Homework Problem Set 7: Database Programming

Overview

In this lab, we will explore database programming of procedures, views, triggers, and functions.

Learning Objectives

Upon completion of the lab, you should be able to:

- Write your own data logic as user-defined functions.
- Write your own data logic as triggers.
- Write your own data logic as stored procedures.
- Use built-in functions to solve data-logic-type problems.

What You Will Need

To complete this lab, you will need the learn-databases environment up and running, specifically:

- Microsoft SQL Server DBMS.
- Provision the **TinyU** database using the database provisioner application https://localhost:5000.
- Azure Data Studio connected to SQL Server with an open query window.
- Please review the first lab if you require assistance with these tools.

Questions

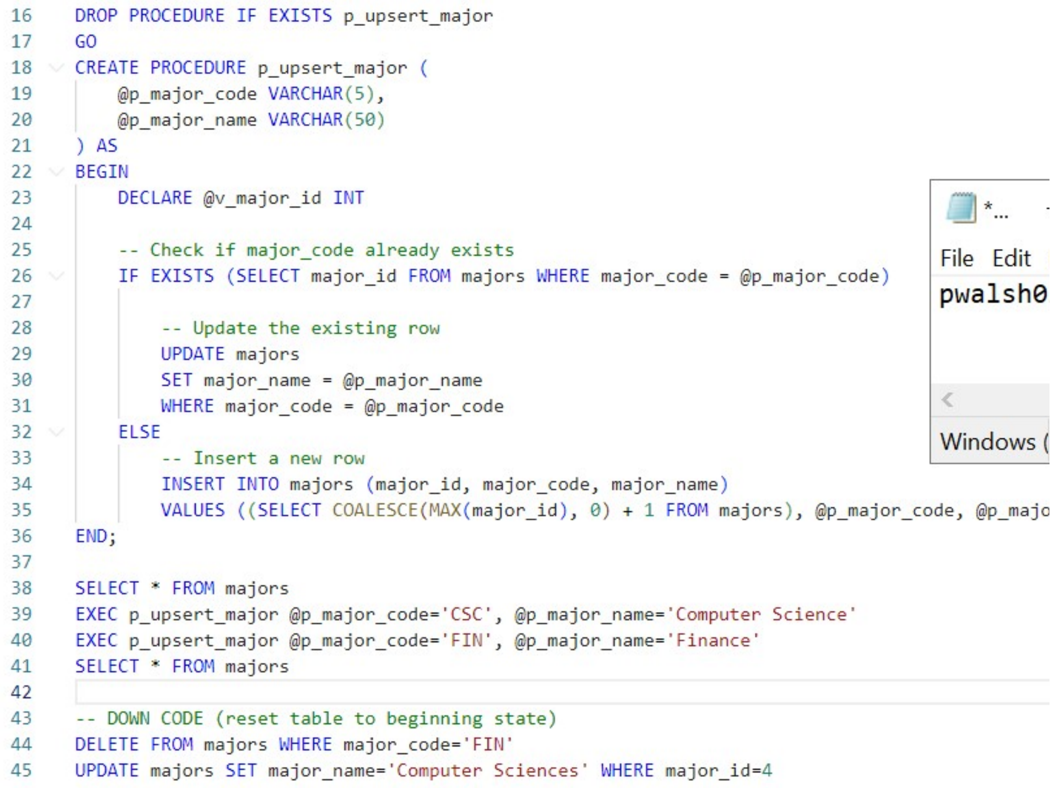
Answer these questions using the problem set submission tem For any screen shots provided, please follow the guidelines for submitting a screen shot.

Write the following as SQL programs. For each, include the SQL as a screen shot with the output of the query.

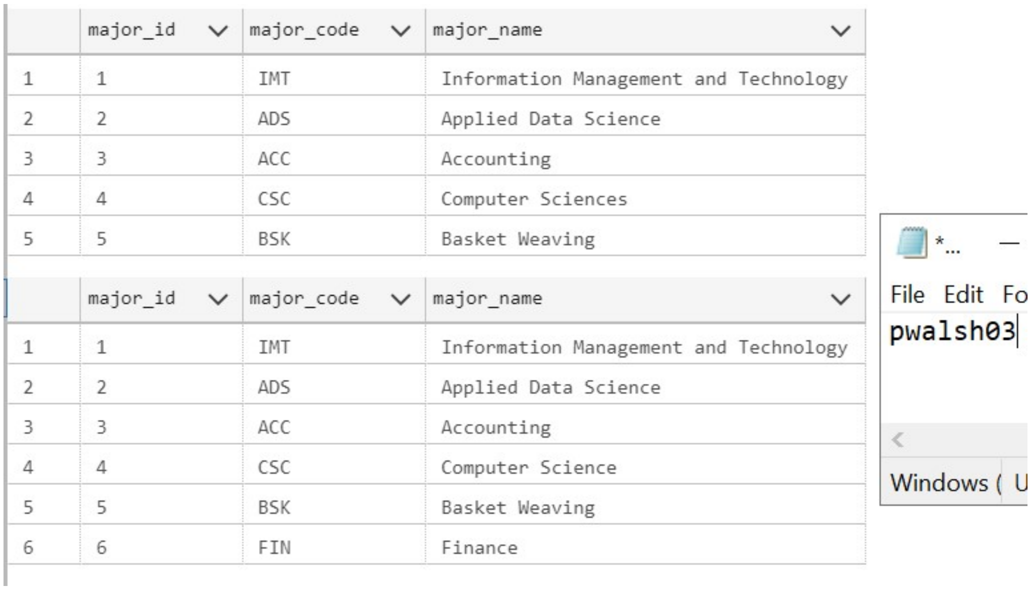
- In the **TinyU** database:
    - Write an SQL Stored procedure called **p_upsert_major**, which, given a major_code (business key) and a major_name, does an Upsert, which is the following:
        - Checks if the major_code exists in the table already.
        - If yes, updates the table and makes the major_name match the new major name.
        - If no, inserts the new major_name and major_code into the table. HINT: major_id is not a surrogate key, so you will need to determine the next ID yourself in code!
    - Test your stored procedure by executing it to make these changes:

- Change : CSC—Computer Sciences to CSC—Computer Science
- Add: FIN—Finance

Make sure your screen shot captures all up/down code in 1.a AND another screen shot captures 1.b—the output of your code execution—to show that it works. SELECT the table before and after!



```
DROP PROCEDURE IF EXISTS p_upsert_major
GO
CREATE PROCEDURE p_upsert_major (
    @p_major_code VARCHAR(5),
    @p_major_name VARCHAR(50)
) AS
BEGIN
    DECLARE @v_major_id INT

    -- Check if major_code already exists
    IF EXISTS (SELECT major_id FROM majors WHERE major_code = @p_major_code)

        -- Update the existing row
        UPDATE majors
        SET major_name = @p_major_name
        WHERE major_code = @p_major_code
    ELSE
        -- Insert a new row
        INSERT INTO majors (major_id, major_code, major_name)
        VALUES ((SELECT COALESCE(MAX(major_id), 0) + 1 FROM majors), @p_major_code, @p_majo
END;

SELECT * FROM majors
EXEC p_upsert_major @p_major_code='CSC', @p_major_name='Computer Science'
EXEC p_upsert_major @p_major_code='FIN', @p_major_name='Finance'
SELECT * FROM majors

-- DOWN CODE (reset table to beginning state)
DELETE FROM majors WHERE major_code='FIN'
UPDATE majors SET major_name='Computer Sciences' WHERE major_id=4
```

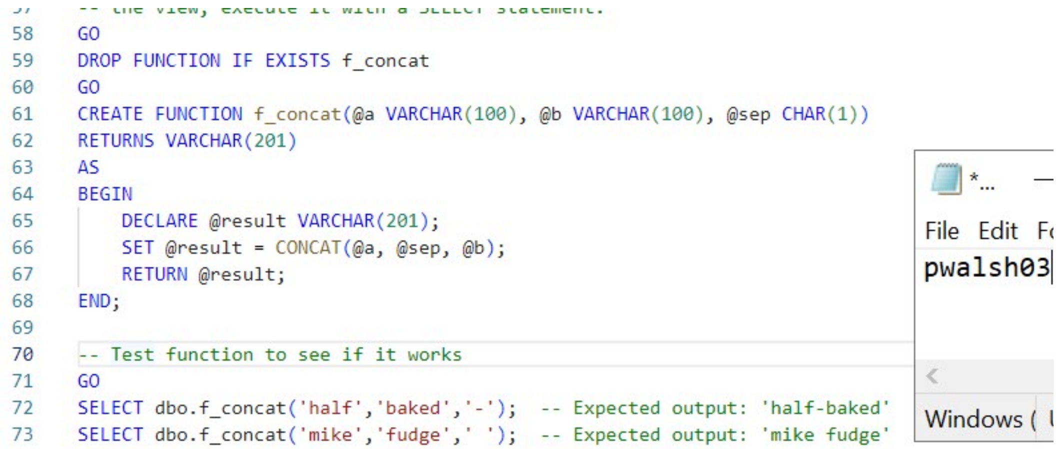


	major_id	major_code	major_name
1	1	IMT	Information Management and Technology
2	2	ADS	Applied Data Science
3	3	ACC	Accounting
4	4	CSC	Computer Sciences
5	5	BSK	Basket Weaving

	major_id	major_code	major_name
1	1	IMT	Information Management and Technology
2	2	ADS	Applied Data Science
3	3	ACC	Accounting
4	4	CSC	Computer Science
5	5	BSK	Basket Weaving
6	6	FIN	Finance

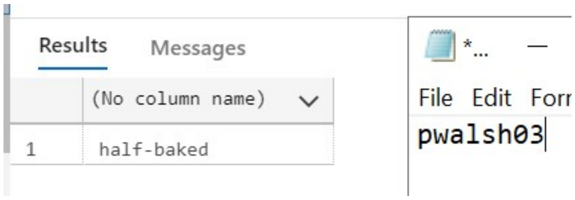
- In the **TinyU** database:
  - Write a user-defined function called **f_concat** that combines the any two varchars @a and @b together with a one-character @sep in between.
    For example:
    ```
    select dbo.f_concat('half','baked','-') -- 'half-baked'
    select dbo.f_concat('mike', 'fudge', ' ') -- 'mike fudge'
    ```
  - Now create a view called **v_students** that displays the student_id, student name (first last), student name (last, first), GPA, and name of major. You should call the function you created in 2.a. After you create the view, execute it with a SELECT statement.

Make sure your screen shot captures all up/down code in 2.a AND another screen shot captures 2.b, along with the output of the SELECT statement on the view (first few rows is fine).



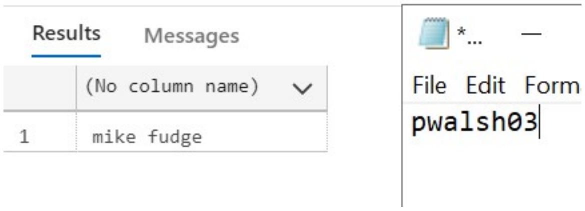
```
-- the view, execute it with a SELECT statement.
GO
DROP FUNCTION IF EXISTS f_concat
GO
CREATE FUNCTION f_concat(@a VARCHAR(100), @b VARCHAR(100), @sep CHAR(1))
RETURNS VARCHAR(201)
AS
BEGIN
    DECLARE @result VARCHAR(201);
    SET @result = CONCAT(@a, @sep, @b);
    RETURN @result;
END;

-- Test function to see if it works
GO
SELECT dbo.f_concat('half','baked','-');  -- Expected output: 'half-baked'
SELECT dbo.f_concat('mike','fudge',' ');  -- Expected output: 'mike fudge'
```

pwalsh03



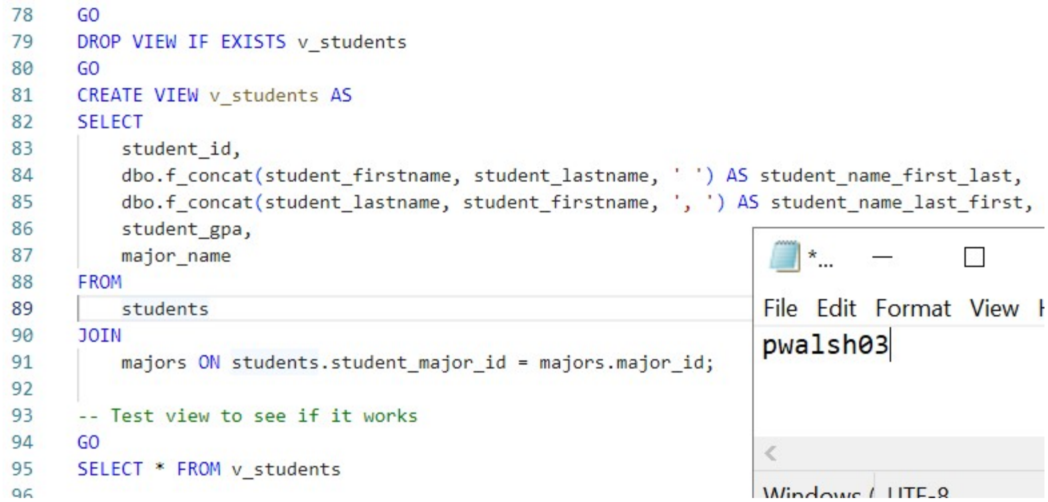
	(No column name)
1	half-baked

pwalsh03



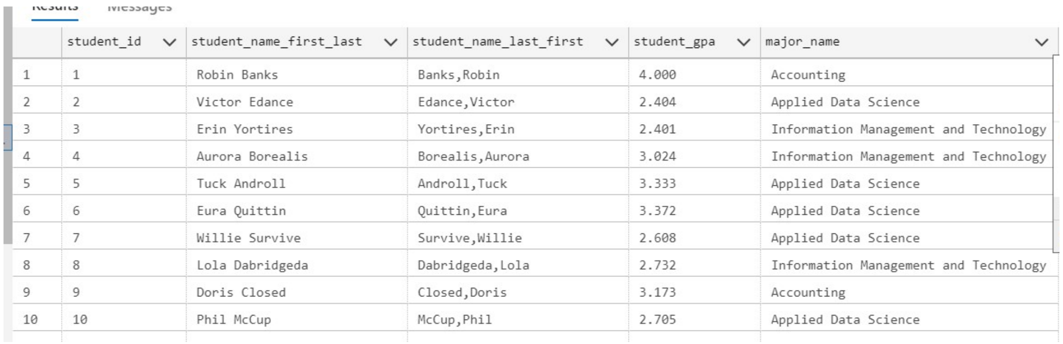
	(No column name)
1	mike fudge

pwalsh03



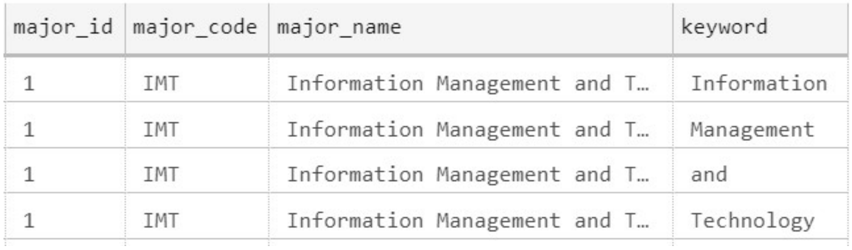
```
GO
DROP VIEW IF EXISTS v_students
GO
CREATE VIEW v_students AS
SELECT
    student_id,
    dbo.f_concat(student_firstname, student_lastname, ' ') AS student_name_first_last,
    dbo.f_concat(student_lastname, student_firstname, ', ') AS student_name_last_first,
    student_gpa,
    major_name
FROM
    students
JOIN
    majors ON students.student_major_id = majors.major_id;

-- Test view to see if it works
GO
SELECT * FROM v_students
```

pwalsh03



	student_id	student_name_first_last	student_name_last_first	student_gpa	major_name
1	1	Robin Banks	Banks,Robin	4.000	Accounting
2	2	Victor Edance	Edance,Victor	2.404	Applied Data Science
3	3	Erin Yortires	Yortires,Erin	2.401	Information Management and Technology
4	4	Aurora Borealis	Borealis,Aurora	3.024	Information Management and Technology
5	5	Tuck Androll	Androll,Tuck	3.333	Applied Data Science
6	6	Eura Quittin	Quittin,Eura	3.372	Applied Data Science
7	7	Willie Survive	Survive,Willie	2.608	Applied Data Science
8	8	Lola Dabridgeda	Dabridgeda,Lola	2.732	Information Management and Technology
9	9	Doris Closed	Closed,Doris	3.173	Accounting
10	10	Phil McCup	McCup,Phil	2.705	Applied Data Science

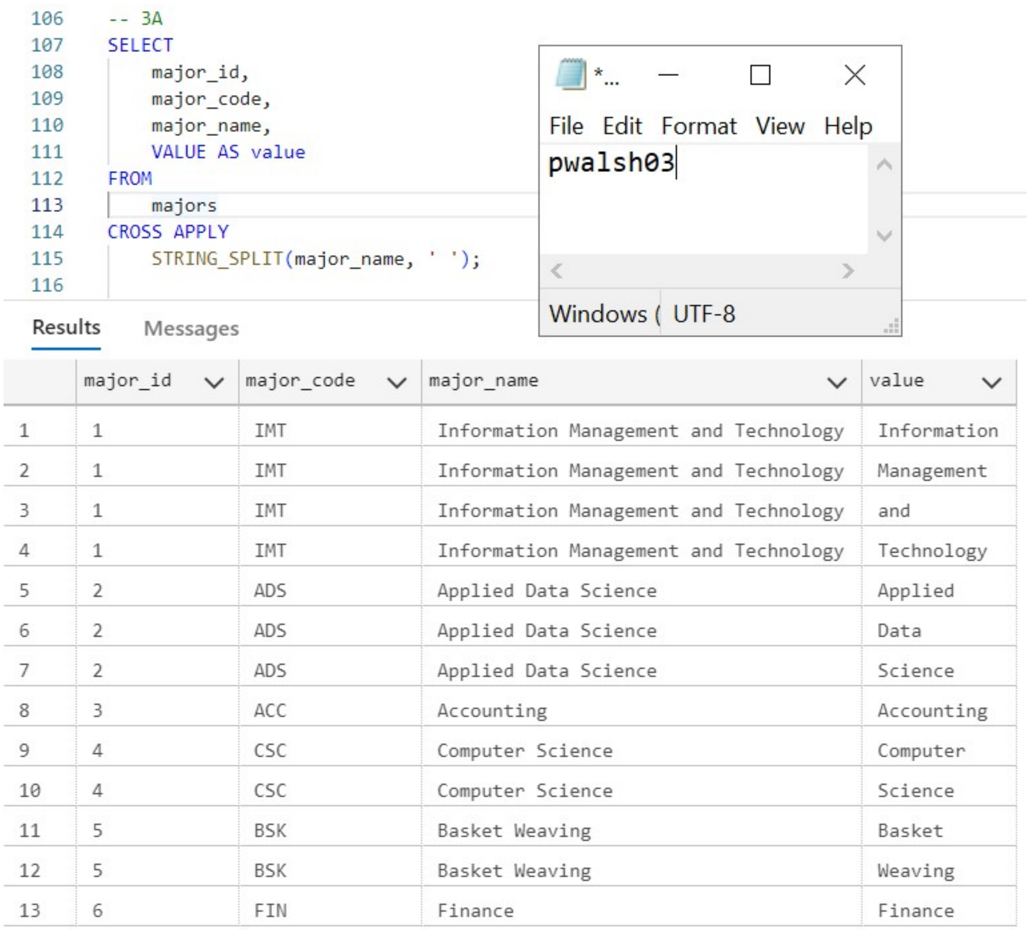
- In the **TinyU** database:
    - Write a query on the **majors** table so that the major_name is broken up into keywords, one per row. HINT: You must use string_split() with cross-apply.



major_id	major_code	major_name	keyword
1	IMT	Information Management and T…	Information
1	IMT	Information Management and T…	Management
1	IMT	Information Management and T…	and
1	IMT	Information Management and T…	Technology

    - Then use the query in 3.a to create a table-valued function **f_search_majors** that allows you to search the majors by keyword. Demonstrate calling the TVF by querying all majors with the “Science” keyword.

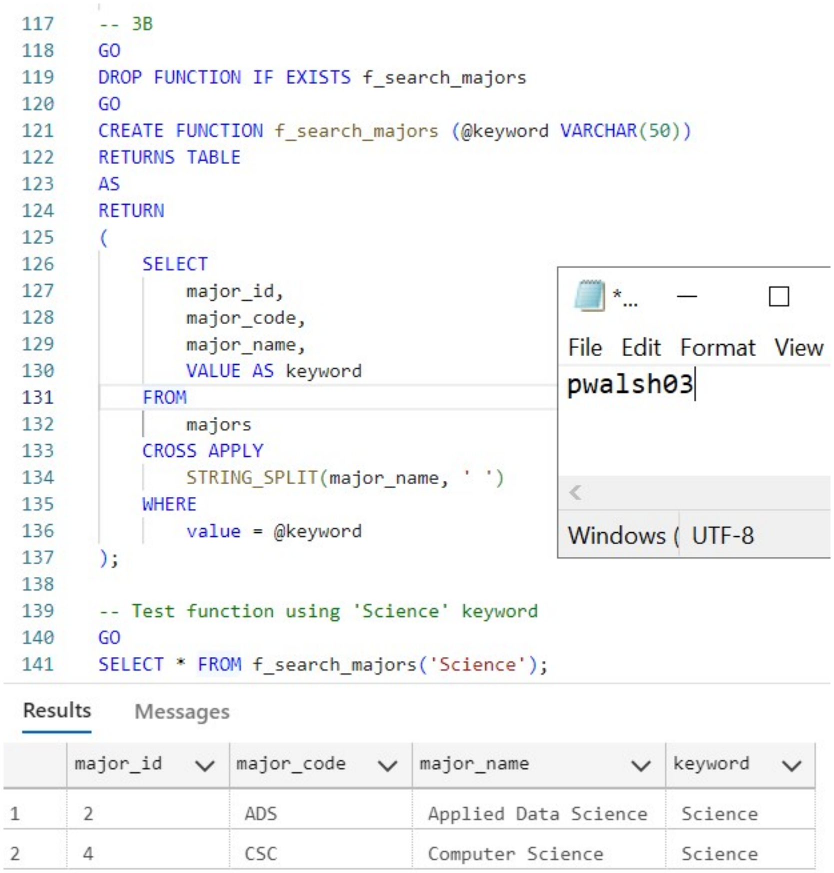
Your screen shot should include the query in 3.a Another screen shot should show the TVF in 3.b and the sample output from the SELECT statement calling the TVF.



```sql
-- 3A
SELECT
    major_id,
    major_code,
    major_name,
    VALUE AS value
FROM
    majors
CROSS APPLY
    STRING_SPLIT(major_name, ' ');
```

pwalsh03

	major_id	major_code	major_name	value
1	1	IMT	Information Management and Technology	Information
2	1	IMT	Information Management and Technology	Management
3	1	IMT	Information Management and Technology	and
4	1	IMT	Information Management and Technology	Technology
5	2	ADS	Applied Data Science	Applied
6	2	ADS	Applied Data Science	Data
7	2	ADS	Applied Data Science	Science
8	3	ACC	Accounting	Accounting
9	4	CSC	Computer Science	Computer
10	4	CSC	Computer Science	Science
11	5	BSK	Basket Weaving	Basket
12	5	BSK	Basket Weaving	Weaving
13	6	FIN	Finance	Finance



```
-- 3B
GO
DROP FUNCTION IF EXISTS f_search_majors
GO
CREATE FUNCTION f_search_majors (@keyword VARCHAR(50))
RETURNS TABLE
AS
RETURN
(
    SELECT
        major_id,
        major_code,
        major_name,
        VALUE AS keyword
    FROM
        majors
    CROSS APPLY
        STRING_SPLIT(major_name, ' ')
    WHERE
        value = @keyword
);

-- Test function using 'Science' keyword
GO
SELECT * FROM f_search_majors('Science');
```

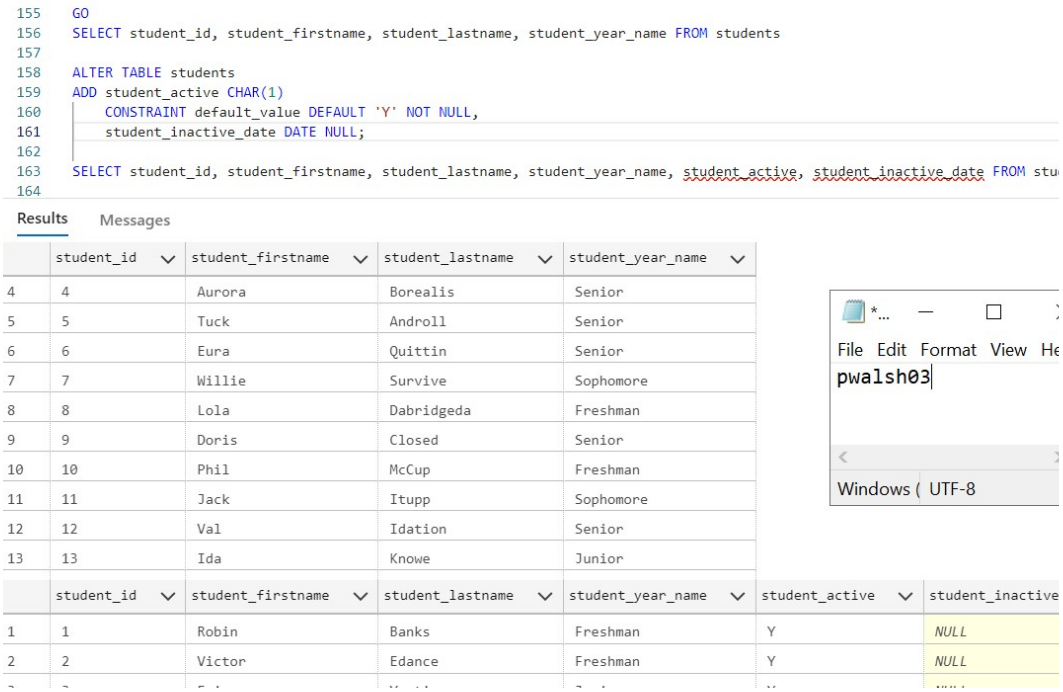
pwalsh03

Results Messages

	major_id	major_code	major_name	keyword
1	2	ADS	Applied Data Science	Science
2	4	CSC	Computer Science	Science

- In the **TinyU** database:
    - Alter the **students** table and add the following columns:
        - student_active char(1) default (‘Y’) not null
        - student_inactive_date date null
    - Create a trigger on the **students** table: when there is an student_inactive_date set, set student_active to ‘N’, and whenever there is not a student_inactive_date, then student_active is set to ‘Y’.
    - Write SQL code to deactivate all the ‘Graduate’ students with a date of ‘2020-08-01’.
    - Write SQL code to reactivate all the ‘Graduate’ students.

Provide a screen shot of your code from 4.a. and 4.b working. Provide another screen shot demonstrating 4.c worked. Then, provide a final screen shot of code and demonstration of 4.d working.

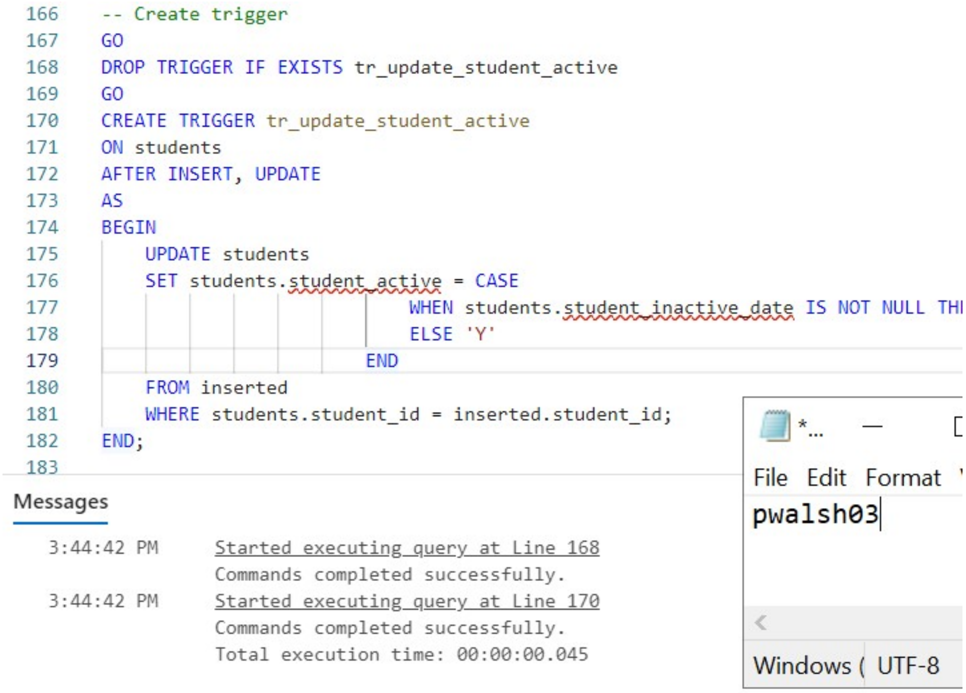


```
155    GO
156    SELECT student_id, student_firstname, student_lastname, student_year_name FROM students
157
158    ALTER TABLE students
159    ADD student_active CHAR(1)
160        CONSTRAINT default_value DEFAULT 'Y' NOT NULL,
161        student_inactive_date DATE NULL;
162
163    SELECT student_id, student_firstname, student_lastname, student_year_name, student_active, student_inactive_date FROM stu
164
```

	student_id	student_firstname	student_lastname	student_year_name
4	4	Aurora	Borealis	Senior
5	5	Tuck	Androll	Senior
6	6	Eura	Quittin	Senior
7	7	Willie	Survive	Sophomore
8	8	Lola	Dabridgeda	Freshman
9	9	Doris	Closed	Senior
10	10	Phil	McCup	Freshman
11	11	Jack	Itupp	Sophomore
12	12	Val	Idation	Senior
13	13	Ida	Knowe	Junior

	student_id	student_firstname	student_lastname	student_year_name	student_active	student_inactive
1	1	Robin	Banks	Freshman	Y	NULL
2	2	Victor	Edance	Freshman	Y	NULL

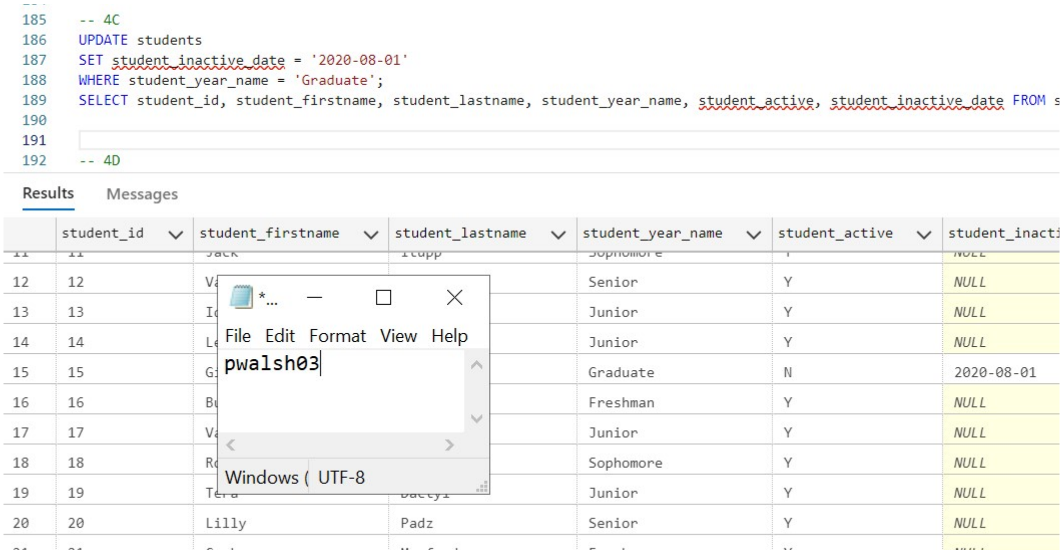
pwalsh03



```
166    -- Create trigger
167    GO
168    DROP TRIGGER IF EXISTS tr_update_student_active
169    GO
170    CREATE TRIGGER tr_update_student_active
171    ON students
172    AFTER INSERT, UPDATE
173    AS
174    BEGIN
175        UPDATE students
176        SET students.student_active = CASE
177                                          WHEN students.student_inactive_date IS NOT NULL THI
178                                          ELSE 'Y'
179                                      END
180        FROM inserted
181        WHERE students.student_id = inserted.student_id;
182    END;
183
```

```
3:44:42 PM     Started executing query at Line 168
               Commands completed successfully.
3:44:42 PM     Started executing query at Line 170
               Commands completed successfully.
               Total execution time: 00:00:00.045
```

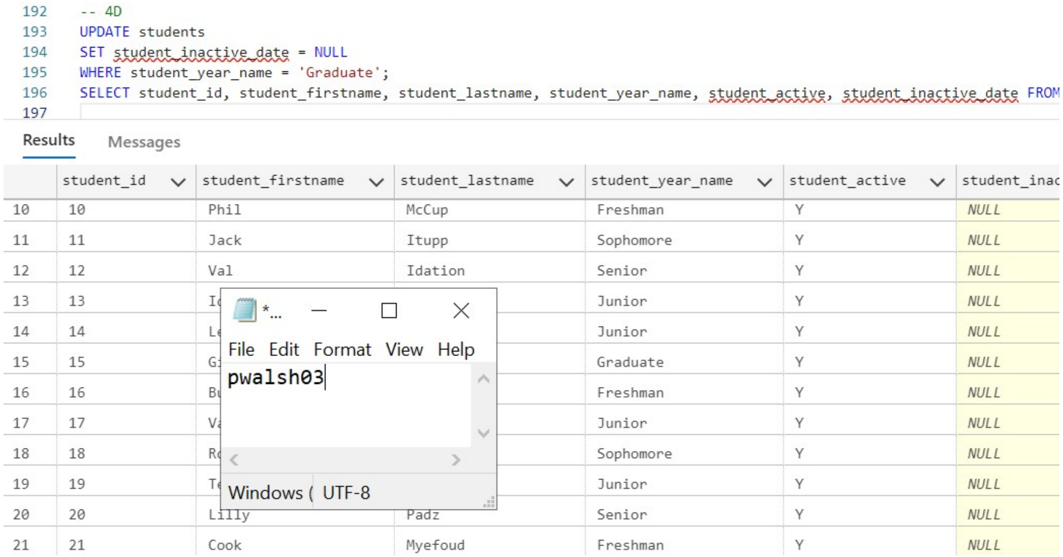
pwalsh03



```
-- 4C
UPDATE students
SET student_inactive_date = '2020-08-01'
WHERE student_year_name = 'Graduate';
SELECT student_id, student_firstname, student_lastname, student_year_name, student_active, student_inactive_date FROM s

-- 4D
```

pwalsh03

	student_id	student_firstname	student_lastname	student_year_name	student_active	student_inacti
11	11	Jack	Itupp	Sophomore	Y	NULL
12	12	Va		Senior	Y	NULL
13	13	Id		Junior	Y	NULL
14	14	Le		Junior	Y	NULL
15	15	Gi		Graduate	N	2020-08-01
16	16	Bu		Freshman	Y	NULL
17	17	Va		Junior	Y	NULL
18	18	Ro		Sophomore	Y	NULL
19	19	Tera	Dactyl	Junior	Y	NULL
20	20	Lilly	Padz	Senior	Y	NULL



```
-- 4D
UPDATE students
SET student_inactive_date = NULL
WHERE student_year_name = 'Graduate';
SELECT student_id, student_firstname, student_lastname, student_year_name, student_active, student_inactive_date FROM
```

pwalsh03

	student_id	student_firstname	student_lastname	student_year_name	student_active	student_inac
10	10	Phil	McCup	Freshman	Y	NULL
11	11	Jack	Itupp	Sophomore	Y	NULL
12	12	Val	Idation	Senior	Y	NULL
13	13	Id		Junior	Y	NULL
14	14	Le		Junior	Y	NULL
15	15	Gi		Graduate	Y	NULL
16	16	Bu		Freshman	Y	NULL
17	17	Va		Junior	Y	NULL
18	18	Ro		Sophomore	Y	NULL
19	19	Te		Junior	Y	NULL
20	20	Lilly	Padz	Senior	Y	NULL
21	21	Cook	Myefoud	Freshman	Y	NULL