Today

- Subqueries (cont)

Assignments

- HW6 out
- Quiz next Tuesday
Example Tables

Customer(number, name, address, c_rating, c_amount, c_balance, salesperson)

Salesperson(number, name, address, office)

<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>

- where Customer.salesperson is a FK to Salesperson.number
A correlated subquery

```sql
SELECT s.number, s.name
FROM salesperson s
WHERE s.number IN (SELECT c.salesperson
                      FROM customer c
                      WHERE c.name = s.name);
```

Q: Why is this a correlated subquery?
   – the subquery mentions an attribute from the outer query

Q: What does this query return?
   – salespeople that have a customer with the same name

• The subquery **must be evaluated for each row** in the outer query
• Because of this, correlated subqueries can be very expensive!

Q: Can this query be rewritten without a subquery?
   – Yes!
EXISTS/NOT EXISTS checks

SELECT c.name
FROM customer c
WHERE EXISTS (SELECT *
    FROM salesperson s
    WHERE s.number = c.salesperson AND
    s.name = c.name);

- If subquery is not empty, then EXISTS returns true
- Alternatively, NOT EXISTS returns true if subquery is empty

Q: What does this return? Is the subquery correlated?
   - Customers who have the same name as their salesperson
   - Yes!
More notes on subqueries

Subqueries can be used many places within an SQL query

Including within FROM clause ...

```
SELECT c1.number, c1.name 
FROM customer c1, 
  (SELECT MAX(rating) as max_rating FROM customer) as c2 
WHERE c1.c_rating = c2.max_rating;
```

Q: What does this query do?

- Finds customers with the largest ratings

And a HAVING clause ...

```
SELECT s.number, s.name, AVG(c.c_balance) 
FROM salesperson s JOIN customer c ON 
  s.number = c.salesperson 
WHERE c.c_rating > 5 
GROUP BY s.number, s.name 
HAVING AVG(c.c_balance) >= ALL (SELECT AVG(c_balance) 
  FROM customer 
  WHERE c_rating > 5 
  GROUP BY salesperson);
```

Q: What does this query do?

- Finds the salespeople whose average customer account balance for those customers with a rating over 5 is greater than the average balance of all customers with a credit rating over 5.
More Examples:

Q: Find the number and name of salespeople with the largest number of customers.

```
SELECT s.number, s.name, COUNT(*)
FROM salesperson s JOIN customer c ON s.number = c.salesperson
GROUP BY s.number, s.name
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                         FROM customer
                         GROUP BY salesperson);
```

Q: Find the number and name of customers with a higher than average rating and a lower than average balance.

```
SELECT number, name
FROM customer
WHERE c_rating > (SELECT AVG(c_rating) FROM customer) AND c_balance < (SELECT AVG(c_balance) FROM customer);
```
A Note on Using Subqueries (Wrap Up)

When possible, rewrite (unnest) your subqueries!

- Don’t get carried away with unnecessary subqueries
- Subqueries (e.g., in MySQL) may not be optimized
- Rewriting them into flat select-from-where will be (optimized)

Note that each query can often be written multiple ways