AdventureWorks Database – 2019

Get the entity relationship diagram: Click here

1. Create a view called dbo.vw_Products that displays a lists of products from Production.Product table joined to Production.ProductCostHistory table. Include columns that describe the product and show the cost history for each product. Test the view by creating a query that retrieves data from the view.

```
DROP VIEW IF EXISTS dbo.vw_Products;
CREATE VIEW dbo.vw_Products AS
SELECT P.ProductID, P.Name, P.ProductNumber, P.color, P.ReorderPoint, PCH.StandardCost
FROM Production.Product P
JOIN Production.ProductCostHistory PCH ON PCH.ProductID = P.ProductID
```

Test view:

```
SELECT * FROM dbo.vw_Products
```

2. Create a view called dbo.vw_CustomerTotals that display the total sales from the TotalDue per year and month for each customer. Test the view by creating a query that retrieves data from the view.

```
DROP VIEW IF EXISTS dbo.vw_CustomerTotals;
CREATE VIEW dbo.vw_CustomerTotals AS
SELECT CustomerID, YEAR(OrderDate) AS YearOrder , MONTH(OrderDate) AS MonthOrder,
SUM(TotalDue) AS SumTotalDue_YearMonth
FROM Sales.SalesOrderHeader
GROUP BY CustomerID, YEAR(OrderDate), MONTH(OrderDate)
```

Test view:

```
SELECT *
FROM dbo.vw_CustomerTotals
ORDER by CustomerID
```

3. Create a user-defined function called dbo.fn_AddTwoNumbers that accept two integer parameters. Return the value that is the sum of the two number. Test the function.

```
DROP FUNCTION IF EXISTS dbo.fn_AddTwoNumbers;
CREATE FUNCTION dbo.fn_AddTwoNumbers(@Num1 INT, @Num2 INT)
RETURNS INT AS
BEGIN
    DECLARE @SUM INT;
    SET @SUM = @Num1 + @Num2;
    RETURN @SUM
END;
```

Test function:

```
SELECT dbo.fn_AddTwoNumbers(10, 2) AS SUM;
```

4. Create a Function dbo.fn_RemoveNumbers that removes any numeric characters from a VARCHAR(250) string. Test the function. Hint: ISNUMBERIC function checks to see whether a string is numeric. Check the online documentation to see how to use it.

```
DROP FUNCTION IF EXISTS dbo.fn_RemoveNumbers;
CREATE FUNCTION dbo.fn_RemoveNumbers(@inputString VARCHAR(250))
RETURNS VARCHAR(250) AS
BEGIN
    DECLARE @position INT = 1;
    DECLARE @LENGTH INT = LEN(@inputString);
    DECLARE @CHAR CHAR(1);
    DECLARE @outputString VARCHAR(250) = '';
    WHILE @position <= @LENGTH
    BEGIN
        SET @CHAR = SUBSTRING(@inputString, @position, 1)
        IF ISNUMERIC(@CHAR) = 0
        BEGIN
            SET @outputString += @CHAR
        END;
        SET @position += 1
    END;
    RETURN @outputString;
END;
```

Test function:

```
SELECT dbo.fn_RemoveNumbers('asdasd123ad123');
```

5. Write a Function called dbo.fn_FormatPhone that takes a string of ten numbers. The function will format the string into this phone number format “ (###) ### - #### ”. Test the function.

```
DROP FUNCTION IF EXISTS dbo.fn_FormatPhone;
CREATE FUNCTION dbo.fn_FormatPhone(@PhoneString VARCHAR(10))
RETURNS VARCHAR(20) AS
BEGIN
    DECLARE @outputFormat VARCHAR(20) = '(';
    DECLARE @Position INT = 1;
    DECLARE @LEN INT = LEN(@PhoneString)
    DECLARE @PhoneNum VARCHAR(1);
    WHILE @Position <= @LEN
    BEGIN
        SET @PhoneNum = SUBSTRING(@PhoneString, @Position, 1);
        SET @outputFormat += @PhoneNum;
        IF LEN(@outputFormat) = 4
            BEGIN
            SET @outputFormat += ') '
            END;
        IF LEN(@outputFormat) = 9
            BEGIN
            SET @outputFormat += ' - '
            END;
        SET @Position += 1
    END;
    RETURN @outputFormat
END;
```

test function

```
SELECT dbo.fn_FormatPhone('1234567890')
```

6. Create a stored procedure called dbo.usp_CustomerTotals instead of a view from Q2 in view exercise. Test the stored procedure.

```
CREATE OR ALTER PROC dbo.usp_CustomerTotals AS
SELECT CustomerID,
       YEAR(OrderDate) AS YearOrder ,
       MONTH(OrderDate) AS MonthOrder,
       COUNT(*) AS totalOrder,
       SUM(TotalDue) AS SumTotalDue_YearMonth
FROM Sales.SalesOrderHeader
GROUP BY CustomerID, YEAR(OrderDate), MONTH(OrderDate)
ORDER BY CustomerID;
GO
EXEC dbo.usp_CustomerTotals;
```

7. Modify the stored procedure in Q6 to include a parameter @customerID. Use the parameter in the WHERE clause of the query in the stored procedure. Test the stored procedure.

```
CREATE OR ALTER PROC dbo.usp_CustomerTotals @CustomerID INT AS
SELECT CustomerID,
       YEAR(OrderDate) AS YearOrder ,
       MONTH(OrderDate) AS MonthOrder,
       COUNT(*) AS totalOrder,
       SUM(TotalDue) AS SumTotalDue_YearMonth
FROM Sales.SalesOrderHeader
WHERE CustomerID = @CustomerID
GROUP BY CustomerID, YEAR(OrderDate), MONTH(OrderDate)
ORDER BY CustomerID;
GO
EXEC dbo.usp_CustomerTotals @CustomerID = 11000;
```

8. Create a stored procedure called dbo.usp_ProductSales that accepts a ProductID for a parameter and has an OUTPUT parameter that returns the total number sold for the product from the Sales.SalesOrderDetail table. Test stored procedure.

```
CREATE OR ALTER PROC dbo.usp_ProductSales @ProductID INT, @numberSold INT OUTPUT AS
SELECT @numberSold = SUM(OrderQty)
  FROM Sales.SalesOrderDetail
 WHERE ProductID = @ProductID
 GROUP BY ProductID
```

Test Procedure:

```
DECLARE @SoldNumbers INT;
EXEC dbo.usp_ProductSales 776, @SoldNumbers OUTPUT;
PRINT(@SoldNumbers)
```