|  |
| --- |
| **National University of Computer and Emerging Sciences** |
| Lab Manual 2 – Task 2  “SQL Review (Functions, Procedures and Triggers)” |
|  |
| Data Warehousing and Data Mining |
|  |
|  |
|  |

|  |  |
| --- | --- |
| Course Instructor | Ishaq Raza |
| Lab Instructor(s) | Farhan Azhar Nouman Ali |
| Section | CS |
| Semester | Fall 2019 |

|  |
| --- |
|  |

Department of Computer Science

FAST-NU, Lahore, Pakistan

Table of Contents

[1. Task Distribution 3](#_Toc365622545)

[2. Objective 4](#_Toc365622546)

[3. Required material 4](#_Toc365622547)

[4. Functions 4](#_Toc365622548)

[5. Writing Stored Procedures 7](#_Toc365622550)

[6. Triggers 8](#_Toc365622553)

[7. Exercise 1 9](#_Toc365622555)

## Task Distribution

*Total (170 minutes)*

* + **Introduction to Functions, Triggers and Stored Procedures** *(90 minutes)*
  + **Practice Exercise** *(80 minutes)*

## Objective

The purpose of this lab activity is to review and revise SQL server 2012 concepts that were previously covered in database systems course.

At the end of this lab you will be able to

* + Write Stored Procedures
  + Write Functions
  + Write Triggers

## Required material

* Script files [**scripts.zip**](scripts.zip)
* SQL server installation guide [SQL server 2012 installation guide.pptx](SQL%20server%202008%20installation%20guide.pptx)

## Functions

|  |  |
| --- | --- |
| Task 1: Writing Functions | **Estimated completion time (min):** 30 |
| **Scalar Function**  Have 1 input parameter and returns 1 output parameter.  CREATEFUNCTION dbo.getname (@EID int)  RETURNSVARCHAR(50)  AS  BEGIN  DECLARE @NameRet VARCHAR(50);  SELECT @NameRet =Name  FROM Employee b  WHERE b.EmployeeID= @EID    IF(@NameRet ISNULL)  SET @NameRet='Name not defined'    RETURN @NameRet  END;  Declare @ENAMEVARCHAR(50)  select @ENAME=dbo.getname(1);  print @ENAME  **Table valued function**  Have multiple attributes returned in the form of table given a specific input.  **Exmaple :**  CREATEFUNCTION [dbo].[tablefunc1] (@Eid int)  RETURNS @RetTable TABLE  (  EmployeeID INTNOTNULL,  AddressID VARCHAR(50)NULL,  AddressLine1 VARCHAR(50)NULL,  City VARCHAR(50)NULL  )  AS  BEGIN  DECLARE @EmpID INT, @AddressID INT, @AddressLine1 VARCHAR(50), @City VARCHAR(50)    SELECT @EmpID =EmployeeID,  @AddressID = AddressID,  @AddressLine1= AddressLine1,  @City=City  FROM EmployeeAddress A  WHERE A.EmployeeID = @Eid    IF @Eid ISNOTNULL  BEGIN  INSERT @RetTable  SELECT @EmpID, @AddressID, @AddressLine1, @City  END;  RETURN;  END;  SELECT\*FROM dbo.[tablefunc1](3);  **Exmaple :**  CREATEFUNCTION [dbo].[tablefuncJoin] (@Eid int)  RETURNS @RetTable TABLE  (  EmployeeID INTNOTNULL,  NameVARCHAR(50)NULL,  AddressID INTNULL,  AddressLine1 VARCHAR(50)NULL,  City VARCHAR(50)NULL  )  AS  BEGIN  DECLARE @EmpID INT, @NAME VARCHAR(50), @AddressID INT, @AddressLine1 VARCHAR(50), @City VARCHAR(50)    SELECT @EmpID =EmployeeID,  @Name=NAME,  @AddressID = AddressID,  @AddressLine1= AddressLine1,  @City=City  FROM  (  SELECT A.EmployeeID, A.NAME, B.AddressID, B.AddressLine1, B.City  FROM Employee A  INNERJOIN EmployeeAddress B  ON A.EmployeeID= B.EmployeeID  ) C  WHERE C.EmployeeID = @Eid    IF @Eid ISNOTNULL  BEGIN  INSERT @RetTable  SELECT @EmpID, @NAME, @AddressID, @AddressLine1, @City  END;  RETURN;  END;  SELECT\*FROM dbo.[tablefuncJoin](3); | |

## Writing Stored Procedures

|  |  |
| --- | --- |
| Task 1: Creating and Writing Stored Procedures | **Estimated completion time (min):** 30 |
| CREATEPROCEDURE dbo.getDetail  @Name VARCHAR(50)  AS  SELECT A.EmployeeID, A.NAME, B.AddressID , B.City  FROM Employee A  INNERJOIN EmployeeAddress B  ON A.EmployeeID= B.EmployeeID  WHERE A.Name = @Name | |
| Executing Stored Procedure | **2 minutes** |
| EXEC dbo.getDetail  @Name ='john' | |

## 

## Triggers

|  |  |
| --- | --- |
| Task 1: Writing Triggers | **Estimated completion time (min):** 30 |
| CREATE TRIGGER SampleTrigger  ON DBO.TriggerEmployee  FOR UPDATE  AS  INSERT INTO TriggerEmployee(EmployeeID, NAME, ContactID, ManagerID, Gender)  VALUES(6, 'TriggerName' , 1108, 1, 'M' )  **Invoking Trigger**  -- SELECT \* FROM TriggerEmployee -- NO CHANGE IN THE TABLE    UPDATE TriggerEmployee  SET NAME = 'changed with trigger'  WHERE EmployeeID= 5    SELECT \* FROM TriggerEmployee | |

## Exercise 1

**Functions**

1. Create a Scalar function that takes Employee ID and retrieves the project that he has worked on for Department Number 2.
2. Create a tabular function that returns the Employee table details for all the Employees whose pay Rate has been changed more than once during their Employment history.

**Stored Procedures**

1. Create a Stored Procedure which returns the Employee Pay History for the Employees who live in the city of ‘*Calgary’* and their *Pay Frequency*>1
2. Create a procedure that returns the Employee table details for all the Employees whose pay Rate has been changed more than once during their Employment history (Convert tabular function specified in Question No 2 into Stored Procedure)

**Triggers**

1. Create a trigger to update the records of a table called Project whenever the insert operation is performed against this particular table.
2. Create a trigger that disallow any delete operation to be performed against EmployeePayHistory and print any appropriate message against it.