

# CSE 344: Section 3

# Grouping and Nesting

January 18th, 2017

# Group By

- Powerful tool to handle “categories”
  - Treat rows with a same attribute as a category
- Careful when selecting
  - Only select attributes appeared in **GROUP BY** or **aggregates**
  - SQLite will guess (arbitrarily pick a value) `^(ツ)^/`
  - SQL Server will throw an error `∩`_´∩`

# Group By - Examples

Do these queries work?

Enrolled(stu\_id, course\_num)

johndoe	311
johndoe	344
maryjane	311
maryjane	351
maryjane	369

```
SELECT stu_id, course_num
FROM Enrolled
GROUP BY stu_id
```

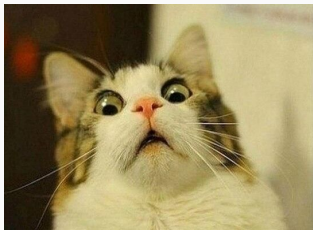
```
SELECT stu_id, count(course_num)
FROM Enrolled
GROUP BY stu_id
```

# Group By - Examples

Do these queries work?

Enrolled(stu\_id, course\_num)

johndoe	?
maryjane	?



```
SELECT stu_id, course_num
FROM Enrolled
GROUP BY stu_id
```

```
SELECT stu_id, count(course_num)
FROM Enrolled
GROUP BY stu_id
```

# Group By - Examples

Do these queries work?

Enrolled(stu\_id, course\_num)

johndoe	311	johndoe	2
johndoe	344		
maryjane	311	maryjane	3
maryjane	351		
maryjane	369		

```
SELECT stu_id, course_num  
FROM Enrolled  
GROUP BY stu_id
```

```
SELECT stu_id, count(course_num)  
FROM Enrolled  
GROUP BY stu_id
```

# Witnessing (i.e. argmax)

Find the student who is taking the most classes.

Student(stu\_id, id\_num)

Enrolled(id\_num, class)

johndoe	973	973	CSE 311
maryjane	712	973	CSE 344
alsmith	899	712	CSE 311
		899	CSE 351

```
SELECT S.stu_id
      FROM Student S, Enrolled E
     WHERE S.id_num = E.id_num
     GROUP BY S.stu_id
    HAVING count(E.class) >= ALL(
        SELECT count(E1.class)
           FROM Enrolled E1
          GROUP BY E1.id_num);
```

# Nested Queries

- Avoid when possible
- Danger of making simple queries slow and complicated
- Just because you can do it, doesn't mean you should



# Subquery in SELECT

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



# Subquery in SELECT

Unnest using JOIN and GROUP BY

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

# Subquery in FROM

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

**More readable:** WITH <name> AS (<subquery>)

# Subquery in FROM

Unnest using WHERE

```
SELECT X.pname  
       FROM Product AS X  
       WHERE X.price < 500 AND X.price > 20;
```

# Subquery in WHERE

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

# Subquery in WHERE

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

# Subquery in WHERE Syntax

- `SELECT ..... WHERE EXISTS (<sub>);`
- `SELECT ..... WHERE NOT EXISTS (<sub>);`
- `SELECT ..... WHERE attribute IN (<sub>);`
- `SELECT ..... WHERE attribute NOT IN (<sub>);`
- `SELECT ..... WHERE attribute > ANY (<sub>);`
- `SELECT ..... WHERE attribute > ALL (<sub>);`

# (Non-)monotonic Queries

- “Can we take back outputs by looking at more data?”
- Is this a monotonic query?

```
SELECT count (*)  
  FROM T1  
 GROUP BY T1.attr
```

# (Non-)monotonic Queries

- “Can we take back outputs by looking at more data?”
- Is this a monotonic query?

```
SELECT count (*)  
  FROM T1  
 GROUP BY T1.attr
```

No! This query does not satisfy **set containment**.

Ex:

Current output:  $\{(6), (23), (10)\}$

After more data:  $\{(6), (23), (11)\}$

$\{(6), (23), (10)\} \not\subseteq \{(6), (23), (11)\}$



# To Nest or Not to Nest

- Not an exact science
- Figuring out what is actually wanted will help you find simpler solutions (best way is to practice)
- Trigger words to use sub-querying
  - Every, All (universal quantifiers)
  - No, None, Never (negation)
  - Only