Ryan Timbrook

IST 659 Data Admin Concepts &Db Mgmt
Date: 9/3/2018
Lab Assignment: Lab 8, Database Programming

Description / Learning Objective

- Demonstrate proficiency in coding and using SQL Server database objects such as Functions, Views, and Stored Procedures

Responses

Part 1, Functions

P1-FUNCTIONS-TODO-1: Declaring a variable

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. | (local) (14.0 RTM) | GSM1900\RTimbro1 (51) | master | 00:00:00 | 2 rows

P1-FUNCTIONS-TODO-2: Code a function that counts the number of VidCasts made by a given user and returns the count to the calling code.

Questions:

- Describe what the code snippet lines 49-53 does.
  - The 'SELECT TOP 10' clause tells the sql server to return the first 10 records from the vc_User table ordered by the condition specified in the 'Order BY' clause, which in this case is the VidCastCount column listed in descending order.
  - The second clause uses our custom function to take as input the vc_UserID's for each record and alias that column output as 'VidCastCount'
- How does it know that the vc_User record with vc_UserID = 20 has 22 vc_VidCast records?
  - By the WHERE clause in the dbo_vc_VidCastCount function and the COUNT function on the vc_UserID input attribute that's set as the return value for our custom function.

P1- FUNCTIONS -TODO-2: Screen Print:

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@placerataugueSed.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	ornare.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

P1- FUNCTIONS -TODO-3: `Code a function that accepts the tag text as a parameter and looks up the vc_tagID for the vc_tag record for that TagText`

Questions:

- Describe what lines 75 and 76 do.
    - These lines use the SELECT clause to execute our custom vc_TagIDLookup function passing a string text value into the function and displaying the results from that query.
- When line 76 executed, why did we receive a NULL value from SQL Server?
    - The value, 'Tunes', that was passed into the function as our input text was not found in the vc_Tag table as a variable value of the TagText attribute.

P1- FUNCTIONS -TODO-3: Screen Print:

Results | Messages

	(No column name)
1	4

	(No column name)
1	NULL

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (56) | IST659_V2 | 00:00:00 | 2 rows

Part 1, Views

P1-VIEWS-TODO-1: `Create a view to retrieve the top 10 vc_Users and their VidCast counts`

Questions:

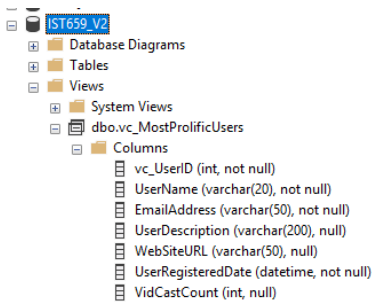
- Describe what lines 79 through 87 do.
    - The code in this snippet creates a VIEW object on the SQL Server which acts like a Table object that simplifies external queries. The internal mechanics of the VIEW utilizes a custom function written earlier that returns a filtered data set of the top 10 VidCastCount records of the vc_VidCast table in descending order.

P1-VIEWS-TODO-1: Screen Print

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.urna@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@placeratauguesed.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@elelfendnec.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	ornare.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

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (52) | IST659_V2 | 00:00:00 | 10 rows



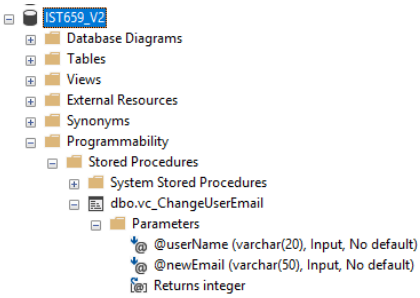
Part 1, Stored Procedures

P1-STORED_PROCEDURES-TODO-1: Create a procedure to update a vc_User's email address

Questions:

- Describe what lines 91 through 104 of assignments doc are doing.
  - This snippet of code creates an executable Stored Procedure on the SQL Server. The procedure is used to Update the vc_User table records based on the input parameters passed into it. Specifically, it will update the EmailAddress value in records where the userName value passed into the procedure matches a record's userName value in the vc_User table.

P1-STORED_PROCEDURES-TODO-1: Screen Prints



Messages

```
(1 row affected)
```

100 %

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (52) | IST659_V2 | 00:00:00 | 0 rows

Results | Messages

	vc_UserID	UserName	EmailAddress	UserDescription	WebSiteURL	UserRegisteredDate
1	6	tardy	kmstudent@syr.edu	Startup leverage growth hacking bootstrapping sc...	http://tardy.vidcast659.site	2017-03-12 15:36:00.000

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (52) | IST659_V2 | 00:00:00 | 1 rows

P1-STORED_PROCEDURES-TODO-2: @@identity property exercises

Questions:

- Explain why the UserLoginTimestamp is different than what's displayed in the screen print on the assignment form.
    - This field is a datetime datatype with a default function, getdate() defined on it.
        - When a new record is added to the vc_UserLogin table this UserLoginTimestamp field will auto set to the SQL servers system time.
        - The time represented will always be the date time of when we execute the stored procedure.
- How could we simplify the code in the stored procedure?
    - ???????? Looks pretty simple, how to make it more simple it the question?

P1-STORED_PROCEDURES-TODO-2: Screen Prints

Results | Messages

	vc_TagID	TagText	TagDescription
1	15	Cat Videos	NULL

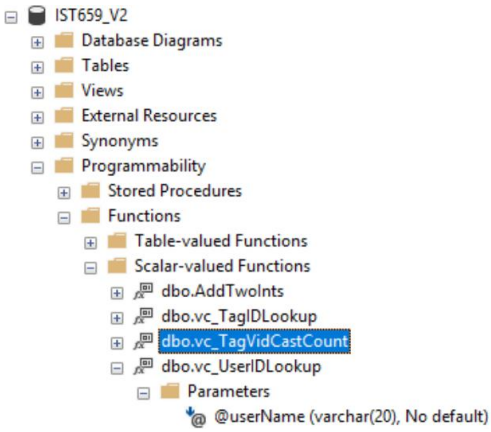
Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (52) | IST659_V2 | 00:00:00 | 1 rows

- IST659_V2
  - Database Diagrams
  - Tables
  - Views
  - External Resources
  - Synonyms
  - Programmability
    - Stored Procedures
      - System Stored Procedures
      - dbo.vc_AddUserLogin
        - Parameters
          - @userName (varchar(20), Input, No default)
          - @loginFrom (varchar(50), Input, No default)
          - Returns integer

Results | Messages

	vc_UserID	UserName	UserLoginTimestamp	LoginLocation
1	6	tardy	2018-08-29 18:51:23.767	localhost

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (52) | IST659_V2 | 00:00:00 | 1 rows

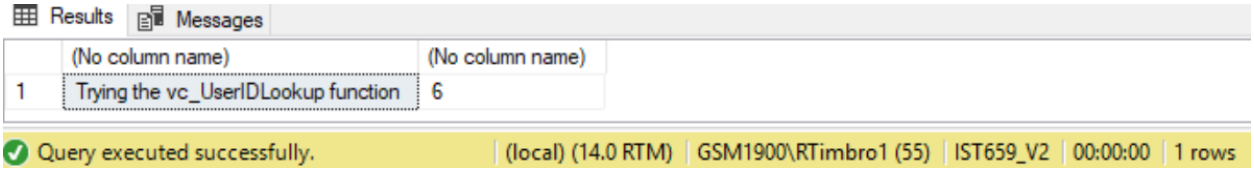


Part 2, Putting All Together

P2-TODO-USER_DEFINED_FUNCTIONS-1: dbo.vc_UserIDLookup

Complete the code to assign the correct vc_UserID to @returnValue

P2-TODO-USER_DEFINED_FUNCTIONS-1: Screen Prints



P2-TODO-USER_DEFINED_FUNCTIONS-2: dbo.vc_TagVidCastCount

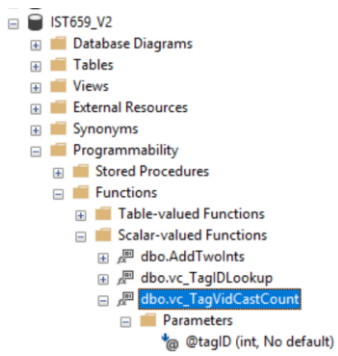
Create a function that calculates the count of vc_VidCastIDs for a given vc_TagID

P2-TODO-USER_DEFINED_FUNCTIONS-2: Screen Prints

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

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (55) | IST659_V2 | 00:00:00 | 15 rows



P2-TODO-USER_DEFINED_FUNCTIONS-3: dbo.vc_VidCastDuration

Create a function that SUMs the total number of minutes of actual duration for VidCasts with a Finished status given a vc_UserID as a parameter.

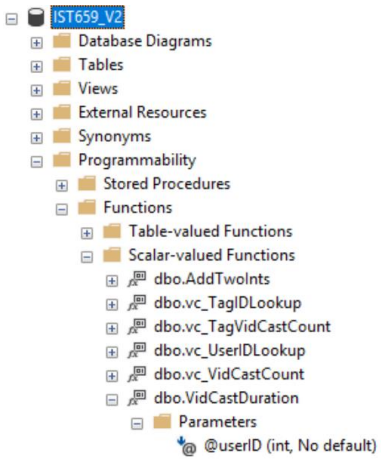
Returns the SUM as an int.

P2-TODO-USER_DEFINED_FUNCTIONS-3: Screen Prints

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	1859
2	2	dispatcher	quam@aptenttacitisociosqu.ca	A/B testing handshake disruptive seed money info...	http://dispatcher.vidcast659.site	2017-12-08 03:36:00.000	1368
3	3	camel	mauris@massanon.edu	User experience founders branding entrepreneur it...	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 ...	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 ...	http://hygienist.vidcast659.site	2017-03-17 23:16:48.000	1139
6	6	tardy	kmstudent@syr.edu	Startup leverage growth hacking bootstrapping scr...	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 c...	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 conversio...	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...	http://doughnut.vidcast659.site	2017-01-31 01:12:00.000	1830
12	12	bewildered	Donec.porttitor.tellus@odioAliquamvulp...	Infrastructure research & development venture bur...	http://bewildered.vidcast659.s...	2017-12-29 09:07:12.000	1944
13	13	albite	nisi@vitaemauris.org	Learning curve partnership buzz value proposition ...	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 iPhon...	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 n...	NULL	2017-10-07 12:43:12.000	1240

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (55) | IST659_V2 | 00:00:00 | 68 rows



P2-TODO-VIEWS-1: Create VIEW vc_TagReport

Create a view that executes a select statement that returns a list of TagText and VidCasts count in descending order.

*Note: Had to use the OFFSET command in the ORDER BY clause to order the return data set in descending order.

P2-TODO-VIEWS-1: Screen Prints

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	Cat Videos	0

Query executed successf... | (local) (14.0 RTM) | GSM1900\RTimbro1 (59) | IST659_V2 | 00:00:00 | 15 rows

P2-TODO-VIEWS-2: Alter VIEW vc_MostProlificUsers

Alter the view vc_MostProlificUsers, add a column called TotalMinutes that calls the vc_VidCastDuration function.

P2-TODO-VIEWS-2: Screen Prints

Results | Messages

	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

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (59) | IST659_V2 | 00:00:00 | 10 rows

- IST659_V2
  - Database Diagrams
  - Tables
  - Views
    - System Views
    - dbo.vc_MostProlificUsers
      - Columns
        - vc_UserID (int, not null)
        - UserName (varchar(20), not null)
        - EmailAddress (varchar(50), not null)
        - UserDescription (varchar(200), null)
        - WebSiteURL (varchar(50), null)
        - UserRegisteredDate (datetime, not null)
        - VidCastCount (int, null)
        - TotalMinutes (int, null)

P2-TODO-STORED_PROCEDURE-1: Create procedure vc_AddTag

Create a stored procedure to use in adding a row to the vc_Tag table.

Inputs:

@tagText: the text of the new tag

@description: a brief description of the tag (nullable)

Returns:

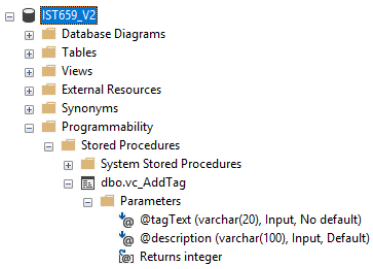
@@identity with the value inserted

P2-TODO- STORED_PROCEDURE -1: Screen Prints

Results | Messages

	vc_TagID	TagText	TagDescription
1	16	SQL	Finally, a SQL Tag

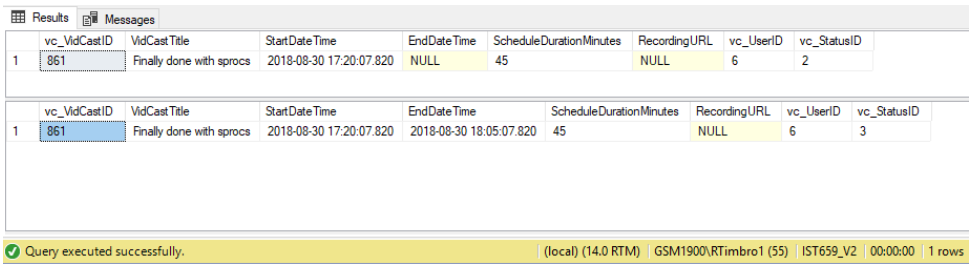
Query... | (local) (14.0 RTM) | GSM1900\RTimbro1 (55) | IST659_V2 | 00:00:00 | 1 rows



P2-TODO- STORED_PROCEDURE -2: vc_FinishVidCast

Create a stored procedure that accepts an int as a parameter that will be a vc_VidCastID that we need to mark as finished.

P2-TODO- STORED_PROCEDURE -2: Screen Prints

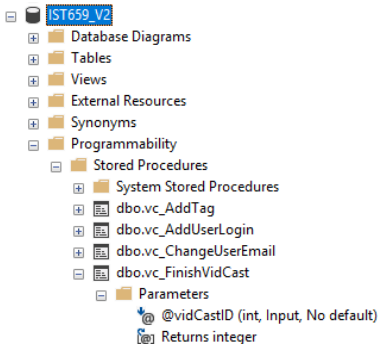


Results | Messages

	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	861	Finally done with sprocs	2018-08-30 17:20:07.820	NULL	45	NULL	6	2

	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	861	Finally done with sprocs	2018-08-30 17:20:07.820	2018-08-30 18:05:07.820	45	NULL	6	3

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (55) | IST659_V2 | 00:00:00 | 1 rows



Final SQL Script Submission:

```
/*
IST 659 Data Admin Concepts &Db Mgmt
Date: 9/3/2018
Lab Assignment: Lab 8, Database Programming

*/
-- Declare a variable, @ is mandatory at the beginning of the variable name in SQL
Server
declare @isThisNull varchar(30)  -- starts out as NULL
Select @isThisNull, ISNULL(@isThisNull,'Yep, it is NULL')

-- Set the variable to something
SET @isThisNull = 'Nope, It is not Null'
SELECT @isThisNull, ISNULL(@isThisNull, 'Yep, it is null')

GO
/*
First User Defined Function
```

```sql
*/
CREATE FUNCTION dbo.AddTwoInts(@firstNumber int, @secondNumber int)
RETURNS int AS -- AS is the keyword that ends the CREATE FUNCTION clause
BEGIN
	-- First, delcare the variable to temporarily hold the results
	DECLARE @returnValue int -- the data type matches the "RETURN" clause

	-- Set the variable to the correct value
	SET @returnValue = @firstNumber + @secondNumber

	-- Return the value to the calling statement
	RETURN @returnValue
END
GO

-- TODO: Execute SQL Select statement against new function
SELECT dbo.AddTwoInts(5,10)

/*
TODO: Code a function that counts the number of VidCasts made by a given user
	and returns the count to the calling code
*/
-- Drop Function if it Exists
GO
IF EXISTS
(SELECT * FROM sys.objects
WHERE object_id=OBJECT_ID(N'dbo.vc_VidCastCount')
AND type in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION dbo.vc_VidCastCount
GO

-- Function to count the VidCasts made by a given User
GO
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 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 dbo.vc_User
ORDER BY VidCastCount DESC
```

```sql
/*
TODO: Code a function that accepts the tag text as a parameter and looks up the
vc_tagID for the vc_tag record for that TagText
*/
-- Drop Function if it exists
IF EXISTS
(SELECT * FROM sys.objects
WHERE object_id=OBJECT_ID(N'dbo.vc_TagIDLookup')
AND type in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION dbo.vc_TagIDLookup
GO

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 as 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

-- Test dbo.vc_TagIDLookup function
SELECT dbo.vc_TagIDLookup('Music')
SELECT dbo.vc_TagIDLookup('Tunes')

/*
Views
*/
-- TODO:  Create a view to retrieve the top 10 vc_Users and their VidCast counts
-- Drop VIEW if it exists
IF EXISTS
(SELECT * FROM sys.views
	WHERE name = 'vc_MostProlificUsers' AND schema_id = SCHEMA_ID('dbo'))
DROP VIEW dbo.vc_MostProlificUsers

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

-- Test the vc_MostProlificUsers VIEW
SELECT * FROM vc_MostProlificUsers
```

```sql
/*
Stored Procedures
*/
-- TODO: 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
-- First DELETE the Stored Procedure if it exists
DROP PROCEDURE IF EXISTS vc_ChangeUserEmail
GO
CREATE PROCEDURE vc_ChangeUserEmail(@userName varchar(20), @newEmail varchar(50))
AS
BEGIN
       UPDATE vc_User SET EmailAddress = @newEmail
       WHERE userName = @userName
END
GO

-- Test the above Stored Procedure
EXEC vc_ChangeUserEmail 'tardy','kmstudent@syr.edu'

-- To see the effect from running the above execution of the vc_ChangeUserEmail Stored
Procedure
       -- run this snippet
SELECT * FROM vc_User WHERE UserName = 'tardy'

-- @@identity
-- TODO: Add a new record to the vc_Tag table and run a query using the @@identity
property
INSERT INTO vc_Tag(TagText) VALUES('Cat Videos')
SELECT * FROM vc_Tag WHERE vc_TagID = @@identity

-- TODO: Create a stored procedure to return the @@identity property
/*
       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 the're logging in.
*/
-- First Drop procedure if exists
DROP PROCEDURE IF EXISTS vc_AddUserLogin
GO
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)

       -- New return the @@identity so the calling code knows where the data end up
       RETURN @@identity
END
```

```sql
GO

-- Test the vc_AddUserLogin procedure by executing the below snippet
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

/*
HOW DO WE MAKE THE AddUserLogin PROCEDURE SIMPLER????
*/

/*
#### PART 2 - Putting All Together ####
*/

/*
       Create a function to retrieve a vc_UserID for a given user name
       P2-TODO-USER_DEFINED_FUNCTIONS-1: dbo.vc_UserIDLookup
*/
-- Drop Function if it exists
IF EXISTS
(SELECT * FROM sys.objects
WHERE object_id=OBJECT_ID(N'dbo.vc_UserIDLookup')
AND type in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION dbo.vc_UserIDLookup
GO
-- CREATE user-defined function
CREATE FUNCTION dbo.vc_UserIDLookup(@userName varchar(20))
RETURNS int AS
BEGIN
       DECLARE @returnValue int

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

       -- Return the vc_UserID found from the input attribute @userName
       RETURN @returnValue
END
GO

-- Test the user-defined function dbo.vc_UserIDLookup
SELECT 'Trying the vc_UserIDLookup function', dbo.vc_UserIDLookup('tardy')

/*
       Create a function that calculates the count of vc_VidCastIDs for a given
vc_TagID
       P2-TODO-USER_DEFINED_FUNCTIONS-2: dbo.vc_TagVidCastCount
*/
-- Drop Function if it exists
IF EXISTS
```

```sql
(SELECT * FROM sys.objects
WHERE object_id=OBJECT_ID(N'dbo.vc_TagVidCastCount')
AND type in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION dbo.vc_TagVidCastCount
GO
-- CREATE user-defined function
GO
CREATE FUNCTION dbo.vc_TagVidCastCount(@tagID int)
RETURNS int AS
BEGIN
	DECLARE @returnValue int

	-- Count vc_VidCastIDs for a given tagID from the vc_VidCastTagList table
	SELECT @returnValue = COUNT(vc_VidCastID) FROM vc_VidCastTagList
	WHERE vc_VidCastTagList.vc_TagID = @tagID

	-- Retrun the count of vc_VidCastIDs
	RETURN @returnValue
END
GO

-- Test the user-defined function dbo.vc_TagVidCastCount
SELECT vc_Tag.TagText, dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) as VidCasts
FROM vc_Tag

/*
	Create a function that SUMs the total number of minutes of actual duration for
VidCasts
		with a Finished status given a vc_UserID as a parameter.
		-> Returns the SUM as an int
	P2-TODO-USER_DEFINED_FUNCTIONS-3: dbo.vc_VidCastDuration
*/
-- Drop Function if it exists
IF EXISTS
(SELECT * FROM sys.objects
WHERE object_id=OBJECT_ID(N'dbo.vc_VidCastDuration')
AND type in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION dbo.vc_VidCastDuration
GO
-- CREATE user-defined function
CREATE FUNCTION dbo.VidCastDuration(@userID int)
RETURNS int AS
BEGIN
	DECLARE @returnValue int

	--
	SELECT @returnValue = SUM(DATEDIFF(n,vc_VidCast.StartDateTime,
vc_VidCast.EndDateTime))
	FROM vc_VidCast
	JOIN vc_Status on vc_Status.vc_StatusID = vc_VidCast.vc_StatusID
	WHERE vc_VidCast.vc_UserID = @userID
	AND vc_Status.StatusText = 'Finished'

	-- Return the SUM of duration of VidCasts
	RETURN @returnValue
END
GO
```

```sql
-- UNIT TEST: REMOVE BELOW SNIPPET
SELECT SUM(DATEDIFF(n,vc_VidCast.StartDateTime, vc_VidCast.EndDateTime)) as
TotalMinutes, vc_Status.StatusText
FROM vc_VidCast
JOIN vc_Status on vc_Status.vc_StatusID = vc_VidCast.vc_StatusID
WHERE vc_VidCast.vc_UserID = 1
AND vc_Status.StatusText = 'Finished'
GROUP BY
       vc_Status.StatusText
-- REMOVE ABOVE SNIPPET

-- Test the user-defined function dbo.vc_VidCastDuration
SELECT *, dbo.VidCastDuration(vc_UserID) as TotalMinutes
FROM vc_User

/*
       CODING YOUR OWN VIEWS
       -- Create a view that executes a SELECT statement
       P2-TODO-VIEWS-1: vc_TagReport
*/
-- Drop VIEW if it exists
IF EXISTS
(SELECT * FROM sys.views
       WHERE name = 'vc_TagReport' AND schema_id = SCHEMA_ID('dbo'))
DROP VIEW dbo.vc_TagReport
-- Create vc_TagReport VIEW
GO
CREATE VIEW vc_TagReport AS
       SELECT
              vc_Tag.TagText,
              dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) as VidCasts
       FROM vc_Tag
       ORDER BY VidCasts DESC OFFSET 0 ROWS -- Use the OFFSET command set to 0
otherwise SQL server won't allow the Order By Clause
GO

-- Test the vc_TagReport VIEW
SELECT * FROM vc_TagReport


/*
       CODING YOUR OWN VIEWS
       -- Alter the view vc_MostProlificUsers, add a column called totalMinutes that
calls the
              vc_VidCastDuration function
       -- P2-TODO-VIEWS-2: vc_MostProlificUsers
*/
GO
ALTER VIEW vc_MostProlificUsers AS
       SELECT TOP 10
              *
              , dbo.vc_VidCastCount(vc_UserID) as VidCastCount
              , dbo.VidCastDuration(vc_UserID) as TotalMinutes
       FROM vc_User
       ORDER BY VidCastCount DESC
GO

-- Test the vc_MostProlificUsers view
```

```sql
SELECT UserName, VidCastCount, TotalMinutes FROM vc_MostProlificUsers

/*
	CODING YOUR OWN STORED PROCEDURES
	-- Create a stored procedure to use in adding a row to the vc_Tag table.
	Inputs:
		@tagText: the text of the new tag
		@description: a brief description of the tag (nullable)
	Returns:
		@@identity with the value inserted
	-- P2-TODO-STORED_PROCEDURE-1: vc_AddTag
*/
-- First DELETE the Stored Procedure if it exists
DROP PROCEDURE IF EXISTS vc_AddTag

-- Create the vc_AddTag procedure
GO
CREATE PROCEDURE vc_AddTag(@tagText varchar(20), @description varchar(100)=NULL) AS
BEGIN
	-- Code the insert procedures here
	INSERT INTO vc_Tag (vc_Tag.TagText,vc_Tag.TagDescription)
	VALUES (@tagText, @description)

	-- Return the @@identity property of the newley inserted record
	RETURN @@identity
END
GO

-- Test the vc_AddTag stored procedure
DECLARE @newTagID int
EXEC @newTagID = vc_AddTag 'SQL', 'Finally, a SQL Tag'
SELECT * FROM vc_Tag WHERE vc_TagID = @newTagID


/*
	CODING YOUR OWN STORED PROCEDURES
	-- Create a stored procedure that accepts an int as a parameter that will be a
		vc_VidCastID that we need to mark as finished.
	-- P2-TODO-STORED_PROCEDURE-2: vc_FinishVidCast
*/
-- First DELETE the Stored Procedure if it exists
DROP PROCEDURE IF EXISTS vc_FinishVidCast

-- Create Stored Procedure
GO
CREATE PROCEDURE vc_FinishVidCast(@vidCastID int) AS
BEGIN
	-- Update VidCast EndDateTime
	UPDATE vc_VidCast
	SET vc_VidCast.EndDateTime = GETDATE(),
	vc_VidCast.vc_StatusID = (SELECT vc_StatusID FROM vc_Status WHERE
vc_Status.StatusText = 'Finished')
	WHERE vc_VidCast.vc_VidCastID = @vidCastID

END
GO

-- Test the vc_FinishVidCast stored procedure
```

```sql
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
```