Administrative Notes – February 14, 2023

- Feb 17: Assignment 3 (pairs if you want) due
  - Make sure you sign up in pairs beforehand as we need time to configure Canvas to allow both partners to view the submission and subsequent feedback
    - Everyone who has signed up as of last night has been put into a group on Canvas

- Feb 17: Assignment 4 will be released

- Feb 20 – 24: Reading Break! (yay!)
  - No lectures, tutorials, or office hours during this week

Review: Examples for Division

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A/B1

sno
s1
s2
s3
s4

A/B2

sno
s1
s4

A/B3

sno
s1

Review: Division in SQL

Find students who've taken all classes.

```
SELECT sname
FROM   Student S
WHERE NOT EXISTS
              ((SELECT  C.name
                 FROM  Class C)
                 EXCEPT
                  (SELECT  E.cname
                   FROM  Enrolled E
                   WHERE E.snum=S.snum))
```

(method 1)

The hard way (without EXCEPT):

```
SELECT sname
FROM Student S
WHERE NOT EXISTS (
      SELECT  C.name
      FROM  Class C
      WHERE  NOT EXISTS (SELECT  E.snum
                         FROM  Enrolled E
                         WHERE  C.name= E.cname AND
                                E.snum=S.snum))
```

(method 2)

In-Class Exercise (SQL 3)

- Canvas → Modules → In Class Exercises
- You can work on it with other people around you. If you work with others, you must **write their names on your submission to acknowledge the collaboration**.
  - Everyone must submit to Canvas
- Reminder: no late submissions accepted

Clicker Question

I am ready to cover the in-class exercise.

A. Yes
B. No
C. I need two more minutes
D. I need five more minutes

Aggregate Operators

- These functions operate on the multiset of values of a column of a relation, and return a value

  AVG: average value
  MIN: minimum value
  MAX: maximum value
  SUM: sum of values
  COUNT: number of values

```
SELECT count(s.snum)
FROM enrolled e, Student S
WHERE e.snum = s.snum
```

- The following versions eliminate duplicates before applying the operation to attribute A:

  COUNT (DISTINCT A)
  SUM (DISTINCT A)
  AVG (DISTINCT A)

```
SELECT count(distinct s.snum)
FROM enrolled e, Student S
WHERE e.snum = s.snum
```

Aggregate Operators: Examples

students

```
SELECT  COUNT(*)
FROM    Student
```

Find name and age of the oldest student(s).

```
SELECT sname, age
FROM Student S
WHERE S.age = (SELECT  MAX(S2.age)
               FROM  Student S2)
```

Finding average age
of SR students.

```
SELECT AVG (age)
FROM Student
WHERE standing='SR'
```

Aggregation Examples

Find the minimum student age.

How many students have taken a class with "Database" in the title.

Student(snum,sname,major,standing,age)
Class(name,meets_at,room,fid)
Enrolled(snum,cname)
Faculty(fid,fname,deptid)

Aggregation Examples

Find the minimum student age.

```
SELECT MIN(age)
FROM student;
```

How many students have taken a class with "Database" in the title.

```
SELECT COUNT(DISTINCT snum)
FROM enrolled
WHERE cname LIKE '%Database%'
```

GROUP BY and HAVING

- Divide tuples into groups and apply aggregate operations to each group.
- Example: *Find the age of the youngest student for each major.*

For i = 'Computer Science',
'Civil Engineering'…

```
SELECT  MIN (age)
FROM     Student
WHERE  major = i
```

Problem:

We don't know how many majors exist, not to mention this is not good practice

Grouping Examples

Find the age of the youngest student who is at least 19, for each major.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
```

Snum	Major	Age
115987938	Computer Science	20
112348546	Computer Science	19
280158572	Animal Science	18
351565322	Accounting	19
556784565	Civil Engineering	21
…	…	…

No Animal Science

Major	Age
Computer Science	19
Accounting	19
Civil Engineering	21
…	…

Grouping Examples with Having

Find the age of the youngest student who is at least 19, for each major with at least 2 <u>such</u> students.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
HAVING COUNT(*) > 1
```

Snum	Major	Age
115987938	Computer Science	20
112348546	Computer Science	19
280158572	Animal Science	18
351565322	Accounting	19
556784565	Civil Engineering	21
…	…	…

Grouping Examples with Having

Find the age of the youngest student who is at least 19, for each major with at least 2 <u>such</u> students.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
HAVING COUNT(*) > 1
```

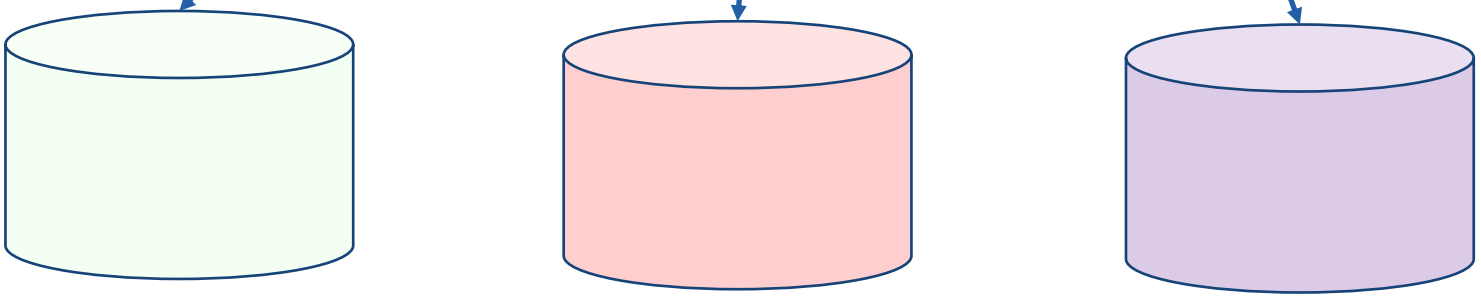
Snum	Major	Age
115987938	Computer Science	20
112348546	Computer Science	19
280158572	Animal Science	18
351565322	Accounting	19
556784565	Civil Engineering	21
…	…	…

Grouping Examples with Having

Find the age of the youngest student who is at least 19, for each major with at least 2 <u>such</u> students.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
HAVING COUNT(*) > 1
```

Snum	Major	Age
115987938	Computer Science	20
112348546	Computer Science	19
280158572	Animal Science	18
351565322	Accounting	19
556784565	Civil Engineering	21
…	…	…

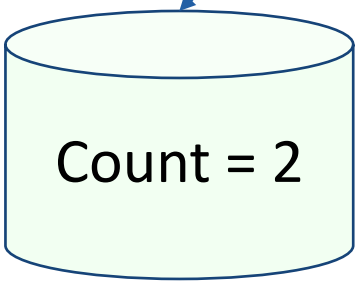


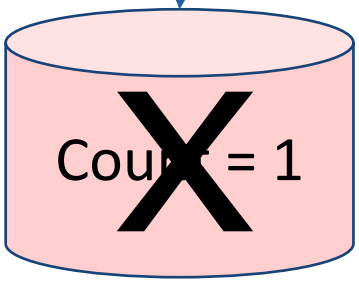
Grouping Examples with Having

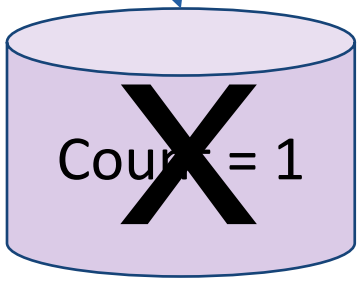
Find the age of the youngest student who is at least 19, for each major with at least 2 <u>such</u> students.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
HAVING COUNT(*) > 1
```

Snum	Major	Age
115987938	Computer Science	20
112348546	Computer Science	19
280158572	Animal Science	18
351565322	Accounting	19
556784565	Civil Engineering	21
…	…	…







Grouping Examples with Having

Find the age of the youngest student who is at least 19, for each major with at least 2 <u>such</u> students.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
HAVING COUNT(*) > 1
```

Snum	Major	Age
115987938	Computer Science	20
112348546	Computer Science	19
280158572	Animal Science	18
351565322	Accounting	19
556784565	Civil Engineering	21
…	…	…

Major	Age
Computer Science	19

And there are rules

Find the age of the youngest student who is at least 19, for each major with at least 2 such students

```
SELECT      major, MIN(age)
FROM        Student
WHERE       age >= 19
GROUP BY    major
HAVING      COUNT(*) > 1
```

- Would it make sense if I select age instead of MIN(age)?
- *Would it make sense if I select snum to be returned?*
- *Would it make sense if I select major to be returned?*

And there are rules

Find the age of the youngest student who is at least 19, for each major with at least 2 <u>such</u> students.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
HAVING COUNT(*) > 1
```

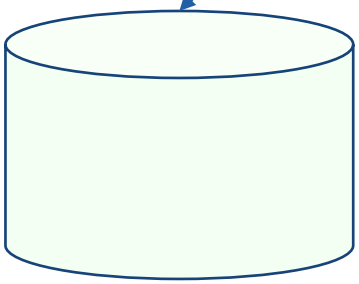
- Would it make sense if I select `age` instead of `MIN(age)`?
- Would it make sense if I select `snum` to be returned?
- Would it make sense if I select `major` to be returned?

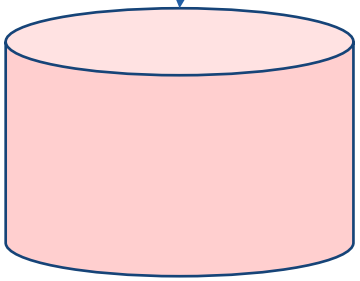
And there are rules

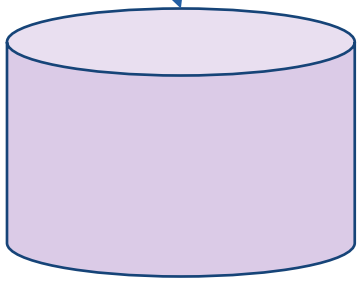
Find the age of the youngest student who is at least 19, for each major with at least 2 <u>such</u> students.

```
SELECT major, MIN(age)
FROM Student
WHERE age >= 19
GROUP BY major
HAVING COUNT(*) > 1
```

Snum	Major	Age
115987938	Computer Science	20
112348546	Computer Science	19
280158572	Animal Science	18
351565322	Accounting	19
556784565	Civil Engineering	21
…	…	…







GROUP BY and HAVING (cont)

```
SELECT [DISTINCT]  target-list
FROM relation-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
ORDER BY target-list
```

- The *target-list* contains
  (i) attribute names
  (ii) terms with aggregate operations (e.g., `MIN(S.age)`).
- Attributes in (i) must also be in *grouping-list*.
  - each answer tuple corresponds to a *group*,
  - *group* = a set of tuples with same value for all attributes in *grouping-list*
  - selected attributes must have a single value per group.
- Attributes in *group-qualification* are either in *grouping-list* or are arguments to an aggregate operator.

Conceptual Evaluation of a Query

1. compute the cross-product of *relation-list* from
2. keep only tuples that satisfy *qualification* where
3. partition the remaining tuples into groups by the value of attributes in *grouping-list* group by
4. keep only the groups that satisfy *group-qualification* (expressions in *group-qualification* must have a *single value per group*!) having
5. delete fields that are not in *target-list* select
6. generate one answer tuple per qualifying group.

Out of Order SQL Clauses

```
6 rows selected.

SQL> ;
  1  SELECT a1.x, a2.y, COUNT(*)
  2  FROM Arc a1, Arc a2
  3  WHERE a1.y = a2.x
  4* GROUP BY a1.x, a2.y
SQL>
SQL>
SQL>
SQL> SELECT a1.x, a2.y, COUNT(*)
  2  FROM Arc a1, Arc a2
  3  GROUP BY a1.x, a2.y
  4  WHERE a1.y = a2.x;
WHERE a1.y = a2.x
*
ERROR at line 4:
ORA-00933: SQL command not properly ended
```

```
SQL> SELECT b, count(c)
  2  FROM R
  3  WHERE B > 5
  4  GROUP BY B
  5  ORDER BY count(c);

         B   COUNT(C)
---------- ----------
         6          1

SQL> SELECT b, count(c)
  2  FROM R
  3  WHERE B > 5
  4  ORDER BY count(c)
  5  GROUP BY B;
GROUP BY B
*
ERROR at line 5:
ORA-00933: SQL command not properly ended
```

GROUP BY and HAVING

Student(snum,sname,major,standing,age)
Class(name,meets_at,room,fid)
Enrolled(snum,cname)
Faculty(fid,fname,deptid)

Example 1: For each class, find the age of the youngest student who has enrolled in this class.

```
SELECT cname, MIN(age)
FROM Student S, Enrolled E
WHERE S.snum= E.snum
GROUP BY cname
```

Example 2: For each course with more than 1 enrollment, find the age of the youngest student who has taken this class.

```
SELECT cname,  MIN(age)
FROM Student S, Enrolled E
WHERE S.snum = E.snum
GROUP BY cname
HAVING COUNT(*) > 1
```

← per group qualification!

Clicker Question: Grouping

clickergrouping.sql

Compute the result of the query:

```
SELECT a1.x, a2.y, COUNT(*)
FROM Arc a1, Arc a2
WHERE a1.y = a2.x
GROUP BY a1.x, a2.y
```

Which of the following is in the result?

A. (1,3,2)

B. (4,2,6)

C. (4,3,1)

D. All of the above

E. None of the above

x	y	COUNT(*)
1	3	2
2	4	2
3	1	6
3	2	2
4	2	6
4	3	1

Arc

x	y
1	2
1	2
2	3
3	4
3	4
4	1
4	1
4	1
4	2

Clicker Question: Grouping

```
SELECT a1.x, a2.y, COUNT(*)
FROM Arc a1, Arc a2
WHERE a1.y = a2.x
GROUP BY a1.x, a2.y
```

Arc a1

x	y
1	2
1	2
2	3
3	4
3	4
4	1
4	1
4	1
4	2

Arc a2

x	y
1	2
1	2
2	3
3	4
3	4
4	1
4	1
4	1
4	2

Which of the following is in the result?

A. (1,3,2)
B. (4,2,6)
C. (4,3,1)
D. All of the above
E. None of the above

Clicker Question: Grouping

```
SELECT a1.x, a2.y, COUNT(*)
FROM Arc a1, Arc a2
WHERE a1.y = a2.x
GROUP BY a1.x, a2.y
```

Arc a1

x	y
1	2
1	2
2	3
3	4
3	4
4	1
4	1
4	1
4	2

Arc a2

x	y
1	2
1	2
2	3
3	4
3	4
4	1
4	1
4	1
4	2

Which of the following is in the result?

A. (1,3,2)
B. (4,2,6)
C. (4,3,1)
D. All of the above
E. None of the above

Clicker Question: Grouping

```
SELECT a1.x, a2.y, COUNT(*)
FROM Arc a1, Arc a2
WHERE a1.y = a2.x
GROUP BY a1.x, a2.y
```

Arc a1

x	y
1	2
1	2
2	3
3	4
3	4
4	1
4	1
4	1
4	2

Arc a2

x	y
1	2
1	2
2	3
3	4
3	4
4	1
4	1
4	1
4	2

Which of the following is in the result?

A. (1,3,2)
B. (4,2,6)
C. (4,3,1)
D. All of the above
E. None of the above

Groupies of your very own

Find the average age for each standing (e.g., Freshman).

```
SELECT standing, avg(age)
FROM Student
GROUP BY standing
```

Groupies of your very own

Find the deptID and # of faculty members for each department having a department id > 20

Option 1:

```
SELECT count(*), deptid
FROM faculty
WHERE deptid > 20
GROUP BY deptid
```

Option 2:

```
SELECT count(*), deptid
FROM faculty
GROUP BY deptid
HAVING deptid > 20
```

Which one works?

A: Just 1
B: Just 2
C: Both
D: Neither