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.)

What happens if we didn’t write DISTINCT?
Example 2

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

Existential

Universal

Example 2

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

bar serves only beers that X does not like =
bar that does NOT serve some beer that X does like

Let’s find the others (drop the NOT):

Drinkers that frequent some bars 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

Example 3

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

Universal

Existential

SELECT DISTINCT X.drinker
FROM Frequents X
WHERE NOT EXISTS (SELECT *
FROM Serves Y, Likes Z
WHERE X.bar = Y.bar AND
Y.beer = Z.beer AND
X.drinker = Z.drinker
WHERE EXISTS (SELECT * FROM Serves Y, Likes Z
WHERE X.bar = Y.bar AND
Y.beer = Z.beer)

Now negate!

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
WHERE X.bar = Y.bar AND
Y.beer = Z.beer)

Show Your Work
Example 3

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

X frequents only bars that serve some beer X likes = X does NOT frequent some bar that serves only beer X doesn't 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!

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)

Now need three nested queries

UNNESTING AGGREGATES

Find the number of companies in each city

SELECT X.city, (SELECT count(*) 
FROM Company Y 
WHERE X.city = Y.city) 
FROM Company X 
GROUP BY city 
(Note: no need for DISTINCT 
(DISTINCT is the same as GROUP BY)

PRODUCT (pname, price, cid) 
COMPANY (cid, cname, city)
Unnesting Aggregates

Find the number of companies in each city

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

Equivalent queries

```
SELECT city,  count(*)
FROM Company
GROUP BY city
```

Wait… are they equivalent?

More Unnesting

Grouping vs. Nested Queries

```sql
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

```
SELECT DISTINCT x.product, (SELECT Sum(y.quantity)
FROM Purchase y
WHERE x.product = y.product
AND y.price > 1)
AS TotalSales
FROM Purchase x
WHERE x.price > 1
```

Why twice?

Finding Witnesses

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

```
SELECT x.city, max(y.price)
FROM Company x, Product y
WHERE x.cid = y.cid
GROUP BY x.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

```
SELECT Author.name
FROM Author, Wrote
WHERE Author.login=Wrote.login
GROUP BY Author.name
HAVING count(Wrote.url) >= 10
```

This is SQL by an expert

Attempt 2: using GROUP BY and HAVING

```
SELECT name
FROM Author, Wrote
WHERE Author.login=Wrote.login
GROUP BY name
HAVING count(url) >= 10
```

This is SQL by a novice

Finding the maximum price is easy…

This is SQL by an expert

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

Finding the maximum price is easy…
Finding Witnesses

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

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

Or we can use a subquery in where clause

$$\text{SELECT u.city, v.pname, v.price}
\text{FROM Company u, Product v}
\text{WHERE u.cid = v.cid AND}
\text{v.price >= ALL (SELECT y.price}
\text{FROM Company x, Product y}
\text{WHERE u.city = x.city}
\text{and x.cid = y.cid};$$

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.

$$\text{SELECT u.city, v.pname, v.price}
\text{FROM Company u, Product v, Company x, Product y}
\text{WHERE u.cid = v.cid AND u.city = x.city and x.cid = y.cid}
\text{GROUP BY u.city, v.pname, v.price}
\text{HAVING v.price = max(y.price);}$$