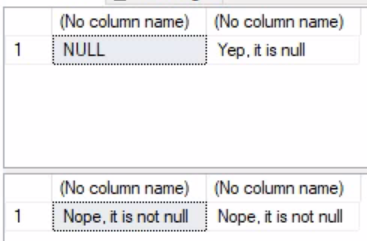
Lab 8  
Forrest Fallon  
  
1.  


2. The select function is calling upon the database function we created to count VidCasts that User’s have created (that are documented in our database). This function counts the amount of times a userID appears in the VidCast table, which implies them creating a vidcast. Our select statement then counts all instances of that happening for each user, and returns the most instances and sorts them in a descending order. Specifically, there are 22 different times that UserID 20 appears on the VidCast table, therefore they show up in this function with a count of 22 at the top of our list.

3. Lines 75 and 76 call upon the dbo.vc\_TagIDLookup function that we created in the previous block, which searches our database for specific TagTexts that a vidcast may be labeled with. The particular search in line 76 searches for a TagText that hasn’t been placed in our system yet.

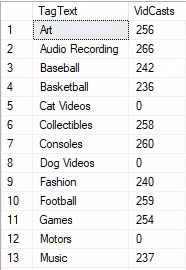
4. Lines 79 through 87 create a simplified input that anyone can use to access the top 10 uploaders of VidCasts as shown by our database. This view simplifies a complicated select statement into a simple, one input executable that is also read-only, so it can be passed along with confidence.

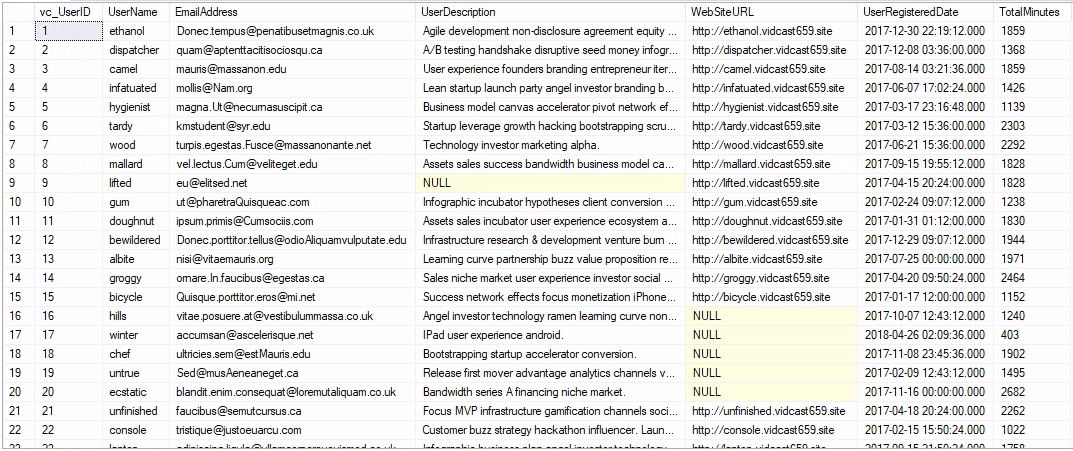
5. Lines 91 through 104 create a procedure that allows for someone without full access to the database to update email addresses for users already within the database. This allows for changes to be made to specifically the email addresses on file without the risk of updating files not needing to be updated.

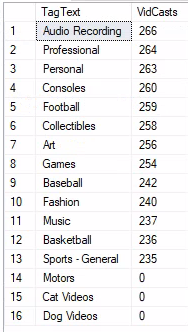
6. The login timestamps are different because the execution of the @addedValue function was completed within my database at the time of completing this lab, meanwhile the record created for the lab was done at the time of creating the lab!  
To simplify, we could simply run a SELECT \* from dbo.vc\_UserLogin query to return the same statement, minus the UserName.

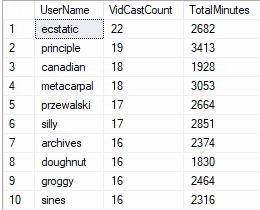
7. dbo.vc\_UserIDLookup



8. VidCastTags  


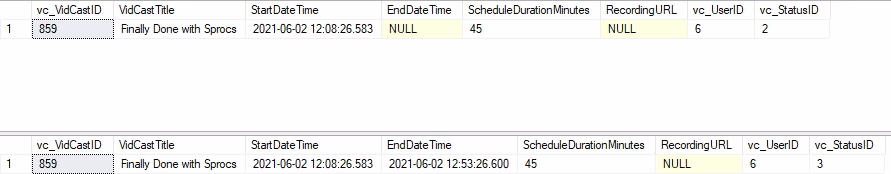
9. TotalDuration  


10. VidCasts view  


11. Alter MostProlificUsers  


12. vc\_AddTag  


13. FinishedVidCast



--declare a variable

declare @isThisNull varchar(30) --starts out as null

select @isThisNull, isnull(@isThisNull, 'Yep, it is null')

--set the variable as 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 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 amount of vidcasts a user has made

create function dbo.vc\_VidCastCount(@UserID int)

returns int as --count() is an integer value, so it returns it as an int

begin

declare @returnValue int --matches 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 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

--function to retrieve the vc\_TagID for 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 functions 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 the 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()

/\* createa 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 where they log in from.

\*/

create or alter procedure vc\_AddUserLogin(@userName varchar(20), @loginFrom varchar(50))

as

begin

--we have the user name, but we need the UserID from 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 INSERT statement

insert into vc\_UserLogin (vc\_UserID, LoginLocation)

values (@userID, @loginFrom)

--lastly, return the scopeidentity() 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

select \* from dbo.vc\_UserLogin

/\*

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 fuction.', dbo.vc\_UserIDLookup('tardy')

create or alter function dbo.vc\_TagVidCastCount(@VidCastID int)

returns int as

begin

declare @returnValue int

select @returnValue = count(vc\_TagID) from vc\_VidCastTagList

where vc\_TagID = @VidCastID

return @returnValue

end

go

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

From vc\_Tag

create 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\_UserID = @UserID

return @returnValue

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) as 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) as 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(@vidCastID int)

as

begin

update vc\_VidCast set EndDateTime = getdate(), vc\_StatusID = '3'

where vc\_VidCastID = @vidCastID

return scope\_identity()

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