Nick Videtti
IST-659 Week 8 Lab Assignment

PART 1:

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

This SELECT statement selects the top 10 Users from the vc_User table based on the number of VidCasts they have made. It also orders them by number of VidCasts made, in descending order. The reason that this code knows that the vc_User record with vc_UserID = 20 has 22 vc_VidCast records is because we pass the vc_UserID into the vc_VidCastCount function that we created, which takes a user a returns that number of VidCasts that user has created.

```
SELECT dbo.vc_TagIDLookup('Music')
SELECT dbo.vc_TagIDLookup('Tunes')
```

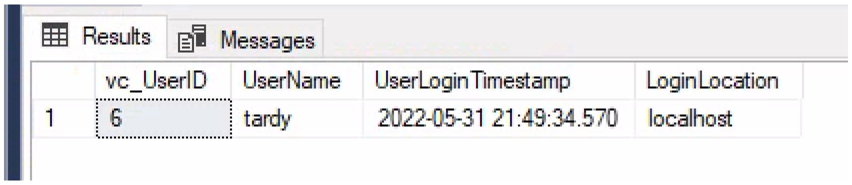
These statements above call the dbo.vc_TagIDLookup function that we have just created. The first line passes ‘Music’ into the function, returning the vc_TagID that matches the TagText that says ‘Music’. The second line passes ‘Tunes’ into the function, returning the vc_TagID that matches the TagText that says ‘Tunes’. We received a NULL when executing the second line because there is not a record in the vc_Tag table that has a TagText of ‘Tunes’, meaning that there is not a vc_TagID corresponding to a tag whose TagText is ‘Tunes.

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

This code above is creating a view called vc_MostProlificUsers that contains the top 10 records from the vc_User table based on the output from the dbo.VidCastCount function that we created earlier in the lab. The records are also sorted in descending order by the value returned from the dbo.VidCastCount function.

```
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', 'kmstudent@syr.edu'

SELECT * FROM vc_User WHERE UserName = 'tardy'
```

The code above creates a stored procedure called vc_ChangeUserEmail that takes two parameters, the first being the vc_Users UserName and the second being the email address. This procedure changes the passed-in vc_User's email address to the one that was passed in. Then, we execute the procedure by setting the vc_User's email address with the UserName 'tardy' to 'kmstudent@syr.edu'. The final line queries the vc_User table for only the vc_User whose UserName is 'tardy' and shows us that the email address is indeed 'kmstudent@syr.edu'.



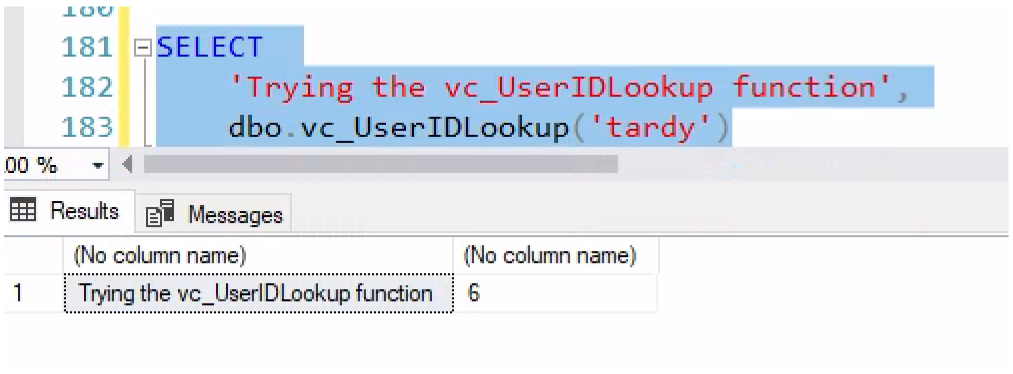
	vc_UserID	UserName	UserLoginTimestamp	LoginLocation
1	6	tardy	2022-05-31 21:49:34.570	localhost

The reason that this UserLogin Timestamp is different from the lab document is that this column is set to be the current system timestamp.

```
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
```

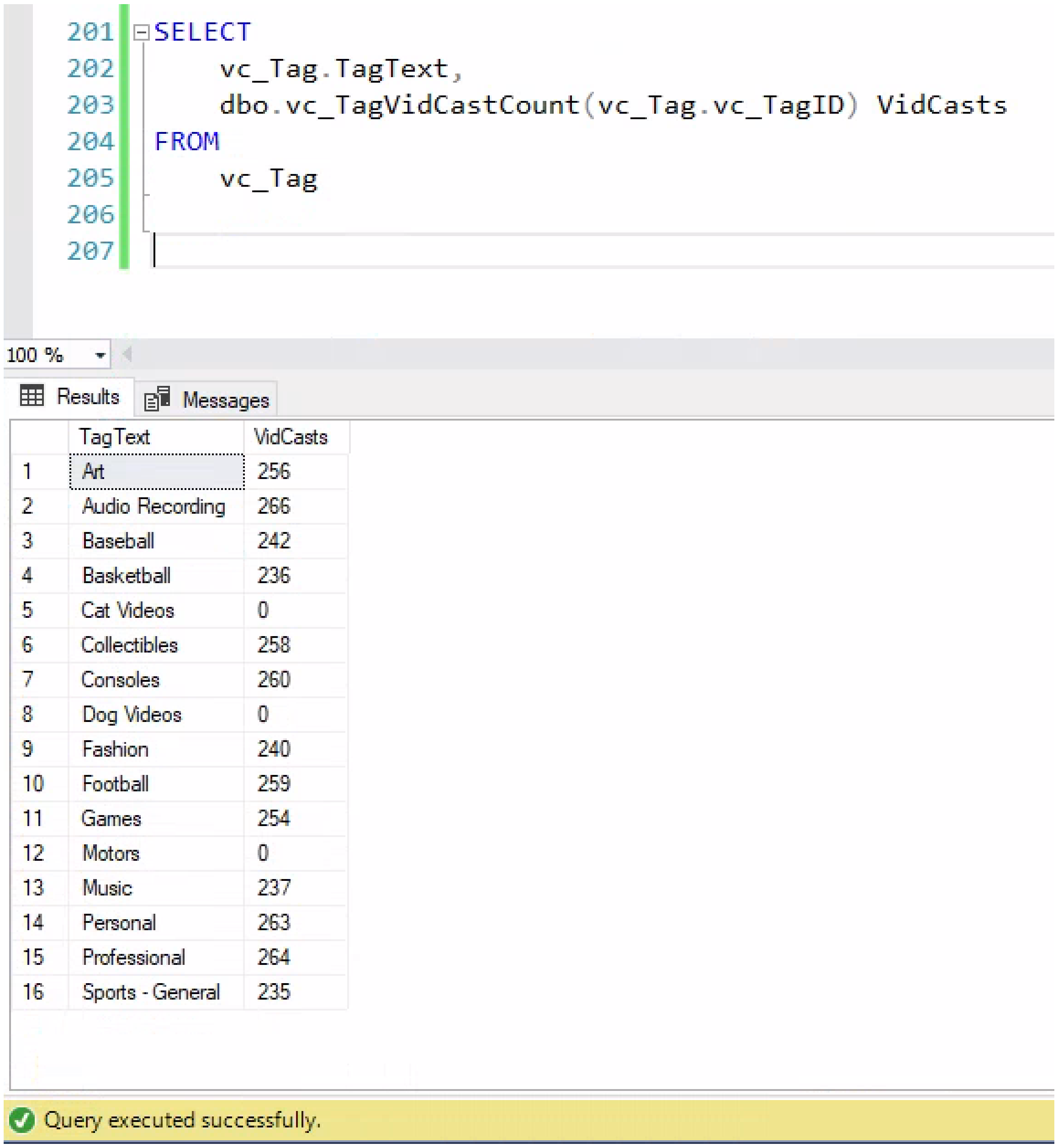
This code could be simplified by using the SCOPE_IDENTITY() function rather than declaring and using the addedValue variable

PART 2:



```
SELECT
    'Trying the vc_UserIDLookup function',
    dbo.vc_UserIDLookup('tardy')
```

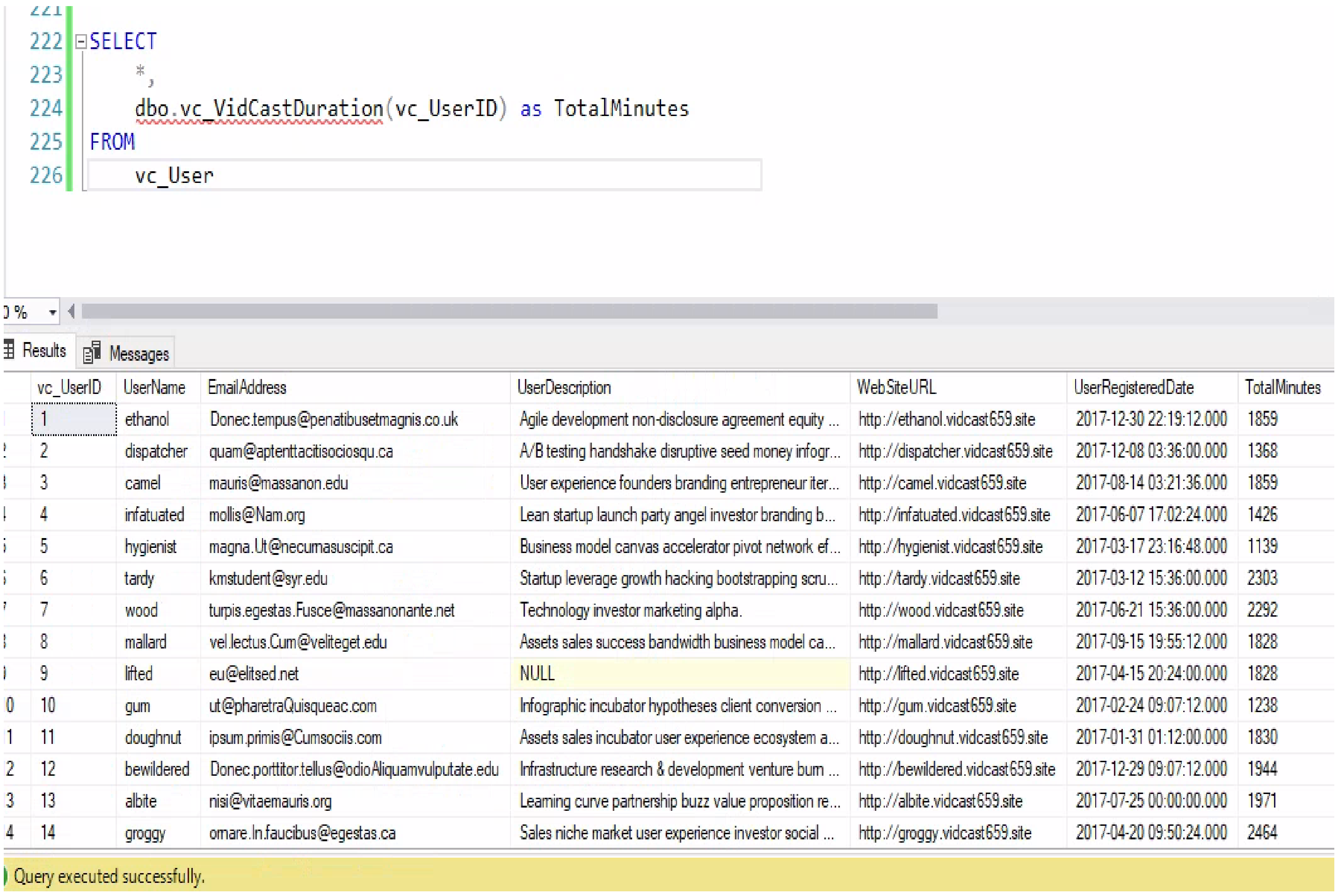
	(No column name)	(No column name)
1	Trying the vc_UserIDLookup function	6



```sql
SELECT
    vc_Tag.TagText,
    dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) VidCasts
FROM
    vc_Tag

```

	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	Dog Videos	0
9	Fashion	240
10	Football	259
11	Games	254
12	Motors	0
13	Music	237
14	Personal	263
15	Professional	264
16	Sports - General	235

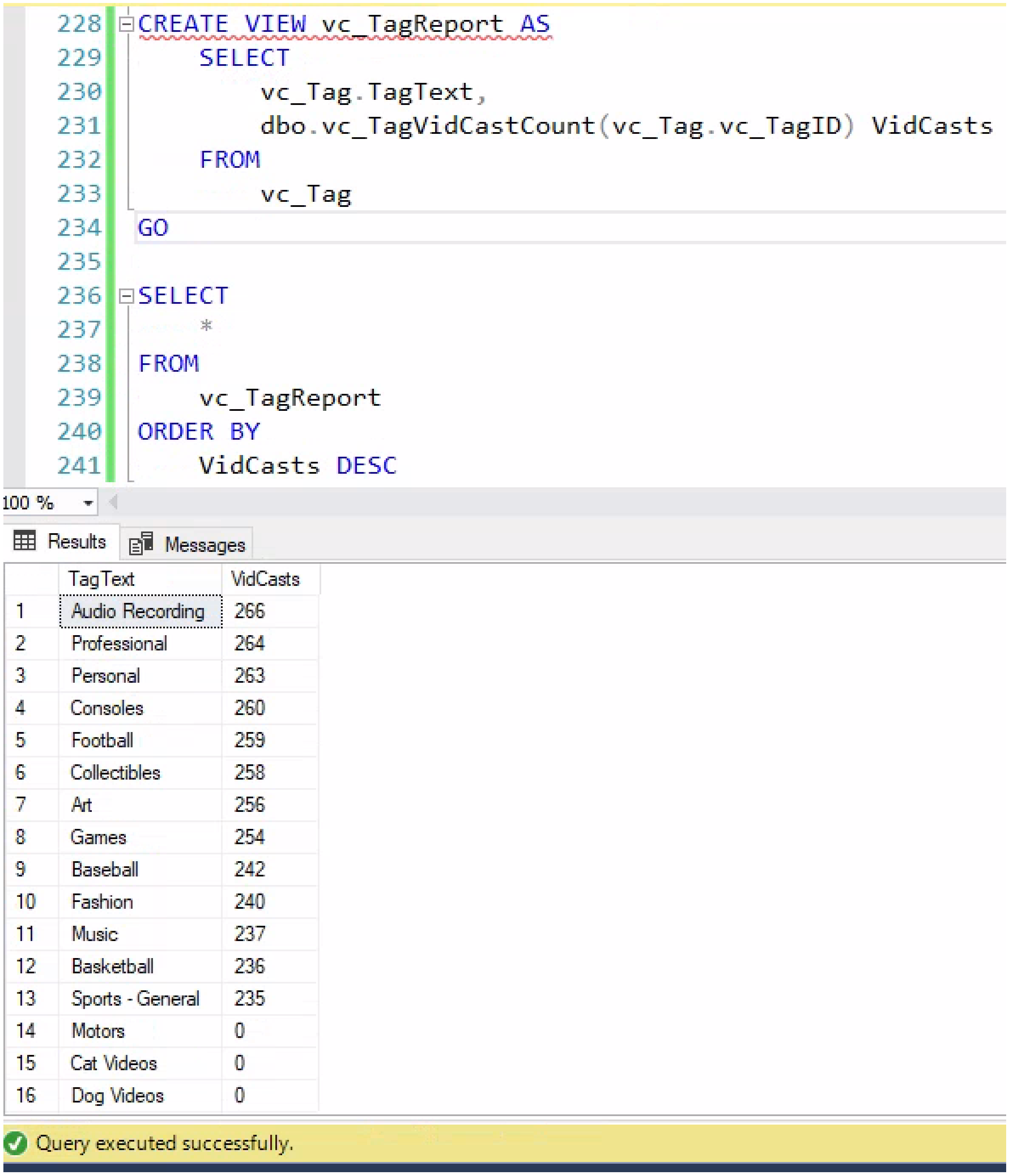
Query executed successfully.



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

vc_UserID	UserName	EmailAddress	UserDescription	WebSiteURL	UserRegisteredDate	TotalMinutes
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	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	camel	mauris@massanon.edu	User experience founders branding entrepreneur iter...	http://camel.vidcast659.site	2017-08-14 03:21:36.000	1859
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	hygienist	magna.Ut@necurnasuscipit.ca	Business model canvas accelerator pivot network ef...	http://hygienist.vidcast659.site	2017-03-17 23:16:48.000	1139
6	tardy	kmstudent@syr.edu	Startup leverage growth hacking bootstrapping scru...	http://tardy.vidcast659.site	2017-03-12 15:36:00.000	2303
7	wood	turpis.egestas.Fusce@massanonante.net	Technology investor marketing alpha.	http://wood.vidcast659.site	2017-06-21 15:36:00.000	2292
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	lifted	eu@elitsed.net	NULL	http://lifted.vidcast659.site	2017-04-15 20:24:00.000	1828
10	gum	ut@pharetraQuisqueac.com	Infographic incubator hypotheses client conversion ...	http://gum.vidcast659.site	2017-02-24 09:07:12.000	1238
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	bewildered	Donec.porttitor.tellus@odioAliquamvulputate.edu	Infrastructure research & development venture burn ...	http://bewildered.vidcast659.site	2017-12-29 09:07:12.000	1944
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	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

Query executed successfully.



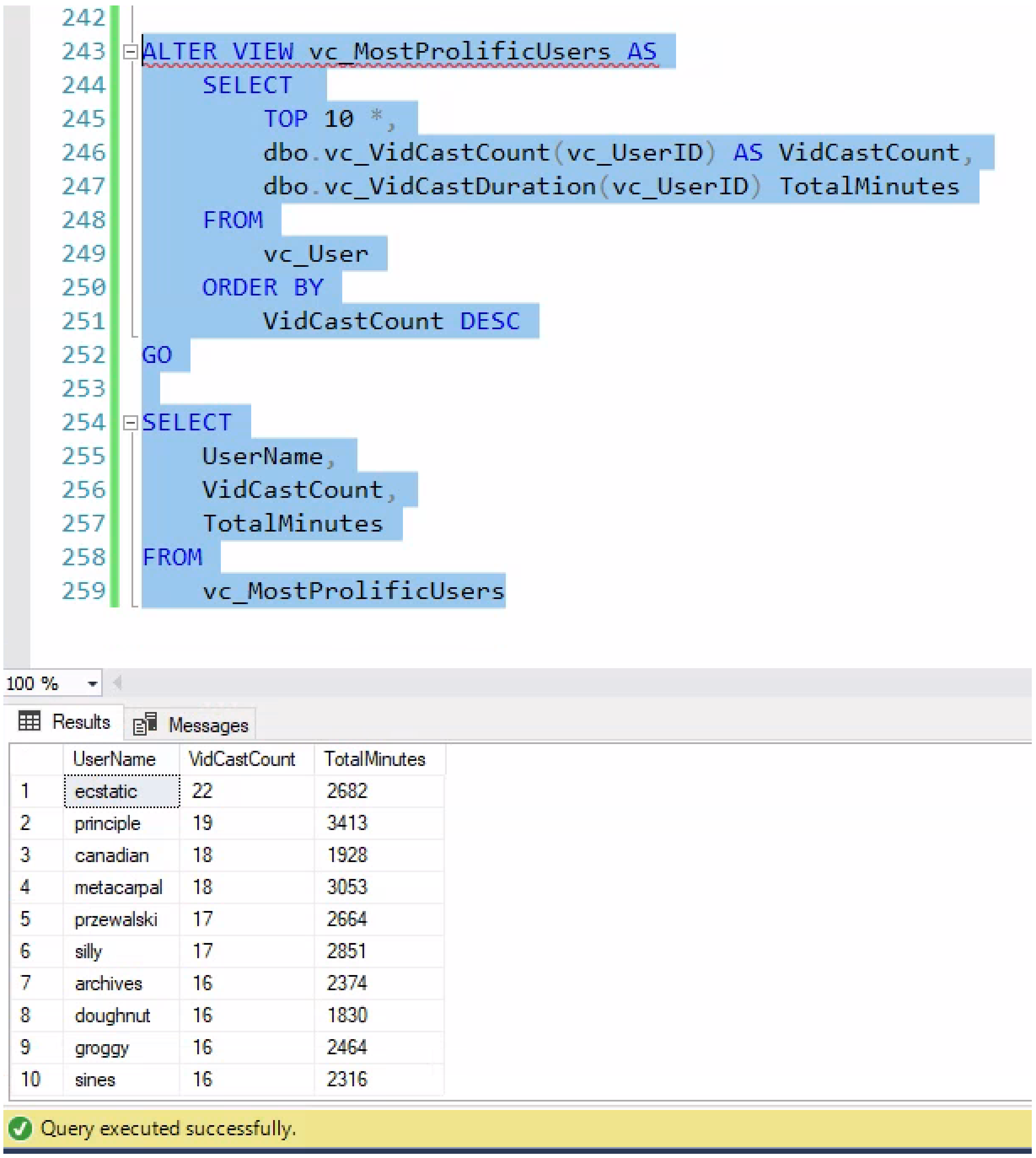
```sql
CREATE VIEW vc_TagReport AS
    SELECT
        vc_Tag.TagText,
        dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) VidCasts
    FROM
        vc_Tag
GO

SELECT
    *
FROM
    vc_TagReport
ORDER BY
    VidCasts DESC
```

100 %

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
16	Dog Videos	0

Query executed successfully.



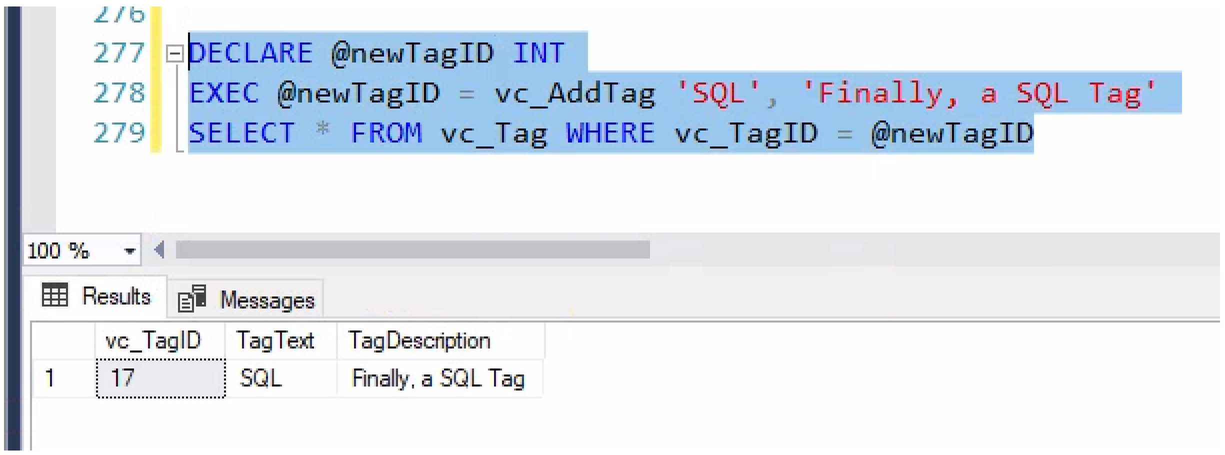
```
ALTER VIEW vc_MostProlificUsers AS
    SELECT
        TOP 10 *,
        dbo.vc_VidCastCount(vc_UserID) AS VidCastCount,
        dbo.vc_VidCastDuration(vc_UserID) TotalMinutes
    FROM
        vc_User
    ORDER BY
        VidCastCount DESC
GO

SELECT
    UserName,
    VidCastCount,
    TotalMinutes
FROM
    vc_MostProlificUsers
```

100 %

Results | Messages

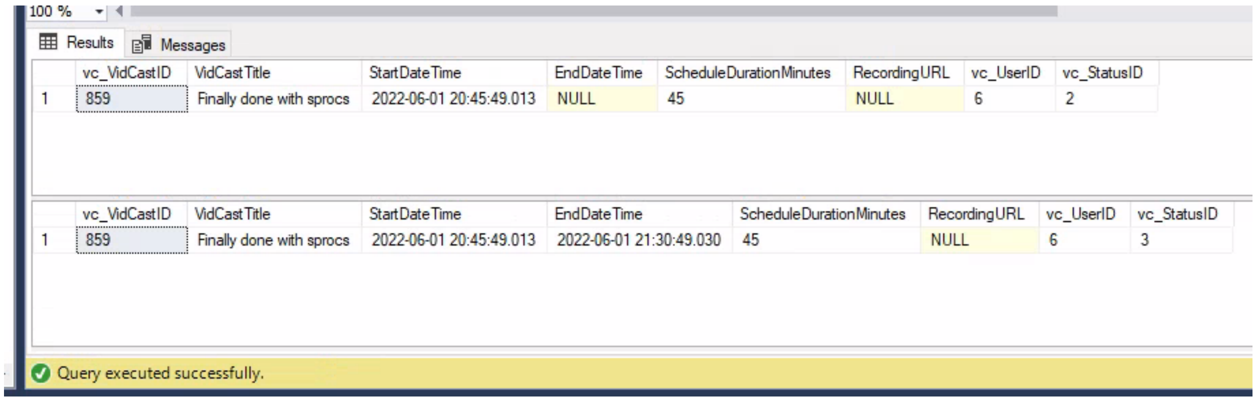
	UserName	VidCastCount	TotalMinutes
1	ecstatic	22	2682
2	principle	19	3413
3	canadian	18	1928
4	metacarpal	18	3053
5	przewalski	17	2664
6	silly	17	2851
7	archives	16	2374
8	doughnut	16	1830
9	groggy	16	2464
10	sines	16	2316

Query executed successfully.



```
DECLARE @newTagID INT
EXEC @newTagID = vc_AddTag 'SQL', 'Finally, a SQL Tag'
SELECT * FROM vc_Tag WHERE vc_TagID = @newTagID
```

	vc_TagID	TagText	TagDescription
1	17	SQL	Finally, a SQL Tag



	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	859	Finally done with sprocs	2022-06-01 20:45:49.013	NULL	45	NULL	6	2

	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	859	Finally done with sprocs	2022-06-01 20:45:49.013	2022-06-01 21:30:49.030	45	NULL	6	3

Query executed successfully.

SQL CODE:

/*
Author: Nick Videtti
Course : IST659 M407
Term : Spring (April - June) 2022
*/

----------LAB 8----------

-----PART 1-----

--Declare a variable
DECLARE @ISTHISNULL VARCHAR(30) --Starts out as null
SELECT @ISTHISNULL, ISNULL(@ISTHISNULL, 'Yep, it is null')

```
--Set the variable to something other than NULL
SET @ISTHISNULL = 'Nope. It is not NULL'
SELECT @ISTHISNULL, ISNULL(@ISTHISNULL, 'Yep, it is null')

CREATE FUNCTION dbo.AddTwoInts(@firstnumber INT, @secondnumber INT)
RETURNS INT AS
BEGIN
    --First, declare the variable to temporarily hold the result
    DECLARE @returnvalue INT --the data type matches the RETURNS clause

    --Do whatever needs to be done to se the variable to the correct value
    SET @returnvalue = @firstnumber + @secondnumber

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

SELECT dbo.AddTwoInts(5,10)

--Function to count the VidCasts made by a specific 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 function's return type

    /*Get the count of 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) 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 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

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_MostProlificUsers AS
	SELECT
		TOP 10 *,
		dbo.vc_VidCastCount(vc_UserID) AS VidCastCount
	FROM
		vc_User
	ORDER BY
```

```
                    VidCastCount DESC
GO

SELECT
          *
FROM
          vc_MostProlificUsers

--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', 'kmstudent@syr.edu'

SELECT * FROM vc_User WHERE UserName = 'tardy'

INSERT INTO vc_Tag (TagText) VALUES ('Cat Videos')
SELECT * FROM vc_Tag WHERE vc_TagID = @@IDENTITY

INSERT INTO vc_Tag (TagTExt) VALUES ('Dog Videos')
SELECT * FROM vc_Tag WHERE vc_TagID = SCOPE_IDENTITY()

/*Create a procedure that adds a row to the UserLogin table.
This procedure should be run when a user logs in.
It will record who they are and from where they're logging in.*/

CREATE OR ALTER PROCEDURE vc_AddUserLogin(@userName varchar(20),
@loginFrom varchar(50))
AS
BEGIN
          --We have the user name, but we need the user ID for the login table
          --First, declare a variable to hold the ID
```

```
	DECLARE @userID int

	SELECT @userID = vc_UserID FROM vc_User
	WHERE UserName = @userName

	--Now we can add the row using an INSERY statement
	INSERT INTO vc_UserLogin (vc_UserID, LoginLocation)
	VALUES(@userID, @loginFrom)

	--Lastly, return the SCOPE_IDENTITY() so the calling code knows the primary
key of the row we just added
	RETURN SCOPE_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-----

--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

SELECT
    'Trying the vc_UserIDLookup function',
    dbo.vc_UserIDLookup('tardy')

CREATE FUNCTION dbo.vc_TagVidCastCount(@vc_TagID INT)
RETURNS INT AS
BEGIN
    DECLARE @returnvalue INT

    SELECT
        @returnvalue = COUNT(DISTINCT vc_VidCastID)
    FROM
        vc_VidCastTagList
    WHERE
        vc_TagID = @vc_TagID

    RETURN @returnvalue
END
GO

SELECT
    vc_Tag.TagText,
    dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) VidCasts
FROM
    vc_Tag

CREATE FUNCTION vc_VidCastDuration(@vc_UserID INT)
RETURNS INT AS
BEGIN
    DECLARE @VCDUR INT
```

```
        SELECT
                @VCDUR = SUM(DATEDIFF(n,StartDateTime,EndDateTime))
        FROM
                vc_VidCast
        WHERE
                vc_StatusID = (SELECT vc_StatusID FROM vc_Status WHERE
StatusText = 'Finished')
                AND vc_UserID = @vc_UserID
        RETURN @VCDUR
END
GO

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

CREATE VIEW vc_TagReport AS
        SELECT
                vc_Tag.TagText,
                dbo.vc_TagVidCastCount(vc_Tag.vc_TagID) VidCasts
        FROM
                vc_Tag
GO

SELECT
        *
FROM
        vc_TagReport
ORDER BY
        VidCasts DESC

ALTER VIEW vc_MostProlificUsers AS
        SELECT
                TOP 10 *,
                dbo.vc_VidCastCount(vc_UserID) AS VidCastCount,
                dbo.vc_VidCastDuration(vc_UserID) TotalMinutes
        FROM
                vc_User
```

```
	ORDER BY
		VidCastCount DESC
GO

SELECT
	UserName,
	VidCastCount,
	TotalMinutes
FROM
	vc_MostProlificUsers

/*Create a stored 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*/

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

DECLARE @newTagID INT
EXEC @newTagID = vc_AddTag 'SQL', 'Finally, a SQL Tag'
SELECT * FROM vc_Tag WHERE vc_TagID = @newTagID

CREATE PROCEDURE vc_FinishVidCast(@VID_CAST_ID INT) AS
BEGIN
	UPDATE vc_VidCast
		SET
			EndDateTime = GETDATE(),
			vc_StatusID = (SELECT vc_StatusID FROM vc_Status WHERE
StatusText = 'Finished')
		WHERE
```

```
                              vc_VidCastID = @VID_CAST_ID
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 = SCOPE_IDENTITY()
SELECT * FROM vc_VidCast WHERE vc_VidCastID = @newVC
EXEC vc_FinishVidCast @newVC
SELECT * FROM vc_VidCast WHERE vc_VidCastID = @newVC
```