Session 3 reflection

1. A query that contains a join of any kind.

SELECT firstname, course, grade FROM students JOIN grades USING (studentid) ORDER BY grade;

Table

Description automatically generated with medium confidence

2. A query that contains a join of any kind and filtering into two or more cases (for example students who pass or fail, from the example data or a variant thereof). Case with pass over 65, or add letter grade.

SELECT studentid, firstname, programname, grade,

CASE

WHEN grade < 50 THEN 'F'

WHEN grade >= 50 AND grade < 55 THEN 'D'

WHEN grade >= 55 AND grade < 60 THEN 'C'

WHEN grade >= 60 AND grade < 65 THEN 'C+'

WHEN grade >= 65 AND grade < 70 THEN 'B-'

WHEN grade >= 70 AND grade < 75 THEN 'B'

WHEN grade >= 75 AND grade < 80 THEN 'B+'

WHEN grade >= 80 AND grade < 85 THEN 'A'

WHEN grade >= 50 THEN 'A+'

END AS letterGrade

FROM students JOIN grades USING (studentid);

Teams

Description automatically generated with medium confidence

3. A query that contains a join of any kind, an ordering of some sort, and a cut-off to show only a part of the results (for example the names of the students who hold the best five grades in the example data).

SELECT firstname, AVG(grade) AS avggrade FROM students JOIN grades USING (studentid) GROUP BY studentid ORDER BY avggrade DESC LIMIT 5;

Table

Description automatically generated

4. A query that groups the results of a join by one or more columns.

SELECT programname, course, COUNT(grade) AS total FROM students JOIN grades USING (studentid) GROUP BY (students.programname, grades.course) ORDER BY programname, total DESC;

Table

Description automatically generated

5. A query that counts matches, possibly by groups.

SELECT programname, COUNT(grade) AS total FROM students JOIN grades USING (studentid) GROUP BY (students.programname) ORDER BY total DESC;

A picture containing application

Description automatically generated

6. A query that sums over a column, possibly by groups.

SELECT programname, course, SUM(grade) AS total FROM students JOIN grades USING (studentid) GROUP BY (students.programname, grades.course) ORDER BY programname, total DESC;

Table

Description automatically generated

7. A query that averages over a column, possibly by groups.

SELECT programname, AVG(grade) AS total FROM students JOIN grades USING (studentid) GROUP BY (students.programname) ORDER BY total DESC;

A picture containing table

Description automatically generated

8. A query that uses a join and null fields to gain useful insights (such as how many students have not completed any courses, per program, in the example data).

SELECT programname, COUNT(\*) FROM students FULL OUTER JOIN grades USING (studentid) WHERE course ISNULL GROUP BY (students.programname);

A picture containing graphical user interface

Description automatically generated

9. Describe a situation in which a cross join could be useful or, failing to come up with one, discuss why that tends to be an impractical thing to compute. (You are welcome to discuss the latter even if you come up with the former.)

Add new to a university programme, like ours (with fixed subjects all compulsory and no optional ones), it’s easier just to cross join two tables. One containing the list of subjects in the program, and another containing the list of the new students.

SELECT studentid, firstname, subject FROM students CROSS JOIN programs ORDER by studentid;

A picture containing table

Description automatically generated