#### Introduction to Database Systems **CSE 414**

Lecture 6: SQL Subqueries

CSE 414 - Autumn 2018

#### Announcements

- · Web Quiz 2 due Friday night
- HW 2 due Tuesday at midnight
- · Section this week important for HW 3, must attend

CSE 414 - Autumn 2018

#### **Announcements**

· Many students did not turn in hw1 correctly need to make sure your files are here:

https://gitlab.cs.washington.edu/cse414-2018au/cse414-[username]/tree/master/hw/hw[homework#]/submission

E.g. https://qitlab.cs.washington.edu/cse414-2018au/cse414-maas/tree/master/hw/hw1/submission

AND you have the hw1 tag here:

https://gitlab.cs.washington.edu/cse414-2018au/cse414-[username]/tags

- · Commit, then use ./turnInHw.sh hw2 script.
- MUST have this correct for HW2 CSE 414 Autumn 2018

# Semantics of SQL With **Group-By**

SELECT S FROM  $R_1,...,R_n$ WHERE C1 GROUP BY a<sub>1</sub>,...,a<sub>k</sub> HAVING C2

**FWGHOS** 

**Evaluation steps:** 

- 1. Evaluate FROM-WHERE using Nested Loop Semantics
- Group by the attributes a<sub>1</sub>,...,a<sub>k</sub>
- Apply condition C2 to each group (may have aggregates)
- Compute aggregates in S and return the result

Product(<u>pid</u>,pname,manufacturer) Purchase(<u>id</u>,product\_id,price,month)

# Aggregate + Join

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

CSE 414 - Autumn 2018

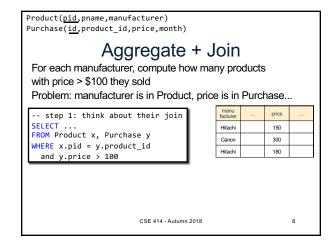
Product(pid,pname,manufacturer)  $Purchase(\overset{'}{\underline{id}}, product\_id, price, month)$ 

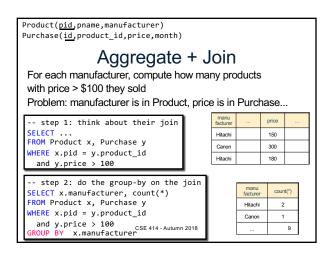
# Aggregate + Join

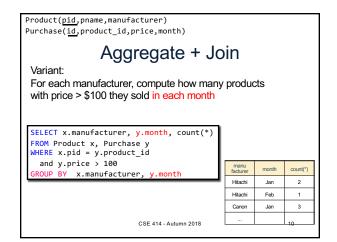
For each manufacturer, compute how many products with price > \$100 they sold

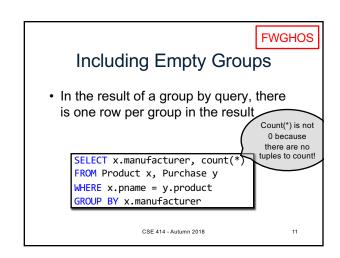
Problem: manufacturer is in Product, price is in Purchase...

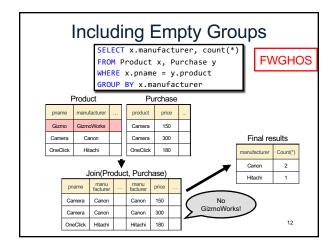
CSE 414 - Autumn 2018

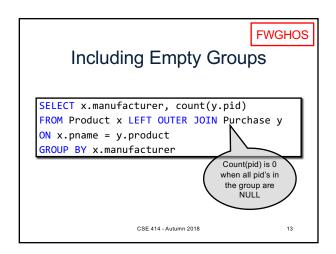


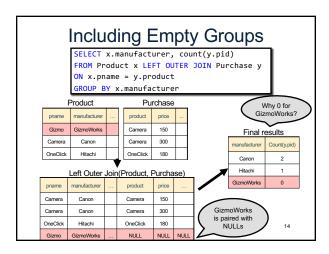


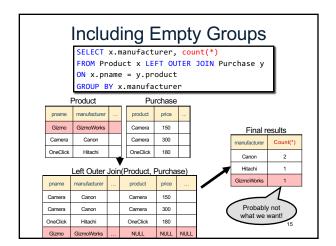












### What we have in our SQL toolbox

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

Make sure you read the textbook!

CSE 414 - Autumn 2018

# Subqueries

- In the relational model, the output of a query is also a relation
- Can use output of one query as input to another

CSE 414 - Autumn 2018

17

# 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

FWGHOS
CSE 414 - Autumn 2018

18

# 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

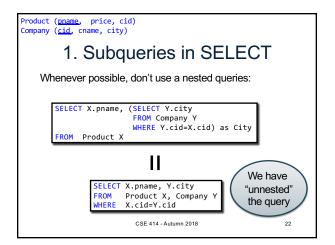
CSE 414 - Autumn 2018

2018 19

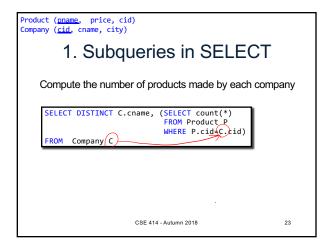
# Subqueries... Subqueries are often: Intuitive to write Be careful!

Slow

```
1. Subqueries in SELECT
  Product (pname, price, cid)
  Company (cid, cname, city)
For each product return the city where it is manufactured
                                                'correlated
SELECT X.pname, (SELECT Y.city
                FROM Company Y
WHERE Y.cid=X.cid as City
                                                subquery"
 What happens if the subquery returns more than one city?
 We get a runtime error
       (and SQLite simply ignores the extra values...)
                     CSE 414 - Autumn 2018
                                                      21
```



CSE 414 - Autumn 2018



```
Product (<u>pname</u>, price, cid)
Company (<u>cid</u>, cname, city)
          1. Subqueries in SELECT
    Compute the number of products made by each company
     SELECT DISTINCT C.cname,
                                     FROM Product P
                                     WHERE P.cid=C.cid)
      FROM Company C
                           SELECT C.cname, count(*)
      Better: we can
                            FROM Company C, Product P
WHERE C.cid=P.cid
GROUP BY C.cname
      unnest using a
                            WHERE
      GROUP BY
                             CSE 414 - Autumn 2018
```

```
Product (<u>pname</u>, price, cid)
Company (<u>cid</u>, cname, city)
           1. Subqueries in SELECT
     But are these really equivalent?
     SELECT DISTINCT C.cname. (SELECT count(*)
                                        FROM Product P
                                       WHERE P.cid=C.cid)
            Company C
     SELECT C.cname, count(*)
FROM Company C, Product P
     WHERE C.cid=P.cid
GROUP BY C.cname
                                CSE 414 - Autumn 2018
                                                                          25
```

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

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

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

CSE 414-Autumn 2018 26
```

```
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
```

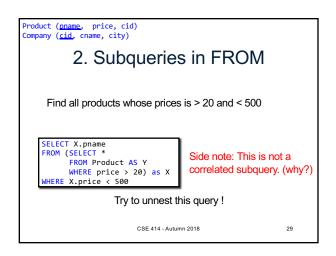
Product (pname, price, cid)
Company (cid. cname, city)

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 to unnest this query!



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

3. Subqueries in WHERE

Find all companies that make some products with price < 200

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

3. Subqueries in WHERE

Find all companies that make some products with price < 200

Existential quantifiers

CSE 414 - Autumn 2018

33

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

3. Subqueries in WHERE

Find all companies that make some products with price < 200

Existential quantifiers

Using EXISTS:

SELECT DISTINCT C.cname
FROM Company C
WHERE EXISTS (SELECT *
FROM Product P
WHERE C.cid = P.cid and P.price < 200)

CSE 414-Autumn 2018 34
```

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

3. Subqueries in WHERE

Find all companies that make some products with price < 200

Existential quantifiers

Using IN

SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid IN (SELECT P.cid
FROM Product P
WHERE P.price < 200)

CSE 414 - Autumn 2018

35
```

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

3. Subqueries in WHERE

Find all companies that make some products with price < 200

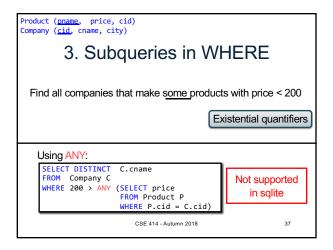
Existential quantifiers

Using ANY:

SELECT DISTINCT C.cname
FROM Company C
WHERE 200 > ANY (SELECT price
FROM Product P
WHERE P.cid = C.cid)

CSE 414 - Autumn 2018

36
```



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

3. Subqueries in WHERE

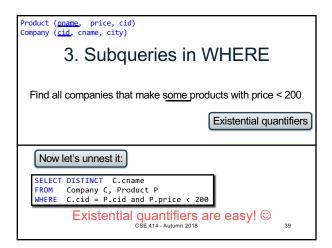
Find all companies that make some products with price < 200

Existential quantifiers

Now let's unnest it:

SELECT DISTINCT C.cname
FROM Company C, Product P
WHERE C.cid = P.cid and P.price < 200

CSE 414 - Autumn 2018 38
```



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

3. Subqueries in WHERE
Find all companies s.t. all their products have price < 200
same as:
Find all companies that make only products with price < 200
```

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

3. Subqueries in WHERE
Find all companies s.t. all their products have price < 200
same as:
Find all companies that make only products with price < 200

Universal quantifiers
```

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

3. Subqueries in WHERE
Find all companies s.t. all their products have price < 200
same as:
Find all companies that make only products with price < 200

Universal quantifiers

Universal quantifiers are hard!
```

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

3. Subqueries in WHERE

Find all companies s.t. all their products have price < 200

1. Find the other companies that make some product ≥ 200

SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid IN (SELECT P.cid
FROM Product P
WHERE P.price >= 200)
```

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

3. Subqueries in WHERE

Find all companies s.t. all their products have price < 200

1. Find the other companies that make some product ≥ 200

SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid IN (SELECT P.cid
FROM Product P
WHERE P.price >= 200)

2. Find all companies s.t. all their products have price < 200

SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid NOT IN (SELECT P.cid
FROM Product P
WHERE P.price >= 200)
```

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

3. Subqueries in WHERE

Find all companies s.t. all their products have price < 200

Universal quantifiers

Using EXISTS:

SELECT DISTINCT C.cname
FROM Company C
WHERE NOT EXISTS (SELECT *
FROM Product P
WHERE P.cid = C.cid and P.price >= 200)

CSE 414-Autumn 2018

45
```

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

3. Subqueries in WHERE

Find all companies s.t. all their products have price < 200

Universal quantifiers

Using ALL:

SELECT DISTINCT C.cname
FROM Company C
WHERE 200 >= ALL (SELECT price
FROM Product P
WHERE P.cid = C.cid)

CSE 414 - Autumn 2018

46
```

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

3. Subqueries in WHERE

Find all companies s.t. all their products have price < 200

Universal quantifiers

Using ALL:

SELECT DISTINCT C.cname
FROM Company C
WHERE 200 >= ALL (SELECT price
FROM Product P
WHERE P.cid = C.cid)

CSE 414-Autumn 2018

47
```

# Question for Database Theory Fans and their Friends

- Can we unnest the universal quantifier query?
- We need to first discuss the concept of monotonicity

CSE 414 - Autumn 2018

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

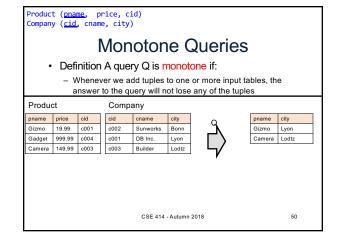
Monotone Queries

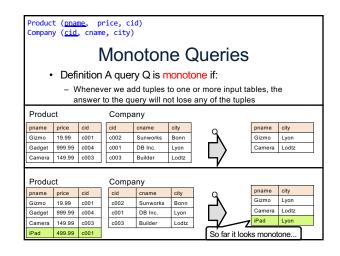
• Definition A query Q is monotone if:

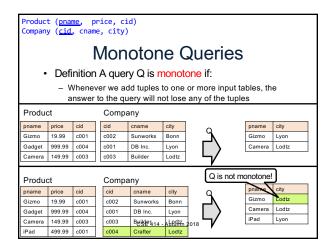
- Whenever we add tuples to one or more input tables, the answer to the query will not lose any of the tuples

CSE 414 - Autumn 2018

49







# Monotone Queries n: If Q is a SELECT-FROM-WHER

 Theorem: If Q is a SELECT-FROM-WHERE query that does not have subqueries, and no aggregates, then it is monotone.

CSE 414 - Autumn 2018

------

53

#### Monotone Queries

- Theorem: If Q is a SELECT-FROM-WHERE query that does not have subqueries, and no aggregates, then it is monotone.
- Proof. We use the nested loop semantics: if we insert a tuple in a relation R<sub>i</sub>, this will not remove any tuples from the answer

SELECT aı, a², ..., ak FROM R. AS xı, R² AS x², ..., Rn AS xn WHERE Conditions for x<sub>1</sub> in R<sub>1</sub> do for x<sub>2</sub> in R<sub>2</sub> do ... for x<sub>n</sub> in R<sub>n</sub> do if Conditions output (a<sub>1,...,ak</sub>)

CSE 414 - Autumn 2018

Product (<u>pname</u>, price, cid) Company (<u>cid</u>, cname, city)

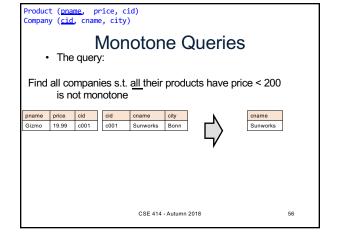
#### Monotone Queries

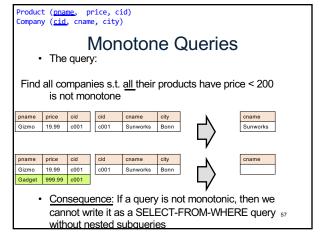
· The query:

Find all companies s.t. <u>all</u> their products have price < 200 is not monotone

CSE 414 - Autumn 2018

55





### Queries that must be nested

 Queries with universal quantifiers or with negation

CSE 414 - Autumn 2018

### Queries that must be nested

- Queries with universal quantifiers or with negation
- · Queries that use aggregates in certain ways
  - sum(..) and count(\*) are NOT monotone, because they do not satisfy set containment
  - select count(\*) from R is not monotone!

CSE 414 - Autumn 2018

mn 2018 59

Author(login,name)
Wrote(login,url)

More Unnesting

Find authors who wrote ≥ 10 documents:

Author(login, name) Wrote(login,url) More Unnesting Find authors who wrote ≥ 10 documents: This is Attempt 1: with nested queries SQL by a novice SELECT DISTINCT Author.name FROM Author (SELECT count(Wrote.url) WHERE FROM Wrote WHERE Author.login=Wrote.login) >= 10

Author(login, name) Wrote(login,url) More Unnesting Find authors who wrote ≥ 10 documents: Attempt 1: with nested queries Attempt 2: using GROUP BY and HAVING SELECT Author.name This is FROM Author, Wrote SQL by WHERE Author.login=Wrote.login **GROUP BY Author.name** an expert **HAVING** count(wrote.url) >= 10

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

Finding Witnesses

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

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

Finding Witnesses

For each city, find the most expensive product made in that city
Finding the maximum price is easy...

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
```

```
Product (pname, price, cid)
Company (cid, cname, city)
             Finding Witnesses
  To find the witnesses, compute the maximum price
  in a subquery (in FROM)
 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
                                     Joining three
       GROUP BY x.city) w
                                        tables
 WHERE u.cid = v.cid
        and u.city = w.city
        and v.price = w.maxprice;
                                                   65
```

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

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);
```