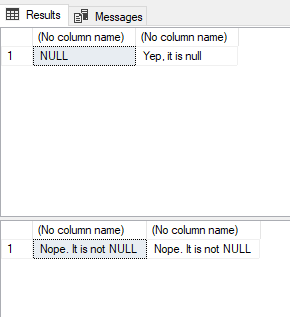
**Part 1—Introducing Functions, Views, and Stored Procedures**

**ISNULL Function:**



**Lines 49-53:**

These lines are selecting the Top 10 rows from the vc\_User table and adding the VidCastCount column using the dbo.vc\_VidCastCount function we created before. They are also sorting the results in descending order of VidCastCount.

**Lines 75-76:**

These lines are selecting the value created by running our dbo.vc\_TagIDLookup function when the vc\_Tag equals Music and Tunes, but since we didn’t alias, there is no column name.

**Lines 79-87:**

These lines are creating a view that displays the top 10 users (by VidCastCount) and sorting them in descending order of VidCastCount.

**Lines 91-104:**

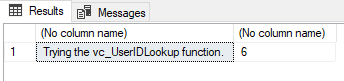
These lines are creating a procedure to change the email of a user already in our database that requires us to input the user name and email address. Additionally, they are executing this function to update and show the results of the update to tardy’s email address.

**Lines 110-144:**

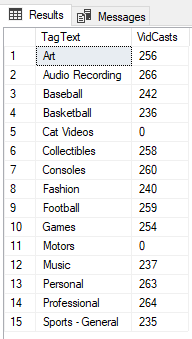
The UserLoginTimestamp value is different than the example because it outputs the date and time that the record is entered into the database which was different for my database than the example. We could simplify this code by removing all the comments.

**Part 2—Putting All Together**

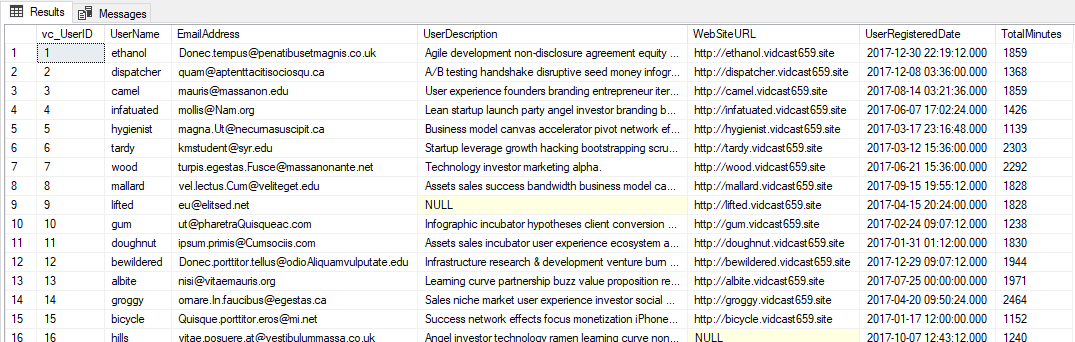
**vc\_UserIDLookup Function**:



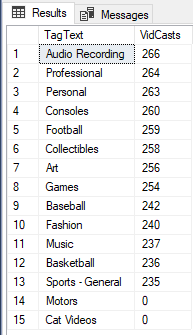
**vc\_TagVidCastCount Function:**



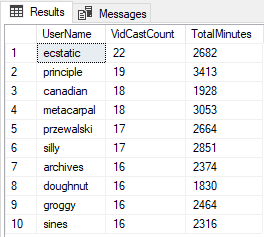
**vc\_VidCastDuration Function:**



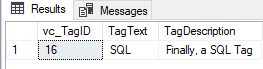
**vc\_TagReport View:**



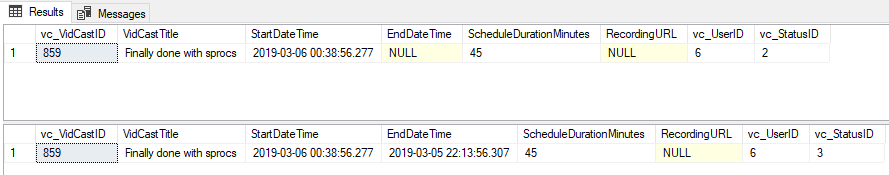
**Altered vc\_MostProlificUsers View:**



**vc\_AddTag Stored Procedure:**



**vc\_FinishVidCast Procedure:**



**SQL Query:**

-- 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?

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

RETURNS int AS

BEGIN

--First, delcare the variable to temporarily 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)

--Function to count the VidCasts made by a given 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 --mathces the function's return type

/\*

Get the count of the 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) as 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

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

This procedure is run when a user logs in and we need

to record who they are and from where they're logging in.

\*/

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

AS

BEGIN

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

--First, declare a variable to hold the ID

DECLARE @userID int

--Get the vc\_UserID for the UserName provided and store it in @userID

SELECT @userID = vc\_UserID FROM vc\_User

WHERE UserName = @userName

--Now we can add the row using the INSERT statement

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

VALUES (@userID, @loginFrom)

--Now return the @@identity so the calling code knows where

--the data ended up

RETURN @@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

/\*

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\_User.vc\_UserID FROM vc\_User

WHERE @userName = vc\_User.UserName

RETURN @returnValue

END

GO

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

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

RETURNS int AS

BEGIN

DECLARE @returnValue int

SELECT @returnValue = COUNT(vc\_VidCastTagList.vc\_VidCastTaglistID) FROM vc\_VidCastTagList

WHERE @tagID = vc\_VidCastTagList.vc\_TagID

RETURN @returnValue

END

GO

SELECT

vc\_Tag.TagText

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

FROM vc\_Tag

GO

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

RETURNS int AS

BEGIN

DECLARE @returnValue int

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

WHERE @userID = vc\_VidCast.vc\_UserID AND vc\_VidCast.vc\_StatusID = 3

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

/\*

Create a stored procedure to add a new Tag to the database

Inputs:

@tagText : the text of the new tag

@description : a briefdescription of the tag (nullable)

Returns:

@@identity witht he value inserted

\*/

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

BEGIN

--Codethe procedure here!

DECLARE @tagID int

SELECT @tagID = vc\_TagID FROM vc\_Tag

WHERE @tagText = vc\_Tag.TagText AND @description = vc\_Tag.TagDescription

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

CREATE PROCEDURE vc\_FinishVidCast(@vidCastID int) AS

BEGIN

UPDATE vc\_VidCast SET EndDateTime = GETDATE(), vc\_StatusID = 3

WHERE @vidCastID = vc\_Vidcast.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, 145, 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