1. The Production.Product table contains a column named Name and a column named Price.
 You want to write a query that returns the values in these columns.
 In your resultset, you want the values in the Name column to be returned in a column named ProductName.

Which two of the following SELECT statements can you use?

```
SELECT Name AS ProductName, Price FROM Production.Product;, SELECT Name ProductName, Price FROM Production.Product;, - correct
SELECT Name AS ProductName, Price FROM Production.Product;
SELECT ProductName, Price FROM Production.Product;
SELECT Name ProductName, Price FROM Production.Product;
SELECT Name, ProductName, Price FROM Production.Product;
SELECT Name, Price FROM Production.Product AS ProductName
```

(Answer: 1,3)

2. The Sales.SalesOrder table contains the following columns:

```
    OrderNumber (integer)
    OrderDate (datetime)
    CustomerID (integer)
    Subtotal (money)
    Tax (money)
```

You write a query to return the OrderNumber for each row and a column named Total, which is calculated as Subtotal added to Tax.

Which of the following queries produces the required results?

```
SELECT OrderNumber, Subtotal, Tax FROM Sales.SalesOrder;
SELECT OrderNumber, Subtotal + Tax FROM Sales.SalesOrder;
SELECT OrderNumber, Subtotal + Tax AS Total FROM Sales.SalesOrder;
SELECT OrderNumber, Total FROM Sales.SalesOrder
```

 ANSWER: SELECT OrderNumber, Subtotal + Tax AS Total FROM Sales.SalesOrder;

3. The Sales.LineItem table includes the following columns:

```
    ProductID (integer)
    Quantity (integer)
    Price (varchar(6))
```

The Price column contains decimal numeric values, for example 14.79.

You write a query that returns each ProductID and a column named Subtotal, in which the values for Quantity and Price are multiplied together.

Which two of the following queries return the correct results?

```
SELECT ProductID, Quantity * CAST(Price AS money) AS Subtotal FROM Sales.LineItem;, SELECT ProductID, Quantity *
CONVERT(money, Price) AS Subtotal FROM Sales.LineItem;, - correct
SELECT ProductID, Quantity * Price AS Subtotal FROM Sales.LineItem;
SELECT ProductID, Quantity * CAST(Price AS money) AS Subtotal FROM Sales.LineItem;
SELECT ProductID, STR(Quantity) * Price AS Subtotal FROM Sales.LineItem;
SELECT ProductID, Quantity * CONVERT(money, Price) AS Subtotal FROM Sales.LineItem;
SELECT ProductID, CONVERT(money, Quantity * Price) AS Subtotal FROM Sales.LineItem;
```

Answer: 2, 4

4.

The HumanResources.Employee table contains the following values:

EmployeeID	FirstName	MiddleName	LastName
1	Dan	NULL	Drayton
2	Victoria	(an empty string)	Gray
3	Rajab	NULL	Shammas

You write the following query:

```
SELECT EmployeeID, FirstName + ' ' + NULLIF(MiddleName, '') + ' ' + LastName AS EmployeeName

FROM HumanResources.Employee;
```

In the results returned by this query, what does the EmployeeName column contain?

NULL for employees 1, 2, and 3.
NULL for employees 1 and 3, the FirstName and LastName for employee 2.
The FirstName, and empty string, and LastName for employees 1, 2, and 3.
The FirstName and LastName for employee 2, NULL for employees 1 and 3.

Answer: 1

5. The Sales.CustomerAddress table contains the following columns:

```
    CustomerID (integer)
    AddressID (integer)
    StreetAddress (nvarchar(100))
    City (nvarchar(30))
    CountryRegion (nvarchar(30))
```

You write a query to return City and CountryRegion values in the table. Each unique combination of City and CountryRegion should appear only once in the results.

Which of the following queries returns the correct results?

```
SELECT City, CountryRegion FROM Sales.CustomerAddress;
SELECT DISTINCT City, CountryRegion FROM Sales.CustomerAddress;
SELECT TOP 1 City, CountryRegion FROM Sales.CustomerAddress ORDER BY City, CountryRegion;
SELECT City + CountryRegion FROM Sales.CustomerAddress;
```

Answer: 2

6. The Production.Product table contains the following columns:

```
    ProductID (integer)
    ProductName (nvarchar(50))
    ProductCategoryID (integer)
    ListPrice (money)
```

You write the following SELECT query:

```
SELECT ProductName, ListPrice FROM Production.Product
```

You want the results of the query to return the products in ascending order of ProductCategoryID. The products in each category should be sorted by ListPrice, with the highest priced items in each category shown first.

Which of the following ORDER BY clauses should you add to the query?

```
ORDER BY ListPrice, ProductCategoryID
ORDER BY ProductCategoryID, ListPrice
ORDER BY ProductCategoryID ASC, ListPrice DESC
ORDER BY ListPrice DESC
```

Answer: 4

7. You write the following Transact-SQL query to retrieve data from the Sales.SalesOrderHeader table:

```
SELECT SalesOrderID, OrderDate, Amount

FROM Sales.SalesOrderHeader
```

You want to limit the results to include only sales in the first calendar quarter of 2015. Which two of the following WHERE clauses can you use to accomplish this goal? (assume US formats for dates, for example 3/31/2015 represents March 31st 2015)

```
WHERE OrderDate BETWEEN '1/1/2015' AND '3/31/2015'
WHERE OrderDate BETWEEN '1/1/2015' AND '3/31/2015'
WHERE OrderDate > '1/1/2015' AND OrderDate < '3/31/2015'
WHERE OrderDate >= '1/1/2015' AND OrderDate <= '3/31/2015'
WHERE OrderDate BETWEEN '12/31/2014' AND '4/1/2015'
WHERE OrderDate IN ('1/1/2015', '3/31/2015')
```

Answer: 1, 3

8. You write the following query to retrieve details of products that have been ordered:

```
SELECT p.ProductName, p.SupplierID, o.OrderDate, o.Quantity
FROM Sales.Order AS o

JOIN Inventory.Product AS p ON o.ProductID = p.ProductID;
```

You want to modify this query so that it includes products that have not been ordered, with NULL values for the OrderDate and Quantity columns.

To which of the following should you change the JOIN clause?

```
INNER JOIN Inventory.Product AS p ON o.ProductID = p.ProductID;
LEFT JOIN Inventory.Product AS p ON o.ProductID = p.ProductID;
RIGHT JOIN Inventory.Product AS p ON o.ProductID = p.ProductID;
CROSS JOIN Inventory.Product AS p;
```

Answer: 4

9.

You write the following queries to retrieve all customers and customers who have placed orders.

```
SELECT CustomerID, FirstName, LastName
FROM Sales.Customer
_____________
SELECT o.CustomerID, c.FirstName, c.LastName
FROM Sales.Order AS o
JOIN Sales.Customer AS c ON o.CustomerID = c.CustomerID

ORDER BY LastName
```

Which statement should you use to combine the queries so that the results include only customers who have not placed any orders?

Answer: EXCEPT

10.

The sales department occasionally run promotions that offer a discount on sales orders. Some promotions are associated with a specific product, while others are more general.

The Inventory.Product table contains the following columns:

    ProductID (integer)
    ProductName (nvarchar(50))
    ListPrice (money)

The Sales.Promotion table contains the following columns:

    PromotionID (integer)
    StartDate (date)
    EndDate (date)
    ProductID (integer - can be NULL)
    Discount (decimal)

You write the following query, which returns all product-based promotions and the products with which they are associated:

```
SELECT prm.StartDate, prm.EndDate, prm.Discount, prd.ProductName, prd.ListPrice
FROM Sales.Promotion as prm

JOIN Inventory.Product as prd ON prm.ProductID = prd.ProductID;
```

How should you modify the JOIN clause so that the query returns all promotions (with NULL ProductName and ListPrice values for promotions that are not associated with a product) and all products (with NULL StartDate, EndDate, and Discount values for products that have never been associated with a promotion)?

```
LEFT OUTER JOIN Inventory.Product as prd ON prm.ProductID = prd.ProductID;
FULL OUTER JOIN Inventory.Product as prd ON prm.ProductID = prd.ProductID;
RIGHT OUTER JOIN Inventory.Product as prd ON prm.ProductID = prd.ProductID;
CROSS JOIN Inventory.Product as prd
```

ANSWER: 2

11.

You write the following query to return the last date on which each product was sold. Products that have never been sold should return a NULL in the LastSale column.

```
SELECT p.ProductName, MAX(o.OrderDate) AS LastSold
FROM Sales.SalesOrder AS o
JOIN Sales.SalesOrderDetail AS od ON o.SalesOrderID = od.SalesOrderID

RIGHT JOIN Inventory.Product AS p ON od.ProductID = p.ProductID
```

Which of the following clauses should you add to complete the query?

```
ORDER BY LastSold DESC;
WHERE MAX(o.OrderDate) IS NOT NULL;
GROUP BY p.ProductName;
HAVING o.OrderDate IS NOT NULL
```

Answer: 3

12. You write the following query:

```
SELECT ProductID, IIF(ISNUMERIC(ISNULL(Code, 0)) = 1,
                                    IIF(ISNULL(TRY_CAST(Code AS decimal), 0) >= 20, 'Special', 'Standard'),
                                     'Custom') AS ItemType

FROM Inventory.Product;
```

What value is returned in the ItemType column for a product with a Code value of 'X21'?

ANSWER: Custom

13. You write the following query to return the lowest sales amount for each salesperson.

```
SELECT Salesperson, MIN(Amount) AS LowestSale
FROM Sales.vSalesOrders
GROUP BY Salesperson

_____________________________________________
```

Which clause should you add to the query so that the results include only salespeople whose lowest sales amount is greater than the average sale amount for all sales

Answer: Having MIN(Amount) > (SELECT AVG(AMOUNT) FROM Sales.vSalesOrders)

14.

The Sales.Customer table includes the following columns:

```
    CustomerID (integer)
    FirstName (nvarchar(50))
    LastName (nvarchar(50))
```

The Sales.SalesOrder table includes the following columns:

```
    SalesOrderNumber (integer)
    OrderDate (date)
    CustomerID (integer)
    Amount (money)
```

Some customers have placed mulitple orders over a period of years. You write the following query to retrieve the last date on which each customer placed an order:

```
SELECT c.CustomerID, c.FirstName, c.LastName,
            -- correlated subquery goes here
                AS LastOrderDate

FROM Sales.Customer AS c;
```

Which of the following subqueries should you use to complete the query?

```
-- Option 1
    (SELECT MAX(o.OrderDate)
    FROM Sales.SalesOrder AS o
    WHERE o.CustomerID = c.CustomerID)
-- Option 1 <div style="margin-bottom:0px;">    (SELECT MAX(o.OrderDate)</div> <div style="margin-bottom:0px;">    FROM
Sales.SalesOrder AS o</div> <div style="margin-bottom:10px;">    WHERE o.CustomerID = c.CustomerID)</div> - correct
-- Option 2
    (SELECT MAX(c.OrderDate)
    FROM Sales.SalesOrder AS c)
-- Option 3
    (SELECT o.OrderDate
    FROM Sales.SalesOrder AS o
    WHERE o.CustomerID = c.CustomerID)
-- Option 4
    (SELECT MAX(c.OrderDate)
    FROM Sales.Customer AS c)
```

Answer: 4

15.

You write the folowing code in SQL Server Management Studio

```
DECLARE @tab AS table(ProductID integer, StockCount integer)
INSERT INTO @tab
SELECT ProductID, InStock + OnOrder
FROM Inventory.Product;
GO

SELECT * FROM @tab;
```

When you execute the code, an error occurs. Which two of the following actions could you take to prevent the error?

Modify the INSERT statement to:

```
INSERT INTO @tab
```

```
SELECT ProductID, InStock
FROM Inventory.Product;
```

Remove the GO command

Use a temporary table named #tab instead of the @tab variable
Add a second GO command after the final SELECT statement

Answer: 2,3

16.
You write the following query that includes a common table expression:

```
WITH CustomerNames (CustomerID, CustomerName)
AS
(
  SELECT CustomerID, CONCAT(FirstName, CONCAT(' ', LastName))
  FROM Sales.Customer
)

-- Retrieve customer names and total revenue
```

Which of the following SELECT statements can you use to complete the query so that it uses the CTE and returns the customer name and total sales amount for each customer that has placed an order?

```
-- Option 1
UNION ALL
SELECT CustomerID, SUM(Amount) AS CustomerRevenue
FROM Sales.SalesOrder

GROUP BY CustomerName;
-- Option 2
SELECT c.CustomerName, SUM(o.Amount) AS CustomerRevenue
FROM CustomerNames AS c
JOIN Sales.SalesOrder AS o ON o.CustomerID = c.CustomerID

GROUP BY c.CustomerName;


-- Option 3
SELECT c.FirstName + ' ' c.LastName AS CustomerName, SUM(o.Amount) AS CustomerRevenue
FROM Sales.Customer AS c
JOIN Sales.SalesOrder AS o ON o.CustomerID = c.CustomerID

GROUP BY CustomerName;
-- Option 4
```

```
SELECT c.CustomerName, (SELECT SUM(Amount) FROM Sales.SalesOrder) AS CustomerRevenue
FROM CustomerNames AS c

GROUP BY c.CustomerName;
```

Answer: option 2

17. You create the following query:

```
SELECT c.City, c.CountryRegion, SUM(o.TotalDue) AS Revenue
FROM Sales.Customer AS c
JOIN Sales.SalesOrder AS o ON o.CustomerID = c.CustomerID

GROUP BY GROUPING SETS (CountryRegion, (CountryRegion, City), ())
```

Which three of the following values does this query return?
A grand total of revenue for all sales., A revenue total for each CountryRegion., A revenue total for each CountryRegion and City combination., - correct
A grand total of revenue for all sales.
A revenue total for each individual customer.
A revenue total for each CountryRegion.
A revenue total for each City.
A revenue total for each CountryRegion and City combination.

Answers: 1,3,5

18.

Question 18 (1/1 point)

A consultant has sent you some sales projections in the following format:

```
MonthOfYear     Asia    Europe  N.America
January 20125   36999   27675
February        22165   38779   29079
March   25149   40016   31472
```

You load this data into a temporary table named #forecast.

The sales manager has asked for the forecast data in the following format:

```
MonthOfYear     Region  Forecast
January Asia    20125
January Europe  36999
January N.America       27675
February        Asia    22165
February        Europe  38779
February        N.America       29079
March   Asia    25149
March   Europe  40016
March   N.America       31472
```

Which of the following queries returns the correct results?

```
-- Option 1
SELECT * FROM
(SELECT Region, Forecast
 FROM #forecast) as src
PIVOT (SUM(Forecast) FOR Region IN(Asia, Europe, N.America)) AS Pvt
-- Option 2
SELECT MonthOfYear, CountryRegion, Forecast
FROM
(SELECT MonthOfYear, Asia, Europe, N.America
 FROM #forecast) AS pvt
UNPIVOT(Forecast FOR CountryRegion IN (Asia, Europe, N.America)) AS unpvt
ORDER BY MonthOfYear, CountryRegion

-- Option 3
SELECT MonthOfYear, SUM(Asia) AS Asia, SUM(Europe) AS Europe, SUM(N.America) AS N.America
FROM #forecast
GROUP BY ROLLUP;
-- Option 4
SELECT MonthOfYear, SUM(Asia) AS Asia, SUM(Europe) AS Europe, SUM(N.America) AS N.America
FROM #forecast
GROUP BY CUBE;
```

Answer: Option 2

19.
The Sales.Customer table contains the following columns:

Column Name	Data Type	NULL	Default
CustomerID	integer	False	Identity
FirstName	nvarchar(50)	False	
MiddleName	nvarchar(50)	True	
LastName	nvarchar(50)	False	
Active	bit	False	1

You want to insert a new record for the following customer:

    First Name: Sophia
    Middle Name: none

```
    Last Name: Garner
    Active: yes
```

Which two of the following INSERT statements successfully inserts the record?

```
-- Option 1
INSERT INTO Sales.Customer (FirstName, LastName)
VALUES

('Sophia', 'Garner');

-- Option 2
INSERT INTO Sales.Customer
VALUES

('Sophia', 'Garner');
-- Option 3
INSERT INTO Sales.Customer
VALUES

('Sophia', NULL, 'Garner', DEFAULT);
-- Option 4
INSERT INTO Sales.Customer
VALUES

(DEFAULT, 'Sophia', DEFAULT, 'Garner', DEFAULT);
-- Option 5
INSERT INTO Sales.Customer
VALUES

(IDENTITY, 'Sophia', NULL, 'Garner', NULL);
```

ANS: 1,3

20.

The Inventory.Product table contains the following columns:

```
    ProductID (integer)
    ProductName (nvarchar(100))
    ListPrice (money)
    Discontinued (bit)
    LastUpdated (datetime)
```

You want to increase the list price of all non-discontinued products by 10%. The LastUpdated column for all records that are modified should be set to the current date and time.

Which of the following UPDATE statements performs the required modification correctly?

```
-- Option 1
UPDATE Inventory.Product
SET ListPrice = ListPrice * 1.1,
        Discontinued = 0

WHERE LastUpdated = GETDATE();
-- Option 2
UPDATE Inventory.Product
SET ListPrice = ListPrice * 1.1,
WHERE LastUpdated = GETDATE()

AND Discontinued = 0;
-- Option 3
UPDATE Inventory.Product
SET ListPrice = ListPrice * 1.1,
        LastUpdated = GETDATE(),

        Discontinued = 0;
-- Option 4
UPDATE Inventory.Product
SET ListPrice = ListPrice * 1.1,
        LastUpdated = GETDATE()

WHERE Discontinued = 0;
```

Ans: Option 4

quizes

You need to determine the most recently inserted IDENTITY column in a specific table.

Which statement should you use?
SELECT @@IDENTITY
SELECT SCOPE_IDENTITY()
SELECT IDENT_CURRENT('table_name')
SELECT NEXT VALUE FOR table_name

Ans. 3

ou use an INSERT statement to insert specific values into a table.

For which of the following columns do you not need to provide values?
IDENTITY columns, Nullable columns that have a default value

Non-Nullable columns that have no default value
Nullable columns that have no default value
Non-Nullable columns that have a default value
Nullable columns that have a default value

Ans. 1,3,4,5

You want to use a single statement to load new product catalog data from a staging table into the Production.Products table. Products are uniquely identified by a ProductID value. You want to apply the following logic:

If the ProductID exists in the staging table but not the Production.Products table, insert a new row into the Production.Products table.

If the ProductID exists in the Production.Products table but not the staging table, update the row for that product in the Production.Products table to set its Discontinued column to 1.

If the ProductID exists in both tables, update the row in the Production.Products table and set all column values to match the values in the staging table.

What type of statement should you use?
An INSERT statement with a FROM clause
An UPDATE statement with a FROM clause
A MERGE statement A MERGE statement
A SELECT INTO statement

Ans: A MERGE statement A MERGE statement

You write a query that returns the Name and Price columns from a table named Product in the Production schema. In the resulting rowset, you want the Name column to be named ProductName.

Which of the following Transact-SQL statements should you use?

```
SELECT * FROM Product;
SELECT ProductName, Price FROM Production.Product;
SELECT Name AS ProductName, Price FROM Production.Product;
SELECT Name, Price FROM dbo.Product;
```

You need to retrieve data from a column that is defined as char(1). If the value in the column is a digit between 0 and 9, the query should return it as an integer value. Otherwise, the query should return NULL.

Which two functions can you use to accomplish this?
TRY_CAST, TRY_CONVERT
TRY_CAST
CAST
STR
CONVERT
TRY_CONVERT

Ans: 1, 5

You write a Transact-SQL query that returns the Cellphone column from the Sales.Customer table. Cellphone is a varchar column that permits NULL values. For rows where the Cellphone value is NULL, your query should return the text 'None'. Select the correct function to complete the following query:

```
SELECT FirstName, LastName, __________(Cellphone, 'None') AS Cellphone

FROM Sales.Customer;
```

ANs. ISNULL

You write a Transact-SQL query to list the available sizes for products. Each individual size should be listed only once.

Which query should you use?

```
SELECT Size FROM Production.Product;
SELECT DISTINCT Size FROM Production.Product;
SELECT ALL Size FROM Production.Product;
SELECT TOP 1 Size FROM Production.Product ORDER BY Size;
```

Ans: 2

The organization you work for has a fiscal year that runs from July to June. For example, FY 2015 covers the period from July 1st 2014 to June 30th 2015. The Accounts.Transactions table contains records of financial transactions, and the table includes an FY column to indicate in which fiscal year each transaction occurred.

You need to write a script that returns all transaction data for the current fiscal year.
Which two of the following scripts accomplishes this?

```
 -- Option 1
SELECT * FROM Sales.SalesOrder WHERE FY = YEAR(GETDATE()) + 1;

-- Option 2
DECLARE @fy int;
IF MONTH(GETDATE()) <= 6
 SET @fy = YEAR(GETDATE());
ELSE
 SET @fy = YEAR(GETDATE())+1;

SELECT * FROM Sales.SalesOrder WHERE FY = @fy

-- Option 3
DECLARE @fy int;
SET @fy = YEAR(GETDATE());

SELECT * FROM Sales.SalesOrder WHERE FY = @fy

-- Option 4
DECLARE @fy int = YEAR(GETDATE());
IF MONTH(GETDATE()) > 6

 SET @fy = @fy + 1;

SELECT * FROM Sales.SalesOrder WHERE FY = @fy
-- Option 5
```

```
DECLARE @fy int;
SET @fy = YEAR(GETDATE());
IF @fy > 6
 SET @fy = YEAR(GETDATE()) + 1;

SELECT * FROM Sales.SalesOrder WHERE FY = @fy
```

ANS: 2,4

How many times will the loop in the following code be executed?

```
DECLARE @i int = 1;
WHILE @i < 10
BEGIN
 PRINT @i;
END
```

Ans: infinite

You are using the following Transact-SQL code to generate test data. The code loops once for each day between a year ago from the current date and today, and for each day it inserts rows into the #Orders temporary table for every combination of product and customer. You want to stop inserting data when the #Orders table contains over 1 million rows.

```
DECLARE @d DATETIME = DATEADD(dd, -365, GETDATE());
WHILE @d < GETDATE()
BEGIN
 INSERT INTO #Orders
 SELECT @d, c.CustomerID, p.ProductID
 FROM Sales.Customer AS c
 CROSS JOIN Production.Product AS p;
 DECLARE @c int;
 SELECT @c = COUNT(*) FROM #Orders;
 IF @c > 1000000
  _____________;
 SET @d = DATEADD(dd, 1, @d);

END
```

Select the statement that is missing in the code.

ANS: BREAK

You write a query that returns a list of all sales employees that have taken sales orders. Employees who have not taken sales orders should not be included in the results.

Select the appropriate join type to complete the following query.

```
SELECT e.FirstName, e.LastName, s.Amount
FROM HumanResources.Employee AS e

________________ Sales.SalesOrder AS s ON s.SalesPersonID = e.EmployeeID;
```

ANS: INNER JOIN

The Products table contains a row for each product that your company sells, and the SalesItems table contains a row for each line item that has been sold, including the ProductID value of the sold product from the Products table.

You want to write a query that returns rows from the Products table and the SalesItems table. The results should include all products, with sales data from the SalesItems table for products that have been sold, and NULL values for products that have not been sold.

Which type of join should you use?
An outer join
A cross join
An inner join
An equi-join
A self-join

Ans: 1

You write a query to generate test data from a table of customers and a table of products. You want to generate a set of sales orders in which every customer has ordered every product.

What kind of join should you use?
An inner join
A full outer join
A cross join
An equi-join
A self-join

ANS: 3

The Sales.SalesOrderDetail table contains a row for every line item sold, and includes a ProductID column to identify the product that was ordered.

The Production.Product table contains a row for every product, and includes a Discontinued column where the value 1 indicates that the product has been discontinued.

What result does the following query return?

```
SELECT DISTINCT ProductID
FROM Sales.SalesOrderDetail
EXCEPT
SELECT ProductID
FROM Production.Product

WHERE Discontinued = 1;
```

All products that have been sold and all products that have been discontinued.
All discontinued products that have been sold.
All non-discontinued products that have been sold.
All non-discontinued products that have never been sold

ANS: 3

You write the following Transact-SQL query that returns the CustomerName, StreetAddress, City, and PostalCode columns from the Sales.BillingAddress table:

```
SELECT CustomerName, StreetAddress, City, PostalCode

FROM Sales.BillingAddress;
```

You have also written the following query to retrieve the same fields from the Sales.ShippingAddress table:

```
SELECT CustomerName, StreetAddress, City, PostalCode

FROM Sales.ShippingAddress;
```

What set operator can you use to combine the results of these queries and return only rows for customers whose billing address is the same as their shipping address?
UNION
UNION ALL
EXCEPT
INTERSECT

ANs: 4

You write a query that returns the set of products that are available in the color 'Red' or the size 'XL' by combining the results of two queries. Any products that are available in both red and size XL should be included as a duplicate row in the resultset.

Select the appropriate missing keyword.

```
SELECT ProductID, Name, Color, Size
FROM Production.Product
WHERE Color = 'Red'
__________
SELECT ProductID, Name, Color, Size
FROM Production.Product
WHERE Size = 'XL'

ORDER BY ProductID;
```

Ans: UNION ALL

What value will the following query return in the OrderStatus column for rows with a Status value of 2?

```
SELECT OrderNumber, CHOOSE(Status, 'Ordered', 'Shipped', 'Delivered') AS OrderStatus

FROM Sales.SalesOrderHeader
```

ANS: Shipped

The Sales.SalesOrderDetail table contains a row for each line item that has been ordered. Each row includes the following columns:

- ProductID: The unique ID of the product that was ordered.
- Quantity: How many units of the product were ordered.
- UnitPrice: The price per unit charged for the product.

You run the following query:

```
SELECT COUNT(ProductID) AS ProductIDCount, COUNT(Quantity) AS QuantityCount, SUM(UnitPrice) AS PriceSum

FROM Sales.SalesOrderDetail;
```

What does the value returned by the query for the QuantityCount column represent?

The number of orders
The total number of product units that have been ordered
The number of distinct products that have been ordered
The number of rows containing a Quantity value

You write a query that counts customers. Each customer lives in a city, and each city is located within a state. The results should show the count of customers in each city in every state.

Select the code required to complete the query:

```
SELECT COUNT(CustomerID) AS CustomerCount, City AS CustomerCity, State AS CustomerState
FROM Sales.Customer

______________________ ;
```

ANS: GROUP BY State, City

You write a stored procedure that updates the price of a product. The product and its new price are specified parameters named @ProductID and @NewPrice. If the specified product does not exist and no rows are affected by the update operation, you want a custom error to be returned to the calling client application.

The partial stored procedure code is shown here:

```
UPDATE Production.Product
SET ListPrice = @NewPrice
WHERE ProductID = @ProductID;
IF @@ROWCOUNT < 1

-- create an error here
```

Which two of these statements could you use to achieve the required result.

```
THROW 50001, 'Product not found', 0;
SELECT @@ERROR, 'Product not found';
RAISERROR('Product not found', 16, 0);
CATCH ERROR_MESSAGE();
THROW
```

Ans: 1, 3

You write the following Transact-SQL code, which deletes customers and their orders:

```
BEGIN TRY
  BEGIN TRANSACTION
    DELETE FROM Sales.SalesOrder WHERE CustomerID = @CustomerID;
    DELETE FROM Sales.Customer WHERE CustomerID = @CustomerID;
  COMMIT TRANSACTION
END TRY
BEGIN CATCH
  --code to handle exceptions

END CATCH
```

You want to implement the CATCH block so that:

    Any transactions in process are rolled back.
    The original exception is re-thrown to the calling application.

The XACT_ABORT option is not enabled.

Which code block implements the CATCH block correctly?

```
--Option 1

THROW;
--Option 2
ROLLBACK TRANSACTION

THROW 50001, ERROR_MESSAGE(), 0;

-- Option 3
IF @@TRANCOUNT > 0
  ROLLBACK TRANSACTION

THROW;
-- Option 4
IF ERROR_NUMBER() > 50000
  ROLLBACK TRANSACTION

THROW @@ERROR;
```

Ans: 4

You write a stored procedure that transfers funds from one account to another. You use the following code to perform this task:

```
UPDATE Banking.Account
SET Balance = Balance - @amount
WHERE AccountNumber = @SourceAccount;
UPDATE Banking.Account
```

```
SET Balance = Balance + @amount

WHERE AccountNumber = @TargetAccount;
```

If an error occurs after the second UPDATE statement has started, but before it has completed, what will be the status of the data?
Both the source and target balances will have been updated.
Neither the source or target balances will have been updated.
Only source account balance will have been updated.
Only target account balance will have been updated

Ans: 3

You write a Transact-SQL script to delete a customer and all of their purchase history. The code must ensure that all relevant records are deleted, or if an error occurs part-way through the operation, that no rows are deleted.

You write the following code:

```
SET XACT_ABORT ON
BEGIN TRY
 BEGIN TRANSACTION
  DELETE FROM Sales.SalesOrder WHERE CustomerID = @CustomerID;
  DELETE FROM Sales.Customer WHERE CustomerID = @CustomerID;
 _____________
END TRY
BEGIN CATCH
 THROW 50001, 'The operation failed', 0;
END CATCH;

SET XACT_ABORT OFF
```

Which statement should you use to replace the missing code?

ANS: COMMIT TRANSACTION

The Sales.SalesOrder table contains details of sales orders made by customers. Customers are identified by a unique CustomerID column, which is the primary key of the Sales.Customer table. The Sales.Customer table also contains a City column that contains the name of the city in which the customer lives.

Select the appropriate WHERE clause for the following query so that it returns sales order data for all customers who live in New York:

```
SELECT SalesOrderID, OrderDate, Amount
```

```
FROM Sales.SalesOrder
WHERE CustomerID = (SELECT CustomerID FROM Sales.Customer WHERE City = 'New York');
WHERE CustomerID IN (SELECT City FROM Sales.Customer WHERE City = 'New York');
WHERE CustomerID IN (SELECT CustomerID FROM Sales.Customer WHERE City = 'New York');
WHERE City = 'New York';
```

ANS: 3

Select the appropriate subquery WHERE predicate to complete the following query so that the OrderCount column contains a count of the orders placed by each customer.

```
SELECT CustomerID, CompanyName,
 (SELECT COUNT(*) FROM Sales.SalesOrder AS o
  WHERE ___________________________________) AS OrderCount

FROM Sales.Customer AS c;
```

ANS: o.CustomerID=c.CustomerID

Your database contains a table-valued function named dbo.fn_CurrentYearOrders that returns the SalesOrderID, OrderDate, and SalesAmount for each order placed in the current year by the customer specified in a CustomerID parameter.

You write the following query to return the CompanyName column from the Sales.Customer table and the sum of revenue for each customer in the current year, and you plan to use the dbo.fn_CurrentYearOrders function to accomplish this. Only data for customers who have placed orders in the current year should be included in the results.

```
SELECT c.CompanyName, SUM(cyo.SalesAmount) AS Revenue
FROM SalesLT.Customer AS c
_____________________________________________________

GROUP BY c.CompanyName;
```

Which line of code is the correct one for the query?

```
CROSS APPLY dbo.fn_CurrentYearOrders(c.CustomerID)
OUTER APPLY dbo.fn_CurrentYearOrders(c.CustomerID) AS cyo
CROSS JOIN dbo.fn_CurrentYearOrders(c.CustomerID) AS cyo
OUTER JOIN dbo.fn_CurrentYearOrders(c.CustomerID) AS cyo
```

ANs: 1

You write a Transact-SQL script and want to store a rowset in a temporary object that will be automatically deleted. The script will reference the temporary rowset from mulitple queries in the same batch.

Which two types of object can you use to achieve your goals?

View
Temporary table
Table variable
Table-valued function
Derived table
Common table expression

ANs: 2 and 3

You write a query that contains a derived table. The derived table should retrieve each product category ID and the count of products in those categories from the Production.Product table. The outer query then joins the derived table to the Production.ProductCategory table to display the product category names and their product counts.

Your outer query looks like this:

```
SELECT cat.Name AS Category, prd_cnts.ProductCount
FROM Production.ProductCategory AS cat
JOIN
-- (derived table goes here ) --

ON cat.ProductCategoryID = prd_cnts.CategoryID;
```

Which two of the following derived table definitions can you use to complete the query?

```
-- Option 1
(SELECT ProductCategoryID AS CategoryID, COUNT(ProductID) AS ProductCount
FROM Production.Product

GROUP BY ProductCategoryID) AS prd_cnts
-- Option 2
(SELECT ProductCategoryID AS CategoryID, COUNT(ProductID) AS ProductCount
FROM Production.Product AS prd_cnts

GROUP BY ProductCategoryID)
-- Option 3
(SELECT ProductCategoryID, COUNT(ProductID)
FROM Production.Product

GROUP BY ProductCategoryID) AS prd_cnts(CategoryID, ProductCount)
-- Option 4
(SELECT ProductCategoryID, COUNT(ProductID)
FROM Production.Product AS prd_cnts (CategoryID, ProductCount)

GROUP BY ProductCategoryID)
```

ANS: 2 and 4