Today’s Agenda

• Quiz

• More SQL extensions
  – Group by
  – Having
  – Order by
  – Subqueries (WHERE clause)
Sample Database (from last week)

- We’ll use this database in our examples

\[
\text{Customer}(\text{Number, Name, Address, CRating, CAmount, CBalance, Salesperson})
\]

\[
\text{Salesperson}(\text{Number, Name, Address, Office})
\]

Foreign key: Customer.Salesperson \rightarrow Salesperson.Number

SQL GROUP BY

- Any SQL query can have the answer “grouped”
  - In which there is one output row for each group

\[
\text{SELECT Salesperson, COUNT(\ast) FROM Customer;}
\]

versus

\[
\text{SELECT Salesperson, COUNT(\ast) FROM Customer GROUP BY Salesperson;}
\]
SQL GROUP BY

Customer

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Address</th>
<th>CRating</th>
<th>CAmount</th>
<th>CBalance</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>smith</td>
<td>xxx</td>
<td>5</td>
<td>1000</td>
<td>1000</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>jones</td>
<td>yyy</td>
<td>7</td>
<td>5000</td>
<td>4000</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>wei</td>
<td>zzz</td>
<td>10</td>
<td>10000</td>
<td>10000</td>
<td>NULL</td>
</tr>
</tbody>
</table>

SELECT Salesperson, COUNT(*)
FROM Customer
GROUP BY Salesperson;

SQL HAVING

Customer

SELECT Salesperson, COUNT(*)
FROM Customer
GROUP BY Salesperson
HAVING COUNT(*) > 1;

- The HAVING clause specifies a predicate evaluated against each group
- A group is in the answer if it satisfies the HAVING condition
### SQL GROUP BY Example

**Customer**

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>...</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>mary</td>
<td>...</td>
<td>5</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>...</td>
<td>8</td>
</tr>
<tr>
<td>103</td>
<td>dave</td>
<td>...</td>
<td>NULL</td>
</tr>
<tr>
<td>106</td>
<td>sam</td>
<td>...</td>
<td>5</td>
</tr>
<tr>
<td>107</td>
<td>oliver</td>
<td>...</td>
<td>5</td>
</tr>
<tr>
<td>109</td>
<td>susan</td>
<td>...</td>
<td>2</td>
</tr>
<tr>
<td>110</td>
<td>luis</td>
<td>...</td>
<td>8</td>
</tr>
</tbody>
</table>

SELECT Salesperson, COUNT(*)
FROM Customer
GROUP BY Salesperson;

---

** Intermediate Result:**

4 Groups

- 103 dave ... NULL
- 101 mary ... 5
- 106 sam ... 5
- 107 oliver ... 5
- 102 John ... 8
- 110 luis ... 8
- 109 Susan ... 2
**SQL GROUP BY Example**

Here we evaluate “SELECT Salesperson, Count(*)" for each group

```
SELECT Salesperson, COUNT(*)
FROM Customer
GROUP BY Salesperson;
```

### Intermediate Result: 4 Groups

<table>
<thead>
<tr>
<th>Salesperson</th>
<th>Count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

**Query Answer:** 1 row per group!

### Query Answer:

1 row per group!

**CPSC 421, 2009 9**

---

**SQL GROUP BY Example**

Here we check if COUNT(*) > 1 holds for each group, and then evaluate “SELECT Salesperson, Count(*)" for each group

```
SELECT Salesperson, COUNT(*)
FROM Customer
GROUP BY Salesperson
HAVING COUNT(*) > 1;
```

### Intermediate Result: 4 Groups

<table>
<thead>
<tr>
<th>Salesperson</th>
<th>Count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

**Query Answer:** 1 row per group!

### Query Answer:

1 row per group!

**CPSC 421, 2009 10**
SQL GROUP BY, HAVING

SELECT Salesperson, COUNT(*)
FROM Customer
GROUP BY Salesperson;

- The only attribute that can appear in a “grouped” query are the grouping attributes or aggregate operators (that are applied to the group)

SQL GROUP BY

- The following query is not legal (Why?)

SELECT Name
FROM Customer
GROUP BY Salesperson;

- … because there can be many names for each group!
Exercise …

1. Form groups of 2
2. Use this Baseball schema:
   Team(Name, Games, Wins, Losses, Conference)
   Player(Name, Hits, AtBats, HomeRuns, Team)
   Player.Team -> Team.Name
3. Write SQL queries for the following:
   a) Find the average number of wins and losses across teams
   b) Find the average number of wins and losses per conference
   c) Find the average number of home runs per conference
   d) Find the batting average for each player, where batting average is the number of hits divided by at bats
4. Write an interesting query that uses both GROUP BY and HAVING

SQL ORDER BY

• Used to sort the result of a query

SELECT Number, Name, Salesperson
FROM Customer
ORDER BY Name;

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>mary</td>
<td>5</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>8</td>
</tr>
<tr>
<td>103</td>
<td>dave</td>
<td>NULL</td>
</tr>
<tr>
<td>106</td>
<td>sam</td>
<td>5</td>
</tr>
<tr>
<td>107</td>
<td>oliver</td>
<td>5</td>
</tr>
<tr>
<td>109</td>
<td>susan</td>
<td>2</td>
</tr>
<tr>
<td>110</td>
<td>luis</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>mary</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>dave</td>
<td>NULL</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>8</td>
</tr>
<tr>
<td>110</td>
<td>luis</td>
<td>8</td>
</tr>
<tr>
<td>101</td>
<td>mary</td>
<td>5</td>
</tr>
<tr>
<td>111</td>
<td>mary</td>
<td>4</td>
</tr>
<tr>
<td>107</td>
<td>oliver</td>
<td>5</td>
</tr>
<tr>
<td>106</td>
<td>sam</td>
<td>5</td>
</tr>
<tr>
<td>109</td>
<td>susan</td>
<td>2</td>
</tr>
</tbody>
</table>
SQL ORDER BY

• Used to sort the result of a query

SELECT Number, Name, Salesperson
FROM Customer
ORDER BY Name DESC;

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>mary</td>
<td>5</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>8</td>
</tr>
<tr>
<td>103</td>
<td>dave</td>
<td>NULL</td>
</tr>
<tr>
<td>106</td>
<td>sam</td>
<td>5</td>
</tr>
<tr>
<td>107</td>
<td>oliver</td>
<td>5</td>
</tr>
<tr>
<td>109</td>
<td>susan</td>
<td>2</td>
</tr>
<tr>
<td>110</td>
<td>luis</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>mary</td>
<td>4</td>
</tr>
</tbody>
</table>

SQL ORDER BY

• Used to sort the result of a query

SELECT Number, Name, Salesperson
FROM Customer
ORDER BY Name, Salesperson;

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>mary</td>
<td>5</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>8</td>
</tr>
<tr>
<td>103</td>
<td>dave</td>
<td>NULL</td>
</tr>
<tr>
<td>106</td>
<td>sam</td>
<td>5</td>
</tr>
<tr>
<td>107</td>
<td>oliver</td>
<td>5</td>
</tr>
<tr>
<td>111</td>
<td>mary</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>dave</td>
<td>NULL</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>8</td>
</tr>
<tr>
<td>110</td>
<td>luis</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>mary</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>dave</td>
<td>NULL</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>8</td>
</tr>
<tr>
<td>110</td>
<td>luis</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>mary</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>dave</td>
<td>NULL</td>
</tr>
<tr>
<td>102</td>
<td>john</td>
<td>8</td>
</tr>
<tr>
<td>110</td>
<td>luis</td>
<td>8</td>
</tr>
<tr>
<td>111</td>
<td>mary</td>
<td>4</td>
</tr>
</tbody>
</table>
SQL Subqueries

Subqueries can be used in the where clause (in addition to the from clause) …

SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.CRaYng = (SELECT MAX(C2.CRaYng)
                        FROM Customer C2);

• The subquery is the “Inner Query” and the rest is the “Outer Query”

SQL Subqueries

• What does this query return?

SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.CRaYng = (SELECT MAX(C2.CRaYng)
                        FROM Customer C2);

• What does the inner query return?
  – A single value (the max credit rating)

• What does the outer query return?
  – The name and number of the customers with the highest credit ratings
SQL Subqueries

- How would you evaluate this query?

```sql
SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.CRatings = (SELECT MAX(C2.CRatings)
                     FROM Customer C2);
```

- Start with the FROM clause in the outer query
- Take a row from the Customer table
- Check if the row satisfies the WHERE clause
- … which involves evaluating the inner query

---

```
<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Address</th>
<th>CRating</th>
<th>CAmount</th>
<th>CBalance</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Smith</td>
<td>123 X St</td>
<td>5</td>
<td>10000</td>
<td>9000</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>222 Y St</td>
<td>7</td>
<td>8000</td>
<td>3750</td>
<td>65</td>
</tr>
<tr>
<td>3</td>
<td>Wei</td>
<td>111 Z St</td>
<td>10</td>
<td>11000</td>
<td>9000</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```

SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.CRatings = (SELECT MAX(C2.CRatings)
                     FROM Customer C2);

- Here, we can evaluate the inner query first (answer: "10")
- For each row, is CRating equal to the inner query answer?
**SQL Subqueries**

Subqueries can be used in the where clause (in addition to the from clause) …

SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.CRating = (SELECT MAX(C2.Crating)
                      FROM Customer C2);

• The **comparator** can be any of the six standard ones:
  - `=`, `>`, `>=`, `<=`, `<`, `<>` (not equal)

• For these, the inner query must return a single value!

---

**SQL Subqueries**

Subqueries can be used in the where clause (in addition to the from clause) …

SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.CRating = (SELECT MAX(C2.Crating)
                      FROM Customer C2);

• If the inner query does not mention any attributes from the outer query (C1 not mentioned in the inner query)
  - Then you only need to evaluate the inner query once
  - The inner (sub) query is **NOT correlated**
SQL Subqueries: SOME/ALL comparison

SELECT S.Number, S.Name
FROM Salesperson S
WHERE S.Name = SOME (SELECT C.Name
                        FROM Customer C);

• For SOME, the expression must be true for at least one row in the subquery answer
  – “ANY” is equivalent to SOME
  – What does this query return?

• For ALL, the expression must be true for all rows in the subquery answer

SQL Subqueries: SOME/ALL comparison

SELECT S.Name
FROM Salesperson S
WHERE S.Number = SOME (SELECT C.Salesperson
                          FROM Customer C
                          WHERE C.CRating = 3);

• What does this query return?
  – The name of each salespeople that has a customer with a credit rating of 3
**SQL Subqueries: SOME/ALL comparison**

```sql
SELECT S.Name
FROM Salesperson S
WHERE S.Number = ALL (SELECT C.Salesperson
                           FROM Customer C
                           WHERE C.CRating = 3);
```

- **What does this query return?**
  - The name of the salesperson that has all the customers with a rating of 3 (if such a salesperson exists)

**SQL Subqueries: IN / NOT IN comparison**

```sql
SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.Name IN (SELECT Name
                    FROM Salesperson);
```

- **With IN, the attribute matches at least one value returned from the subquery**
  - Same as “= SOME”
SQL Subqueries: IN / NOT IN comparison

SELECT C1.Number, C1.Name
FROM Customer C1
WHERE C1.Name NOT IN (SELECT Name
                         FROM Salesperson);

• With NOT IN, the attribute matches none of the values returned from the subquery
  – Same as “<> ALL”

SQL Subqueries: IN / NOT IN comparison

SELECT S.Number, S.Name
FROM Salesperson S
WHERE S.Number IN (SELECT C.Salesperson
                     FROM Customer C);

SELECT DISTINCT S.Number, S.Name
FROM Salesperson S, Customer C
WHERE S.Number = S.Salesperson;

• Are these equivalent?
• Do we need to use DISTINCT for these to be equivalent?
• Is the subquery correlated?
**SQL Subqueries: IN / NOT IN comparison**

SELECT S.Number, S.Name
FROM Salesperson S
WHERE S.Number IN (SELECT C.Salesperson
                      FROM Customer C
                      WHERE C.Name = S.Name);

• Because the subquery mentions an attribute from a table in the outer query
  – The subquery must be (re-)evaluated for each row in the outer query (each time the WHERE clause is evaluated)
  – Correlated subqueries can be very expensive!

**SQL Subqueries: EXISTS / NOT EXISTS**

SELECT C.Name
FROM Customer C
WHERE EXISTS (SELECT *
               FROM Salesperson S
               WHERE S.Number = C.Salesperson AND
                     S.Name = C.Name);

• If the answer to the subquery is not empty … then the EXISTS predicate returns TRUE
  – Is this subquery correlated?
  – What does this query return?
SQL Subqueries: EXISTS / NOT EXISTS

SELECT C.Name
FROM Customer C
WHERE EXISTS (SELECT *
               FROM Salesperson S
               WHERE S.Number = C.Salesperson AND
                     S.Name = C.Name);

• Four predicates can be applied to a subquery
  EXISTS … is the subquery answer non-empty?
  NOT EXISTS … is the subquery answer empty?
  UNIQUE … does the subquery return just one row?
  NOT UNIQUE … does the subquery return multiple rows?

For Thursday

• Reading
  – Make sure you are caught up in the reading
  – We’ll finish up SQL queries on Thursday and start on
    conceptual modeling

• Be sure to know:
  – Group by, having, order by, subqueries

• Homework
  – Homework 2 due on Thursday
  – Should form groups this week, start on project proposal