Nick Videtti

IST-659 Week 8 Lab Assignment

**PART 1:**

***SELECT***

***TOP 10 \*,***

***dbo.vc\_VidCastCount(vc\_UserID) VidCastCount***

***FROM***

***vc\_User***

***ORDER BY***

***VidCastCount DESC***

This SELECT statement selects the top 10 Users from the vc\_User table based on the number of VidCasts they have made. It also orders them by number of VidCasts made, in descending order. The reason that this code knows that the vc\_User record with vc\_UserID = 20 has 22 vc\_VidCast records is because we pass the vc\_UserID into the vc\_VidCastCount function that we created, which takes a user a returns that number of VidCasts that user has created.

***SELECT dbo.vc\_TagIDLookup('Music')***

***SELECT dbo.vc\_TagIDLookup('Tunes')***

These statements above call the dbo.vc\_TagIDLookup function that we have just created. The first line passes ‘Music’ into the function, returning the vc\_TagID that matches the TagText that says ‘Music’. The second line passes ‘Tunes’ into the function, returning the vc\_TagID that matches the TagText that says ‘Tunes’. We received a NULL when executing the second line because there is not a record in the vc\_Tag table that has a TagText of ‘Tunes’, meaning that there is not a vc\_TagID corresponding to a tag whose TagText is ‘Tunes.

***CREATE VIEW vc\_MostProlificUsers AS***

***SELECT***

***TOP 10 \*,***

***dbo.vc\_VidCastCount(vc\_UserID) AS VidCastCount***

***FROM***

***vc\_User***

***ORDER BY***

***VidCastCount DESC***

***GO***

This code above is creating a view called vc\_MostProlificUsers that contains the top 10 records from the vc\_User table based on the output from the dbo.VidCastCount function that we created earlier in the lab. The records are also sorted in descending order by the value returned from the dbo.VidCastCount function.

***CREATE PROCEDURE vc\_ChangeUserEmail(@userName varchar(20), @newEmail varchar(50))***

***AS***

***BEGIN***

***UPDATE vc\_User SET EmailAddress = @newEmail***

***WHERE UserName = @userName***

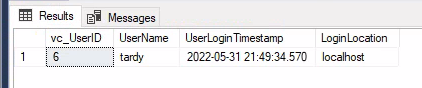
***END***

***GO***

***EXEC vc\_ChangeUserEmail 'tardy', 'kmstudent@syr.edu'***

***SELECT \* FROM vc\_User WHERE UserName = 'tardy'***

The code above creates a stored procedure called vc\_ChangeUserEmail that takes two parameters, the first being the vc\_Users UserName and the second being the email address. This procedure changes the passed-in vc\_User’s email address to the one that was passed in. Then, we execute the procedure by setting the vc\_User’s email address with the UserName ‘tardy’ to ‘[kmstudent@syr.edu](mailto:kmstudent@syr.edu)’. The final line queries the vc\_User table for only the vc\_User whose UserName is ‘tardy’ and shows us that the email address is indeed ‘[kmstudent@syr.edu](mailto:kmstudent@syr.edu)’.



The reason that this UserLogin Timestamp is different from the lab document is that this column is set to be the current system timestamp.

***DECLARE @addedValue INT***

***EXEC @addedValue = vc\_AddUserLogin 'tardy', 'localhost'***

***SELECT***

***vc\_User.vc\_UserID,***

***vc\_User.UserName,***

***vc\_UserLogin.UserLoginTimestamp,***

***vc\_UserLogin.LoginLocation***

***FROM***

***vc\_User***

***JOIN***

***vc\_UserLogin***

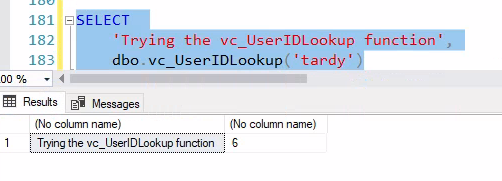
***ON vc\_User.vc\_UserID = vc\_UserLogin.vc\_UserID***

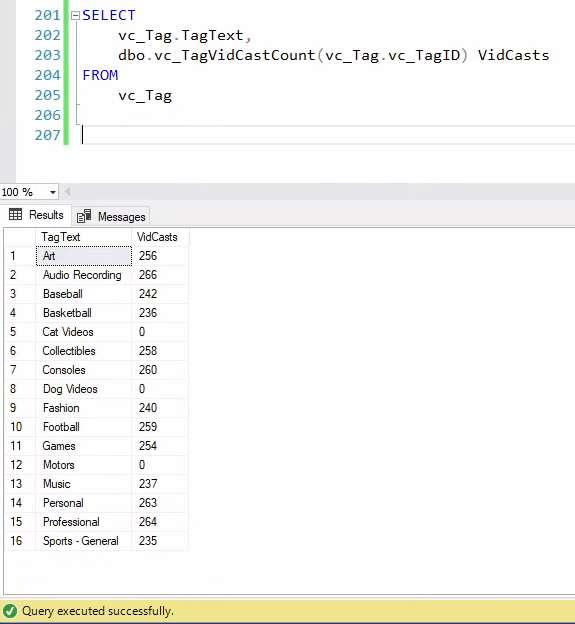
***WHERE***

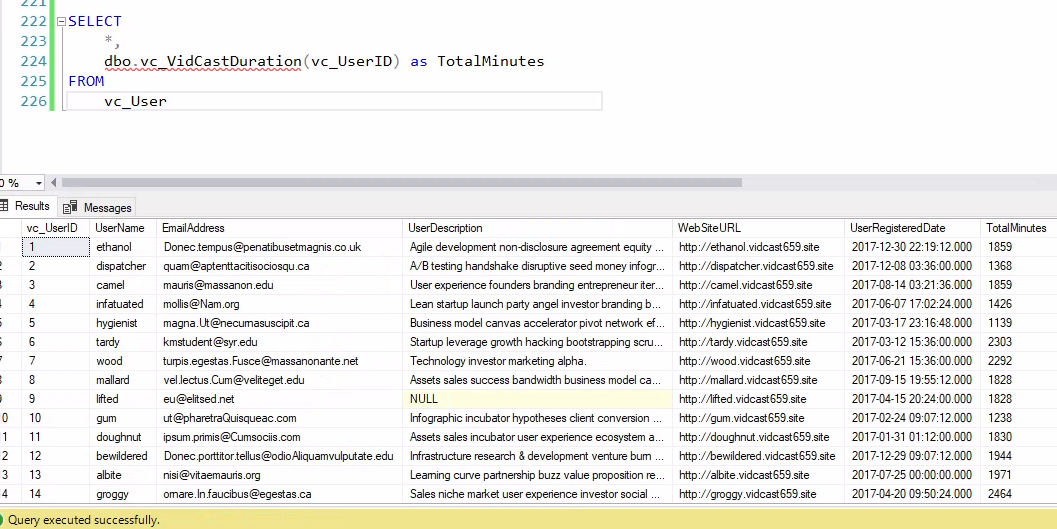
***vc\_UserLoginID = @addedValue***

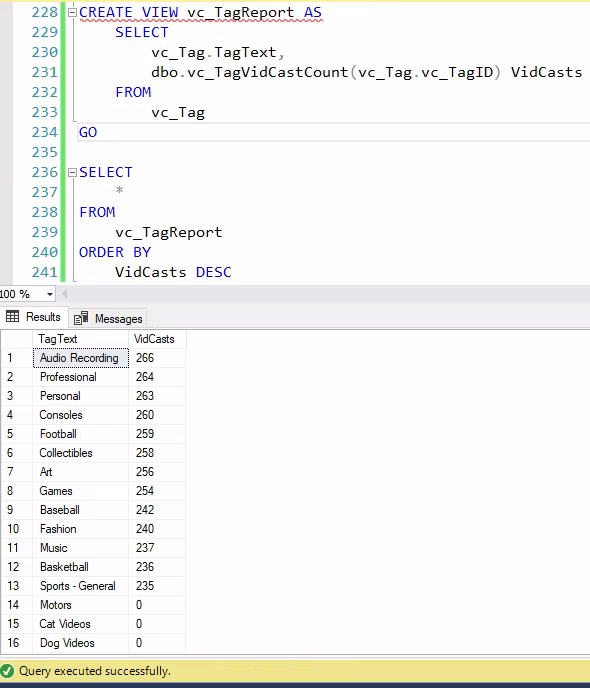
This code could be simplified by using the SCOPE\_IDENTITY() function rather than declaring and using the addedValue variable

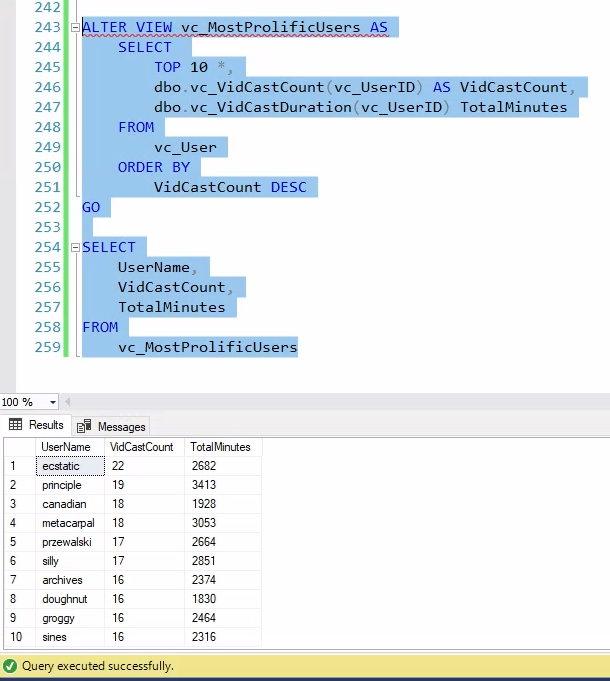
**PART 2:**

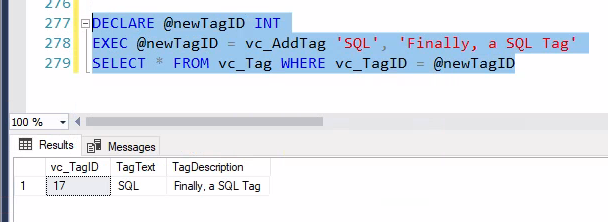


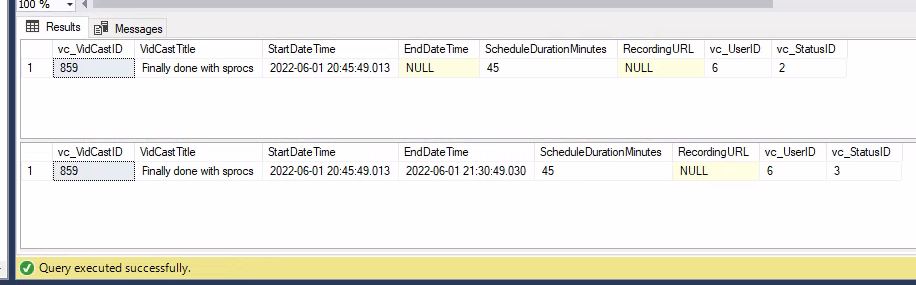












**SQL CODE:**

/\*

Author : Nick Videtti

Course : IST659 M407

Term : Spring (April - June) 2022

\*/

----------LAB 8----------

-----PART 1-----

--Declare a variable

DECLARE @ISTHISNULL VARCHAR(30) --Starts out as null

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

--Set the variable to something other than NULL

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

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

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

RETURNS INT AS

BEGIN

--First, declare the variable to temporarily hold the result

DECLARE @returnvalue INT --the data type matches the RETURNS clause

--Do whatever needs to be done to se the variable to the correct value

SET @returnvalue = @firstnumber + @secondnumber

--Return the value to the calling statement

RETURN @returnvalue

END

GO

SELECT dbo.AddTwoInts(5,10)

--Function to count the VidCasts made by a specific user

CREATE FUNCTION dbo.vc\_VidCastCount(@userID INT)

RETURNS INT AS --COUNT() is an integer value, so return it as an int

BEGIN

DECLARE @returnvalue INT --matches the function's return type

/\*Get the count of VidCasts for the provided userID and assign that value to

@returnvalue. Note that we use the @userID parameter in the WHERE clause to

limit our count to that user's VidCast records.\*/

SELECT @returnvalue = COUNT(vc\_UserID) FROM vc\_VidCast

WHERE vc\_VidCast.vc\_UserID = @userID

--Return @returnvalue to the calling code.

RETURN @returnvalue

END

GO

SELECT

TOP 10 \*,

dbo.vc\_VidCastCount(vc\_UserID) VidCastCount

FROM

vc\_User

ORDER BY

VidCastCount DESC

GO

--Function to retrieve the vc\_TagID for a given tag's text

CREATE FUNCTION dbo.vc\_TagIDLookup(@tagText varchar(20))

RETURNS INT AS --vc\_TagID is an int, so we'll match that

BEGIN

DECLARE @returnvalue int --Matches the function's return type

/\*Get the vc\_TagID of the vc\_Tag record whose TagText

matches the parameter and assign that value to @returnvalue.\*/

SELECT

@returnValue = vc\_TagID

FROM

vc\_Tag

WHERE

TagText = @tagText

--Send the vc\_TagID back to the caller

RETURN @returnvalue

END

GO

SELECT dbo.vc\_TagIDLookup('Music')

SELECT dbo.vc\_TagIDLookup('Tunes')

--Create a view to retrieve the top 10 vc\_Users and their VidCast counts

CREATE VIEW vc\_MostProlificUsers AS

SELECT

TOP 10 \*,

dbo.vc\_VidCastCount(vc\_UserID) AS VidCastCount

FROM

vc\_User

ORDER BY

VidCastCount DESC

GO

SELECT

\*

FROM

vc\_MostProlificUsers

--Create a procedure to update a vc\_User's email address

--The first parameter is the user name for the user to change

--The second is the new email address

CREATE PROCEDURE vc\_ChangeUserEmail(@userName varchar(20), @newEmail varchar(50))

AS

BEGIN

UPDATE vc\_User SET EmailAddress = @newEmail

WHERE UserName = @userName

END

GO

EXEC vc\_ChangeUserEmail 'tardy', 'kmstudent@syr.edu'

SELECT \* FROM vc\_User WHERE UserName = 'tardy'

INSERT INTO vc\_Tag (TagText) VALUES ('Cat Videos')

SELECT \* FROM vc\_Tag WHERE vc\_TagID = @@IDENTITY

INSERT INTO vc\_Tag (TagTExt) VALUES ('Dog Videos')

SELECT \* FROM vc\_Tag WHERE vc\_TagID = SCOPE\_IDENTITY()

/\*Create a procedure that adds a row to the UserLogin table.

This procedure should be run when a user logs in.

It will record who they are and from where they're logging in.\*/

CREATE OR ALTER PROCEDURE vc\_AddUserLogin(@userName varchar(20), @loginFrom varchar(50))

AS

BEGIN

--We have the user name, but we need the user ID for the login table

--First, declare a variable to hold the ID

DECLARE @userID int

SELECT @userID = vc\_UserID FROM vc\_User

WHERE UserName = @userName

--Now we can add the row using an INSERY statement

INSERT INTO vc\_UserLogin (vc\_UserID, LoginLocation)

VALUES(@userID, @loginFrom)

--Lastly, return the SCOPE\_IDENTITY() so the calling code knows the primary key of the row we just added

RETURN SCOPE\_IDENTITY()

END

GO

DECLARE @addedValue INT

EXEC @addedValue = vc\_AddUserLogin 'tardy', 'localhost'

SELECT

vc\_User.vc\_UserID,

vc\_User.UserName,

vc\_UserLogin.UserLoginTimestamp,

vc\_UserLogin.LoginLocation

FROM

vc\_User

JOIN

vc\_UserLogin

ON vc\_User.vc\_UserID = vc\_UserLogin.vc\_UserID

WHERE

vc\_UserLoginID = @addedValue

-----PART 2-----

--Create a function to retrieve a vc\_UserID for a given user name

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')

CREATE FUNCTION dbo.vc\_TagVidCastCount(@vc\_TagID INT)

RETURNS INT AS

BEGIN

DECLARE @returnvalue INT

SELECT

@returnvalue = COUNT(DISTINCT vc\_VidCastID)

FROM

vc\_VidCastTagList

WHERE

vc\_TagID = @vc\_TagID

RETURN @returnvalue

END

GO

SELECT

vc\_Tag.TagText,

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

FROM

vc\_Tag

CREATE FUNCTION vc\_VidCastDuration(@vc\_UserID INT)

RETURNS INT AS

BEGIN

DECLARE @VCDUR INT

SELECT

@VCDUR = SUM(DATEDIFF(n,StartDateTime,EndDateTime))

FROM

vc\_VidCast

WHERE

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

AND vc\_UserID = @vc\_UserID

RETURN @VCDUR

END

GO

SELECT

\*,

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

FROM

vc\_User

CREATE VIEW vc\_TagReport AS

SELECT

vc\_Tag.TagText,

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

FROM

vc\_Tag

GO

SELECT

\*

FROM

vc\_TagReport

ORDER BY

VidCasts DESC

ALTER VIEW vc\_MostProlificUsers AS

SELECT

TOP 10 \*,

dbo.vc\_VidCastCount(vc\_UserID) AS VidCastCount,

dbo.vc\_VidCastDuration(vc\_UserID) TotalMinutes

FROM

vc\_User

ORDER BY

VidCastCount DESC

GO

SELECT

UserName,

VidCastCount,

TotalMinutes

FROM

vc\_MostProlificUsers

/\*Create a stored procedure to add a new Tag to the database

Inputs:

@tagText: the text of the new tag

@description: a brief description of the tag (nullable)

Returns:

@@identity with the value inserted\*/

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(@VID\_CAST\_ID INT) AS

BEGIN

UPDATE vc\_VidCast

SET

EndDateTime = GETDATE(),

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

WHERE

vc\_VidCastID = @VID\_CAST\_ID

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 = SCOPE\_IDENTITY()

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

EXEC vc\_FinishVidCast @newVC

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