# Ryan Timbrook

## IST 659 Data Admin Concepts &Db Mgmt

## Date: 9/10/2018

## Lab Assignment: Lab 9, Data Security

## Description / Learning Objective

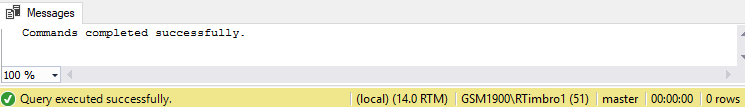
* Demonstrate proficiency in creating database users and administering to their user privileges on database objects
* Demonstrate proficiency in preserving data integrity using transactions

## Responses

### Part 1 - Securing Data Objects

P1-TODO-1: Creating a Database User

P1-TODO-1: Screen Prints



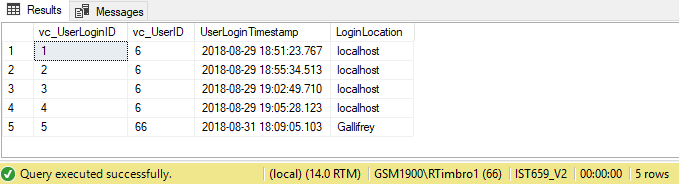
P1-TODO-1: Code Snippet



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

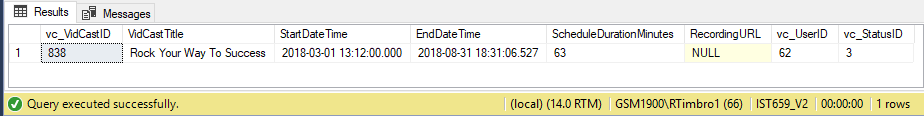
P1-TODO-2a: vc\_UserLogin Table

P1-TODO-2a: Screen Prints

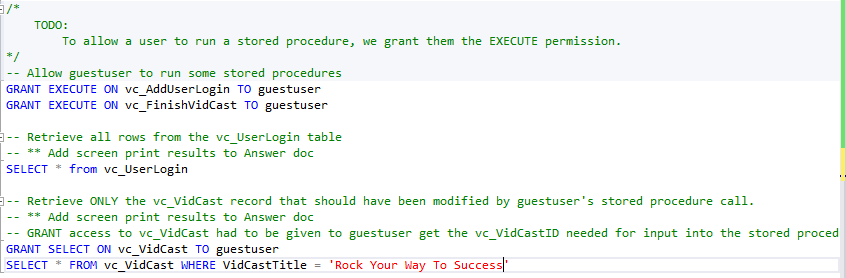


P1-TODO-2b: vc\_VidCast records

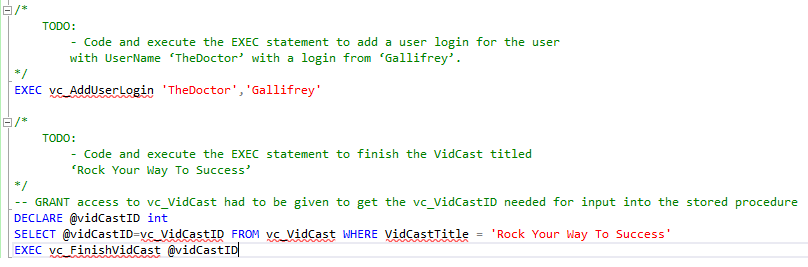
P1-TODO-2b: Screen Prints



P1-TODO-2: Code Snippets - my tab



P1-TODO-2: Code Snippets – guestuser tab



### Part 2 – Data Integrity Through Transactions

P2-TODO-1:

P2-TODO-1: Screen Prints

|  |  |
| --- | --- |
| First Run: With Expected Error | Second Run: Without Error |

P2-TODO-1: Question/Answer

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

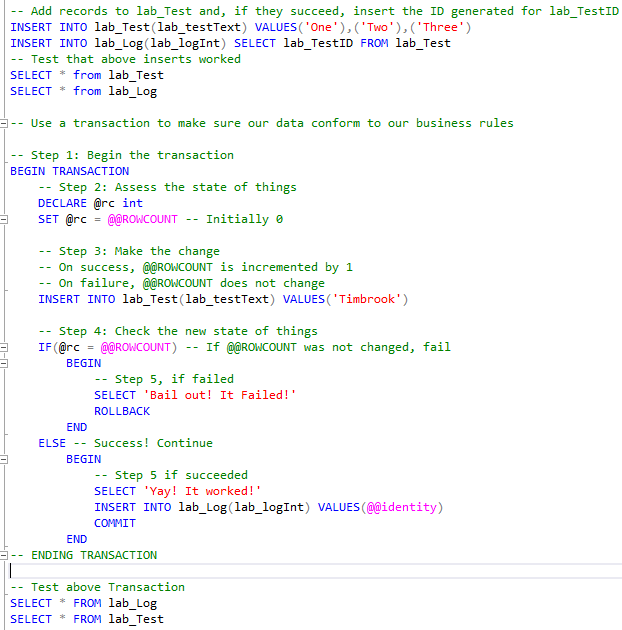
A: There was already a record with the value of ‘One’ as the lab\_testText. There’s a unique constraint that prevents us from adding another record with the same value.

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



## 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 |