T-SQL Exercises

LAB Stored Procedures

```
USE TSQL2018;
GO

-- Start fresh (if needed)
IF OBJECT_ID( 'Sales.GetTopCustomers' ) IS NOT NULL DROP PROCEDURE
Sales.GetTopCustomers
GO
```

```
-- Task 1
-- Execute the provided T-SQL code to return the custid, contactname and total sales for
the top ten customers based on their total sales value.

SELECT
        c.custid,
        c.contactname,
        SUM(o.val) AS salesvalue
FROM Sales.OrderValues AS o
JOIN Sales.Customers AS c
ON c.custid = o.custid
GROUP BY c.custid, c.contactname
ORDER BY salesvalue DESC
OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;
GO
```

-- Task 2

-- Create a stored procedure named Sales.GetTopCustomers. Use the query above for the procedure definition.

```
USE TSQL2018;
GO

CREATE PROCEDURE Sales.GetTopCustomers
AS
BEGIN
	SELECT
		c.custid,
		c.contactname,
		SUM(o.val) AS salesvalue
	FROM Sales.OrderValues AS o
	JOIN Sales.Customers AS c
	ON c.custid = o.custid
	GROUP BY c.custid, c.contactname
	ORDER BY salesvalue DESC
	OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY
END;
GO
```

-- Write a T-SQL statement to execute the created procedure.

```
EXEC Sales.GetTopCustomers;
GO
```

-- Task 3

-- Modify the Sales.GetTopCustomers stored procedure to include a parameter of type int named @orderyear.

-- Add a WHERE clause to the query in the function definition to filter by order year.

-- Use the YEAR() function and the orderdate column from the Orders table, check if it equals the @orderyear parameter

```
ALTER PROCEDURE Sales.GetTopCustomers @orderyear INT
AS
BEGIN
	SELECT
		c.custid,
		c.contactname,
		SUM(o.val) AS salesvalue
	FROM Sales.OrderValues AS o
	JOIN Sales.Customers AS c
	ON c.custid = o.custid
	WHERE YEAR(orderdate) = @orderyear
	GROUP BY c.custid, c.contactname
	ORDER BY salesvalue DESC
	OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY
END;
GO
```

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure for the year 2016.

```
EXEC Sales.GetTopCustomers @orderyear = 2016;
GO
```

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure for the year 2017.

```
EXEC Sales.GetTopCustomers @orderyear = 2017;
GO
```

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure without a parameter.

```
EXEC Sales.GetTopCustomers;
GO
```

-- Execute the T-SQL statement. What happened? What is the error message?

Error: Procedure or function 'GetTopCustomers' expects parameter '@orderyear', which was not supplied.

-- If an application was designed to use the exercise 1 version of the stored procedure, would the modification made to the stored procedure in this exercise impact the usability of that application?

Yes. Initially it was designed to return data without any filter. Now it requires the parameter which filters a data.

-- Task 4

-- Modify the Sales.GetTopCustomers stored procedure to give the @orderyear parameter a default NULL

```
ALTER PROCEDURE Sales.GetTopCustomers @orderyear INT = NULL
AS
BEGIN
```

```
    SELECT
        c.custid,
        c.contactname,
        SUM(o.val) AS salesvalue
    FROM Sales.OrderValues AS o
    JOIN Sales.Customers AS c
    ON c.custid = o.custid
    WHERE YEAR(orderdate) = @orderyear
    GROUP BY c.custid, c.contactname
    ORDER BY salesvalue DESC
    OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY
END;
GO
```

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure without a parameter.

```
EXEC Sales.GetTopCustomers;
GO
```

-- Task 4

-- Add the integer parameter @n to the Sales.GetTopCustomers stored procedure. Use this parameter to specify how many customers you want retrieved.

-- Use a default value of 10.

```
ALTER PROCEDURE Sales.GetTopCustomers @orderyear INT = NULL, @n int = 10
AS
BEGIN
    SELECT
```

```
        c.custid,
        c.contactname,
        SUM(o.val) AS salesvalue
    FROM Sales.OrderValues AS o
    JOIN Sales.Customers AS c
    ON c.custid = o.custid
    WHERE YEAR(orderdate) = @orderyear
    GROUP BY c.custid, c.contactname
    ORDER BY salesvalue DESC
    OFFSET 0 ROWS FETCH NEXT @n ROWS ONLY
END;
GO
```

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure without any parameters.

```
EXEC Sales.GetTopCustomers;
GO
```

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure for order year 2017 and five customers.

```
EXEC Sales.GetTopCustomers @orderyear = 2017, @n = 5;
GO
```

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure for the order year 20016.

```
EXEC Sales.GetTopCustomers @orderyear = 2016;
```

GO

-- Write an EXECUTE statement to invoke the Sales.GetTopCustomers stored procedure to retrieve 20 customers.

EXEC Sales.GetTopCustomers @n = 20;

GO

-- Do the applications using the stored procedure need to be changed because another parameter was added?

In order to add another parameter we need to alter the existing application. Only alter where the change is needed.