University Of Engineering and Technology, Lahore
Computer Engineering Department

Course Name: Database Systems	**Course Code: CS363L**
Assignment Type: Lab	**Dated: 18-04-2022**
Semester: 6th	**Session: 2019**
Lab/Project/Assignment #: Lab 12	**CLOs to be covered: CLO2**
Lab Title: Procedural Language, Creating Procedures, Functions Programming objects	**Teacher Name: Ms. Darakhshan**

Lab Evaluation:

CLO2	Construct DDL queries to manage relations, constraints, triggers and indexes.					
Levels (Marks)	**Level1**	**Level2**	**Level3**	**Level4**	**Level5**	**Level6**
Cognitive (5)						
					Total	**/5**

Rubrics for Current Lab:

Scale	Marks	Level	Rubric
Excellent	**5**	L1	Thoroughly completed homework questions with complete knowledge of each command + Rubric 4 requirement
Very Good	**4**	L2	Thoroughly completed lab tasks and wrote queries for IF/ELSE and CASE statements only for homework questions + Rubric 3 requirement
Good	**3**	L3	Only completed lab tasks + Rubric 2 requirements
Basic	**2**	L4	Attempted questions for Variables and Batch.
Barely Acceptable	**1**	L5	Read the entire manual and understood the requirements
Not Acceptable	**0**	L6	Did not attempt anything

LAB DETAILS:

Lab Goals/Objectives:

- Variables, if/else, loops
- Routines (User-defined functions, Stored procedures)
- Sequences
- Batch
- Error-handling

Theory/Relevant Material:

- Chapter 11 from TSQL book.

Reading:
Variables:

Temporarily store data for later use.

```
DECLARE @i AS INT;
SET @i = 10;

 DECLARE @i AS INT = 10;
```

For a given employee schema;

```
DECLARE @empname AS NVARCHAR(61);

SET @empname = (SELECT firstname + N' ' + lastname
                FROM HR.Employees
                WHERE empid = 3);

SELECT @empname AS empname;
```

This code returns the following output:

```
empname
----------
Judy Lew
```

Batch:

A batch of SQL statements is a group of two or more SQL statements or a single SQL statement that has the same effect as a group of two or more SQL statements. In some implementations, the entire batch statement is executed before any results are available.
First complete batch is parsed, if there is any syntax error, then whole batch is not submitted to SQL Server for execution.

```
-- Valid batch
PRINT 'First batch';
USE TSQLV4;
GO
-- Invalid batch
PRINT 'Second batch';
SELECT custid FROM Sales.Customers;
SELECT orderid FOM Sales.Orders;
GO
-- Valid batch
PRINT 'Third batch';
SELECT empid FROM HR.Employees;
```

A variable is local to the batch in which it is defined.

(More details present in TSQL chapter)

GO N:

GO command is usually used by SSMS, to denote the end of a batch. GO N supports how many times you want to execute the batch.

SET NOCOUNT ON;

It suppresses default output i.e. how many rows are affected, produced by DML statements for saving network bandwidth.

GENERATING SEQUENCES:

Read from following link:
https://www.sqlservertutorial.net/sql-server-basics/sql-server-sequence/
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-sequence-transact-sql?view=sql-server-ver15
https://stackoverflow.com/questions/44988294/how-does-the-cache-option-of-create-sequence-work

IDENTITY:

https://www.sqlservertutorial.net/sql-server-basics/sql-server-identity/
Although, you have not studied transactions yet but it is simply set of SQL statements that starts with **BEGIN** and ends at **END**. It will be discussed in coming labs.

PROGRAMMING:

IF/ELSE:

```
IF Boolean_expression
     { sql_statement | statement_block }
[ ELSE
     { sql_statement | statement_block } ]
```

SQL statement that follows an IF keyword and its condition is executed if the condition is satisfied: the Boolean expression returns TRUE. The optional ELSE keyword introduces another Transact-SQL statement that is executed when the IF condition is not satisfied: the Boolean expression returns FALSE.

```
IF DATENAME(weekday, GETDATE()) IN (N'Saturday', N'Sunday')
       SELECT 'Weekend';
ELSE
       SELECT 'Weekday';
```

An IF...ELSE construct can be used in batches, in stored procedures, and in ad hoc queries. When this construct is used in a stored procedure, it is frequently used to test for the existence of some parameter.
IF tests can be nested after another IF or following an ELSE. The limit to the number of nested levels depends on available memory.
Reference: MSDN

WHILE FLOW ELEMENT:

Syntax:

```
DECLARE @i AS INT = 1;
WHILE @i <= 10
BEGIN
  PRINT @i;
  SET @i = @i + 1;
END;
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Note: You can also **BREAK** to stop the loop and **CONTINUE** to skip the remaining code in loop (lines below CONTINUE keyword).

CASE:

Evaluates a list of conditions and returns one of multiple possible result expressions.
The CASE expression has two formats:

- The simple CASE expression compares an expression to a set of simple expressions to determine the result.
- The searched CASE expression evaluates a set of Boolean expressions to determine the result.

Both formats support an optional ELSE argument.
CASE can be used in any statement or clause that allows a valid expression. For example, you can use CASE in statements such as SELECT, UPDATE, DELETE and SET, and in clauses such as select_list, IN, WHERE, ORDER BY, and HAVING.

```
Simple CASE expression:
CASE input_expression
     WHEN when_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END
Searched CASE expression:
CASE
     WHEN Boolean_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END
```

ROUTINES:

Routines are programmable objects that encapsulates code to calculate a result or to execute activity. SQL Server supports three types of routines: user-defined functions, stored procedures, and triggers.

User-defined functions (UDFs):

- Encapsulate logic that calculates something based on inputs and return a result.

- UDFs: Scalar UDF (returns a single value), Table-valued (returns a table)
- UDFs are not allowed to apply any schema or data changes in the database

```
DROP FUNCTION IF EXISTS dbo.GetAge;
GO

CREATE FUNCTION dbo.GetAge
(
  @birthdate AS DATE,
  @eventdate AS DATE
)
RETURNS INT
AS
BEGIN
  RETURN
    DATEDIFF(year, @birthdate, @eventdate)
    - CASE WHEN 100 * MONTH(@eventdate) + DAY(@eventdate)
              < 100 * MONTH(@birthdate) + DAY(@birthdate)
           THEN 1 ELSE 0
      END;
END;
GO
```

Stored procedures:

- Encapsulate code
- It can have input and output parameters, they can return result sets of queries
- You can modify data through stored procedures.
- You can also apply schema changes through them
- You can change the implementation of a stored procedure using ALTER PROC command.
- Provides security. For example, you may allow someone to delete X Customer using procedure but you do not want to allow them to delete rows from table.
- You can check validity of data
- Incorporate error-handling and resolving them silently.
- Performance benefits (Reduction in network traffic. Less data to be sent to procedures)

Syntax:

```
DROP PROC IF EXISTS Sales.GetCustomerOrders;
GO

CREATE PROC Sales.GetCustomerOrders
  @custid   AS INT,
  @fromdate AS DATETIME = '19000101',
  @todate   AS DATETIME = '99991231',
  @numrows  AS INT OUTPUT
AS
SET NOCOUNT ON;

SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE custid = @custid
  AND orderdate >= @fromdate
  AND orderdate < @todate;

SET @numrows = @@rowcount;
GO
```

Execute:

```
DECLARE @rc AS INT;

EXEC Sales.GetCustomerOrders
  @custid   = 1,
  @fromdate = '20150101',
  @todate   = '20160101',
  @numrows  = @rc OUTPUT;

SELECT @rc AS numrows;
```

Output:

```
orderid     custid      empid       orderdate
----------- ----------- ----------- -----------
10643       1           6           2015-08-25
10692       1           4           2015-10-03
10702       1           4           2015-10-13

numrows
-----------
3
```

Error handling:
For error handling, you can place your set of commands or code in a TRY block (between BEGIN TRY and END TRY). You will write all your error-handling code in CATCH block (between BEGIN CATCH and END CATCH). If TRY block has no error, then CATCH block is simply skipped or not executed. If the TRY block has an error, control is passed to the corresponding CACTH block.

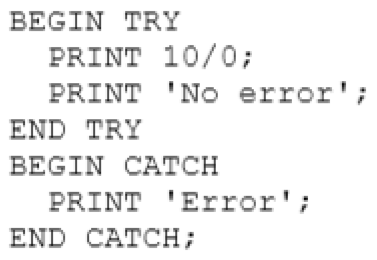
Syntax:

```
BEGIN TRY
  PRINT 10/2;
  PRINT 'No error';
END TRY
BEGIN CATCH
  PRINT 'Error';
END CATCH;
```

Output:

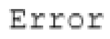
```
5
No error
```

Code2:



```
BEGIN TRY
  PRINT 10/0;
  PRINT 'No error';
END TRY
BEGIN CATCH
  PRINT 'Error';
END CATCH;
```

Output:



```
Error
```

(See more details in TSQL book)

Lab Tasks:

Variables:

Use Northwind schema unless you are asked to construct your own
tables.

1. Store ContactTitle of Customer with Id "BERGS" in a variable using sub-query and then display it.
2. Store name, title and Address of Customer with Id "BERGS" in a variable without sub-query and display it.
3. Use SET and SELECT statements to display the variable that has the quantity of Product for Order ID 10248.

Explain the behavior of their outputs.

Batch

1. Create a Procedure that accepts ID of customer as input and displays the details of that customer. Conduct the code properly by dropping and creating the Procedure in batches.

GON, SET NCOUNT N, GENERATING SEQUENCES, IDENTITY:

1. Create a table called FirstMultiplesOf10 containing one integer column. Use sequence and GO N to fill the table with the first 10 multiples of 10. Display the table.
2. Create another table SecondMultiplesof10 that contains the 10- 20 multiples of 10 using the same sequence that was generated in the previous question. Display the table.
3. Create a table with ID and Productname using IDENTITY that is incremented by 3 for each row.

Homework Questions:

IF/ELSE:

1. Create 10Multiplesof10 table using While loop instead of Go N.
2. Display the names of the first 50 Products that have available Stock (Not 0 in quantity in Stock) using While.

CASE:

1. Use Case to display "Subordinate" for the employees that report to someone and "Superior" for those who report to no other employee.

Stored Procedures:

1. Create a function that takes a number as input and calculates and displays its first 10 multiples. Use the function to calculate tables of 1, 2, 3, 4, 5 (Do not write the same query 5 times, find a more efficient way).
2. Create both, a Procedure and a function that takes order ID as input and calculates the total bill for each order of that ID and then displays the bills for each order.

Error Handling:

1. Use error handling to catch the error from the third question in the variables section using SET i.e. Use SET statement to display the variable that has the quantity of Product for Order ID 10248.
2. Define a procedure "ErrorInfo" that displays all error information. Upon catching the error, the ErrorInfo procedure should be called.

Submission Instructions:

Assemble all your queries file in one zip file or you can write your queries in one file and name it as DBLab12_2019_CE_X.sql add supporting SQL scripts of your homework and submit on google classroom by Sunday, 24th April, 2022 9 P.M