Lecture 21:
• Having
• Subqueries

Announcements:
• Project Step 3 due
• HW-6 out (due Tues, Nov 22)
• PS-4 out (due Tues, Nov 29)
• Exam 2 Thurs

Running Example

Schema:

Customer(c_num, name, addr, c_rating, c_amount, c_bal, sp_num)
Salesperson(sp_num, name, address, office)

with FK: customer.sp_num → salesperson.sp_num

Example Customer instance

<table>
<thead>
<tr>
<th>c_num</th>
<th>name</th>
<th>address</th>
<th>c_rating</th>
<th>c_amount</th>
<th>c_balance</th>
<th>sp_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>xxx</td>
<td>5</td>
<td>1000</td>
<td>1000</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>yyy</td>
<td>7</td>
<td>5000</td>
<td>4000</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>Chuck</td>
<td>zzz</td>
<td>10</td>
<td>10000</td>
<td>1000</td>
<td>102</td>
</tr>
</tbody>
</table>
Group By

When using GROUP BY, the SELECT clause can only contain:
- the grouping attributes (attributes in GROUP BY clause); or
- aggregate operators (which are applied to the group)

This query is not legal SQL ... but legal in MySQL

```
SELECT name FROM customer GROUP BY sp_num;
```

**Check In:** Why wouldn’t name be allowed in the SELECT here?
- there could potentially be many name values in each group

**Check In:** What if we group by a primary key though?
- then there could only one value for the other attributes
- e.g., SELECT name FROM customer GROUP BY c_num
- *note:* applies generally to functional dependencies (group by lhs)

Having Clause

**The HAVING clause filters groups**

**Check In:** Can you guess what this query returns?

```
SELECT sp_num, COUNT(*)
FROM customer
GROUP BY sp_num
HAVING COUNT(*) > 1;
```

<table>
<thead>
<tr>
<th>sp_num</th>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>2</td>
</tr>
</tbody>
</table>

- The HAVING clause is evaluated **against each group**
- A group is in the answer if it satisfies the HAVING conditions
- HAVING should (generally) be accompanied by a GROUP BY
Having Clause

A **HAVING clause can contain comparisons:**

- to attributes in the GROUP BY clause; or
- to expressions (e.g., aggregates) resulting in **one** value for the group

(*) note: same rules apply with respect to keys / FDs

A **HAVING clause without a GROUP BY**

- Treats result of FROM and WHERE as a single group

**Check In**: How does this differ from a **WHERE** clause?

- WHERE applies to each row
- HAVING applies to each group

Group By with Having Example

**Check In**: Find the minimum credit rating and average credit balance of each salesperson’s customers who have a credit amount over $5,000 and who work in the “Spokane” office. Only include salespeople whose corresponding customers have an average credit balance over $1,000.

```
SELECT sp_num, MIN(c_rating), AVG(c_balance)
FROM salesperson JOIN customer USING (sp_num)
WHERE c_amount > 5000 AND office = 'Spokane'
GROUP BY sp_num
HAVING AVG(c_balance) > 1000;
```
Group By with Having Example

Check In: Find the average credit rating of each salesperson’s customers for salespeople whose average customer credit balance is less than half of their average credit amount.

\[
\text{SELECT sp_num, AVG(c\_rating) FROM customer GROUP BY sp_num HAVING AVG(c\_balance) < (AVG(c\_amount)/2)}
\]

Check In: Find the average credit rating of customers associated with each office for offices with an average customer credit amount over $5,000.

\[
\text{SELECT office, AVG(c\_rating) FROM customer JOIN salesperson USING (sp\_num) GROUP BY office HAVING AVG(c\_amount) > 5000}
\]

Subqueries

A subquery is a “nested” query

• Primarily used within WHERE and FROM clauses
• Can also be used in SELECT and HAVING clauses

\[
\text{SELECT c1.c\_num, c1.name FROM customer c1 WHERE c1.c\_rating = (SELECT MAX(c2.c\_rating) FROM customer c2)}
\]

• the subquery is the inner query
• the rest is the outer query

Check In: What does the inner query return? ... single value (highest rating)

Check In: What does the outer query return? ... customers with highest rating
Subqueries

The subquery in the example is evaluated as follows:

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 (must evaluate inner query)
4. if so, output the number and name
5. grab another row, check if satisfied (evaluate inner query again), etc.

In this example 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

Check In: How does the following query differ?

```sql
SELECT c1.cust_num, c2.name
FROM customer c1 JOIN (SELECT MAX(c_rating) max_rating FROM customer) c2
ON (c1.c_rating = c2.max_rating)
```
- same query, but inner query evaluated only once!
- note: must name subquery and aggregate result in FROM clause