Unit 11 Problem Set Submission Form

Overview

Your Name	Jainish Savaliya
Your SU Email	jsavaliy@syr.edu

Instructions

Put your name and SU email at the top. Answer these questions all from the lab. When asked to include screenshots, please follow the screen shot guidelines from the first lab.

Remember as you complete the problem sets it is not only about getting it right / correct. We will discuss the answers in class so it's important to articulate anything you would like to contribute to the discussion in your answer:

- If you feel the question is vague, include any assumptions you've made.
- If you feel the answer requires interpretation or justification provide it.
- If you do not know the answer to the question, articulate what you tried and how you are stuck.

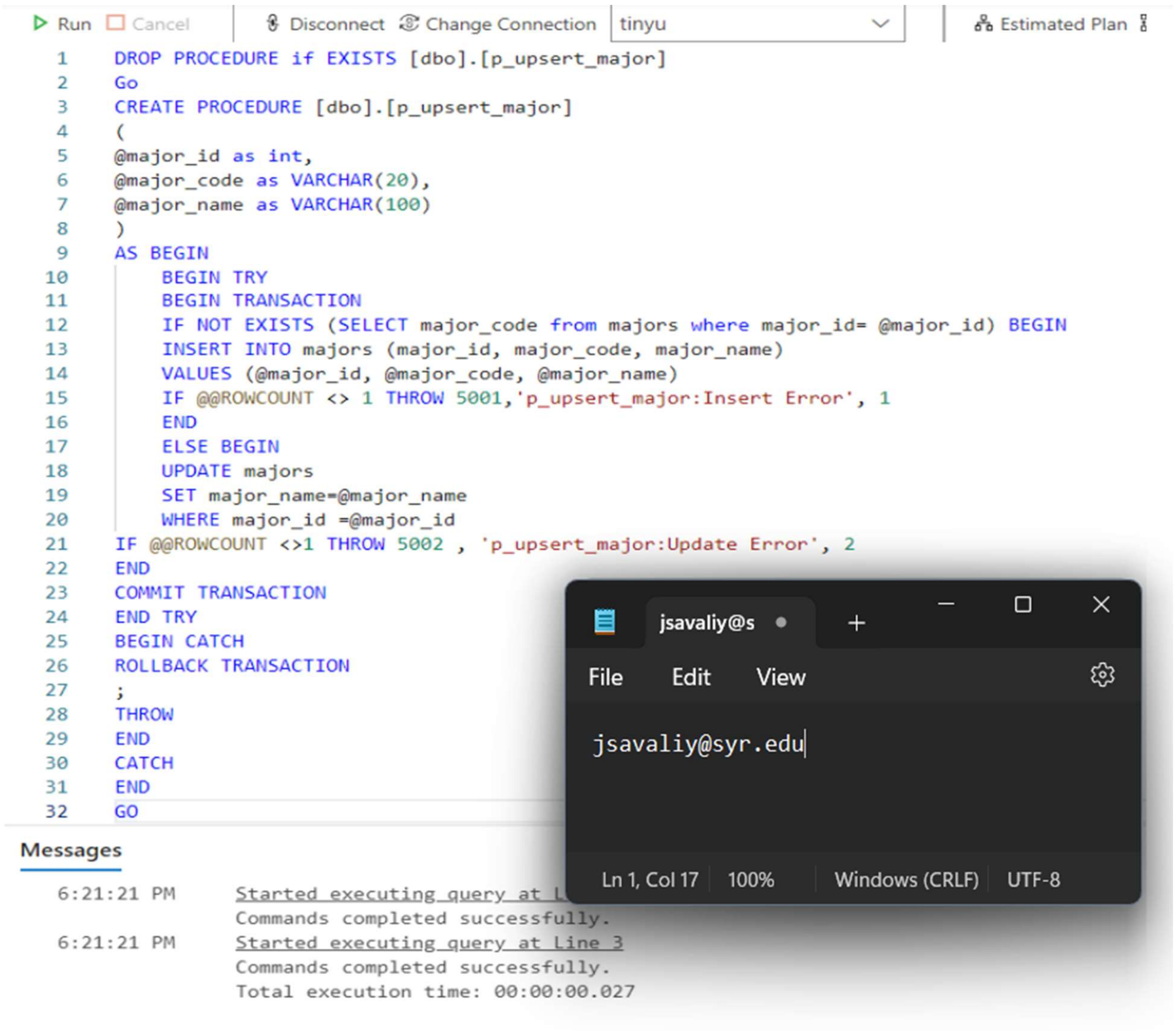
This how you receive credit for answering questions which might not be correct.

Questions

Answer these questions using the problem set submission template. You will need to consult the logical model in the overview section for details. For any screenshots provided, please follow the guidelines for submitting a screenshot.

Write the following as SQL programs. For each, include the SQL as a screenshot with the output of the SQL Code.

1. Provide a screenshot of your code execution from the walkthrough were you modified **p_upsert_major** in the **TinyU** database to be transaction-safe.



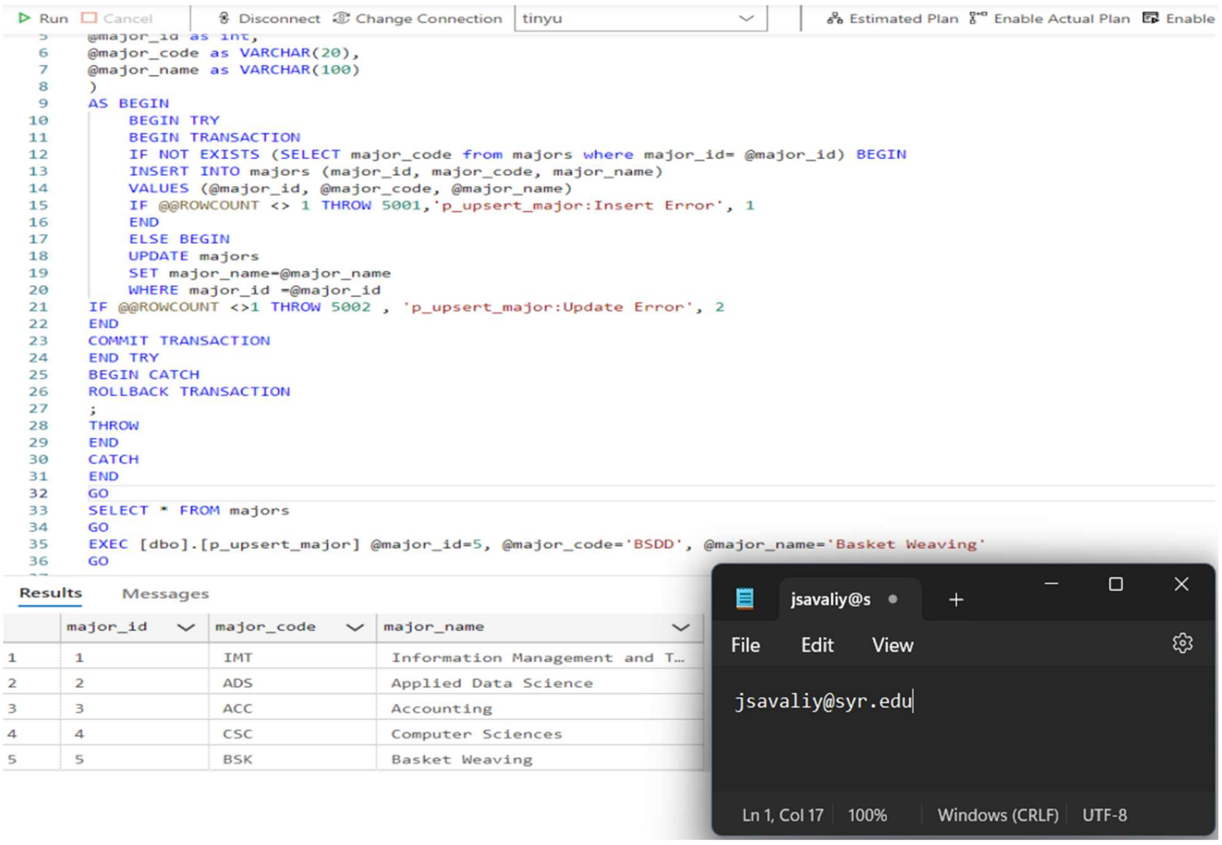
```
DROP PROCEDURE if EXISTS [dbo].[p_upsert_major]
Go
CREATE PROCEDURE [dbo].[p_upsert_major]
(
@major_id as int,
@major_code as VARCHAR(20),
@major_name as VARCHAR(100)
)
AS BEGIN
    BEGIN TRY
    BEGIN TRANSACTION
    IF NOT EXISTS (SELECT major_code from majors where major_id= @major_id) BEGIN
    INSERT INTO majors (major_id, major_code, major_name)
    VALUES (@major_id, @major_code, @major_name)
    IF @@ROWCOUNT <> 1 THROW 5001,'p_upsert_major:Insert Error', 1
    END
    ELSE BEGIN
    UPDATE majors
    SET major_name=@major_name
    WHERE major_id =@major_id
IF @@ROWCOUNT <>1 THROW 5002 , 'p_upsert_major:Update Error', 2
END
COMMIT TRANSACTION
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION
;
THROW
END
CATCH
END
GO
```

```
jsavaliy@syr.edu
```

Messages

```
6:21:21 PM     Started executing query at L
               Commands completed successfully.
6:21:21 PM     Started executing query at Line 3
               Commands completed successfully.
               Total execution time: 00:00:00.027
```

2. Provide a screenshot of examples of executing the **p_upsert_major** procedure to demonstrate it is transaction safe.

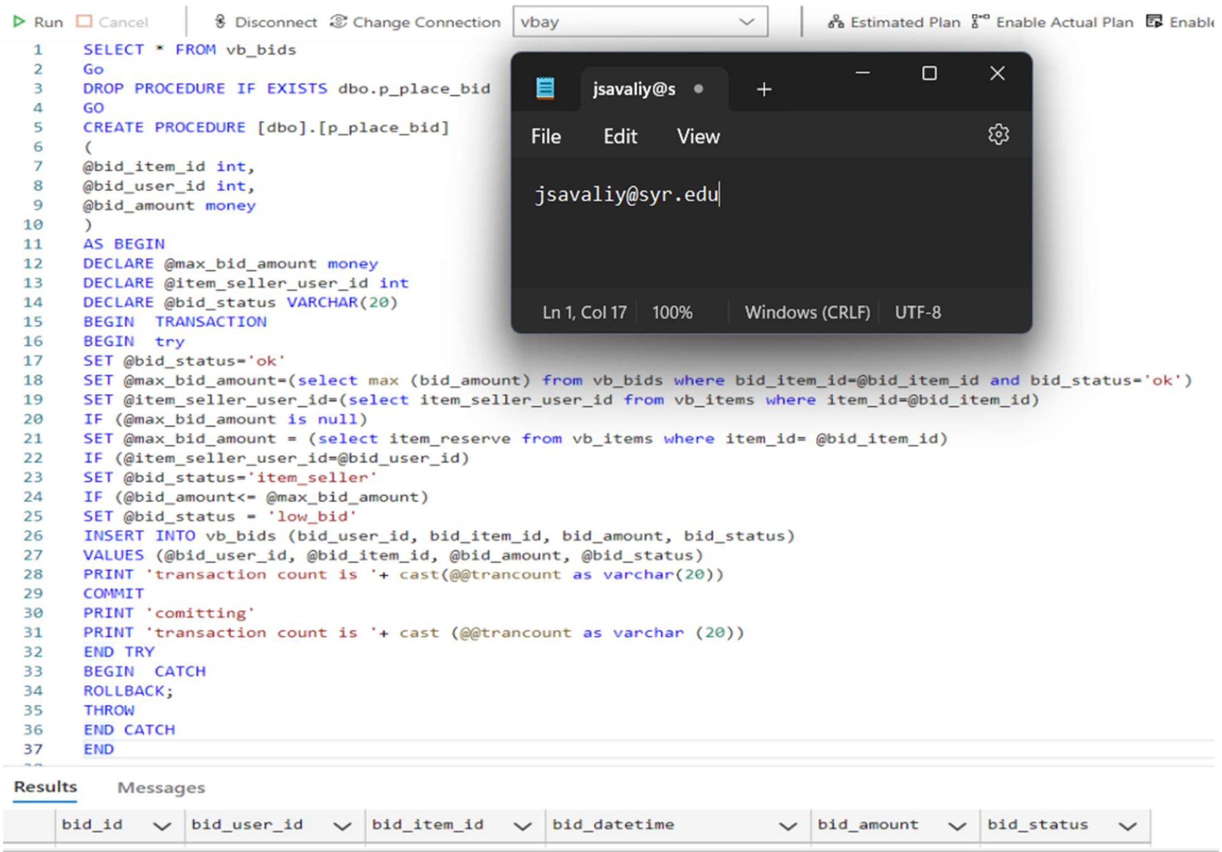


```sql
@major_id as int,
@major_code as VARCHAR(20),
@major_name as VARCHAR(100)
)
AS BEGIN
    BEGIN TRY
    BEGIN TRANSACTION
    IF NOT EXISTS (SELECT major_code from majors where major_id= @major_id) BEGIN
    INSERT INTO majors (major_id, major_code, major_name)
    VALUES (@major_id, @major_code, @major_name)
    IF @@ROWCOUNT <> 1 THROW 5001,'p_upsert_major:Insert Error', 1
    END
    ELSE BEGIN
    UPDATE majors
    SET major_name=@major_name
    WHERE major_id =@major_id
IF @@ROWCOUNT <>1 THROW 5002 , 'p_upsert_major:Update Error', 2
END
COMMIT TRANSACTION
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION
;
THROW
END
CATCH
END
GO
SELECT * FROM majors
GO
EXEC [dbo].[p_upsert_major] @major_id=5, @major_code='BSDD', @major_name='Basket Weaving'
GO
```

	major_id	major_code	major_name
1	1	IMT	Information Management and T…
2	2	ADS	Applied Data Science
3	3	ACC	Accounting
4	4	CSC	Computer Sciences
5	5	BSK	Basket Weaving

jsavaliy@syr.edu

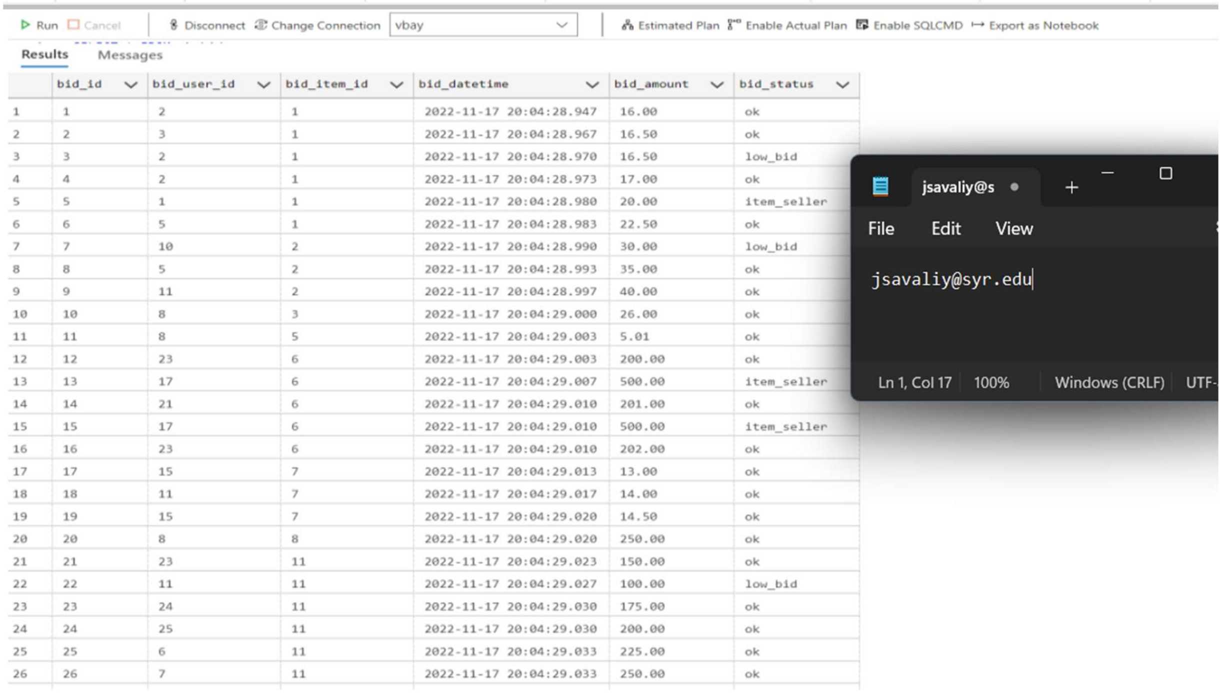
3. Re-write the **p_place_bid** stored procedure from the **vBay** database so that it is transaction safe. Provide a screenshot of the code and its execution.



```sql
SELECT * FROM vb_bids
Go
DROP PROCEDURE IF EXISTS dbo.p_place_bid
GO
CREATE PROCEDURE [dbo].[p_place_bid]
(
@bid_item_id int,
@bid_user_id int,
@bid_amount money
)
AS BEGIN
DECLARE @max_bid_amount money
DECLARE @item_seller_user_id int
DECLARE @bid_status VARCHAR(20)
BEGIN  TRANSACTION
BEGIN  try
SET @bid_status='ok'
SET @max_bid_amount=(select max (bid_amount) from vb_bids where bid_item_id=@bid_item_id and bid_status='ok')
SET @item_seller_user_id=(select item_seller_user_id from vb_items where item_id=@bid_item_id)
IF (@max_bid_amount is null)
SET @max_bid_amount = (select item_reserve from vb_items where item_id= @bid_item_id)
IF (@item_seller_user_id=@bid_user_id)
SET @bid_status='item_seller'
IF (@bid_amount<= @max_bid_amount)
SET @bid_status = 'low_bid'
INSERT INTO vb_bids (bid_user_id, bid_item_id, bid_amount, bid_status)
VALUES (@bid_user_id, @bid_item_id, @bid_amount, @bid_status)
PRINT 'transaction count is '+ cast(@@trancount as varchar(20))
COMMIT
PRINT 'comitting'
PRINT 'transaction count is '+ cast (@@trancount as varchar (20))
END TRY
BEGIN  CATCH
ROLLBACK;
THROW
END CATCH
END
```

```
jsavaliy@syr.edu
```

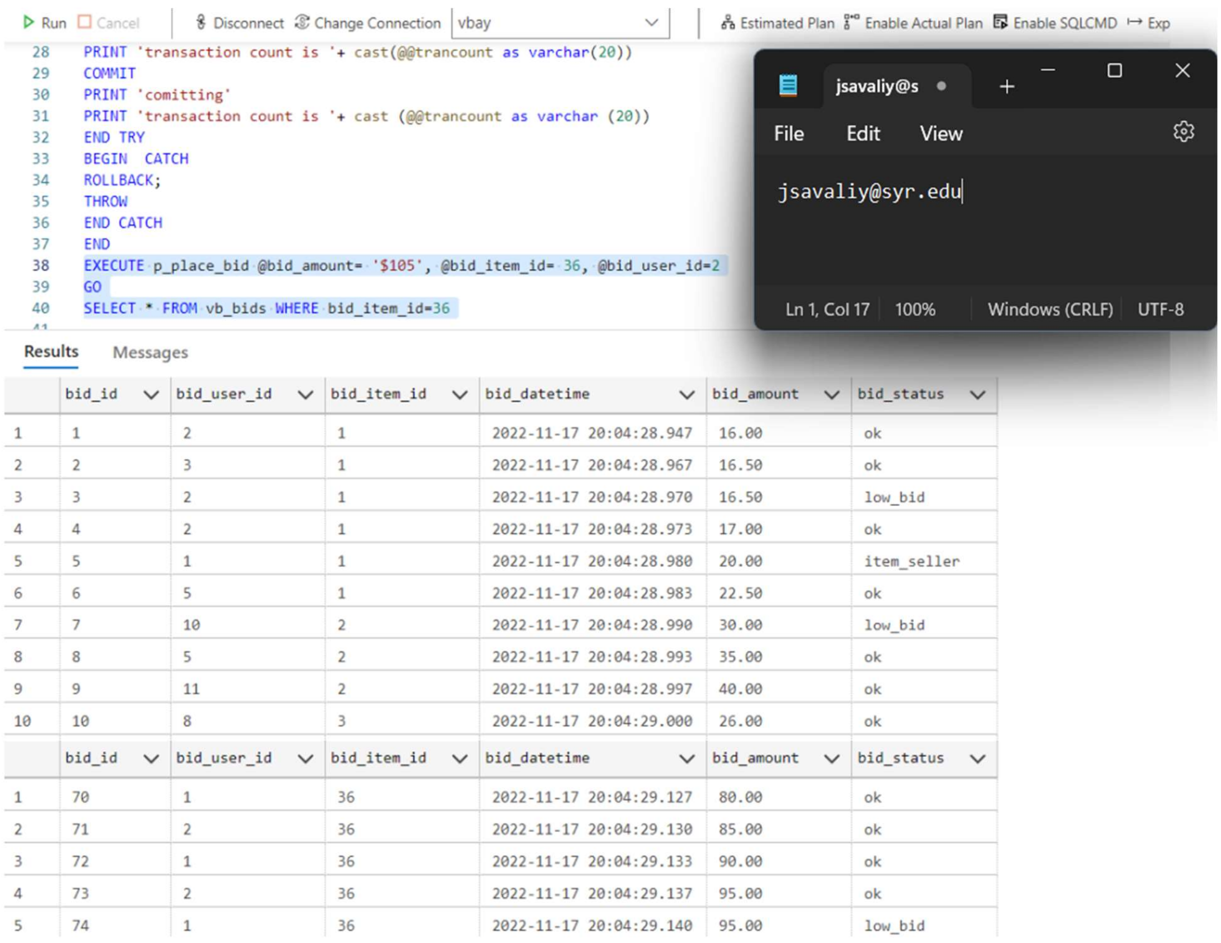
	bid_id	bid_user_id	bid_item_id	bid_datetime	bid_amount	bid_status



	bid_id	bid_user_id	bid_item_id	bid_datetime	bid_amount	bid_status
1	1	2	1	2022-11-17 20:04:28.947	16.00	ok
2	2	3	1	2022-11-17 20:04:28.967	16.50	ok
3	3	2	1	2022-11-17 20:04:28.970	16.50	low_bid
4	4	2	1	2022-11-17 20:04:28.973	17.00	ok
5	5	1	1	2022-11-17 20:04:28.980	20.00	item_seller
6	6	5	1	2022-11-17 20:04:28.983	22.50	ok
7	7	10	2	2022-11-17 20:04:28.990	30.00	low_bid
8	8	5	2	2022-11-17 20:04:28.993	35.00	ok
9	9	11	2	2022-11-17 20:04:28.997	40.00	ok
10	10	8	3	2022-11-17 20:04:29.000	26.00	ok
11	11	8	5	2022-11-17 20:04:29.003	5.01	ok
12	12	23	6	2022-11-17 20:04:29.003	200.00	ok
13	13	17	6	2022-11-17 20:04:29.007	500.00	item_seller
14	14	21	6	2022-11-17 20:04:29.010	201.00	ok
15	15	17	6	2022-11-17 20:04:29.010	500.00	item_seller
16	16	23	6	2022-11-17 20:04:29.010	202.00	ok
17	17	15	7	2022-11-17 20:04:29.013	13.00	ok
18	18	11	7	2022-11-17 20:04:29.017	14.00	ok
19	19	15	7	2022-11-17 20:04:29.020	14.50	ok
20	20	8	8	2022-11-17 20:04:29.020	250.00	ok
21	21	23	11	2022-11-17 20:04:29.023	150.00	ok
22	22	11	11	2022-11-17 20:04:29.027	100.00	low_bid
23	23	24	11	2022-11-17 20:04:29.030	175.00	ok
24	24	25	11	2022-11-17 20:04:29.030	200.00	ok
25	25	6	11	2022-11-17 20:04:29.033	225.00	ok
26	26	7	11	2022-11-17 20:04:29.033	250.00	ok

```
jsavaliy@syr.edu
```

4. Execute your stored procedure in step 3 to demonstrate the procedure works. Make user 2, Bid $105 on item 36 and show the bid was placed with a SELECT.



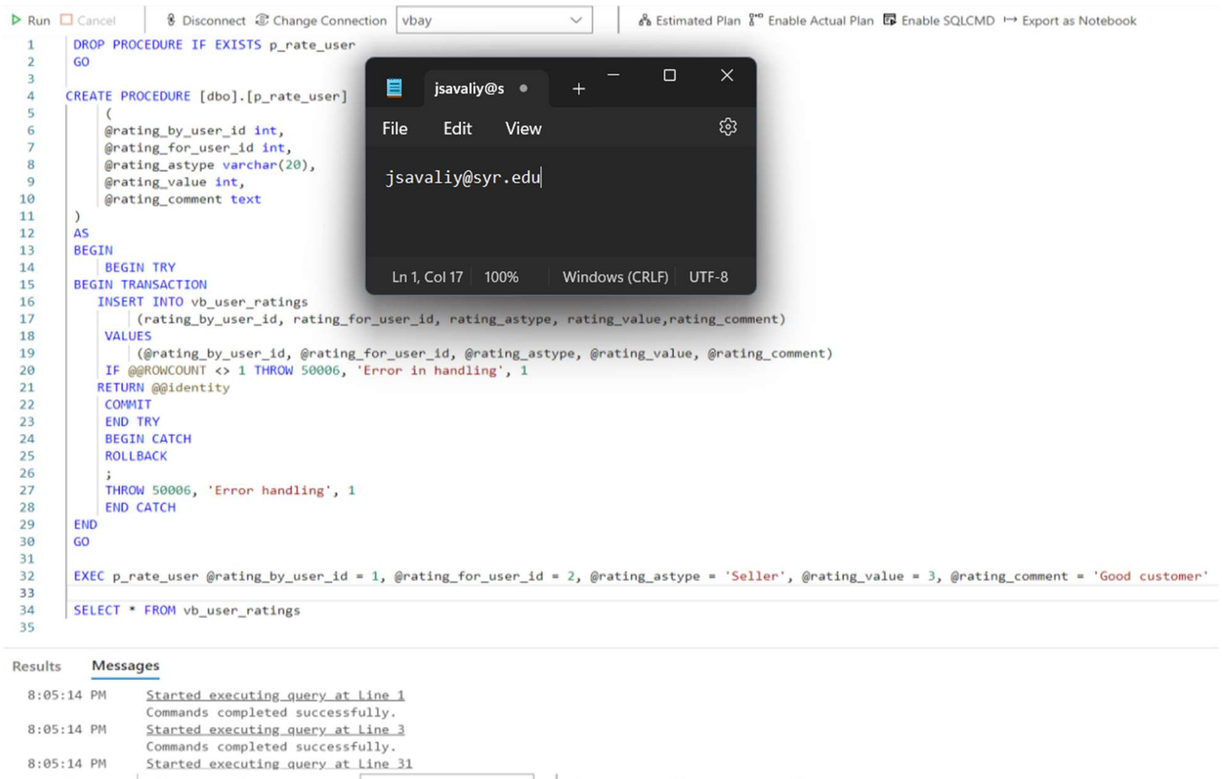
```
PRINT 'transaction count is '+ cast(@@trancount as varchar(20))
COMMIT
PRINT 'comitting'
PRINT 'transaction count is '+ cast (@@trancount as varchar (20))
END TRY
BEGIN  CATCH
ROLLBACK;
THROW
END CATCH
END
EXECUTE p_place_bid @bid_amount= '$105', @bid_item_id= 36, @bid_user_id=2
GO
SELECT * FROM vb_bids WHERE bid_item_id=36
```

```
jsavaliy@syr.edu
```

	bid_id	bid_user_id	bid_item_id	bid_datetime	bid_amount	bid_status
1	1	2	1	2022-11-17 20:04:28.947	16.00	ok
2	2	3	1	2022-11-17 20:04:28.967	16.50	ok
3	3	2	1	2022-11-17 20:04:28.970	16.50	low_bid
4	4	2	1	2022-11-17 20:04:28.973	17.00	ok
5	5	1	1	2022-11-17 20:04:28.980	20.00	item_seller
6	6	5	1	2022-11-17 20:04:28.983	22.50	ok
7	7	10	2	2022-11-17 20:04:28.990	30.00	low_bid
8	8	5	2	2022-11-17 20:04:28.993	35.00	ok
9	9	11	2	2022-11-17 20:04:28.997	40.00	ok
10	10	8	3	2022-11-17 20:04:29.000	26.00	ok

	bid_id	bid_user_id	bid_item_id	bid_datetime	bid_amount	bid_status
1	70	1	36	2022-11-17 20:04:29.127	80.00	ok
2	71	2	36	2022-11-17 20:04:29.130	85.00	ok
3	72	1	36	2022-11-17 20:04:29.133	90.00	ok
4	73	2	36	2022-11-17 20:04:29.137	95.00	ok
5	74	1	36	2022-11-17 20:04:29.140	95.00	low_bid

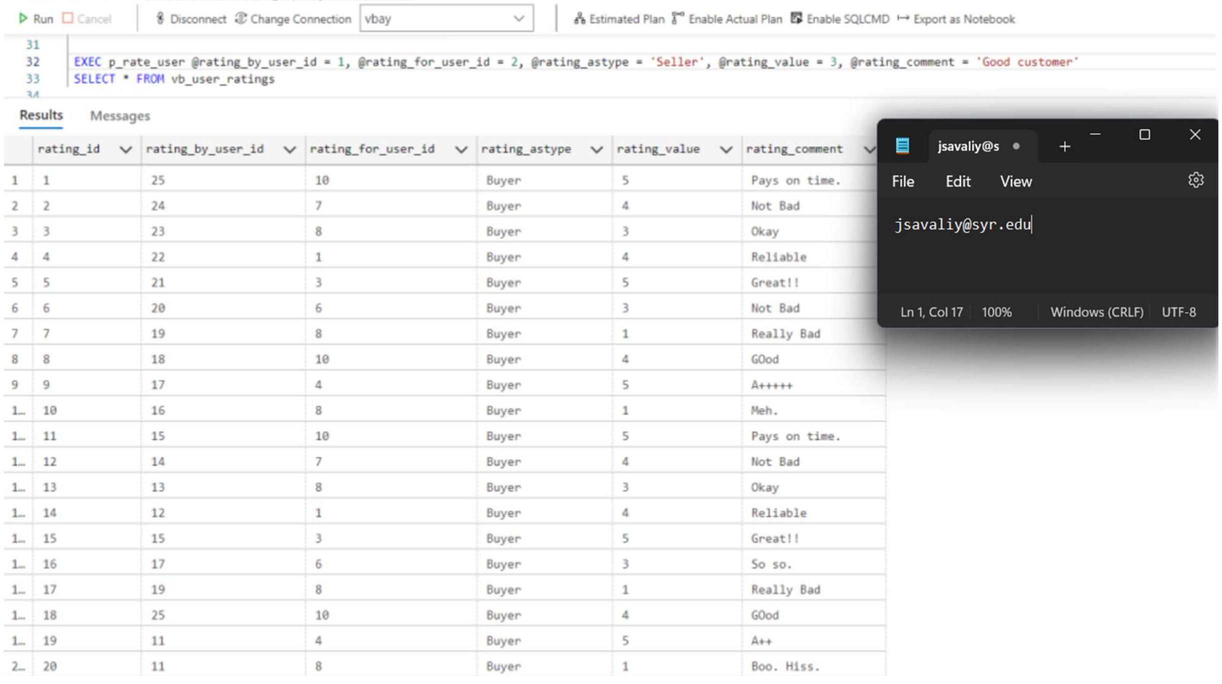
5. Re-write the **p_rate_user** stored procedure from the **VBay** database so that it is transaction safe. Provide a screenshot of the code and its execution.



jsavaliy@syr.edu

```sql
DROP PROCEDURE IF EXISTS p_rate_user
GO

CREATE PROCEDURE [dbo].[p_rate_user]
    (
    @rating_by_user_id int,
    @rating_for_user_id int,
    @rating_astype varchar(20),
    @rating_value int,
    @rating_comment text
)
AS
BEGIN
    BEGIN TRY
BEGIN TRANSACTION
   INSERT INTO vb_user_ratings
        (rating_by_user_id, rating_for_user_id, rating_astype, rating_value,rating_comment)
    VALUES
        (@rating_by_user_id, @rating_for_user_id, @rating_astype, @rating_value, @rating_comment)
    IF @@ROWCOUNT <> 1 THROW 50006, 'Error in handling', 1
   RETURN @@identity
    COMMIT
    END TRY
    BEGIN CATCH
    ROLLBACK
    ;
    THROW 50006, 'Error handling', 1
    END CATCH
END
GO

EXEC p_rate_user @rating_by_user_id = 1, @rating_for_user_id = 2, @rating_astype = 'Seller', @rating_value = 3, @rating_comment = 'Good customer'

SELECT * FROM vb_user_ratings
```

```
8:05:14 PM    Started executing query at Line 1
              Commands completed successfully.
8:05:14 PM    Started executing query at Line 3
              Commands completed successfully.
8:05:14 PM    Started executing query at Line 31
```

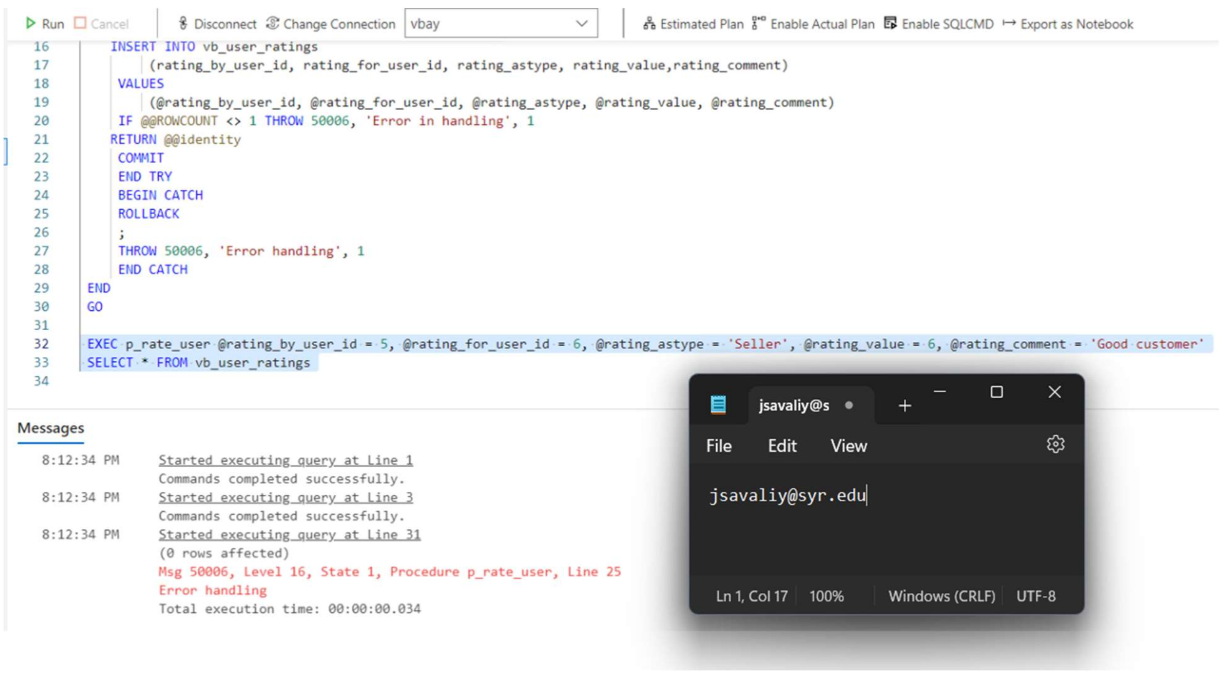


jsavaliy@syr.edu

```sql
EXEC p_rate_user @rating_by_user_id = 1, @rating_for_user_id = 2, @rating_astype = 'Seller', @rating_value = 3, @rating_comment = 'Good customer'
SELECT * FROM vb_user_ratings
```

	rating_id	rating_by_user_id	rating_for_user_id	rating_astype	rating_value	rating_comment
1	1	25	10	Buyer	5	Pays on time.
2	2	24	7	Buyer	4	Not Bad
3	3	23	8	Buyer	3	Okay
4	4	22	1	Buyer	4	Reliable
5	5	21	3	Buyer	5	Great!!
6	6	20	6	Buyer	3	Not Bad
7	7	19	8	Buyer	1	Really Bad
8	8	18	10	Buyer	4	GOod
9	9	17	4	Buyer	5	A+++++
1…	10	16	8	Buyer	1	Meh.
1…	11	15	10	Buyer	5	Pays on time.
1…	12	14	7	Buyer	4	Not Bad
1…	13	13	8	Buyer	3	Okay
1…	14	12	1	Buyer	4	Reliable
1…	15	15	3	Buyer	5	Great!!
1…	16	17	6	Buyer	3	So so.
1…	17	19	8	Buyer	1	Really Bad
1…	18	25	10	Buyer	4	GOod
1…	19	11	4	Buyer	5	A++
2…	20	11	8	Buyer	1	Boo. Hiss.

6. Execute the stored procedure in step 5 to demonstrate the rollback works. You should give a 6 star rating and then execute again where someone attempts to rate themselves. Produce as screen shot as evidence the rollback worked.



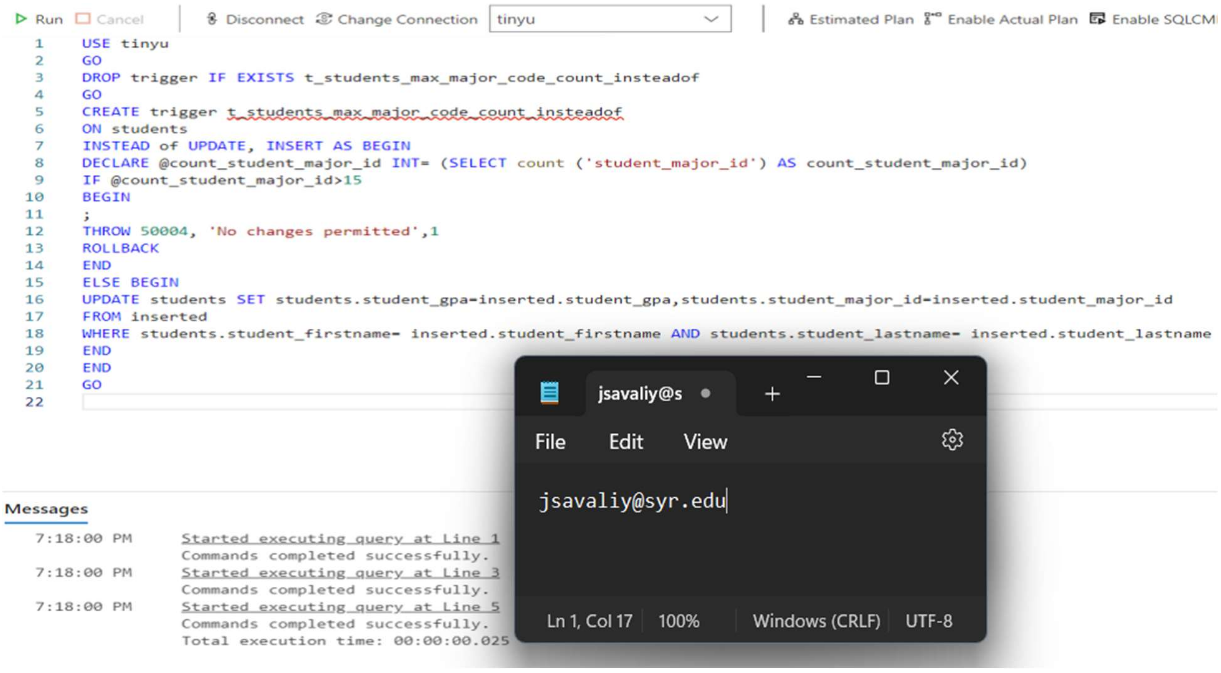
```sql
    INSERT INTO vb_user_ratings
        (rating_by_user_id, rating_for_user_id, rating_astype, rating_value,rating_comment)
     VALUES
        (@rating_by_user_id, @rating_for_user_id, @rating_astype, @rating_value, @rating_comment)
     IF @@ROWCOUNT <> 1 THROW 50006, 'Error in handling', 1
    RETURN @@identity
     COMMIT
     END TRY
     BEGIN CATCH
     ROLLBACK
     ;
     THROW 50006, 'Error handling', 1
     END CATCH
END
GO

EXEC p_rate_user @rating_by_user_id = 5, @rating_for_user_id = 6, @rating_astype = 'Seller', @rating_value = 6, @rating_comment = 'Good customer'
SELECT * FROM vb_user_ratings
```

Messages

```
8:12:34 PM    Started executing query at Line 1
              Commands completed successfully.
8:12:34 PM    Started executing query at Line 3
              Commands completed successfully.
8:12:34 PM    Started executing query at Line 31
              (0 rows affected)
              Msg 50006, Level 16, State 1, Procedure p_rate_user, Line 25
              Error handling
              Total execution time: 00:00:00.034
```

jsavaliy@syr.edu

7. There is a conceptual data requirement which says that no **TinyU** major can have more than 15 students in it. (I know, this seems silly but think of the bigger problem – how to we enforce a specific minimum or maximum cardinality instead of just 1 or "many"?) Write data logic using an instead of trigger to do this.



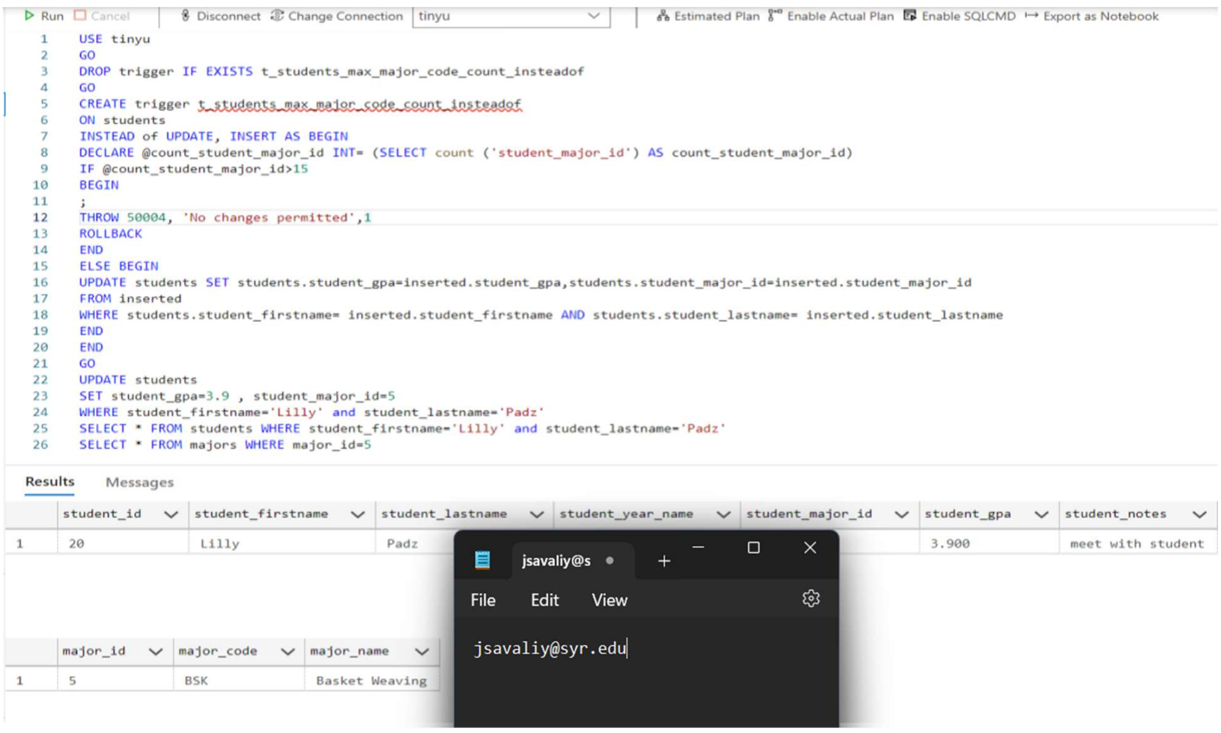
```sql
USE tinyu
GO
DROP trigger IF EXISTS t_students_max_major_code_count_insteadof
GO
CREATE trigger t_students_max_major_code_count_insteadof
ON students
INSTEAD of UPDATE, INSERT AS BEGIN
DECLARE @count_student_major_id INT= (SELECT count ('student_major_id') AS count_student_major_id)
IF @count_student_major_id>15
BEGIN
;
THROW 50004, 'No changes permitted',1
ROLLBACK
END
ELSE BEGIN
UPDATE students SET students.student_gpa=inserted.student_gpa,students.student_major_id=inserted.student_major_id
FROM inserted
WHERE students.student_firstname= inserted.student_firstname AND students.student_lastname= inserted.student_lastname
END
END
GO
```

Messages

```
7:18:00 PM    Started executing query at Line 1
              Commands completed successfully.
7:18:00 PM    Started executing query at Line 3
              Commands completed successfully.
7:18:00 PM    Started executing query at Line 5
              Commands completed successfully.
              Total execution time: 00:00:00.025
```

jsavaliy@syr.edu

8. Test step 7 by trying to add or update a student and change their major to ADS. The ADS major has 15 students already. Your code should drop/create the trigger and also test the success and failure of the trigger.



```sql
USE tinyu
GO
DROP trigger IF EXISTS t_students_max_major_code_count_insteadof
GO
CREATE trigger t_students_max_major_code_count_insteadof
ON students
INSTEAD of UPDATE, INSERT AS BEGIN
DECLARE @count_student_major_id INT= (SELECT count ('student_major_id') AS count_student_major_id)
IF @count_student_major_id>15
BEGIN
;
THROW 50004, 'No changes permitted',1
ROLLBACK
END
ELSE BEGIN
UPDATE students SET students.student_gpa=inserted.student_gpa,students.student_major_id=inserted.student_major_id
FROM inserted
WHERE students.student_firstname= inserted.student_firstname AND students.student_lastname= inserted.student_lastname
END
END
GO
UPDATE students
SET student_gpa=3.9 , student_major_id=5
WHERE student_firstname='Lilly' and student_lastname='Padz'
SELECT * FROM students WHERE student_firstname='Lilly' and student_lastname='Padz'
SELECT * FROM majors WHERE major_id=5
```

	student_id	student_firstname	student_lastname	student_year_name	student_major_id	student_gpa	student_notes
1	20	Lilly	Padz			3.900	meet with student

	major_id	major_code	major_name
1	5	BSK	Basket Weaving

```
jsavaliy@syr.edu
```

Reflection

Use this section to reflect on your learning. To achieve the highest grade on the assignment you must be as descriptive and personal as possible with your reflection.

1. What are the key things you learned through the process of completing this assignment?
   -> In this assignment the key thing I learnt about was transaction safe.

2. What were the challenges or roadblocks (if any) you encountered on the way to completing it?
   -> I was able to complete the assignment without any major challenges.

3. Were you prepared for this assignment? What can you do to be better prepared?
   -> I watched the video and practiced the queries to be prepared for this assignment

4. Now that you have completed the assignment rate your comfort level with this week's material. This should be an honest assessment: (choose one)
   4 ==> I understand this material and can explain it to others.