Alpha) statement. You are adding a constraint to an existing table, so you should use the ALTER TABLE statement.

You should not use the CREATE TABLE Person.Exemplar ADD CONSTRAINT AK_Alpha UNIQUE (Alpha) statement. You are adding a constraint to an existing table, so you should use the ALTER TABLE statement.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements)

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Unique Constraints and Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187550.aspx

Create Unique Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190024.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 108 (461P_2.1_05)

The AdventureWorks2012 database has two tables that you want to query, Production.Product and Sales.SalesOrderDetail.

The Production.Product table has the following columns:

Name
ProductID

The Sales.SalesOrderDetail table has the following columns:

OrderQty
UnitPrice
UnitPriceDiscount
ProductID

You want to calculate the revenue for each product in each sales order.

Which of the following Transact-SQL statements can you use to accomplish this goal?

```
1.  >>SELECT 'Total income is', ((OrderQty * UnitPrice) * (1.0 - UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName ASC;
```

        <Correct>

```
2.  SELECT 'Total income is', (OrderQty * UnitPrice), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName ASC;
```

```
3.  SELECT 'Total income is', ((OrderQty * UnitPrice - UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName ASC;
```

```
4.  SELECT 'Total income is', ((OrderQty * UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName ASC;
```

Explanation:
You should use the following statement:

```
SELECT 'Total income is', ((OrderQty * UnitPrice) * (1.0 - UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
```

```
ORDER BY ProductName ASC;
```

Total income is calculated by multiplying the product of order quantity and unit price by (1 - unit price discount). This is represented in Transact-SQL as ((OrderQty * UnitPrice) * (1.0 - UnitPriceDiscount)).

You should not use the following statement:

```
SELECT 'Total income is', (OrderQty * UnitPrice), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName ASC;
```

This query incorrectly calculates total incoming by multiplying Order Quantity by Unit Price but not factoring in the discount.

You should not use the following statement:

```
SELECT 'Total income is', ((OrderQty * UnitPrice - UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName ASC;
```

This query incorrectly calculates total income by multiplying order quantity by unit price and then subtracting unit price discount. The discount applies on a per unit basis.

You should not use the following statement:

```
SELECT 'Total income is', ((OrderQty * UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName ASC;
```

This query incorrectly calculates total income by multiplying order quantity by unit price discount.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 109 (461P_3.4_11)

Which of the following queries that use string functions will return the result 'AUSTRALIA'?

```
1.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEFT(@testvalue, 3);
```

```
2.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RIGHT(@testvalue, 3);
```

```
3.  >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  UPPER(@testvalue);        <Correct>
```

```
4.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  REVERSE(@testvalue);
```

Explanation:
The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT UPPER(@testvalue);
```

will return the result 'AUSTRALIA' because it will convert all characters from the string 'australia' to uppercase.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEFT(@testvalue, 3)
```

will return the result 'aus' because it will return the first three characters from the left of the string 'australia'.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REVERSE(@testvalue);
```

will return the result 'ailartsua' because it will reverse all the characters from the string 'australia'.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RIGHT(@testvalue, 3);
```

will return the result 'lia' because it will return the three characters from the string from the right.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

UPPER (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180055.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 110 (461P_4.3_03)

Which of the following Transact-SQL statements do you use if you want to retrieve rows from a result set allocated to a cursor?

1. DEALLOCATE
2. >>FETCH <Correct>
3. CLOSE
4. OPEN

Explanation:
You use the FETCH statement to retrieve a specific row from a CURSOR.

You use the DEALLOCATE statement to remove a cursor reference.

You use the OPEN statement to open a Transact-SQL cursor. The cursor is populated by the results of the Transact-SQL statement used when the cursor is created.

You use the CLOSE statement to release the current result set. This releases cursor locks on rows on which the cursor is positioned.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Evaluate the use of row-based operations vs. set-based operations.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 16 - Lesson 1

Fetch (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180152.aspx

Cursors (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181441.aspx

Question Number (ID) : 111 (461P_3.4_15)

Which of the following queries will return the result 'auspicious'?

1. >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT STUFF(@testvalue,4,7,'picious'); <Correct>

2. DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RTRIM(@testvalue);

3. DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEN(@testvalue);

4. DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REPLACE(@testvalue,'alia','ia');

Explanation:
The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT STUFF(@testvalue,4,7,'picious');
```

will insert the string 'picious' into the string in @testvalue, replacing seven characters, starting with the fourth character. This will return the result 'auspicious'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEN(@testvalue);
```

will measure the length of the string in @testvalue, which is nine characters in length. It will return the result '9'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REPLACE(@testvalue,'alia','ia');
```

will replace the string 'alia' in the string in @testvalue with 'ia'. This will return the result 'austria'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RTRIM(@testvalue);
```

will remove all trailing spaces from the string in @testvalue. This will return the result 'australia' because @testvalue is set to a value with no trailing spaces.

Objective:
Modify Data

Sub Objective(s):

Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

STUFF (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188043.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 112 (461P_3.1_05)

You want to create a stored procedure associated with the HumanResources schema that queries the HumanResources.vJobCandidate view to generate a list of candidates that includes their first names, last names, and skills.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
  SET NOCOUNT ON;
  SELECT [Name.First], [Name.Last], Skills
  FROM  HumanResources.vJobCandidate;                <Correct>

2.  CREATE PROCEDURE Person.uspCandidateSkills
AS
  SET NOCOUNT ON;
  SELECT [Name.First], [Name.Last], Skills
  FROM  HumanResources.vJobCandidate;

3.  CREATE PROCEDURE Person.uspCandidateSkills
AS
  SET NOCOUNT ON;
  SELECT JobCandidateID, [Name.Last], Skills
  FROM  HumanResources.vJobCandidate;

4.  CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
  SET NOCOUNT ON;
  SELECT JobCandidateID, [Name.Last], Skills
  FROM  HumanResources.vJobCandidate;
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
  SET NOCOUNT ON;
  SELECT [Name.First], [Name.Last], Skills
  FROM HumanResources.vJobCandidate;
```

This statement creates a stored procedure associated with the HumanResources schema that returns a candidate's first name, last name, and skills.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
  SET NOCOUNT ON;
  SELECT JobCandidateID, [Name.Last], Skills
  FROM HumanResources.vJobCandidate;
```

This statement creates a stored procedure associated with the HumanResources schema that returns a candidate's job candidate ID, last name, and skills. You want to create a stored procedure associated with the HumanResources schema that returns a candidate's first name, last name, and skills.

You should not use the statement:

```
CREATE PROCEDURE Person.uspCandidateSkills
AS
  SET NOCOUNT ON;
```

```
   SELECT [Name.First], [Name.Last], Skills
   FROM HumanResources.vJobCandidate;
```

This statement creates a stored procedure associated with the Person schema that returns a candidate's first name, last name, and skills. You want to create a stored procedure associated with the HumanResources schema that returns a candidate's first name, last name, and skills.

You should not use the statement:

```
CREATE PROCEDURE Person.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT JobCandidateID, [Name.Last], Skills
   FROM HumanResources.vJobCandidate;
```

This statement creates a stored procedure associated with the Person schema that returns a candidate's job candidate ID, last name, and skills. You want to create a stored procedure associated with the HumanResources schema that returns a candidate's first name, last name, and skills.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 113 (461P_1.1_02)

You want to create a new table, named HumanResources.TestTable, in the AdventureWorks 2012 database. This table should have two columns, Name and GroupName, and be of the same data type as those with the same name in the HumanResources.Department table. The new table should be populated with the column data present in the HumanResources.Department table.

Which of the following statements could you execute to accomplish this goal?

1. >>SELECT * INTO HumanResources.TestTable FROM HumanResources.Department

2. SELECT Name, GroupName INTO HumanResources.Department FROM HumanResources.TestTable

3. SELECT * INTO HumanResources.Department FROM HumanResources.TestTable

4. SELECT Name, GroupName INTO HumanResources.TestTable FROM HumanResources.Department
<Correct>

Explanation:
You should use the SELECT Name, GroupName INTO HumanResources.TestTable FROM HumanResources.Department statement. This statement will create a new table named HumanResources.TestTable and populate it with information created from the Name and GroupName columns of the HumanResources.Department table.

You should not use the SELECT * INTO HumanResources.TestTable FROM HumanResources.Department statement. This statement will create the correct table, but this table will be a duplicate of the HumanResources.Department table rather than just using the Name and GroupName columns.

You should not use the SELECT Name, GroupName INTO HumanResources.Department FROM HumanResources.TestTable statement. This statement will create a new table named HumanResources.Department based on the contents of columns in the HumanResources.TestTable table.

You should not use the SELECT * INTO HumanResources.Department FROM HumanResources.TestTable statement. This statement will create a new table named HumanResources.Department based on the contents of the HumanResources.TestTable table.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

INTO Clause (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188029.aspx

CREATE TABLE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174979.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 114 (461P_3.3_04)

Your database has two tables related to Hovercraft models, named Hovercraft.OldDesign and Hovercraft.NewDesign. You want to produce a list of Hovercraft models from the OldDesign table as long as those Hovercraft models do not have duplicate entries in the NewDesign table.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

2. SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign ORDER BY Name

3. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION ALL SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

4. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name <Correct>

5. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

Explanation:
You should use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of Hovercraft models from the OldDesign table as long as those Hovercraft models do not have duplicate entries in the NewDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign ORDER BY Name statement. This will produce a list of Hovercraft models from the NewDesign table as long as those Hovercraft models do not have duplicate entries in the OldDesign table. You want to produce a list of Hovercraft models from the OldDesign table as long as those Hovercraft models do not have duplicate entries in the NewDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of all Hovercraft models in both tables without duplicate entries. You want to produce a list of Hovercraft models from the OldDesign table as long as those Hovercraft models do not have duplicate entries in the NewDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce only those entries that are in the output of both queries. You want to produce a list of Hovercraft models from the OldDesign table as long as those Hovercraft models do not have duplicate entries in the NewDesign table.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION ALL SELECT HovercraftModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of all Hovercraft models in both tables but will output duplicate entries. You want to produce a list of Hovercraft models from the OldDesign table as long as those Hovercraft models do not have duplicate entries in the NewDesign table.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

Set Operators (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ff848745.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 115 (461P_2.2_08)

You want to generate a list of products that are not in the Mountain Bikes subcategory of the AdventureWorks2012 database.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT Style FROM Production.Product WHERE NOT EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')

2. SELECT Name FROM Production.Product WHERE EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')

3. SELECT Style FROM Production.Product WHERE EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')

4. >>SELECT Name FROM Production.Product WHERE NOT EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')
   <Correct>

Explanation:
You should use the statement:

SELECT Name FROM Production.Product WHERE NOT EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')

This statement will generate a list of products that are not in the Mountain Bikes subcategory

You should not use the statement:

SELECT Name FROM Production.Product WHERE EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')

This statement will generate a list of products that are in the Mountain Bikes subcategory. You want to generate a list of products that are not in the Mountain Bikes subcategory.

You should not use the statement:

SELECT Style FROM Production.Product WHERE NOT EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')

This will generate a list of styles rather than products. You want to generate a list of products that are not in the Mountain Bikes subcategory.

You should not use the statement:

SELECT Style FROM Production.Product WHERE EXISTS (SELECT * FROM Production.ProductSubcategory WHERE ProductSubcategoryID = Production.Product.ProductSubcategoryID AND Name = 'Mountain Bikes')

This will generate a list of styles rather than products. You want to generate a list of products that are not in the Mountain Bikes subcategory.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

Subqueries with NOT EXISTS
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms184297(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 116 (461P_4.1_08)

You are optimizing several Transact-SQL statements. You have determined that a particular statement will perform more efficiently if a lock is applied at the row level rather than at the page or table level.

Which of the following table hints will you use in your Transact-SQL statements when performing optimization?

1. >>ROWLOCK <Correct>
2. PAGLOCK
3. READPAST
4. TABLOCK

Explanation:
When you use the ROWLOCK table query hint, row locks will be used when page or table locks are likely to be used.

When you use the READPAST table query hint, rows in the table locked by other transactions are skipped rather than blocking the transaction.

When you use the TABLOCK table query hint, a lock is applied at the table level rather than at the row or page level.

When you use the PAGLOCK table query hint, a lock is placed at the page level rather than at the row or table level.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Table Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187373.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 117 (461P_2.4_01)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued vacation hours. You want to determine the average vacation hours accrued by employees who have the job title starting with Design Engineer.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT VAR(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
2.  >>SELECT AVG(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';               <Correct>
```

```
3.  SELECT MIN(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
4.  SELECT MAX(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

Explanation:
You should use the statement:

```
SELECT AVG(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement returns the average vacation hours assigned to employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT MIN(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement returns the minimum vacation hours assigned to any employee who has the job title Design Engineer. You want to have the statement return the average vacation hours assigned to employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT MAX(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement returns the maximum vacation hours assigned to any employee who has the job title Design Engineer. You want to have the statement return the average vacation hours assigned to employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT VAR(VacationHours) AS 'Average Vacation Hours'
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement returns the statistical variance across vacation hours assigned to employees who have the job title Design Engineer. You want to have the statement return the average vacation hours assigned to employees who have the job title Design Engineer.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

AVG (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177677.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 118 (461P_4.1_05)

You have determined that a particular JOIN will work more efficiently if you use a HASH join.

Which of the following Transact-SQL statements contains the appropriate query hint to accomplish this goal?

```
1.  >>SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER HASH JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;    <Correct>
```

```
2.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER MERGE JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

```
3.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER LOOP JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

```
4.  SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER JOIN Purchasing.PurchaseOrderDetail AS pd
   ON  ph.PurchaseOrderID  =  pd.PurchaseOrderID;
```

Explanation:
The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER HASH JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the HASH JOIN hint and will use the HASH join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER MERGE JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the MERGE JOIN hint and will use the MERGE join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

uses no join hint. You need to specify the hint if you want to force the use of a particular join method.

The statement:

```
SELECT ph.PurchaseOrderID, ph.OrderDate, pd.ProductID, pd.DueDate, ph.VendorID
FROM Purchasing.PurchaseOrderHeader AS ph
INNER LOOP JOIN Purchasing.PurchaseOrderDetail AS pd
   ON ph.PurchaseOrderID = pd.PurchaseOrderID;
```

contains the LOOP JOIN hint and will use the LOOP join method.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Join Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173815.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

Question Number (ID) : 119 (461P_2.4_15)

By querying the HumanResources.EmployeePayHistory table of the AdventureWorks2012 database, you want to create a list of five groups of employees who are paid every two weeks based on pay rate. The first 20% should be in group one, the second 20% in group two, and so on.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
2.  >>SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;            <Correct>
```

```
3.  SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
4.  SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

Explanation:
You should use the statement:

```
SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

The NTILE() function partitions the results into equal sections based on the number you use with the NTILE function. This query would partition the results into five ranks.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement outputs results by using the RANK function. When you use the RANK function and multiple results share the same rank, those results are assigned that rank. The next result is assigned the rank based on how many rows there are after the first row. Hence, ranking results can be of the sequence 1,2,2,2,5 rather than of 1,2,2,2,3.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use DENSE_RANK, the statement ranks output and the same results share the same rank; however, the results do not appear in the same way as they do with the RANK statement. The next result, when you use DENSE_RANK, sequentially follows the last RANK, for example, 1,2,2,2,3 rather than 1,2,2,2,5.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use the ROW_NUMBER function, ranks are assigned sequentially, even when the result is the same. Instead of 1,2,2,2,5, by which the second, third, and fourth results are the same, the result would be 1,2,3,4,5.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

NTILE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175126.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 120 (461P_3.4_13)

Which of the following queries will return the result '9'?

```
1.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  STUFF(@testvalue,4,7,'picious');
```

```
2.  >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  LEN(@testvalue);             <Correct>
```

```
3.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  RTRIM(@testvalue);
```

```
4.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  REPLACE(@testvalue,'alia','ia');
```

Explanation:
The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEN(@testvalue);
```

will measure the length of the string in @testvalue, which is nine characters in length. It will return the result '9'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REPLACE(@testvalue,'alia','ia');
```

will replace the string 'alia' in the string in @testvalue with 'ia'. This will return the result 'austria'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RTRIM(@testvalue);
```

will remove all trailing spaces from the string in @testvalue. This will return the result 'australia' because @testvalue is set to a value with no trailing spaces.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT STUFF(@testvalue,4,7,'picious');
```

will insert the string 'picious' into the string in @testvalue, replacing seven characters, starting with the fourth character. This will return the result 'auspicious'.

Objective:
Modify Data

Sub Objective(s):

Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

LEN (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190329.aspx

Question Number (ID) : 121 (461P_2.3_04)

You want to have the output of the ModifiedDate column in the Production.WorkOrder column of the AdventureWorks2012 database listed in the Day/Month/Year format rather than in the Month/Day/Year format.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT WorkOrderID, CAST(ModifiedDate as varchar(12)) FROM Production.WorkOrder

2. SELECT WorkOrderID, Convert(varchar(12),ModifiedDate,101) FROM Production.Workorder

3. >>SELECT WorkOrderID, CAST(ModifiedDate as nvarchar(12)) FROM Production.WorkOrder

4. SELECT WorkOrderID, Convert(varchar(12),ModifiedDate,103) FROM Production.Workorder <Correct>

Explanation:
You should use the SELECT WorkOrderID, Convert(varchar(12),ModifiedDate,103) FROM Production.Workorder statement. This converts the output into the Day/Month/Year format rather than the Month/Day/Year format.

You should not use the SELECT WorkOrderID, Convert(varchar(12),ModifiedDate,101) FROM Production.Workorder statement. This statement outputs data in the Month/Day/Year format instead of in the Day/Month/Year format.

You should not use the SELECT WorkOrderID, CAST(ModifiedDate as varchar(12)) FROM Production.WorkOrder statement. This statement outputs data in the Month/Day/Year format instead of in the Day/Month/Year format.

You should not use the SELECT WorkOrderID, CAST(ModifiedDate as nvarchar(12)) FROM Production.WorkOrder statement. This statement outputs data in the Month/Day/Year format instead of in the Day/Month/Year format.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 2 - Lesson 2

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

CAST and CONVERT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187928.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 122 (461P_3.4_14)

Which of the following queries will return the result 'austria'?

```
1.  >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  REPLACE(@testvalue,'alia','ia');                    <Correct>
```

```
2.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  RTRIM(@testvalue);
```

```
3.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  LEN(@testvalue);
```

```
4.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  STUFF(@testvalue,4,7,'picious');
```

Explanation:
The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REPLACE(@testvalue,'alia','ia');
```

will replace the string 'alia' in the string in @testvalue with 'ia'. This will return the result 'austria'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEN(@testvalue);
```

will measure the length of the string in @testvalue, which is nine characters in length. It will return the result '9'.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RTRIM(@testvalue);
```

will remove all trailing spaces from the string in @testvalue. This will return the result 'australia' because @testvalue is set to a value with no trailing spaces.

The query:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT STUFF(@testvalue,4,7,'picious');
```

will insert the string 'picious' into the string in @testvalue, replacing seven characters, starting with the fourth character. This will return the result 'auspicious'.

Objective:
Modify Data

Sub Objective(s):

Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

REPLACE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms186862.aspx

Question Number (ID) : 123 (461P_4.1_09)

You are optimizing several Transact-SQL statements. You have determined that a particular statement will perform more efficiently if a lock is applied at the table level rather than at the page or row level.

Which of the following table hints will you use in your Transact-SQL statements when performing optimization?

1. >>TABLOCK <Correct>

2. ROWLOCK

3. PAGLOCK

4. READPAST

Explanation:
When you use the TABLOCK table query hint, a lock is applied at the table level rather than at the row or page level.

When you use the READPAST table query hint, rows in the table locked by other transactions are skipped rather than blocking the transaction.

When you use the ROWLOCK table query hint, row locks will be used when page or table locks are likely to be used.

When you use the PAGLOCK table query hint, a lock is placed at the page level rather than at the row or table level.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Optimize queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 14 - Lesson 1

Table Hints (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187373.aspx

Query Tuning
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176005(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 124 (461P_4.2_14)

You are configuring an appropriate transaction isolation level for a session. You want to ensure that statements in the transaction can read rows altered by other transactions even when those transactions have yet to be committed.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. >>SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED <Correct>
2. SET TRANSACTION ISOLATION LEVEL READ COMMITTED
3. SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
4. SET TRANSACTION ISOLATION LEVEL SNAPSHOT
5. SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

Explanation:
You use the SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED statement when you want to allow statements to read rows altered by other transactions before those transactions have been committed.

You use the SET TRANSACTION ISOLATION LEVEL READ COMMITTED statement when you want to ensure that statements cannot read data altered by other transactions before those transactions have been committed. When you set the READ COMMITTED transaction level, data can be altered by other transactions between individual statements within the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL REPEATABLE READ statement when you want to ensure that statements cannot read data altered by other transactions not yet committed and that other transactions cannot modify data read by the current transactions until the current transaction commits. When the REPEATABLE READ isolation level is set, it is possible for other transactions to insert new rows that meet the search conditions of statements included in the current transaction.

You use the SET TRANSACTION ISOLATION LEVEL SNAPSHOT statement when you want to ensure that data read by any statement in a transaction will be transactionally consistent with how that data existed at the start of the transaction. SNAPSHOT transactions do not block other transactions from writing data.

You use the SET TRANSACTION ISOLATION LEVEL SERIALIZABLE statement when you want to ensure that Transact-SQL statements cannot read data that has been altered but not committed, that other transactions cannot modify data read by the current transaction until the transaction commits, and other transactions cannot insert new rows with key values that would fall into the range read by statements in the current transaction until that transaction completes.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

SET TRANSACTION ISOLATION LEVEL (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173763.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 125 (461P_1.5_08)

You want to rename a trigger in a SQL Server 2012 database that you are responsible for managing.

Which of the following statements would you use to accomplish this goal? (Each correct answer presents part of a complete solution. Choose two.)

1. ENABLE TRIGGER
2. >>CREATE TRIGGER <Correct>
3. >>DROP TRIGGER <Correct>
4. ALTER TRIGGER

Explanation:
You must use the DROP TRIGGER and CREATE TRIGGER statements when renaming a trigger. You need to drop the original trigger and then re-create the trigger with the new name.

You cannot use the ALTER TRIGGER statement to change the name of a DML trigger. You use the ALTER TRIGGER statement to change the definition of a DML trigger.

You cannot use the ENABLE TRIGGER statement to change the name of a DML trigger. You use the ENABLE TRIGGER statement to enable a trigger that has been disabled.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Modify or Rename DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190671.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Question Number (ID) : 126 (461P_1.4_05)

You are modifying an existing SQL Server 2012 database. You need to remove a primary key constraint named PK_HovercraftID from the Registration table. This table is located in the Hovercraft schema of the Contoso2012 database. The constraint is applied on the HovercraftID column.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER COLUMN PK_HovercraftID DROP CONSTRAINT HovercraftID

2. ALTER COLUMN HovercraftID DROP CONSTRAINT PK_HovercraftID

3. ALTER TABLE Hovercraft.Registration DROP CONSTRAINT HovercraftID

4. >>ALTER TABLE Hovercraft.Registration DROP CONSTRAINT PK_HovercraftID <Correct>

Explanation:
You should use the ALTER TABLE Hovercraft.Registration DROP CONSTRAINT PK_HovercraftID statement. This statement drops the constraint PK_HovercraftID from the Registration table located on the Hovercraft schema.

You should not use the ALTER COLUMN HovercraftID DROP CONSTRAINT PK_HovercraftID statement. You cannot use the ALTER COLUMN statement to remove a constraint.

You should not use the ALTER TABLE Hovercraft.Registration DROP CONSTRAINT HovercraftID statement. This statement incorrectly references the name of the column rather than the name of the constraint.

You should not use the ALTER COLUMN PK_HovercraftID DROP CONSTRAINT HovercraftID statement. You cannot use the ALTER COLUMN statement to remove a constraint.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Delete Primary Keys
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190621.aspx

Primary and Foreign Key Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179610.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 127 (461P_1.3_07)

You are creating a view that will list all employees at your organization who make less than $50,000 per year. Users in Payroll will use this view to update salary information for these employees. All modifications to salary occur through the view. Users in Payroll should not be able to increase any employee's salary so that it exceeds $50,000.

Which clause would you use when creating the view to ensure that this restriction is enforced?

1. ENCRYPTION
2. >>CHECK OPTION <Correct>
3. VIEW_METADATA
4. SCHEMABINDING

Explanation:
Using the CHECK OPTION clause will mean that the view cannot be used to make a modification to data that would cause that data to disappear from the scope of the view.

The ENCRYPTION option means that the view definition will not be visible to unauthorized users.

The SCHEMABINDING option means that changes cannot be made to the underlying table structure that would affect the view.

The VIEW_METADATA option means that the SQL Server instance will return metadata information about the view instead of about base table or tables.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx

Question Number (ID) : 128 (461P_4.3_04)

Which of the following Transact-SQL statements do you use to populate a Transact-SQL cursor with the results from the statement used to declare the cursor?

1. DEALLOCATE

2. FETCH

3. >>OPEN <Correct>

4. CLOSE

Explanation:
You use the OPEN statement to open a Transact-SQL cursor. The cursor is populated by the results of the Transact-SQL statement used when the cursor is created.

You use the DEALLOCATE statement to remove a cursor reference.

You use the FETCH statement to retrieve a specific row from a CURSOR.

You use the CLOSE statement to release the current result set. This releases cursor locks on rows on which the cursor is positioned.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Evaluate the use of row-based operations vs. set-based operations.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 16 - Lesson 1

OPEN (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190500.aspx

Cursors (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181441.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 129 (461P_3.1_02)

You want to modify an existing stored procedure.

Which of the following statements can you use to accomplish this task?

1. ALTER ASSEMBLY
2. >>ALTER PROC <Correct>
3. CREATE PROC
4. CREATE ASSEMBLY

Explanation:
You should use the ALTER PROC statement. You use this statement modify an existing stored procedure.

You should not use the CREATE PROC statement. You use this statement to create a stored procedure. You cannot use this statement to modify an existing stored procedure.

You should not use the CREATE ASSEMBLY statement. You use this statement to create a managed application module that contains both class metadata and managed code. This module can be referenced to create common language runtime (CLR) functions, stored procedures, triggers, user-defined aggregates, and user defined times. You cannot use this statement to modify an existing stored procedure.

You should not use the ALTER ASSEMBLY statement. You use this statement to modify a managed application module. You cannot use this statement to modify an existing stored procedure.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements)

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

ALTER PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms189762.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 130 (461P_2.1_02)

You run the following query against the Hovercraft.Orders table:

```
SELECT country FROM Hovercraft.Orders;
```

This query produces the following results:

```
Country
--------------------
Tonga
Tonga
New Zealand
Tonga
New Zealand
Fiji
Fiji
Australia
```

You want to have each country listed once, and you want to have the list of countries in alphabetical order.

Which of the following Transact-SQL statements can you use to accomplish this goal? (Each correct answer presents a complete solution. Choose two.)

1. SELECT DISTINCT country FROM Hovercraft.Orders ORDER BY DESC;

2. >>SELECT country FROM Hovercraft.Orders GROUP BY country ORDER BY ASC; <Correct>

3. SELECT country FROM Hovercraft.Orders GROUP BY country ORDER BY DESC;

4. >>SELECT DISTINCT country FROM Hovercraft.Orders ORDER BY country ASC; <Correct>

Explanation:
You can use the SELECT DISTINCT country FROM Hovercraft.Orders ORDER BY country ASC; statement. This statement removes duplicates and displays the output in alphabetical order.

You can use the SELECT country FROM Hovercraft.Orders GROUP BY country ORDER BY ASC; statement. This statement removes duplicates and displays the output in alphabetical order.

You should not use the SELECT DISTINCT country FROM Hovercraft.Orders ORDER BY DESC; statement. This statement sorts the output in reverse alphabetical order.

You should not use the SELECT country FROM Hovercraft.Orders GROUP BY country ORDER BY DESC; statement. This statement sorts the output in reverse alphabetical order.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

GROUP BY (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177673.aspx

Question Number (ID) : 131 (461P_2.3_05)

Which of the following statements can you use to convert the 'mars 15 2012' string in the French language to 2012-03-15 00:00:00.0000000 as the datetime2 data type?

1. >>SELECT PARSE ('mars 15 2012' AS datetime2 using 'fr-FR') <Correct>
2. SELECT CONVERT ('mars 15 2012' AS datetime2 using 'fr-FR')
3. SELECT PARSE ('mars 15 2012' AS datetime2 using 'en-US')
4. SELECT CAST ('mars 15 2012' AS datetime2 using 'fr-FR')

Explanation:
You should use the SELECT PARSE ('mars 15 2012' AS datetime2 using 'fr-FR') statement. This statement will translate the 'mars 15 2012' string as 2012-03-15 00:00:00.0000000.

You should not use the SELECT PARSE ('mars 15 2012' AS datetime2 using 'en-US') statement. This statement will not translate the 'mars 15 2012' string as 2012-03-15 00:00:00.0000000 because the cultural setting is en-US rather than fr-FR.

You should not use the SELECT CONVERT ('mars 15 2012' AS datetime2 using 'fr-FR') statement. You should use the PARSE statement to translate from one cultural setting to another.

You should not use the SELECT CAST ('mars 15 2012' AS datetime2 using 'fr-FR') statement. You should use the PARSE statement to translate from one cultural setting to another.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 2 - Lesson 2

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

PARSE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh213316.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 132 (461P_3.1_01)

Which of the following Transact-SQL statements would you use when creating a common language runtime (CLR) stored procedure in SQL Server 2012?

1. ALTER ASSEMBLY

2. ALTER PROC

3. CREATE PROC <Correct>

4. >>CREATE ASSEMBLY

Explanation:
You should use the CREATE PROC statement, which allows you to create a common language runtime (CLR) stored procedure.

You should not use the ALTER PROC statement. You use this statement to modify an existing stored procedure. You cannot use this statement to create a CLR stored procedure.

You should not use the CREATE ASSEMBLY statement. You use this statement to create a managed application module that contains both class metadata and managed code. This module can be referenced to create common language runtime (CLR) functions, stored procedures, triggers, user-defined aggregates, and user-defined times.

You should not use the ALTER ASSEMBLY statement. You use this statement to modify a managed application module. You cannot use this statement to create a common language runtime stored procedure.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 133 (461P_4.2_05)

Which of the following statements can you use to specify the end of a transaction?

1. SAVE TRANSACTION

2. ROLLBACK WORK

3. ROLLBACK TRANSACTION

4. >>COMMIT WORK <Correct>

Explanation:
You use the COMMIT WORK statement to mark the end of a successful implicit or explicit transaction. COMMIT WORK works in the same way as the COMMIT TRANSACTION statement does.

You use the ROLLBACK WORK statement to roll back an implicit or explicit transaction. This statement works in the same way that the ROLLBACK TRANSACTION statement works except that it can only roll back to the beginning of the transaction, not to the savepoint.

You use the SAVE TRANSACTION statement to create a savepoint within a transaction.

The ROLLBACK TRANSACTION statement will roll back to an implicit or explicit transaction. This statement works in the same way as the ROLLBACK WORK statement does.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

COMMIT WORK (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178628.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

Question Number (ID) : 134 (461P_1.2_03)

You are developing a database that will be used for tracking Hovercraft rentals. An existing view returns a list of all hovercraft rented and the name of the customer who rented them. You want to modify this view to ensure that only the most recent rental information is returned by the view.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER TABLE

2. >>ALTER VIEW <Correct>

3. CREATE VIEW

4. CREATE TABLE

Explanation:
You use the ALTER VIEW statement to modify the properties of a current view, such as adding a WHERE clause to limit output.

You use the ALTER TABLE statement to modify a table. You cannot use this statement to modify a view.

You use the CREATE VIEW statement to create a view. You cannot use this statement to modify a view.

You use the CREATE TABLE statement to create a table. You cannot use this statement to modify a view.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

Modify Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178076.aspx

Question Number (ID) : 135 (461P_3.1_07)

You want to create a stored procedure associated with the Production schema that queries the Production.vProductAndDescription view to generate a list of product names, IDs, and descriptions.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  ALTER PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT Name, ProductID, Description
   FROM  Production.vProductAndDescription;
```

```
2.  ALTER PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT ProductModel, ProductID, Description
   FROM  Production.vProductAndDescription;
```

```
3.  >>CREATE PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT Name, ProductID, Description
   FROM  Production.vProductAndDescription;          <Correct>
```

```
4.  CREATE PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT ProductModel, ProductID, Description
   FROM  Production.vProductAndDescription;
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT Name, ProductID, Description
   FROM Production.vProductAndDescription;
```

This statement creates a stored procedure associated with the Production schema that returns the product name, ID, and description from the vProductAndDescription view.

You should not use the statement:

```
CREATE PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT ProductModel, ProductID, Description
   FROM Production.vProductAndDescription;
```

This statement creates a stored procedure associated with the Production schema that returns the product model, ID, and description from the vProductAndDescription view. You want to create a stored procedure associated with the Production schema that will return the product name, ID, and description from the vProductAndDescription view.

You should not use the statement:

```
ALTER PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT Name, ProductID, Description
   FROM Production.vProductAndDescription;
```

You need to use the CREATE PROCEDURE statement to create a procedure, not the ALTER PROCEDURE statement.

You should not use the statement:

```
ALTER PROCEDURE Production.uspProductInfo
AS
   SET NOCOUNT ON;
   SELECT ProductModel, ProductID, Description
   FROM Production.vProductAndDescription;
```

You need to use the CREATE PROCEDURE statement to create a procedure, not the ALTER PROCEDURE statement.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 136 (461P_2.3_08)

You are creating a table that will store Hovercraft engine test data. Test data values are between 1000 and 5000 and must be recorded with an accuracy of four decimal places. You also need to minimize the amount of space used by the column.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>CREATE TABLE TestData (READING decimal(8,4)) <Correct>

2. CREATE TABLE TestData (READING bigint)

3. CREATE TABLE TestData (READING decimal(10,4))

4. CREATE TABLE TestData (READING int)

Explanation:
You should use the CREATE TABLE TestData (READING decimal(8,4)) statement. This statement creates a column that uses the decimal type and has a precision of 8 digits and a scale of 4 digits. Using this data type will meet your needs because it enables you to store the appropriate numerical data while minimizing the space used by the column.

You should not use the CREATE TABLE TestData (READING decimal(10,4)) statement. This statement does create a column that uses the decimal type but has a precision of 10 digits, which is more than the required 8. This leads to the column using more than the minimum amount of storage space necessary to accomplish your goal.

You should not use the CREATE TABLE TestData (READING int) statement. You cannot store decimal numbers by using the integer data type.

You should not use the CREATE TABLE TestData (READING bigint) statement. You cannot store decimal numbers by using the big integer data type.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 2 - Lesson 2

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

decimal and numeric (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187746.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 137 (461P_3.3_03)

Your database has two tables related to product models, named Production.ProductModel and Hovercraft.NewDesign. You want to produce a list of all products in both tables. Your list should include duplicate entries.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT ProductModelID, Name FROM Hovercraft.NewDesign EXCEPT SELECT ProductModelID, Name FROM Production.ProductModel ORDER BY Name

2. SELECT ProductModelID, Name FROM Production.ProductModel EXCEPT SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

3. SELECT ProductModelID, Name FROM Production.ProductModel INTERSECT SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

4. SELECT ProductModelID, Name FROM Production.ProductModel UNION SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name

5. >>SELECT ProductModelID, Name FROM Production.ProductModel UNION ALL SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name <Correct>

Explanation:
You should use the SELECT ProductModelID, Name FROM Production.ProductModel UNION ALL SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of all products in both tables and will output duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel UNION SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of all products in both tables but without duplicate entries. Your list should include duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel INTERSECT SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce only those entries that are in the output of both queries. You want to produce a list of all products in both tables. Your list should include duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Production.ProductModel EXCEPT SELECT ProductModelID, Name FROM Hovercraft.NewDesign ORDER BY Name statement. This will produce a list of products from the ProductModel table as long as those products do not have duplicate entries in the NewProductModel table. You want to produce a list of all products in both tables, including duplicate entries.

You should not use the SELECT ProductModelID, Name FROM Hovercraft.NewDesign EXCEPT SELECT ProductModelID, Name FROM Production.ProductModel ORDER BY Name statement. This will produce a list of products from the NewProductModel table as long as those products do not have duplicate entries in the ProductModel table. You want to produce a list of all products in both tables, including duplicate entries.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

Set Operators (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ff848745.aspx

Question Number (ID) : 138 (461P_2.5_03)

You execute the following statement:

```
declare @tempDoc xml
set @tempDoc = '<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>';
```

Which of the following queries would you have to append to this statement to generate the following result?

```
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
```

1. >>SELECT @tempDoc.query('/Root/ProductDescription/Features'); <Correct>

2. SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance');

3. SELECT @tempDoc.query('/Root');

4. SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers');

Explanation:
The query SELECT @tempDoc.query('/Root/ProductDescription/Features') produces the result:

```
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
```

The query SELECT @tempDoc.query('/Root') produces the result:

```
<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers') produces the result:

```
<Passengers>60 passenger capacity</Passengers>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance') produces the result:

```
<Maintenance>5 year parts and labor extended maintenance </Maintenance>
```

Objective:
Work with Data

Sub Objective(s):
Query and manage XML data.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 7 - Lesson 1

Query() Method (xml Data Type)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191474.aspx

XML Data (SQL Server)
MSDN
Link: http://msdn.microsoft.com/en-us/library/bb522446.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 139 (461P_4.2_04)

You want to revert changes made during a transaction to a specific savepoint.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. SAVE TRANSACTION

2. COMMIT TRANSACTION

3. BEGIN TRANSACTION

4. >>ROLLBACK TRANSACTION <Correct>

Explanation:
You use the ROLLBACK TRANSACTION statement to roll back an implicit or explicit transaction. You can use this statement to roll back to a savepoint or to the beginning of the transaction.

You use the BEGIN TRANSACTION statement to mark the start point of an explicit local transaction.

You use the COMMIT TRANSACTION statement to mark the end of a successful implicit or explicit transaction.

You use the SAVE TRANSACTION statement to create a savepoint within a transaction.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

ROLLBACK TRANSACTION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms181299.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 140 (461P_2.1_09)

The Hovercraft.Wages table records the salaries paid to workers at Contoso. Workers get either a daily rate or a yearly salary. This table contains the following columns:

EmployeeID, daily_rate, yearly_salary

Workers get only one type of income rate, and the other columns in their records have values of NULL. For example, if a worker has an entry in the yearly_salary column, the daily_rate column will have an entry of NULL. You want to run a query calculating each employee's total salary based on the assumption that people work 5 days a week, 52 weeks per year.

Which of the following statements could you use to accomplish this goal?

1. >>SELECT EmployeeID, CAST(COALESCE(daily_rate * 5 * 52, yearly_salary) AS money) AS 'Total Salary' from Hovercraft.Wages; <Correct>

2. SELECT EmployeeID, CAST(ISNULL(daily_rate * 5 * 52, yearly_salary) AS money) AS 'Total Salary' from Hovercraft.Wages;

3. SELECT EmployeeID, CAST(daily_rate * 5 * 52 AS money) AS 'Total Salary' from Hovercraft.Wages;

4. SELECT EmployeeID, CAST(yearly_salary AS money) AS 'Total Salary' from Hovercraft.Wages;

Explanation:
You should use the SELECT EmployeeID, CAST(COALESCE(daily_rate * 5 * 52, yearly_salary) AS money) AS 'Total Salary' from Hovercraft.Wages; statement. COALESCE enables you to use the first value that is not NULL. This expression will calculate each employee's total salary independently of whether he or she is paid by a daily rate or a yearly salary.

You should not use the SELECT EmployeeID, CAST(ISNULL(daily_rate * 5 * 52, yearly_salary) AS money) AS 'Total Salary' from Hovercraft.Wages; statement. You should use COALESCE rather than ISNULL. ISNULL enables you to replace NULL values with another value. COALESCE enables you to use the first value that is not null.

You should not use the SELECT EmployeeID, CAST((daily_rate * 5 * 52) AS money) AS 'Total Salary' from Hovercraft.Wages; statement. This statement will only calculate the wages for employees who have a daily rate.

You should not use the SELECT EmployeeID, CAST(yearly_salary AS money) AS 'Total Salary' from Hovercraft.Wages; statement. This statement will only calculate the wages for employees who have a yearly salary.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

COALESCE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190349.aspx

Question Number (ID) : 141 (461P_2.2_03)

You want to write a query that returns the productID, product name, and list price of the product or products with the maximum unit price in the Production.Product table.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT COUNT(listprice) FROM Production.Product);

2. SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT AVG(listprice) FROM Production.Product);

3. >>SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MAX(listprice) FROM Production.Product); <Correct>

4. SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MIN(listprice) FROM Production.Product);

Explanation:
You should use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MAX(listprice) FROM Production.Product); statement. This statement will select any product whose list price is the maximum list price in the Production.Product table.

You should not use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MIN(listprice) FROM Production.Product); statement. This statement will select any product whose list price is the minimum list price in the Production.Product table. You want a statement that selects any product whose list price is the maximum list price in the Production.Product table.

You should not use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT AVG(listprice) FROM Production.Product); statement. This statement will select any product whose list price is the average list price of the Production.Product table. You want a statement that selects any product whose list price is the maximum list price in the Production.Product table.

You should not use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT COUNT (listprice) FROM Production.Product); statement. This statement will select any product whose list price is the same as the count of the list price in the Production.Product table. You want a statement that selects any product whose list price is the maximum list price in the Production.Product table.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 142 (461P_1.1_04)

Which of the following Transact-SQL statements can you use to move a table from one database schema to another?

1. ALTER TABLE

2. ALTER VIEW

3. ALTER DATABASE

4. >>ALTER SCHEMA <Correct>

Explanation:
You use the ALTER SCHEMA to move a table from one schema to another.

You cannot use the ALTER TABLE statement to move a table from one schema to another. You use this statement to change the properties of an existing table.

You cannot use the ALTER VIEW statement to move a table from one schema to another. You use this statement to change the properties of an existing view.

You cannot use the ALTER DATABASE statement to move a table from one schema to another. You use this statement to change the properties of an existing database.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

ALTER SCHEMA (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173423.aspx

ALTER Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/cc879314.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

Question Number (ID) : 143 (461P_2.2_07)

You want to generate a list of Hovercraft model names that have the minimum price for their chassis category from the Hovercraft.Products table.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MAX (Price) FROM Hovercraft.Products AS P2 WHERE P2.HovercraftModelName = P1.HovercraftModelName);

2. SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MIN (Price) FROM Hovercraft.Products AS P2 WHERE P2.HovercraftModelName = P1.HovercraftModelName);

3. >>SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MIN (Price) FROM Hovercraft.Products AS P2 WHERE P2.ChassisCategory = P1.ChassisCategory); <Correct>

4. SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MAX (Price) FROM Hovercraft.Products AS P2 WHERE P2.ChassisCategory = P1.ChassisCategory);

Explanation:
You should use the SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MIN(Price) FROM Hovercraft.Products AS P2 WHERE P2.ChassisCategory = P1.ChassisCategory); statement. This statement will find the minimum price per chassis category and return the Hovercraft model name and price for that category.

You should not use the SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MAX(Price) FROM Hovercraft.Products AS P2 WHERE P2.ChassisCategory = P1.ChassisCategory); statement. This statement will find the maximum price per chassis category and return the Hovercraft model name and price for that category. You want the minimum price per chassis category.

You should not use the SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MIN(Price) FROM Hovercraft.Products AS P2 WHERE P2.HovercraftModelName = P1.HovercraftModelName); statement. This query will find the minimum price per Hovercraft model name rather than minimum price per chassis category. You want the minimum price per chassis category.

You should not use the SELECT ChassisCategory, HovercraftModelName, Price FROM Hovercraft.Products AS P1 WHERE Price = (SELECT MAX(Price) FROM Hovercraft.Products AS P2 WHERE P2.HovercraftModelName = P1.HovercraftModelName); statement. This query will find the maximum price per Hovercraft model name rather than minimum price per chassis category. You want the minimum price per chassis category.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

Question Number (ID) : 144 (461P_3.1_10)

You want to create a stored procedure that returns two sets of results. The first set of results should return 10 records that list business entity ID, last name, and first name from the Person.Person table. The second set of results should return 10 records that list customer ID and account number from the Sales.Customer table.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  ALTER PROCEDURE dbo.uspMultipleResults
  AS
  SELECT TOP(10) BusinessEntityID, Lastname, FirstName FROM Person.Person;
  SELECT TOP(10) CustomerID, AccountNumber FROM Sales.Customer;

2.  ALTER PROCEDURE dbo.uspMultipleResults
  AS
  SELECT TOP(5) BusinessEntityID, Lastname, FirstName FROM Person.Person;
  SELECT TOP(5) CustomerID, AccountNumber FROM Sales.Customer;

3.  CREATE PROCEDURE dbo.uspMultipleResults
  AS
  SELECT TOP(5) BusinessEntityID, Lastname, FirstName FROM Person.Person;
  SELECT TOP(5) CustomerID, AccountNumber FROM Sales.Customer;

4.  >>CREATE PROCEDURE dbo.uspMultipleResults
  AS
  SELECT TOP(10) BusinessEntityID, Lastname, FirstName FROM Person.Person;
  SELECT TOP(10) CustomerID, AccountNumber FROM Sales.Customer;                <Correct>
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE dbo.uspMultipleResults
  AS
  SELECT TOP(10) BusinessEntityID, Lastname, FirstName FROM Person.Person;
  SELECT TOP(10) CustomerID, AccountNumber FROM Sales.Customer;
```

This statement creates a stored procedure that returns two sets of results. The first set of results will be 10 records that list business entity ID, last name, and first name from the Person.Person table. The second set of results will be 10 records that list customer ID and account number from the Sales.Customer table.

You should not use the statement:

```
CREATE PROCEDURE dbo.uspMultipleResults
  AS
  SELECT TOP(5) BusinessEntityID, Lastname, FirstName FROM Person.Person;
  SELECT TOP(5) CustomerID, AccountNumber FROM Sales.Customer;
```

This statement returns only the first five results from the Person.Person and Sales.Customer tables. You want to create a stored procedure that returns two sets of results. The first set of results should return 10 records that list business entity ID, last name, and first name from the Person.Person table. The second set of results should return 10 records that list customer ID and account number from the Sales.Customer table.

You should not use the statement:

```
ALTER PROCEDURE dbo.uspMultipleResults
  AS
  SELECT TOP(10) BusinessEntityID, Lastname, FirstName FROM Person.Person;
  SELECT TOP(10) CustomerID, AccountNumber FROM Sales.Customer;
```

You do not use the ALTER PROCEDURE statement to create stored procedures. You use the CREATE PROCEDURE statement to create stored procedures.

You should not use the statement:

```
ALTER PROCEDURE dbo.uspMultipleResults
   AS
   SELECT TOP(5) BusinessEntityID, Lastname, FirstName FROM Person.Person;
   SELECT TOP(5) CustomerID, AccountNumber FROM Sales.Customer;
```

You do not use the ALTER PROCEDURE statement to create stored procedures. You use the CREATE PROCEDURE statement to create stored procedures.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

Question Number (ID) : 145 (461P_2.4_16)

You want to generate a list of pay rates for employees who are paid every two weeks. This list should compare the employee's rate with that of the top-paid employee by listing them side by side.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE  PayFrequency=2) AND PayFrequency = 2;
```

```
2.  SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE  PayFrequency=2) AND PayFrequency = 2;
```

```
3.  SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE  PayFrequency=2;              <Correct>
```

```
4.  SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

Explanation:
You should use the statement:

```
SELECT BusinessEntityID, Rate, FIRST_VALUE(Rate) OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2;
```

This statement generates a list of pay rates for employees who are paid every two weeks, comparing each employee's pay rate with that of the top-paid employee who is paid every two weeks.

You should not use the statement:

```
SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MIN(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2;
```

This statement lists the lowest pay rate for employees paid every two weeks.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, Percent_Rank() OVER (ORDER BY Rate DESC)
FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement ranks employee pay in terms of its percentage rank within the organization.

You should not use the statement:

```
SELECT BusinessEntityID, Rate FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE rate = (SELECT MAX(Rate) FROM AdventureWorks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency=2) AND PayFrequency = 2;
```

This statement lists only the top rate for employees who are paid every two weeks.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

FIRST_VALUE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh213018.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 146 (461P_3.3_10)

You want to update the contents of the Hovercraft.CurrentDesign table with the contents of the Hovercraft.NewDesign table. If a design from the Hovercraft.NewDesign table already exists within the Hovercraft.CurrentDesign table, its DesignName and DesignSpec columns should be updated with the contents present in the Hovercraft.NewDesign table.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. MERGE INTO Hovercraft.NewDesign AS nd USING Hovercraft.CurrentDesign AS cd ON nd.DesignID = cd.DesignID WHEN MATCHED THEN DELETE;

2. MERGE INTO Hovercraft.CurrentDesign AS cd USING Hovercraft.NewDesign AS nd ON cd.DesignID = nd.DesignID WHEN MATCHED THEN DELETE;

3. >>MERGE INTO Hovercraft.CurrentDesign AS cd USING Hovercraft.NewDesign AS nd ON cd.DesignID = nd.DesignID WHEN MATCHED THEN UPDATE SET cd.DesignName = nd.DesignName, cd.DesignSpec = nd.DesignSpec; <Correct>

4. MERGE INTO Hovercraft.NewDesign AS nd USING Hovercraft.CurrentDesign AS cd ON nd.DesignID = cd.DesignID WHEN MATCHED THEN UPDATE SET nd.DesignName = cd.DesignName, nd.DesignSpec = cd.DesignSpec;

Explanation:
You should use the MERGE INTO Hovercraft.CurrentDesign AS cd USING Hovercraft.NewDesign AS nd ON cd.DesignID = nd.DesignID WHEN MATCHED THEN UPDATE SET cd.DesignName = nd.DesignName, cd.DesignSpec = nd.DesignSpec; statement. This statement will update the contents of the Hovercraft.CurrentDesign table with the contents of the Hovercraft.NewDesign table. If a design from the Hovercraft.NewDesign table already exists within the Hovercraft.CurrentDesign table, its DesignName and DesignSpec columns will be updated with the contents in the Hovercraft.NewDesign table.

You should not use the MERGE INTO Hovercraft.NewDesign AS nd USING Hovercraft.CurrentDesign AS cd ON nd.DesignID = cd.DesignID WHEN MATCHED THEN UPDATE SET nd.DesignName = cd.DesignName, nd.DesignSpec = cd.DesignSpec; statement. This statement updates the contents of the Hovercraft.NewDesign table with the contents of the Hovercraft.CurrentDesign table. You want to update the contents of the Hovercraft.CurrentDesign table with the contents of the Hovercraft.NewDesign table.

You should not use the MERGE INTO Hovercraft.NewDesign AS nd USING Hovercraft.CurrentDesign AS cd ON nd.DesignID = cd.DesignID WHEN MATCHED THEN DELETE; statement. This statement updates the contents of the Hovercraft.NewDesign table with the contents of the Hovercraft.CurrentDesign table. You want to update the contents of the Hovercraft.CurrentDesign table with the contents of the Hovercraft.NewDesign table.

You should not use the MERGE INTO Hovercraft.CurrentDesign AS cd USING Hovercraft.NewDesign AS nd ON cd.DesignID = nd.DesignID WHEN MATCHED THEN DELETE; statement. Although this statement does merge the content of the Hovercraft.NewDesign table into the Hovercraft.CurrentDesign table, it deletes rather than replaces content when there is a match.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

MERGE (Transact-SQL)
MSDN
Link: http://technet.microsoft.com/en-us/library/bb510625.aspx

DELETE (Transact-SQL)
MSDN

Link: http://technet.microsoft.com/en-us/library/ms189835.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 147 (461P_1.2_02)

Which of the following Transact-SQL statements do you use to create an indexed view?

1. >>ALTER VIEW
2. EXEC SP_RENAME
3. DROP VIEW
4. CREATE VIEW <Correct>

Explanation:
You use the CREATE VIEW statement to create an indexed view.

You use the ALTER VIEW statement to modify an existing view. You cannot use this statement to create a new view.

You use the DROP VIEW statement to remove an existing view. You cannot use this statement to create a new view.

You use the EXEC SP_RENAME stored procedure to rename database objects, including views. You cannot use this statement to create a new view.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190174.aspx

Create Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175503.aspx

Question Number (ID) : 148 (461P_2.3_06)

You have decided to use GUIDs as a data type for a primary key. The key will be on the HovercraftID column of the Hovercraft table. This table is stored on a database stored on a single disk. You want to optimize performance by ensuring that new rows are always inserted at the end of the table.

Which of the following statements would you use to accomplish this goal?

1. CREATE TABLE Hovercraft (HovercraftID rowversion PRIMARY KEY)

2. >>CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY) <Correct>

3. CREATE TABLE Hovercraft (HovercraftID timestamp PRIMARY KEY)

4. CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWID() PRIMARY KEY)

Explanation:
You should use the CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY) statement. This statement will create a GUID that can be used for a primary key that increases in value. This means that new insert operations will occur at the end of the table.

You should not use the CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWID() PRIMARY KEY) statement. This statement will create a GUID that can be used for a primary key, but the ID value will be random, meaning that new insert operations will not always occur at the end of the table.

You should not use the CREATE TABLE Hovercraft (HovercraftID rowversion PRIMARY KEY) statement. This statement will not create a GUID. When creating a GUID, you use the uniqueidentifier data type.

You should not use the CREATE TABLE Hovercraft (HovercraftID timestamp PRIMARY KEY) statement. This statement will not create a GUID. When creating a GUID, you use the uniqueidentifier data type.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 2 - Lesson 2

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

NEWSEQUENTIALID() (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189786.aspx

Question Number (ID) : 149 (461P_2.3_09)

You are creating a table named Fluctuation for the Hovercraft database. You are creating a column named Reading to store integer values between -30,000 and 30,000. You want to minimize the space used by the column.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. >>CREATE TABLE Fluctuation (Reading smallint) <Correct>
2. CREATE TABLE Fluctuation (Reading tinyint)
3. CREATE TABLE Fluctuation (Reading bigint)
4. CREATE TABLE Fluctuation (Reading int)

Explanation:
You should use the CREATE TABLE Fluctuation (Reading smallint) statement. The smallint data type consumes 2 bytes and can record integer values between -32,768 and 32,767. This is appropriate for your goal.

You should not use the CREATE TABLE Fluctuation (Reading int) statement. Although the int type can record the values required, the data type consumes 4 bytes. You can use less space by using the smallint data type.

You should not use the CREATE TABLE Fluctuation (Reading tinyint) statement. The tinyint data type can only record values between 0 and 255 and is inappropriate for your needs.

You should not use the CREATE TABLE Fluctuation (Reading bigint) statement. The bigint data type can record the values required, but the data type consumes 8 bytes. You can use less space by using the smallint data type.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 2 - Lesson 2

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

int, bigint, smallint, and tinyint (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187745.aspx

Question Number (ID) : 150 (461P_2.3_01)

Columns of the Production.WorkOrder table in the AdventureWorks2012 database are configured with the following data types:

WorkOrderID: int, not null
OrderQty: int, not null
DueDate: datetime, not null

A query against the table generates the output shown in the exhibit under Before. You need to modify this query so that the output appears as shown in the exhibit under After.

Which of the following queries should you issue to accomplish this goal?

1. SELECT WorkOrderID, OrderQty, CONVERT(varchar(8), DueDate,112) as DueDate from Production.WorkOrder <Correct>

2. >>SELECT WorkOrderID, OrderQty, CONVERT(numeric, DueDate,112) as DueDate from Production.WorkOrder

3. SELECT WorkOrderID, OrderQty, CONVERT(bigint, DueDate,112) as DueDate from Production.WorkOrder

4. SELECT WorkOrderID, OrderQty, CONVERT(int, DueDate,112) as DueDate from Production.WorkOrder

Explanation:
You should use the SELECT WorkOrderID, OrderQty, CONVERT(varchar(8), DueDate,112) as DueDate from Production.WorkOrder statement. This statement correctly converts the DateTime column into the column displayed in the exhibit (20050715).

You should not use the SELECT WorkOrderID, OrderQty, CONVERT(int, DueDate,112) as DueDate from Production.WorkOrder statement. Using this statement converts the date string into an integer, which will result in displaying the DateTime column as 38546 rather than as 20050715.

You should not use the SELECT WorkOrderID, OrderQty, CONVERT(bigint, DueDate,112) as DueDate from Production.WorkOrder statement. Using this statement converts the date string into a large integer, which will result in displaying the DateTime column as 38546 rather than as 20050715.

You should not use the SELECT WorkOrderID, OrderQty, CONVERT(numeric, DueDate,112) as DueDate from Production.WorkOrder statement. Using this statement converts the date string into a numeric type, which will result in displaying the DateTime column as 38546 rather than as 20050715.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:
Modify or Rename DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190671.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 151 (461P_2.1_03)

The Hovercraft.Orders table records information on customers' Hovercraft orders. Customers make multiple orders. You want to view the maximum order cost for each customer. You run the following query against the Hovercraft.Orders table:

SELECT customer, ordertotal FROM Hovercraft.Orders group by customer;

When you execute this query, you get an error.

Which of the following queries could you use to accomplish your goal without receiving an error?

1. >>SELECT customer, MAX(ordertotal) as maxordertotal FROM Hovercraft.Orders GROUP BY customer;
<Correct>

2. SELECT customer, ordertotal FROM Hovercraft.Orders group by ordertotal;

3. SELECT customer, MIN(ordertotal) as maxordertotal FROM Hovercraft.Orders GROUP BY customer;

4. SELECT DISTINCT customer, ordertotal FROM Hovercraft.Orders;

Explanation:
You should use the SELECT customer, MAX(ordertotal) as maxordertotal FROM Hovercraft.Orders GROUP BY customer; statement. This statement displays the maximum order cost for each customer.

You should not use the SELECT DISTINCT customer, ordertotal FROM Hovercraft.Orders; statement. This statement will generate an error due to the multiple order totals for each customer.

You should not use the SELECT customer, ordertotal FROM Hovercraft.Orders group by ordertotal; statement. This statement will generate an error due to the multiple order totals for each customer.

You should not use the SELECT customer, MIN(ordertotal) as maxordertotal FROM Hovercraft.Orders GROUP BY customer; statement. This statement displays the minimum order cost for each customer rather than the maximum.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 152 (461P_3.4_19)

You want to use the appropriate function to convert a binary value to the datetime2 data type. The conversion should not return a null value if this is not possible.

Which of the following built-in functions can you use to accomplish this goal?

1. PARSE

2. TRY_CONVERT

3. TRY_PARSE

4. >>CONVERT <Correct>

Explanation:
CONVERT enables you to convert data stored in one data type, such as binary, to another, such as datetime2. The conversion might fail if you attempt to convert between data types where conversion is not possible.

The PARSE function enables you to change a string written in one cultural format (such as en-AU) to data types such as datetime and datetime2. If you do not use PARSE, strings that are not in the appropriate format will not convert.

TRY_CONVERT enables you to attempt to convert data stored in one data type to another. If this conversion fails, a NULL value is written.

TRY_PARSE enables you to change a string written in one cultural format (such as en-AU) to other data types such as datetime and datetime2. When you use TRY_PARSE, if the operation fails, the conversion occurs with the value being set to NULL.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

CAST and CONVERT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187928.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 153 (461P_2.5_04)

You execute the following statement:

```
declare @tempDoc xml
set @tempDoc = '<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>';
```

Which of the following queries would you have to append to this statement to generate the following result?

```
<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>
```

1. SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers');

2. SELECT @tempDoc.query('/Root/ProductDescription/Features');

3. >>SELECT @tempDoc.query('/Root'); <Correct>

4. SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance');

Explanation:
The query SELECT @tempDoc.query('/Root') produces the result:

```
<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features') produces the result:

```
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers') produces the result:

```
<Passengers>60 passenger capacity</Passengers>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance') produces the result:

```
<Maintenance>5 year parts and labor extended maintenance </Maintenance>
```

Objective:
Work with Data

Sub Objective(s):
Query and manage XML data.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 7 - Lesson 1

Query() Method (xml Data Type)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191474.aspx

XML Data (SQL Server)
MSDN
Link: http://msdn.microsoft.com/en-us/library/bb522446.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 154 (461P_1.1_06)

You want to alter the properties of the Model column of the Hovercraft table in the Rentals schema so that it is of data type nvarchar 25 rather than nvarchar 50.

Which of the following statements can you use to accomplish this goal?

1. ALTER TABLE Models nvarchar(25) ALTER COLUMN [Rentals].[Hovercraft]

2. ALTER COLUMN Models nvarchar(25)

3. ALTER COLUMN [Rentals].[Hovercraft] Models nvarchar(25)

4. >>ALTER TABLE [Rentals].[Hovercraft] ALTER COLUMN Models nvarchar(25) <Correct>

Explanation:
You should use the ALTER TABLE [Rentals].[Hovercraft] ALTER COLUMN Models nvarchar(25) statement.

You should not use the ALTER COLUMN Models nvarchar(25) statement. The ALTER COLUMN statement must be used as part of the ALTER TABLE statement.

You should not use the ALTER COLUMN [Rentals].[Hovercraft] Models nvarchar(25) statement. The ALTER COLUMN statement must be used as part of the ALTER TABLE statement.

You should not use the ALTER TABLE Models nvarchar(25) ALTER COLUMN [Rentals].[Hovercraft] statement. This statement incorrectly has the name of the column placed where the name of the table should be. It also has the name of the table placed where the name of the column should be.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

Modify Columns (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190259.aspx

ALTER Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/cc879314.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

Question Number (ID) : 155 (461P_2.4_06)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued sick leave hours. You want to determine the minimum sick leave hours accrued by any single employee who has a job title like Marketing Manager.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1. SELECT MAX(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';

2. >>SELECT MIN(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';          <Correct>

3. SELECT AVG(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';

4. SELECT VAR(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

Explanation:
You should use the statement:

```
SELECT MIN(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the minimum value for sick leave hours for employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT MAX(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the maximum value for sick leave hours for employees who have the job title Marketing Manager. You want to find the minimum value for sick leave hours for employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT AVG(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the average value for SickLeave hours for employees with the job title Marketing Manager. You want to find the minimum value for sick leave hours for employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT VAR(SickLeaveHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement finds the statistical variance in SickLeave hours for employees with the job title Marketing Manager. You want to find the minimum value for sick leave hours for employees who have the job title Marketing Manager.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

MIN (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179916.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 156 (461P_4.4_05)

An error occurs when you execute a block of Transact-SQL code, causing a transaction to roll back. You want to determine the severity of this error. You configure the appropriate TRY . . . CATCH construct, using the code so that you can further diagnose the error.

Which of the following system functions can you use to determine this information?

1. ERROR_PROCEDURE()
2. ERROR_LINE()
3. ERROR_STATE()
4. >>ERROR_SEVERITY() <Correct>
5. ERROR_NUMBER()
6. ERROR_MESSAGE()

Explanation:
You can use the ERROR_SEVERITY() system function to determine the severity of the error.

You can use the ERROR_LINE() system function to determine the line number at which the error occurred.

You can use the ERROR_MESSAGE() system function to determine the text of the message that is returned to the application.

You can use the ERROR_NUMBER() system function to determine the error number.

You can use the ERROR_PROCEDURE() system function to determine the name of the trigger of stored procedure in which the error occurred.

You can use the ERROR_STATE() system function to determine the error state number. An error state number differs from an error number. For example, the 1105 error can have a state of 0, 1, 2 or 3.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Implement error handling.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 2

ERROR_SEVERITY (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178567.aspx

Retrieving Error Information in Transact-SQL
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179495(SQL.105).aspx

Question Number (ID) : 157 (461P_4.4_02)

An error occurs when you execute a block of Transact-SQL code. You configure the appropriate TRY . . . CATCH construct, using the code so that you can further diagnose the error. You want to determine the exact message text returned by this error.

Which of the following system functions can you use to determine this information?

1. ERROR_STATE()
2. ERROR_LINE()
3. ERROR_SEVERITY()
4. >>ERROR_MESSAGE() <Correct>
5. ERROR_PROCEDURE()
6. ERROR_NUMBER()

Explanation:
You can use the ERROR_MESSAGE() system function to determine the text of the message that is returned to the application.

You can use the ERROR_LINE() system function to determine the line number at which the error occurred.

You can use the ERROR_NUMBER() system function to determine the error number.

You can use the ERROR_PROCEDURE() system function to determine the name of the trigger of stored procedure in which the error occurred.

You can use the ERROR_SEVERITY() system function to determine the severity of the error.

You can use the ERROR_STATE() system function to determine the error state number. An error state number differs from an error number. For example, the 1105 error can have a state of 0, 1, 2, or 3.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Implement error handling.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 2

ERROR_MESSAGE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190358.aspx

Retrieving Error Information in Transact-SQL
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179495(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 158 (461P_1.5_02)

You are creating a trigger that will report the number of rows in the Inserted table during its operation.

Which of the following statements would you include when defining this trigger to accomplish this goal?

1. SELECT * FROM Deleted
2. >>SELECT COUNT(*) FROM Inserted <Correct>
3. SELECT COUNT(*) FROM Deleted
4. SELECT * FROM Inserted

Explanation:
You can use the SELECT COUNT(*) FROM Inserted statement in a trigger to determine the number of rows that have been inserted.

You should not use the SELECT COUNT(*) FROM Deleted statement. This statement counts the number of rows in the Deleted table. You are interested in the number of rows in the Inserted table.

You should not use the SELECT * FROM Inserted statement. This statement selects all content in the Inserted table, but it does not count the rows. You are interested in the number of rows in the Inserted table.

You should not use the SELECT * FROM Deleted statement. This statement selects all content from the Deleted table, but it does not count the rows in the inserted table.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Use the inserted and deleted Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191300.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 159 (461P_2.3_03)

The SpatialLocation column in the Person.Address table of the AdventureWorks2012 database uses the geography data type. You want to create a query that outputs the AddressID column with the contents of the SpatialLocation column as coordinates in longitude and latitude format.

Which of the following Transact-SQL statements can you use to accomplish this goal? (Each correct answer presents a complete solution. Choose two.)

1. >>SELECT AddressID, CAST(SpatialLocation as FLOAT) as "LON/LAT" from Person.Address

2. SELECT AddressID, CAST(SpatialLocation as BIGINT) as "LON/LAT" from Person.Address

3. SELECT AddressID, CAST(SpatialLocation as VARCHAR(45)) as "LON/LAT" from Person.Address <Correct>

4. >>SELECT AddressID, CONVERT(VARCHAR(45), SpatialLocation) as "LON/LAT" from Person.Address
<Correct>

Explanation:
You can use the SELECT AddressID, CONVERT(VARCHAR(45), SpatialLocation) as "LON/LAT" from Person.Address statement. This statement will convert the data stored in the SpatialLocation column using the Geography data type to coordinates in longitude and latitude format.

You can use the SELECT AddressID, CAST(SpatialLocation as VARCHAR(45)) as "LON/LAT" from Person.Address statement. This statement will convert the data stored in the SpatialLocation column using the Geography data type to coordinates in longitude and latitude format.

You cannot use the SELECT AddressID, CAST(SpatialLocation as FLOAT) as "LON/LAT" from Person.Address statement. You cannot convert the Geography data type to the FLOAT data type.

You cannot use the SELECT AddressID, CAST(SpatialLocation as BIGINT) as "LON/LAT" from Person.Address statement. You cannot convert the Geography data type to the BIGINT data type.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:
Get Information about DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179309.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Data Type Conversion (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191530.aspx

geography (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/cc280766.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 160 (461P_3.2_06)

Six months ago, you created the Hovercraft.Chassis table by executing the following statement:

```
CREATE TABLE Hovercraft.Orders
(
orderid INT NOT NULL IDENTITY(1, 1)
CONSTRAINT PK_Orders_orderid PRIMARY KEY,
custid INT NOT NULL,
empid INT NOT NULL,
orderdate DATE NOT NULL
CONSTRAINT DFT_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)), shipcountry NVARCHAR(15) NOT
NULL,
freight MONEY NOT NULL
);
```

You now want to make a copy of some of the data in that table related to Australian orders in a new table, named Hovercraft.OrdersAust.

Which of the following statements can you use to accomplish this goal?

1. SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.Orders FROM Hovercraft.OrdersAust

2. SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.OrdersAust FROM Hovercraft.Orders WHERE shipcountry = N'Australia' <Correct>

3. SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.OrdersAust FROM Hovercraft.Orders

4. >>SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.Orders FROM Hovercraft.OrdersAust WHERE shipcountry = N'Australia'

Explanation:
You should use the SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.OrdersAust FROM Hovercraft.Orders WHERE shipcountry = N'Australia' statement.

You should not use the SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.Orders FROM Hovercraft.OrdersAust WHERE shipcountry = N'Australia' statement. This statement incorrectly specifies the source and destination tables.

You should not use the SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.OrdersAust FROM Hovercraft.Orders statement. This statement incorrectly does not limit the select statement so that only data where shipcountry = Australia is selected.

You should not use the SELECT orderid, custid, orderdate, shipcountry, freight INTO Hovercraft.Orders FROM Hovercraft.OrdersAust statement. This statement incorrectly specifies the source and destination tables. This statement also incorrectly does not limit the select statement so that only data where shipcountry = Australia is selected.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

Inserting Rows by Using SELECT INTO

MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190750(SQL.105).aspx

INTO Clause (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188029.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 161 (461P_1.4_02)

You want to add a primary key constraint to the existing Person.Exemplar table. You want to configure this primary key constraint to use the Alpha column. The Alpha column is configured to disallow nulls.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. CREATE TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha UNIQUE (Alpha)

2. >>ALTER TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha PRIMARY KEY (Alpha) <Correct>

3. ALTER TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha UNIQUE (Alpha)

4. CREATE TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha PRIMARY KEY (Alpha)

Explanation:
You should use the ALTER TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha PRIMARY KEY (Alpha) statement. This statement adds a primary key constraint named PK_Alpha on the Alpha column of the Person.Exemplar table.

You should not use the ALTER TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha UNIQUE (Alpha) statement. This statement adds a unique constraint named PK_Alpha on the Alpha column of the Person.Exemplar table rather than a primary key constraint.

You should not use the CREATE TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha PRIMARY KEY (Alpha) statement. You are adding a constraint to an existing table, so you should use the ALTER TABLE statement.

You should not use the CREATE TABLE Person.Exemplar ADD CONSTRAINT PK_Alpha UNIQUE (Alpha) statement. You are adding a constraint to an existing table, so you should use the ALTER TABLE statement.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Primary and Foreign Key Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms179610.aspx

Create Primary Keys
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189039.aspx

Question Number (ID) : 162 (461P_2.4_03)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued vacation hours. You want to determine the maximum vacation hours accrued by any single employee who has a job title like Design Engineer.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  SELECT VAR(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
2.  SELECT MIN(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
3.  >>SELECT MAX(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';              <Correct>
```

```
4.  SELECT AVG(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

Explanation:
You should use the statement:

```
SELECT MAX(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the maximum value for vacation hours for employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT MIN(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the minimum value for vacation hours for employees who have the job title Design Engineer. You want to find the maximum value for vacation hours for employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT AVG(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the average value for vacation hours for employees with the job title Design Engineer. You want to find the maximum value for vacation hours for employees who have the job title Design Engineer.

You should not use the statement:

```
SELECT VAR(VacationHours)
FROM HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement finds the statistical variance in vacation hours for employees with the job title Design Engineer. You want to find the maximum value for vacation hours for employees who have the job title Design Engineer.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

MAX (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187751.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 163 (461P_2.5_05)

You execute the following statement:

```
declare @tempDoc xml
set @tempDoc = '<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>';
```

Which of the following queries would you have to append to this statement to generate the following result?

```
<Passengers>60 passenger capacity</Passengers>
```

1. SELECT @tempDoc.query('/Root');

2. SELECT @tempDoc.query('/Root/ProductDescription/Features');

3. SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance');

4. >>SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers'); <Correct>

Explanation:
The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Passengers') produces the result:

```
<Passengers>60 passenger capacity</Passengers>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features') produces the result:

```
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
```

The query SELECT @tempDoc.query('/Root') produces the result:

```
<Root>
<ProductDescription ProductID="1" ProductName="Hovercraft">
<Features>
  <Passengers>60 passenger capacity</Passengers>
  <Performance>Top speed: 200 KMPH</Performance>
  <Maintenance>5 year parts and labor extended maintenance </Maintenance>
</Features>
</ProductDescription>
</Root>
```

The query SELECT @tempDoc.query('/Root/ProductDescription/Features/Maintenance') produces the result:

```
<Maintenance>5 year parts and labor extended maintenance </Maintenance>
```

Objective:
Work with Data

Sub Objective(s):
Query and manage XML data.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 7 - Lesson 1

Query() Method (xml Data Type)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms191474.aspx

XML Data (SQL Server)
MSDN
Link: http://msdn.microsoft.com/en-us/library/bb522446.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 164 (461P_3.4_17)

You want to use the appropriate function to change a string that has the date written in Australian format so that it can be written to a column that uses the datetime2 data type. The conversion should not return a null value if this is not possible.

Which of the following built-in functions can you use to accomplish this goal?

1. TRY_CONVERT

2. TRY_PARSE

3. >>CONVERT

4. PARSE <Correct>

Explanation:
The PARSE function enables you to change a string written in one cultural format (such as en-AU) to data types such as datetime and datetime2. If you do not use PARSE, strings that are not in the appropriate format will not convert.

TRY_CONVERT enables you to attempt to convert data stored in one data type to another. If this conversion fails, a NULL value is written.

TRY_PARSE enables you to change a string written in one cultural format (such as en-AU) to other data types such as datetime and datetime2. When you use TRY_PARSE, if the operation fails, the conversion occurs with the value being set to NULL.

CONVERT enables you to convert data stored in one data type to another. The conversion might fail if you attempt to convert between data types where conversion is not possible.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

PARSE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/hh213316.aspx

Question Number (ID) : 165 (461P_3.3_08)

The Hovercraft database has three tables related to Hovercraft design, OldDesign, CurrentDesign, and FutureDesign. Some designs are present in more than one table. You want to generate a list of all items in the FutureDesign table as well as FutureDesign items that are present in both the OldDesign and CurrentDesign tables.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

2. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

3. >>SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign <Correct>

4. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

Explanation:
You should use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the FutureDesign table as well as FutureDesign items that are present in both the OldDesign and CurrentDesign tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in all three tables. You want to generate a list of all items in the FutureDesign table as well as FutureDesign items that are present in both the OldDesign and CurrentDesign tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in the OldDesign and CurrentDesign tables but not present in the FutureDesign table. You want to generate a list of all items in the FutureDesign table as well as FutureDesign items that are present in both the OldDesign and CurrentDesign tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the OldDesign table as well as OldDesign items that are present in both the CurrentDesign and FutureDesign tables. You want to generate a list of all items in the FutureDesign table as well as FutureDesign items that are present in both the OldDesign and CurrentDesign tables.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

EXCEPT and INTERSECT (Transact-SQL)

MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188055.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 166 (461P_1.2_08)

You are reviewing the design of a database hosted on SQL Server 2012. You want to move the Exemplar view from the HumanResources schema to the Person schema.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER VIEW [Person] TRANSFER [HumanResources].[Exemplar]

2. ALTER SCHEMA [HumanResources] TRANSFER [Person].[Exemplar]

3. >>ALTER SCHEMA [Person] TRANSFER [HumanResources].[Exemplar] <Correct>

4. ALTER VIEW [HumanResources] TRANSFER [Person].[Exemplar]

Explanation:
You should use the ALTER SCHEMA [Person] TRANSFER [HumanResources].[Exemplar] statement. This statement transfers the Exemplar view from the HumanResources schema to the Person schema.

You should not use the ALTER SCHEMA [HumanResources] TRANSFER [Person].[Exemplar] statement. This statement would transfer the Exemplar view from the Person schema to the HumanResources schema.

You should not use the ALTER VIEW [Person] TRANSFER [HumanResources].[Exemplar] statement. You cannot use the ALTER VIEW statement to move a view to a different schema.

You should not use the ALTER VIEW [HumanResources] TRANSFER [Person].[Exemplar] statement. You cannot use the ALTER VIEW statement to move a view to a different schema.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

ALTER SCHEMA (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173423.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 167 (461P_1.4_08)

You want to ensure that values inserted in the score column of the results table are between 2 and 9, inclusive.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER TABLE dbo.Results ADD CONSTRAINT PK_Results PRIMARY KEY (Results)

2. ALTER TABLE dbo.Results ADD CONSTRAINT CHK_Results CHECK (Score < 1 and Score > 10)

3. ALTER TABLE dbo.Results ADD CONSTRAINT UN_Results UNIQUE (Results)

4. >>ALTER TABLE dbo.Results ADD CONSTRAINT CHK_Results CHECK (Score > 1 and Score < 10) <Correct>

Explanation:
You should use the ALTER TABLE dbo.Results ADD CONSTRAINT CHK_Results CHECK (Score > 1 and Score < 10) statement. This statement creates a check constraint that ensures that values inserted in the score column are between 2 and 9, inclusive.

You should not use the ALTER TABLE dbo.Results ADD CONSTRAINT CHK_Results CHECK (Score < 1 and Score > 10) statement. This statement creates a check constraint, but it allows values that are less than 1 and greater than 10.

You should not use the ALTER TABLE dbo.Results ADD CONSTRAINT UN_Results UNIQUE (Results) statement. This statement creates a unique constraint. It will not ensure that data inserted into this column is between 2 and 9, inclusive.

You should not use the ALTER TABLE dbo.Results ADD CONSTRAINT PK_Results PRIMARY KEY (Results) statement. This statement creates a primary key constraint. It will not ensure that data inserted into this column is between 2 and 9, inclusive.

Objective:
Create Database Objects

Sub Objective(s):
Create and modify constraints (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 2

Unique Constraints and Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187550.aspx

Create Check Constraints
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190377.aspx

Question Number (ID) : 168 (461P_3.1_20)

You asked your subordinate to create a stored procedure that generates a list of employees that includes their first and last names and is sorted by when they started working at the organization. Although your subordinate did create a stored procedure that outputs this information, called HumanResources.uspEmployeeDates, the information was not sorted. You want to modify this stored procedure so that the output information is sorted by start date.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY StartDate;                   <Correct>
```

```
2.  >>CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY StartDate;
```

```
3.  ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY LastName;
```

```
4.  CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY LastName;
```

Explanation:
You should use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY StartDate;
```

This statement modifies the HumanResources.uspEmployeeDates stored procedure so that it returns employee last names, first names, and start dates sorted by start date.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY LastName;
```

This statement modifies the HumanResources.uspEmployeeDates stored procedure so that it returns employee last names, first names, and start date sorted by last name. This information should be sorted by start date.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY StartDate;
```

The CREATE PROCEDURE statement creates a new stored procedure. You want to alter an existing stored procedure. To do this, you need to use the ALTER PROCEDURE statement.

```
CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY LastName;
```

The CREATE PROCEDURE statement creates a new stored procedure. You want to alter an existing stored procedure. To do this, you need to use the ALTER PROCEDURE statement.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

ALTER PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms189762.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 169 (461P_1.2_05)

You are creating several views to be used with a confidential database. This database will be replicated to another site, but you want to ensure that several of the views that you are creating are not replicated.

Which of the following options would you include in the CREATE VIEW statement to ensure that this goal is accomplished?

1. CHECK OPTION

2. VIEW_METADATA

3. >>ENCRYPTION <Correct>

4. SCHEMABINDING

Explanation:
You would use ENCRYPTION. ENCRYPTION encrypts the text of the CREATE VIEW statement. A side effect of using ENCRYPTION is that the view cannot be published as part of SQL Server replication.

You would not use SCHEMABINDING. This option ensures that the base table or tables cannot be modified in any way that would affect the view definition.

You would not use CHECK OPTION. This option forces all statements that modify data executed against the view to follow criteria set within the statement that defines the view.

You would not use VIEW_METADATA. When you use this option, browse mode metadata will return the view name and not the underlying table names when it lists columns from the view in the result set.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187956.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 170 (461P_1.5_05)

You want to change the definition of a DML trigger.

Which of the following Transact-SQL statements would you use to accomplish this goal?

1. CREATE TRIGGER
2. >>ALTER TRIGGER <Correct>
3. DROP TRIGGER
4. ALTER SCHEMA

Explanation:
You can use the ALTER TRIGGER statement to modify the definition of a DML trigger.

You can use the CREATE TRIGGER statement to create a new DML trigger. You cannot use this statement to modify the definition of an existing DML trigger directly. It is possible to modify a DML trigger by dropping the trigger and then re-creating it, but in this case, only one answer is possible.

You can use the DROP TRIGGER statement to delete an existing DML trigger. You cannot use this statement to modify the definition of an existing DML trigger directly. It is possible to modify a DML trigger by dropping the trigger and then re-creating it, but in this case, only one answer is possible.

You use the ALTER SCHEMA statement to transfer objects to different schemas. You cannot use this statement to modify the definition of an existing DML trigger.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

ALTER TRIGGER (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176072.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Question Number (ID) : 171 (461P_3.3_07)

The Hovercraft database has three tables related to Hovercraft design, OldDesign, CurrentDesign, and FutureDesign. Some designs are present in more than one table. You want to generate a list of all items in the OldDesign table as well as OldDesign items that are present in both the CurrentDesign and FutureDesign tables.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

2. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

3. SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign

4. >>SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign
<Correct>

Explanation:
You should use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the OldDesign table as well as OldDesign items that are present in both the CurrentDesign and FutureDesign tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in all three tables. You want to generate a list of all items in the OldDesign table as well as OldDesign items that are present in both the CurrentDesign and FutureDesign tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign EXCEPT SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will find items that are present in the OldDesign and CurrentDesign tables but not present in the FutureDesign table. You want to generate a list of all items in the OldDesign table as well as OldDesign items that are present in both the CurrentDesign and FutureDesign tables.

You should not use the SELECT HovercraftModelID, Name FROM Hovercraft.OldDesign INTERSECT SELECT HovercraftModelID, Name FROM Hovercraft.CurrentDesign UNION SELECT HovercraftModelID, Name FROM Hovercraft.FutureDesign statement. This statement will list all items in the FutureDesign table as well as FutureDesign items that are present in both the OldDesign and CurrentDesign tables. You want to generate a list of all items in the OldDesign table as well as OldDesign items that are present in both the CurrentDesign and FutureDesign tables.

Objective:
Modify Data

Sub Objective(s):
Combine datasets.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 11 - Lesson 1

UNION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms180026.aspx

EXCEPT and INTERSECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms188055.aspx

Question Number (ID) : 172 (461P_2.2_01)

You need to create a subquery that retrieves one instance of each product in the AdventureWorks2012 database when the product name includes Mountain Bike Sock and the ProductModelID numbers match in the Production.Product and Production.ProductModel tables.

Which of the following queries can you use to accomplish this goal? (Each correct answer presents a complete solution. Choose two.)

```
1.  >>SELECT DISTINCT Name
FROM Production.Product
WHERE ProductModelID IN
   (SELECT ProductModelID
    FROM Production.ProductModel
    WHERE Name LIKE 'Mountain Bike Sock%');
         <Correct>
```

```
2.  >>SELECT DISTINCT Name
FROM Production.Product AS pp
WHERE EXISTS
   (SELECT *
    FROM Production.ProductModel AS pm
    WHERE pp.ProductModelID = pm.ProductModelID
         AND pm.Name LIKE 'Mountain Bike Sock%');
         <Correct>
```

```
3.  SELECT Name
FROM Production.Product AS pp
WHERE EXISTS
   (SELECT *
    FROM Production.ProductModel AS pm
    WHERE pp.ProductModelID = pm.ProductModelID
         AND pm.Name LIKE 'Mountain Bike Sock%');
```

```
4.  SELECT Name
FROM Production.Product
WHERE ProductModelID IN
   (SELECT ProductModelID
    FROM Production.ProductModel
    WHERE Name LIKE 'Mountain Bike Sock%');
```

Explanation:
You can use the following statement:

```
SELECT DISTINCT Name
FROM Production.Product AS pp
WHERE EXISTS
   (SELECT *
    FROM Production.ProductModel AS pm
    WHERE pp.ProductModelID = pm.ProductModelID
         AND pm.Name LIKE 'Mountain Bike Sock%');
```

This statement will retrieve one instance of each product in the AdventureWorks2012 database when the product name includes Mountain Bike Sock and the ProductModelID numbers match in the Production.Product and Production.ProductModel tables.

You can use the following statement:

```
SELECT DISTINCT Name
```

```
FROM Production.Product
WHERE ProductModelID IN
   (SELECT ProductModelID
    FROM Production.ProductModel
    WHERE Name LIKE 'Mountain Bike Sock%');
```

This statement will retrieve one instance of each product in the AdventureWorks2012 database when the product name includes Mountain Bike Sock and the ProductModelID numbers match in the Production.Product and Production.ProductModel tables.

You should not use the following statement:

```
SELECT Name
FROM Production.Product AS pp
WHERE EXISTS
   (SELECT *
    FROM Production.ProductModel AS pm
    WHERE pp.ProductModelID = pm.ProductModelID
         AND pm.Name LIKE 'Mountain Bike Sock%');
```

This statement does not use a DISTINCT clause, which means that it might retrieve more than one instance of each product in the AdventureWorks2012 database when the product name includes Mountain Bike Sock and the ProductModelID numbers match in the Production.Product and Production.ProductModel tables.

You should not use the following statement:

```
SELECT Name
FROM Production.Product
WHERE ProductModelID IN
   (SELECT ProductModelID
    FROM Production.ProductModel
    WHERE Name LIKE 'Mountain Bike Sock%');
```

This statement does not use a DISTINCT clause, which means that it might retrieve more than one instance of each product in the AdventureWorks2012 database when the product name includes Mountain Bike Sock and the ProductModelID numbers match in the Production.Product and Production.ProductModel tables.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 173 (461P_1.1_08)

You want to remove the Odometer column of the Hovercraft table. This table is located in the Rentals schema of the AdventureWorks2012 database.

Which of the following statements can you use to accomplish this goal?

1. >>ALTER TABLE [Rentals].[Hovercraft] DROP COLUMN Odometer <Correct>

2. ALTER TABLE [Hovercraft].[Rentals] DROP COLUMN Odometer

3. DROP COLUMN [Hovercraft].[Rentals] Odometer

4. DROP COLUMN [Rentals].[Hovercraft] Odometer

Explanation:
You should use the ALTER TABLE [Rentals].[Hovercraft] DROP COLUMN Odometer statement. This statement removes the Odometer column from the Hovercraft table associated with the Rentals schema.

You should not use the ALTER TABLE [Hovercraft].[Rentals] DROP COLUMN Odometer statement. This statement removes the Odometer column, but from the Rentals table associated with the Hovercraft schema.

You should not use the DROP COLUMN [Hovercraft].[Rentals] Odometer statement. You can only use the DROP COLUMN statement from within the ALTER TABLE statement.

You should not use the DROP COLUMN [Rentals].[Hovercraft] Odometer statement. You can only use the DROP COLUMN statement from within the ALTER TABLE statement.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

Modify Columns (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190259.aspx

ALTER Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/cc879314.aspx

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 174 (461P_3.2_10)

Which of the following statements can you use to delete all orders from the Hovercraft.Orders table where the productID is 343 except those orders where the shipping country is set to Australia?

1. DELETE FROM Hovercraft.Orders WHERE productID = 343 OR NOT shippingcountry = N'Australia'

2. >>DELETE FROM Hovercraft.Orders WHERE productID = 343 AND NOT shippingcountry = N'Australia'
<Correct>

3. DELETE FROM Hovercraft.Orders WHERE productID = 343 OR shippingcountry = N'Australia'

4. DELETE FROM Hovercraft.Orders WHERE productID = 343 AND shippingcountry = N'Australia'

Explanation:
You should use the DELETE FROM Hovercraft.Orders WHERE productID = 343 AND NOT shippingcountry = N'Australia' statement. This statement will delete all data where the product ID is set to 343 except when the shipping country is Australia.

You should not use the DELETE FROM Hovercraft.Orders WHERE productID = 343 AND shippingcountry = N'Australia' statement. This statement will delete data where the product ID is set to 343 and the shipping country is set to Australia.

You should not use the DELETE FROM Hovercraft.Orders WHERE productID = 343 OR shippingcountry = N'Australia' statement. This will delete data where the product ID is 343, independent of what the shipping country is. This statement will also delete data when the shipping country = Australia, independent of the product id.

You should not use the DELETE FROM Hovercraft.Orders WHERE productID = 343 OR NOT shippingcountry = N'Australia' statement. This statement will delete all data where the product ID is 343. It will also delete all data where the shipping country is not Australia.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

DELETE (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189835.aspx

NOT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189455.aspx

Question Number (ID) : 175 (461P_4.2_06)

You want to revert changes made during a transaction to the start of that transaction.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. ROLLBACK WORK <Correct>

2. COMMIT WORK

3. >>SAVE TRANSACTION

4. BEGIN TRANSACTION

Explanation:
You use the ROLLBACK WORK statement to roll back an implicit or explicit transaction. This statement works in the same way that the ROLLBACK TRANSACTION statement works except that it can only roll back to the beginning of the transaction, not to the savepoint.

You use the BEGIN TRANSACTION statement to mark the start point of an explicit local transaction.

You use the SAVE TRANSACTION statement to create a savepoint within a transaction.

You use the COMMIT WORK statement to mark the end of a successful implicit or explicit transaction. COMMIT WORK works in the same way as the COMMIT TRANSACTION statement does.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

ROLLBACK WORK (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174973.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

Question Number (ID) : 176 (461P_4.2_07)

Which of the following statements will list the number of BEGIN TRANSACTION statements that have occurred through the current connection without a COMMIT TRANSACTION, COMMIT WORK, ROLLBACK TRANSACTION, or ROLLBACK WORK statement being issued?

1. >>PRINT @@TRANCOUNT <Correct>
2. PRINT @@ROWCOUNT
3. PRINT @@IDENTITY
4. PRINT @@ERROR

Explanation:
You use the @@TRANCOUNT system function to learn the number of BEGIN TRANSACTION statements that have occurred through the current connection without a ROLLBACK or COMMIT statement being issued. Each time you issue a BEGIN TRANSACTION statement, @@TRANCOUNT is incremented by one. Every time you issue a ROLLBACK or COMMIT statement, @@TRANCOUNT decrements by one unless you are rolling back to a savepoint.

You use the @@IDENTITY system function to learn the last-inserted identity value.

You use the @@ROWCOUNT system function to learn the number of rows affected by the most recent statement.

You use the @@ERROR system function to learn the error number for the last Transact-SQL statement executed.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

@@TRANCOUNT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187967.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

Question Number (ID) : 177 (461P_2.1_04)

The AdventureWorks2012 database has two tables that you want to query, Production.Product and Sales.SalesOrderDetail. The Production.Product table has the following columns, named for their data types:

Name
ProductID

The Sales.SalesOrderDetail table has the following columns:

OrderQty
UnitPrice
UnitPriceDiscount
ProductID

You want to view the total sales and the discounts for each product, sorted by product name.

Which of the following Transact-SQL statements can you use to accomplish this goal?

```
1.  SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY UnitPrice DESC;
```

```
2.  SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY OrderQty DESC;
```

```
3.  SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY UnitPriceDiscount DESC;
```

```
4.  >>SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName DESC;
        <Correct>
```

Explanation:
You should use the following statement:

```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY ProductName DESC;
```

This statement will output the total sales and the discounts for each product, sorted by product name.

You should not use the following statement:

```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY UnitPrice DESC;
```

This statement will output the total sales and the discounts for each product, sorted by unit price.

You should not use the following statement:

```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY OrderQty DESC;
```

This statement will output the total sales and the discounts for each product, sorted by OrderQty.

You should not use the following statement:

```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS s
ON p.ProductID = s.ProductID
ORDER BY UnitPriceDiscount DESC;
```

This statement will output the total sales and the discounts for each product, sorted by unit price discount.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

SELECT Examples (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187731.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 178 (461P_3.1_06)

You want to create a stored procedure associated with the HumanResources schema that queries the HumanResources.vJobCandidate view to generate a list of candidates that includes their first names, last names, email addresses, and skills.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT JobCandidateID, [Name.Last], Email, Skills
   FROM  HumanResources.vJobCandidate;

2.  ALTER PROCEDURE HumanResources.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT JobCandidateID, [Name.Last], Email, Skills
   FROM  HumanResources.vJobCandidate;

3.  ALTER PROCEDURE HumanResources.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT [Name.First], [Name.Last], Email, Skills
   FROM  HumanResources.vJobCandidate;

4.  >>CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT [Name.First], [Name.Last], Email, Skills
   FROM  HumanResources.vJobCandidate;              <Correct>
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT [Name.First], [Name.Last], Email, Skills
   FROM HumanResources.vJobCandidate;
```

This statement creates a stored procedure associated with the HumanResources schema that returns a job candidate's first name, last name, email address, and skills.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT JobCandidateID, [Name.Last], Email, Skills
   FROM HumanResources.vJobCandidate;
```

This statement creates a stored procedure associated with the HumanResources schema that returns a job candidate's ID, last name, email address, and skills. You want to create a stored procedure associated with the HumanResources schema that returns a job candidate's first name, last name, email address, and skills.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.uspCandidateSkills
AS
```

```
   SET NOCOUNT ON;
   SELECT [Name.First], [Name.Last], Email, Skills
   FROM HumanResources.vJobCandidate;
```

You need to use the CREATE PROCEDURE statement rather than the ALTER PROCEDURE statement to create a stored procedure associated with the HumanResources schema that returns a job candidate's first name, last name, email address, and skills.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.uspCandidateSkills
AS
   SET NOCOUNT ON;
   SELECT JobCandidateID, [Name.Last], Email, Skills
   FROM HumanResources.vJobCandidate;
```

You need to use the CREATE PROCEDURE statement rather than the ALTER PROCEDURE statement to create a stored procedure associated with the HumanResources schema that will return a job candidate's first name, last name, email address, and skills.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

Question Number (ID) : 179 (461P_3.4_10)

Which of the following queries that use string functions will return the result 'lia'?

```
1.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  REVERSE(@testvalue);
```

```
2.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEFT(@testvalue,  3);
```

```
3.  >>DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RIGHT(@testvalue,  3);      <Correct>
```

```
4.  DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT  UPPER(@testvalue);
```

Explanation:
The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT RIGHT(@testvalue, 3);
```

will return the result 'lia' because it will return the three characters from the string from the right.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT LEFT(@testvalue, 3);
```

will return the result 'aus' because it will return the first three characters from the left of the string 'australia'.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT UPPER(@testvalue);
```

will return the result 'AUSTRALIA' because it will convert all characters from the string 'australia' to uppercase.

The statement:

```
DECLARE @testvalue nvarchar(20);
SET @testvalue = 'australia';
SELECT REVERSE(@testvalue);
```

will return the result 'ailartsua' because it will reverse all the characters from the string 'australia'.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

RIGHT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177532.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 180 (461P_3.2_03)

You create the Hovercraft.OrdersAPAC table by using the following statement:

```
CREATE TABLE Hovercraft.OrdersAPAC
(
orderid INT NOT NULL IDENTITY(1, 1)
CONSTRAINT PK_Orders_orderid PRIMARY KEY,
custid INT NOT NULL,
empid INT NOT NULL,
orderdate DATE NOT NULL
CONSTRAINT DFX_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)), shipcountry NVARCHAR(15) NOT
NULL,
freight MONEY NOT NULL
);
```

The Hovercraft.Orders statement already exists and was created using the following statement:

```
CREATE TABLE Hovercraft.Orders
(
orderid INT NOT NULL IDENTITY(1, 1)
CONSTRAINT PK_Orders_orderid PRIMARY KEY,
custid INT NOT NULL,
empid INT NOT NULL,
orderdate DATE NOT NULL
CONSTRAINT DFZ_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)), shipcountry NVARCHAR(15) NOT
NULL,
freight MONEY NOT NULL
);
```

You create the Hovercraft.OrdersForCountry procedure by using the following statement:

```
CREATE PROC Hovercraft.OrdersForCountry
@country AS NVARCHAR(15)
AS
SELECT orderid, custid, empid, orderdate, shipcountry, freight
FROM Hovercraft.Orders
Where shipcountry = @country;
```

You want to populate the Hovercraft.OrdersAPAC table with all the order data from the Hovercraft.Orders table related to countries in the Asia/Pacific region such as Australia and New Zealand.

Which of the following statements can you use to accomplish this goal of inserting order data where New Zealand is the ship country?

1. INSERT INTO Hovercraft.OrdersAPAC(custid, orderid, empid, orderdate, shipcountry, freight) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand'

2. INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, freight, shipcountry) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand'

3. INSERT INTO Hovercraft.OrdersAPAC(empid, custid, orderid, orderdate, shipcountry, freight) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand'

4. >>INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, shipcountry, freight) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand' <Correct>

Explanation:
You should use the INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, shipcountry, freight) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand' statement. This statement correctly orders each column in the source and destination tables. The order in which the columns appear in the procedure needs to match the order used in the INSERT statement.

You should not use the INSERT INTO Hovercraft.OrdersAPAC(custid, orderid, empid, orderdate, shipcountry, freight) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand' statement. This statement incorrectly orders each column in the source and destination tables. The order in which the columns appear in the procedure needs to match the order used in the INSERT statement. The order defined in the procedure is orderid, custid, empid, orderdate, shipcountry, freight.

You should not use the INSERT INTO Hovercraft.OrdersAPAC(empid, custid, orderid, orderdate, shipcountry, freight) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand' statement. This statement incorrectly orders each column in the source and destination tables. The order in which the columns appear in the procedure needs to match the order used in the INSERT statement. The order defined in the procedure is orderid, custid, empid, orderdate, shipcountry, freight.

You should not use the INSERT INTO Hovercraft.OrdersAPAC(orderid, custid, empid, orderdate, freight, shipcountry) EXEC Hovercraft.OrdersForCountry @country = N'New Zealand' statement. This statement incorrectly orders each column in the source and destination tables. The order in which the columns appear in the procedure needs to match the order used in the INSERT statement. The order defined in the procedure is orderid, custid, empid, orderdate, shipcountry, freight.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

INSERT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174335.aspx

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

Question Number (ID) : 181 (461P_2.4_12)

You want to generate a ranked list of pay rates for employees who are paid every two weeks by querying the HumanResources.EmployeePayHistory table of the AdventureWorks2012 database. If more than one employee has the same rate, they should all be assigned the same rank. The employee who has the rate that is next below this rate should be ranked from the beginning as shown in the exhibit between rows 7 and 8.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;            <Correct>
```

```
2.  SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
3.  SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

```
4.  SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

Explanation:
You should use the statement:

```
SELECT BusinessEntityID, Rate, RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

This statement outputs results using the RANK function. When you use the RANK function and multiple results share the same rank, those results will be assigned that rank. The next result will be assigned the rank based on how many rows there are after the first row. Hence, ranking results can be of the sequence 1,2,2,2,5 rather than 1,2,2,2,3.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, DENSE_RANK() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use DENSE_RANK, the statement ranks output, and the same results share the same rank; however, the results do not appear in the same way as they do with the RANK statement. The next result when you use DENSE_RANK sequentially follows the last RANK. For example, 1,2,2,2,3 rather than 1,2,2,2,5.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, NTILE(5) OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

The NTILE() function partitions the results into equal sections based on the number you use with the NTILE function. This query would partition the results into five ranks rather than the ranking order shown in the exhibit.

You should not use the statement:

```
SELECT BusinessEntityID, Rate, ROW_NUMBER() OVER (ORDER BY Rate DESC) as Rank
FROM Adventureworks2012.HumanResources.EmployeePayHistory
WHERE PayFrequency = 2;
```

When you use the ROW_NUMBER function, ranks are assigned sequentially, even when the result is the same. Instead of 1,2,2,2,5, in which the second, third, and fourth results are the same, the result would be 1,2,3,4,5.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

RANK (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms176102.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 182 (461P_2.3_07)

You have decided to use GUIDs as a data type for a primary key. The key will be on the HovercraftID column of the Hovercraft table. This table is stored on a database stored across multiple disks. You have determined that insert performance will be improved if you use randomly generated keys.

Which of the following statements would you use to accomplish this goal?

1. CREATE TABLE Hovercraft (HovercraftID timestamp PRIMARY KEY)

2. CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY)

3. CREATE TABLE Hovercraft (HovercraftID rowversion PRIMARY KEY)

4. >>CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWID() PRIMARY KEY) <Correct>

Explanation:
You should use the CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWID() PRIMARY KEY) statement. This statement will create random GUIDs that can be used for a primary key. This type of key can be suitable for large databases spread across large disk arrays.

You should not use the CREATE TABLE Hovercraft (HovercraftID uniqueidentifier DEFAULT NEWSEQUENTIALID() PRIMARY KEY) statement. This statement will create sequential GUIDs. The question states that a random GUID must be generated for performance reasons.

You should not use the CREATE TABLE Hovercraft (HovercraftID rowversion PRIMARY KEY) statement. This statement will not create a random GUID. When creating a GUID, you use the uniqueidentifier data type.

You should not use the CREATE TABLE Hovercraft (HovercraftID timestamp PRIMARY KEY) statement. This statement will not create a random GUID. When creating a GUID, you use the uniqueidentifier data type.

Objective:
Work with Data

Sub Objective(s):
Implement data types.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 2 - Lesson 2

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

NEWID (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190348.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 183 (461P_3.4_05)

Which of the following mathematical functions would you use to return the smallest integer greater than or equal to a specific numeric value?

1. FLOOR

2. ABS

3. >>CEILING <Correct>

4. POWER

Explanation:
You use the CEILING function to return the smallest integer greater than or equal to a specific numeric value.

You use the FLOOR function to return the smallest integer less than or equal to a specific numeric value.

You use the POWER function to raise a numeric value to a specific exponent.

You use the ABS function to determine the absolute value of a specific numeric value.

Objective:
Modify Data

Sub Objective(s):
Work with functions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 3 - Lesson 2

Built-in Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174318.aspx

CEILING (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189818.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 184 (461P_1.1_09)

You want to remove a table in a database.

Which of the following statements can you use to accomplish this goal?

1. CREATE TABLE

2. SELECT . . . INTO

3. ALTER TABLE

4. >>DROP TABLE <Correct>

Explanation:
You use the DROP TABLE to remove a table from a database.

You should not use the CREATE TABLE statement. You use the CREATE TABLE statement to add a table to a database. You cannot use this statement to remove a table from a database.

You should not use the ALTER TABLE statement. You use the ALTER TABLE statement to change the properties of an existing table. You cannot use this statement to remove a table from a database.

You should not use the SELECT . . . INTO statement. You use the SELECT . . . INTO statement to create a new table based on the results of a SELECT query. You cannot use this statement to remove a table from a database.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

Delete Tables (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190616.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 185 (461P_1.3_03)

Your subordinate created a view named dbo.vSalesAustralia that was supposed to return information in the Sales table only where the Country column is set to Australia. Instead, the view returns information in the Sales table only where the Country column is set to Tonga.

Which of the following statements would you use to rectify this problem?

1. CREATE VIEW

2. >>ALTER VIEW <Correct>

3. DROP VIEW

4. ALTER PROCEDURE

Explanation:
You would use the ALTER VIEW statement to modify the properties of an existing view.

You would use the CREATE VIEW statement to create a new view. You cannot use this statement to modify the properties of an existing view.

You use the DROP VIEW statement to delete an existing view. You cannot use this statement to modify the properties of an existing view.

You use the ALTER PROCEDURE statement to modify an existing stored procedure. You cannot use this statement to modify the properties of an existing view.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

ALTER VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms173846.aspx

Question Number (ID) : 186 (461P_1.1_10)

You are reviewing a database created by a contractor. You have noticed that the contractor has misnamed the Hovercraft table in the Vehicles schema as Bovercraft. You need to change the Bovercraft table name to Hovercraft.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ALTER TABLE [Vehicles].[Bovercraft] [Vehicles].[Hovercraft]

2. >>EXEC sp_rename 'Vehicles.Hovercraft', 'Bovercraft'

3. ALTER TABLE [Vehicles].[Hovercraft] [Vehicles].[Bovercraft]

4. EXEC sp_rename 'Vehicles.Bovercraft', 'Hovercraft' <Correct>

Explanation:
You should use the EXEC sp_rename 'Vehicles.Bovercraft', 'Hovercraft' statement. This statement renames the Bovercraft table associated with the Vehicles schema to Hovercraft.

You should not use the EXEC sp_rename 'Vehicles.Hovercraft', 'Bovercraft' statement. This statement renames the table Hovercraft associated with the Vehicles schema to Bovercraft.

You should not use the ALTER TABLE [Vehicles].[Bovercraft] [Vehicles].[Hovercraft] statement. You cannot use the ALTER TABLE statement to rename a table.

You should not use the ALTER TABLE [Vehicles].[Hovercraft] [Vehicles].[Bovercraft] statement. You cannot use the ALTER TABLE statement to rename a table.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter tables using T-SQL syntax (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 8 - Lesson 1

Tables
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189084.aspx

Rename Tables (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/aa337520.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 187 (461P_3.2_04)

You create the table Hovercraft.Orders by executing the following statement:

```
CREATE TABLE Hovercraft.Orders
(
orderid INT NOT NULL IDENTITY(1, 1)
CONSTRAINT PK_Orders_orderid PRIMARY KEY,
custid INT NOT NULL,
empid INT NOT NULL,
orderdate DATE NOT NULL
CONSTRAINT DFT_MyOrders_orderdate DEFAULT (CAST(SYSDATETIME() AS DATE)), shipcountry NVARCHAR(15) NOT
NULL,
freight MONEY NOT NULL
);
```

Which of the following statements can you use to insert data into this table? (Each correct answer presents a complete solution. Choose two.)

1. >>INSERT INTO Hovercraft.Orders (custid, empid, shipcountry, freight) VALUES (4, 11, N'Australia', 10000.00) <Correct>

2. >>INSERT INTO Hovercraft.Orders (shipcountry, custid, empid, freight) VALUES (N'New Zealand', 5, 12, 5000.00) <Correct>

3. INSERT INTO Hovercraft.Orders (custid, empid, freight, shipcountry) VALUES (7, 14, N'Fiji', 5000.00)

4. INSERT INTO Hovercraft.Orders (empid, shipcountry, custid, freight) VALUES (N'Tonga', 6, 13, 5000.00)

Explanation:
You can use the INSERT INTO Hovercraft.Orders (custid, empid, shipcountry, freight) VALUES (4, 11, N'Australia', 10000.00) statement. This statement correctly matches input data with the data types defined in the table.

You can use the INSERT INTO Hovercraft.Orders (shipcountry, custid, empid, freight) VALUES (N'New Zealand', 5, 12, 5000.00) statement. This statement correctly matches input data with the data types defined in the table.

You cannot use the INSERT INTO Hovercraft.Orders (custid, empid, freight, shipcountry) VALUES (7, 14, N'Fiji', 5000.00) statement. This statement incorrectly attempts to insert the Fiji string into the freight column. The table creation statement lists this column as of the money data type. This will result in an error.

You cannot use the INSERT INTO Hovercraft.Orders (empid, shipcountry, custid, freight) VALUES (N'Tonga', 6, 13, 5000.00) statement. This statement incorrectly attempts to insert the Tonga string into the empid column. The table creation statement defines this column as using the int data type. This will result in an error.

Objective:
Modify Data

Sub Objective(s):
Modify data by using INSERT, UPDATE, and DELETE statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 10 - Lesson 1

INSERT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174335.aspx

Data Types (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187752.aspx

Question Number (ID) : 188 (461P_1.5_01)

You have created a DML trigger that fires when an UPDATE operation is performed. You want to ensure that this DML trigger does not fire in the event that no rows are affected by the UPDATE operation.

Which of the following statements must you include in the first line of the trigger to accomplish this goal?

1. IF @@ROWCOUNT = 1 RETURN
2. >>IF @@ROWCOUNT = 0 RETURN <Correct>
3. IF @@ROWCOUNT = 2 RETURN
4. IF @@ROWCOUNT = 3 RETURN

Explanation:
You need to include the IF @@ROWCOUNT = 0 RETURN statement in the first line if you want to prevent the trigger from firing when no rows are affected by the UPDATE operation.

You should not use the IF @@ROWCOUNT = 1 RETURN statement. This means that the trigger will not fire if one row is affected, rather than if no rows are affected.

You should not use the IF @@ROWCOUNT = 2 RETURN statement. This means that the trigger will not fire if two rows, rather than no rows, are affected.

You should not use the IF @@ROWCOUNT = 3 RETURN statement. This means that the trigger will not fire if three rows, rather than no rows, are affected.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:
@@ROWCOUNT (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187316.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 189 (461P_3.1_11)

You want to create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and end date as output.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE HumanResources.uspEmployeEndDate
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   WHERE FirstName = @FirstName AND LastName = @LastName;

2.  CREATE PROCEDURE HumanResources.uspEmployeEndDate
   @LastName nvarchar(50),
   @MiddleName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, StartDate
   FROM HumanResources.vEmployeeDepartmentHistory
   WHERE MiddleName = @MiddleName AND LastName = @LastName;

3.  >>CREATE PROCEDURE HumanResources.uspEmployeEndDate
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory
   WHERE FirstName = @FirstName AND LastName = @LastName;    <Correct>

4.  CREATE PROCEDURE HumanResources.uspEmployeEndDate
   @LastName nvarchar(50),
   @MiddleName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory
   WHERE MiddleName = @MiddleName AND LastName = @LastName;
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeEndDate
   @LastName nvarchar(50),
   @FirstName nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT FirstName, LastName, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory
   WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement creates a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and end date as output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeEndDate
  @LastName nvarchar(50),
  @MiddleName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT MiddleName, LastName, EndDate
  FROM HumanResources.vEmployeeDepartmentHistory
  WHERE MiddleName = @MiddleName AND LastName = @LastName;
```

This statement creates a stored procedure that takes the last name and middle name as input parameters rather than first name and last name. You want to create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and end date as output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeEndDate
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, StartDate
  FROM HumanResources.vEmployeeDepartmentHistory
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

This stored procedure returns an employee's first name, last name, and start date. You want to create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and end date as output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeEndDate
  @LastName nvarchar(50),
  @MiddleName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT MiddleName, LastName, StartDate
  FROM HumanResources.vEmployeeDepartmentHistory
  WHERE MiddleName = @MiddleName AND LastName = @LastName;
```

This stored procedure takes an employee's middle name and last name as input and returns the start date. You want to create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and end date as output.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 190 (461P_1.5_06)

You want to disable the existing trigger named Hovercraft.service_reminder that applies to the Hovercraft.Logbook table. You might need to re-enable this trigger in the future.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. ENABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook
2. >>DISABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook <Correct>
3. DROP TRIGGER Hovercraft.service_reminder
4. ALTER TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook

Explanation:
You use the DISABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook statement to disable the Hovercraft.service_reminder trigger that applies to the Hovercraft.Logbook table.

You should not use the ENABLE TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook statement. This statement enables the trigger rather than disabling it.

You should not use the DROP TRIGGER Hovercraft.service_reminder statement. This statement deletes the trigger rather than disabling it.

You should not use the ALTER TRIGGER Hovercraft.service_reminder ON Hovercraft.Logbook statement. This statement modifies the trigger, but it cannot be used to disable the trigger.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter DML triggers.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 2

Delete or Disable DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175506.aspx

DML Triggers
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms178110.aspx

Question Number (ID) : 191 (461P_2.4_08)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued sick leave hours. You want to determine the total number of sick leave hours belonging to employees who have a job title like Marketing Manager.

Which of the following Transact-SQL statements can you use to accomplish this task?

1. SELECT AVG(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';

2. SELECT MAX(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';

3. >>SELECT SUM(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%'; <Correct>

4. SELECT COUNT(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';

Explanation:
You should use the statement:

```
SELECT SUM(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This statement determines the total sum of sick leave hours of all employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT COUNT(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This query finds the number of employees with the title Marketing Manager who have values set for sick leave hours other than NULL. You want to determine the total sum of sick leave hours of all employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT AVG(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This query finds the average number of sick leave hours for employees who have the job title Marketing Manager. You want to determine the total sum of sick leave hours of all employees who have the job title Marketing Manager.

You should not use the statement:

```
SELECT MAX(SickLeaveHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Marketing Manager%';
```

This query finds the maximum number of sick leave hours that any employee who has the job title Marketing Manager has. You want to determine the total sum of sick leave hours of all employees who have the job title Marketing Manager.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

SUM (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187810.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 192 (461P_2.1_01)

You run the following query against the Hovercraft.Orders table:

SELECT country FROM Hovercraft.Orders;

This query produces the following results:

```
Country
--------------------
Tonga
Tonga
New Zealand
Tonga
New Zealand
Fiji
Fiji
Australia
```

You want a list of countries but with no duplicates.

Which of the following queries should you issue to accomplish this goal? (Each correct answer presents a complete solution. Choose two.)

1. SELECT country FROM Hovercraft.Orders WHERE NOT country= N'Tonga';

2. SELECT country FROM Hovercraft.Orders WHERE country= N'Tonga';

3. >>SELECT country FROM Hovercraft.Orders GROUP BY country; <Correct>

4. >>SELECT DISTINCT country FROM Hovercraft.Orders; <Correct>

Explanation:
You can use the SELECT DISTINCT country FROM Hovercraft.Orders; statement. This query will output each country separately without duplicates.

You can use the SELECT country FROM Hovercraft.Orders GROUP BY country; statement. This query will output each country separately without duplicates.

You cannot use the SELECT country FROM Hovercraft.Orders WHERE NOT country= N'Tonga'; statement. This statement will not remove duplicates; it will output all countries except Tonga.

You cannot use the SELECT country FROM Hovercraft.Orders WHERE country= N'Tonga'; statement. This statement will not remove duplicates; it will output all records where the country is listed as Tonga.

Objective:
Work with Data

Sub Objective(s):
Query data by using SELECT statements.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 1 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

GROUP BY (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms177673.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 193 (461P_3.1_13)

You want to create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and email address as output.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @LastName AND LastName = @FirstName;
```

```
2.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

```
3.  CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @LastName AND LastName = @FirstName;
```

```
4.  >>CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;    <Correct>
```

Explanation:
You should use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

This statement creates a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and email address as output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @FirstName AND LastName = @LastName;
```

This stored procedure outputs first name, last name, and phone number. You want create a stored procedure that accepts employee first name and last name as input parameters and returns the first name, last name, and email address as output.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, PhoneNumber
  FROM HumanResources.vEmployee
  WHERE FirstName = @LastName AND LastName = @FirstName;
```

This statement incorrectly maps the LastName input to FirstName and the FirstName input to LastName. It also returns the employee's phone number rather than email address.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeEmail
  @LastName nvarchar(50),
  @FirstName nvarchar(50)
AS
  SET NOCOUNT ON;
  SELECT FirstName, LastName, EmailAddress
  FROM HumanResources.vEmployee
  WHERE FirstName = @LastName AND LastName = @FirstName;
```

This statement incorrectly maps the LastName input to FirstName and the FirstName input to LastName.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

CREATE PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187926.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

Question Number (ID) : 194 (461P_2.4_09)

The HumanResources.Employee table of the AdventureWorks2012 database includes information on each employee's accrued vacation hours. When an employee takes a vacation and uses all his or her hours, the hours are reset to NULL. You want to determine the total number of employees who have a job title like Design Engineer who have outstanding vacation hours.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1. SELECT SUM(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
2. >>SELECT COUNT(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';                <Correct>
```

```
3. SELECT MAX(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

```
4. SELECT AVG(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

Explanation:
You should use the statement:

```
SELECT COUNT(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This query finds the number of employees with the title Design Engineer who have values set for vacation hours other than NULL.

You should not use the statement:

```
SELECT SUM(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This statement determines the total sum of vacation hours of all employees who have the job title Design Engineer. You want to find the number of employees with the title Design Engineer who have values set for vacation hours other than NULL.

You should not use the statement:

```
SELECT AVG(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This query finds the average number of vacation hours for employees who have the job title Design Engineer. You want to find the number of employees with the title Design Engineer who have values set for vacation hours other than NULL.

You should not use the statement:

```
SELECT MAX(VacationHours)
FROM AdventureWorks2012.HumanResources.Employee
WHERE JobTitle LIKE 'Design Engineer%';
```

This query finds the maximum number of vacation hours that any employee who has the job title Design Engineer has. You want to find the number of employees with the title Design Engineer who have values set for vacation hours other than NULL.

Objective:
Work with Data

Sub Objective(s):
Implement aggregate queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 5 - Lesson 3

COUNT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms175997.aspx

Aggregate Functions (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173454.aspx

Question Number (ID) : 195 (461P_1.3_04)

One of the developers in your organization created a number of views without using the SCHEMABINDING clause. Several modifications to the objects underlying the view have meant that queries run against these views are generating unexpected results.

Which stored procedure should you run to resolve this issue?

1. sp_rename
2. >>sp_refreshview <Correct>
3. sp_tableoption
4. sp_recompile

Explanation:
The sp_refreshview system stored procedure updates metadata for a non-schema-bound view. You should run this system stored procedure when there are changes to the underlying objects on which the view depends.

The sp_recompile system stored procedure enables you to force stored procedures and triggers to recompile the next time they run. You would not use sp_recompile to update metadata for a non-schema-bound view.

The sp_rename system stored procedure enables you to rename user-created objects. You would not use sp_rename to update metadata for a non-schema-bound view.

The sp_tableoption system stored procedure sets option values for user-defined tables. You would not use sp_tableoption to update metadata for a non-schema-bound view.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

sp_refreshview (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms187821(SQL.105).aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 196 (461P_4.2_02)

You are developing some Transact-SQL code. You want to ensure that you correctly mark the end of successful transactions.

Which of the following statements would you use to accomplish this goal?

1. SAVE TRANSACTION
2. >>COMMIT TRANSACTION <Correct>
3. BEGIN TRANSACTION
4. ROLLBACK TRANSACTION

Explanation:
You use the COMMIT TRANSACTION statement to mark the end of a successful implicit or explicit transaction.

You use the BEGIN TRANSACTION statement to mark the start point of an explicit local transaction.

You use the SAVE TRANSACTION statement to create a savepoint within a transaction.

You use the ROLLBACK TRANSACTION statement to roll back an implicit or explicit transaction. You can use this statement to roll back to a savepoint or to the beginning of the transaction.

Objective:
Troubleshoot and Optimize

Sub Objective(s):
Manage transactions.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 12 - Lesson 1

COMMIT TRANSACTION (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190295.aspx

Transaction Statements (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms174377.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 197 (461P_1.2_09)

The Alpha and Beta views are associated with the Australia schema. You want to move the Beta view to the Europe schema and delete the Alpha view.

Which of the following statements would you use to accomplish this goal? (Each correct answer presents part of a complete solution. Choose two.)

1. >>ALTER SCHEMA [Europe] TRANSFER [Australia].[Beta] <Correct>
2. DROP VIEW [Europe].[Beta]
3. ALTER VIEW [Australia] TRANSFER [Europe].[Beta]
4. >>DROP VIEW [Australia].[Alpha] <Correct>

Explanation:
You need to use the ALTER SCHEMA [Europe] TRANSFER [Australia].[Beta] statement. This statement moves the Beta view from the Australia schema to the Europe schema.

You need to use the DROP VIEW [Australia].[Alpha] statement. This statement removes the Alpha view from the Australia schema.

You do not need to use the DROP VIEW [Europe].[Beta] statement. This statement drops the Beta view located in the Europe schema.

You do not need to use the ALTER VIEW [Australia] TRANSFER [Europe].[Beta] statement. This statement moves the Beta view from the Europe schema to the Australia schema.

Objective:
Create Database Objects

Sub Objective(s):
Create and alter views (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

ALTER SCHEMA (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms173423.aspx

Question Number (ID) : 198 (461P_2.2_04)

You want to write a query that returns the productID, product name, and list price of the product or products with the minimum unit price in the Production.Product table.

Which of the following Transact-SQL statements can you use to accomplish this goal?

1. SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MAX(listprice) FROM Production.Product);

2. SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT COUNT(listprice) FROM Production.Product);

3. >>SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MIN(listprice) FROM Production.Product); <Correct>

4. SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT AVG(listprice) FROM Production.Product);

Explanation:
You should use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MIN(listprice) FROM Production.Product); statement. This statement will select any product whose list price is the minimum list price in the Production.Product table.

You should not use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT MAX(listprice) FROM Production.Product); statement. This statement will select any product whose list price is the maximum list price in the Production.Product table. You want a statement that selects any product whose list price is the minimum list price in the Production.Product table.

You should not use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT AVG(listprice) FROM Production.Product); statement. This statement will select any product whose list price is the average list price of the Production.Product table. You want a statement that selects any product whose list price is the minimum list price in the Production.Product table.

You should not use the SELECT productid, name, listprice FROM Production.Product WHERE listprice = (SELECT COUNT (listprice) FROM Production.Product); statement. This statement will select any product whose list price is the same as the count of the list price in the Production.Product table. You want a statement that selects any product whose list price is the minimum list price in the Production.Product table.

Objective:
Work with Data

Sub Objective(s):
Implement sub-queries.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 4 - Lesson 1

SELECT (Transact-SQL)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189499.aspx

Subquery Fundamentals
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189575(SQL.105).aspx

Question Number (ID) : 199 (461P_3.1_19)

Your subordinate created a stored procedure named HumanResources.uspEmployeeDates. The stored procedure was meant to provide a list of all employees with their last names, first names, start dates, and end dates sorted by employee last name. Unfortunately, the stored procedure does not provide this information and, instead, only provides an unsorted list of employee last names and start dates. You want to modify the stored procedure to rectify this error.

Which of the following Transact-SQL statements can you use to accomplish this task?

```
1.  >>ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
  SET NOCOUNT ON;
  SELECT LastName, FirstName, StartDate, EndDate
  FROM HumanResources.vEmployeeDepartmentHistory
  ORDER BY StartDate;
```

```
2.  CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
  SET NOCOUNT ON;
  SELECT LastName, FirstName, StartDate, EndDate
  FROM HumanResources.vEmployeeDepartmentHistory
  ORDER BY StartDate;
```

```
3.  ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
  SET NOCOUNT ON;
  SELECT LastName, FirstName, StartDate, EndDate
  FROM HumanResources.vEmployeeDepartmentHistory
  ORDER BY LastName;             <Correct>
```

```
4.  CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
  SET NOCOUNT ON;
  SELECT LastName, FirstName, StartDate, EndDate
  FROM HumanResources.vEmployeeDepartmentHistory
  ORDER BY LastName;
```

Explanation:
You should use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
  SET NOCOUNT ON;
  SELECT LastName, FirstName, StartDate, EndDate
  FROM HumanResources.vEmployeeDepartmentHistory
  ORDER BY LastName;
```

This statement modifies the existing HumanResources.uspEmployeeDates procedure to generate a list of all employees with their last names, first names, start dates, and end dates sorted by employee last name.

You should not use the statement:

```
ALTER PROCEDURE HumanResources.uspEmployeeDates
AS
  SET NOCOUNT ON;
  SELECT LastName, FirstName, StartDate, EndDate
  FROM HumanResources.vEmployeeDepartmentHistory
  ORDER BY StartDate;
```

You want to modify the existing HumanResources.uspEmployeeDates procedure to generate a list of all employees with their last names, first names, start dates, and end dates sorted by employee last name. This modification sorts on the basis of employee StartDate.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY LastName;
```

You use the CREATE PROCEDURE statement to create a new stored procedure. Because there is an existing stored procedure with this name, running this statement will cause an error. You need to run the ALTER PROCEDURE statement.

You should not use the statement:

```
CREATE PROCEDURE HumanResources.uspEmployeeDates
AS
   SET NOCOUNT ON;
   SELECT LastName, FirstName, StartDate, EndDate
   FROM HumanResources.vEmployeeDepartmentHistory
   ORDER BY StartDate;
```

You use the CREATE PROCEDURE statement to create a new stored procedure. Because there is an existing stored procedure with this name, running this statement will cause an error. You need to run the ALTER PROCEDURE statement.

Objective:
Modify Data

Sub Objective(s):
Create and alter stored procedures (simple statements).

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 13 - Lesson 1

ALTER PROCEDURE (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms189762.aspx

Stored Procedures (Database Engine)
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms190782.aspx

TS: Querying Microsoft SQL Server 2012

Question Number (ID) : 200 (461P_1.3_05)

Which of the following statements about designing views is true?

1. You can create a view on a temporary table.
2. You can associate AFTER triggers with views.
3. >>You can associate INSTEAD OF triggers with views. <Correct>
4. You can use the COMPUTE clause in a view.

Explanation:
You can associate INSTEAD OF triggers with views.

You cannot associate AFTER triggers with views.

You cannot create a view on a temporary table.

You cannot use the COMPUTE clause in a view.

Objective:
Create Database Objects

Sub Objective(s):
Design views.

References:

Self-Paced Training Kit (Exam 70-461): Querying Microsoft SQL Server 2012
Chapter 9 - Lesson 1

CREATE VIEW (Transact-SQL)
Microsoft TechNet
Link: http://technet.microsoft.com/en-us/library/ms187956.aspx

Designing and Implementing Views
MSDN
Link: http://msdn.microsoft.com/en-us/library/ms189918(SQL.105).aspx