SQL Aggregate Queries

Aggregate Operators

Significant extension of relational algebra

- `COUNT(*)`
- `COUNT(DISTINCT A)`
- `SUM(DISTINCT A)`
- `AVG(DISTINCT A)`
- `MAX(A)`
- `MIN(A)`

A is a single column

Result is single value obtained by applying aggregate over all qualifying tuples

```
SELECT COUNT(*)
FROM Sailors S
```

Aggregate Queries Examples

```
SELECT AVG(S.age)
FROM Sailors S
WHERE S.rating=10

SELECT COUNT(DISTINCT S.rating)
FROM Sailors S
WHERE S.sname='Bob'

SELECT S.sname, MAX(S.age)
FROM Sailors S
WHERE S.rating = (SELECT MAX(S2.rating)
                 FROM Sailors S2)
```

Aggregate + nested!

Common Mistake with Aggregates

Illegal Query!

- Can't have both aggregates and non-aggregates in SELECT
  - Exception: GROUP BY (later in this class)
  - Reason: it is not guaranteed that there is only one tuple with the MAX value

Grouping Results

- So far, aggregates applied to all qualifying tuples
- We may want to apply them to each of several groups

“Find the age of the youngest sailor for each rating level”

- In general, we don’t know how many rating levels exist, and what the rating values for these levels are!
- Suppose we know that rating values go from 1 to 10

```
SELECT MIN(S.age)
FROM Sailors S
WHERE S.rating = 1

SELECT MIN(S.age)
FROM Sailors S
WHERE S.rating = 2

SELECT MIN(S.age)
FROM Sailors S
WHERE S.rating = 10
```

How to achieve this?

Queries With GROUP BY and HAVING

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
```

- The target-list contains:
  - (i) attribute names list
  - (ii) terms with aggregate operations (e.g., `MIN(S.age)`)  

- The attribute list (i) must be a subset of grouping-list
- A group is a set of tuples that have the same value for all attributes in grouping-list
- Each answer tuple corresponds to a group, so these attributes must have a single value per group.
### Conceptual Evaluation

1. Compute cross-product of relation-list
2. Discard tuples that fail qualification, 'unnecessary' fields are deleted
3. Remaining tuples are partitioned into groups by the value of attributes in grouping-list
4. Discard groups that fail group-qualification
   - Expressions in group-qualification must have a single value per group!
   - An attribute in group-qualification that is not an argument of an aggregate operation must appear in grouping-list (unless EVERY or ANY used)
5. Generate single answer tuple per qualifying group

### GROUPBY Query Example

"Find age of the youngest sailor with age at least 18, for each rating with at least 2 such sailors"

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

<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>
```

### GROUPBY Conceptual Evaluation Example

"Find age of the youngest sailor with age at least 18, for each rating with at least 2 such sailors"

<table>
<thead>
<tr>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>1</td>
<td>33.0</td>
</tr>
<tr>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>8</td>
<td>25.5</td>
</tr>
<tr>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>10</td>
<td>16.0</td>
</tr>
<tr>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>3</td>
<td>63.5</td>
</tr>
<tr>
<td>3</td>
<td>25.5</td>
</tr>
</tbody>
</table>

### More Group Qualification Functions

- So far, we have seen group qualification based on a property of the group
- E.g., aggregate function computed for entire group
- But recent SQL standard version allow group qualification based on a property of individual records
  - EVERY(condition): TRUE if condition holds for every group tuple
  - ANY(condition): TRUE if condition holds for some group tuple

### Pay attention to order of steps!

- **HAVING** executes AFTER **WHERE**

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

"Find age of the youngest sailor with age at least 18, for each rating with at least 2 sailors (of any age)"

Wrong!!!
Find age of the youngest sailor with age >= 18, for each rating with at least 2 sailors (of any age)

<table>
<thead>
<tr>
<th>rating</th>
<th>age</th>
<th>rating</th>
<th>age</th>
<th>rating</th>
<th>age</th>
<th>rating</th>
<th>minage</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>45.0</td>
<td>7</td>
<td>45.0</td>
<td>7</td>
<td>45.0</td>
<td>7</td>
<td>35.0</td>
</tr>
<tr>
<td>1</td>
<td>33.0</td>
<td>1</td>
<td>33.0</td>
<td>1</td>
<td>33.0</td>
<td>1</td>
<td>25.5</td>
</tr>
<tr>
<td>8</td>
<td>55.5</td>
<td>8</td>
<td>55.5</td>
<td>8</td>
<td>55.5</td>
<td>8</td>
<td>25.5</td>
</tr>
<tr>
<td>9</td>
<td>35.0</td>
<td>9</td>
<td>35.0</td>
<td>10</td>
<td>35.0</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

Pay attention to order of steps!

"Find age of the youngest sailor with age >= 18, for each rating with at least 2 sailors (of any age)"

```sql
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)
```

Pay attention to order of steps!

**Summary of cases** – INFORMAL!

- Can group validation condition be evaluated on “intermediate” relation alone?
  - If NO, then we need subquery in HAVING
  - If YES, then we do not need subquery and we have two further cases:
    - Group validation condition DOES NOT depend on individual tuples in group, only aggregates and group-by attributes appear in the HAVING clause
    - Group validation DOES depend on individual tuples in group, in which case non-group-by attributes may appear with ANY or EVERY operator

- Note: this is just a guideline for most cases, it is actually possible to have a mix of the above!!!