CSCC43 UTSC

Tutorial Week 5 – Introduction to SQL

Part 1:

Schema

Student (sID, surName, firstName, campus, email, cgpa)

Course (dept, cNum, name, breadth)

Offering (oID, dept, cNum, term, instructor)

Took (sID, oID, grade)

Offering [dept, cNum] ⊆ Course [dept, cNum]

Took [sID] ⊆ Student [sID]

Took [oID] ⊆ Offering [oID]

Questions

Write a query for each of the following:

1. Answer each of the following questions with an arithmetic expression. Suppose a row occurs n times in table R and m times in table S.
   (a) Using bag semantics, how many times will it occur in table $R \cup S$?
   (b) Using bag semantics, how many times will it occur in table $R \cap S$?
   (c) Using bag semantics, how many times will it occur in table $R - S$?

   Solution:
   $(a)\ n + m$
   $(b)\ min(n, m)$
   $(c)\ max(n - m, 0)$

2. Use a set operation to find all terms when Jepson and Suzuki were both teaching. Include every occurrence of a term from the result of both operands.

   Output:

```
term
-------
20089
20081
20081
```

```
(3 rows)
```

Solution

```
(SELECT Term FROM Offering WHERE instructor = 'Suzuki')
intersect all
(SELECT Term FROM Offering WHERE instructor = 'Jepson');
```

3. Find the sID of students who have earned a grade of 85 or more in some course, or who have passed a course taught by Atwood.
   Ensure that no sID occurs twice in the result.
   Output:

```
  sid
-------
   157
 98000
 99132
 99999
(4 rows)
```

   Solution

```
(SELECT sid FROM took WHERE grade >= 85)
UNION
(SELECT sid
FROM Took, Offering
WHERE Took.oid = Offering.oid AND instructor = 'Atwood' AND grade >= 50)
```

4. Find all terms when csc369 was not offered.
   Output:

```
term
-------
 20081
 20089
(2 rows)
```

   Solution

```
(SELECT term
FROM Offering)
EXCEPT
(SELECT term
FROM Offering
WHERE dept = 'csc' AND cNum = 369);
```

5. Make a table with two columns: oID and results.
   In the results column, report either “high” (if that offering had an average grade of 80 or higher), or “low” (if that offering had an average under 60). Offerings with an average in between will not be included.
   Hints:
   use a set operation.
   You can use the SELECT clause to put a literal value into a column.

For example:

```
SELECT 'high' as results ....
```

Output:

```
oid  | results
-----+---------
  38 | high
  14 | low
   8 | high
   7 | high
   3 | high
  28 | high
  13 | high
  39 | high
  15 | low
   1 | high
(10 rows)
```

Solution

```
(SELECT oID, 'high' AS results
FROM Took
GROUP BY oID
HAVING avg(grade) >=80)
UNION
(SELECT oID, 'low' as results
FROM Took
GROUP by oID
HAVING avg(grade) < 60);
```

Part 2:

1. Write a query to find the average grade, minimum grade, and maximum grade for each offering.

   Solution

```
SELECT oid, avg(grade), min(grade), max(grade)
FROM Took
GROUP BY oid;
```

Output:

```
 oid |         avg          | min | max
-----+----------------------+-----+-----
  31 | 78.0000000000000000 |  70 |  82
  34 | 60.6666666666666667 |  45 |  75
             . . . rows omitted
   8 | 92.0000000000000000 |  91 |  93
  11 | 79.0000000000000000 |  39 |  99
(23 rows)
```

2. Which of these queries is legal?

```
SELECT surname, sid
FROM Student, Took
WHERE Student.sid = Took.sid
GROUP BY sid;
```

```
SELECT surname, Student.sid
FROM Student, Took
WHERE Student.sid = Took.sid
GROUP BY campus;
```

```
SELECT instructor, max(grade),
count(Took.oid)
FROM Took, Offering
WHERE Took.oid = Offering.oid
GROUP BY instructor;
```

```
SELECT Course.dept, Course.cnum,
       count(oid), count(instructor)
FROM Course, Offering
WHERE Course.dept = Offering.dept and
      Course.cnum = Offering.cnum
GROUP BY Course.dept, Course.cnum
ORDER BY count(oid);
```

Solution

```
ERROR: column reference "sid"
is ambiguous
LINE 1: SELECT surname, sid
                        ^
```

```
ERROR: column reference "sid" is ambiguous

LINE 1: LINE 1: SELECT surname, Student.sid
                        ^
```

```
instructor | max | count
-----------+-----+-------
 Jepson    |  89 |     5
 Heap      |  82 |     1
           .  .  .
 Mendel    |  75 |     3
(17 rows)
```

```
 dept | cnum | count | count
------+------+-------+-------
 ENV  |  320 |     1 |     1
 CSC  |  369 |     1 |     1
      .  .  .
 CSC  |  343 |     5 |     5
(18 rows)
```

3. Find the sid and minimum grade of each student with an average over 80.
   Solution

```
SELECT SID, min(grade)
FROM Took
GROUP BY sID
HAVING AVG(grade) > 80;
```

   Output:

```
  sid  | min
-------+-----
 98000 |  54
 99999 |  52
(2 rows)
```

4. Find the sid, surname, and average grade of each student, but keep the data only for those students who have taken at least 10 courses.
   Solution

```
SELECT Student.sID, surname, avg(grade)
FROM Student, Took
WHERE Student.sID = Took.sID
GROUP BY Student.sID
HAVING count(grade) >= 10;
```

   Output:

```
  sid  |  surname   |         avg
-------+------------+--------------------
   157 | Lakemeyer  | 75.9333333333333333
 99999 | Ali        | 84.5833333333333333
 98000 | Fairgrieve | 83.2000000000000000
(3 rows)
```

5. For each student who has passed at least 10 courses, report their sid and average grade on the courses that they passed.
   Solution

```
SELECT sid, AVG(grade)
FROM took
WHERE grade >= 50
GROUP BY sid
HAVING count(*) >= 10;
```

   Output:

```
  sid  |         avg
-------+--------------------
 98000 | 83.2000000000000000
 99999 | 84.5833333333333333
   157 | 78.5714285714285714
(3 rows)
```

There is a lot going on here. Be sure you are clear on the difference between WHERE and HAVING, and which rows are left at the moment where the HAVING condition is checked for each group.

6. For each student who has passed at least 10 courses, report their sid and average grade on all of their courses.
   Solution:
   Here, because we don't want a filter applied (only passing grades count) when choosing which students to report on, but we don't want that filter applied when we compute their average grade. A single query, with a single WHERE clause, can't accomplish this. Views to the rescue!

```
CREATE VIEW Seniors AS
SELECT sid
FROM Took
WHERE grade >= 50
GROUP BY sid
HAVING count(*) >= 10;
SELECT Seniors.sid, AVG(grade)
FROM Seniors, Took
WHERE seniors.sid = Took.sid
GROUP BY Seniors.sid;
```

   Output:

```
  sid  |         avg
-------+---------------------
 98000 | 83.2000000000000000
 99999 | 84.5833333333333333
   157 | 75.9333333333333333
(3 rows)
```

7. Which of these queries is legal?

```
SELECT dept
FROM Took, Offering
WHERE Took.oID = Offering.oID
GROUP BY dept
HAVING avg(grade) > 75;
```

```
SELECT Took.oID, avg(grade)
FROM Took, Offering
WHERE Took.oID = Offering.oID
GROUP BY Took.oID
HAVING avg(grade) > 75;
```

```
SELECT Took.oID, dept, cNum,
       avg(grade)
FROM Took, Offering
WHERE Took.oID = Offering.oID
GROUP BY Took.oID
HAVING avg(grade) > 75;
```

```
SELECT oID, avg(grade)
FROM Took
GROUP BY sID
HAVING avg(grade) > 75;
```

Solution:

Here's the result of each:

```
dept
------
 HIS
 CSC
 EEB
 ANT
(4 rows)
```

```
oid |       avg
-----+--------------------
 31 | 78.0000000000000000
  3 | 82.0000000000000000
 28 | 91.0000000000000000
 13 | 95.6666666666666667
  9 | 78.0000000000000000
  7 | 83.0000000000000000
  1 | 87.2500000000000000
 38 | 92.0000000000000000
 39 | 97.0000000000000000
 11 | 79.0000000000000000
  8 | 92.0000000000000000
(11 rows)
```

```
ERROR: column "offering.dept"
must appear in the GROUP BY
clause or be used in an
aggregate function
LINE 1: SELECT Took.oID, dept,
cNum, avg(grade)
```

```
ERROR: column "took.oid" must
appear
in the GROUP BY clause or be
used in an
aggregate function
LINE 1: SELECT oID, avg(grade)
```