Today ...

- Quiz 7
- Subqueries
- Outer Joins

Homework

- HW 8 out
Consider the following schema

\[
\text{customer}(\text{cust\_num}, \text{name}, \text{address}, \text{c\_rating}, \text{c\_amount}, \text{c\_balance}, \text{sp\_num})
\]
\[
\text{salesperson}(\text{sp\_num}, \text{name}, \text{address}, \text{office})
\]

*Foreign Key:* \(\text{customer.sp\_num} \rightarrow \text{salesperson.sp\_num}\)

**EXISTS/NOT EXISTS checks**

\[
\begin{align*}
\text{SELECT} & \quad \text{c\_name} \\
\text{FROM} & \quad \text{customer} \ c \\
\text{WHERE} & \quad \text{EXISTS} \ (\text{SELECT} \ * \\
& \quad \text{FROM} \ \text{salesperson} \ s \\
& \quad \text{WHERE} \ s\_sp\_num = c\_sp\_num \ \text{AND} \\
& \quad \ \text{s\_name} = c\_name); \\
\end{align*}
\]

- 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 ...

```sql
SELECT c1.cust_num, 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 ...

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

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 salespeople with the most customers

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

Q: Find customers with a higher than avg rating and lower than avg balance

```sql
SELECT c1.cust_num, c1.name
FROM customer c1
WHERE c1.c_rating > (SELECT AVG(c2.c_rating) FROM customer c2) AND
      c1.c_balance < (SELECT AVG(c2.c_balance) FROM customer c2);
```
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 unnested queries may help optimization

Note that each query can often be written multiple ways
**Outer Joins**

The basic join is called an **“inner join”**

- An inner join is the default ... the plain **JOIN** keyword
- We can also write an inner join this way:
  ```sql
  SELECT *
  FROM customer INNER JOIN salesperson USING (sp_num);
  ```
- The answer includes all "matches"
- The answer **excludes**:
  - Customer rows that do **not** have a Salesperson
  - Salesperson rows that are **not** assigned any Customers

An **“outer join”** includes the **“non-matches”**

- A **LEFT OUTER JOIN** includes all matches plus all
  - Customers that do **not** have a Salesperson
- A **RIGHT OUTER JOIN** includes all matches plus all
  - Salespeople that are **not** assigned to any customers
- A **FULL OUTER JOIN** includes all of these

The "missing” attribute values in result are assigned **NULL**
INNER vs. OUTER JOIN

- **INNER JOIN** ON `c.salesperson = s.number` gives:
  
  1, Alice, 55, 55, Dave
  2, Bob, 65, 65, Eddy

- **LEFT OUTER JOIN** ON `c.salesperson = s.number` gives:
  
  1, Alice, 55, 55, Dave
  2, Bob, 65, 65, Eddy
  3, Chuck, NULL, NULL, NULL

- **RIGHT OUTER JOIN** ON `c.salesperson = s.number` gives:
  
  1, Alice, 55, 55, Dave
  2, Bob, 65, 65, Eddy
  NULL, NULL, NULL, 75, Fionna

- **FULL OUTER JOIN** ON `c.salesperson = s.number` gives:
  
  1, Alice, 55, 55, Dave
  2, Bob, 65, 65, Eddy
  3, Chuck, NULL, NULL, NULL
  NULL, NULL, NULL, 75, Fionna