Daniela A. Gonzalez
Assigment 1 Technical Report

Assigment 1

Scenario 1

Objective: use aggregation, grouping, and subqueries to find data insights. In this case for getting the Departments where the average course credits are highest.
Tasks: write a query that finds the department(s) where the average course credits are the highest.
Analysis: I worked with the tables Department and Course. I retrieved the information from Department using the attribute department_name and also computed the average number of credit courses from Course.Course_credits. Then I joined the tables using the key department_id. Subsequently, I grouped the courses by Department and output the first two Departments with the highest averages.
Code:
```
SELECT Department.department_name, AVG(Course.Credits) AS avg_credits
FROM Department JOIN Course ON Department.department_id =
Course.department_id GROUP BY Department.department_name ORDER BY
avg_credits DESC LIMIT 2;
```
Results: a csv file with the two Departments with the highest averages of credits courses.The importance of this query lies in the fact that the information obtained may be useful when designing academic programs.
Challenges: I did not find a significant challenge in this query since we covered similar examples in class.

Scenario 2

Objective: Explore the use of grouping, counting, and comparison with subqueries to solve this problem which consists of identify (with name and last name) students enrolled in every course offered by their departments.
Tasks: Write an SQL query to find students who are enrolled in all the courses offered by the department they are part of.
Analysis: I got from the directions the tables I had to work with: Students, Enrollment and Course. I used a query to compare the total number of courses in each student's department with the total number of courses they are enrolled in; if the number is equal that meant that they were enrolled in all the courses offered be the department.
Code:
```
SELECT Student.first_name, Student.last_name FROM Student JOIN
Enrollment ON Student.student_id = Enrollment.student_id JOIN Course ON
Enrollment.course_id = Course.Course_id WHERE Student.department_id =
```

`Course.department_id GROUP BY Student.student_id HAVING COUNT(DISTINCT Course.Course_id) = ( SELECT COUNT(*) FROM Course WHERE Course.department_id = Student.department_id );`
Results: a csv file with the name and last name of the student who is enrolled in every course offered by their department. This query could help the university to detect possible students interested in a given area, which is in fact very useful information to develop projects with really interested students.
Challenges: This one was a quite hard for me to figure out. The main challenge was in the HAVING clause. I was able to did it following the examples for this given in W3.school website in the SQL section as well with the book "Learning Data Science" from Lau, Gonzalez and Nolan (2023).

Scenario 3

Objective: Identify Faculty members who are teaching Courses with fewer than 10 available seats.
Tasks: Query a list of faculty members and list the courses with low enrollment (fewer than 10 available seats).
Analysis: I selected Faculty.faculty_name to retrieve the names of the faculty members referencing the Course table which has the seats available. Then, I JOIN Department ON Course.department_id = Department.department_id to relate each course to the corresponding department (to access the faculty information through the department). After that I joined the Faculty table to get the faculty members associated with each department and finally filtered using the condition SeatsAvailable ¡ 10.
Code: `SELECT Faculty.faculty_name FROM Course JOIN Department ON Course.department_id = Department.department_id JOIN Faculty ON Department.faculty_id = Faculty.faculty_id WHERE Course.SeatsAvailable<10;`
Results: a csv file with none record in it since there is no offered course with less than 10 available seats. This query could offer essential information to the University to see which courses could be deleted, updated or added.
Challenges: I did not find any significant challenge in this query.