Announcements

- WQ3 is out, due Sunday 11pm
- HW2 is due tomorrow (Tue) 11pm
  - H3 will be posted later this week
  - you will be using Microsoft Azure
  - we will send out codes for free student use
    - good for 6 months and up to $600

Recap from last lecture

- Subqueries can occur in many clauses:
  - SELECT
  - FROM
  - WHERE
- Monotone queries: SELECT-FROM-WHERE
  - Existential quantifier
- Non-monotone queries
  - Universal quantifier
  - Aggregation

Examples of Complex Queries

1. Find drinkers that frequent some bar that serves some beer they like.

2. Find drinkers that frequent some bar that serves only beers they don’t like.

3. Find drinkers that frequent only bars that serves some beer they like.

Example 1

Find drinkers that frequent some bar that serves some beer they like.

```
SELECT DISTINCT X.drinker
FROM Frequents X, Serves Y, Likes Z
WHERE X.bar = Y.bar AND
  Y.beer = Z.beer AND
  X.drinker = Z.drinker
```

drinker + bar they frequent + beer served that they like
=> drinker is an answer

(even though we only want the drinker,
we need the rest to know it’s an answer.)
Example 2

Find drinkers that frequent some bar that serves only beers they don’t like

Let’s find the others (drop the NOT):
Drinkers that frequent some bars that serve some beer they like.

That’s the previous query… Let’s write it with a subquery:

```
SELECT DISTINCT X.drinker
FROM Frequents X, Serves Y, Likes Z
WHERE X.bar = Y.bar AND
Y.beer = Z.beer AND
X.drinker = Z.drinker
```
Example 3

Find drinkers that frequent only bars that serve some beer they like.

\[
X \text{ frequents only bars that serve some beer } X = \ \text{X does NOT frequent some bar that serves only beer } X \text{ doesn't like}
\]

Let's find the others (drop the NOT):
Drinkers that frequent some bar that serves only beer they don't like.

\[
\text{SELECT DISTINCT } X.\text{drinker} \\
\text{FROM Frequents } X \\
\text{WHERE NOT EXISTS (SELECT * FROM Serves Y, Likes Z WHERE X.bar=Y.bar AND X.drinker=Z.drinker AND Y.beer = Z.beer)}
\]

That's the previous query!

Example 3

Find drinkers that frequent only bars that serves some beer they like.

Let's find the others (drop the NOT):
Drinkers that frequent some bar that serves only beer they don't like.

\[
\text{SELECT DISTINCT U.drinker} \\
\text{FROM Frequents U WHERE U.drinker NOT IN (SELECT DISTINCT X.drinker FROM Frequents X WHERE NOT EXISTS (SELECT * FROM Serves Y, Likes Z WHERE X.bar=Y.bar AND X.drinker=Z.drinker AND Y.beer = Z.beer))}
\]

That's the previous query! But write it as a nested query:

```
```

Example 3

Find drinkers that frequent only bars that serves some beer they like.

Let's find the others (drop the NOT):
Drinkers that frequent some bar that serves only beer they don't like.

\[
\text{SELECT DISTINCT U.drinker} \\
\text{FROM Frequents U WHERE U.drinker NOT IN (SELECT DISTINCT X.drinker FROM Frequents X WHERE NOT EXISTS (SELECT * FROM Serves Y, Likes Z WHERE X.bar=Y.bar AND X.drinker=Z.drinker AND Y.beer = Z.beer))}
\]

That's the previous query! But write it as a nested query:

```
```

Example 3

Find drinkers that frequent only bars that serves some beer they like.

Let's find the others (drop the NOT):
Drinkers that frequent some bar that serves only beer they don't like.

\[
\text{SELECT DISTINCT U.drinker} \\
\text{FROM Frequents U WHERE U.drinker NOT IN (SELECT DISTINCT X.drinker FROM Frequents X WHERE NOT EXISTS (SELECT * FROM Serves Y, Likes Z WHERE X.bar=Y.bar AND X.drinker=Z.drinker AND Y.beer = Z.beer))}
\]

That's the previous query! But write it as a nested query:

```
```

Example 3

Find drinkers that frequent only bars that serves some beer they like.

Let's find the others (drop the NOT):
Drinkers that frequent some bar that serves only beer they don't like.

That's the previous query!

Now negate!

```
```

Now need three nested queries

Unnesting Aggregates

Find the number of companies in each city

```
SELECT DISTINCT X.city, (SELECT count(*) FROM Company Y WHERE X.city = Y.city) FROM Company X
```

Note: no need for DISTINCT (DISTINCT is the same as GROUP BY)

CSE 414 - Spring 2017

3
Unnesting Aggregates

Find the number of companies in each city

\[
\text{SELECT DISTINCT X.city, (SELECT count(*) }
\text{FROM Company Y }
\text{WHERE X.city = Y.city)}
\]

\[
\text{FROM Company X}
\]

\[
\text{SELECT city, count(*)}
\text{FROM Company }
\text{GROUP BY city}
\]

Equivalent queries

\[
\text{SELECT DISTINCT X.city, (SELECT count(*) }
\text{FROM Company Y }
\text{WHERE X.city = Y.city)}
\]

\[
\text{FROM Company X}
\]

\[
\text{SELECT city, count(*)}
\text{FROM Company }
\text{GROUP BY city}
\]

Wait... are they equivalent?

Grouping vs Nested Queries

Find authors who wrote ≥ 10 documents:

Attempt 1: with nested queries

\[
\text{SELECT DISTINCT Author.name}
\text{FROM Author, Wrote }
\text{WHERE 10 <= (SELECT count(url) }
\text{FROM Wrote }
\text{WHERE Author.login=Wrote.login)}
\]

This is SQL by a novice

Finding Witnesses

For each city, find the most expensive product made in that city

Finding the maximum price is easy...

\[
\text{SELECT x.city, max(y.price)}
\text{FROM Company x, Product y }
\text{WHERE x.city = y.city)
\]

But we need the witnesses, i.e. the products with max price

More Unnesting

Find authors who wrote ≥ 10 documents:

Attempt 1: with nested queries

\[
\text{SELECT product, Sum(quantity) AS TotalSales}
\text{FROM Purchase }
\text{WHERE price > 1 }
\text{GROUP BY product}
\]

\[
\text{SELECT DISTINCT x.product, (SELECT Sum(y.quantity) }
\text{FROM Purchase y }
\text{WHERE x.product = y.product}
\text{AND y.price > 1)}
\text{AS TotalSales}
\]

\[
\text{FROM Purchase x}
\text{WHERE x.price > 1}
\]

Why twice?

More Unnesting

Find authors who wrote ≥ 10 documents:

Attempt 2: using GROUP BY and HAVING

\[
\text{SELECT name}
\text{FROM Author, Wrote }
\text{WHERE Author.login=Wrote.login}
\text{GROUP BY name}
\text{HAVING count(url) >= 10}
\]

This is SQL by an expert

More Unnesting

Find authors who wrote ≥ 10 documents:

Attempt 1: with nested queries

\[
\text{SELECT product, Sum(quantity) AS TotalSales}
\text{FROM Purchase }
\text{WHERE price > 1 }
\text{GROUP BY product}
\]

\[
\text{SELECT DISTINCT x.product, (SELECT Sum(y.quantity) }
\text{FROM Purchase y }
\text{WHERE x.product = y.product}
\text{AND y.price > 1)}
\text{AS TotalSales}
\]

\[
\text{FROM Purchase x}
\text{WHERE x.price > 1}
\]

Why twice?
Finding Witnesses

To find the witnesses: compute the maximum price in a subquery

```
SELECT DISTINCT u.city, v.pname, v.price
FROM Company u, Product v,
(SELECT x.city, max(y.price) as maxprice
FROM Company x, Product y
WHERE x.cid = y.cid
GROUP BY x.city) w
WHERE u.cid = v.cid
and u.city = w.city
and v.price = w.maxprice;
```

Not a bad solution...

Finding Witnesses

Or we can use a subquery in where clause

```
SELECT u.city, v.pname, v.price
FROM Company u, Product v
WHERE u.cid = v.cid AND
  v.price >= ALL (SELECT y.price
                  FROM Company x, Product y
                  WHERE u.city = x.city
                  and x.cid = y.cid);
```

Finding Witnesses

There is a more concise solution here:

Idea: Product JOIN Product ON “made in the same city”
Then group by first product
Then check that first product is more expensive than all of the second products in the group.

```
SELECT u.city, v.pname, v.price
FROM Company u, Product v, Company x, Product y
WHERE u.cid = v.cid
and u.city = x.city
and x.cid = y.cid
GROUP BY u.city, v.pname, v.price
HAVING v.price = max(y.price);
```

BigQuery Demo

Supports SQL queries on TB of data

(we won’t use it in this class, but useful to know about)