# Ryan Timbrook

## IST 659 Data Admin Concepts &Db Mgmt

## Date: 9/3/2018

## Lab Assignment: Lab 8, Database Programming

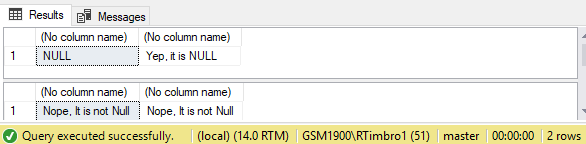
## Description / Learning Objective

* 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

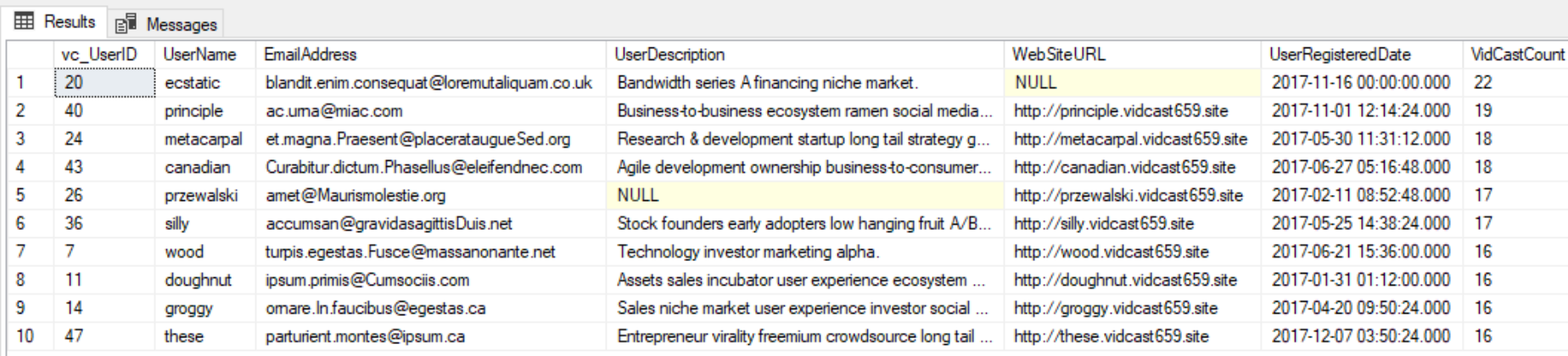


P1-FUNCTIONS-TODO-2: Code a function that counts the number of VidCasts made by a given user and returns the count to the calling code.

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:

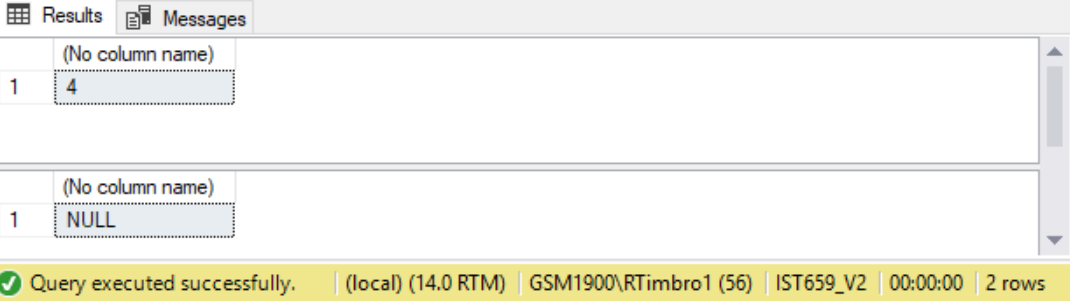


P1- FUNCTIONS -TODO-3: Code a function that accepts the tag text as a parameter and looks up the vc\_tagID for the vc\_tag record for that TagText

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:



### Part 1, Views

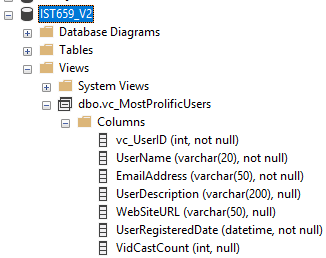
#### P1-VIEWS-TODO-1: Create a view to retrieve the top 10 vc\_Users and their VidCast counts

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





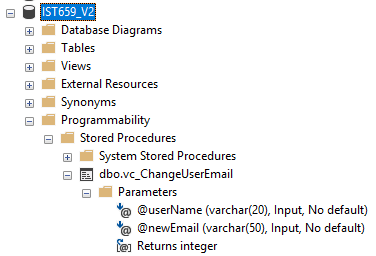
### Part 1, Stored Procedures

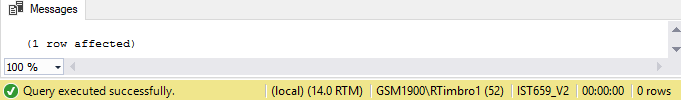
P1-STORED\_PROCEDURES-TODO-1: Create a procedure to update a vc\_User’s email address

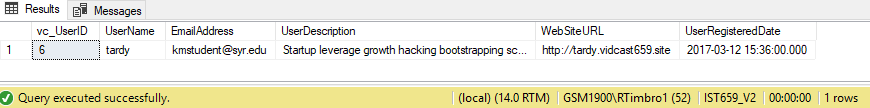
Questions:

* 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





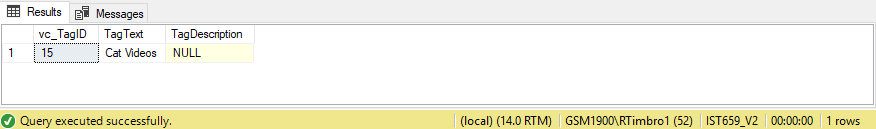


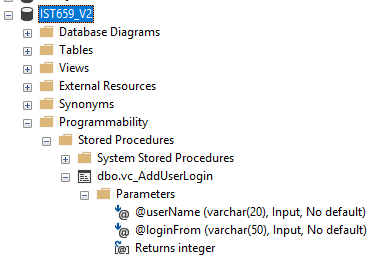
P1-STORED\_PROCEDURES-TODO-2: @@identity property exercises

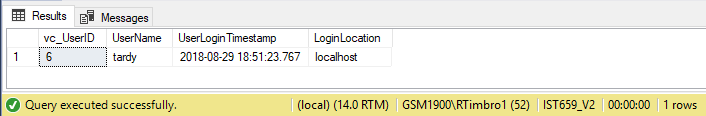
Questions:

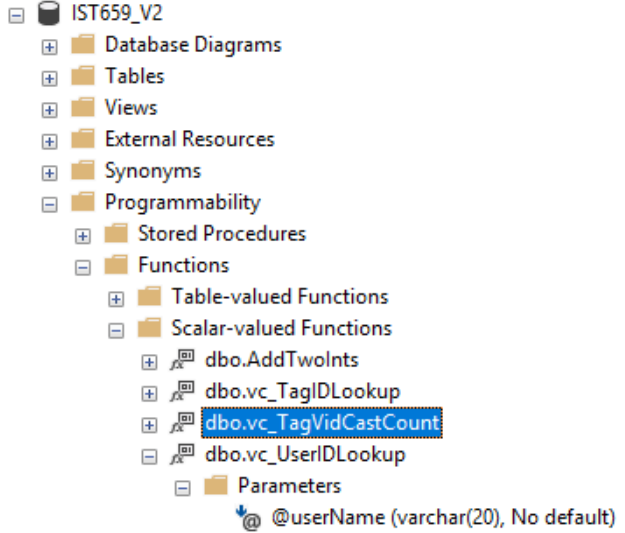
* Explain why the UserLoginTimestamp is different than what’s displayed in the screen print on the assignment form.
  + This field is a datetime datatype with a default function, getdate() defined on it.
    - When a new record is added to the vc\_UserLogin table this UserLoginTimestamp field will auto 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







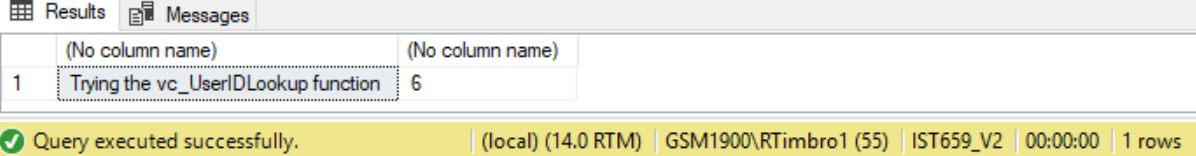


### Part 2, Putting All Together

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

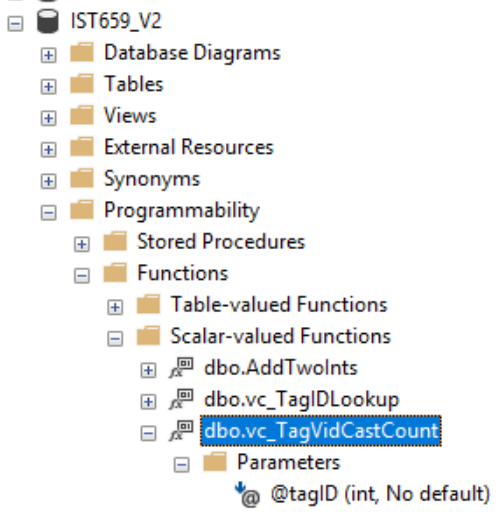


P2-TODO-USER\_DEFINED\_FUNCTIONS-2: dbo.vc\_TagVidCastCount

Create a function that calculates the count of vc\_VidCastIDs for a given vc\_TagID

P2-TODO-USER\_DEFINED\_FUNCTIONS-2: Screen Prints



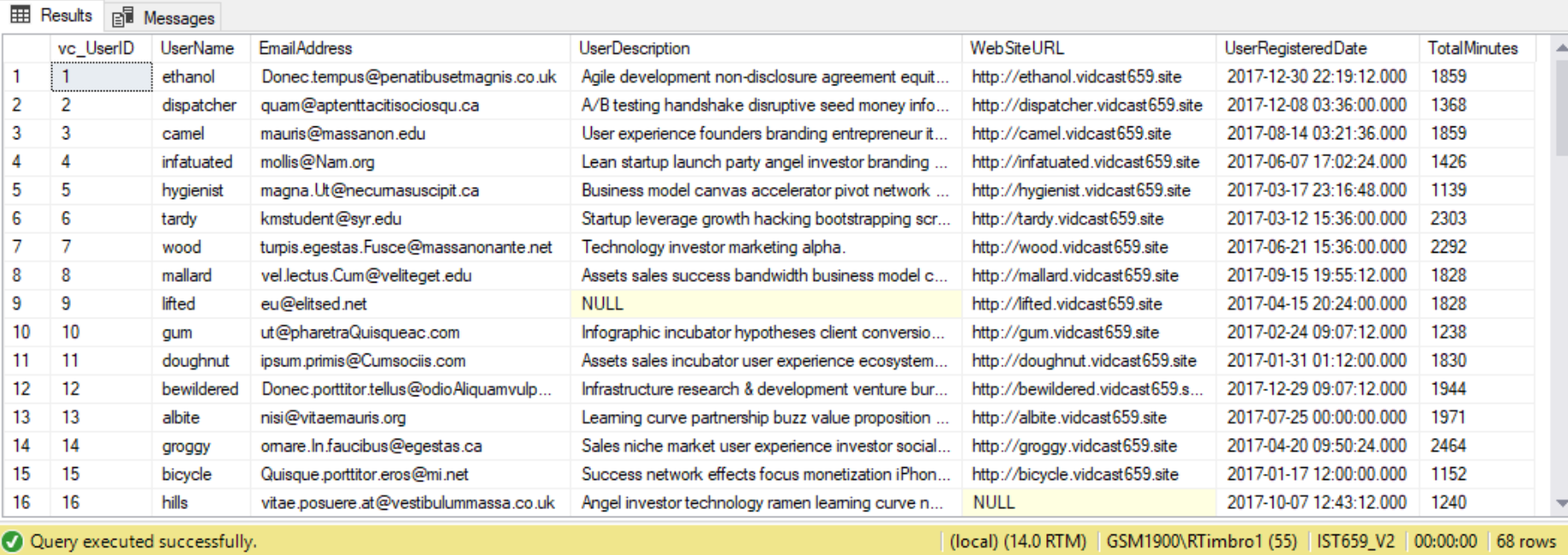


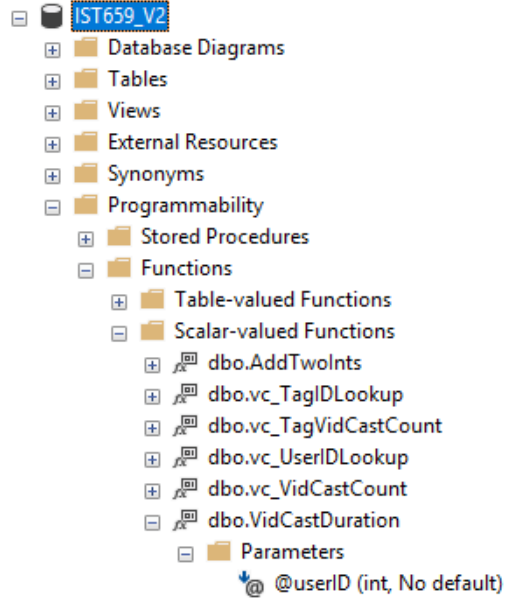
P2-TODO-USER\_DEFINED\_FUNCTIONS-3: dbo.vc\_VidCastDuration

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



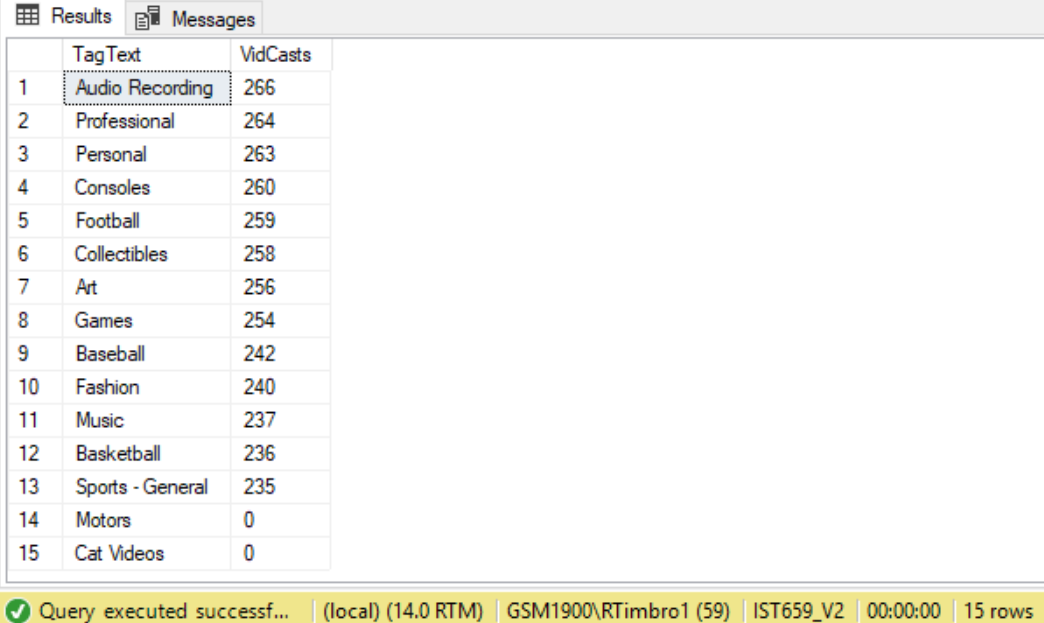


P2-TODO-VIEWS-1: Create VIEW vc\_TagReport

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.

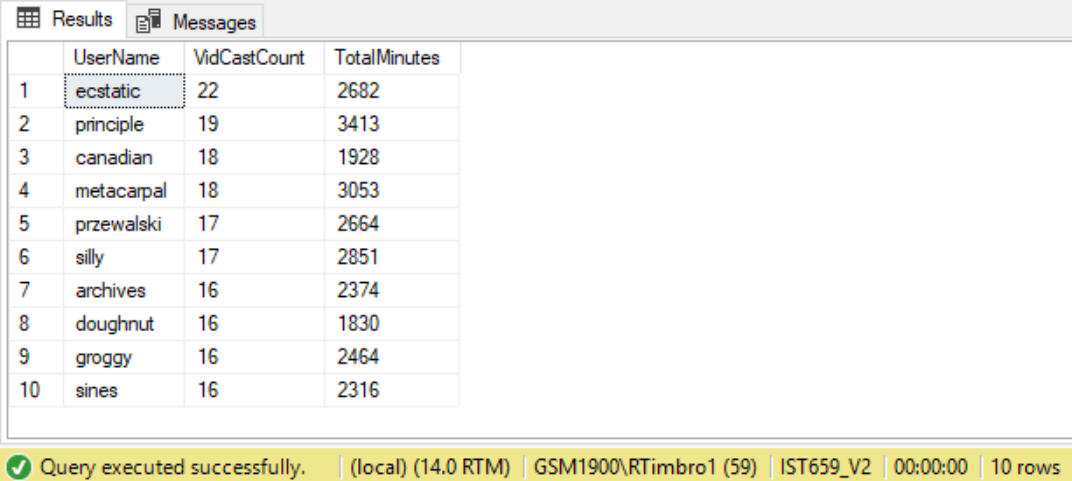
P2-TODO-VIEWS-1: Screen Prints

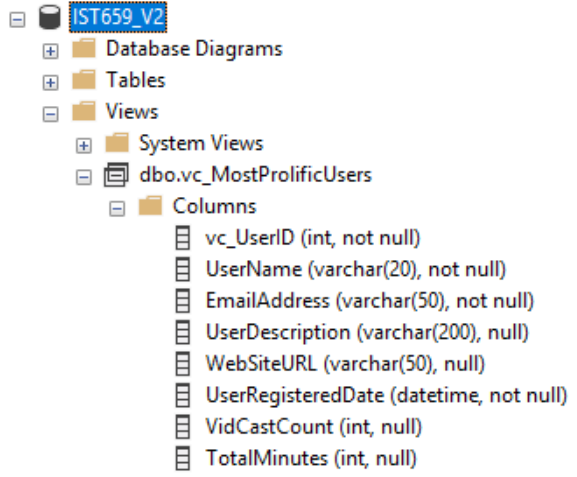


P2-TODO-VIEWS-2: Alter VIEW vc\_MostProlificUsers

Alter the view vc\_MostProlificUsers, add a column called TotalMinutes that calls the vc\_VidCastDuration function.

P2-TODO-VIEWS-2: Screen Prints





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:

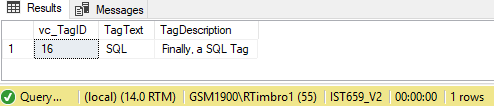
@tagText: the text of the new tag

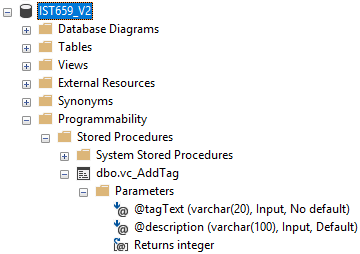
@description: a brief description of the tag (nullable)

Returns:

@@identity with the value inserted

P2-TODO- STORED\_PROCEDURE -1: Screen Prints

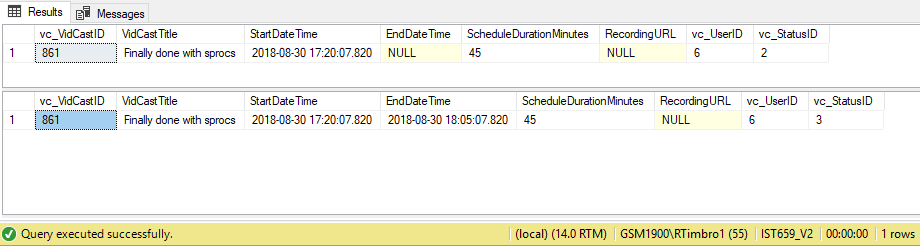


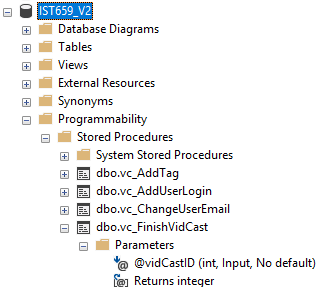


P2-TODO- STORED\_PROCEDURE -2: vc\_FinishVidCast

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





### 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  \*/  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  /\*  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  /\*  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  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  (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  -- 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  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  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 |