CS3700 - Introduction to Database Systems

Assignment 2 - SQL on Academic DB

Name - Jay Prajapati
Roll Number - ME21B143
Group No. - 11

Table of Contents

1 Professors Without Advisor or HOD Roles

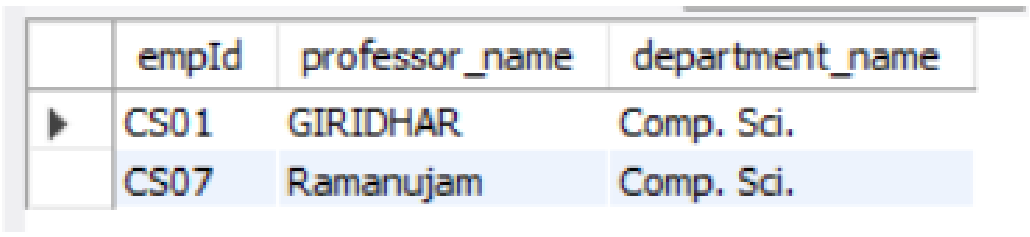
Give the list of professors and their department that are neither the advisor of any student nor hod of any department.

```
SELECT p.empId, p.name AS professor_name, d1.name
   AS department_name
FROM professor p
LEFT JOIN student s ON p.empId = s.advisor
LEFT JOIN department d ON p.empId = d.hod
JOIN department d1 ON p.deptNo = d1.deptId
WHERE s.advisor IS NULL AND d.hod IS NULL;
```

Breakdown of the query:

1. **Professor Table (p):** This is the main table from which we are retrieving professors information.
2. **LEFT JOIN with Student Table (s):** This join is used to check if a professor is an advisor. If a professor is not an advisor, `s.advisor` will be `NULL`.
3. **LEFT JOIN with Department Table (d):** This join checks if a professor is the HOD of any department. If not, `d.hod` will be `NULL`.
4. **JOIN with Department Table (d1):** This join is necessary to get the department name for each professor because if we use d for retrieving department name it will be NULL.
5. **WHERE Clause:** Filters out professors who are either advisors or HODs by ensuring both `s.advisor IS NULL` and `d.hod IS NULL`.

Output of the query:



	empId	professor_name	department_name
▶	CS01	GIRIDHAR	Comp. Sci.
	CS07	Ramanujam	Comp. Sci.

2 Comfortable Courses for Students

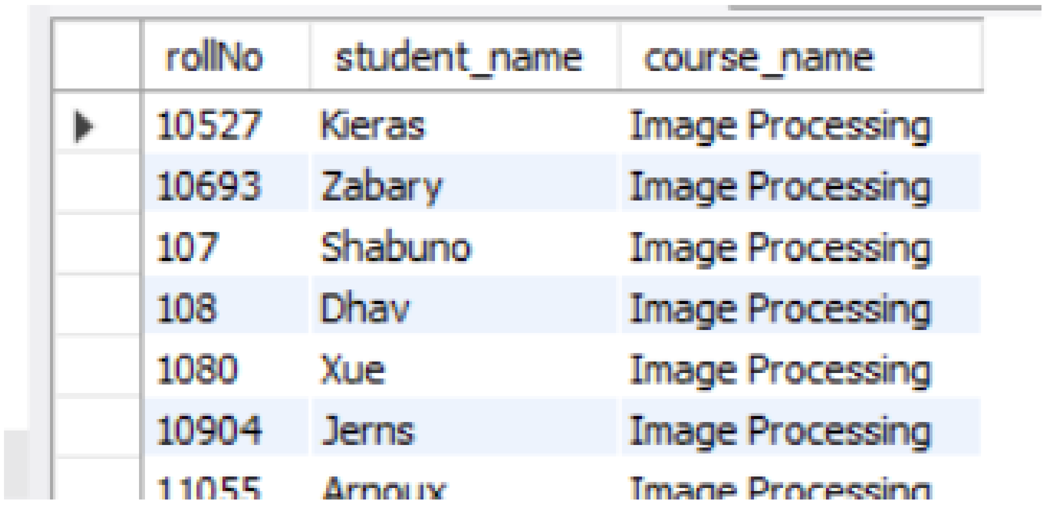
Give a list of students with the courses which can be comfortably done by the student. A course can be comfortably done if the course is not taught by the student's advisor or its less than 4 credits course.

```sql
SELECT s.rollNo, s.name AS student_name, c.cname AS
    course_name
FROM student s
JOIN enrollment e ON s.rollNo = e.rollNo
JOIN course c ON e.courseId = c.courseId
JOIN teaching t ON e.courseId = t.courseId
LEFT JOIN professor p ON s.advisor = p.empId AND
    t.empId = p.empId
WHERE p.empId IS NULL OR c.credits < 4;
```

Breakdown of the Query:

1. **Student Table (s):** This is the main table from which we are retrieving students.
2. **JOIN with Enrollment Table (e):** This join is used to identify the courses each student is enrolled in.
3. **JOIN with Course Table (c):** This join retrieves the names of the courses from their IDs.
4. **JOIN with Teaching Table (t):** This join identifies who teaches each course.
5. **LEFT JOIN with Professor Table (p):** This join checks if the course is taught by the student's advisor. If not, `p.empId` will be `NULL`.
6. **WHERE Clause:** Filters courses to include only those not taught by the student's advisor (`p.empId IS NULL`) or those with less than 4 credits (`c.credits < 4`).

Output of the query:



rollNo	student_name	course_name
10527	Kieras	Image Processing
10693	Zabary	Image Processing
107	Shabuno	Image Processing
108	Dhav	Image Processing
1080	Xue	Image Processing
10904	Jerns	Image Processing
11055	Arnoux	Image Processing

3 Prerequisite Course Completion History

Give list of students and courses and its prerequisite course with year in which they were done by a student, such that the prerequisite course was done in some year before the new course.

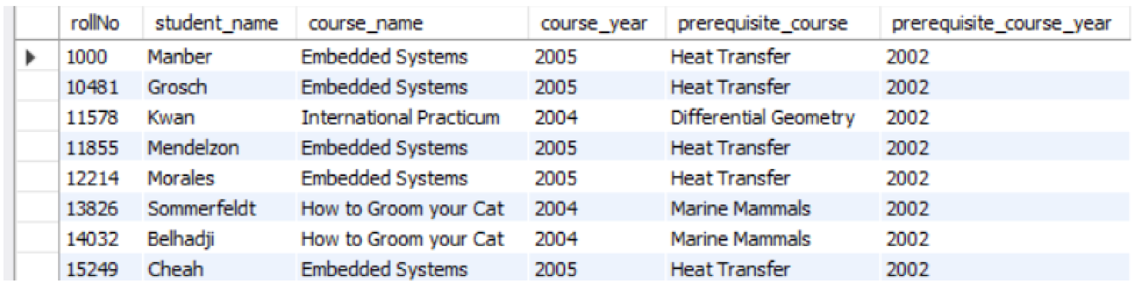
```
SELECT s.rollNo, s.name AS student_name, c1.cname
   AS course_name, e1.year as course_year, c2.cname
   AS prerequisite_course, e2.year as
   prerequisite_course_year
FROM student s
JOIN enrollment e1 ON s.rollNo = e1.rollNo
JOIN course c1 ON e1.courseId = c1.courseId
JOIN preRequisite pr ON c1.courseId = pr.courseId
JOIN course c2 ON pr.preReqCourse = c2.courseId
JOIN enrollment e2 ON (s.rollNo = e2.rollNo AND
   e2.courseId = c2.courseId)
WHERE e1.year > e2.year;
```

Breakdown of the Query:

1. **Student Table (s):** This is the main table from which we are retrieving students.
2. **JOIN with Enrollment Table (e1):** This join identifies the courses each student is enrolled in.

3. **JOIN with Course Table (c1):** This join retrieves the names of the courses from their IDs.
4. **JOIN with PreRequisite Table (pr):** This join identifies the prerequisite courses for each course.
5. **JOIN with Course Table (c2):** This join retrieves the names of the prerequisite courses.
6. **JOIN with Enrollment Table (e2):** This join ensures that the prerequisite course was completed by the same student.
7. **WHERE Clause:** Filters courses to include only those where the prerequisite course was completed in an earlier year.

Output of the query:



rollNo	student_name	course_name	course_year	prerequisite_course	prerequisite_course_year
1000	Manber	Embedded Systems	2005	Heat Transfer	2002
10481	Grosch	Embedded Systems	2005	Heat Transfer	2002
11578	Kwan	International Practicum	2004	Differential Geometry	2002
11855	Mendelzon	Embedded Systems	2005	Heat Transfer	2002
12214	Morales	Embedded Systems	2005	Heat Transfer	2002
13826	Sommerfeldt	How to Groom your Cat	2004	Marine Mammals	2002
14032	Belhadji	How to Groom your Cat	2004	Marine Mammals	2002
15249	Cheah	Embedded Systems	2005	Heat Transfer	2002

4 Even Semester Courses by Department

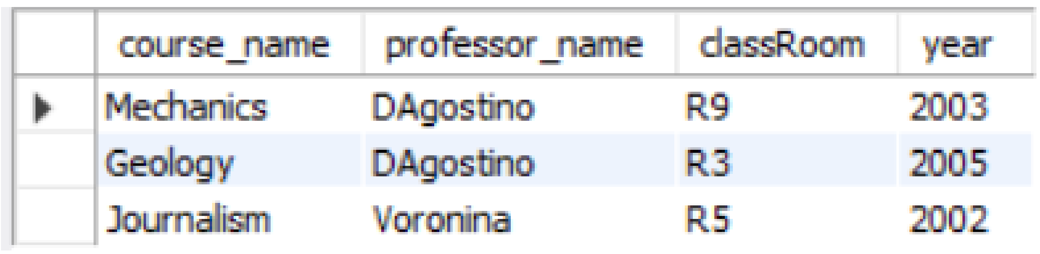
Give list of all classrooms with the course and professor name who taught in that classroom in even semester, which are offered by Physics or Psychology department which are taught by a professor who joined the institution on or after 1992.

```
SELECT c.cname AS course_name, p.name AS
   professor_name, t.classRoom, t.year
FROM teaching t
JOIN course c ON t.courseId = c.courseId
JOIN professor p ON t.empId = p.empId
JOIN department d ON c.deptNo = d.deptId
WHERE (d.name = 'Physics' OR d.name = 'Psychology')
AND t.sem = "even" AND p.startYear >= 1992;
```

Breakdown of the Query:

1. **Teaching Table (t):** This is the main table from which we are retrieving teaching information.
2. **JOIN with Course Table (c):** This join retrieves the names of the courses from their IDs.
3. **JOIN with Professor Table (p):** This join identifies the professors who taught the courses.
4. **JOIN with Department Table (d):** This join filters courses by department.
5. **WHERE Clause:** Filters courses to include only those taught in the even semester, offered by the Physics or Psychology department, and taught by professors who joined on or after 1992.

Output of the query:



	course_name	professor_name	classRoom	year
▶	Mechanics	DAgostino	R9	2003
	Geology	DAgostino	R3	2005
	Journalism	Voronina	R5	2002

The following 4 queries are implemented using aggregate functions, group by and having clause.

5 Professor to Student Ratio for Department

Give the list of departments with their professor to student ratio in the department, for departments having at least 3 courses, in decreasing order of this ratio.

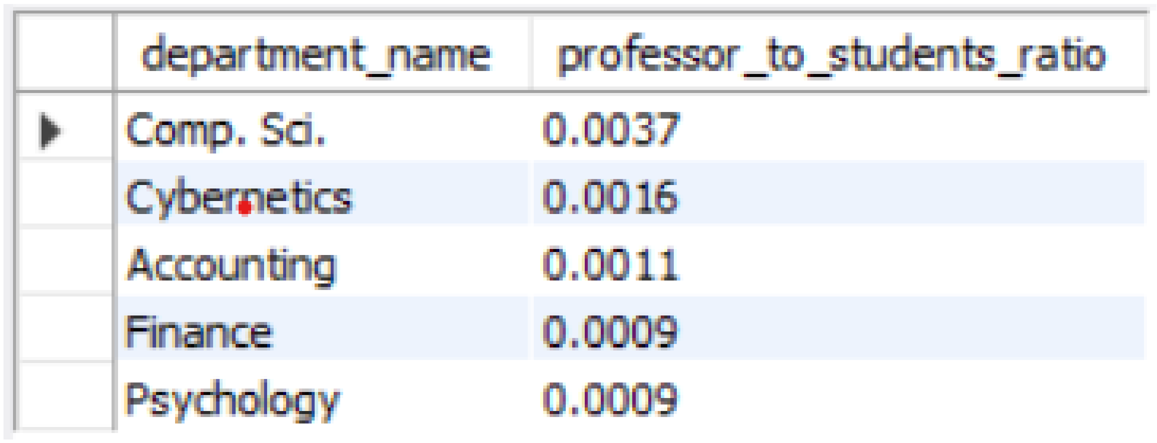
```sql
SELECT d.name AS department_name, COUNT(DISTINCT
  p.empId) / COUNT(DISTINCT s.rollNo) AS
  professor_to_students_ratio
FROM department d
JOIN course c ON d.deptId = c.deptNo
JOIN teaching t ON c.courseId = t.courseId
JOIN professor p ON t.empId = p.empId
JOIN enrollment e ON c.courseId = e.courseId
JOIN student s ON e.rollNo = s.rollNo
GROUP BY d.name
HAVING COUNT(DISTINCT c.courseId) >= 3
ORDER BY professor_to_students_ratio DESC;
```

Breakdown of the Query:

1. **Department Table (d):** This is the main table from which we are retrieving departments.
2. **JOIN with Course Table (c):** This join identifies the courses offered by each department.
3. **JOIN with Teaching Table (t):** This join identifies the professors teaching each course.
4. **JOIN with Professor Table (p):** This join retrieves the professors' information.
5. **JOIN with Enrollment Table (e):** This join identifies the students enrolled in each course.
6. **JOIN with Student Table (s):** This join retrieves the students' information.
7. **GROUP BY Clause:** Groups the results by department name.

8. **HAVING Clause:** Filters departments to include only those with at least 3 courses.
9. **ORDER BY Clause:** Orders the results by the professor to student ratio in descending order.

Output of the query:



	department_name	professor_to_students_ratio
▶	Comp. Sci.	0.0037
	Cybernetics	0.0016
	Accounting	0.0011
	Finance	0.0009
	Psychology	0.0009

6 Degree Enrollment Statistics

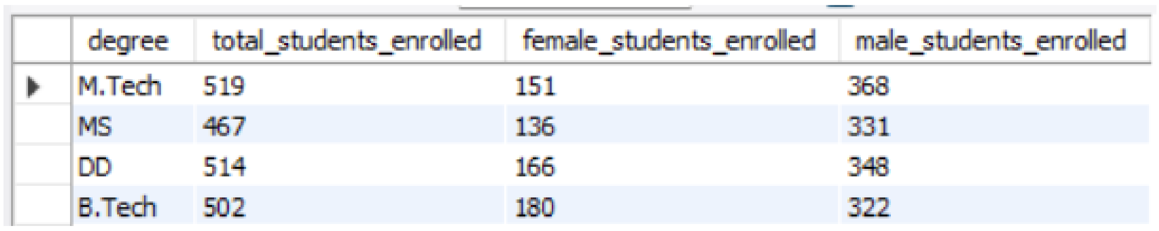
Give the list of different degrees with the total number of students, number of female and male enrolled in it, such that the number of students enrolled in the degree has at least 20% female students, in increasing order of female to male ratio.

```
SELECT degree, COUNT(rollNo) AS
   total_students_enrolled, COUNT(sex = 'female' OR
   NULL) AS female_students_enrolled, COUNT(sex =
   'male' OR NULL) AS male_students_enrolled
FROM student
GROUP BY degree
HAVING SUM(sex = 'female' OR NULL) >= 0.2 *
   COUNT(rollNo)
ORDER BY COUNT(sex = 'female' OR NULL)/COUNT(sex =
   'male' OR NULL);
```

Breakdown of the Query:

1. **Student Table:** This is the main table from which we are retrieving student information.
2. **GROUP BY Clause:** Groups the results by degree.
3. **COUNT Functions:** - `COUNT(rollNo)` counts the total number of students enrolled in each degree. - The query attempts to count female and male students using `COUNT(sex = ’female’ OR NULL)` and `COUNT(sex = ’male’ OR NULL)`.
4. **HAVING Clause:** Filters degrees to include only those where at least 20
5. **ORDER BY Clause:** Orders the results by the female to male ratio in increasing order.

Output of the query:



degree	total_students_enrolled	female_students_enrolled	male_students_enrolled
M.Tech	519	151	368
MS	467	136	331
DD	514	166	348
B.Tech	502	180	322

7 Professors with High ’S’ Grade Counts

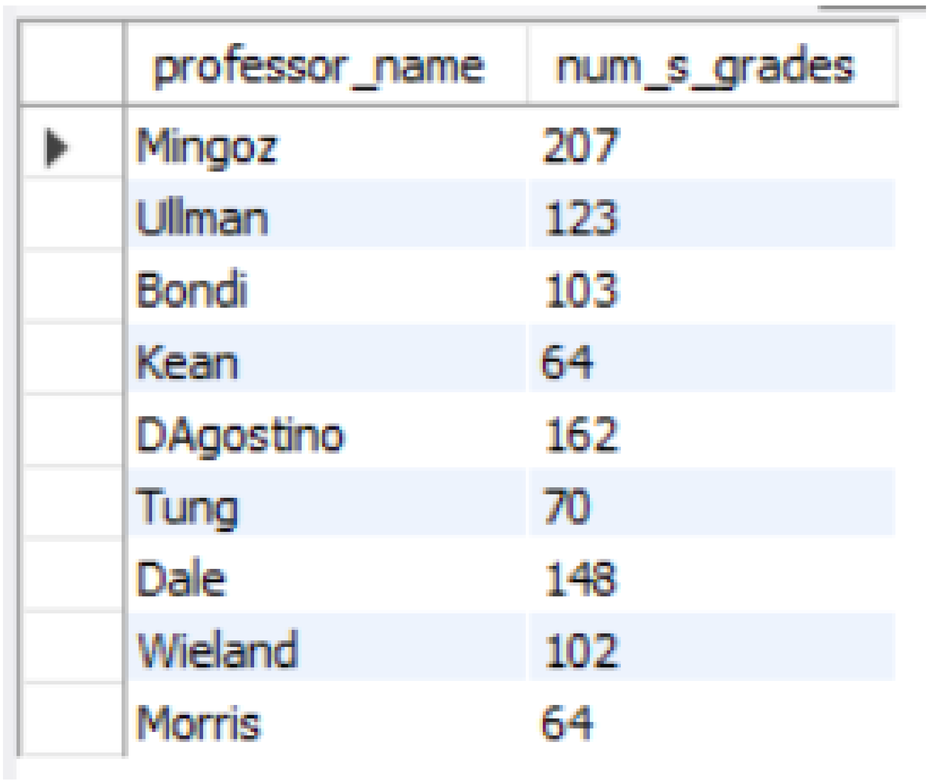
Give list of professors with the count of total number of ’S’ grades given by him/her to the students who enrolled to any course taught by that professor, such that the count of ’S’ grades is higher than the average number of ’S’ grades provided by all professors in institute.

```
SELECT p.name AS professor_name, COUNT(e.grade =
   'S' OR NULL) AS num_s_grades
FROM professor p
JOIN teaching t ON p.empId = t.empId
JOIN enrollment e ON t.courseId = e.courseId
GROUP BY p.empId
HAVING COUNT(e.grade = 'S' OR NULL) > (
  SELECT AVG(num_s_grades)
  FROM (
    SELECT COUNT(e.grade = 'S' OR NULL) AS
       num_s_grades
    FROM professor p
    JOIN teaching t ON p.empId = t.empId
    JOIN enrollment e ON t.courseId = e.courseId
    GROUP BY p.empId
  ) AS avg_s_grades
);
```

Breakdown of the Query:

1. **Professor Table (p):** This is the main table from which we are retrieving professors.
2. **JOIN with Teaching Table (t):** This join identifies the courses taught by each professor.
3. **JOIN with Enrollment Table (e):** This join retrieves the grades given by each professor.
4. **GROUP BY Clause:** Groups the results by professor ID.
5. **COUNT Function:** - The query attempts to count 'S' grades using `COUNT(e.grade = 'S' OR NULL)`
6. **HAVING Clause:** Filters professors to include only those whose 'S' grade count exceeds the average 'S' grade count across all professors.
7. **Subquery:** Calculates the average number of 'S' grades given by all professors.

Output of the query:



	professor_name	num_s_grades
▶	Mingoz	207
	Ullman	123
	Bondi	103
	Kean	64
	DAgostino	162
	Tung	70
	Dale	148
	Wieland	102
	Morris	64

8 Top Advisors of Department

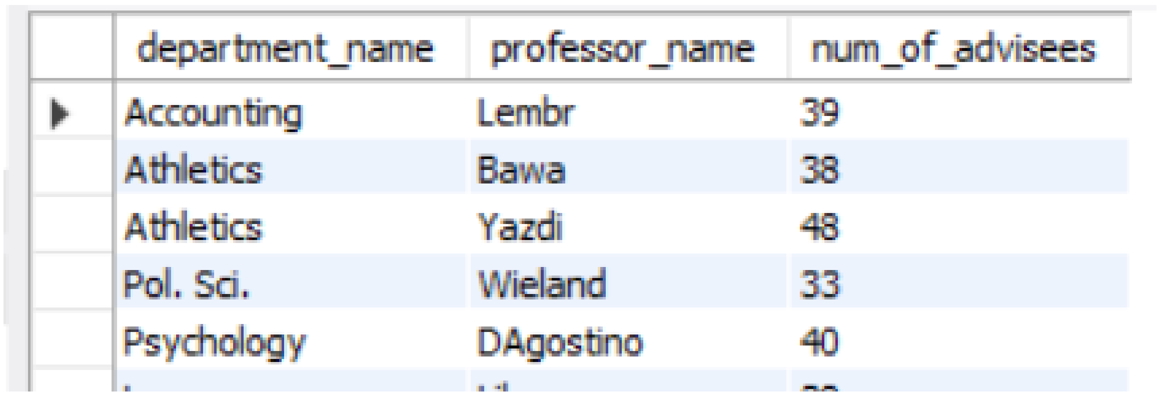
Give list of professors who is advisor of maximum number of students in each department

```
SELECT dd.name AS department_name, pp.name as
   professor_name,
p.max_advisees AS num_of_advisees
FROM (
  SELECT p.empId, MAX(p.advisees) AS max_advisees
  FROM (
    SELECT p.empId, COUNT(p.empId) AS advisees
    FROM department d
    JOIN professor p ON p.deptNo = d.deptId
    JOIN student s ON s.advisor = p.empId
    GROUP BY p.empId
  ) AS p
  GROUP BY p.empId
) AS p
JOIN professor pp ON p.empId = pp.empId
JOIN department dd ON pp.deptNo = dd.deptId;
```

Breakdown of the Query:

1. **Department Table (d):** This is one of the tables used to filter professors by department.
2. **Professor Table (p):** This table contains information about professors, including their department affiliation.
3. **Student Table (s):** This table is used to identify the advisor for each student.
4. **JOIN Operations:** The query joins the department, professor, and student tables to link advisors with their advisees.
5. **GROUP BY Clause:** Groups the results by professor name to count the number of students each professor advises.
6. **HAVING Clause:** This clause is used in a subquery to find the maximum number of advisees per department.
7. **Subquery:** Calculates the maximum number of advisees for each department.

Output of the query:



department_name	professor_name	num_of_advisees
Accounting	Lembr	39
Athletics	Bawa	38
Athletics	Yazdi	48
Pol. Sci.	Wieland	33
Psychology	DAgostino	40
[illegible]	[illegible]	[illegible]