Data management for data science

prof. Riccardo Rosati
Master (laurea magistrale) in Data Science
Sapienza Università di Roma, 2021/2022

Exercise on SQL

We want to build a relational database about the domain of students and exams. In particular, we want to store information about students (name, birthdate, ID (matricola), enrollment year, city, address), course editions (name, number of CFUs, year, semester, professor), exams (student, course name, year, grade), exam reservations (student, course name, year).

1. Write SQL statements that define the schema of the above described database;
2. Write SQL statements that insert some tuples in each of the tables defined at the previous point;
3. Write SQL statements that express the following queries:
   (a) return the names of all the students living in Rome;
   (b) return the names of the professors of the exams passed by John Doe;
   (c) return the names of the professors of the exams passed by John Doe in 2021;
   (d) return the ID and the birthdate of all the students that have passed at least an exam in 2021;
   (e) return name and number of CFUs of all the courses that were passed by the students enrolled in 2020;
   (f) return the names of the professors that have registered exams that were not reserved by students;
   (g) for every student, return the name and the number of exams passed by the student;
   (h) for every student, return the name and the average grade of the of exams passed by the student;
   (i) for every student, return the name and the number of exams that were reserved but not passed by the student;
   (j) return the ID and the birthdate of every student such that the total amount of CFUs of the exams passed by the student in 2021 is less than 20;
   (k) return the professor(s) who registered the maximum number of exams with the maximum grade (either 30 or 30 cum laude).

Solution of point 1:

Schema:

```
CREATE TABLE Student (name CHAR(30), birthdate INT, studentID INT,
  enrollmentYear INT, city CHAR(30), address CHAR(30));
CREATE TABLE CourseEdition (name CHAR(30), cfus INT, year INT,
  semester INT, professor CHAR(30));
CREATE TABLE Exam (studentID INT, courseName CHAR(30), year INT, grade INT);
CREATE TABLE ExamReservation (studentID INT, courseName CHAR(30), year INT);
```

Solution of point 3 (b):

Solution 1:

```
SELECT professor
FROM CourseEdition ce, Exam e, student s
WHERE e.courseName=ce.name
AND e.year=ce.year
AND s.studentID=e.studentID
AND s.name='John Doe';
```

Solution 2:

```
SELECT professor
FROM CourseEdition ce
WHERE (ce.name,ce.year) IN (
  SELECT e.courseName, e.year
  FROM Exam e, student s
  WHERE s.studentID=e.studentID
  AND s.name='John Doe');
```

Solution 3:

```
SELECT professor
FROM CourseEdition ce
WHERE EXISTS (
  SELECT *
  FROM Exam e
  WHERE e.courseName=ce.name
  AND e.year=ce.year
  AND EXISTS (
    SELECT *
    FROM Student s
    WHERE s.studentID=e.studentID
    AND s.name='John Doe' ));
```

Solution of point 3 (d):

Solution 1:

```
SELECT s.studentID, s.birthdate
FROM Exam e, Student s
WHERE s.studentID=e.studentID
AND e.year=2021;
```

Solution 2:

```
SELECT s.studentID, s.birthdate
FROM Exam e JOIN Student s ON s.studentID=e.studentID
WHERE e.year=2021;
```

Solution 3:

```
SELECT s.studentID, s.birthdate
FROM Student s
WHERE EXISTS (
  SELECT *
  FROM Exam e
  WHERE s.studentID=e.studentID
  AND e.year=2021);
```

Solution of point 3 (e):

Solution 1:

```
SELECT ce.name, ce.cfus
FROM CourseEdition ce, Exam e, Student s
WHERE s.studentID=e.studentID
AND ce.name=e.courseName
AND ce.year=e.year
AND s.enrollmentYear=2021;
```

Solution 2:

```
SELECT ce.name, ce.cfus
FROM CourseEdition ce
WHERE (ce.name,ce.year) IN (
  SELECT e.courseName, e.year
  FROM Exam e, Student s
  WHERE s.studentID=e.studentID
  AND s.enrollmentYear=2021);
```

Solution of point 3 (f):

Solution 1:

```
SELECT professor
FROM CourseEdition ce
WHERE EXISTS (
  SELECT *
  FROM Exam e
  WHERE e.courseName=ce.name
  AND e.year=ce.year
  AND NOT EXISTS (
    SELECT *
    FROM ExamReservation er
    WHERE er.studentID=e.studentID
    AND er.courseName=e.courseName
    AND er.year=e.year))
```

Solution 2:

```
SELECT professor
FROM CourseEdition ce, Exam e
WHERE e.courseName=ce.name
AND e.year=ce.year
AND NOT EXISTS (
  SELECT *
  FROM ExamReservation er
  WHERE er.studentID=e.studentID
  AND er.courseName=e.courseName
  AND er.year=e.year)
```

Solution 3:

```
SELECT professor
FROM CourseEdition ce, Exam e
WHERE e.courseName=ce.name
AND e.year=ce.year
AND (e.studentID, e.courseName, e.year) NOT IN (
  SELECT *
  FROM ExamReservation er)
```

Solution of point 3 (g):

Solution 1:

```
SELECT s.name, count(*)
FROM Student s, Exam e
WHERE s.studentID=e.studentID
GROUP BY s.name;
```

This solution does not take into account the possibility that two (or more) students may have the same name.

Solution 2:

```
SELECT s.name, count(*)
FROM Student s, Exam e
WHERE s.studentID=e.studentID
GROUP BY s.name, s.studentID;
```

This solution groups by student name AND student ID, thus overcoming the problem of the previous solution. Actually, if we assume that the student ID is unique, we can just group by this attribute without changing the meaning of the query:

```
SELECT s.name, count(*)
FROM Student s, Exam e
WHERE s.studentID=e.studentID
GROUP BY s.studentID;
```

Solution of point 3 (h):

Solution 1:

```
SELECT s.name, avg(e.grade)
FROM Student s, Exam e
WHERE s.studentID=e.studentID
GROUP BY s.name;
```

This solution does not take into account the possibility that two (or more) students may have the same name.

Solution 2:

```
SELECT s.name, avg(e.grade)
FROM Student s, Exam e
WHERE s.studentID=e.studentID
GROUP BY s.studentID;
```

As illustrated in the previous point, this solution groups by student ID rather than student name, thus overcoming the problem of the previous solution.

Solution of point 3 (i):

Solution 1:

First, let us define the SQL query returning the exams that were reserved but not passed by the students:

```
SELECT *
FROM Exam e
WHERE (e.studentID, e.courseName, e.year) NOT IN (
  SELECT *
  FROM examReservation);
```

Using the above query as a subquery (in the FROM clause), we can now write the query requested by point (i):

```
SELECT s.name, count(*)
FROM Student s,
     (SELECT *
      FROM Exam e
      WHERE (e.studentID, e.courseName, e.year) NOT IN (
        SELECT *
        FROM examReservation)) examNotReserved
WHERE s.studentID=examNotReserved.studentID
GROUP BY s.name;
```

Solution 2:

As already explained, if in the previous solution we group by student ID rather than by student name, we can correctly handle the presence of multiple students with the same name:

```
SELECT s.name, count(*)
FROM Student s,
     (SELECT *
      FROM Exam e
      WHERE (e.studentID, e.courseName, e.year) NOT IN (
        SELECT *
        FROM examReservation)) examNotReserved
WHERE s.studentID=examNotReserved.studentID
GROUP BY s.studentID;
```

Solution 3:

```
SELECT s.name, count(*)
FROM Student s, Exam e
WHERE s.studentID=e.studentID
AND (e.studentID, e.courseName, e.year) NOT IN (
  SELECT *
  FROM examReservation)
GROUP BY s.studentID;
```