|  |  |  |
| --- | --- | --- |
| **Name** | : | Sathish Kumar Rajendiran |
| **Current Date** | : | 03/07/2020 |
| **Lab Number** | : | 08 |

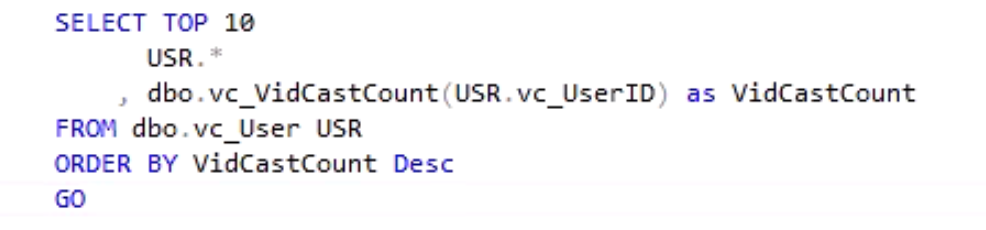
**Part 1 – Introducing Functions, Views, and Stored Procedures**

* ***Variable declaration***

**A screenshot of a cell phone

Description automatically generated**

* ***Top 10 Users by number of VidCasts posted***

****

**A screenshot of a cell phone

Description automatically generated**

**Explanation:**

vc\_VidCastCount - Function we have created gets the count of VidCasts for the provided UserID and assign that return value to variable returnvalue. In addition, UserID parameter in the WHERE clause restricts to that user's VidCast records. In this example, UserID 20 belongs to User “Ecstatic” has posted 22 VidCasts that value is aggregated by in the user defined function. So, the values are counted for respective users.

* ***Performing Data lookups using Functions***

**A screenshot of a cell phone

Description automatically generated**

A screenshot of a cell phone

Description automatically generated

**Explanation:**

vc\_TagIDLookup - Function looks for input value TagText from vc\_Tag and return the respective TagID. In this case, as there were no entries for “Tunes”, the lookup did not return TagID; hence, NULL value as result.

* ***Most Prolific Users - View***

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

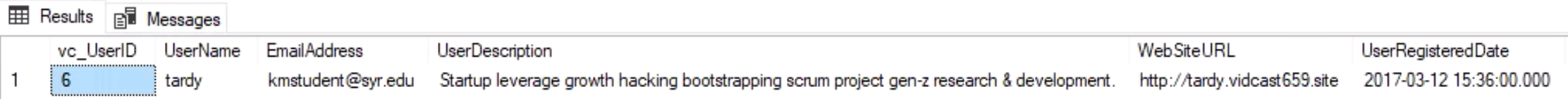
**Explanation:**

Vc\_MostProlificUsers - View is just a pre-packaged SQL Statement that calls vc\_VidCastCount function internally to compute the count of VidCasts per user. In addition, the result of this function is then wrapped in this View as a select statement and displays in descending order. Views generally doesn’t support Order by Clauses except it involves TOP or OFFSET clause in its definition. In this case, Top by clause is present in the Select Clause.

* ***Stored Procedure to Update User Email***

***A close up of a logo

Description automatically generated***

******

**A screenshot of a social media post

Description automatically generated**

**Explanation:**

vc\_ChangeUserEmail – is Stored procedure same as function but can also perform Insert/Update/delete operations. In this case, the Stored procedure accepts two parameters; username of the person whose email needs an update. User, tardy’s email is updated as ‘kmstudent@syr.edu’ because of this functionality.

* ***Stored Procedure to add Login location***

A screenshot of a cell phone

Description automatically generated

**Explanation:**

Yes, we can simplify the procedure vc\_AddUserLogin in couple of ways

* + - Lookup vc\_UserID dynamically from vc\_Users table for the respective input parameter @userName
    - We can also create a user defined function to pull the UserID and call it in the Insert Statement.

A screenshot of a social media post

Description automatically generated

**Part 2– Putting all Together**

* ***Function – vc\_UserIDLookup***

**A screenshot of a cell phone

Description automatically generated**

* ***Function – vc\_TagVidCastCount***

**A screenshot of a cell phone

Description automatically generated**

* ***Function – vc\_VidCastDuration***

**A screenshot of a cell phone

Description automatically generated**

* ***View – vc\_TagReport***

**A screenshot of a cell phone

Description automatically generated**

* ***View – Vc\_MostProlificUsers***

**A screenshot of a cell phone

Description automatically generated**

* ***Stored Procedure – vc\_AddTag***

**A screenshot of a cell phone

Description automatically generated**

* ***Stored Procedure – vc\_FinishVidCast***

**A screenshot of a cell phone

Description automatically generated**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SQL Script – find below\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

USE [IST659\_M406\_srajendi]

GO

/\*

Title : Lab 08 - SQL Scripts

Author : Sathish Kumar Rajendiran

Course : IST659 M406

Term : March 2020

\*/

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Introduction to Variables

example:

Declare a variable and return values respective to the parameter values

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

DECLARE @isThisNull varchar(30) -- Starts out as NULL

SELECT

@isThisNull as Value\_of\_Variable

, ISNULL(@isThisNull, 'Yep, It is NULL') as Is\_Result\_NULL

---- Set the variable to something other than NULL

SET @isThisNull = 'Nope. It is not NULL'

SELECT

@isThisNull as Value\_of\_Variable

, ISNULL(@isThisNull, 'Nope. It is not NULLl') as Is\_Result\_NULL

GO

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Introduction to User Defined Functions

example:

Define a scalar function to Add 2 Integers

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

----Function to Add 2 Integers

----Define Function

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='AddtwoInts')

BEGIN

DROP FUNCTION AddtwoInts

END

GO

CREATE FUNCTION

dbo.AddtwoInts(@firstnumber int, @secondnumbr int)

RETURNS Int

AS

BEGIN

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

DECLARE @returnValue int -- data type matches RETURN Value

SET @returnValue=@firstnumber+@secondnumbr

RETURN @returnValue

END

GO

----Validate the function

SELECT dbo.AddTwoInts(5, 10) as Sum\_of\_2\_Numbers

GO

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Lab 8 - "Function"

To-Do:

1. Function to count the VidCasts made by a given User

2. Function to retrieve the vc\_TagID for a given tag's text

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

-----1. Function to count the VidCasts made by a given User

-----Define function to Count the VidCasts made by a given User

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_VidCastCount')

BEGIN

DROP FUNCTION vc\_VidCastCount

END

GO

CREATE FUNCTION

dbo.vc\_VidCastCount(@userid int)

RETURNS Int

AS

BEGIN

DECLARE @returnValue int -- data type matches RETURN Value

/\*

Get the count of VidCasts for the provided UserID and assign that return value to variable @returnvalue.

Note that we use @userid parameter in the WHERE clause to limit our count to that user's VidCast records

\*/

Select @returnValue= COUNT(VC.vc\_UserID) FROM vc\_VidCast VC

Where VC.vc\_UserID = @userid

--Return @returnvalue to the calling code

RETURN @returnValue

END

GO

----Validate the function

SELECT TOP 10

USR.\*

, dbo.vc\_VidCastCount(USR.vc\_UserID) as VidCastCount

FROM dbo.vc\_User USR

ORDER BY VidCastCount Desc

GO

-----2. Function to retrieve the vc\_TagID for a given tag's text

-----Define Function to retrieve the vc\_TagID for a given tag's text

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_TagIDLookup')

BEGIN

DROP FUNCTION vc\_TagIDLookup

END

GO

CREATE FUNCTION

dbo.vc\_TagIDLookup(@tagText varchar(20))

RETURNS Int

AS

BEGIN

DECLARE @returnValue int -- data type matches RETURN Value

/\*

Get the vc\_TagID of the vc\_Tag record whose tagText matches the provided parameter and assign that return value to variable @returnvalue.

\*/

Select @returnValue= vc\_TagID FROM vc\_Tag T

Where T.TagText = @tagText

--Return @returnvalue to the calling code

RETURN @returnValue

END

GO

----Validate the function

SELECT dbo.vc\_TagIDLookup ('Music') as vc\_TagID

SELECT dbo.vc\_TagIDLookup ('Tunes') as vc\_TagID

GO

----View all records from vc\_Tag to validate the above function results

Select \* FROM vc\_Tag

Where TagText in ('Music','Tunes')

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Lab 8 - "Views"

To Do:

1. Create a View to retrieve top 10 vc\_Users and thier VidCast counts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

-----1. Create a View to retrieve top 10 vc\_Users and thier VidCast counts

-----Define View

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.VIEWS WHERE TABLE\_NAME='vc\_MostProlificUsers')

BEGIN

DROP VIEW vc\_MostProlificUsers

END

GO

CREATE VIEW dbo.vc\_MostProlificUsers

AS

SELECT TOP 10

USR.\*

, dbo.vc\_VidCastCount(USR.vc\_UserID) as VidCastCount

FROM dbo.vc\_User USR

ORDER BY VidCastCount Desc

GO

----Retrive values from the vc\_MostProlificUsers View

SELECT \* FROM dbo.vc\_MostProlificUsers

GO

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Lab 8 - "Stored Procedures"

To Do:

1. Create a Procedure to update a vc\_User's email address by passing

two parameters

i.e Username and Email Address

2. @@identity - using this global variable in Stored Procedure

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

-----To Do 1. Create a Procedure to update a vc\_User's email address

-----Define Procedure

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_ChangeUserEmail')

BEGIN

DROP PROCEDURE vc\_ChangeUserEmail

END

GO

CREATE PROCEDURE dbo.vc\_ChangeUserEmail (@userName varchar(20),@newEmail varchar(50))

AS

BEGIN

UPDATE USR

SET USR.EmailAddress = @newEmail

FROM dbo.vc\_User USR

WHERE USR.UserName=@userName

END

----Execute vc\_ChangeUserEmail Stored procedure

EXEC dbo.vc\_ChangeUserEmail 'tardy','kmstudent@syr.edu'

GO

----Retrive values from the vc\_Users table

SELECT \* FROM dbo.vc\_User Where UserName ='tardy'

GO

-----To do 2. @@identity - using this global variable in Stored Procedure

IF (SELECT dbo.vc\_TagIDLookup ('Cat Videos')) IS NULL

BEGIN

INSERT INTO dbo.vc\_Tag(TagText) Values ('Cat Videos')

SELECT \* FROM vc\_Tag WHERE vc\_TagID = @@IDENTITY

END

SELECT \* FROM vc\_Tag WHERE TagText ='Cat Videos'

GO

/\*

To do 3. Create a procedure that adds row into UserLogin table

This procedure is run when a user logs in and we need to record

who they are and from where they're logged in.

\*/

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_AddUserLogin')

BEGIN

DROP PROCEDURE vc\_AddUserLogin

END

GO

CREATE PROCEDURE dbo.vc\_AddUserLogin (@userName varchar(20),@loginFrom varchar(50))

AS

BEGIN

--We have UserName but we need the ID for the Login table

--First, decare a variable to hold the ID

DECLARE @userID int

--Get the vc\_UserID for the UserName provided and store it in @userID

SELECT @userID = USR.vc\_UserID

FROM dbo.vc\_User USR

WHERE USR.UserName=@userName

--Now we can add the row using an INSERT Statement

INSERT INTO dbo.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

--select \* from dbo.vc\_UserLogin

--select \* from dbo.vc\_User

----Execute vc\_AddUserLogin Stored procedure

DECLARE @addedvalue int

EXEC @addedvalue = dbo.vc\_AddUserLogin 'tardy','localhost'

----Retrive values from the vc\_Users table

SELECT

USR.vc\_UserID

,USR.UserName

,UL.UserLoginTimestamp

,UL.LoginLocation

FROM dbo.vc\_User USR

JOIN dbo.vc\_UserLogin UL on USR.vc\_UserID=UL.vc\_UserID

Where UL.vc\_UserLoginID=@addedvalue

GO

-----Alternative version of the above stored procedure

-- define the procedure

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_AddUserLogin\_v1')

BEGIN

DROP PROCEDURE vc\_AddUserLogin\_v1

END

GO

CREATE PROCEDURE dbo.vc\_AddUserLogin\_v1 (@userName varchar(20),@loginFrom varchar(50))

AS

BEGIN

--We have UserName but we need the ID for the Login table

--Check if User is available in vc\_Users table then get the UserID dynamically. We might as well create a function for this.

--LoginLocation is part of the variable

--vc\_UserLoginID is an identity value; so it can be retrived by calling @@identity global variable

--LoginTimstamp has default value if its missing in the insert statement.

IF (SELECT vc\_UserID FROM dbo.vc\_User WHERE UserName=@userName) IS NOT NULL

BEGIN

INSERT INTO dbo.vc\_UserLogin(vc\_UserID,LoginLocation)

VALUES

((SELECT vc\_UserID FROM dbo.vc\_User WHERE UserName=@userName), @loginFrom)

END

--- Select the recently added value.

---if the existing Username doesnt not exist in the User table; then it returns the most recent value added in this session.

SELECT \* FROM dbo.vc\_UserLogin WHERE vc\_UserLoginID = @@IDENTITY

END

GO

----Execute vc\_AddUserLogin\_v1 Stored procedure

EXEC dbo.vc\_AddUserLogin\_v1 'camel8','localhost'

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Part 2 - "Function"

To-Do:

1. Function to retrive vc\_UserID for the given User

2. Function to calculate count of vc\_VidCastIDs for a given vc\_TagID

3. Function to calculate VidCasts Counts

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

----- 1.Function to retrieve the vc\_UserID for a given UserName

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_UserIDLookup')

BEGIN

DROP FUNCTION vc\_UserIDLookup

END

GO

CREATE FUNCTION

dbo.vc\_UserIDLookup(@userName varchar(20))

RETURNS Int

AS

BEGIN

DECLARE @returnValue int -- data type matches RETURN Value

/\*

Get the vc\_UserID of the vc\_User record whose userName matches the provided parameter and assign that return value to variable @returnvalue.

\*/

Select @returnValue= vc\_UserID FROM vc\_User U

Where U.UserName = @userName

--Return @returnvalue to the calling code

RETURN @returnValue

END

GO

----Validate the function

SELECT 'Trying the vc\_UserIDLookup function.' as Label, dbo.vc\_UserIDLookup('tardy') as vc\_UserID

SELECT 'Trying the vc\_UserIDLookup function.' as Label, dbo.vc\_UserIDLookup('sathish') as vc\_UserID

GO

-----2. Create a function that calculates the count of vc\_VidCastIDs for a given vc\_TagID

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_TagVidCastCount')

BEGIN

DROP FUNCTION vc\_TagVidCastCount

END

GO

CREATE FUNCTION

dbo.vc\_TagVidCastCount(@tagID int)

RETURNS Int

AS

BEGIN

DECLARE @returnValue int -- data type matches RETURN Value

/\*

Get the count of VidCasts for the provided tagID and assign that return value to variable @returnvalue.

Note that we use @@tagID parameter in the WHERE clause to limit our count to that user's vc\_Tags records

\*/

SELECT

@returnValue = Count(VC.vc\_VidCastID)

FROM vc\_Tag T

LEFT JOIN vc\_VidCastTagList VCT ON T.vc\_TagID=VCT.vc\_TagID

LEFT JOIN vc\_VidCast VC on VC.vc\_VidCastID=VCT.vc\_VidCastID

Where T.vc\_TagID = @tagID

Order by Count(VC.vc\_VidCastID) desc

--Return @returnvalue to the calling code

RETURN @returnValue

END

GO

----Validate the function

SELECT

TagText

, dbo.vc\_TagVidCastCount(vc\_TagID) as VidCasts

FROM dbo.vc\_Tag

-----3. Function to calculate the VidCast duration as a number of minutes for each individual vc\_VidCast record

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_VidCastDuration')

BEGIN

DROP FUNCTION vc\_VidCastDuration

END

GO

CREATE FUNCTION

dbo.vc\_VidCastDuration(@userID int)

RETURNS Int

AS

BEGIN

DECLARE @returnValue int -- data type matches RETURN Value

/\*

Calculate the count of vc\_VidCastIDs for a given vc\_TagID and assign that return value to variable @returnvalue.

Note that we use @@tagID parameter in the WHERE clause to limit our count

\*/

SELECT

@returnValue=Sum(DateDiff(n,StartDatetime,EndDateTime))

FROM dbo.vc\_VidCast as VidCast

Join dbo.vc\_User as USR ON USR.vc\_UserID=VidCast.vc\_UserID

Where USR.vc\_UserID = @userID

--Return @returnvalue to the calling code

RETURN @returnValue

END

GO

----Validate the function

SELECT \*

, dbo.vc\_VidCastDuration(vc\_UserID) as TotalMinutes

FROM dbo.vc\_User

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Part2 - "Views"

To Do:

1. Create a view called vc\_TagReport

2. Modify vc\_MostProlificUsers view

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

-----1. Create a view called vc\_TagReport

-----Define View

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.VIEWS WHERE TABLE\_NAME='vc\_TagReport')

BEGIN

DROP VIEW vc\_TagReport

END

GO

CREATE VIEW dbo.vc\_TagReport

AS

SELECT T.TagText

,dbo.vc\_TagVidCastCount(vc\_TagID) as VidCasts

FROM dbo.vc\_Tag T

GO

----Retrive values from the vc\_TagReport View

SELECT \* FROM dbo.vc\_TagReport

Order by VidCasts desc

GO

-----2. Alter the vc\_MostProlificUsers View to include TotalMinutes that calls the vc\_VidCastDuration function

-----Modify View

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.VIEWS WHERE TABLE\_NAME='vc\_MostProlificUsers')

BEGIN

EXECUTE('CREATE VIEW dbo.vc\_MostProlificUsers

AS

SELECT TOP 10

USR.\*

, dbo.vc\_VidCastCount(USR.vc\_UserID) as VidCastCount

, dbo.vc\_VidCastDuration(USR.vc\_UserID) as TotalMinutes

FROM dbo.vc\_User USR

ORDER BY VidCastCount Desc')

END

GO

ALTER VIEW dbo.vc\_MostProlificUsers

AS

SELECT TOP 10

USR.\*

, dbo.vc\_VidCastCount(USR.vc\_UserID) as VidCastCount

, dbo.vc\_VidCastDuration(USR.vc\_UserID) as TotalMinutes

FROM dbo.vc\_User USR

ORDER BY VidCastCount Desc

GO

----Retrive values from the vc\_MostProlificUsers View

SELECT UserName, VidCastCount, TotalMinutes

FROM vc\_MostProlificUsers

GO

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Part 2 - Stored Procedures

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\*

Create a 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

\*/

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_AddTag')

BEGIN

DROP PROCEDURE vc\_AddTag

END

GO

CREATE PROCEDURE dbo.vc\_AddTag (@tagText varchar(20),@description varchar(100)=NULL)

AS

IF NOT EXISTS( SELECT \* FROM dbo.vc\_Tag WHERE TagText = @tagText)

BEGIN

--Now we can add the row using an INSERT Statement

INSERT INTO dbo.vc\_Tag(TagText,TagDescription)

VALUES (@tagText,@description);

RETURN @@IDENTITY

END

ELSE

--BEGIN

-- SELECT \* FROM dbo.vc\_Tag Where TagText = @tagText

----Now return the @@identity so the calling code knows where the data ended up

--END

RETURN (SELECT vc\_TagID FROM dbo.vc\_Tag Where TagText = @tagText)

GO

----Execute vc\_AddTag Stored procedure

DECLARE @newTagID int

EXEC @newTagID = dbo.vc\_AddTag 'SQL','Finally, a SQL Tag'

----Retrive values from the vc\_Tag table

SELECT

\*

FROM dbo.vc\_Tag

Where vc\_TagID = @newTagID

GO

/\*

Create a procedure that accepts vc\_VidCastID and update its status as "Finished"; meaning change its EndDateTime to be the current Date and Time

Input:

@vidCastID : int parameter

Outputs:

vc\_StatusID --> to have the respective StatusID for "Finished"Status

EndDateTime --> Getdate()

Returns: @@identity with the value inserted

\*/

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.ROUTINES WHERE ROUTINE\_NAME='vc\_FinishVidCast')

BEGIN

DROP PROCEDURE vc\_FinishVidCast

END

GO

CREATE PROCEDURE dbo.vc\_FinishVidCast (@vidCastID int)

AS

BEGIN

UPDATE VC

SET

VC.vc\_StatusID = (SELECT vc\_StatusID FROM vc\_Status WHERE StatusText='Finished')

,VC. EndDateTime = GETDATE()

FROM dbo.vc\_VidCast VC

Where VC.vc\_VidCastID= @vidCastID

--Now return the @@identity so the calling code knows where the data ended up

RETURN @@identity

END

GO

----Execute vc\_FinishVidCast Stored procedure

DECLARE @newVC int

INSERT INTO dbo.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