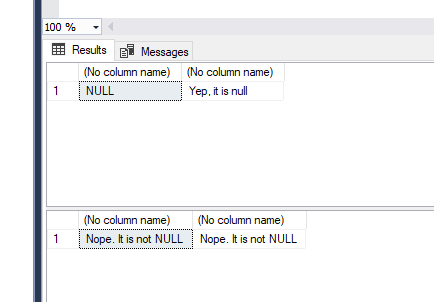
Nicholas Lichtsinn

Lab 8

6/2/2021

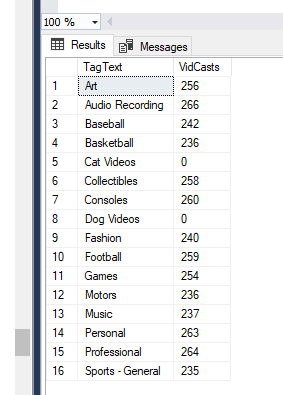
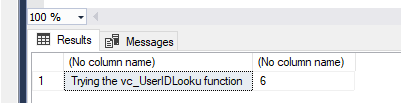
Part 1 - Introducing Functions, Views, and Stored Procedures

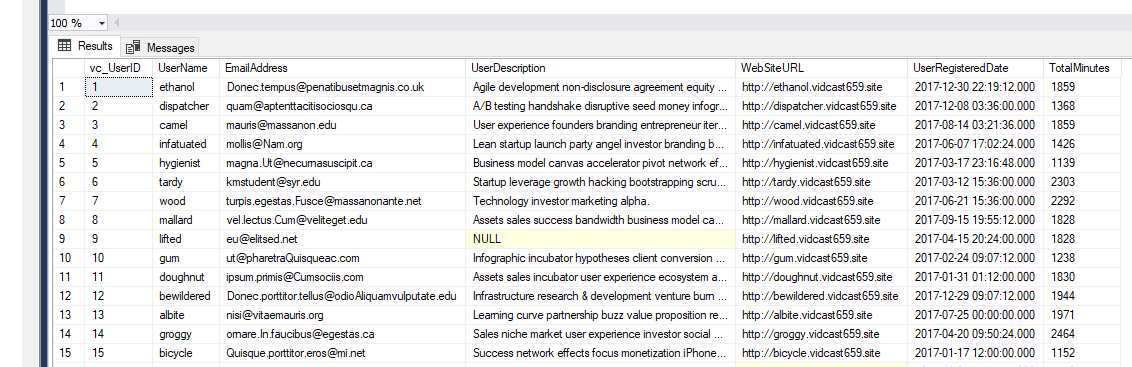


2. Lines 49-53 are selecting the top 10 User ID’s with the most VidCasts in descending order. The code knows that the vc\_User record with userID=20 has 22 records because in the function we created we said to count the number of Vidcasts for the given userID.
3. Lines 75 & 76 are calling the function vc\_TagIDLookup that we just created, calling the tag ID that corresponds with the tag text that we input. Line 76 pulls NULL because there is not a TagID that corresponds with the tagtext ‘Tunes’.
4. In lines 94-104 we are creating a stored procedure to update a user’s email address where we have to input the username of the email we want to update so that we are not updating everything.
5. The UserLoginTimeStamp value will be different depending on when you call the AddUserLogin function. We could just run a SELECT statement with SELECT dbo.vc\_AddUserLogin and then specify the username and location.

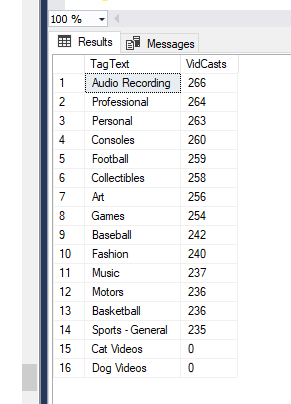
Part 2 - Putting It All Together

vc\_UserIDLookup

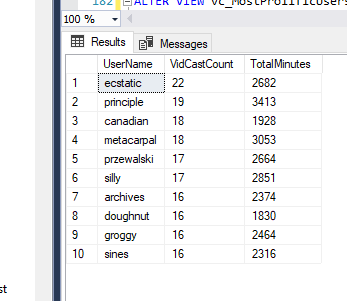
1. vc\_TagVidCastCount
2. vc\_VidCastDuration



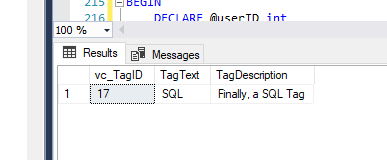
1. vc\_TagReport



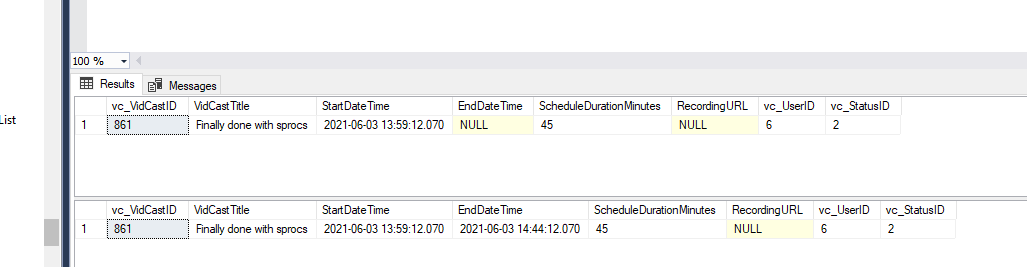
1. vc\_MostProlificUsers



1. vc\_AddTag



1. vc\_FinishVidCast



SQL 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') -- How about now?

GO

CREATE FUNCTION dbo.AddTwoInts(@firstNumber int, @secondNumber int)

RETURNS int AS

BEGIN

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

DECLARE @returnValue int -- the data type matches the "RETURNS" clause

-- Do whatever needs to be done to set that 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)

GO

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

RETURNS int AS

BEGIN

DECLARE @returnValue int

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

WHERE vc\_VidCast.vc\_UserID = @UserID

RETURN @returnValue

END

GO

SELECT TOP 10

\*

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

FROM vc\_User

ORDER BY VidCastCount DESC

GO

CREATE FUNCTION dbo.vc\_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.vc\_TagIDLookup('Music')

SELECT dbo.vc\_TagIDLookup('Tunes')

GO

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

GO

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()

GO

CREATE PROCEDURE vc\_AddUserLogin(@userNAme varchar(20), @loginFrom varchar(50))

AS

BEGIN

DECLARE @userID int

SELECT @userID = vc\_UserID FROM vc\_User

WHERE UserName = @userName

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

VALUES (@userID, @loginFrom)

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

GO

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\_UserIDLooku function', dbo.vc\_UserIDLookup('tardy')

GO

CREATE FUNCTION dbo.vc\_TagVidCastCount(@tagID int)

RETURNS int AS

BEGIN

DECLARE @returnValue int

SELECT @returnValue = COUNT(vc\_VidCastID) FROM vc\_VidCastTagList

WHERE vc\_VidCastTagList.vc\_TagID = @tagID

RETURN @returnValue

END

GO

SELECT

vc\_Tag.TagText

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

FROM vc\_Tag

GO

ALTER FUNCTION dbo.vc\_VidCastDuration(@userID int)

RETURNS int AS

BEGIN

DECLARE @returnValue int

SELECT @returnValue = SUM(DATEDIFF(n, StartDateTime, EndDateTime)) FROM vc\_VidCast

WHERE vc\_VidCast.vc\_UserID = @userID

RETURN @returnValue

END

GO

SELECT

\*

, dbo.vc\_vidcastduration(vc\_userID) as TotalMinutes

FROM vc\_User

GO

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

GO

ALTER VIEW vc\_MostProlificUsers AS

SELECT TOP 10

\*

, dbo.vc\_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

GO

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

BEGIN

DECLARE @tagID int

SELECT @tagID = vc\_TagID FROM vc\_Tag

WHERE TagText = @tagText

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

GO

ALTER PROCEDURE vc\_FinishVidCast(@vc\_VidCastID int) AS

BEGIN

UPDATE vc\_VidCast SET EndDateTime = GETDATE()

WHERE vc\_VidCastID = @vc\_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 = SCOPE\_IDENTITY()

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

EXEC vc\_FinishVidCast @newVC

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