Ho Chi Minh City University of Technology
FACULTY OF COMPUTER SCIENCE & ENGINEERING

...☼...

DATABASE SYSTEM LAB

ASSIGNMENT

Assignment Task 2

Database Design: School Management System

Lecturer: Nguyễn Thị Ái Thảo

Group: 3

Student	Student ID
Nguyễn Mỹ Khanh	2052525
Vũ Đồng Tuệ Quyên	2052234
Trần Gia Linh	2053182
Văn Thị Hà Trân	2053519

Ho Chi Minh City – 2022

Contents

Task1: Designing the database at the physical level.

I. Business description:

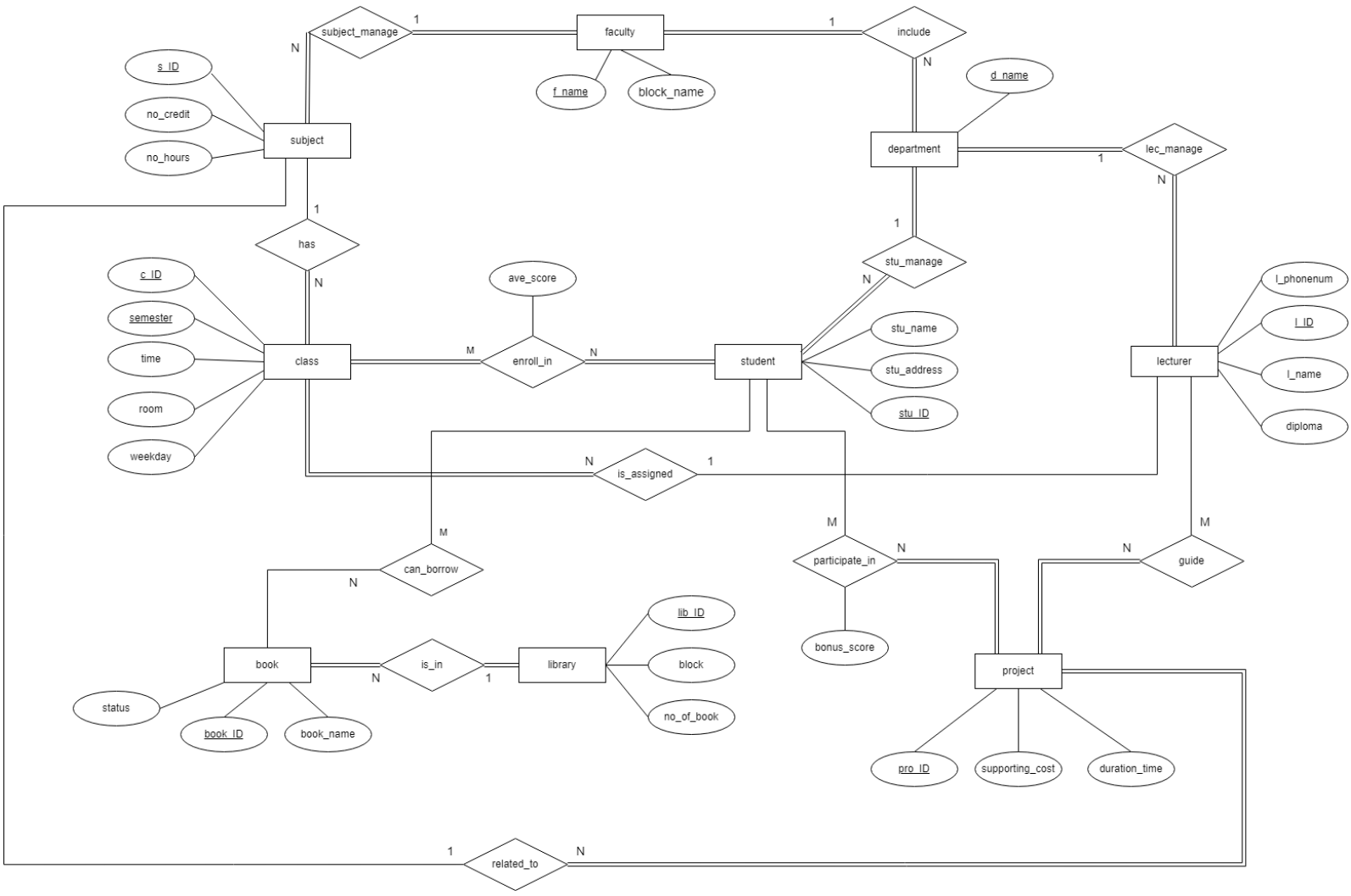
A school named X has some faculties, identified by faculty's name (f_name). Each faculty only works in 1 block, identified by the block's name (block_name). Each faculty manages some departments identified by department's name (d_name) (for example, CS&CE faculty manages 2 departments: CS department and CE department). Each faculty also takes responsibility for its subjects, identified by subject's ID (s_ID), each subject is assigned a fixed number of credits (no_credit) and teaching time (no_hours). Each faculty has different subjects, and because a large number of students enroll in 1 subject, a subject may be divided into one or more classes, defined by a composition key include class's ID (c_ID) and semester name (semester), other information as learning schedule (time, room, weekday) is displayed to student. In addition, the ID of a class should be written in a fixed form as subjectID_class (for example CO2017_1). All of the subjects of a department will be teached in that department's block.

Meanwhile, each department manages its own lecturers and students by lecturer_ID (l_ID) and student_ID (stu_ID). For those managing work, the department wants to collect contact information of students and lecturers such as student's address, student's name, lecturer's name, lecturer diploma and lecturer's phone number. A student can enroll in classes of different subjects, while a lecturer must be assigned to a class of her/his own faculties but can teach different subjects. For each enrollment of 1 student in 1 class, students will gain an average score score for that subject (ave_score).

There are projects about the content of each separated subject guided by one or more lecturers and joined by multiple students. Each project is identified by its own ID (pro_ID), its duration time is recorded in days and operation fee in VND is supported (supporting_cost) by the department. For every participation of a student in each project, he/she can get some bonus score (bonus_score) for the average score of the related subject.

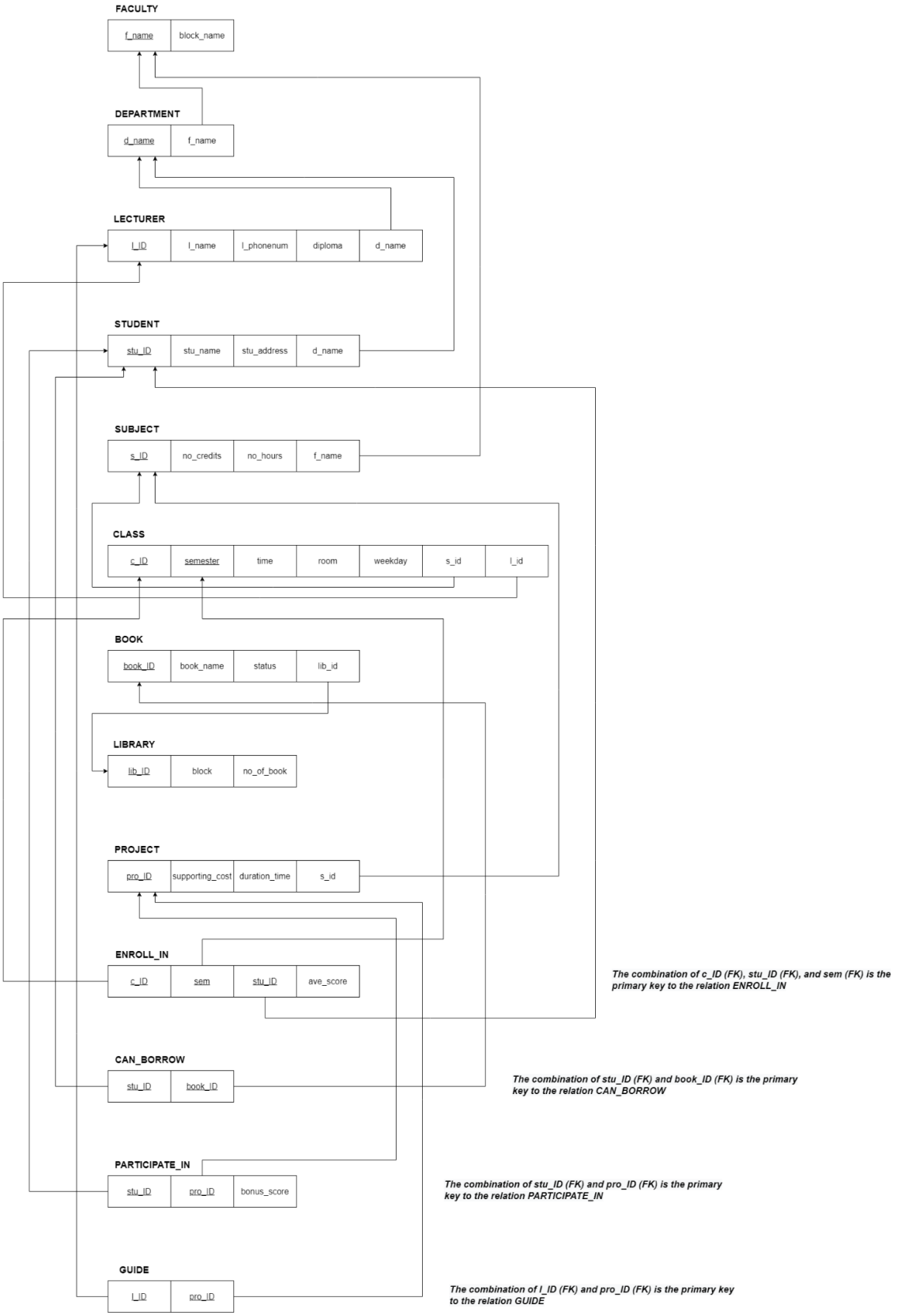
Students in school can borrow a limited quantity of books from several libraries which are defined by the library's ID. There are some libraries in school and students can take a look for the location by their block's name. Books are managed by an identical ID, name and status. Status of books can only be one of these fixed value (available for loan, on loan)

II. ER diagram:



*Note: Please follow this link to have a better quality PNG file or get the designing file of both ER Diagram and relational database schema.

III. Relational database schema:



Task 2: Implementing the above physical database.

I. Languages, Tools and Enviroment:

For first 5 task, we implementing the database by *SQLite3* with *DB for SQLite3* as supporting application for easier observation.
We build web application by *Flask-Python, HTML, CSS* and *SQLAlchemy* base on the above existing database.
However, in the last task - create user and assign privilege, we have some trouble in coding, so we decided to build only this task by *Oracle* with *SQLDeveloper* as supporting application.

II. Database Implementation:

At first, in order to connect to SQLite3, a connection and cursor must be created to connect to database, and before end program, this connection must be closed.

```
conn = sqlite3.connect('data.db', timeout=100)
c = conn.cursor()
conn.execute("PRAGMA foreign_keys = ON")
```

```
conn.close()
```

Next step, each table is created by following syntax:

```
#table with no foreign key
create_faculties_table = """
    create table if not exists faculties (
        f_name char(40) not null primary key,
        block_name char(40)
    );
"""
```

```
#table with foreign key
#set on delete cascade and on update cascade to synchronize data
create_classes_table = """
    create table if not exists classes (
        c_ID char(40) not null,
        semester integer not null,
        time char(40),
```

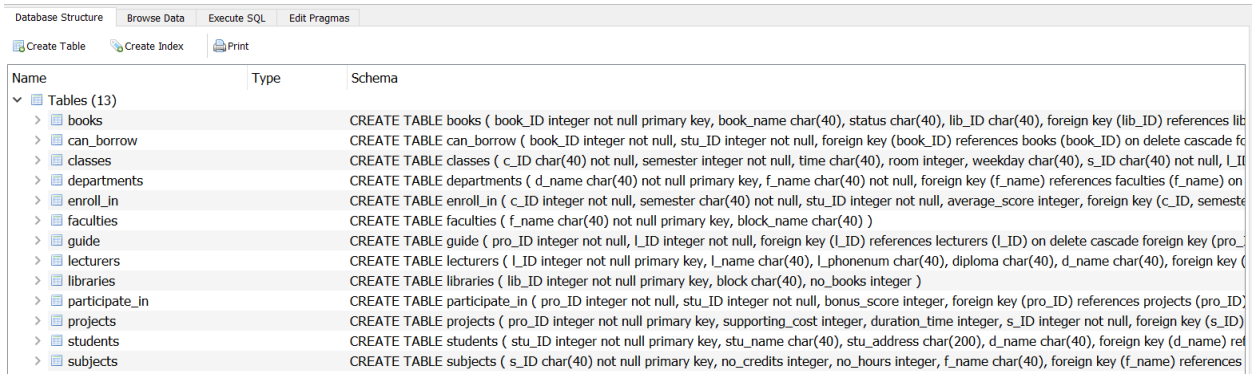
```
        room integer,
        weekday char(40),
        s_ID char(40) not null,
        l_ID integer not null,
        primary key (c_ID, semester)
        foreign key (s_ID) references subjects (s_ID) on delete cascade on
update cascade,
        foreign key (l_ID) references lecturers (l_ID) on delete cascade
on update cascade
    );
"""
```

Finally, execute these statement and commit all the changes into database:

```
c.execute(create_faculties_table)
conn.commit()
c.execute(create_classes_table)
conn.commit()
…
```

III. Creating results:

All the statement is executed and all the changes are committed successfully.



Task 3: Insert data for all tables in the database.

I. Insert data:

Data of each table is inserted into by following syntax:

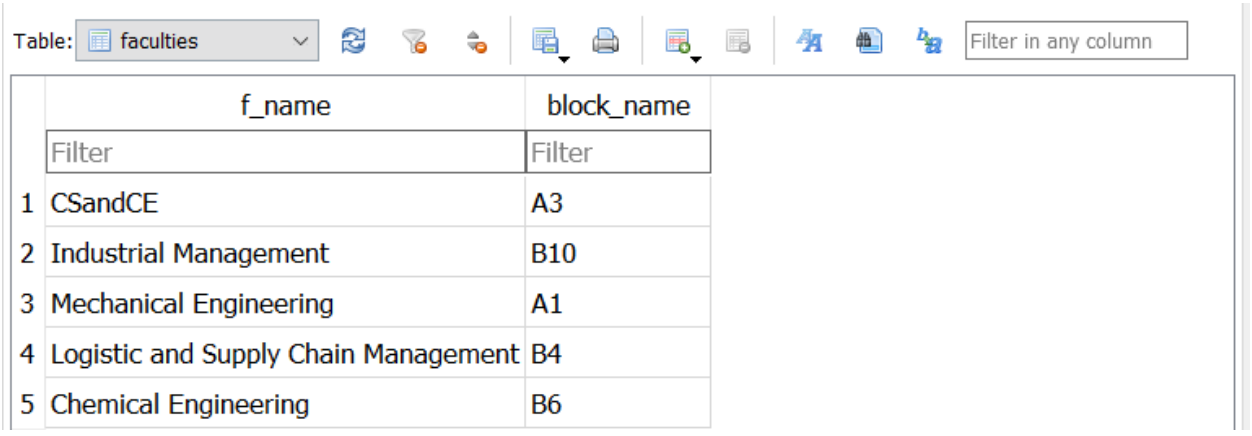
```
#multivalue insert
insert_faculties_value = """
    insert into faculties
    select 'CSandCE' as f_name, 'A3' as block_name
    union all select 'Industrial Management', 'B10'
    union all select 'Mechanical Engineering', 'A1'
    union all select 'Logistic and Supply Chain Management', 'B4'
    union all select 'Chemical Engineering', 'B6'
"""
```

Execute these statement and commit all the changes into database:

```
c.execute(insert_faculties_value)
conn.commit()
```

II. Inserting results:

Each table after executing insert statement:



	f_name	block_name
	Filter	Filter
1	CSandCE	A3
2	Industrial Management	B10
3	Mechanical Engineering	A1
4	Logistic and Supply Chain Management	B4
5	Chemical Engineering	B6

Table: departments

Filter in any column

	d_name	f_name
	Filter	Filter
1	Computer Science	CSandCE
2	Computer Engineering	CSandCE
3	Marketing	Industrial Management
4	Manufacturing	Industrial Management
5	Organic Chemistry	Chemical Engineering
6	Manufacturing Engineering	Mechanical Engineering

Table: lecturers

Filter in any column

	l_ID	l_name	l_phonenum	diploma	d_name
	Filter	Filter	Filter	Filter	Filter
1	3518	Thinh Duc Bao	(206) 342-8631	Professor	Computer Science
2	3520	Vu Tung Linh	(717) 550-1675	Professor	Marketing
3	3522	Cao Vu Minh	(248) 762-0356	Doctor	Manufacturing
4	3524	Mai Duc Toan	(253) 644-2182	Doctor	Computer Science
5	3525	Chester Woods	(212) 658-3916	Professor	Organic Chemistry
6	3526	Alina Wilkerson	(202) 918-2132	Doctor	Manufacturing Engineering

Table: students

Filter in any column

	stu_ID	stu_name	stu_address	d_name
	Filter	Filter	Filter	Filter
1	1822241	Huynh Phat	To Hien Thanh, Phu Nhuan District	Manufacturing
2	1912239	Nhat Truong	Bac Hai, District 10	Marketing
3	2052233	Vu Quyen	Hiep Binh Chanh, Thu Duc	Computer Science
4	2052235	Ha Tran	Nguyen Van Thuong, Binh Thanh	Marketing
5	2052237	Hoang Nhat	Nguyen Xi, Binh Thanh	Computer Engineering

Table: subjects

Filter in any column

	s_ID	no_credits	no_hours	f_name
	Filter	Filter	Filter	Filter
1	CO3017	4	45	Industrial Management
2	CO1003	3	55	CSandCE
3	CO2015	4	50	CSandCE
4	CO2017	2	55	Mechanical Engineering
5	CO3002	2	40	Chemical Engineering

Table: classes

Filter in any column

	c_ID	semester	time	room	weekday	s_ID	l_ID
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	CO3017_1	221	7 a.m	306	Mon	CO3017	3522
2	CO3017_1	211	4 p.m	512	Mon	CO3017	3520
3	CO1003_3	213	1 p.m	112	Tue	CO1003	3518
4	CO2015_7	221	1 p.m	512	Fri	CO2015	3518
5	CO2017_11	223	3 p.m	207	Fri	CO2017	3526
6	CO2015_12	221	8 a.m	405	Sun	CO2015	3524

Table: enroll_in

Filter in any column

	c_ID	semester	stu_ID	average_score
	Filter	Filter	Filter	Filter
1	CO3017_1	221	2052233	8
2	CO3017_1	211	2052235	9
3	CO1003_3	213	2052237	7
4	CO2015_7	221	1912239	6
5	CO2017_11	223	1822241	7
6	CO2015_12	221	2052233	6

Table: projects

	pro_ID	supporting_cost	duration_time	s_ID
	Filter	Filter	Filter	Filter
1	1	5000000	80	CO3017
2	2	7000000	50	CO3017
3	3	10000000	60	CO1003
4	4	5200000	55	CO2017
5	5	500000	20	CO3002

Table: guide

	pro_ID	l_ID
	Filter	Filter
1	1	3518
2	2	3520
3	3	3520
4	3	3518
5	4	3524

Table: participate_in

	pro_ID	stu_ID	bonus_score
	Filter	Filter	Filter
1	1	2052233	1
2	2	2052235	1
3	3	2052237	2
4	2	1912239	2
5	4	1912239	3

Table: libraries

Filter in any column

	lib_ID	block	no_books
	Filter	Filter	Filter
1	1	A2	2
2	2	A4	3
3	3	H6	1

Table: books

Filter in any column

	book_ID	book_name	status	lib_ID
	Filter	Filter	Filter	Filter
1	111	Fundamental of Database System	On loan	1
2	112	Fundamental of Software Architecture	On loan	1
3	113	Clean Architecture	On loan	2
4	114	Rescuing Human Rights	Available for loan	2
5	115	The Science of Learning and ...	On loan	2
6	116	Earthopolis	On loan	3

Table: can_borrow

Filter in any column

	book_ID	stu_ID
	Filter	Filter
1	111	2052235
2	112	1912239
3	113	2052237
4	115	1822241
5	116	2052235

Task 4: Performing database operations, such as SELECT, UPDATE, DELETE…

I. Query 1:

We start with the most simplest query: *Retrieve the block name of department 'Manufaturing'*

Statement: query is execute and methos fetchall is used to show the results to python terminal.

```
query1 = """
    select block_name
    from faculties
    inner join departments on faculties.f_name = departments.f_name
    where d_name = 'Manufacturing'
"""
c.execute(query1)
print(c.fetchall())
```

Result:

	block_name
1	B10

II. Query 2:

Query 2: *List the name of all lecturers who guide 2 or more projects*

Theory solve:

T1 (l_ID, no_project) $\leftarrow {}_{l\_ID}\Im_{COUNT(pro\_ID)}$ (GUIDE)

T2 $\leftarrow \sigma_{no\_project > 1}$ (T1)

RESULT $\leftarrow \pi_{l\_name}$ (T2 * LECTURER)

Statement: temporary table is created only for query purpose.

```
queryt2_1 = """
    create temporary table t1 as
    select l_ID,count(pro_ID)
    from guide
    group by l_ID
"""
```

```
query2_2 = """
    create temporary table t2 as
    select * from t1
    where [count(pro_ID)] > 1
"""

query2_3 = """
    select l_name
    from t2
    natural join lecturers
"""
c.execute(queryt2_1)
conn.commit()
c.execute(query2_2)
conn.commit()
c.execute(query2_3)
print(c.fetchall())
```

Result:

	l_name
1	Thinh Duc Bao
2	Vu Tung Linh

Table t1:

```
1   SELECT * FROM t1;
```

	l_ID	count(pro_ID)
1	3518	2
2	3520	2
3	3524	1

Table t2:

```
1    SELECT * FROM t2;
```

	l_ID	count(pro_ID)
1	3518	2
2	3520	2

III. Query 3:

Query 3: *Retrieve the names of the students with no bonus score (these student do not participate in any project)*

Theory solve:

ALL_STU (ID) ← π_{stu_ID} (STUDENT)

STU_WITH_PROJ (ID) ← π_{stu_id} (PARTICIPATE_IN)

STU_WITHOUT_PROJ ← ALL_STU - STU_WITH_PROJ

RESULT ← $\pi_{stu\_ID \text{ AND } stu\_name}$ (STU_WITHOUT_PROJ * STUDENT)

Statement:

```
query3_1 = """create temporary table all_stu as
select distinct stu_ID from students;"""

query3_2 = """create temporary table stu_with_proj as
select distinct stu_ID from participate_in;"""

query3_3 = """create temporary table stu_without_proj as
select stu_ID from all_stu
except select stu_ID from stu_with_proj;"""

query3_4 = """select stu_ID, stu_name from
stu_without_proj natural join students;"""

c.execute(query3_1)
conn.commit()
c.execute(query3_2)
conn.commit()
```

```
c.execute(query3_3)
conn.commit()
c.execute(query3_4)
print(c.fetchall())
```

Result:

	stu_ID	stu_name
1	1822241	Huynh Phat

Table all_stu:

```
1  select * from all_stu
2
```

	stu_ID
1	1822241
2	1912239
3	2052233
4	2052235
5	2052237

Table stu_with_proj:

```
1  select * from stu_with_proj
2
```

	stu_ID
1	2052233
2	2052235
3	2052237
4	1912239

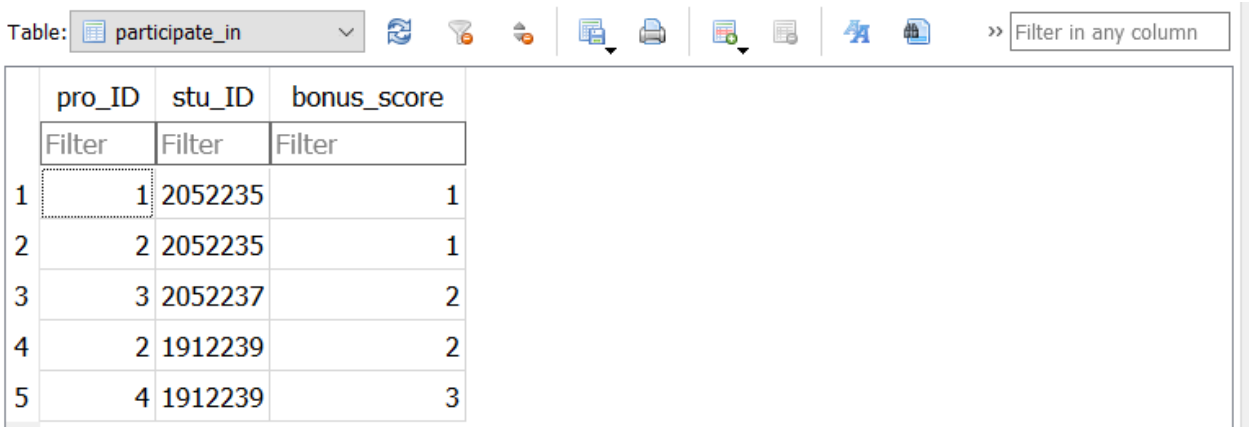
IV. Update:

In this part, we change hange participated student of project 1 to another student who already participated in other project and observe the changes.

Statement:

```
update = """update participate_in
set stu_ID = '2052235'
where pro_ID = '1'"""
c.execute(update)
conn.commit()
```

Update result:



	pro_ID	stu_ID	bonus_score
	Filter	Filter	Filter
1	1	2052235	1
2	2	2052235	1
3	3	2052237	2
4	2	1912239	2
5	4	1912239	3

Then, we do again query 3. The result was changed at this time. There are 2 student do not participate in any project because the only participation of “VU QUYEN” in project 1 was change to another student.

Query result:

	stu_ID	stu_name
1	1822241	Huynh Phat
2	2052233	Vu Quyen

V. Delete:

We show delete on cascade: delete all projects of subjects CO3017, when any project is deleted, the coresponding data from other tables is also being deleted

Statement:

```
delete = """delete from projects
where s_ID = 'CO3017'"""
```

```
c.execute(delete)
conn.commit()
```

Result of table projects: the projects which are related to that subject is deleted (which are project 1 and 2)

```
1  select * from projects;
2
3
```

	pro_ID	supporting_cost	duration_time	s_ID
1	3	10000000	60	CO1003
2	4	5200000	55	CO2017
3	5	500000	20	CO3002

Result of other tables use pro_ID as foreign key:
Table participate_in:

	pro_ID	stu_ID	bonus_score
1	3	2052237	2
2	4	1912239	3

Table guide:

```
1  select * from guide;
2
3
```

	pro_ID	l_ID
1	3	3520
2	3	3518
3	4	3524

Task 5: Write TRIGGER, FUNCTION, STORE PROCEDURE for the queries and constraints which were described in the assignment 1.

I. Constraint 1:

Lecturers can only be assigned to a class of subject belonging to the faculty that they are in. ERD showed no condition in assigning one subject to one lecturer.

This trigger will retrieve the faculty of newly inserted subject and lecturer. If true faculties are the same, successfully inserting, otherwise system will abort an error message and input command is not executed.
We also create 1 valid statement and 1 invalid statement to compare the differences.

Statement:

```
create_triiger_on_classes = """create trigger assign_class_to_lecturer
    before insert on classes
begin
    select case
        when (select distinct f_name from faculties
                natural join subjects
                where s_ID = new.s_ID) != (select distinct f_name from
faculties
                                            natural join departments
                                            natural join lecturers
                                            where l_ID = new.l_ID)
then
            raise(abort, 'Lecturer can only be assigned to the subject of
the same faculties')
    end;
end;"""

insert_classes_invalid = """insert into classes(c_ID, semester, time,
room, weekday, s_ID, l_ID)
values ('CO1003_5', '222', '7 a.m', '206', 'Sat', 'CO1003', '0003525');"""

insert_classes_valid = """insert into classes(c_ID, semester, time, room,
weekday, s_ID, l_ID)
values ('CO3002_5', '222', '7 a.m', '206', 'Sat', 'CO3002', '0003525');"""

c.execute(create_triiger_on_classes)
```

```
conn.commit()
```

Invalid insert result:

```
Execution finished with errors.
Result: Lecturer can only be assigned to the subject of the same faculties
At line 1:
insert into classes(c_ID, semester, time, room, weekday, s_ID, l_ID)
values ('CO1003_5', '222', '7 a.m', '206', 'Sat', 'CO1003', '0003525');
```

Table classes after valid inserting: 1 class was added

Table: classes

	c_ID	semester	time	room	weekday	s_ID	l_ID
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	CO3017_1	221	7 a.m	306	Mon	CO3017	3522
2	CO3017_1	211	4 p.m	512	Mon	CO3017	3520
3	CO1003_3	213	1 p.m	112	Tue	CO1003	3518
4	CO2015_7	221	1 p.m	512	Fri	CO2015	3518
5	CO2017_11	223	3 p.m	207	Fri	CO2017	3526
6	CO2015_12	221	8 a.m	405	Sun	CO2015	3524
7	CO3002_5	222	7 a.m	206	Sat	CO3002	3525

II. Constraint 2:

Status of books can only be one of these fixed value (available for loan, on loan)

This trigger will check if user insert true data format for status of books, which are either “Available for loan” or “On loan”. If the user input wrong phrase, system will abort an error message and input command is not executed.
We create 1 valid statement and 1 invalid statement to compare the differences.

Statement:

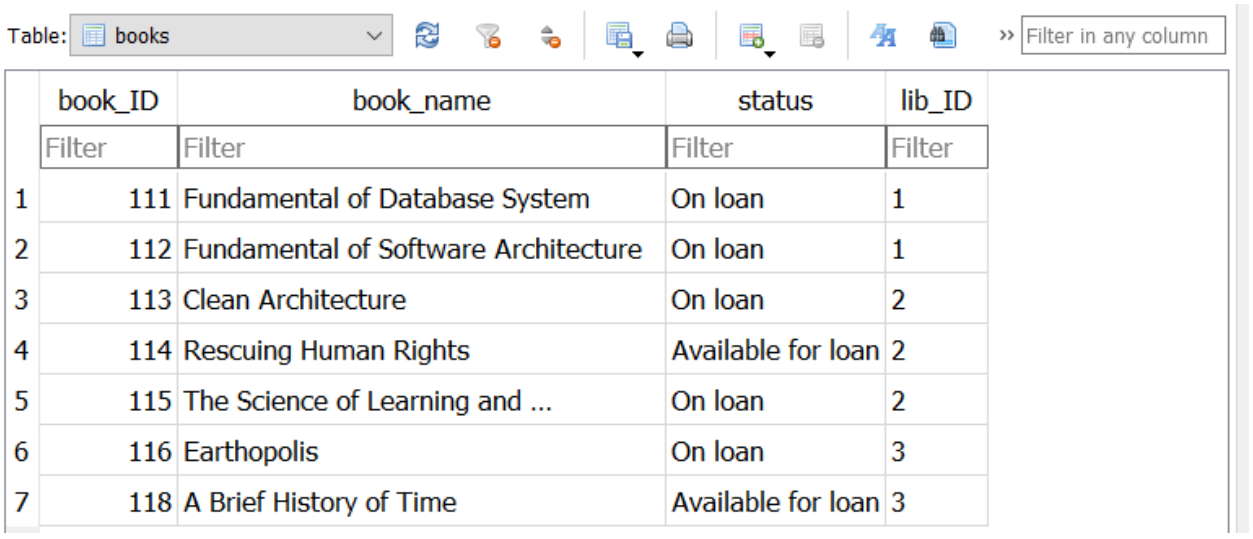
```
create_trigger_on_books = """
create trigger validate_book_status_insert
    before insert on books
begin
    select
        case
            when new.status not in('Available for loan','On loan') then
```

```
                raise(abort, 'Status of book must be "Available for loan"
or "On loan"')
        end;
end;"""

insert_book_invalid = """insert into books (book_ID, book_name, status,
lib_ID)
values ('117', 'Nineteen Eighty-Four', 'is loaned', 3);"""


insert_book_valid = """insert into books (book_ID, book_name, status,
lib_ID)
values ('118', 'A Brief History of Time', 'Available for loan', 3);"""

c.execute(create_trigger_on_books)
conn.commit()
```

Invalid insert result:

```
Execution finished with errors.
Result: Status of book must be "Available for loan" or "On loan"
At line 1:
insert into books (book_ID, book_name, status, lib_ID)
values ('117', 'Nineteen Eighty-Four', 'is loaned', 3);
```

Table books after valid inserting: 1 book was added



	book_ID	book_name	status	lib_ID
	Filter	Filter	Filter	Filter
1	111	Fundamental of Database System	On loan	1
2	112	Fundamental of Software Architecture	On loan	1
3	113	Clean Architecture	On loan	2
4	114	Rescuing Human Rights	Available for loan	2
5	115	The Science of Learning and ...	On loan	2
6	116	Earthopolis	On loan	3
7	118	A Brief History of Time	Available for loan	3

III. Constraint 3:

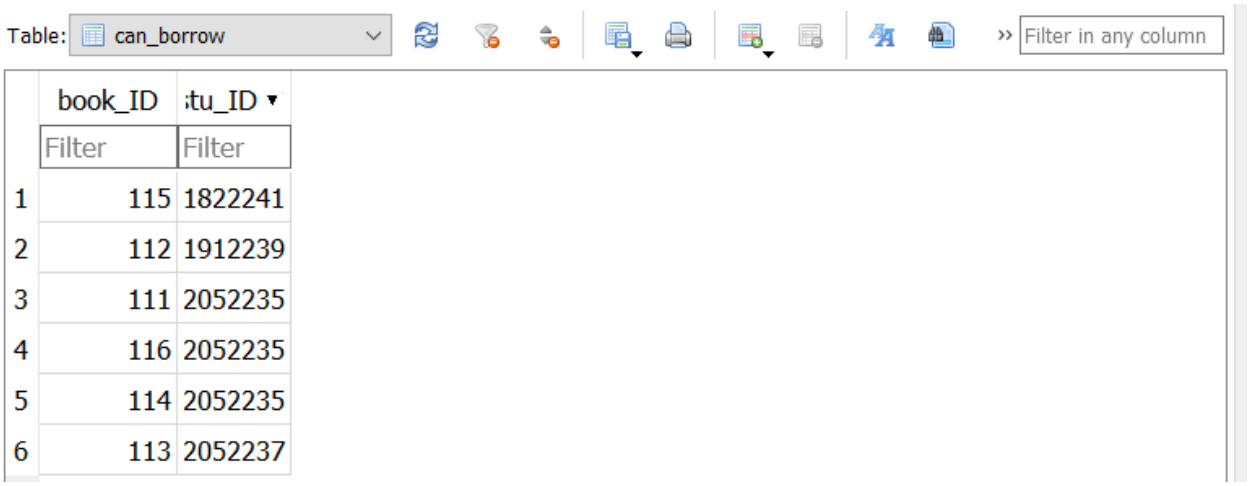
One student can borrow a limited quantity of books.

This trigger will limit the number of books one student can borrow at the same time. We assume that number of books is 3. If current total number of books is equal or greater than 3, the system abort an error message to user and input command is not executed. Otherwise, student can borrow more books.
In the current database, student “2052235” already borrow 2 books, 2 more books was added to this student for checking the final results.

Statement:

```
create_trigger_on_can_borrow = """
create trigger limit_borrow_book_per_student
    before insert on can_borrow
begin
    select case
          when (select count(*) from can_borrow where stu_ID = new.stu_ID)
>= 3 then
              raise(abort, 'Max number of books student can loan at a time
is 3.')
    end;
end;"""

insert_borrow_valid = """insert into can_borrow (book_ID, stu_ID)
values ('114', '2052235');"""

insert_borrow_invalid = """insert into can_borrow (book_ID, stu_ID)
values ('118', '2052235'); """

c.execute(create_trigger_on_can_borrow)
conn.commit()
```

Table can_borrow after adding third book: student “2052235” now borrowing 3 books



Adding fourth book:

```
Execution finished with errors.
Result: Max number of books student can loan at a time is 3.
At line 1:
insert into can_borrow (book_ID, stu_ID)
values ('118', '2052235');
```

Task 6: Building a desktop, web, or mobile application to connect to the database.

I. Code implementation:

Starting with this web, we need following Python Libraries:

```
from flask import Flask, redirect, url_for, render_template, request,
session, flash
from datetime import timedelta
from flask_sqlalchemy import SQLAlchemy
from os import path
import sqlite3
from sqlalchemy.ext.automap import automap_base
```

Connecting to database by SQLAlchemy:

```
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///data.db"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

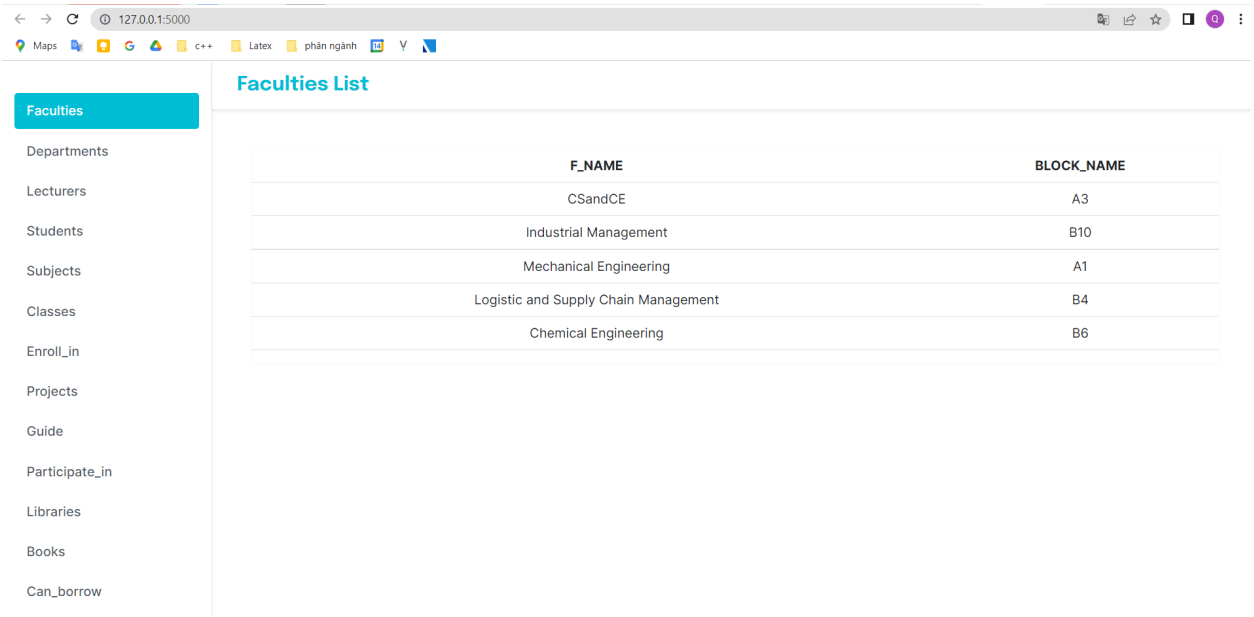
Connecting to tables of database with reflex and automap.

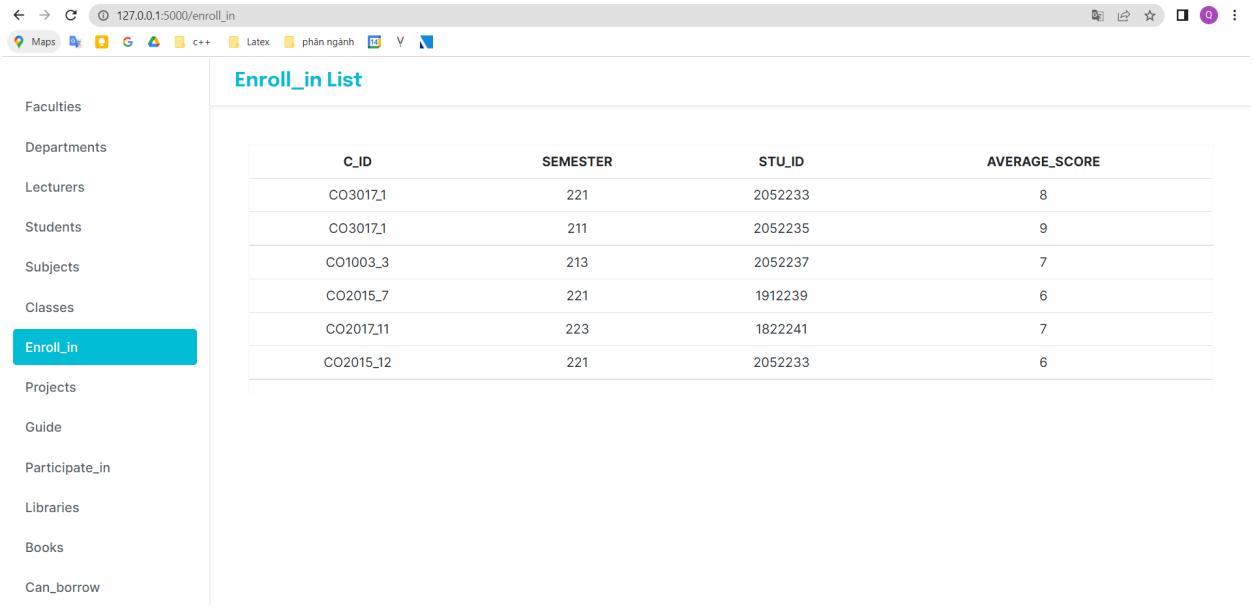
```
Base = automap_base()
Base.prepare (db.engine, reflect=True)
faculties = Base.classes.faculties
departments = Base.classes.departments
lecturers = Base.classes.lecturers
students = Base.classes.students
subjects = Base.classes.subjects
classes = Base.classes.classes
projects = Base.classes.projects
libraries = Base.classes.libraries
books = Base.classes.books

enroll_in = db.Table('enroll_in', db.metadata, autoload=True,
autoload_with=db.engine)
guide = db.Table('guide', db.metadata, autoload=True,
autoload_with=db.engine)
participate_in = db.Table('participate_in', db.metadata, autoload=True,
autoload_with=db.engine)
can_borrow = db.Table('can_borrow', db.metadata, autoload=True,
autoload_with=db.engine)
```

The remaining parts are create routes of app, combine with HTML and CSS for making UI for these tables.

II. Brief observation of web application:

Web application only for observing data table, no function as insert, update, delete… were performed.
Web application is run on local host.
The navbar shows up table name, from there, user can check 13 table currently in database.





III. Source code:

Source code can be obtained from our Github repository or directly from the attachment folder of this report. Please check for README file before running the code.

Github: https://github.com/vdtq12/DB_asm2.git

TASK 7: CREATE USER. Log in to the database with DBA privileges such as SYS / SYSTEM ...., create some users and assign the appropriate privileges to these users

In this task, we proceed to create a number of users that represent groups of users who share the same rights to the database, including:

- Users belong to management group (manager role): these users are managers of school system, have all privileges to the whole database.
- Users belong to teaching group (lecturer role): these users are lecturers of school. We assume they have view on tables, select privilege on some certain table and can update, delete, insert the things that they are managing as projects and participating member of these project.
- Users belong to learning group (student role): these users are students of the school, only have privileges to view data table and select some certain table. They do not have any privilege to change any information of the system by themselves.
- Users do not belong to school system (outside people): the users which are visitors, janitors, canteen staffs,...

Statement:
At first, we create roles and grant them privileges corresponding to each role

```
create role manager;
create role lecturer;
create role student;

grant all privileges to manager;
grant create session, create view to lecturer;
grant select on lecturers to lecturer;
grant select on classes to lecturer;
grant select on students to lecturer;
grant select, update, insert, delete on do_projects to lecturer;
grant select, update, insert, delete on participate_in to lecturer;

grant create session, create view to student;
grant select on participate_in to student;
grant select on enroll_in to student;
grant select on classes to student;
grant select on can_borrow to student;
grant select on books to student;
```

Next step, creating users and assign them to the roles.

```
create user manager_x identified by man;
create user lecturer_x identified by lec;
create user student_x identified by stu;
create user outside_people identified by out;

grant manager to manager_x;
grant lecturer to lecturer_x;
grant student to student_x;
```

In following sections, we can check how the logic work.

I. Users belong to management group (manager role):

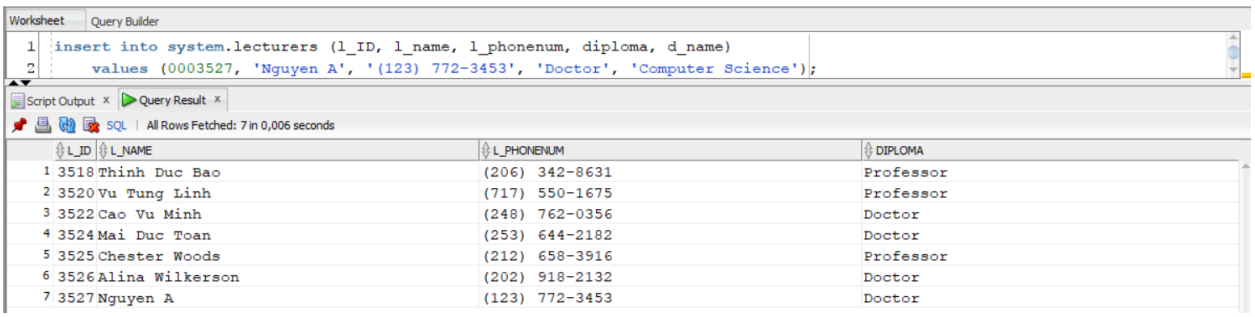
This user successfully query all things.

```
select * from system.lecturers;
```

Query Result — All Rows Fetched: 7 in 0,072 seconds

	L_ID	L_NAME	L_PHONENUM	DIPLOMA	D_NAME
1	3518	Thinh Duc Bao	(206) 342-8631	Professor	Computer Science
2	3520	Vu Tung Linh	(717) 550-1675	Professor	Marketing
3	3522	Cao Vu Minh	(248) 762-0356	Doctor	Manufacturing
4	3524	Mai Duc Toan	(253) 644-2182	Doctor	Computer Science
5	3525	Chester Woods	(212) 658-3916	Professor	Organic Chemistry
6	3526	Alina Wilkerson	(202) 918-2132	Doctor	Manufacturing Engineering
7	3527	Nguyen A	(123) 772-3453	Doctor	Computer Science

This user can insert values to table.

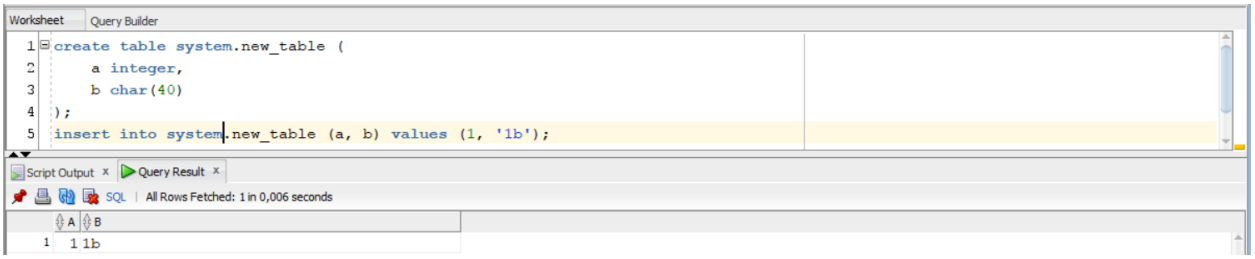


```
insert into system.lecturers (l_ID, l_name, l_phonenum, diploma, d_name)
    values (0003527, 'Nguyen A', '(123) 772-3453', 'Doctor', 'Computer Science');
```

Script Output — Query Result — All Rows Fetched: 7 in 0,006 seconds

	L_ID	L_NAME	L_PHONENUM	DIPLOMA
1	3518	Thinh Duc Bao	(206) 342-8631	Professor
2	3520	Vu Tung Linh	(717) 550-1675	Professor
3	3522	Cao Vu Minh	(248) 762-0356	Doctor
4	3524	Mai Duc Toan	(253) 644-2182	Doctor
5	3525	Chester Woods	(212) 658-3916	Professor
6	3526	Alina Wilkerson	(202) 918-2132	Doctor
7	3527	Nguyen A	(123) 772-3453	Doctor

Creating and insert new table



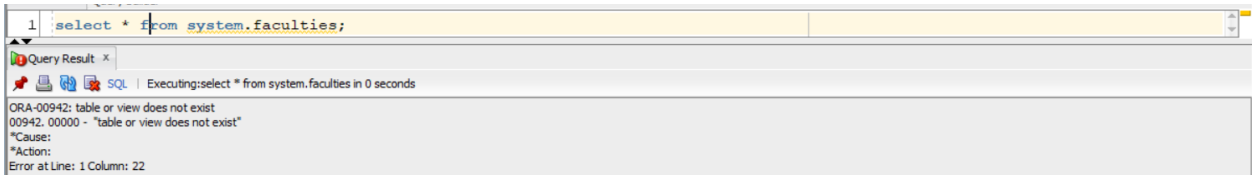
```
create table system.new_table (
    a integer,
    b char(40)
);
insert into system.new_table (a, b) values (1, '1b');
```

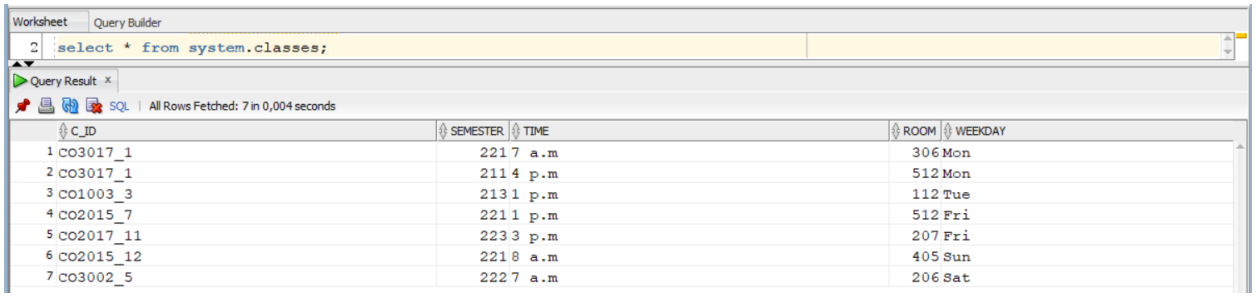
Script Output — Query Result — All Rows Fetched: 1 in 0,006 seconds

	A	B
1	1	1b

II. Users belong to teaching group (lecturer role):

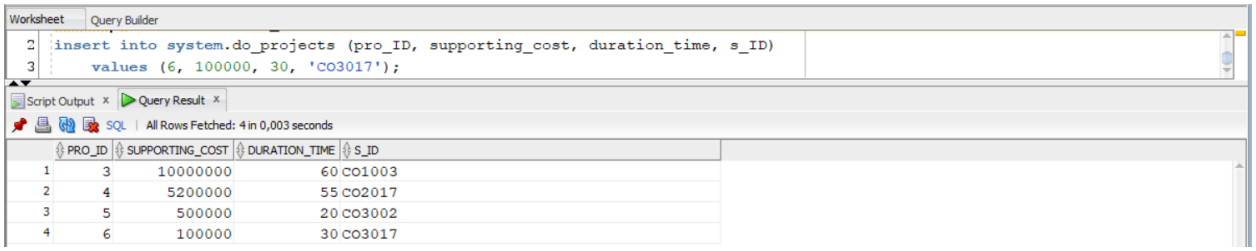
This user have access right for some certain tables which are already granted privileges.





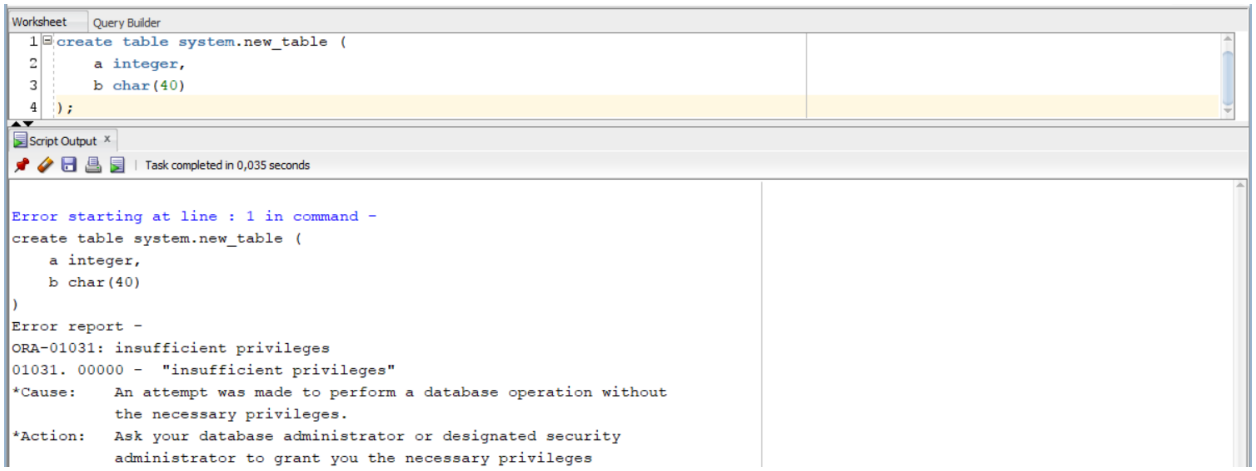
	C_ID	SEMESTER	TIME	ROOM	WEEKDAY
1	CO3017_1	221	7 a.m	306	Mon
2	CO3017_1	211	4 p.m	512	Mon
3	CO1003_3	213	1 p.m	112	Tue
4	CO2015_7	221	1 p.m	512	Fri
5	CO2017_11	223	3 p.m	207	Fri
6	CO2015_12	221	8 a.m	405	Sun
7	CO3002_5	222	7 a.m	206	Sat

This user have insert right for some certain tables which are already granted privileges.



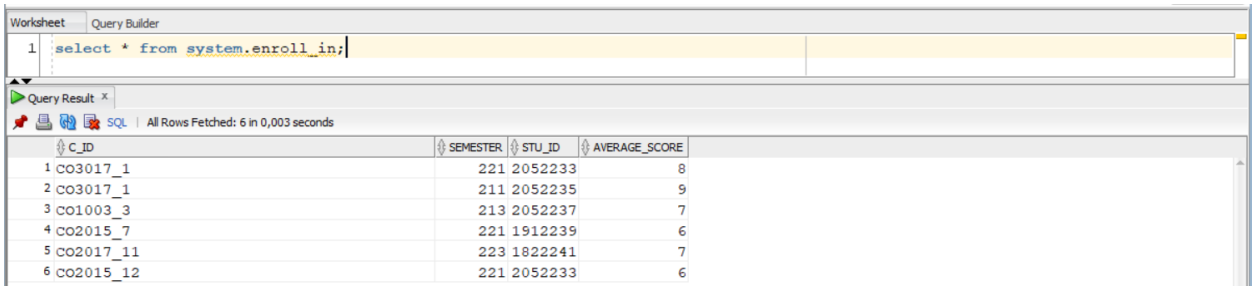
	PRO_ID	SUPPORTING_COST	DURATION_TIME	S_ID
1	3	10000000	60	CO1003
2	4	5200000	55	CO2017
3	5	500000	20	CO3002
4	6	100000	30	CO3017

This user can not create table because insufficient privileges.

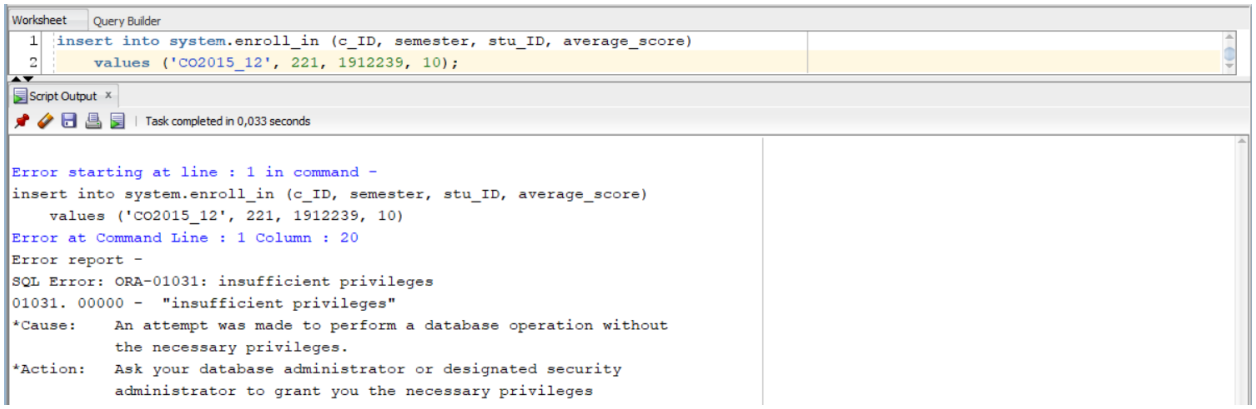


III. Users belong to learning group (student role):

This user have access and select right for some certain tables which are already granted privileges.



This user can not create or insert anything



IV. Users do not belong to school system (outside people role):

When this user try to access the database connection, he or she is denied.