Today …
• subqueries

Homework
• HW 7 due
• HW 8 out
• Quiz on Tuesday (group by and having)
Consider the following schema

```
customer(cust_num, name, address, c_rating, c_amount, c_balance, sp_num)
salesperson(sp_num, name, address, office)
```

*Foreign Key: customer.sp_num → salesperson.sp_num*

Exercises:

Q: Write an SQL query to find the average credit rating of each salesperson's customers for salespeople whose average customer credit balance is greater than half of their average credit amount.

```
SELECT sp_num, AVG(c_rating)
FROM customer
GROUP BY sp_num
HAVING AVG(c_balance) > (AVG(c_amount)/2)
```

Q: Write an SQL query to find the average credit rating of customers associated with each office for offices with an average customer credit amount over $10,000.

```
SELECT s.office, AVG(c.c_rating)
FROM customer c JOIN salesperson s USING (sp_num)
GROUP BY s.office
HAVING AVG(c.c_amount) > 10000
```
Subqueries

A subquery is a “nested” query

- Primarily used within WHERE and FROM clauses
- Can also be used in SELECT and HAVING clauses
- We’ll primarily look at the WHERE case

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

- the subquery is the inner query
- the rest is the outer query

Q: What does the inner query return?
   - a single value (the max credit rating)

Q: What does the outer query return?
   - name and number of customers with the highest credit ratings
How could we evaluate this query?

1. start with the FROM clause in the outer query
2. take a row from the customer table
3. check if the row satisfies the WHERE clause
   - which involves evaluating the inner query
4. if so, output the number and name
5. etc.

- In this case the inner query always returns the same result
  - and so only needs to be evaluated once!
- The inner query is not “correlated”
  - it does not use any attributes from the outer query (c1)
  - whereas correlated subqueries use attributes from outer query
**WHERE subquery comparators**

**Single-valued comparisons**

```sql
SELECT c1.cust_num, c1.name
FROM customer c1
WHERE c1.c_rating = (SELECT MAX(c2.rating)
                      FROM customer c2);
```

- The comparator can be any of the six standard ones:
  - <, <=, =, !=, >=, >
- For these comparators, the inner query must return a single value!

**ANY/ALL comparisons**

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

For **ANY**

- subquery can return more than one answer
- the expression (here =) must be true for at least one subquery answer
- “**SOME**” is equivalent to ANY

Q: What does this query return? Can it be written without a subquery?

- Salespersons that are also customers
- Yes!
- Try to rewrite this query ...

For **ALL**, expression must be true for every subquery answer
More examples

```
SELECT s.name
FROM salesperson s
WHERE s.sp_num = ANY (SELECT c.sp_num
                        FROM customer c
                        WHERE c.c_rating = 3);
```

Q: What does this query return?

- Salespeople that have at least one customer with a credit rating of 3

Q: Can this query be rewritten without subqueries? If so, rewrite it.

```
SELECT s.name
FROM salesperson s
WHERE s.sp_num = ALL (SELECT c.sp_num
                        FROM customer c
                        WHERE c.c_rating = 3);
```

Q: What does this query return?

- Salespeople that have all the customers with a credit rating of 3

Q: Can this be rewritten w/out subqueries (given SQL we've seen)? Try it.
IN/NOT IN comparisons

```
SELECT c.cust_num, c.name
FROM customer c
WHERE c.name IN (SELECT s.name FROM salesperson s);
```

- For IN, the attribute matches **at least one** subquery value
  - Same as “= ANY”

```
SELECT c.cust_num, c.name
FROM customer c
WHERE c.name NOT IN (SELECT s.name FROM salesperson s);
```

- For NOT IN, the attribute matches **none of** the subquery values
  - Same as “!= ALL”
EXISTS/NOT EXISTS checks

SELECT c.name
FROM customer c
WHERE EXISTS (SELECT *
                FROM salesperson s
                WHERE s.sp_num = c.sp_num 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!