Introduction to Database Systems
CSE 414

Lecture 6: SQL Subqueries
Announcements

• HW2 and WQ2 released
  – Both due next Tuesday

• Please fill in the Azure questionnaire by tonight!
  – See HW2 writeup for details
Simple Aggregations

Five basic aggregate operations in SQL

```sql
select count(*) from Purchase
select sum(quantity) from Purchase
select avg(price) from Purchase
select max(quantity) from Purchase
select min(quantity) from Purchase
```

Except count, all aggregations apply to a single attribute
Everything in SELECT must be either a GROUP-BY attribute, or an aggregate.

---

**Need to be Careful...**

```
SELECT product, max(quantity)
FROM Purchase
GROUP BY product
```

```
SELECT product, quantity
FROM Purchase
GROUP BY product
```

-- what does this mean?

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

```
<table>
<thead>
<tr>
<th>Product</th>
<th>Max(quantity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>50</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>??</td>
</tr>
</tbody>
</table>
```

CSE 414 - Spring 2018
Semantics of SQL With Group-By

Evaluation steps:
1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes $a_1, \ldots, a_k$
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result
Exercise

Compute the total income per month
Show only months with less than 10 items sold
Order by quantity sold and display as “TotalSold”
Exercise

Compute the total income per month
Show only months with less than 10 items sold
Order by quantity sold and display as “TotalSold”

```
FROM Purchase
```
Exercise

Compute the total income per month
Show only months with less than 10 items sold
Order by quantity sold and display as “TotalSold”

```
FROM Purchase
GROUP BY month
```
Exercise

Compute the total income per month
Show only months with less than 10 items sold
Order by quantity sold and display as “TotalSold”

```
FROM Purchase
GROUP BY month
HAVING sum(quantity) < 10
```
Exercise

Compute the total income per month
Show only months with less than 10 items sold
Order by quantity sold and display as “TotalSold”

```
SELECT month, sum(price*quantity), sum(quantity) as TotalSold
FROM Purchase
GROUP BY month
HAVING sum(quantity) < 10
```
Exercise

Compute the total income per month
Show only months with less than 10 items sold
Order by quantity sold and display as “TotalSold”

```
SELECT month, sum(price*quantity), sum(quantity) as TotalSold
FROM Purchase
GROUP BY month
HAVING sum(quantity) < 10
ORDER BY sum(quantity)
```
WHERE vs HAVING

• WHERE condition is applied to individual rows
  – The rows may or may not contribute to the aggregate
  – No aggregates allowed here

• HAVING condition is applied to the entire group
  – Only applicable if GROUP BY is involved
  – Entire group is returned, or removed
  – May use aggregate functions on the group
Product($\text{pid}, \text{pname}, \text{manufacturer}$)
Purchase($\text{id}, \text{product_id}, \text{price}, \text{month}$)

**Aggregate + Join**

For each manufacturer, compute how many products with price > $100 they sold
Product(pid, pname, manufacturer)
Purchase(id, product_id, price, month)

Aggregate + Join

For each manufacturer, compute how many products with price > $100 they sold

Problem: manufacturer is in Product, price is in Purchase...
Aggregate + Join

For each manufacturer, compute how many products with price > $100 they sold

Problem: manufacturer is in Product, price is in Purchase...

---

-- step 1: think about their join

SELECT ...  
FROM Product x, Purchase y  
WHERE x.pid = y.product_id  
   and y.price > 100

<table>
<thead>
<tr>
<th>manufacturer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitachi</td>
<td>150</td>
</tr>
<tr>
<td>Canon</td>
<td>300</td>
</tr>
<tr>
<td>Hitachi</td>
<td>180</td>
</tr>
</tbody>
</table>
Product\((\text{pid}, \text{pname}, \text{manufacturer})\)  
Purchase\((\text{id}, \text{product\_id}, \text{price}, \text{month})\)

**Aggregate + Join**

For each manufacturer, compute how many products with price > $100 they sold

Problem: manufacturer is in Product, price is in Purchase...

---

**step 1: think about their join**

```
SELECT ...
FROM Product x, Purchase y
WHERE x.pid = y.product_id
  and y.price > 100
```

---

**step 2: do the group-by on the join**

```
SELECT x.manufacturer, count(*)
FROM Product x, Purchase y
WHERE x.pid = y.product_id
  and y.price > 100
GROUP BY x.manufacturer
```
Aggregate + Join

Variant:
For each manufacturer, compute how many products with price > $100 they sold in each month

```sql
SELECT x.manufacturer, y.month, count(*)
FROM Product x, Purchase y
WHERE x.pid = y.product_id
    and y.price > 100
GROUP BY x.manufacturer, y.month
```

<table>
<thead>
<tr>
<th>manufacturer</th>
<th>month</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitachi</td>
<td>Jan</td>
<td>2</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Feb</td>
<td>1</td>
</tr>
<tr>
<td>Canon</td>
<td>Jan</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Including Empty Groups

• In the result of a group by query, there is one row per group in the result

```
SELECT x.manufacturer, count(*)
FROM Product x, Purchase y
WHERE x.pname = y.product
GROUP BY x.manufacturer
```
Including Empty Groups

```sql
SELECT x.manufacturer, count(y.pid)
FROM Product x LEFT OUTER JOIN Purchase y
ON x.pname = y.product
GROUP BY x.manufacturer
```

Count(pid) is 0 when all pid’s in the group are NULL
What we have in our SQL toolbox

- Projections (SELECT * / SELECT c1, c2, …)
- Selections (aka filtering) (WHERE cond)
- Joins (inner and outer)
- Aggregates
- Group by
- Inserts, updates, and deletes

Make sure you read the textbook!
Subqueries

• A subquery is a SQL query nested inside a larger query
• Such inner-outer queries are called nested queries
• A subquery may occur in:
  – A SELECT clause
  – A FROM clause
  – A WHERE clause

• Rule of thumb: avoid nested queries when possible
  – But sometimes it’s impossible, as we will see
Subqueries…

• Can return a single value to be included in a SELECT clause
• Can return a relation to be included in the FROM clause, aliased using a tuple variable
• Can return a single value to be compared with another value in a WHERE clause
• Can return a relation to be used in the WHERE or HAVING clause under an existential quantifier
1. Subqueries in SELECT

Product (\textit{pname}, price, cid)
Company (\textit{cid}, \textit{cname}, \textit{city})

For each product return the city where it is manufactured

\begin{verbatim}
SELECT X.pname, (SELECT Y.city FROM Company Y WHERE Y.cid=X.cid) as City
FROM Product X
\end{verbatim}

What happens if the subquery returns more than one city?
We get a runtime error
(and SQLite simply ignores the extra values…)

"correlated subquery"
Whenever possible, don’t use a nested queries:

```
SELECT X.pname, (SELECT Y.city 
    FROM Company Y 
    WHERE Y.cid = X.cid) as City 
FROM Product X
```

We have “unnested” the query.
1. Subqueries in SELECT

Compute the number of products made by each company

```
SELECT DISTINCT C.cname, (SELECT count(*)
FROM Product P
WHERE P.cid=C.cid)
FROM Company C
```
1. Subqueries in SELECT

Compute the number of products made by each company

```sql
SELECT DISTINCT C.cname, 
(SELECT count(*)
FROM Product P
WHERE P.cid=C.cid)
FROM Company C
```

Better: we can unnest using a GROUP BY

```sql
SELECT C.cname, count(*)
FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname
```
1. Subqueries in SELECT

But are these really equivalent?

```
SELECT DISTINCT C.cname, (SELECT count(*)
    FROM Product P
    WHERE P.cid=C.cid)
FROM Company C
```

```
SELECT C.cname, count(*)
FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname
```
1. Subqueries in SELECT

But are these really equivalent?

```
SELECT DISTINCT C.cname, (SELECT count(*)
    FROM Product P
    WHERE P.cid=C.cid)
FROM Company C
```

```
SELECT C.cname, count(*)
FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname
```

No! Different results if a company has no products

```
SELECT C.cname, count(pname)
FROM Company C LEFT OUTER JOIN Product P
    ON C.cid=P.cid
GROUP BY C.cname
```
2. Subqueries in FROM

Find all products whose prices is $> 20$ and $< 500$

```
SELECT X.pname
FROM (SELECT *
    FROM Product AS Y
    WHERE price > 20) as X
WHERE X.price < 500
```

Product (pname, price, cid)
Company (cid, cname, city)
2. Subqueries in FROM

Find all products whose prices is > 20 and < 500

```
SELECT X.pname
FROM (SELECT *
      FROM Product AS Y
      WHERE price > 20) as X
WHERE X.price < 500
```

Try unnest this query!
2. Subqueries in FROM

Find all products whose prices is > 20 and < 500

```
SELECT X.pname
FROM (SELECT *
      FROM Product AS Y
      WHERE price > 20) as X
WHERE X.price < 500
```

Try unnest this query!

Side note: This is not a correlated subquery. (why?)
2. Subqueries in FROM

Sometimes we need to compute an intermediate table only to use it later in a SELECT-FROM-WHERE

• Option 1: use a subquery in the FROM clause
• Option 2: use the WITH clause
  – See textbook for details
2. Subqueries in FROM

Product (pname, price, cid)
Company (cid, cname, city)

WITH myTable AS (SELECT * FROM Product AS Y WHERE price > 20)
SELECT X.pname
FROM myTable as X
WHERE X.price < 500

A subquery whose result we called myTable