Lecture 34: Aggregation and Grouping

Announcements.

• Glookup grades are up.
• Use form (see Piazza) to report apparent errors, request regrades.
Aggregation, Again

- We briefly saw examples of aggregation in a previous lecture:

```sql
> select max(score) from grades;
20

> select avg(score) from grades;
12.0769230769231

> select avg(score) from grades ...
... where assign="hw1";
2.0
```

<table>
<thead>
<tr>
<th>name</th>
<th>assign</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Brown</td>
<td>hw1</td>
<td>2</td>
</tr>
<tr>
<td>Walt Green</td>
<td>hw1</td>
<td>3</td>
</tr>
<tr>
<td>Valerie Blue</td>
<td>hw1</td>
<td>1</td>
</tr>
<tr>
<td>Simon Red</td>
<td>hw2</td>
<td>3</td>
</tr>
<tr>
<td>John Brown</td>
<td>test1</td>
<td>20</td>
</tr>
<tr>
<td>Walt Green</td>
<td>test1</td>
<td>14</td>
</tr>
<tr>
<td>John Brown</td>
<td>test2</td>
<td>19</td>
</tr>
<tr>
<td>Valerie Blue</td>
<td>test1</td>
<td>14</td>
</tr>
<tr>
<td>Simon Red</td>
<td>test1</td>
<td>17</td>
</tr>
<tr>
<td>Walt Green</td>
<td>test2</td>
<td>12</td>
</tr>
<tr>
<td>Valerie Blue</td>
<td>test2</td>
<td>15</td>
</tr>
<tr>
<td>Sarah Tan</td>
<td>test2</td>
<td>19</td>
</tr>
<tr>
<td>Sarah Tan</td>
<td>test1</td>
<td>18</td>
</tr>
</tbody>
</table>
Aggregation

• Sometimes, we’d like a query that groups the data into subsets and aggregates each.

• A clumsy approach:

```sql
> select assign, avg(score) from grades where assign="hw1" union
... select assign, avg(score) from grades where assign="hw2" union
... select assign, avg(score) from grades where assign="test1" union
... select assign, avg(score) from grades where assign="test2";
```

• But it is generally cleaner to let SQL do the grouping for you:

```sql
> select assign, avg(score) from grades group by assign;
  hw1|2.0
  hw2|3.0
  test1|16.6
  test2|16.25
```

• First, groups rows with the same `assign` column value. Then runs the query on each group separately, unioning the results.
Selecting Groups

• Just as we often want to filter rows, we may also need to filter groups.

• Example: I want a summary of assignments that have at least two submissions.

• The where clause isn’t quite right, because it happens before grouping.

• So for groups, we use a new clause: having:

\[
\text{having count(*) } \geq 2;
\]

> select assign, avg(score)
from grades
... group by assign
    having count(*) >= 2;

hw1|3|2.0
test1|5|16.6
test2|4|16.25

<table>
<thead>
<tr>
<th>name</th>
<th>assign</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Brown</td>
<td>hw1</td>
<td>2</td>
</tr>
<tr>
<td>Walt Green</td>
<td>hw1</td>
<td>3</td>
</tr>
<tr>
<td>Valerie Blue</td>
<td>hw1</td>
<td>1</td>
</tr>
<tr>
<td>Simon Red</td>
<td>hw2</td>
<td>3</td>
</tr>
<tr>
<td>John Brown</td>
<td>test1</td>
<td>20</td>
</tr>
<tr>
<td>Walt Green</td>
<td>test1</td>
<td>14</td>
</tr>
<tr>
<td>John Brown</td>
<td>test2</td>
<td>19</td>
</tr>
<tr>
<td>Valerie Blue</td>
<td>test1</td>
<td>14</td>
</tr>
<tr>
<td>Simon Red</td>
<td>test1</td>
<td>17</td>
</tr>
<tr>
<td>Walt Green</td>
<td>test2</td>
<td>12</td>
</tr>
<tr>
<td>Valerie Blue</td>
<td>test2</td>
<td>15</td>
</tr>
<tr>
<td>Sarah Tan</td>
<td>test2</td>
<td>19</td>
</tr>
<tr>
<td>Sarah Tan</td>
<td>test1</td>
<td>18</td>
</tr>
</tbody>
</table>

grades
A Bit Fancier

- I'd like average scores for each *category* of assignment:

<table>
<thead>
<tr>
<th>assign</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw1</td>
<td>hw</td>
</tr>
<tr>
<td>hw2</td>
<td>hw</td>
</tr>
<tr>
<td>test1</td>
<td>test</td>
</tr>
<tr>
<td>test2</td>
<td>test</td>
</tr>
</tbody>
</table>
A Bit Fancier, Continued

- I’d like average scores for each category of assignment:

<table>
<thead>
<tr>
<th>assign</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw1</td>
<td>hw</td>
</tr>
<tr>
<td>hw2</td>
<td>hw</td>
</tr>
<tr>
<td>test1</td>
<td>test</td>
</tr>
<tr>
<td>test2</td>
<td>test</td>
</tr>
</tbody>
</table>

```sql
> select type, avg(score) from grades, categories
... where grades.assign = categories.assign
... group by type;
hw | 2.25
---
test | 16.444444444444444
```
Some Bells and Whistles

• We can sort the rows presented, and can filter out duplicates:

```sql
> select name from grades
... order by name;
John Brown
John Brown
John Brown
Sarah Tan
Sarah Tan
Simon Red
Simon Red
Valerie Blue
Valerie Blue
Valerie Blue
Walt Green
Walt Green
Walt Green
```

```sql
> select distinct name from grades
... order by name;
John Brown
Sarah Tan
Simon Red
Valerie Blue
Walt Green
```
**One More Bell**

- Finally, can limit the number of responses:

```sql
> select name from grades order by name limit 8;
John Brown
John Brown
John Brown
Sarah Tan
Sarah Tan
Simon Red
Simon Red
Valerie Blue
```
Syntax of Select

Extracted from https://www.sqlite.org/lang.html

Last modified: Mon Apr 18 04:34:47 2016