NYU Tandon School of Engineering
CS-UY 3083-B
Professor Salim Arfaoui

Assignment 04: Advanced SQL

Due: 11:59pm, Sunday, April 06, 2025

This assignment covers the following topics

- **Multiple Tables Queries**
    - **Subqueries**
    - **Joins**
- **Procedural SQL**
    - **Procedures**
    - **Functions**
    - **Triggers**

Submission instructions

- You should submit your assignment on Gradescope.
- For this assignment you should turn in 2 separate files:
    - *(40 pts) MULTIPLE.sql*
    - *(60 pts) PLSQL.sql*

Each file you submit should contain a header comment block as follows:

```
--Author: [Your name here]
--Assignment# ???? / Part ???? (etc.)
--Date due: ?????
--I pledge that I have completed this assignment without collaborating
--with anyone else, in conformance with the NYU School of Engineering
--Policies and Procedures on Academic Misconduct.
```

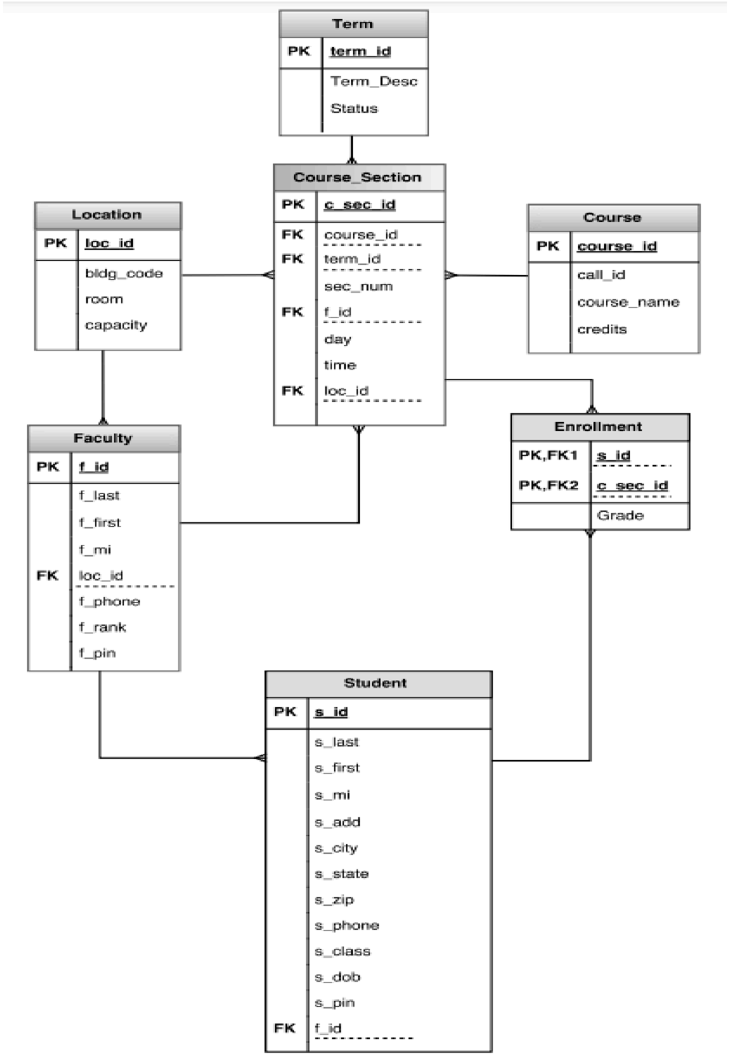
Grading:
- This assignment is made up of 2 parts, collectively worth 100 points.
- Homework must be submitted on time.
- Late submissions will not be accepted.

Getting Started:

This assignment gives you an opportunity to generate SQL queries against multiple tables. In addition, you will create stored procedures, functions, and triggers to test against the **`university`** database (you have created for assignment 1).

There is no starter file for this assignment. You should be able to complete this assignment given the **`university`** schema provided for assignment1.

```
location(loc_id, bldg_code, room, capacity)
faculty(f_id,f_last,f_first,f_mi,loc_id,f_phone, f_rank,f_pin,f_image)
student(s_id,s_last,s_first,s_mi,s_add,s_city,s_state,s_zip,s_phone,s_class,
                                                                  s_dob,s_pin,f_id)
term(term_id,term_desc,status)
course(course_id,call_id,course_name,credits)
course_section(c_sec_id,course_id,,term_id,sec_num,f_id,day,time,loc_id,max_enrl)
enrollment(s_id,c_sec_id,grade)
```



Part1: Multiple Tables Queries

Q1) Write a query that lists the building code and room of every room that is either currently in use as a faculty office or in use as a classroom during the Summer 2004 term. (Hint: You MAY use the UNION).

Q2) Write a query that retrieves a list of students who have never been taught by ‘Kim Cox’.

Q3) Write a query that calculates the total number of students taught by John Blanchard.

Q4) Write a query that calculates the total number of students taught by John Blanchard during the ‘Summer 2004’ term

Q5) Write a query that returns the full names of students, course section id, and grade for section id 1000. Order the records by grade, showing the A's first, then the B's, etc.

Q6) Write a query that returns the full names of faculty who taught course sections having more than 5 students enrolled.

Q7) Write a query that returns the course name(s) that generated the highest number of course sections.

Q8) Write a query that returns the course section ids that are either taught by a faculty with a last name starting with ‘B’ or taken by a student with last name containing ‘an’

What to submit:

- *`(40 pts / 5 pts each )`* ***`MULTIPLE.sql`***
- **The file should include all the `SQL` statements used to solve the above 8 questions.**

Part2: PL SQL: There is no starter file for this assignment. You should be able to complete this assignment given the **`university`** schema provided for **assignment1**.

You are to submit the `SQL` statement(s) used to solve the following 9 questions along with the descriptions where applicable.

Q1 *5pts*) Write a procedure **`track_student(first)`** that accepts a student first name and returns(displays) a result set that contains a list of the course_section_ids the student is enrolled in. The result set should contain the student full name and the course_section_ids the student is enrolled in.

Q2 *5pts*) Write a procedure **`track_faculty(first, last)`** that accepts a faculty full name and returns(displays) a result set that contains a count of the students taught by that faculty. The result set should contain the faculty name and the number of students taught by that faculty.

Q3 *5pts*) Write a function named **`count_enroll(sid, termid)`**. It accepts a student's id and a term id, and returns the total count of course sections the student is enrolled in during that term.

Q4 *5pts*) Write a function named **`course_most_sections()`** that returns the name of the course that generated the highest number of course sections.

Q5) To keep track of the total number of our admitted students in our university,

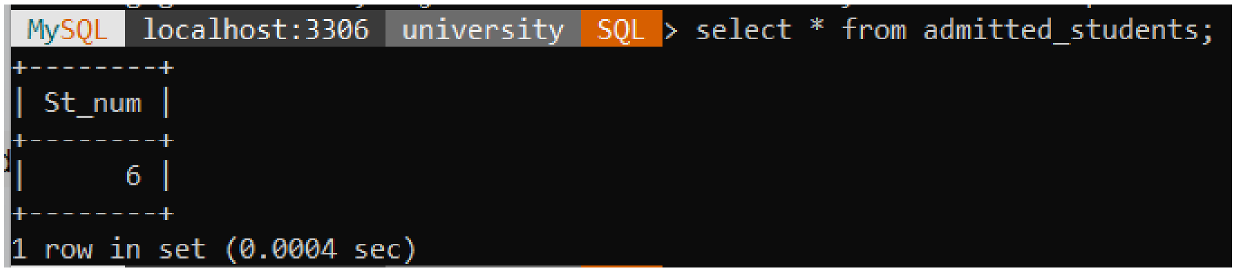
A) *2pts)* let's create the table **`admitted_students`** using the following **`SQL CREATE:`**

```
CREATE TABLE admitted_students(
      st_num NUMERIC (4)
);
```

B) *2pts)* Now let's insert into **`admitted_students`** one record with the current count of students in our database.

```
INSERT INTO admitted_students
VALUES((SELECT count(s_id) from student));
```

This is how the admitted_students should look like after completing the previous 2 steps:



```
MySQL  localhost:3306  university  SQL > select * from admitted_students;
+--------+
| St_num |
+--------+
|      6 |
+--------+
1 row in set (0.0004 sec)
```

C) *`10pts)`* Write a trigger that updates the value for the field **`st_num`** in the **`admitted_students`** table. The trigger should update the value of **`st_num`** in **`admitted_students`** table after a new student is added to the **`student`** table. Name the trigger **`student_watch_dog`**.

D) *`5pts)`* Next, Insert the following records into the database:

```
(106, 'Cruz', 'Ana', 'S', '100 Northern Blvd.', 'Eau Claire',
'WI', '54703', '7154449870', 'SR', '1982-08-13', 8891, 1);
(107, 'Katz', 'Daniel', 'B', '400 St. John''s Street', 'Eau Claire',
'WI', '54702', '7155552000', 'SR', '1982-04-10', 1230, 1);
```

E) *`5pts)`* Now execute a **`SELECT * FROM admitted_students;`**
Describe your findings.

Q6 *`3pts`***)** Execute the given prepared statement that calls **`track_student`** procedure with the argument ('Sarah')**.** Describe the outcome (*copy the results*)

```
CALL track_student('Sarah');
```

Q7 *`5pts`***)** Create and execute a prepared statement that calls **`track_faculty`** procedure with the argument ('Kim', 'Cox')**.** Describe the outcome (*copy the results*)

Q8 *`3pts`***)** Execute the given prepared statement that calls **`count_enroll`** function with the arguments (101, 5)**.** Describe the outcome (*copy the results*)

```
SET @count_e = count_enroll(101, 5);
SELECT @count_e;
```

Q9 *`5pts`***)** Create and execute a prepared statement that calls **`course_most_sections`**`()` function**.** Describe the outcome (*copy the results*)

What to submit:

- *`(60 pts)`* ***`PLSQL.sql`***
- **The file should include all the `SQL` statements used to solve the above 9 questions along with the descriptions where applicable.**