Ryan Timbrook

IST 659 Data Admin Concepts &Db Mgmt
Date: 9/10/2018
Lab Assignment: Lab 9, Data Security

Description / Learning Objective

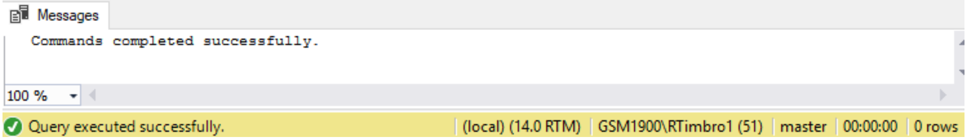
- Demonstrate proficiency in creating database users and administering to their user privileges on database objects
- Demonstrate proficiency in preserving data integrity using transactions

Responses

Part 1 - Securing Data Objects

P1-TODO-1: Creating a Database User

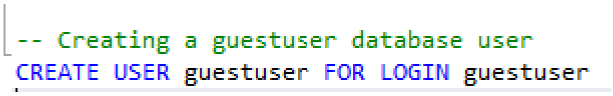
P1-TODO-1: Screen Prints



```
Commands completed successfully.
```

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (51) | master | 00:00:00 | 0 rows

P1-TODO-1: Code Snippet

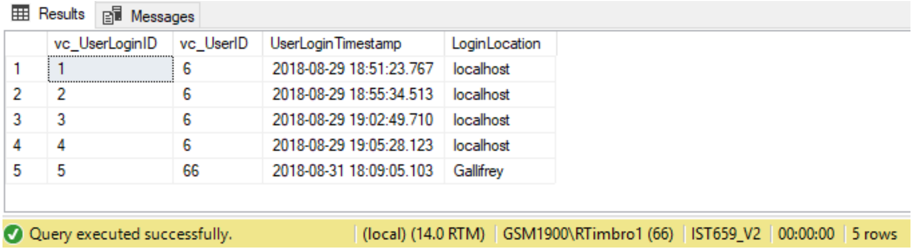


```
-- Creating a guestuser database user
CREATE USER guestuser FOR LOGIN guestuser
```

P1-TODO-2: Managing a User's Permissions

P1-TODO-2a: vc_UserLogin Table

P1-TODO-2a: Screen Prints

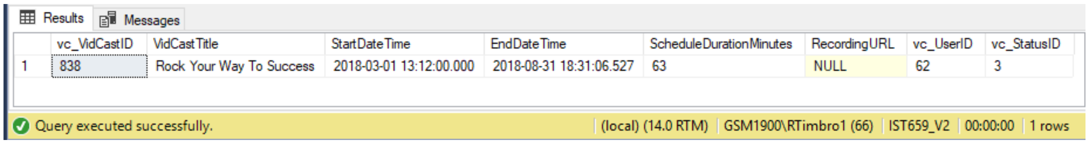


	vc_UserLoginID	vc_UserID	UserLoginTimestamp	LoginLocation
1	1	6	2018-08-29 18:51:23.767	localhost
2	2	6	2018-08-29 18:55:34.513	localhost
3	3	6	2018-08-29 19:02:49.710	localhost
4	4	6	2018-08-29 19:05:28.123	localhost
5	5	66	2018-08-31 18:09:05.103	Gallifrey

Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (66) | IST659_V2 | 00:00:00 | 5 rows

P1-TODO-2b: vc_VidCast records

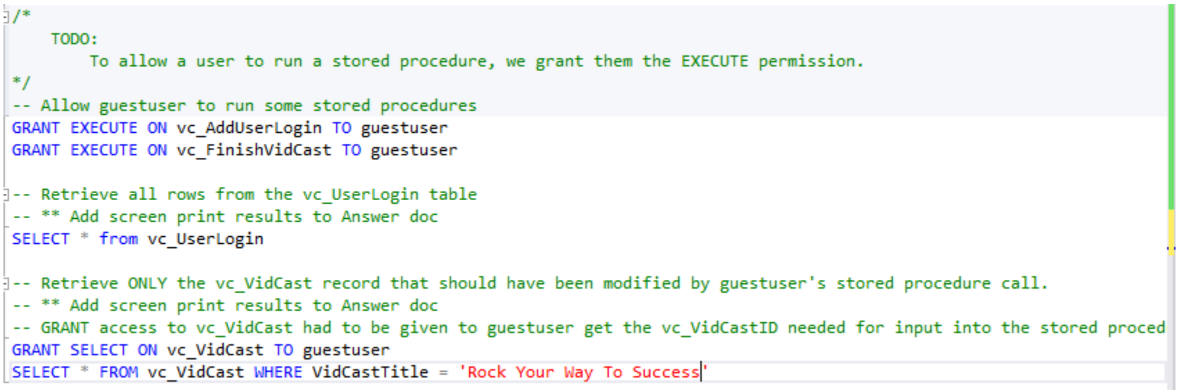
P1-TODO-2b: Screen Prints



	vc_VidCastID	VidCastTitle	StartDateTime	EndDateTime	ScheduleDurationMinutes	RecordingURL	vc_UserID	vc_StatusID
1	838	Rock Your Way To Success	2018-03-01 13:12:00.000	2018-08-31 18:31:06.527	63	NULL	62	3

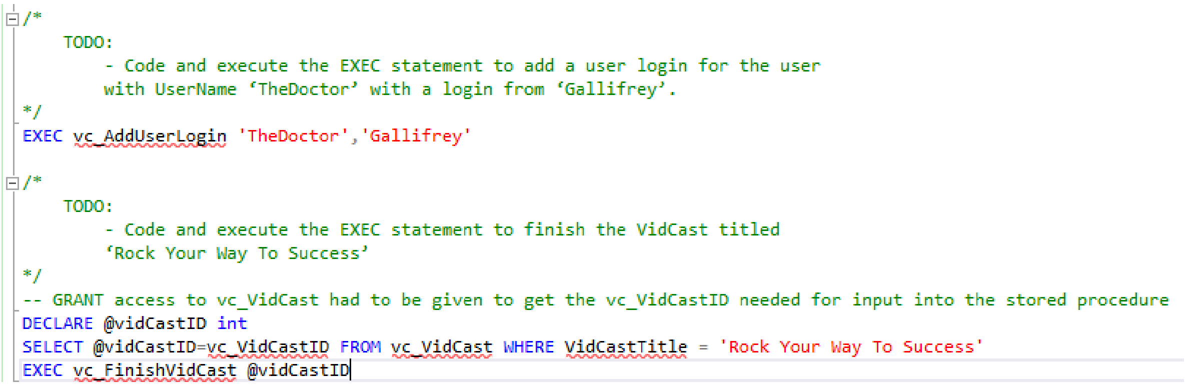
Query executed successfully. | (local) (14.0 RTM) | GSM1900\RTimbro1 (66) | IST659_V2 | 00:00:00 | 1 rows

P1-TODO-2: Code Snippets - my tab



```
/*
    TODO:
        To allow a user to run a stored procedure, we grant them the EXECUTE permission.
*/
-- Allow guestuser to run some stored procedures
GRANT EXECUTE ON vc_AddUserLogin TO guestuser
GRANT EXECUTE ON vc_FinishVidCast TO guestuser

-- Retrieve all rows from the vc_UserLogin table
-- ** Add screen print results to Answer doc
SELECT * from vc_UserLogin

-- Retrieve ONLY the vc_VidCast record that should have been modified by guestuser's stored procedure call.
-- ** Add screen print results to Answer doc
-- GRANT access to vc_VidCast had to be given to guestuser get the vc_VidCastID needed for input into the stored proced
GRANT SELECT ON vc_VidCast TO guestuser
SELECT * FROM vc_VidCast WHERE VidCastTitle = 'Rock Your Way To Success'
```

P1-TODO-2: Code Snippets – guestuser tab



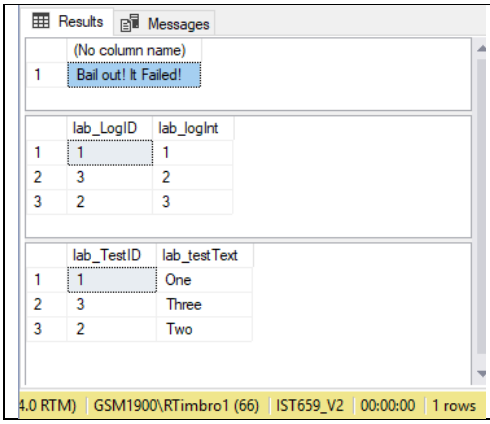
```
/*
    TODO:
        - Code and execute the EXEC statement to add a user login for the user
        with UserName ‘TheDoctor’ with a login from ‘Gallifrey’.
*/
EXEC vc_AddUserLogin 'TheDoctor','Gallifrey'

/*
    TODO:
        - Code and execute the EXEC statement to finish the VidCast titled
        ‘Rock Your Way To Success’
*/
-- GRANT access to vc_VidCast had to be given to get the vc_VidCastID needed for input into the stored procedure
DECLARE @vidCastID int
SELECT @vidCastID=vc_VidCastID FROM vc_VidCast WHERE VidCastTitle = 'Rock Your Way To Success'
EXEC vc_FinishVidCast @vidCastID
```

Part 2 – Data Integrity Through Transactions

P2-TODO-1:

P2-TODO-1: Screen Prints

First Run: With Expected Error	*Second Run: Without Error*

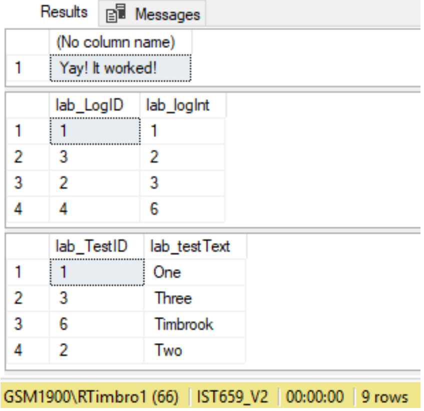


	(No column name)
1	Bail out! It Failed!

	lab_LogID	lab_logInt
1	1	1
2	3	2
3	2	3

	lab_TestID	lab_testText
1	1	One
2	3	Three
3	2	Two

4.0 RTM) | GSM1900\RTimbro1 (66) | IST659_V2 | 00:00:00 | 1 rows



	(No column name)
1	Yay! It worked!

	lab_LogID	lab_logInt
1	1	1
2	3	2
3	2	3
4	4	6

	lab_TestID	lab_testText
1	1	One
2	3	Three
3	6	Timbrook
4	2	Two

GSM1900\RTimbro1 (66) | IST659_V2 | 00:00:00 | 9 rows

P2-TODO-1: Question/Answer

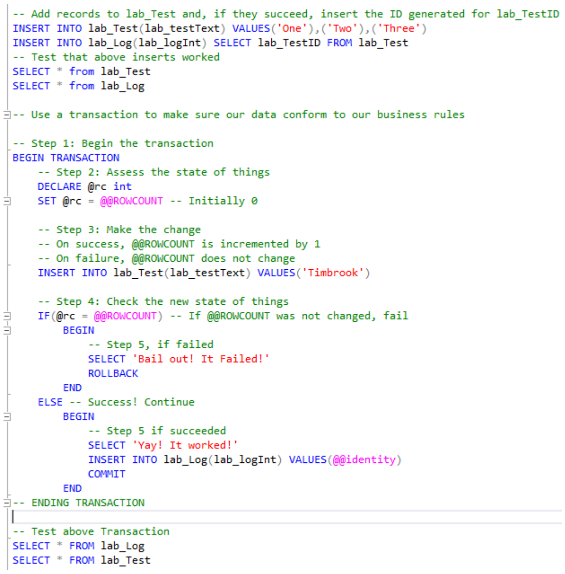
Q: Explain the reason the first execution failed, but the second did not.

A: There was already a record with the value of 'One' as the lab_testText. There's a unique constraint that prevents us from adding another record with the same value. Error Message Response: "Violation of UNIQUE KEY constraint 'UQ__lab_Test__0C6272FF5747EE6F'. Cannot insert duplicate key in object 'dbo.lab_Test'. The duplicate key value is (One)."

Q: Was there anything that happened that you didn't expect?

A: No, everything executed as expected

P2-TODO-1: Code Snippet



```
-- Add records to lab_Test and, if they succeed, insert the ID generated for lab_TestID
INSERT INTO lab_Test(lab_testText) VALUES('One'),('Two'),('Three')
INSERT INTO lab_Log(lab_logInt) SELECT lab_TestID FROM lab_Test
-- Test that above inserts worked
SELECT * from lab_Test
SELECT * from lab_Log

-- Use a transaction to make sure our data conform to our business rules

-- Step 1: Begin the transaction
BEGIN TRANSACTION
    -- Step 2: Assess the state of things
    DECLARE @rc int
    SET @rc = @@ROWCOUNT -- Initially 0

    -- Step 3: Make the change
    -- On success, @@ROWCOUNT is incremented by 1
    -- On failure, @@ROWCOUNT does not change
    INSERT INTO lab_Test(lab_testText) VALUES('Timbrook')

    -- Step 4: Check the new state of things
    IF(@rc = @@ROWCOUNT) -- If @@ROWCOUNT was not changed, fail
        BEGIN
            -- Step 5, if failed
            SELECT 'Bail out! It Failed!'
            ROLLBACK
        END
    ELSE -- Success! Continue
        BEGIN
            -- Step 5 if succeeded
            SELECT 'Yay! It worked!'
            INSERT INTO lab_Log(lab_logInt) VALUES(@@identity)
            COMMIT
        END
-- ENDING TRANSACTION

-- Test above Transaction
SELECT * FROM lab_Log
SELECT * FROM lab_Test
```

Code Submission

my tab:

```
/*
        IST 659 Data Admin Concepts &Db Mgmt
```

```
       Date: 9/10/2018
       Lab Assignment: Lab 9, Data Security
*/

/*
       TODO:
              Create Guest User
*/

-- Creating a guestuser database user
CREATE USER guestuser FOR LOGIN guestuser
/*
       TODO:
              Grant read permissions
*/
-- Grant read permission on the user table
GRANT SELECT ON vc_User to guestuser

/*
       TODO:
              Code and execute the following statements to revoke the select permission
              on vc_User and grant the select permission on the vc_MostProlificUsers
view.
*/
-- Revoke the select permissions
REVOKE SELECT ON vc_User to guestuser

-- Give them the view instead
GRANT SELECT ON vc_MostProlificUsers to guestuser

/*
       TODO:
              To allow a user to run a stored procedure, we grant them the EXECUTE
permission.
*/
-- Allow guestuser to run some stored procedures
GRANT EXECUTE ON vc_AddUserLogin TO guestuser
GRANT EXECUTE ON vc_FinishVidCast TO guestuser

-- Retrieve all rows from the vc_UserLogin table
-- ** Add screen print results to Answer doc
SELECT * from vc_UserLogin

-- Retrieve ONLY the vc_VidCast record that should have been modified by guestuser's
stored procedure call.
-- ** Add screen print results to Answer doc
-- GRANT access to vc_VidCast had to be given to guestuser get the vc_VidCastID needed
for input into the stored procedure
GRANT SELECT ON vc_VidCast TO guestuser
SELECT * FROM vc_VidCast WHERE VidCastTitle = 'Rock Your Way To Success'

/*
       Part 2 – Data Integrity Through Transactions
       The Setup:
              We’re going to set up two simple tables separate from our VidCast tables
to mess with.
*/
-- Creating a new table
```

```
CREATE TABLE lab_Test(
       lab_TestID int identity primary key,
       lab_testText varchar(20) unique not null
)

/*
       This will be a table to keep a log of created lab_Test records.
       We don't want to add a row to this if the insert into lab_Test fails
*/
CREATE TABLE lab_Log(
       lab_LogID int identity primary key,
       lab_logInt int unique not null
)

-- Add records to lab_Test and, if they succeed, insert the ID generated for lab_TestID
INSERT INTO lab_Test(lab_testText) VALUES('One'),('Two'),('Three')
INSERT INTO lab_Log(lab_logInt) SELECT lab_TestID FROM lab_Test
-- Test that above inserts worked
SELECT * from lab_Test
SELECT * from lab_Log

-- Use a transaction to make sure our data conform to our business rules

-- Step 1: Begin the transaction
BEGIN TRANSACTION
       -- Step 2: Assess the state of things
       DECLARE @rc int
       SET @rc = @@ROWCOUNT -- Initially 0

       -- Step 3: Make the change
       -- On success, @@ROWCOUNT is incremented by 1
       -- On failure, @@ROWCOUNT does not change
       INSERT INTO lab_Test(lab_testText) VALUES('Timbrook')

       -- Step 4: Check the new state of things
       IF(@rc = @@ROWCOUNT) -- If @@ROWCOUNT was not changed, fail
              BEGIN
                     -- Step 5, if failed
                     SELECT 'Bail out! It Failed!'
                     ROLLBACK
              END
       ELSE -- Success! Continue
              BEGIN
                     -- Step 5 if succeeded
                     SELECT 'Yay! It worked!'
                     INSERT INTO lab_Log(lab_logInt) VALUES(@@identity)
                     COMMIT
              END
-- ENDING TRANSACTION

-- Test above Transaction
SELECT * FROM lab_Log
SELECT * FROM lab_Test
```

guestuser tab:

```
/*
       IST 659 Data Admin Concepts &Db Mgmt
```

```
	Date: 9/10/2018
	Lab Assignment: Lab 9, Data Security

	**** GUEST USER ****
*/

-- Guestuser's tab
SELECT * FROM vc_User

/*
	Above select privilages to the vc_User table were revoked.
	In replace of direct access to the vc_User table, select privilages were granted
	to the vc_MostProlificUsers view.
*/
-- Access granted to view
SELECT * FROM vc_MostProlificUsers

/*
	TODO:
		- Code and execute the EXEC statement to add a user login for the user
		with UserName ‘TheDoctor’ with a login from ‘Gallifrey’.
*/
EXEC vc_AddUserLogin 'TheDoctor','Gallifrey'

/*
	TODO:
		- Code and execute the EXEC statement to finish the VidCast titled
		‘Rock Your Way To Success’
*/
-- GRANT access to vc_VidCast had to be given to get the vc_VidCastID needed for input
into the stored procedure
DECLARE @vidCastID int
SELECT @vidCastID=vc_VidCastID FROM vc_VidCast WHERE VidCastTitle = 'Rock Your Way To
Success'
EXEC vc_FinishVidCast @vidCastID
```