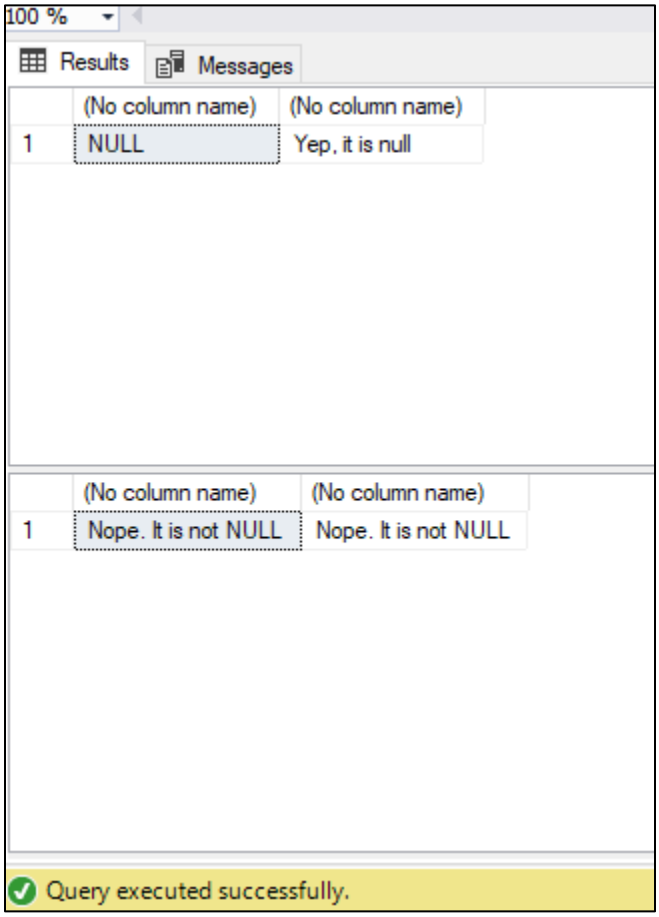
11/19/2020
Lab 8

Part 1 Introduce Functions, Views, and Stored Procedures

Is Null Function



100 %

Results | Messages

	(No column name)	(No column name)
1	NULL	Yep, it is null

	(No column name)	(No column name)
1	Nope. It is not NULL	Nope. It is not NULL

Query executed successfully.

Abstracting Routine Calculation

Essentially what lines 49-53 are doing is:

1. Apply the newly created function, dbo.vc_VidCastCount, to each of the UserID's in the vc_User table and add the result of that function as a new column in that table.
2. Sort the table in descending order by the new colum, i.e. the number of VidCasts that each user has created, in other words this will be from greatest to least starting with greatest at the top
3. Lastly, now that we have our column created and a sorted table, the SQL command only spits out the first 10 rows of that table because that's what we specified for in line 49 of the code.
4. Since we used the * in our select clause, it is going to return all columns in the table.

The code knows that, for example userID 20, has 22 VidCasts because that's exactly what the purpose of the function is. The function tells us how many vidcasts a user has, and we called on that function for each userID in the vc_User table.

11/19/2020
Lab 8

Performing Data Lookups

Lines 75 and 76 are two different examples of testing out the function that was created. It will take each parameter, which in this case is 'Music' and 'Tunes', and plug that into the function and run it. Music is a match and returns us back that the tagID for the tagText of 'Music' is 4. With 'Tunes', however, we get a null back which means that it was not a match and therefore 'Tunes' does not exist anywhere in the tagText attribute in the database.

Most Prolific Users View

This select statement looks familiar, because it is, it is the same one that we used earlier just wrapped in a view. Now that we have this saved as a view, we can call upon it at any time and it will show us that result without having to retype all of the code for the select statement. Views are kind of like shortcuts in a way.

Stored Procedure Update Email

This code is first building the stored procedure to change an email address, there are two parameters that need to be entered and that is the userName and the new email address that is desired. We are already familiar with an update statement from previous coursework, but this just gets wrapped in a beign and end clause so that SQL knows it is an individual block of code that needs to be run. Lastly, we use the built in exec function to actually execute the stored procedure we just made and then we run a select statement to make sure that it was in fact successful.

Stored Procedure User Login TimeStamp

The timestamp is different because it is taken at the time that a user logs in, which in this case is at the time when the stored procedure was executed. To make this simpler, we could even create a function that processes all of the code to execute the stored procedure.

Part 2 Introduce Functions, Views, and Stored Procedures

vc_UserIDLookup function

Results | Messages

	(No column name)
1	6

11/19/2020
Lab 8

Count the VidCastID's for a given TagID function

	TagText	VidCasts
1	Art	256
2	Audio Recording	266
3	Baseball	242
4	Basketball	236
5	Cat Videos	0
6	Collectibles	258
7	Consoles	260
8	Fashion	240
9	Football	259
10	Games	254
11	Motors	0
12	Music	237
13	Personal	263
14	Professional	264
15	Sports - General	235

Count VidCast minutes per User ID

100 %

Results | Messages

	vc_UserID	UserName	EmailAddress	UserDescription	WebSiteURL	UserRegisteredDate	TotalMinutes
1	1	ethanol	Donec.tempus@penatibusetmagnis.co.uk	Agile development non-disclosure agreement equity ...	http://ethanol.vidcast659.site	2017-12-30 22:19:12.000	1859
2	2	dispatcher	quam@aptenttacitisociosqu.ca	A/B testing handshake disruptive seed money infogr...	http://dispatcher.vidcast659.site	2017-12-08 03:36:00.000	1368
3	3	camel	mauris@massanon.edu	User experience founders branding entrepreneur iter...	http://camel.vidcast659.site	2017-08-14 03:21:36.000	1859
4	4	infatuated	mollis@Nam.org	Lean startup launch party angel investor branding b...	http://infatuated.vidcast659.site	2017-06-07 17:02:24.000	1426
5	5	hygienist	magna.Ut@necumasuscipit.ca	Business model canvas accelerator pivot network ef...	http://hygienist.vidcast659.site	2017-03-17 23:16:48.000	1139
6	6	tardy	kmstudent@syr.edu	Startup leverage growth hacking bootstrapping scru...	http://tardy.vidcast659.site	2017-03-12 15:36:00.000	2303
7	7	wood	turpis.egestas.Fusce@massanonante.net	Technology investor marketing alpha.	http://wood.vidcast659.site	2017-06-21 15:36:00.000	2292
8	8	mallard	vel.lectus.Cum@veliteget.edu	Assets sales success bandwidth business model ca...	http://mallard.vidcast659.site	2017-09-15 19:55:12.000	1828
9	9	lifted	eu@elitsed.net	NULL	http://lifted.vidcast659.site	2017-04-15 20:24:00.000	1828
10	10	gum	ut@pharetraQuisqueac.com	Infographic incubator hypotheses client conversion ...	http://gum.vidcast659.site	2017-02-24 09:07:12.000	1238
11	11	doughnut	ipsum.primis@Cumsociis.com	Assets sales incubator user experience ecosystem a...	http://doughnut.vidcast659.site	2017-01-31 01:12:00.000	1830
12	12	bewildered	Donec.porttitor.tellus@odioAliquamvulputate.edu	Infrastructure research & development venture burn ...	http://bewildered.vidcast659.site	2017-12-29 09:07:12.000	1944
13	13	albite	nisi@vitaemauris.org	Learning curve partnership buzz value proposition re...	http://albite.vidcast659.site	2017-07-25 00:00:00.000	1971
14	14	groggy	ornare.In.faucibus@egestas.ca	Sales niche market user experience investor social ...	http://groggy.vidcast659.site	2017-04-20 09:50:24.000	2464
15	15	bicycle	Quisque.porttitor.eros@mi.net	Success network effects focus monetization iPhone...	http://bicycle.vidcast659.site	2017-01-17 12:00:00.000	1152
16	16	hills	vitae.posuere.at@vestibulummassa.co.uk	Angel investor technology ramen learning curve non...	NULL	2017-10-07 12:43:12.000	1240
17	17	winter	accumsan@ascelerisque.net	IPad user experience android.	NULL	2018-04-26 02:09:36.000	403
18	18	chef	ultricies.sem@estMauris.edu	Bootstrapping startup accelerator conversion.	NULL	2017-11-08 23:45:36.000	1902
19	19	untrue	Sed@musAeneaneget.ca	Release first mover advantage analytics channels v...	NULL	2017-02-09 12:43:12.000	1495

Query executed successfully. | DESKTOP-G777CHI (15.0 RTM) | DESKTOP-G777CHI\lilgi ... | IST_659_Vidcast | 00:00:00 | 68 rows

11/19/2020
Lab 8

vc_TagReport View

	TagText	VidCasts
1	Audio Recording	266
2	Professional	264
3	Personal	263
4	Consoles	260
5	Football	259
6	Collectibles	258
7	Art	256
8	Games	254
9	Baseball	242
10	Fashion	240
11	Music	237
12	Basketball	236
13	Sports - General	235
14	Motors	0
15	Cat Videos	0

Query executed successfully.

Alter view vc_MostProlificUsers

	UserName	VidCastCount	TotalMinutes
1	ecstatic	22	2682
2	principle	19	3413
3	canadian	18	1928
4	metacarpal	18	3053
5	przewalski	17	2664
6	silly	17	2851
7	archives	16	2374
8	doughnut	16	1830
9	groggy	16	2464
10	sines	16	2316

11/19/2020
Lab 8

Stored procedure to add a new tag

	vc_TagID	TagText	TagDescription
1	16	SQL	Finally, a SQL Tag

Stored Procedure to update VidCast status

Results | Messages

	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	863	Finally done with sprocs	2020-11-19 23:20:05.740	NULL	45	NULL	6	2

	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	863	Finally done with sprocs	2020-11-19 23:20:05.740	2020-11-20 00:05:05.817	45	NULL	6	3

Query executed successfully. | DESKTOP-G777CHI (15.0 RTM) | DESKTOP-G777CHI\lilgi ... | IST_659_Vidcast | 00:00:00 | 2 rows

```
/*
	Course: IST 659
	Term: Fall 2020
*/

-- 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


--our first user defined function
create function dbo.AddTwoInts (@firstNumber int, @secondNumber int) -- create the
function, name, and parameters
returns int as begin declare @returnValue int
	-- First, declare the variable to temporarily hold the result
set @returnValue = @firstNumber + @secondNumber                                  --
do whatever needs to be done to set that variable to the correct value
return @returnValue end
                    -- return the value to the calling statement
go --call on the first user defined function
select dbo.AddTwoInts(5, 10)


go -- function to count the vidcasts made by a given user
create function dbo.vc_VidCastCount(@userID int)
returns int as begin declare @returnValue int
/* get the count of the vidcasts for the provided userID and
assign that value to @return value. 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 end
go -- call the function that we just created
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 begin declare @returnValue int
select @returnValue = vc_TagID from vc_Tag
where TagText = @tagText
return @returnValue end
go -- execute the function that was just created
select dbo.vc_TagIDLookup('Music')
select dbo.vc_TagIDLookup('Tunes')


go -- 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 -- test function to retrieve a vc_UserID for a given user name
select dbo.vc_UserIDLookup('tardy')


go -- create a function to count the VidCastID's for a given TagID
create function dbo.vc_VidCastsPerTag(@tagID int)
returns int as begin declare @returnValue int
select @returnValue = count(vc_VidCastID) from vc_VidCastTagList
where vc_TagID = @tagID
return @returnValue end
go -- test the function to count the VidCastID's for a given TagID
select vc_Tag.TagText, dbo.vc_VidCastsPerTag(vc_Tag.vc_TagID) as VidCasts from vc_Tag


go -- Code function called vc_VidCastDuration to count vidcast minutes per user
create function dbo.vc_VidCastDuration (@userID int)
returns int as begin declare @returnValue int
select @returnValue = sum(datediff(n, StartDateTime, EndDateTime))
from vc_VidCast
join vc_Status on vc_Status.vc_StatusID = vc_VidCast.vc_StatusID
where vc_Status.StatusText = 'Finished' and vc_UserID = @userID
return @returnValue end
go -- test function called vc_VidCastDuration to count vidcast minutes per user
select * , dbo.vc_VidCastDuration(vc_UserID) as TotalMinutes from vc_User


go --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
```

11/19/2020
Lab 8

```
from vc_User order by VidCastCount desc
go --call the view that was just created
select * from vc_MostProlificUsers


go -- alter view vc_MostProlificUsers to add a column called totalMinutes
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 --call the view that was just created
select UserName, VidCastCount, TotalMinutes from vc_MostProlificUsers


go -- create a view called vc_TagReport
create view vc_TagReport as select vc_Tag.TagText,
dbo.vc_VidCastsPerTag(vc_Tag.vc_TagID) as VidCasts
from vc_Tag
go -- call the view that was just created
select * from vc_TagReport order by VidCasts desc


go /*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 -- call the stored procedure that was just created
exec vc_ChangeUserEmail 'tardy', 'kmstudent@syr.edu'
go -- check that the statement worked and change was made
select * from vc_User where UserName = 'tardy'
go -- the @@identity property assigns any added values here
insert into vc_Tag (TagText) values ('Cat Videos')
select * from vc_Tag where vc_TagID = @@identity


/*Start walk through of stored procedure insert*/

go /* 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 an 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 -- execute the stored procedure
declare @addedValue int
exec @addedValue = vc_AddUserLogin 'tardy', 'localhost'
```

11/19/2020
Lab 8

```
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

/*End walk through of stored procedure insert*/


go -- create a stored procedure to add a new tag to the database inputs
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 -- execute the stored procedure and add a new tag to the vc_Tag table
DECLARE @newTagID int
EXEC @newTagID = vc_AddTag 'SQL', 'Finally, a SQL Tag'
SELECT * FROM vc_Tag where vc_TagID = @newTagID


go/*Code a stored procedure called vc_FinishVidCast that accepts an int as a parameter
that
will be a vc_VidCastID that we will need to mark as finished. The act of finishing a
VidCast means we must change its EndDateTime to be the current Date and Time (think
GetDate()) and change the vc_StatusID to the vc_StatusID for the 'Finished' status.
All the work can be done in a single UPDATE statement inside the stored procedure. Be
sure to code the WHERE clause!*/

create procedure vc_FinishVidCast (@vidCastID int) as begin
update vc_VidCast set EndDateTime = getdate()
where vc_VidCastID = @vidCastID
update vc_VidCast set vc_StatusID = 3 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
```