SQL Exercises: Course Database

Example Database

The questions refer to the example database created by the following SQL commands:

```
Create table Students(studentID int primary key, name text);
Create table Courses(courseID int primary key, name text);
Create table Enrollment(courseID int, studentID int);
Alter table Enrollment add primary key (courseID, studentID);
Alter table Enrollment
add foreign key (courseID) references Courses(courseID);
Alter table Enrollment
add foreign key (studentID) references Students(studentID);
```

This database stores information on university courses, students, and their enrollments. For each student, it stores the unique student ID and the student name. For each course, it stores the unique course ID and the course name. The Enrollment table stores pairs of student and course IDs with the semantics that the corresponding student is enrolled for the corresponding course.

Question 1 (Easy)

Write an SQL query that retrieves the students whose name starts with an "A". The query result has two columns, the student ID and the student name.

Question 1 - Solution

```
Select * from Students where name like 'A%';
```

The query uses the SQL LIKE predicate to filter to students whose name starts with an A (note that the query, as formulated, assumes that the first letter is capitalized). The "%" symbol substitutes arbitrary text.

Question 2 (Easy)

Write an SQL query that counts the number of courses in which the student named "Peter Codd" is involved (you can assume that this student's name is unique). The query result contains one single column with the course count.

Question 2 - Solution

```
Select count(*) from students S join enrollment E on
(S.studentID = E.studentID) where S.name = 'Peter Codd';
```

The exercise description refers to the student name which is only available in the Students table. Hence, we need the Students table in our From clause. Also, the exercise description refers to the number of enrollments. As that information is only available in the Enrollment table, we need to join both, the Students and Enrollment table. We filter to the student named "Peter Codd" and finally count the number of result rows (which equals the number of courses).

Question 3 (Easy)

Write an SQL query retrieving the IDs of all students who are enrolled in the courses named "Intro to Databases" and "Advanced Databases". The query result contains one single column with the student IDs. You can assume that no student is enrolled in both courses at the same time.

Question 3 - Solution

```
Select E.studentID From Enrollment E join Courses C on
(E.courseID = C.courseID) and C.name in ('Intro to Databases',
'Advanced Databases');
```

The SQL query filters the Courses table to the two relevant ones (with names "Intro to Databases" and "Advanced Databases") and then joins the filtered table with the Enrollment table. Finally, it selects the student ID from the Enrollment table.

Question 4 (Medium)

Write an SQL query that retrieves the names of all courses in which the student named "Robert Gray" is enrolled. The query result contains one column with the course name.

Question 4 - Solution

```
Select C.name from Courses C, Enrollment E, Students S where
C.courseID = E.courseID and E.studentID = S.studentID and S.name
= 'Robert Gray';
```

The query requires access to all three tables since each of them contains relevant information for the exercise. Only Students contains the name of students, required for filtering. Only the Courses table contains the names of courses (which is the desired output) and only the Enrollment table contains information connecting courses with students. The solution query joins all three tables (the corresponding join conditions can be found in the Where clause), filters students to the one with the right name, and finally selects the course name.

Question 5 (Medium)

Write an SQL query counting for each student the number of courses in which the student is enrolled in. The query result contains two columns with the student name and the associated enrollment count.

Question 5 - Solution

```
Select S.name, count(*) from Students S, Enrollment E where
S.studentID = E.studentID group by S.studentID, S.name;
```

We need the name of students which is only contained in the Students table. Also, we need information on the student's course enrollments which is only contained in Enrollment. Hence, we join the two tables. To calculate the number of enrollments per student, we group by the student ID. As the student ID is the primary key, it implies the student name (which appears in the select clause). Some database system may require using the student name directly in the Group By clause (even though it is, in fact, redundant as the ID implies the name). Note that, strictly speaking, using only the student name in the Group By clause is insufficient as the student name is not guaranteed to be unique.

Question 6 (Medium)

Write an SQL query that sorts courses by the number of enrolled students, sorted in descending order (i.e., starting with the course with most enrollments). The query result contains two columns, the course ID and the enrollment count. You can assume that each course has at least one enrolled student.

Question 6 - Solution

```
Select courseID, count(*) From Enrollment group by courseID
order by count(*) desc;
```

The query only requires access to the Enrollment table (the only table present in the From clause). We group enrollments by the course ID and select the course ID, along with the count as output columns. We order the output rows by the counts in descending order.

Question 7 (Medium)

Write an SQL query retrieving the student with most course enrollments. You can assume that there is only one single such student. The query result contains one single row with the student name.

Question 7 - Solution

```
Select S.name From Students S join Enrollment E on (S.studentID
= E.studentID) group by S.studentID, S.name order by count(*)
desc limit 1;
```

We need access to the Students table (to retrieve the student name) and the Enrollment table (to count the number of enrollments). We join both and group by the student ID and name (the latter is, strictly speaking, redundant, similar to the previous exercise). We sort in decreasing order of counts and select only the first row (which stores the student with most enrollments).

Question 8 (Hard)

Write an SQL query counting for each student the number of enrollments. The query result contains two columns, one with the student ID and one with the enrollment count. Make sure to handle students correctly who are not currently enrolled in any courses.

Question 8 - Solution

```
Select S.studentID, count(courseID) From Students S left outer
join Enrollment E on (S.studentID = E.studentID);
```

We need to make sure that students without any enrollments appear in the output. This is not the case if using an inner join (since unmatched students won't appear in the output). Instead, we need to use an outer join which preserves rows without matches in the output. In this case, we want to preserve students without matching enrollments. Hence, we use a left outer join with the Students table on the left hand side. If students have no matching enrollments, the associated courseID field from the Enrollment table is filled with the SQL NULL value. In that case, the associated rows should not count toward the number of enrollments. For that purpose, we use the count aggregate with the courseID column as parameter (rather than a count(*)).

Question 9 (Hard)

Write an SQL query retrieving all students who are enrolled in the "Advanced Databases" course without having been enrolled in the "Intro to Databases" course. The query result contains one column with the student ID.

Question 9 - Solution

```
Select S.studentID From Students S where
exists (select * from Courses C join Enrollment E on (C.courseID
= E.courseID) where C.name = 'Advanced Database Systems' and
E.studentID = S.studentID) and
not exists (select * from Courses C join Enrollment E on
(C.courseID = E.courseID) where C.name = 'Intro to Databases'
and E.studentID = S.studentID)
```

The query uses two sub-queries (of course, there are many ways to write the query without two sub-queries). For each student, it ensures that an entry exists representing an enrollment for that student in the "Advanced Database Systems" course. Also, it ensures that no entry exists representing an enrollment of that student in the "Intro to Databases" course.

Question 10 (Hard)

Write a query retrieving students, if any, who are enrolled in all courses in the database. The query result contains one single column with the student ID.

Question 10 - Solution

```
Select S.studentID from Students S where not exists (Select *
from Courses C where not exists (Select * from Enrollment E
where E.courseID = C.courseID and E.studentID = S.studentID));
```

This query essentially realizes a "double negation". For each student, it verifies whether there are no courses for which there are no enrollments associated with that course and that student. If there are no courses without corresponding enrollments then the student must be enrolled in all courses.