**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**---□**🕮**□---**

A picture containing text, sign

Description automatically generated

**Database Lab**

**Student Management System (SMS)**

**Group 5**

**Lecturer: Vu Tuyet Trinh**

**Project Report**

**Class: 135409**

**Group: 5**

**Lecturer: Vu Tuyet Trinh**

# Members

| **Name** | **Student ID** | **Tasks** |
| --- | --- | --- |
| Le Minh Duc | 20200164 | Design, Generate data, Optimize queries, Grant users, Code demo |
| Tong Tran Minh Duc | 20205147 | Write functions (2.1.1 → 2.2.3), Create triggers |
| Hoang Van Phuong | 20200478 | Write functions (2.2.4 → 2.3.5), Create views |

# 

# Table of Contents

[**Members 2**](#_862j3orsnb9r)

[**Table of Contents 3**](#_m1q6nmcdpmx1)

[**Title 3**](#_r0j6a0nxrypu)

[**1. Context 3**](#_quvwl6p2tp98)

[**2. Requirements 3**](#_pjkl2zifdwc5)

[**3. Database Structure 5**](#_i4s33rq9agpd)

[3.1. ERD 5](#_cllctxd68p6)

[3.2. Schema 5](#_8yujx7lsdud7)

[3.3. Tables 6](#_obw831zcjzof)

[**4. Generate Data 6**](#_jii1w165ancr)

[**5. Views 7**](#_8jumm34jh2y1)

[**6. Constraints & Triggers 7**](#_85dak7azy58b)

[6.1. Constraints 7](#_1kn42eqiz8rm)

[6.2. Triggers 8](#_9hzs5m5vh0sg)

[**7. Users 8**](#_c5d1quus8lsx)

[**8. Results 8**](#_yjx1o8atna71)

[2.1. Staff (Admin) 9](#_ccjek1bgp841)

[2.2. Students 13](#_lryzxyp7vpdr)

[2.3. Lecturers 18](#_x142r5carftn)

[**9. Optimizations 21**](#_qjkjta1eoq17)

[**10. Conclusion 25**](#_uoanjpgs429c)

# Title

Student Management System (SMS)

# 1. Context

A Student Management System (SMS) is a database-driven application that assists educational institutions in digitizing student data and managing it more efficiently. The system creates a simple interface for students, lecturers, and administrative staff, offering great assistance within and outside classrooms. In this project, we want to develop a simple and effective database, which will then be connected to an application to simulate real-life situations.

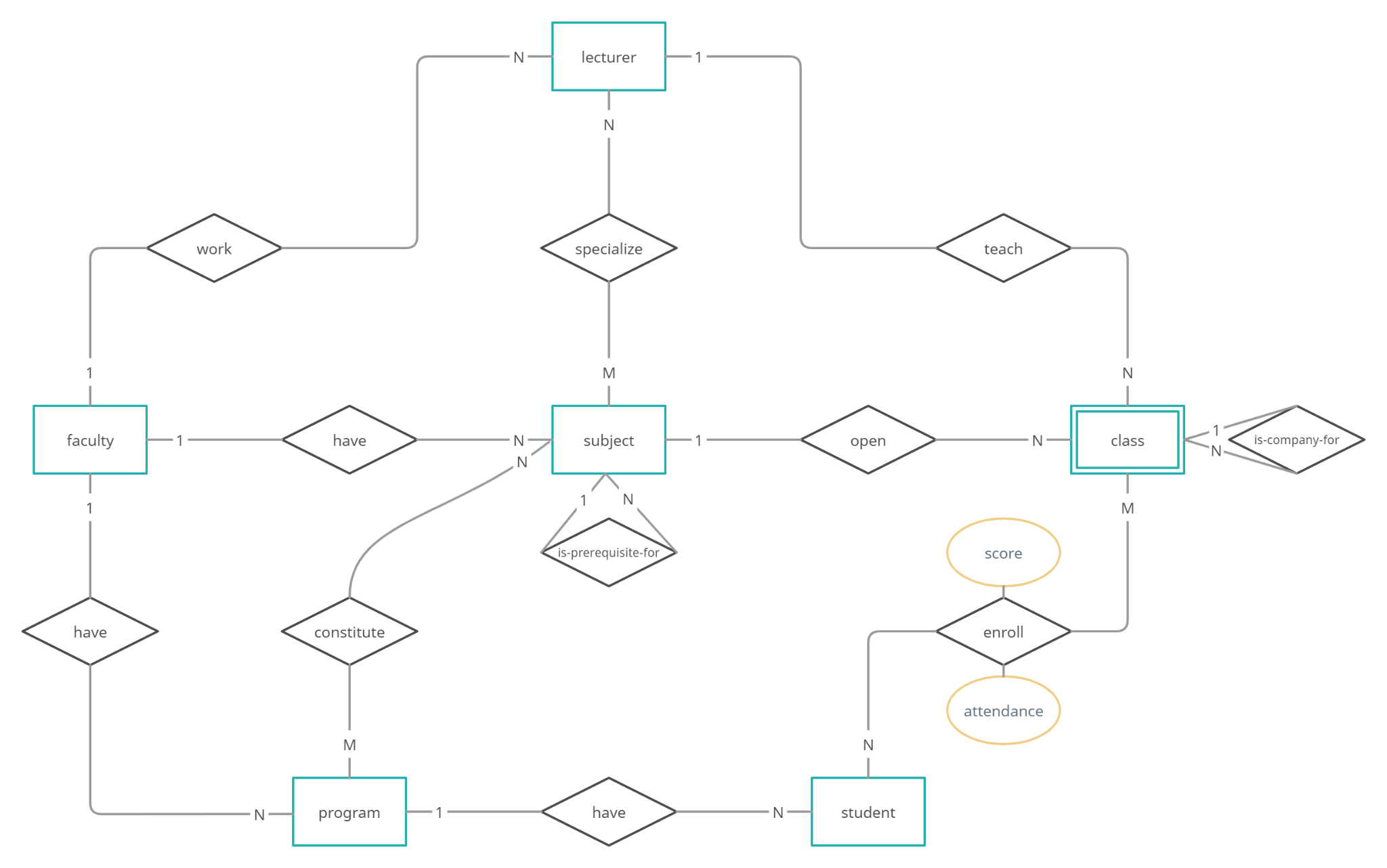
# 2. Requirements

The system should provide different users with different capabilities:

1. Staff (Admin)
2. Adding students, lecturers, subjects, and classes with the following data.
   1. Students: Name, ID, School/Faculty, Program, Date of Birth, Address, Contact, etc.
   2. Lecturers: Name, ID, School/Faculty, Teaching Subjects, Date of Birth, Address, Contact, etc.
   3. Subjects: Prerequisites, Number of credits, etc.
   4. Classes: Time, Location, Slot Availability, Semester, Number of credits, Schedule of classes, etc.
3. Managing information of students, lecturers, subjects, and classes.
4. Creating a tentative timetable for the upcoming semester, i.e creating classes for each subject. A subject can have 0, 1, or many classes open in a semester.
5. Making changes to the timetable.
6. Assigning lecturers to classes based on the timetable.
7. Getting reports on
   1. Classes having a number of enrolled students fewer than [X], used for canceling classes with low enrollment.
   2. Students with credit debt from unfinished (failed) subjects in the range [A, B], used for sending warnings.
   3. Students qualified for semester scholarships with GPA ≥ [G].
8. Students
9. Viewing data of subjects, classes, and results of themselves.
10. View the tentative timetable to find suitable opening classes.
11. Enrolling in classes.
    1. Showing enrolling information such as time, location, credit, slot availability, and prerequisites.
    2. Check for slot availability, and class prerequisites.
    3. Check for time conflicts.
    4. Calculate total studying credits.
    5. Schedules are automatically identified after students enroll in classes.
12. Looking up data related to lecturers, subjects, classes, and other students. Retrieve only essential information, excluding personal ones like address, scores, and student timetable.
    1. Students: Name, ID, School/Faculty, Program, Contact, etc.
    2. Lecturers: Name, ID, School/Faculty, Teaching Subjects, Contact, etc.
    3. Subjects, Classes: Number of credits, Schedule of classes, etc.
13. Getting estimated fees for the current semester.
    1. **Total Fees = Tuition Credits \* Credit Price + Other Fees.**
    2. The Credit Price can vary between programs.
    3. Other Fees can be Insurance Fees, Previous Debt, etc.
    4. Tuition Credits ≠ Study Credits.
14. Getting reports on Study Credits earned, GPA/CPA.
15. Lecturers
16. Viewing data of all teaching subjects, classes, and results of themselves.
17. Looking up data related to students, subjects, classes, and other lecturers. Retrieve only essential information, excluding personal ones like address, scores, and student timetable.
    1. Students: Name, ID, School/Faculty, Program, Contact, etc.
    2. Lecturers: Name, ID, School/Faculty, Teaching Subjects, Contact, etc.
    3. Subjects, Classes: Number of credits, Schedule of classes, etc.
18. Recording student academic performance.
19. Tracking student attendance.
20. Getting reports on
    1. Exam grade distribution (statistics)
    2. Student attendance up to today

# 3. Database Structure

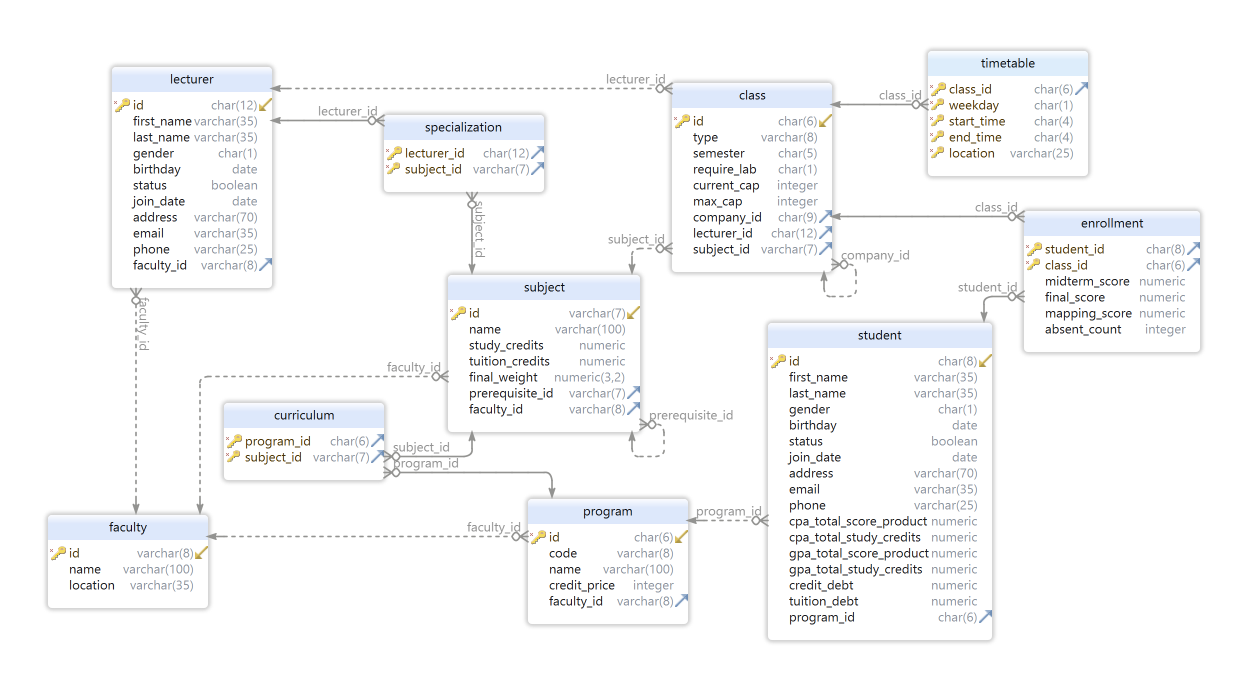
## 3.1. ERD



* Summary of changes:
  + [ERD\_v1](https://github.com/duclm278/database-project/blob/main/structure/erd_v1.jpg): Initial design. The multivalued attributes aren’t highlighted.
  + [ERD\_v2](https://github.com/duclm278/database-project/blob/main/structure/erd_v2.jpg): Added relation between “subject” and “program”. **This diagram doesn’t include attributes**, leading to mapping mistakes of multivalued attributes in the “class” table as can be seen in [SCHEMA\_v1](https://github.com/duclm278/database-project/blob/main/structure/schema_v1.png). Fixed in later schema’s versions.
* Check GitHub for more details:

[<https://github.com/duclm278/database-project/tree/main/structure>]

## 3.2. Schema



* Summary of changes:
  + [SCHEMA\_v1](https://github.com/duclm278/database-project/blob/main/structure/schema_v1.png): Initial mapping from [ERD\_v2](https://github.com/duclm278/database-project/blob/main/structure/erd_v2.jpg).
  + [SCHEMA\_v2](https://github.com/duclm278/database-project/blob/main/structure/schema_v2.png): Fixed mapping mistakes of multivalued attributes in the “class” table by creating the “timetable” table. Add the “curriculum” table. Add cpa, gpa attributes to the “student” table for storing results of the update\_score trigger.
  + [SCHEMA\_v3](https://github.com/duclm278/database-project/blob/main/structure/schema_v3.png): Separate cpa, gpa attributes into their corresponding total\_score\_product and total\_study\_credits attributes. This helps the trigger update cpa and gpa easier.
  + [SCHEMA\_v4](https://github.com/duclm278/database-project/blob/main/structure/schema_v4.png): Add final\_score attribute to the “enrollment” table.
* Check GitHub for more details:

[<https://github.com/duclm278/database-project/tree/main/structure>]

## 3.3. Tables

* Tables are created in the schema “public”.
* Check GitHub for more details:

[<https://github.com/duclm278/database-project/blob/main/sql/sms-create.sql>]

# 4. Generate Data

* Use Python as the main programming language.
* Use Psycopg 2 to connect to and control the database.
* Use Faker to generate data.
* Some details about generated data:
  + 18 faculties, 64 programs, 2361 subjects
  + 12786 classes (5254 classes in 20212 & 7532 classes in 20221)
  + 14894 timetable slots (6096 slots in 20212 & 8798 slots in 20221)
  + Each lecturer is assigned to a maximum of 5 subjects and 25 classes.
  + Each curriculum has a maximum of 50 subjects.
  + Generate 5000 students per year in the total of three years (2020, 2021, 2022).
  + Enroll these students to classes of the semester 20212 (maximum 10 subjects).
* Check GitHub for more details:

[<https://github.com/duclm278/database-project/blob/main/gen.py>]

# 5. Views

There are 4 views for full self info and 3 views for restricted searchings:

* student.self\_view\_info
* student.self\_view\_curriculum
* lecturer.self\_view\_info
* lecturer.self\_view\_specializations
* search.view\_search\_student
* search.view\_search\_lecturer
* search.view\_search\_lecturer\_specialization

Other restricted searchings are implemented by functions that support custom inputs.

Check GitHub for more details:

[<https://github.com/duclm278/database-project/blob/main/sql/sms-view.sql>]

# 6. Constraints & Triggers

## 6.1. Constraints

Suitable constraints beside primary and foreign keys are added while creating tables:

* constraint ck\_lecturer\_gender check (gender in ('F', 'M'))
* constraint ck\_program\_credit\_price check ((credit\_price >= 0))
* constraint ck\_student\_cpa\_total\_score\_product check (cpa\_total\_score\_product >= 0),
* constraint ck\_student\_cpa\_total\_study\_credits check (cpa\_total\_study\_credits >= 0),
* constraint ck\_student\_gpa\_total\_score\_product check (gpa\_total\_score\_product >= 0),
* constraint ck\_student\_gpa\_total\_study\_credits check (gpa\_total\_study\_credits >= 0),
* constraint ck\_student\_credit\_debt check (credit\_debt >= 0)
* constraint ck\_student\_study\_credits check (study\_credits >= 0),
* constraint ck\_student\_tuition\_credits check (tuition\_credits >= 0),
* constraint ck\_subject\_final\_weight check (final\_weight >= 0 and final\_weight <= 1)
* constraint ck\_require\_lab\_same\_term check (require\_lab in ('Y', 'N')),
* constraint ck\_class\_current\_cap check (current\_cap >= 0 and current\_cap <= max\_cap)
* constraint ck\_timetable\_weekday check (weekday in ('2', '3', '4', '5', '6', '7', '8')),
* constraint ck\_timetable\_start\_time check (start\_time < end\_time)
* constraint ck\_enrollment\_midterm\_score check (midterm\_score >= 0 and midterm\_score <= 10),
* constraint ck\_enrollment\_final\_score check (final\_score >= 0 and final\_score <= 10),
* constraint ck\_enrollment\_absent\_count check (absent\_count >= 0)

Check GitHub for more details:

[<https://github.com/duclm278/database-project/blob/main/sql/sms-create.sql>]

## 6.2. Triggers

There are 10 triggers used to maintain the integrity of the data.

* timetable\_conflict\_trigger
* teaching\_conflict\_trigger
* convert\_score\_trigger
* update\_score\_trigger
* check\_cap\_trigger
* check\_prerequisite\_trigger
* check\_lab\_trigger
* check\_time\_enrolled\_trigger
* credits\_trigger
* company\_id\_trigger

Check GitHub for more details:

[<https://github.com/duclm278/database-project/blob/main/sql/sms-staff.sql>]

[<https://github.com/duclm278/database-project/blob/main/sql/sms-student.sql>]

# 7. Users

Beside the default admin user, the demo also creates 2 more users with suitable rights:

* Grant read-only privilege on all tables to student “20200164”
* Grant all privileges on table “enrollment” to student “20200164”
* Revoke privilege on table “student”, “lecturer”
* Instead grant all privileges on schema “student” (having self\_view views, custom functions)
* Grant all privileges on schema “search” (having restricted searching views)

Similarly, grant suitable rights to the lecturer having id “aaaaaaaaaaaa”.

Check GitHub for more details:

[<https://github.com/duclm278/database-project/blob/main/sql/sms-demo.sql#L3-L37>]

# 8. Results

Check GitHub for more details about the demo below:

[<https://github.com/duclm278/database-project/blob/main/sql/sms-demo.sql>]

## Staff (Admin)

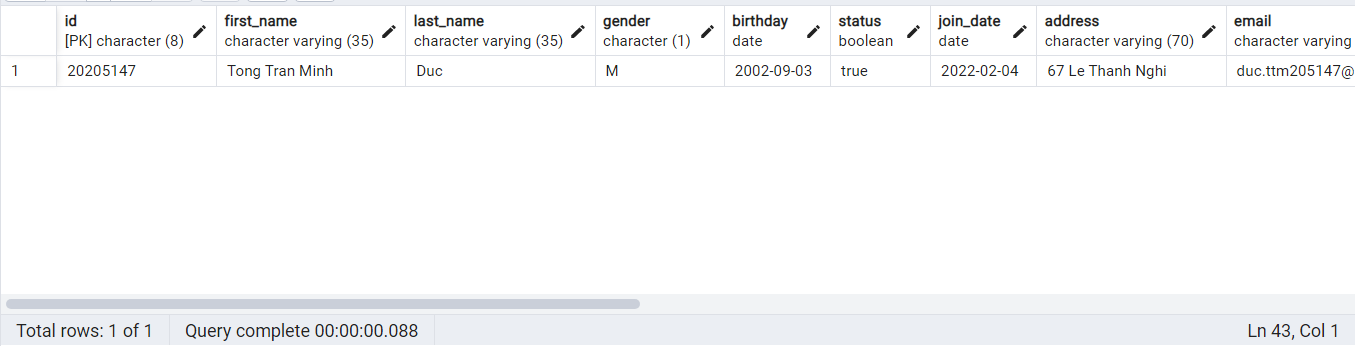
Login as “**postgres”**.

Check GitHub for more details about functions and triggers used for staff:

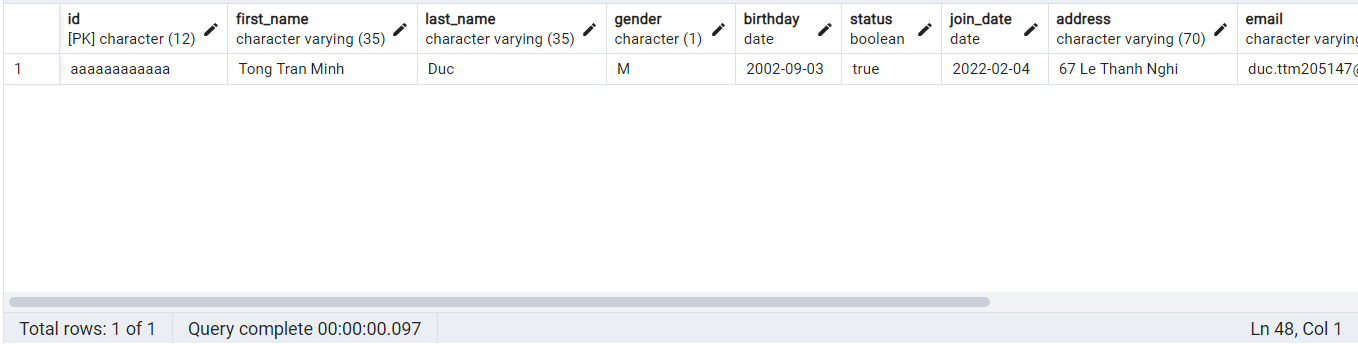
[<https://github.com/duclm278/database-project/blob/main/sql/sms-staff.sql>]

1. Adding students, lecturers, subjects, and classes with the following data.

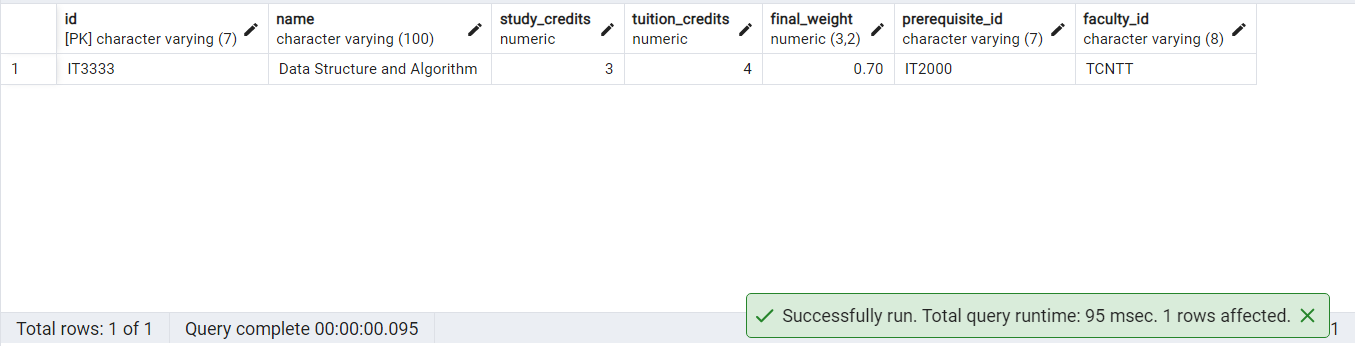
| CALL add\_student('20205147', 'Tong Tran Minh', 'Duc', 'M', '2002-09-03', true, '2022-02-04', '67 Le Thanh Nghi', 'duc.ttm205147@sis.hust.edu.vn', '0902112042', '509463'); SELECT \* FROM student WHERE id LIKE '20205147'; |
| --- |



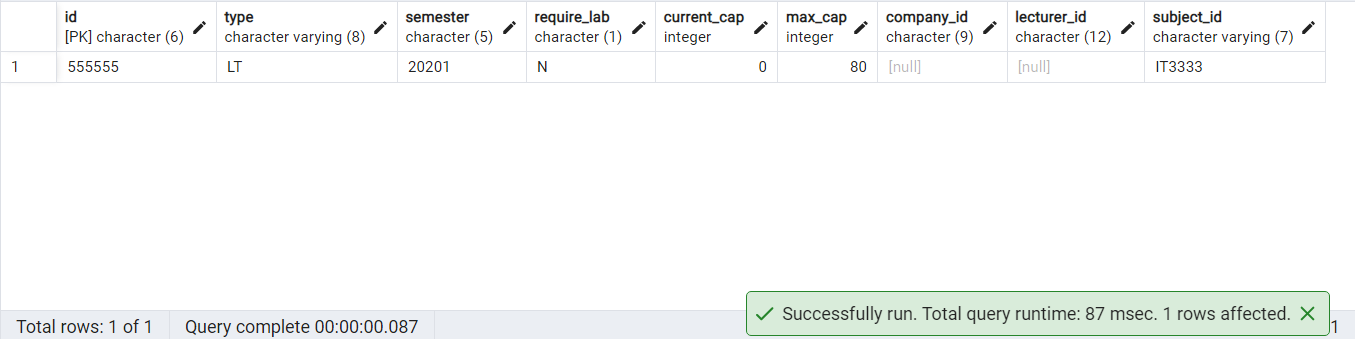
| CALL add\_lecturer('aaaaaaaaaaaa', 'Tong Tran Minh', 'Duc', 'M', '2002-09-03', true, '2022-02-04', '67 Le Thanh Nghi', 'duc.ttm205147@sis.hust.edu.vn', '0902112042', 'TCNTT'); SELECT \* FROM lecturer WHERE id LIKE 'aaaaaaaaaaaa'; |
| --- |



| CALL add\_subject('IT3333', 'Data Structure and Algorithm', 3, 4, 0.7, 'IT2000', 'TCNTT'); SELECT \* FROM subject WHERE id LIKE 'IT3333'; |
| --- |

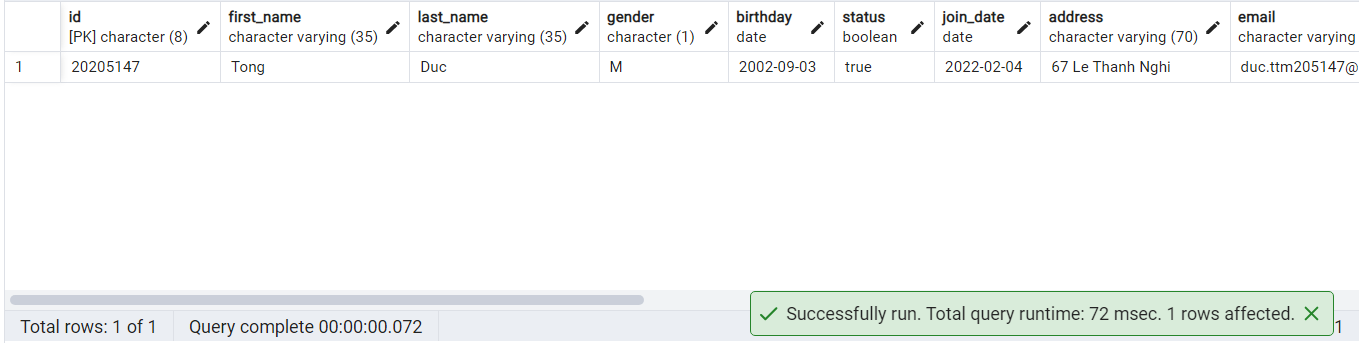


| CALL add\_class('555555', 'LT', '20201', 'N', 0, 80, NULL, NULL, 'IT3333'); SELECT \* FROM class WHERE id LIKE '555555'; |
| --- |



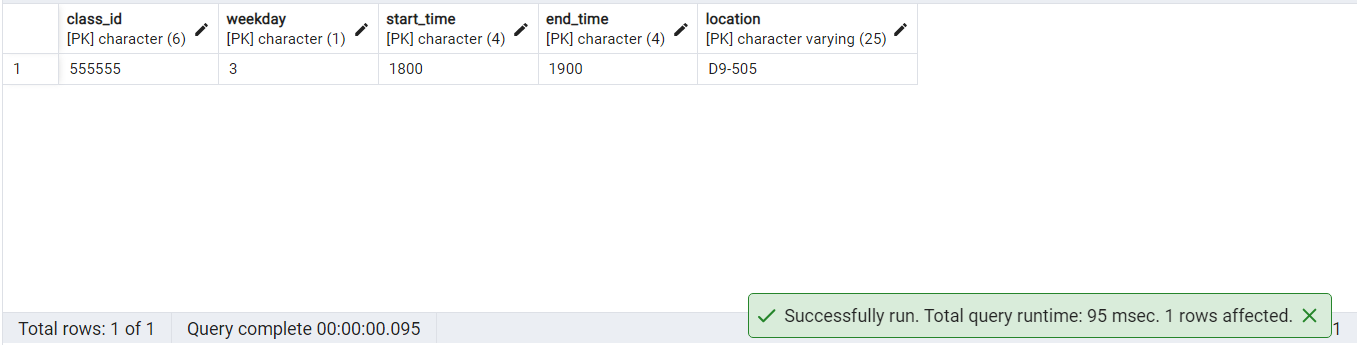
1. Managing information of students, lecturers, subjects, and classes.

| UPDATE student SET first\_name = 'Tong' WHERE id = '20205147'; SELECT \* FROM student WHERE id LIKE '20205147'; |
| --- |



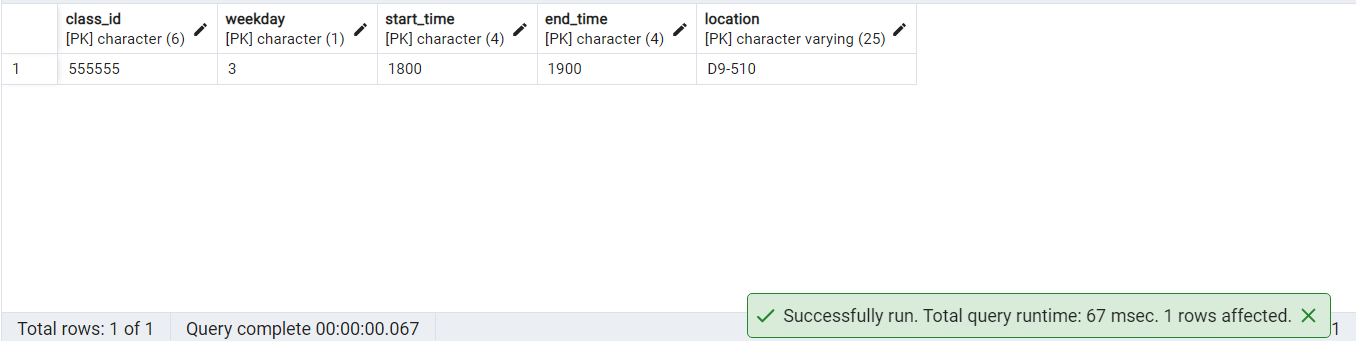
1. Creating a tentative timetable for the upcoming semester, i.e creating classes for each subject. A subject can have 0, 1, or many classes open in a semester.

| CALL add\_timetable('555555', '3', '1800', '1900', 'D9-505'); SELECT \* FROM timetable WHERE class\_id LIKE '555555'; |
| --- |



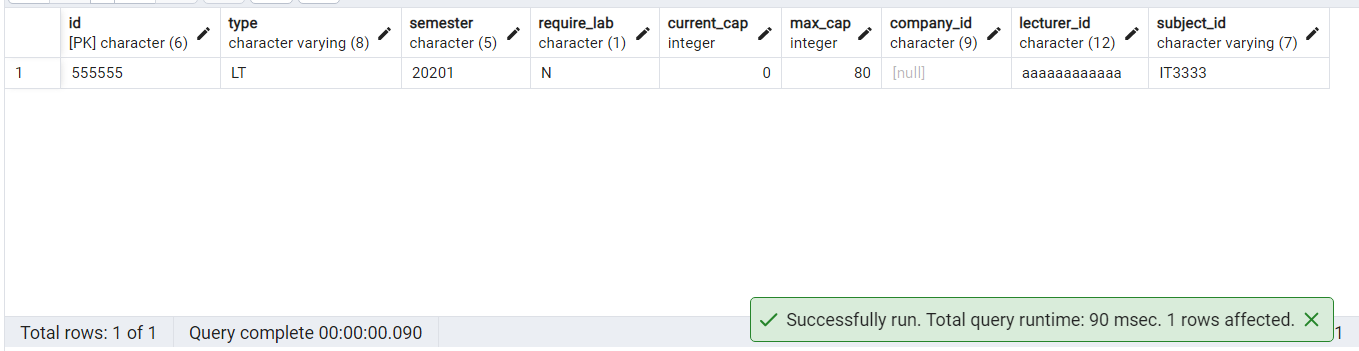
1. Making changes to the timetable.

| UPDATE timetable SET location = 'D9-510' WHERE class\_id = '555555'; SELECT \* FROM timetable WHERE class\_id LIKE '555555'; |
| --- |



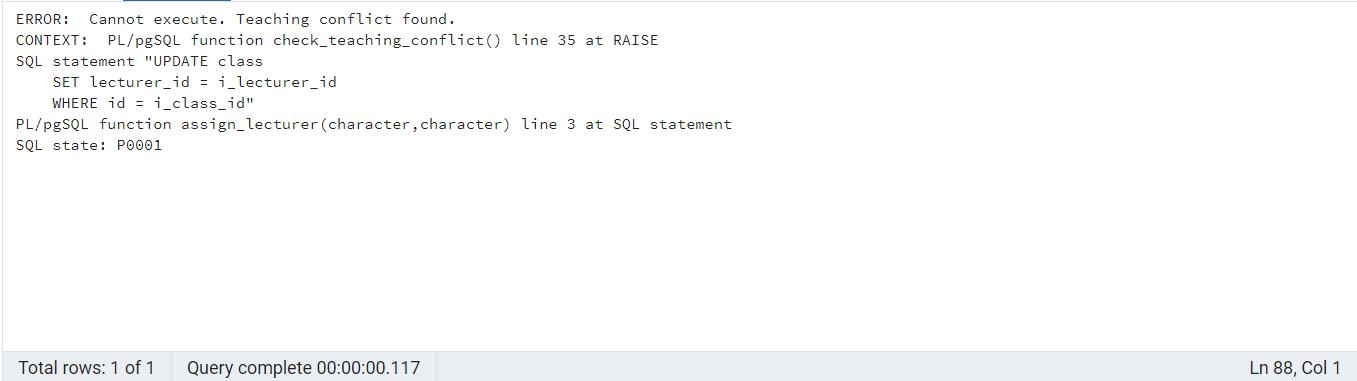
1. Assigning lecturers to classes based on the timetable.

| CALL assign\_lecturer('aaaaaaaaaaaa', '555555'); SELECT \* FROM class WHERE lecturer\_id LIKE 'aaaaaaaaaaaa'; |
| --- |



**Test teaching conflict TRIGGER:**

| CALL add\_class('666666', 'LT', '20201', 'N', 0, 80, NULL, NULL, 'IT3333'); CALL add\_timetable('666666', '3', '1830', '1930', 'D9-505'); CALL assign\_lecturer('aaaaaaaaaaaa', '666666'); |
| --- |

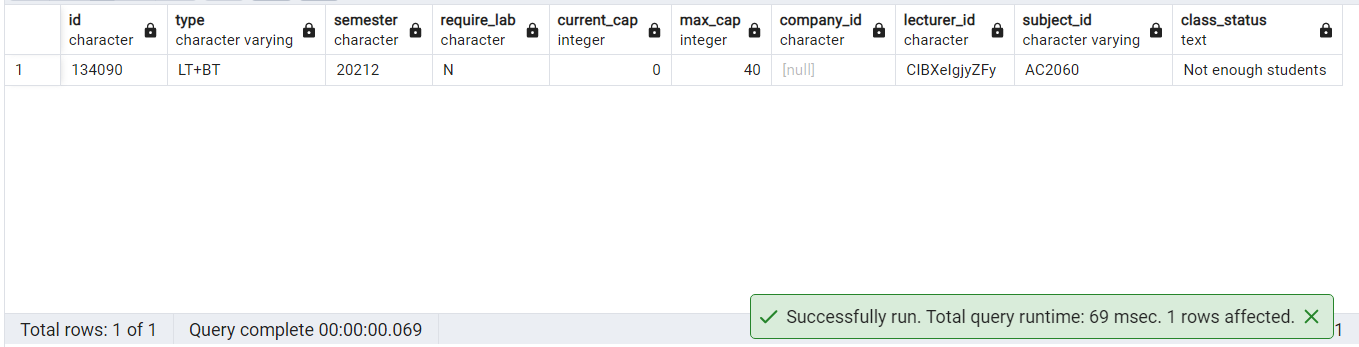


1. Getting reports on
   1. Classes having a number of enrolled students fewer than [X], used for canceling classes with low enrollment.

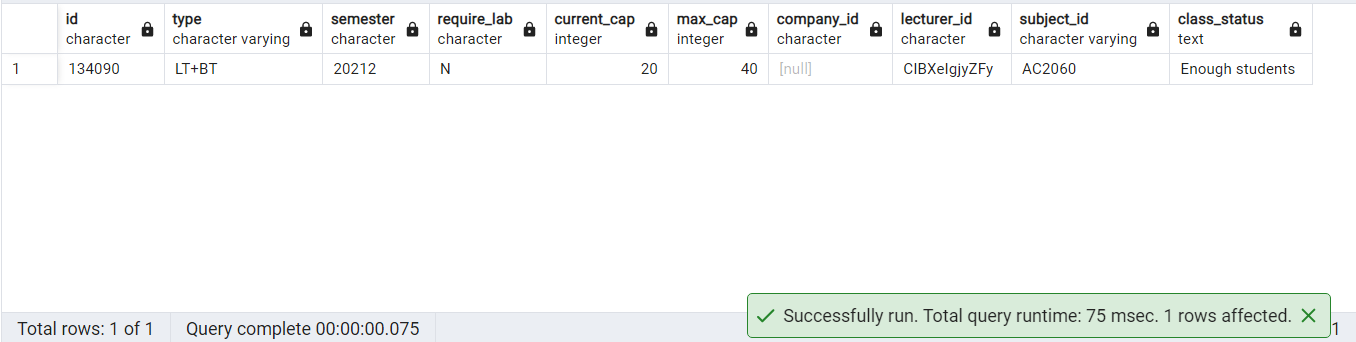
**Update the current\_cap of the class “124090” to have the status “Enough students”**

| SELECT \* FROM report\_enrolled('20212') WHERE id = '134090'; UPDATE class SET current\_cap = max\_cap / 2 WHERE id = '134090'; SELECT \* FROM report\_enrolled('20212') WHERE id = '134090'; |
| --- |

**BEFORE:**

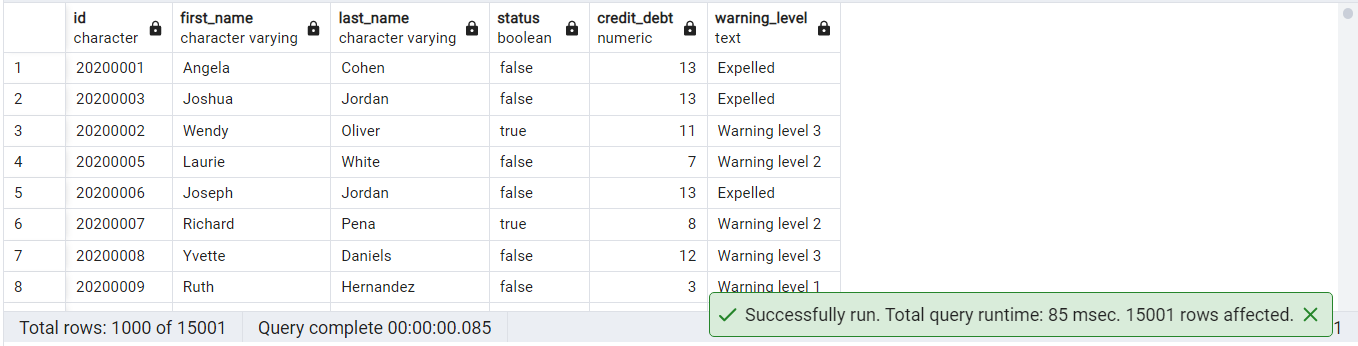


**AFTER:**

****

* 1. Students with credit debt from unfinished (failed) subjects in the range [A, B], used for sending warnings.

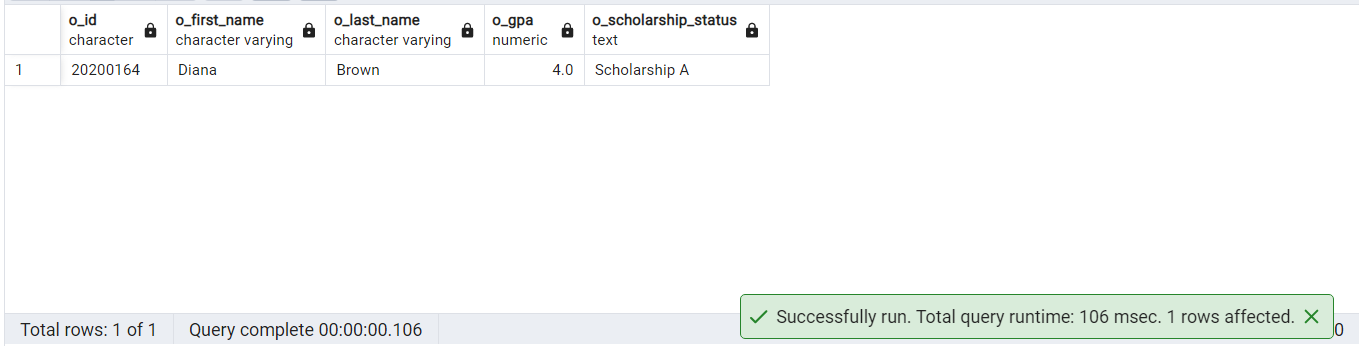
| SELECT \* FROM report\_credit\_debt(); |
| --- |



* 1. Students qualified for semester scholarships with GPA ≥ [G].

**Make the student 20200164 enroll in some classes and get all A+.**

| CALL student.enroll\_class('20200164', '135387'); -- enroll\_class() in 2.2.3f. CALL student.enroll\_class('20200164', '135404'); CALL student.enroll\_class('20200164', '135406'); CALL student.enroll\_class('20200164', '135409'); CALL student.enroll\_class('20200164', '135410'); CALL student.enroll\_class('20200164', '135411'); CALL student.enroll\_class('20200164', '135469'); CALL student.enroll\_class('20200164', '722873'); UPDATE enrollment SET midterm\_score = 10, final\_score = 10 WHERE student\_id = '20200164'  AND class\_id IN ('135387', '135404', '135406', '135409', '135410', '135411', '135469', '722873');  SELECT \* FROM report\_scholarship() WHERE o\_id = '20200164'; |
| --- |



## Students

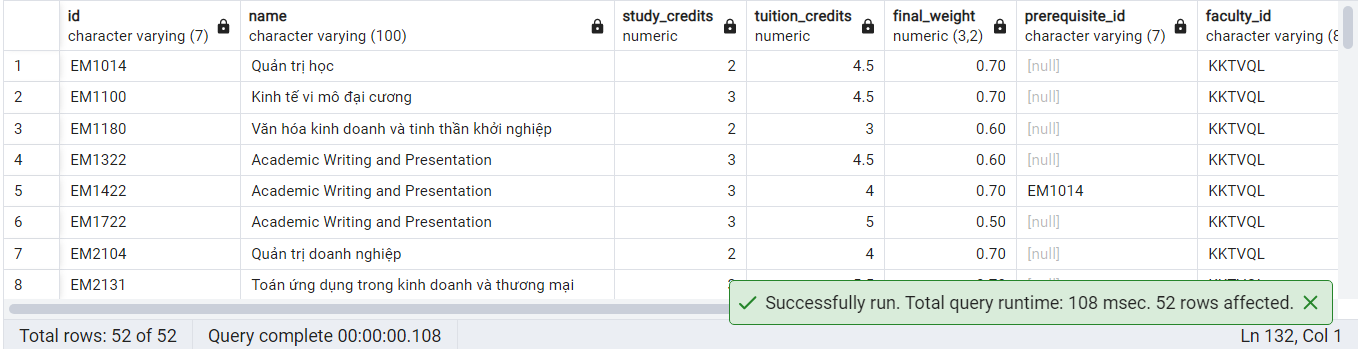
Login as “**20200164”**. Pass is “**demo**”.

Check GitHub for more details about functions and triggers used for students:

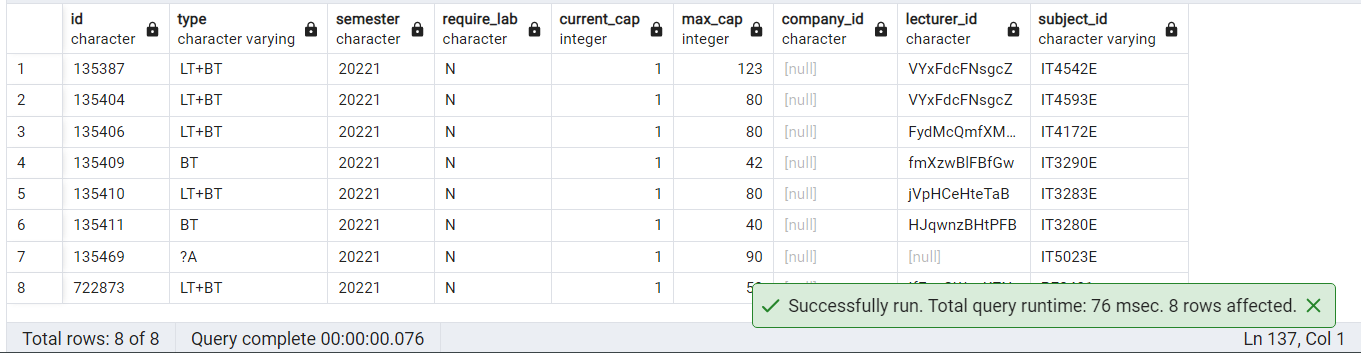
[<https://github.com/duclm278/database-project/blob/main/sql/sms-student.sql>]

1. Viewing data of subjects, classes, and results of themselves.

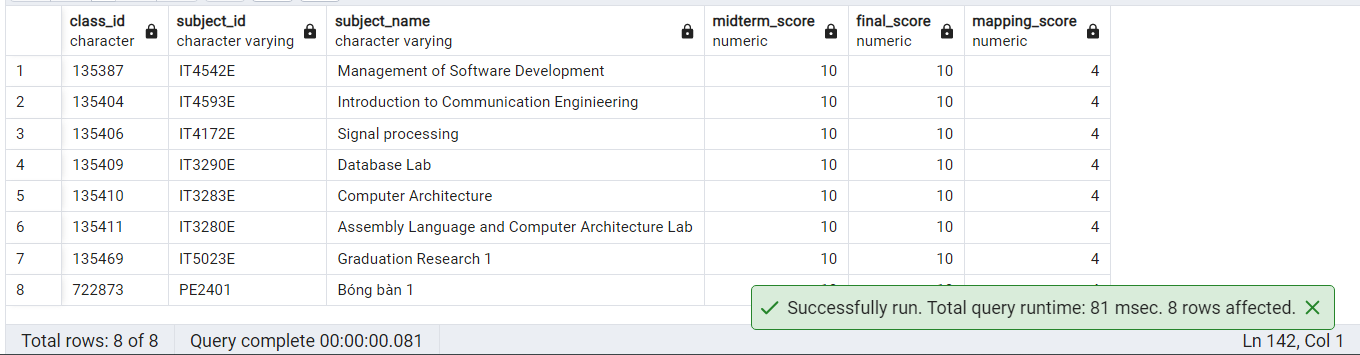
| -- View all subjects of their curriculum SELECT \* FROM student.self\_view\_curriculum; -- As 20200164 |
| --- |



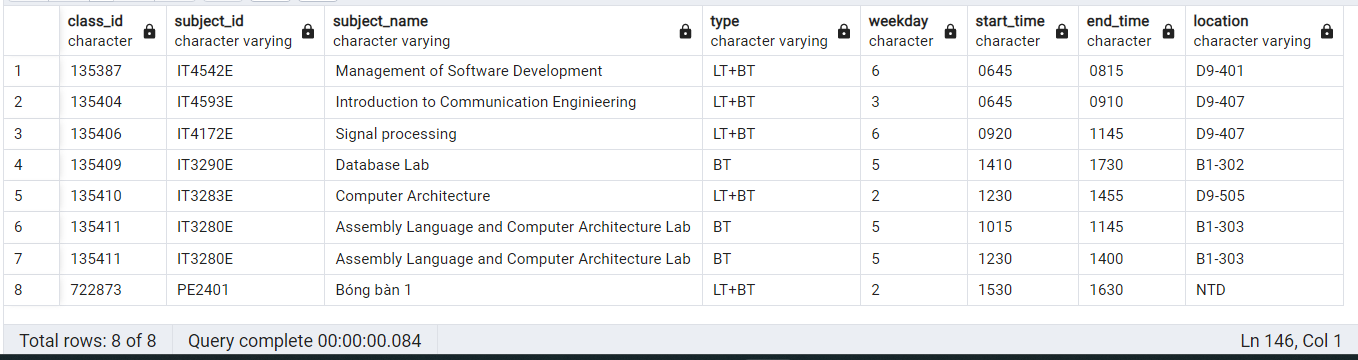
| -- View their classes in of any semesters SELECT \* FROM student.self\_view\_class\_enrolled('20221'); -- As 20200164 |
| --- |



| -- View their results in of any semesters SELECT \* FROM student.self\_view\_results('20221'); -- As 20200164 |
| --- |

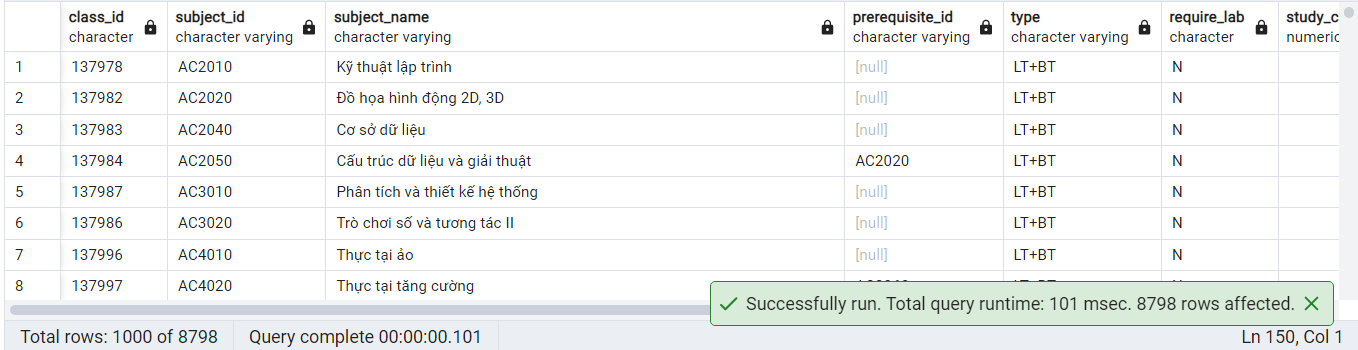


| -- View their timetable in of a semester SELECT \* FROM student.self\_view\_timetable('20221'); |
| --- |



1. View the tentative timetable to find suitable opening classes.

| SELECT \* FROM student.show\_class\_info(NULL, '20221'); -- All classes in 20221 |
| --- |



1. Enrolling in classes.

-- **Done in 2.1.6c.**

* 1. Showing enrolling information such as time, location, credit, slot availability, and prerequisites.

-- **Done in 2.2.2.**

* 1. Check for slot availability, and class prerequisites.

-- **Use TRIGGER check\_cap\_trigger BEFORE INSERT ON enrollment**

-- **Use TRIGGER check\_prerequisite\_trigger BEFORE INSERT ON enrollment**

-- **Use TRIGGER check\_lab\_trigger BEFORE INSERT ON enrollment**

* 1. Check for time conflicts.

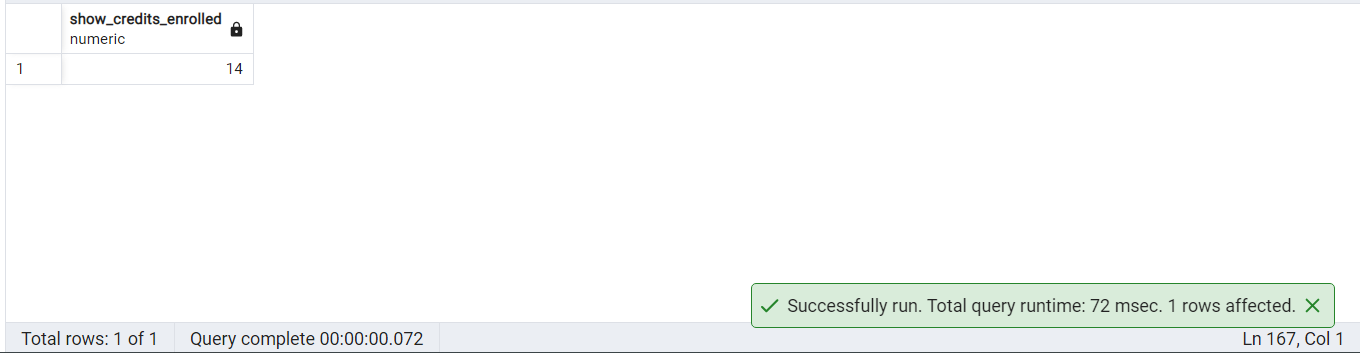
-- **Use TRIGGER check\_cap\_trigger BEFORE INSERT ON enrollment**

-- **Use TRIGGER check\_prerequisite\_trigger BEFORE INSERT ON enrollment**

-- **Use TRIGGER check\_lab\_trigger BEFORE INSERT ON enrollment**

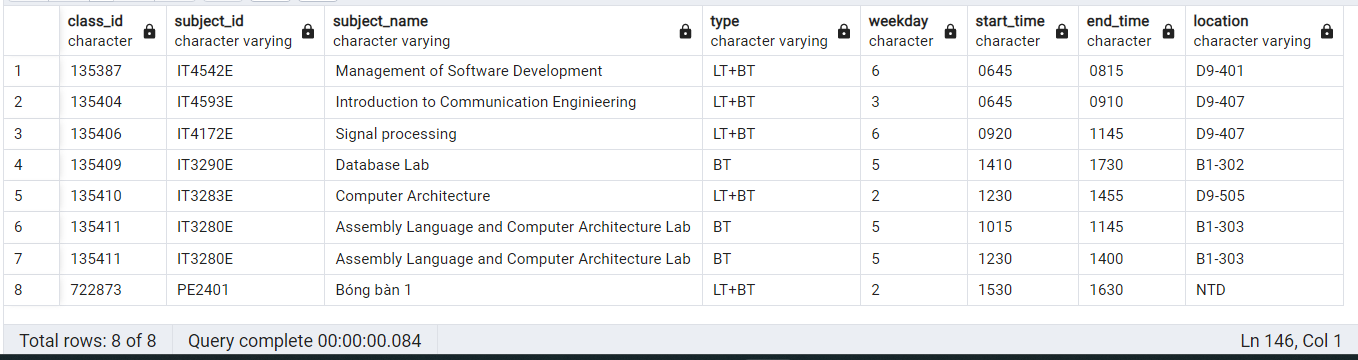
* 1. Calculate total studying credits.

| SELECT \* FROM student.show\_credits\_enrolled(); -- As 20200164 |
| --- |



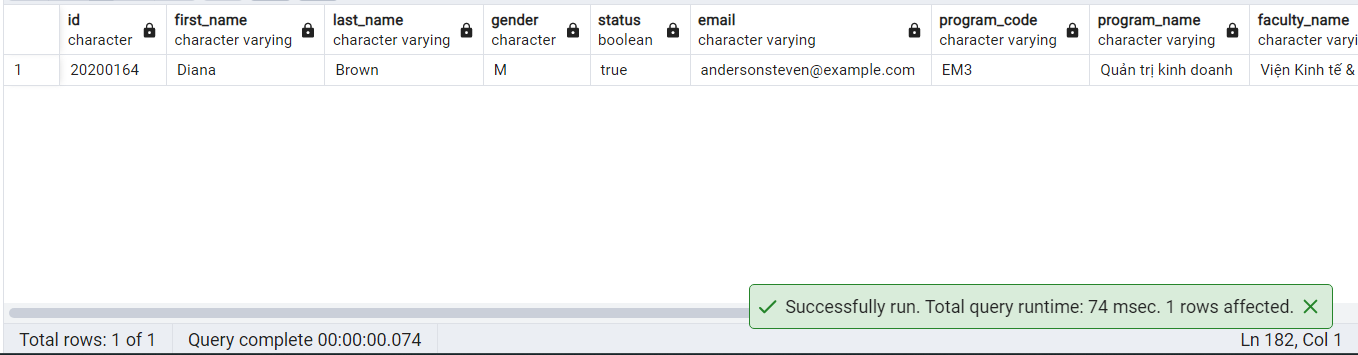
* 1. Schedules are automatically identified after students enroll in classes.

| SELECT \* FROM student.self\_view\_timetable('20221'); -- As 20200164 |
| --- |

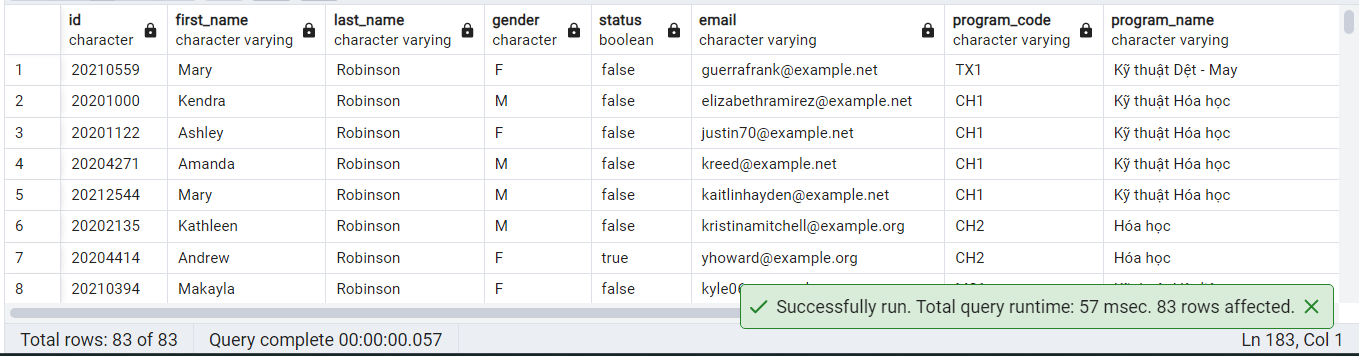


1. Looking up data related to lecturers, subjects, classes, and other students. Retrieve only essential information, excluding personal ones like address, scores, and student timetable.
   1. Students: Name, ID, School/Faculty, Program, Contact, etc.

| SELECT \* FROM search.search\_student\_by\_id('20200164'); |
| --- |

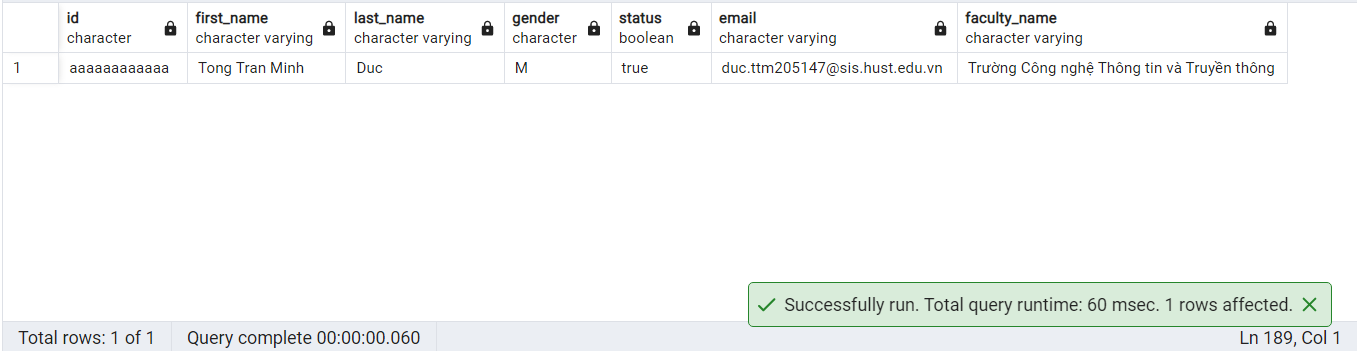


| SELECT \* FROM search.search\_student\_by\_name('Robinson'); |
| --- |

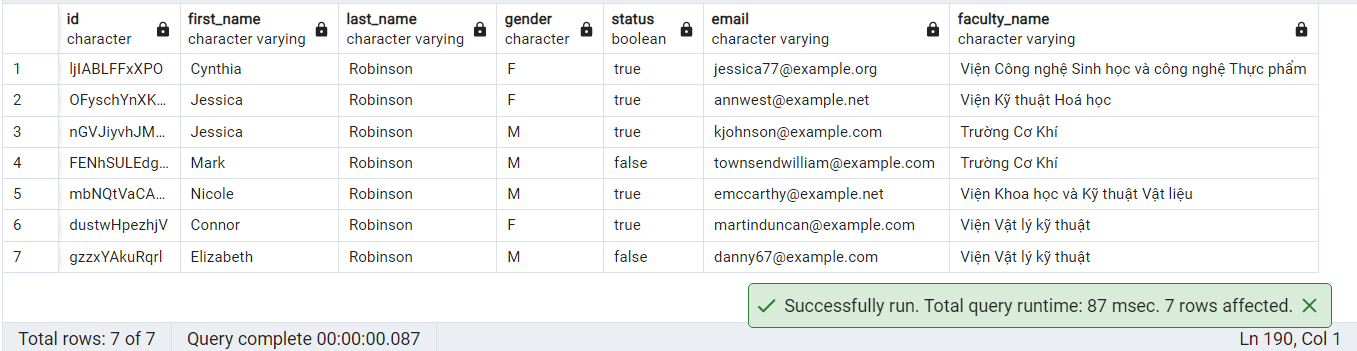


* 1. Lecturers: Name, ID, School/Faculty, Teaching Subjects, Contact, etc.

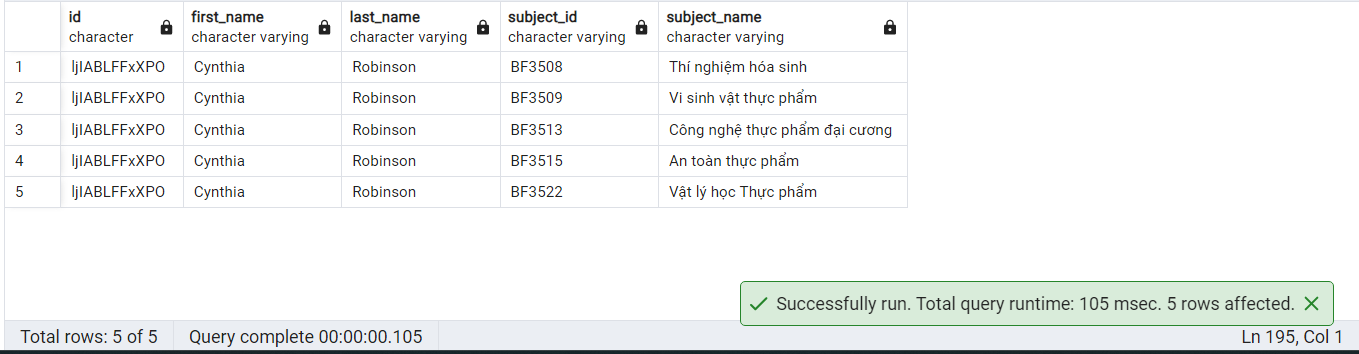
| SELECT \* FROM search.search\_lecturer\_by\_id('aaaaaaaaaaaa'); |
| --- |



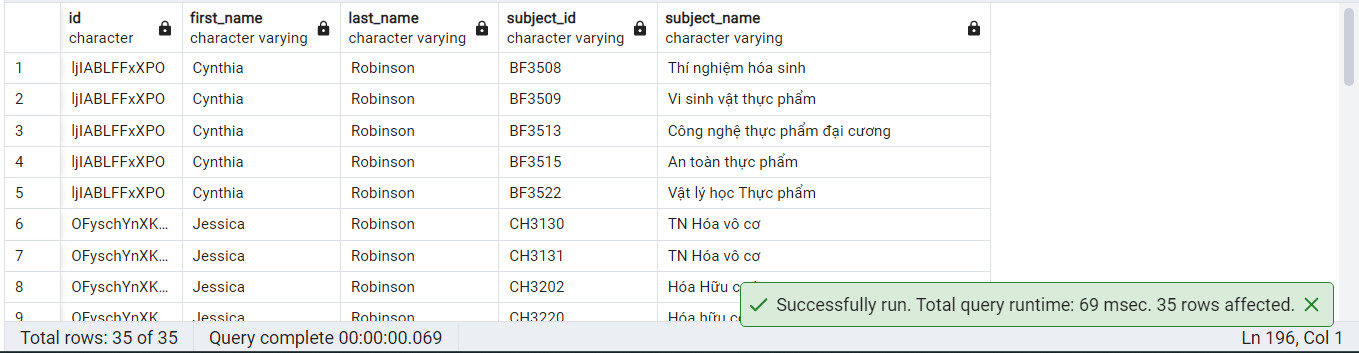
| SELECT \* FROM search.search\_lecturer\_by\_name('Robinson'); |
| --- |



| SELECT \* FROM search.search\_lecturer\_specialization\_by\_id('ljIABLFFxXPO'); |
| --- |



| SELECT \* FROM search.search\_lecturer\_specialization\_by\_name('Robinson'); |
| --- |

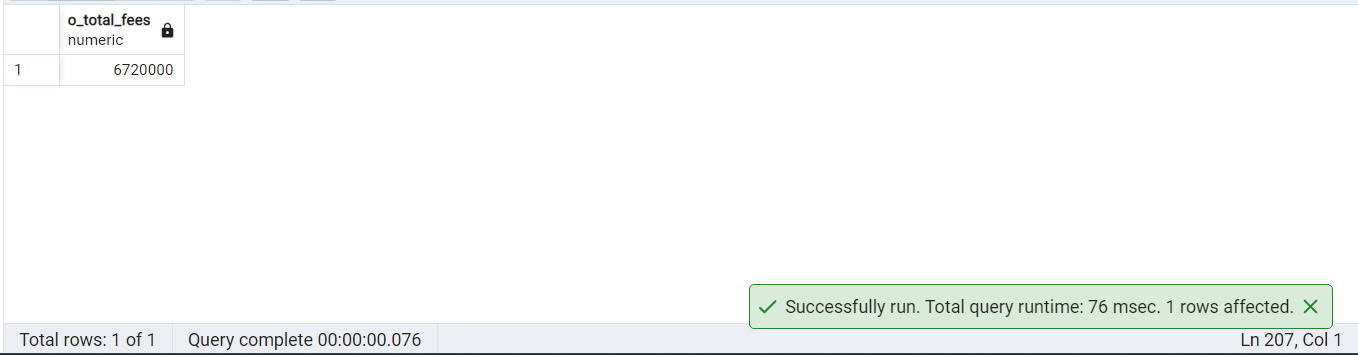


* 1. Subjects, Classes: Number of credits, Schedule of classes, etc.

-- **Done in 2.2.1.**

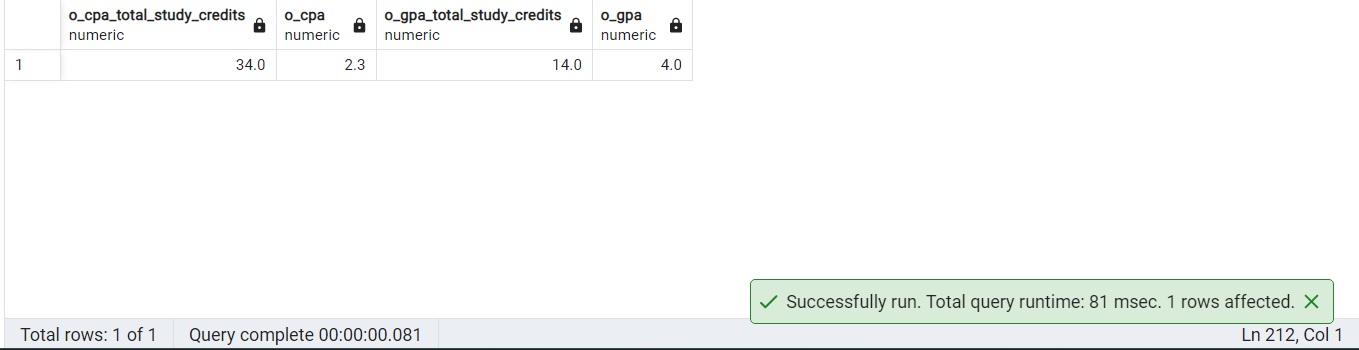
1. Getting estimated fees for the current semester.

| SELECT \* FROM student.show\_estimated\_fees(); -- As 20200164 |
| --- |



1. Getting reports on Study Credits earned, GPA/CPA.

| SELECT \* FROM student.report\_student(); -- As 20200164 |
| --- |



## Lecturers

Login as “**aaaaaaaaaaaa”**. Pass is “**demo”**.

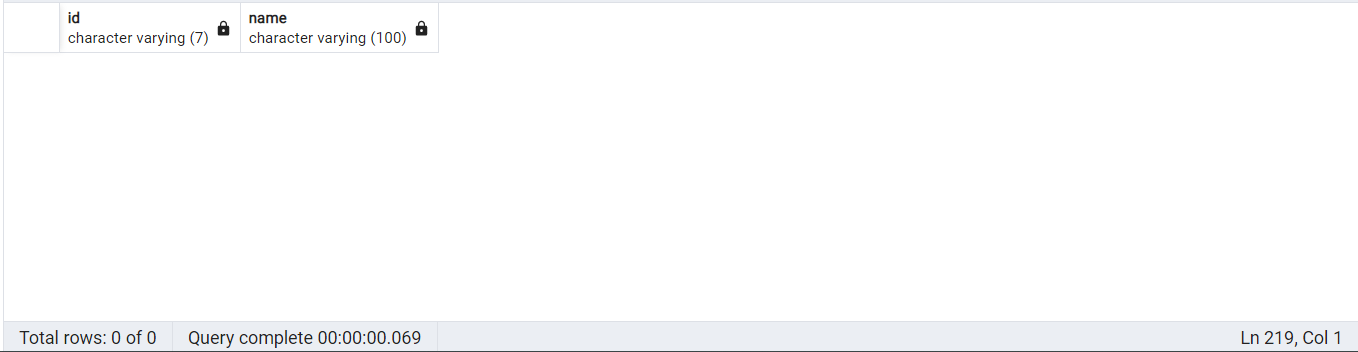
Check GitHub for more details about functions and triggers used for lecturers:

[<https://github.com/duclm278/database-project/blob/main/sql/sms-lecturer.sql>]

1. Viewing data of all teaching subjects, classes, and results of themselves.

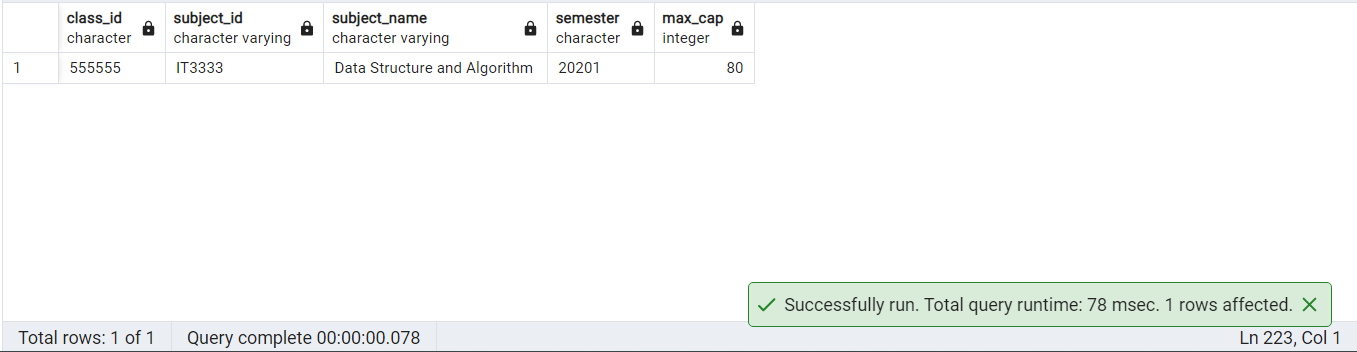
| -- View all subjects of their specializations SELECT \* FROM lecturer.self\_view\_specializations; |
| --- |

**-- Lecturer “aaaaaaaaaaaa” has been just created above and hasn’t had any specializations yet.**

****

**-- Lecturer “aaaaaaaaaaaa” just has been assigned class “555555” in 2.1.**

| -- View all classes of their teachings of any semesters SELECT \* FROM lecturer.self\_view\_class\_assigned('20201'); |
| --- |



1. Looking up data related to students, subjects, classes, and other lecturers. Retrieve only essential information, excluding personal ones like address, scores, and student timetable.
   1. Students: Name, ID, School/Faculty, Program, Contact, etc.

-- **Done in 2.2.4a.**

* 1. Lecturers: Name, ID, School/Faculty, Teaching Subjects, Contact, etc.

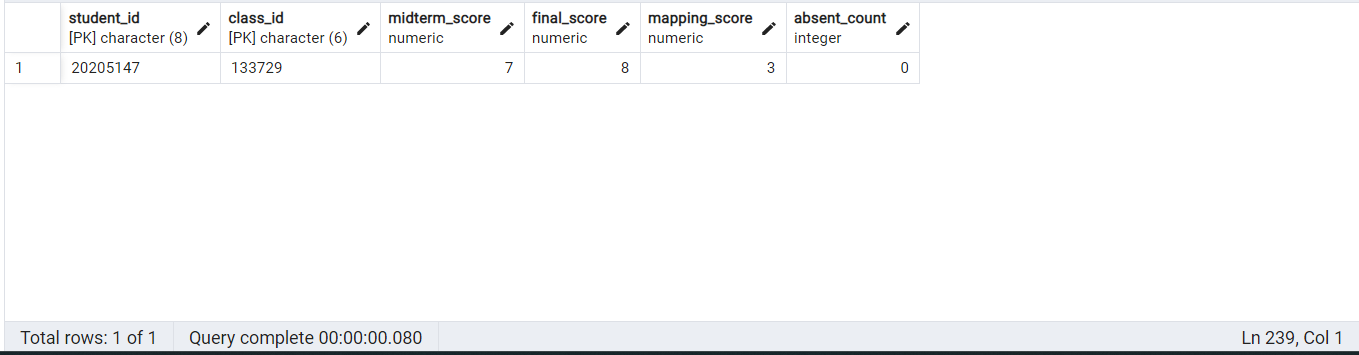
-- **Done in 2.2.4b.**

* 1. Subjects, Classes: Number of credits, Schedule of classes, etc.

-- **Done in 2.2.1.**

1. Recording student academic performance.

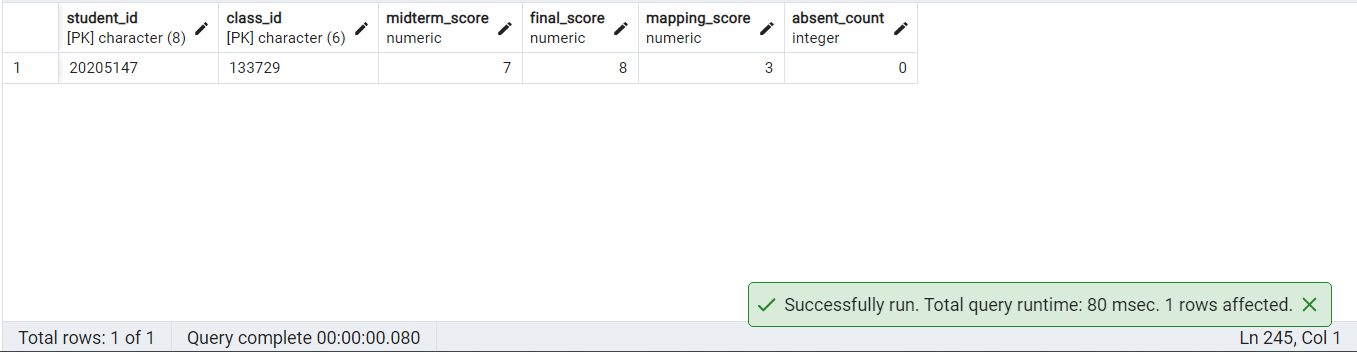
| CALL student.enroll\_class('20205147', '133729'); -- enroll\_class() in 2.2.3f. CALL lecturer.update\_grade('20205147', '133729', 7, 8); SELECT \* FROM enrollment WHERE student\_id = '20205147' AND class\_id = '133729'; |
| --- |



1. Tracking student attendance.

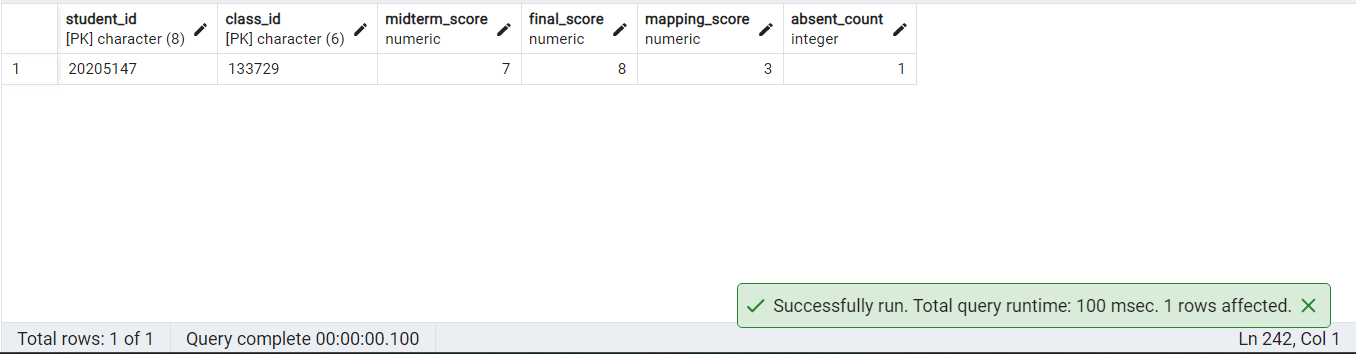
**-- Before using PROCEDURE mark\_absence()**

| SELECT \* FROM enrollment WHERE student\_id = '20205147' AND class\_id = '133729'; |
| --- |



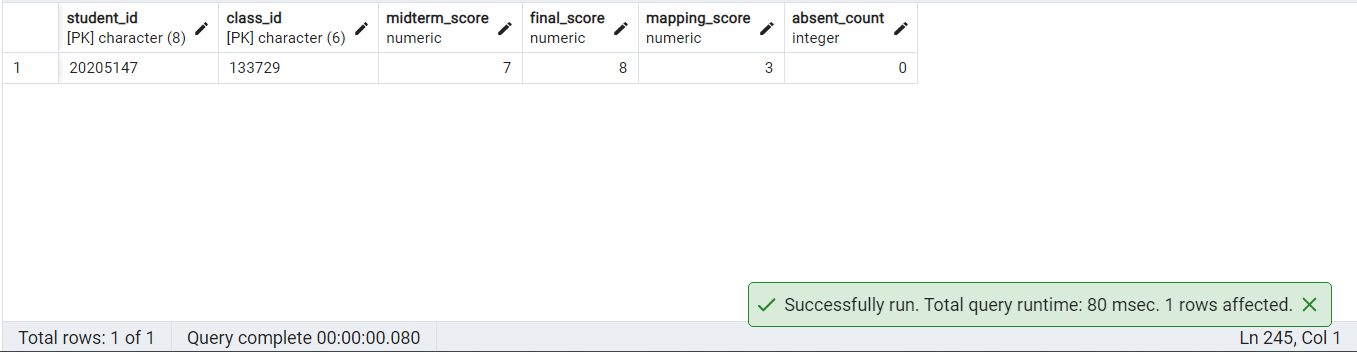
**-- Use PROCEDURE mark\_absence() to mark absence of student**

| CALL lecturer.mark\_absence('20205147', '133729'); SELECT \* FROM enrollment WHERE student\_id = '20205147' AND class\_id = '133729'; |
| --- |



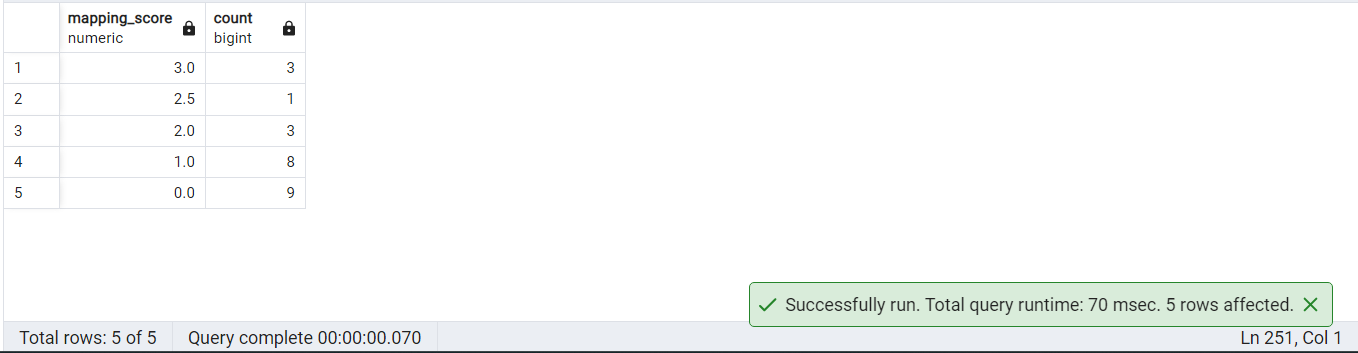
**-- Use PROCEDURE undo\_absence() to undo absence of student**

| CALL lecturer.undo\_absence('20205147', '133729'); SELECT \* FROM enrollment WHERE student\_id = '20205147' AND class\_id = '133729'; |
| --- |



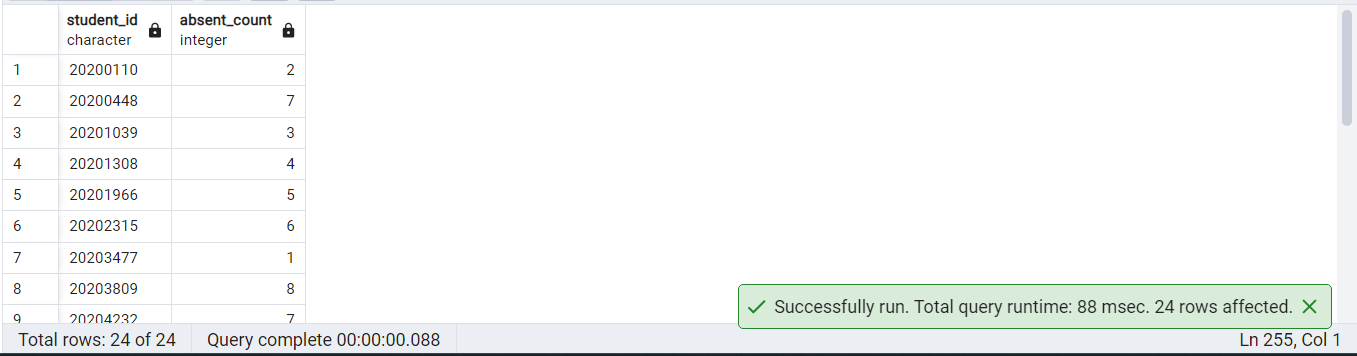
1. Getting reports on
   1. Exam grade distribution (statistics)

| SELECT \* FROM lecturer.report\_grade\_distribution('133729'); |
| --- |



* 1. Student attendance up to today

| SELECT \* FROM lecturer.report\_attendance('133729'); |
| --- |



# 9. Optimizations

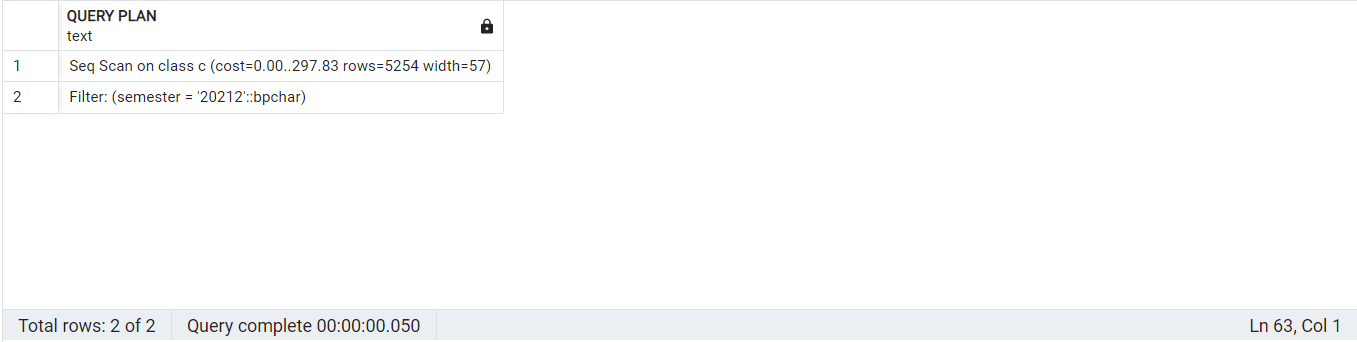
Check GitHub for more details about experiments:

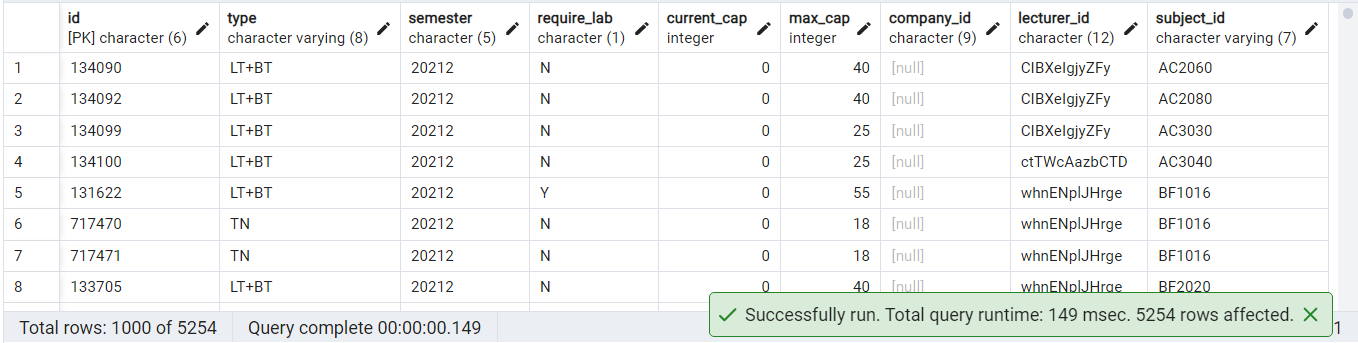
[<https://github.com/duclm278/database-project/blob/main/sql/sms-index.sql>]

1. Create necessary indexes for searching

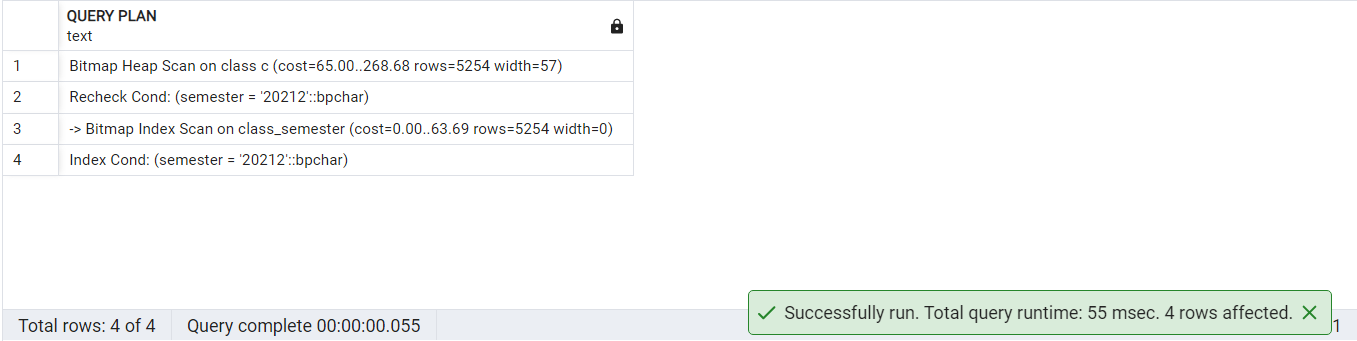
| EXPLAIN SELECT c.\* FROM class c  WHERE c.semester = '20212'; |
| --- |

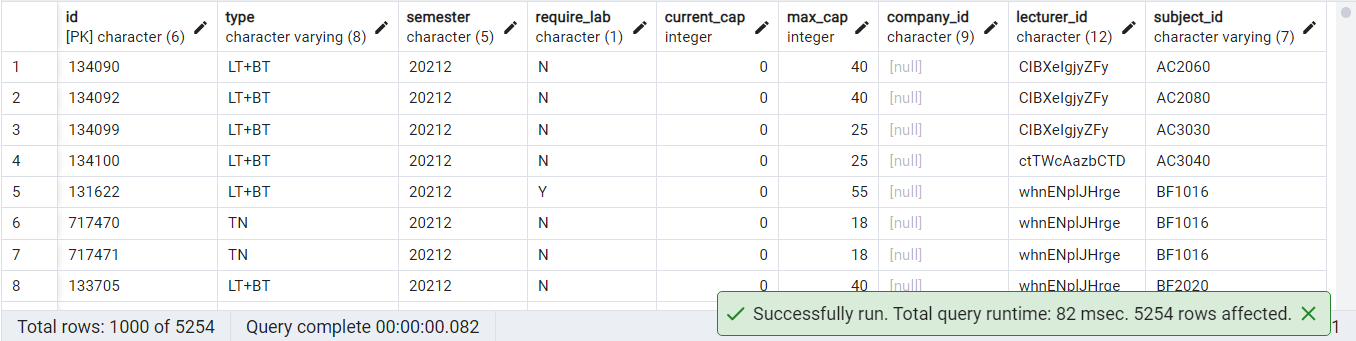
BEFORE: Seq scan on class, **149ms**





AFTER: Used class\_semster, bitmap index scan, **82ms**

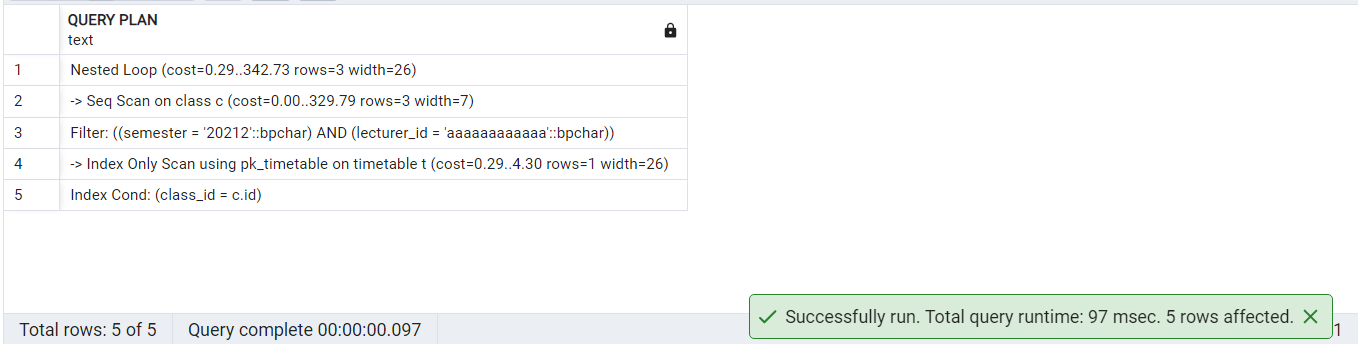


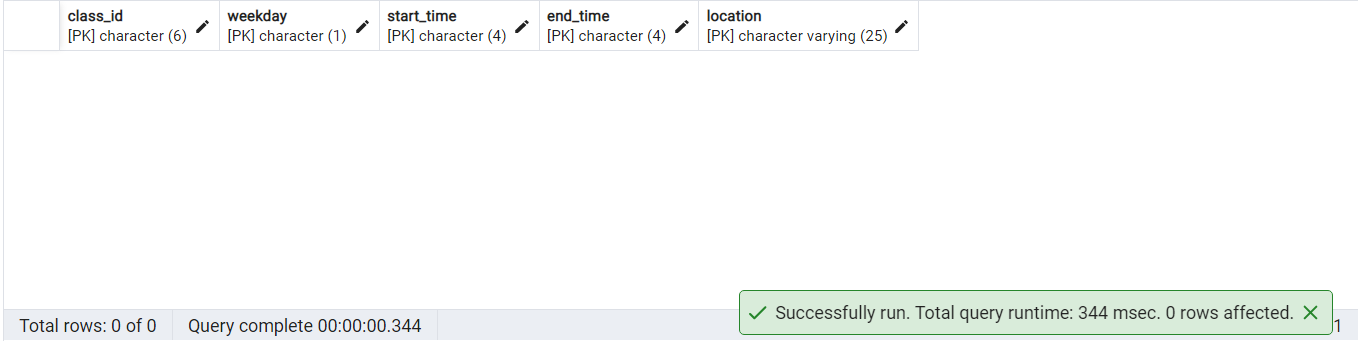
****

1. Create necessary indexes on non-prime attributes while joining

| EXPLAIN SELECT t.\* FROM timetable t JOIN class c ON t.class\_id = c.id WHERE c.lecturer\_id = 'aaaaaaaaaaaa' AND c.semester = '20212';  DROP INDEX IF EXISTS class\_lecturer\_id; CREATE INDEX class\_lecturer\_id ON class (lecturer\_id); |
| --- |

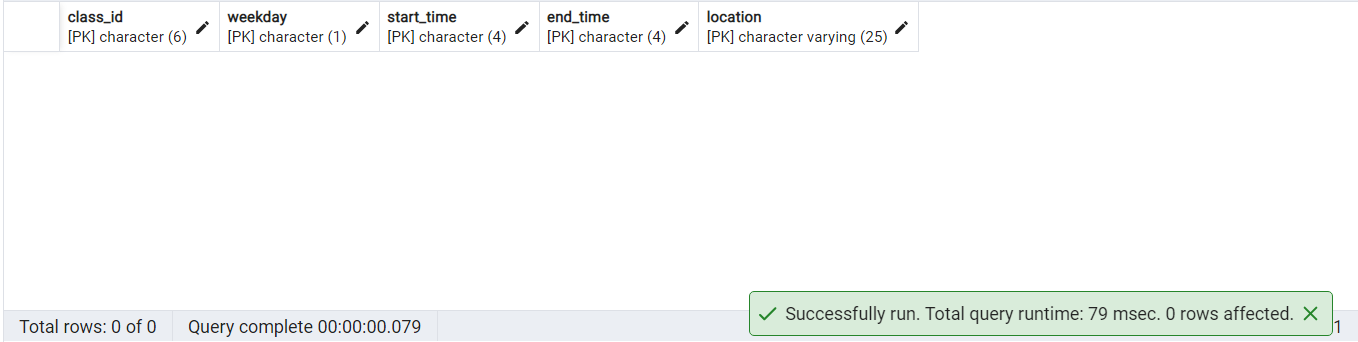
BEFORE: Seq scan on class, **334ms**





AFTER: Used class\_lecturer\_id, bitmap index scan, **79ms**





1. Do not create indexes for large varchar, but create indexes for char

**Searching by id is fast as it uses primary keys of type char.**

**Searching by name is slow as varchar is too big to be indexed and “%” is used to match partially.**

| SELECT \* FROM search.view\_search\_student v WHERE v.first\_name || ' ' || v.last\_name LIKE '%' || i\_student\_name || '%'; |
| --- |

**Sources:**

[<https://stackoverflow.com/questions/8001905/sql-server-worth-indexing-large-string-keys>]

[<https://stackoverflow.com/questions/1388059/sql-server-index-columns-used-in-like>]

1. Do not create indexes for fully joining

| EXPLAIN SELECT c.semester, t.\* FROM timetable t JOIN class c ON c.id = t.class\_id; |
| --- |

BEFORE: Seq scan, hash join



AFTER: Not used due to full table scan.



1. Do not create indexes for table having few rows

**-- E.g. Querying from or joining with self\_view\_info doesn't need indexes as it only stores one record per user!**

# 10. Conclusion

**Conclusion about this project:**

* Searching and joining are fast as the system works mainly with primary keys.
* Triggers and custom views are very powerful as the system needs complex checking and restrictions.

**Insights gained from this project:**

* See how triggers and views are applied in practice.
* Make use of variables and triggers for fast querying and maintaining data.
* The ERD process helps a lot with the design of the database.
* Experience when working with many entities and relations involed.