Course: IST-659

Name: Sharat Sripada

Homework #8

Date Submitted: 6/4/2020

Topic: Lab08 – Database Programming

## Part 1 – Introducing Functions, Views, and Stored

## Procedures

### Functions

A screenshot of a social media post

Description automatically generated

**Fig: Is null usage**

**A screenshot of a social media post

Description automatically generated**

**Fig: My first-function**

### Functions That Are More... well... Functional

A screenshot of a social media post

Description automatically generated

**Fig: VidCastCount function**

**In your own words, in your answers document, describe what lines 49 through 53 above actually do. Also, how is it that this code knows that the vc\_User record with vc\_UserID = 20 has 22 vc\_VidCast records?**

The SELECT statement above essentially calls the function ‘VidCastCount’ and provides the vc\_UserID from table vc\_User as an argument/parameter. The vc\_UserID is scoped then to the local variable userid, which is used in the SELECT statement within it:

SELECT @returnvalue = COUNT(VidCastTitle) from vc\_VidCast

WHERE vc\_UserID=@userid

Finally, the data is sorted in descending order & then the TOP 10 records are shown.

We can code a function that accepts the tag text as a parameter and looks up the vc\_TagID for the vc\_Tag record for that TagText.

A screenshot of a social media post

Description automatically generated

**Fig: TagIDlookup function**

**In your own words, in your answers document, describe what lines 75 and 76 above actually do. Also, when line 76 executed, we received a NULL from SQL Server. How come?**

First, the function TagIDlookup is written to take the TagText as input and return the corresponding vc\_TagID as seen below:

SELECT @returnvalue = vc\_TagID from vc\_Tag

WHERE TagText=@tagtext

Value vc\_TagID is then returned as a result. With respect to the NULL, when examining the vc\_Tag table we see TagText = ‘Tunes’ is NOT PRESENT.

### Views

A screenshot of a social media post

Description automatically generated

**Fig: MostProlificUsers View**

**In your own words, in your answers document, describe what lines 79 through 87 above are doing.**

We are creating a View and placing within a SELECT statement that calls a function (here dbo.VidCastCount) which was previously defined.

## Stored Procedures

Stored procedures vs functions

Stored procedures are like functions in that they perform operations based on provided parameter values, but they are different in that they can perform more complex database activities. For instance, whereas a user-defined function can make no changes to the database in any way, a stored procedure can be used to encapsulate and abstract statements such as INSERT, UPDATE, and DELETE.

A screenshot of a social media post

Description automatically generated

**Fig: Update email using Stored procedure**

**In your own words, in your answers document, describe what lines 91 through 104 above are doing.**

The stored procedure vc\_ChangeUserEmail takes two parameters username and newemail & within it implements an update carefully using the WHERE clause.

## @@Identity

A screenshot of a cell phone

Description automatically generated

**Fig: @@Identity example**

**A screenshot of a social media post

Description automatically generated**

**Your UserLoginTimestamp value will be different than the one shown. On your answers doc, explain why this is.**

The UserLoginTimestamp has data-type datetime. This means, each time a new userlo

**On your answers doc, also identify one way we could simplify the code in the stored procedure above. (Hint: Look back at how we did a lookup with the vc\_Tag table)**

## Part 2 – Putting all together

### Coding your own Functions

Custom Function-1

The code below is the beginning of a function intended to retrieve a vc\_UserID from the vc\_User table given a specified @userName. Complete the code to assign the correct vc\_UserID to @returnValue

A screenshot of a cell phone

Description automatically generated

**Fig: Function to return vc\_UserID given UserName**

Custom Function-2

Author a function called dbo.vc\_TagVidCastCount that calculates the count of vc\_VidCastIDs for a given vc\_TagID. Consult the diagram at the end of this document as a reference for the tables involved.

**A screenshot of a social media post

Description automatically generated**

**Fig: Function to return vc\_UserID given UserName**

Custom Function-3

Code a function called vc\_VidCastDuration that SUMs the total number of minutes of actual duration for VidCasts with a Finished status given a vc\_UserID as a parameter. This function should return the SUM as an int.

A screenshot of a social media post

Description automatically generated

**Fig: Function to return SUM of total minutes for Vidcasts given a UserID**

## Coding your own Views

### Custom View-1

To fetch details from Vidcast utilizing function vc\_TagVidCastCount

A screenshot of a social media post

Description automatically generated

**Fig: View to call function vc\_TagVidCastCount**

### Custom View-2

Alter the view called vc\_MostProlificUsers, adding a column called TotalMinutes that calls the vc\_VidCastDuration function we created earlier in part 2.

A screenshot of a social media post

Description automatically generated

**Fig: Alter View vc\_MostProlificUsers**

## Coding Your Own Stored Procedures

### Custom Stored Procedure-1

### A screenshot of a social media post Description automatically generated

**Fig: Procedure to insert a value into vc\_Tag**

### Custom Stored Procedure-2

**A screenshot of a social media post

Description automatically generated**

**Fig: Modify vc\_Vidcast to mark Status as Finished and update timestamp**

## Appendix – SQL Queries Lab08

Kept losing connection to the remote-desktop server and hence lost some lines of code.

-- Declare a variable (we’ll talk about variables in a minute)

declare @isThisNull varchar(30) -- Starts out as NULL

SELECT @isThisNull, ISNULL(@isThisNull, 'Yep, it is null') -- See?

-- Set the variable to something other than NULL

SET @isThisNull = 'Nope. It is not NULL'

SELECT @isThisNull, ISNULL(@isThisNull, 'Yep, it is null')

-- My first user-defined function in SQL

CREATE FUNCTION dbo.AddTwoInts(@firstnumber int, @secondnumber int)

RETURNS int AS

BEGIN

DECLARE @returnvalue int

SET @returnvalue = @firstnumber + @secondnumber

RETURN @returnvalue

END

GO

SELECT dbo.AddTwoInts(5,10)

--Our VidCast service is interested in the number of VidCasts created by VidCast users

-- First, let's write the select statement

SELECT COUNT(vc\_VidCast.VidCastTitle) from vc\_VidCast

CREATE FUNCTION dbo.VidCastCount(@userid int)

RETURNS int AS

BEGIN

DECLARE @returnvalue int

SELECT @returnvalue = COUNT(VidCastTitle) from vc\_VidCast

WHERE vc\_UserID=@userid

RETURN @returnvalue

END

GO

-- Example

SELECT dbo.VidCastCount(34)

SELECT TOP 10

\*

, dbo.VidCastCount(vc\_UserID) as VidCastCount

FROM vc\_User

ORDER BY VidCastCount DESC

SELECT

\*

, dbo.VidCastCount(vc\_UserID) as VidCastCount

FROM vc\_User

ORDER BY VidCastCount DESC

-- We can code a function that accepts the tag text as a parameter and looks up the vc\_TagID for the vc\_Tag record for that TagText.

CREATE FUNCTION dbo.TagIdLookup(@tagtext varchar(20))

RETURNS int AS

BEGIN

DECLARE @returnvalue int

SELECT @returnvalue = vc\_TagID from vc\_Tag

WHERE TagText=@tagtext

RETURN @returnvalue

END

GO

SELECT dbo.TagIdLookup('Music')

SELECT dbo.TagIdLookup('Tunes')

-- Part-2: Putting it all together

-- Coding your own functions

-- Function to get UserID from table given a username

CREATE FUNCTION dbo.vc\_UserIDLookup(@userName varchar(20))

RETURNS int AS

BEGIN

DECLARE @returnValue int

SELECT @returnValue = vc\_UserID FROM vc\_User

WHERE UserName = @userName

RETURN @returnValue

END

GO

SELECT 'Trying the vc\_UserIDLookup function.', dbo.vc\_UserIDLookup('tardy')

-- LOST SOME WORK HERE...SOB! SOB! --

-- Coding your own Views

-- User defined View-1

CREATE VIEW vc\_TagReport AS

SELECT

vc\_Tag.TagText

, dbo.vc\_TagVidCastCount(vc\_Tag.vc\_TagID) as VidCasts

FROM vc\_Tag

GO

SELECT \* FROM vc\_TagReport

ORDER BY VidCasts DESC

-- Alter VIEW vc\_MostProlificUsers

SELECT \* FROM vc\_MostProlificUsers

ALTER VIEW vc\_MostProlificUsers AS

SELECT TOP 10

\*

, dbo.VidCastCount(vc\_UserID) as VidCastCount

, dbo.vc\_VidCastDuration(vc\_UserID) as TotalMinutes

FROM vc\_User

ORDER BY VidCastCount DESC

GO

SELECT UserName, VidCastCount, TotalMinutes FROM vc\_MostProlificUsers

-- Coding Your Own Stored Procedures

-- Inserting a value into the vc\_Tag table

CREATE PROCEDURE vc\_AddTag(@tagText varchar(20), @description varchar(100)=NULL) AS

BEGIN

INSERT INTO vc\_Tag (TagText, TagDescription)

VALUES (@tagText, @description)

RETURN @@identity

END

GO

DECLARE @newTagID int

EXEC @newTagID = vc\_AddTag 'SQL', 'Finally, a SQL Tag'

SELECT \* FROM vc\_Tag where vc\_TagID = @newTagID

--

CREATE PROCEDURE vc\_FinishVidCast(@vidcastid int) AS

BEGIN

DECLARE @currtime datetime

SET @currtime = GetDate()

UPDATE vc\_VidCast SET EndDateTime=@currtime, vc\_StatusID=3

WHERE vc\_VidCastID = @vidcastid

END

GO

DECLARE @newVC int

INSERT INTO vc\_VidCast

(VidCastTitle, StartDateTime, ScheduleDurationMinutes, vc\_UserID,

vc\_StatusID)

VALUES (

'Finally done with sprocs'

, DATEADD(n, -45, GETDATE())

, 45

, (SELECT vc\_UserID FROM vc\_User WHERE UserName = 'tardy')

, (SELECT vc\_StatusID FROM vc\_Status WHERE StatusText='Started')

)

SET @newVC = @@identity

SELECT \* FROM vc\_VidCast WHERE vc\_VidCastID = @newVC

EXEC vc\_FinishVidCast @newVC

SELECT \* FROM vc\_VidCast WHERE vc\_VidCastID = @newVC