

## PART 1:

### Functions:

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

### Functions That Are More...welll...Functional:

Lines 49 to 53 of the code actually is the query that selects the top 10 Vidcast users by userID from the schema VidCastCount, which is based on the number of Vidcasts made by that user. The query retrieves the top 10 users who have posted videos and orders them from the highest number to the lowest.

The code knows that the vc\_User record with the vc\_UserID = 20 has 22 vc\_VidCasts records because we assigned a count of the Vidcasts for the provided userID and assigned th values to the @returnValu, and used the WHERE clause to limit the count to only that users VidCast record in the previous code (line 30-45)

Results		Messages					
	vc_UserID	UserName	EmailAddress	UserDescription	WebSiteURL	UserRegisteredDate	VidCastCount
1	20	ecstatic	blandit.enim.consequat@loremutaliquam.co.uk	Bandwidth series A financing niche market.	NULL	2017-11-16 00:00:00.000	22
2	40	principle	ac.uma@miac.com	Business-to-business ecosystem ramen social media...	http://principle.vidcast659.site	2017-11-01 12:14:24.000	19
3	24	metacarpal	et.magna.Praesent@placeraaugueSed.org	Research & development startup long tail strategy g...	http://metacarpal.vidcast659.site	2017-05-30 11:31:12.000	18
4	43	canadian	Curabitur.dictum.Phasellus@eleifendnec.com	Agile development ownership business-to-consumer...	http://canadian.vidcast659.site	2017-06-27 05:16:48.000	18
5	26	przewalski	amet@Maurismolestie.org	NULL	http://przewalski.vidcast659.site	2017-02-11 08:52:48.000	17
6	36	silly	accumsan@gravidasagittisDuis.net	Stock founders early adopters low hanging fruit A/B...	http://silly.vidcast659.site	2017-05-25 14:38:24.000	17
7	7	wood	turpis.egestas.Fusce@massanonante.net	Technology investor marketing alpha.	http://wood.vidcast659.site	2017-06-21 15:36:00.000	16
8	11	doughnut	ipsum.primis@Cumsociis.com	Assets sales incubator user experience ecosystem ...	http://doughnut.vidcast659.site	2017-01-31 01:12:00.000	16
9	14	groggy	omare.In.faucibus@egestas.ca	Sales niche market user experience investor social ...	http://groggy.vidcast659.site	2017-04-20 09:50:24.000	16
10	47	these	parturient.montes@ipsum.ca	Entrepreneur virality freemium crowdsource long tail ...	http://these.vidcast659.site	2017-12-07 03:50:24.000	16

### Performing Data Lookups :

Lines 75 and 76 basically are querying values in the column TagText to specify two searches for tags with the word “Music” (line 75) and “Tunes” (line 76). The TagText is set up as the parameter in the previous code, and the texts are looked up based on the vc\_TagID for the vc\_Tag record of that TagText.. When line 76 is executed, we received a Null because there are no TagTexts with ‘Tunes’ listed in the database.

### Views :

Lines 79 through 87 use the CREATE VIEW defined function from the vc\_User table in order to select

Lines 79 through 87 use the CREATE VIEW defined function from the vc\_User table in order to select the top 10 vc\_Users and their count of VidCasts and view the data points in a table format that is ordered from greatest to least of VidCast counts.

### Stored Procedure:

Lines 91 through 104 go through creating two parameters in order to update a vc\_User's email address. The first parameter is the user name for the user to change, and the second one is the user's email address. After, the stored procedure is created, then the vc\_ChangeUserEmail is executed and the username "Tardy" and the email address "kmstudent2syr.edu" is updated. Additionally, the last line (104) is a query to see the new user's information.

### @@Identity:

The UserLoginTimeStamp varies between my answer and the lab instructions because the stamp is from when the user login is last accessed. On my database, it was accessed today. One way we could simplify the code in the stored procedure above is to use the UPDATE function in order to update the UserLoginTimeStamp.

---

## *PART 2:*

### 1. Coding Your Own Functions:

Results Messages		
	(No column name)	(No column name)
1	Trying the vc_UserIDLookup function.	6

### 2. Author function called dbo.vc\_TagVidCastCount results

Results Messages		
	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

### 3. Code function to SUM the total number o minutes if actual duration for Vidcasts with Finished Status

Results		Messages					
	vc_UserID	UserName	EmailAddress	UserDescription	WebSiteURL	UserRegisteredDate	TotalMinutes
1	1	ethanol	Donec.tempus@penatibusetmagnis.co.uk	Agile development non-disclosure agreement equit...	http://ethanol.vidcast659.site	2017-12-30 22:19:12.000	216
2	2	dispatcher	quam@apertentacitisociosqu.ca	A/B testing handshake disruptive seed money info...	http://dispatcher.vidcast659.site	2017-12-08 03:36:00.000	NULL
3	3	camel	mauris@massanon.edu	User experience founders branding entrepreneur it...	http://camel.vidcast659.site	2017-08-14 03:21:36.000	172
4	4	infatuated	mollis@Nam.org	Lean startup launch party angel investor branding ...	http://infatuated.vidcast659.site	2017-06-07 17:02:24.000	15
5	5	hygienist	magna.Ut@necumasuscipit.ca	Business model canvas accelerator pivot network ...	http://hygienist.vidcast659.site	2017-03-17 23:16:48.000	15
6	6	tardy	kmstudent2syr.edu	Startup leverage growth hacking bootstrapping scr...	http://tardy.vidcast659.site	2017-03-12 15:36:00.000	NULL
7	7	wood	turpis.egestas.Fusce@massanonante.net	Technology investor marketing alpha.	http://wood.vidcast659.site	2017-06-21 15:36:00.000	NULL
8	8	mallard	vel.lectus.Cum@veliteget.edu	Assets sales success bandwidth business model c...	http://mallard.vidcast659.site	2017-09-15 19:55:12.000	NULL
9	9	lifted	eu@elitsed.net	NULL	http://lifted.vidcast659.site	2017-04-15 20:24:00.000	86
10	10	gum	ut@pharetraQuisqueac.com	Infographic incubator hypotheses client conversio...	http://gum.vidcast659.site	2017-02-24 09:07:12.000	129
11	11	doughnut	ipsum.primis@Cumsociis.com	Assets sales incubator user experience ecosystem...	http://doughnut.vidcast659.site	2017-01-31 01:12:00.000	NULL
12	12	bewildered	Donec.porttitor.tellus@odioAliquamvulp...	Infrastructure research & development venture bur...	http://bewildered.vidcast659.s...	2017-12-29 09:07:12.000	260
13	13	albite	nisi@vitaemauris.org	Learning curve partnership buzz value proposition ...	http://albite.vidcast659.site	2017-07-25 00:00:00.000	216
14	14	groggy	omare.In.faucibus@egestas.ca	Sales niche market user experience investor social...	http://groggy.vidcast659.site	2017-04-20 09:50:24.000	230
15	15	bicycle	Quisque.porttitor.eros@mi.net	Success network effects focus monetization iPhon...	http://bicycle.vidcast659.site	2017-01-17 12:00:00.000	NULL
16	16	hills	vitae.posuere.at@vestibulummassa.co.uk	Angel investor technology ramen learning curve n...	NULL	2017-10-07 12:43:12.000	NULL
17	17	winter	accumsan@ascelerisque.net	IPad user experience android.	NULL	2018-04-26 02:09:36.000	28
18	18	chef	ultrices.sem@estMauris.edu	Bootstrapping startup accelerator conversion.	NULL	2017-11-08 23:45:36.000	231

### 4.Creat View called vc\_TagReport

Results		Messages	
	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	...		

### 5. Author function called dbo.vc\_TagVidCastCount results

Results		Messages	
	UserName	VidCastCount	TotalMinutes
1	ecstatic	22	43
2	principle	19	29
3	canadian	18	101

4	metacarpal	18	101
5	przewalski	17	NULL
6	silly	17	259
7	archives	16	NULL
8	doughnut	16	NULL
9	groggy	16	230
10	sines	16	29

## 6. Coding Stored Procedure

	vc_TagID	TagText	TagDescription
1	16	SQL	Finally, a SQL Tag

## 7. Coding Stored Procedure called vc\_FinishVidCast

	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	860	Finally done with sprocs	2018-12-04 08:07:12.700	NULL	45	NULL	6	2

  

	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	860	Finally done with sprocs	2018-12-04 08:07:12.700	NULL	45	NULL	6	2

## 8. Final 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, @secondNumner int)
RETURNS int AS
BEGIN
```

```
--First, declare the variable to temporarily hold the results
```

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

```
--Do whatever needs to be done to set tht variable to the correct value
```

```
SET @returnValue = @firstNumber + @secondNumner
```

```
--Return the value to the calling statement
```

```
RETURN @returnValue
```

```
END
GO
```

```
--Functions 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--matches the fuctions return type
```

```
/*
```

```
Get the count of the Vidcasts for the provided userID and
assign that value to @returnValue. Note that we use th
e @userID parameter in the WHERE clause to limit our
count to that users Vidcast record.
```

```
*/
```

```
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 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_MostProlificUser AS
    SELECT TOP 10
        *
        , dbo.vc_VidCastCount(vc_UserID) as VidCastCount
    FROM vc_User
    ORDER BY VidCastCount DESC
GO

SELECT * FROM vc_MostProlificUser

--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', 'kmstudent2syr.edu'

```

```

--See effect of code above
SELECT * FROM vc_User WHERE UserName = 'tardy'

--@@Identity
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 tabl
This procedure is run when a user logs in a 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 the @userID
    SELECT @userID = vc_UserID FROM vc_User
    WHERE UserName = @userName

    --Now we can add the row against 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

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-----
*/

/*
1. Create a function to retrieve a vc_UserID for a given user name
*/

GO
CREATE FUNCTION dbo.vc_UserIDLookup(@userName varchar(20))
RETURNS int AS
BEGIN
    DECLARE @returnValue int

    --TODO: Write the code to assign the correct vc_UserID
    -- to @returnValue
    SELECT @returnValue = vc_UserID FROM vc_User
    WHERE UserName = @userName

    RETURN @returnValue
END

```

```
-- --  
GO
```

```
--Query used to get results of 1. code--  
SELECT 'Trying the vc_UserIDLookup function.', dbo.vc_UserIDLookup('tardy')
```

```
/*  
2. Create a function to calculate count of vc_VidCastIDs for a given vc_TagID  
*/
```

```
CREATE FUNCTION dbo.vc_TagVidCastCount(@TagID int)  
RETURNS int AS  
BEGIN  
    DECLARE @returnValue int  
  
    SELECT @returnValue = COUNT(vc_TagID) from vc_VidCastTagList  
    WHERE vc_VidCastTagList.vc_TagID = @TagID  
  
    RETURN @returnValue  
END  
GO
```

```
--Query used to get results of Tags with number of Vidcasts--  
SELECT  
    vc_Tag.TagText  
    , dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) as Vidcasts  
FROM vc_Tag
```

```
/*  
3. Create a function that sums the total number of minutes of  
actual duration for VidCast swith Finished status given vc_UserID  
as a Parameter. Return SUM as a int.  
*/
```

```
CREATE FUNCTION dbo.vc_VidCastDuration(@UserID varchar(20))  
RETURNS int AS  
BEGIN  
    DECLARE @returnValue int  
    SELECT @returnValue = DATEDIFF(n, StartDateTime, EndDateTime) FROM vc_VidCast  
    WHERE vc_VidCast.vc_UserID = @UserID  
    RETURN @returnValue  
END  
GO
```

```
--Query to get results--
```

```
SELECT  
    *  
    , dbo.vc_VidCastDurational(vc_UserID) as TotalMinutes  
FROM vc_User
```

```
/*  
4. Create view called Vc_TagReport  
*/
```

```
CREATE VIEW vc_TagReport AS
```

```
SELECT dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) as VidCasts
FROM vc_Tag
GO
```

```
--Query--
```

```
SELECT
    vc_Tag.TagText
    , dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) as VidCasts
FROM vc_Tag
ORDER BY VidCasts DESC
```

```
/*
```

```
5. ALter View cc_MostProlificUser and add a column TotalMinutes
```

```
*/
```

```
ALTER VIEW vc_MostProlificUser AS
```

```
SELECT TOP 10
```

```
/*
```

```
    , dbo.vc_VidCastDurational(vc_UserID) as TotalMinutes
```

```
    , dbo.vc_VidCastCount(vc_UserID) as VidCastCount
```

```
FROM vc_User
```

```
ORDER BY VidCastCount DESC
```

```
GO
```

```
SELECT UserName, VidCastCount, TotalMinutes FROM vc_MostProlificUser
```

```
/*
```

```
6. Coding Stored Procedure
```

```
*/
```

```
/*
```

```
Creat a stored procedure to addd a new TAG to the database
```

```
INPUTS
```

```
    @tagText : text of a 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
```

```
    DECLARE @tagID int
```

```
    SELECT @tagID = vc_TagID FROM vc_Tag
```

```
    WHERE TagText = @tagText
```

```
    INSERT INTO vc_AddTag(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
```



```
SELECT * FROM vc_VidCast WHERE vc_VidCastID = @newVidCastID
```

```
/*
```

## 7. Coding Stored Procedure called vc\_FinishVidCast

```
*/
```

```
CREATE PROCEDURE vc_FinishVidCast(@vidCastID int, @newStatusID int)
AS
BEGIN
    DECLARE @currentDateTime datetime = GETDATE()
    SELECT @currentDateTime = EndDateTime FROM vc_VidCast
    UPDATE vc_VidCast SET EndDateTime = @currentDateTime
    WHERE vc_VidCastID = @vidCastID

END
GO
```

```
--query--
```

```
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
```