Introduction to Data Management

Lecture 15
(SQL: Yes, There’s More...)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

- HW and exams:
  - HW #4 info
    - Still ongoing, late deadline is tomorrow (6 PM)
  - Midterm Exam 1
    - Totaling is underway, exams coming back Friday!

- Office hours this week:
  - Today 4-5pm → tomorrow 5-6pm. (Sorry!!)

- Today’s lecture plan:
  - Still more about SQL...!
Find age of the youngest sailor with age $\geq 18$ for each rating with at least 2 sailors between 18 and 60.

$$\text{SELECT S.rating, MIN(S.age) AS minage}$$
$$\text{FROM Sailors S}$$
$$\text{WHERE S.age} \geq 18 \text{ AND S.age} \leq 60$$
$$\text{GROUP BY S.rating}$$
$$\text{HAVING COUNT(*)} \geq 2$$

Sailors instance:

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>29</td>
<td>brutus</td>
<td>1</td>
<td>33.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>32</td>
<td>andy</td>
<td>8</td>
<td>25.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>64</td>
<td>horatio</td>
<td>7</td>
<td>35.0</td>
</tr>
<tr>
<td>71</td>
<td>zorba</td>
<td>10</td>
<td>16.0</td>
</tr>
<tr>
<td>74</td>
<td>horatio</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>85</td>
<td>art</td>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>95</td>
<td>bob</td>
<td>3</td>
<td>63.5</td>
</tr>
<tr>
<td>96</td>
<td>frodo</td>
<td>3</td>
<td>25.5</td>
</tr>
</tbody>
</table>

Answer relation:

<table>
<thead>
<tr>
<th>rating</th>
<th>minage</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>7</td>
<td>35.0</td>
</tr>
<tr>
<td>8</td>
<td>25.5</td>
</tr>
</tbody>
</table>

Find age of the youngest sailor with age $\geq 18$ for each rating with at least 2 such sailors.

rating age
---
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5

rating minage
---
3 25.5
7 35.0
8 25.5
For each red boat, find the number of reservations for this boat

\[
\text{SELECT B.bid, COUNT(*) AS scount}
\text{FROM Sailors S, Boats B, Reserves R}
\text{WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'}
\text{GROUP BY B.bid}
\]

- We’re grouping over a join of three relations!
- What do we get if we remove \text{B.color='red'} from the WHERE clause and add a HAVING clause with this condition? (\textbf{Hint}: Trick question... 😃)
- What if we drop Sailors and the condition involving S.sid?
Find age of the youngest sailor with age > 18 for each rating with at least 2 sailors (of any age)

```
SELECT S.rating, MIN(S.age)
FROM Sailors S
WHERE S.age > 18
GROUP BY S.rating
HAVING 1 < (SELECT COUNT(*)
            FROM Sailors S2
            WHERE S.rating = S2.rating)
```

- Shows HAVING clause can also contain a subquery.
- Compare this with the query where we considered only ratings with 2 or more sailors over 18!
- What if HAVING clause were replaced by:
  - HAVING COUNT(*) > 1

Find those ratings for which the average age is the minimum age over all Sailors

- Aggregate operations can’t be nested! (WRONG...)

```
SELECT S.rating
FROM Sailors S
WHERE S.age = (SELECT MIN(AVG(S2.age))
               FROM Sailors S2)
```

- Correct solution (in SQL/92):

```
SELECT Temp.rating, Temp.avgage
FROM (SELECT S.rating, AVG(S.age) AS avgage
       FROM Sailors S
       GROUP BY S.rating) AS Temp
WHERE Temp.avgage = (SELECT MIN(age)
                      FROM Sailors)
```

Compute the average age for each rating...

Find the overall minimum age
Null Values

- Field values in a tuple are sometimes unknown (e.g., a rating has not been assigned) or inapplicable (e.g., no spouse’s name).
  - SQL provides special value null for such situations.

- The presence of null complicates many issues. E.g.:
  - Special operators needed to check if value is/is not null.
  - Is rating>8 true or false when rating is equal to null? What about AND, OR and NOT connectives?
  - We need a 3-valued logic (true, false and unknown).
  - Meaning of constructs must be defined carefully. (The WHERE clause eliminates rows that don’t evaluate to true.)
  - New operators (in particular, outer joins) possible/needed.

Nulls and SQL’s 3-Valued Logic

<table>
<thead>
<tr>
<th>AND</th>
<th>true</th>
<th>false</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>unknown</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>false</td>
<td>unknown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR</th>
<th>true</th>
<th>false</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>unknown</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOT</th>
<th>true</th>
<th>false</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: SQL arithmetic expressions involving null values will yield null values (Ex: EMP.sal + EMP.bonus)
Tune in next time ....