CIS56

Quiz 1 Sample questions

1. The BEGIN and END keywords delimit a statement block and limit the scope of variables declared in that statement block, like curly braces ({ }) in C#.

   a. True

   b. False

   ANSWER: False

2. The ____keyword forces a WHILE loop to terminate immediately.

   ANSWER: BREAK

3. [Choose one] SQL NULL represents which of the following?
   a. An unknown or missing value
   b. The number 0
   c. An empty (zero-length) string
   d. All of the above

   ANSWER: a

4. Just like database triggers, stored procedures are stored in the database.

   a. True

   b. False

   ANSWER: True

5. One of the major advantages of stored procedures is that they can be used to encapsulate and represent business transactions.

   a. True

   b. False

   ANSWER: True

6. One of the disadvantages of stored procedures is that they increase network traffic.

   a. True

   b. False

   ANSWER: False

7. Stored procedures help reduce code duplication by means of code isolation and code sharing.

   a. True

   b. False

   ANSWER: True

8. Stored procedures must have at least one argument.

   a. True

   b. False

   ANSWER: False

9. Variables can be declared inside a stored procedure.

a. True

b. False

ANSWER: True

10. Stored procedures are executed using the EXEC command.

a. True

b. False

ANSWER: True

11. A stored function is another name for a stored procedure.

a. True

b. False

ANSWER: False

12. The statement SELECT * FROM T1, T2 produces a(n)__join.

a. cross b. natural
c. equi- d. full

ANSWER: a

13. The__clause is used to restrict the output of a GROUP BY query by applying a conditional criteria to the grouped rows.

ANSWER: HAVING

14. A(n)_view is a view that can be used to update attributes in the base table(s) that are used in the view.

ANSWER: updatable

15. List 3 difference between SQL user defined Function and SQL Stored procedure

Differences between Stored Procedure and User Defined Function in SQL Server

Sr.No.	User Defined Function	Stored Procedure
1	Function must return a value.	Stored Procedure may or not return values.
2	Will allow only Select statements, it will not allow us to use DML statements.	Can have select statements as well as DML statements such as insert, update, delete and so on
3	It will allow only input parameters, doesn't support output parameters.	It can have both input and output parameters.
4	It will not allow us to use try-catch blocks.	For exception handling we can use try catch blocks.
5	Transactions are not allowed within functions.	Can use transactions within Stored Procedures.
6	We can use only table variables, it will not allow using temporary tables.	Can use both table variables as well as temporary table in it.
7	Stored Procedures can't be called from a function.	Stored Procedures can call functions.
8	Functions can be called from a select statement.	Procedures can't be called from Select/Where/Having and so on statements. Execute/Exec statement can be used to call/execute Stored Procedure.
9	A UDF can be used in join clause as a result set.	Procedures can't be used in Join clause

16. SQL Server supports three types of user-defined functions: list them?

Answer:

- Scalar functions
- Inline table-valued functions
- Multi-statement table-valued functions

17. Write a scalar UDF that converts a measurement in degrees Fahrenheit to degrees Celsius. The UDF should accept a single float parameter and return a float result.   The algebraic formula for converting Fahrenheit measurements to the Celsius scale is: $C=(F-32.0) \times (5/9)$, where F is the measurement in degrees Fahrenheit and C is the measurement in degrees Celsius.

Answer:

```
CREATE FUNCTION dbo.FahrenheitToCelsius (@Degrees float)
RETURNS float
AS
        BEGIN
                RETURN (@Degrees - 32.0) * (5.0 / 9.0);
 END;
```

18. Write a sample call to the following function and what is the result:

```
CREATE FUNCTION dbo.vaValidEmail(@EMAIL varchar(100))

RETURNS bit as
BEGIN
DECLARE @bitRetVal as Bit
IF (@EMAIL <> '' AND @EMAIL NOT LIKE '_%@__%.__%')
SET @bitRetVal = 0 -- Invalid
ELSE
SET @bitRetVal = 1 -- Valid
RETURN @bitRetVal
END
```

Sample Answer: `SELECT dbo.vaValidEmail('MyEmail@wvm.edu')`
Result : 1

19. Find What is Wrong in this Query?

SELECT subject_code, AVG (marks) FROM students
WHERE AVG(marks) > 75 GROUP BY subject_code;

Answer:
The WHERE clause cannot be used to restrict groups. Instead, the HAVING clause should be used.

```
SELECT subject_code, AVG (marks)
FROM students
HAVING AVG(marks) > 75
GROUP BY subject_code;
```

20. List the command to create the following types of tables:.

- Regular User Tables
- Temporary Tables
- Global Temporary
- Table Variables

Answer (Only command is required, please notice the difference between all the commands is only in red

Regular User Tables (which holds data of user for later on processing and reporting purpose. These are also called physical tables as they physically resides at hard drive until you DROP them intentionally.

```
CREATE TABLE  TestTable
  (
    [TestTableID] [int],
    [FirstCol] [varchar](200),
    [SecondCol] [int]
  )
```

Temporary Tables (created to hold temporary data regarding intermediate results of different quires. These tables will be drooped automatically once the store procedure is executed (if they are used in stored procedure) or once the session is over.)

```
CREATE TABLE #Temp_TestTable
  (
   [TestTableID] [int],
   [FirstCol] [varchar](200),
   [SecondCol] [int]
 )
```

Global Temporary Tables (These are just like simple temporary tables but are available to all sessions and will only be dropped automatically when last session of database will be closed. If single session is active, global temporary tables will remain available)

```
CREATE TABLE ##Temp_TestTable
 (
   [TestTableID] [int],
   [FirstCol] [varchar](200),
   [SecondCol] [int]
 )
```

Table Variables (are just like scalar variables which possess structure of a table and can hold records for intermediate results of different quires)

```
   DECLARE @Temp_TestTable TABLE
 (
   [TestTableID] [int],
   [FirstCol] [varchar](200),
   [SecondCol] [int]
 )
```