**IST 278 Test 2 Hands-On Part**

Name: Ramon Rodriguez Date: 7/2/2024

# **Test Instructions**

1. Type your name and the date in the spaces provided.
2. Use the SQL Server Management Studio and the IST278EagleCorp database.
3. Complete the **3** exercises in this document. Make sure to scroll down. This document has several pages to it.
4. Upload and submit a completed copy of this test document before the due date.

**Exercise 1 (a table valued function exercise):**

Use the IST278EagleCorp13-1 database and create a table-valued function named fnEmployeeBirthDaysXX where the XX are your initials. This function requires one parameters of data type INT. Do not validate the input parameter. Have the function return a result set that consists of the LastName, FirstName, and BirthDate for each active employee (employee that has not been released) that has a birthdate in the month specified by the parameter value passed to the function. For example:

If a 0 is passed to the function, the function is to return an empty result set.

if a 1 is passed to the function, it is to return the LastName, FirstName, and BirthDate of each active employee born in January.

if a 2 is passed to the function, it is to return the LastName, FirstName, and BirthDate of each active employee born in February.

**--- Paste below this line the code you wrote to create the fnEmployeeBirthDaysXX function**

Paste here

USE [IST278EagleCorp13-1];

GO

DROP FUNCTION IF EXISTS dbo.fnEmployeeBirthDaysRR;

GO

CREATE FUNCTION dbo.fnEmployeeBirthDaysRR (@Month INT)

RETURNS TABLE

AS

RETURN

(

SELECT LastName, FirstName, BirthDate FROM Employee

WHERE MONTH(BirthDate) = @Month AND ReleaseDate IS NULL

);

GO

Test the Function you created for exercise one by coding and executing a select statement that invokes the function from within a SELECT statement to return the data for active employees born in July.

**--- Paste below this line the Select statement you wrote to test the fnEmployeeBirthDayXX function**

Paste here

SELECT \* FROM dbo.fnEmployeeBirthDaysRR(7);

**---** **Paste below this line the run results from executing the select you wrote to test the fnEmployeeBirthDayXX function--**

Paste here

A screenshot of a computer

Description automatically generated

**Exercise 2 (a Trigger exercise):**

2a. Use the IST278EagleCorp database and create a new table named SalaryWageChangeTrackingXX where the XX are your initials. Create this table with the following columns:

SalaryWageChangeNo INT NOT NULL IDENTITY PRIMARY KEY,

EmployeeID VARCHAR(10),

LastName VARCHAR(20),

FirstName VARCHAR(15),

DateandTimeOfChange SMALLDATETIME,

PriorSalaryWage DECIMAL(9,2),

NewSalaryWage DECIMAL(9,2)

**--- Paste below this line the create table code you wrote and executed for exercise 2a. --**

Paste here

CREATE TABLE SalaryWageChangeTrackingRR (

SalaryWageChangeNo INT NOT NULL IDENTITY PRIMARY KEY,

EmployeeID VARCHAR(10),

LastName VARCHAR(20),

FirstName VARCHAR(15),

DateandTimeOfChange SMALLDATETIME,

PriorSalaryWage DECIMAL(9,2),

NewSalaryWage DECIMAL(9,2)

);

GO

2b. Use the IST278EagleCorp database and create a trigger named TRSalaryWageChangedXX where the XX are your initials. Code this trigger so that it automatically inserts a record into the SalaryWageChangeTrackingXX table every time an employee’s SalaryWage value is updated. The trigger is to populate the SalaryWageChangeTrackingXX table’s columns as follows:

Attribute Value to assign to attribute

EmployeeID EmployeeID associated with the updated Employee record

LastName LastName associated with the updated Employee record

FirstName FirstName associated with the updated Employee record

DateandTimeOfChange Date and time that the update occurred

PriorSalaryWage SalaryWage value the Employee had before the update

NewSalaryWage SalaryWage value the Employee has after the update

This trigger should fire every time an Employee record is updated, but it should only write a record to the SalaryWageChangeTrackingXX table when the update resulted in a change to the SalaryWage value.

**---** **Paste below this line the Trigger you wrote and executed for exercise 2b –**

Paste here

CREATE TRIGGER TRSalaryWageChangedRR

ON Employee

AFTER UPDATE

AS

BEGIN

INSERT INTO SalaryWageChangeTrackingRR (

EmployeeID, LastName, FirstName, DateandTimeOfChange, PriorSalaryWage, NewSalaryWage

)

SELECT d.EmployeeID, d.LastName, d.FirstName, GETDATE() AS DateandTimeOfChange, d.SalaryWage AS PriorSalaryWage, i.SalaryWage AS NewSalaryWage FROM deleted d

INNER JOIN inserted i ON d.EmployeeID = i.EmployeeID

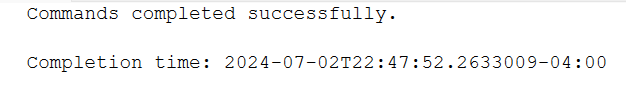
WHERE d.SalaryWage <> i.SalaryWage;

END

GO

**---** **Paste below this line the completion status messages from executing the exercise 2b trigger code-**

Paste here



2c. Use the IST278EagleCorp database and Insert into the Employee Table a new record with the following attribute values:

‘**00200221XX**’ for the EmployeeID (where the XX are your initials)

**Your real first name for the FirstName**

**Your real last name for the LastName**

**‘Software Tester’ for the JobTitle**

**44500.00 for the SalaryWage**

**Null for all other attributes**

An example insert that you can modify, and use is provided below:

INSERT INTO dbo.Employee (EmployeeID, LastName, FirstName, JobTitle, SalaryWage)

VALUES

('00200221XX', 'YourLastName', 'YourFirstName', 'Software Tester', 44500.00)

**--- Paste below this line the Insert you wrote and executed for exercise 2c –**

Paste here

INSERT INTO dbo.Employee (EmployeeID, LastName, FirstName, JobTitle, SalaryWage)

VALUES ('00200221RR', 'Rodriguez', 'Ramon', 'Software Tester', 44500.00);

**---** **Paste below this line the completion status messages from executing your insert table code-**

Paste here

A number and text on a white background

Description automatically generated

2d. Use the IST278EagleCorp database and update the SalaryWage value for the Employee record that you inserted in the last exercise ( the one that has EmployeeID **00200221XX**) to increase the SalaryWage by 10000. An example update statement that you can modify, and use is provided below (don’t forget to change the XX to your initials):

UPDATE Employee

SET SalaryWage = SalaryWage + 10000

WHERE EmployeeID = '00200221XX';

**--- Paste below this line the Update you wrote and executed for exercise 2d –**

Paste here

UPDATE Employee

SET SalaryWage = SalaryWage + 10000

WHERE EmployeeID = '00200221RR';

**---** **Paste below this line the completion status messages from executing the exercise 2d update statement-**

Paste here

A number and numbers on a white background

Description automatically generated

2e. Use the IST278EagleCorp database and code a select \* from the SalaryWageChangeTrackingXX table you created in this lab. When you execute this select \* from SalaryWageChangeTrackingXX you should see a record in the result set. A record should have been inserted into your SalaryWageChangeTrackingXX table when you ran the update because it should have caused the trigger you created in exercise 2b to fire.

**--- Paste below this line the select \* statement you wrote you wrote for exercise 2e –**

Paste here

SELECT \* FROM SalaryWageChangeTrackingRR;

**---** **Paste below this line the run results from executing the exercise 2e select \* --**

Paste here

A screenshot of a computer

Description automatically generated

**Exercise 3 (a Transaction exercise):**

In this exercise you will code and test a script that uses transaction logic to do an explicit BEGIN TRAN, COMMIT TRAN, and ROLLBACK TRAN.

Code your script as specified below doing the steps as specified in the order specified:

3a.Code a USE statement and a GO to make sure the script only runs against the IST278EagleCorp database.

3b. Code a Print statement that prints the following message (use your real name in place of the XXXXXXXXXXX XXXXXXXXXX):

Test 2 Transaction Exercise Script Coded By: XXXXXXXXXX XXXXXXXXXXXX.

3c. Code a BEGIN TRAN statement

3d. Code a DELETE SalaryWageChangeTrackingXX WHERE EmployeeID = '00200221XX'statement (Don’t forget to change the XX’s to your initials. This exercise is making use of a table you created and populated while working an earlier exercise in this test.)

3e. Code if/else logic that checks the number of rows affected by the previous statement. If more than one row was affected, roll back the transaction and print a message that states 'Deletions Rolled Back – More Rows than Expected'. If one or fewer rows were affected, commit the transaction and print a message that states 'Deletions Committed'.

3f. Code a comment after the above logic that states it is the end of your script and include your name in it. An example follows:

-- END OF TEST2 SCRIPT Coded by Bob Whaite

**--- Paste below this line the Script you wrote for exercise 3 --**

Paste here

PRINT 'Test 2 Transaction Exercise Script Coded By: Ramon Rodriguez';

GO

BEGIN TRAN;

GO

DELETE FROM SalaryWageChangeTrackingRR WHERE EmployeeID = '00200221RR';

GO

DECLARE @RowsAffected INT;

SET @RowsAffected = @@ROWCOUNT;

IF @RowsAffected > 1

BEGIN

ROLLBACK TRAN;

PRINT 'Deletions Rolled Back – More Rows than Expected.';

END

ELSE

BEGIN

COMMIT TRAN;

PRINT 'Deletions Committed.';

END

GO

-- END OF TEST2 SCRIPT Coded by Ramon Rodriguez

**---** **Paste below this line the completion status messages from executing the exercise 3 script-**

Paste Here

A screen shot of a computer

Description automatically generated

**---** **Type below an explanation of the completion status messages you pasted above-**

Paste Here

The status message indicates that the script successfully deleted the row and the transaction was committed and printed a message saying that the script was coded by Ramon Rodriguez.