CS 5530

Database Systems
Spring 2020

Adv. Queries II

Project Teams

- Come to the front after class if you don't have a teammate yet

Database Server

- You will have access this week
  - Do not spam it with queries
  - Do not run expensive queries
  - Do not insert large quantities of data
    - Keep your total footprint < 10MB
  - Do not run queries during class

Midterm

- Next Wednesday
- 5 pages of notes

Example

Find all students younger than *everyone* in Databases

Students

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

Enrolled

sID	cID	Grd
1	3500	A
1	3810	A-
1	5530	A
2	3810	A
2	5530	B
3	3500	C
3	3810	B
4	3500	C

Courses

cID	Name
3500	SW Practice
3810	Architecture
5530	Databases

Example

- Find all students younger than everyone in ‘Databases’

```
select s1.sName from Students s1
where s1.DOB > all

    (select s2.DOB from Students s2
    natural join Enrolled natural join
    Courses c where c.cName='Databases’);
```

Example

- Find all students younger than everyone in ‘Databases’

```
select s1.sName from Students s1
where s1.DOB > all

    (select s2.DOB from Students s2 …);
```

Example

- Find all students younger than everyone in ‘Databases’

```
select s1.sName from Students s1
where s1.DOB > all

    (select s2.DOB from Students s2 …);
```

- Think of nested queries as nested for-loops

Nested Queries

```
select s1.sName                    (select s2.DOB
from Students s1                    from Students s2
where s1.DOB > all                  ...);
```

```
foreach Student s1 in Students {
  foreach Student s2 in ... {
    if(s1.DOB <= s2.DOB)
      don't select s
  }
}
```

Nested Queries

- Think of nested queries as nested for-loops

```
select s1.sName
from Students s1
where s1.DOB > all
(select s2.DOB
 from Students s2
 ...);
```

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s1.sName
from Students s1
where s1.DOB > all
(select s2.DOB
 from Students s2
 ...);
```

s1

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s1.sName
from Students s1
where s1.DOB > all
(select s2.DOB
 from Students s2
 ...);
```

s1

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s1.sName
from Students s1
where s1.DOB > all
(select s2.DOB
 from Students s2
 ...);
```

s1

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s1.sName
from Students s1
where s1.DOB > all
(select s2.DOB
 from Students s2
 ...);
```

s1

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

EXISTS

```
select x from y where
EXISTS
(select …);
```

- If any rows exist in nested query, `x` is selected

NOT EXISTS

```
select x from y where
NOT EXISTS
(select …);
```

- If nested query empty, `x` is selected

Division

- Find students taking all classes

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If there is an enrollment for {s, c}
Then this select is non-empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If inner select is non-empty, then this is false

If there is an enrollment for {s, c}
Then this select is non-empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If inner select is non-empty, then this is false

If false for all {s, c}, then this select is empty

If there is an enrollment for {s, c}
Then this select is non-empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

which makes this true

If inner select is non-empty, then this is false

If false for all {s, c}, then this select is empty

If there is an enrollment for {s, c}
Then this select is non-empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If there is **not** an enrollment for {s, c}
Then this select is empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

If inner select is empty, then this is true

If there is **not** an enrollment for {s, c}
Then this select is empty

foreach Student s
  foreach Course c
    foreach Enrollment e

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

If inner select is empty, then this is true

If ever true, then this select is non-empty

If there is **not** an enrollment for {s, c}
Then this select is empty

foreach Student s
  foreach Course c
    foreach Enrollment e

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

which makes this false

If inner select is empty, then this is true

If ever true, then this select is non-empty

If there is **not** an enrollment for {s, c}
Then this select is empty

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
 foreach Course c
  foreach Enrollment e

{3500}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
 foreach Course c
  foreach Enrollment e

{}

{3500}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
 foreach Course c
  foreach Enrollment e

{}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
 foreach Course c
  foreach Enrollment e

{}

{3810}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{}

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

{Hermoine}

{}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

foreach Student s
 foreach Course c
  foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

foreach Student s
 foreach Course c
  foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

foreach Student s
 foreach Course c
  foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

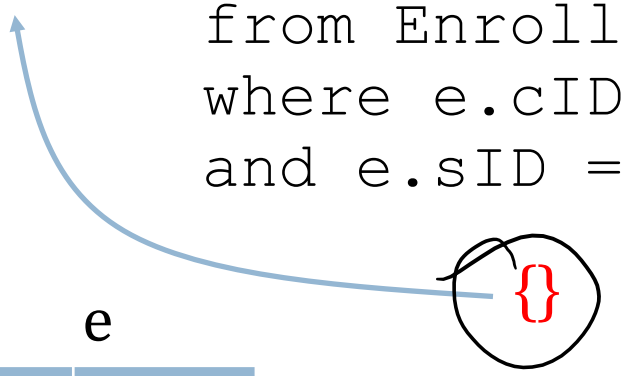
sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

foreach Student s
 foreach Course c
  foreach Enrollment e



s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

{Hermoine}

{3500}

{}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

{3500}

foreach Student s
 foreach Course c
  foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

{Hermoine}

(don't add Harry)

{3500}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

List ... S1

... S2

foreach x in S1

~~foreach ~ in S2~~

...

S2.contains

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

Only student taking all classes

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Strings

- SQL has pseudo regex:

```
… WHERE Title LIKE 'J_%m'
```

- '_' means any one character
- '%' means 0 or more arbitrary characters

Strings

- SQL has pseudo regex:

```
… WHERE Title LIKE 'J_%m'
```

- 'J' followed by at least one arbitrary character followed by 'm'

LIMIT

- Limits the number of rows returned
  - Select the first 5 rows

```
SELECT … LIMIT 5;
```

  - Select rows 3-7

```
SELECT … LIMIT 5 OFFSET 2;
```

LIMIT

- Useful in combination with `ORDER BY`
  - Find … with smallest `CardNum`

```
SELECT … LIMIT 1 ORDER BY CardNum;
```

  - Find … with biggest `CardNum`

```
SELECT … LIMIT 1 ORDER BY CardNum DESC;
```

Aggregate Functions

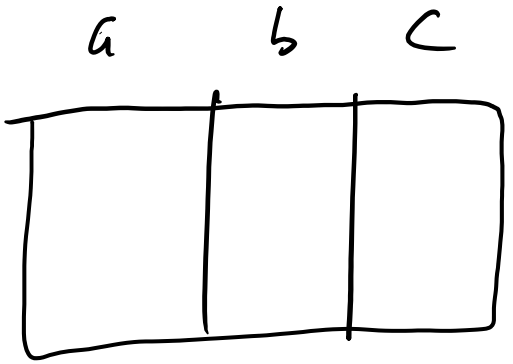
- An aggregate function returns a single number from a multiset

- Usually used on a single column

- `COUNT(...)`
- `MAX(...)`
- `MIN(...)`
- `SUM(...)`
- `AVG(...)`
- …

COUNT

- Returns count instead of rows
- How many copies of 'Harry Potter'?




```
SELECT COUNT(*)
   FROM Titles NATURAL JOIN Inventory
   WHERE Title='Harry Potter';
```

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner

COUNT

- Returns count instead of rows
- How many copies of ‘Harry Potter’?

```
SELECT COUNT(*)
  FROM Titles NATURAL JOIN Inventory
  WHERE Title='Harry Potter';
```

- Demo

GROUP BY

- Gives one representative row for each group in the specified column

```
select Major from Students;
```

ART
CS
CS
PHYS

```
select Major from Students group by
Major;
```

ART
CS
PHYS

GROUP BY

- Isn't this…

```
select Major from Students
group by Major;
```

- the same as this?

```
select DISTINCT Major from Students;
```

GROUP BY

- Gives one representative row for each group in the specified column

```
select * from Students group by Major;
```

u0123456	ART	Sandy
u0654321	CS	Jim
u0555555	PHYS	Ray

GROUP BY

- `GROUP BY` usually used in conjunction with an aggregate function

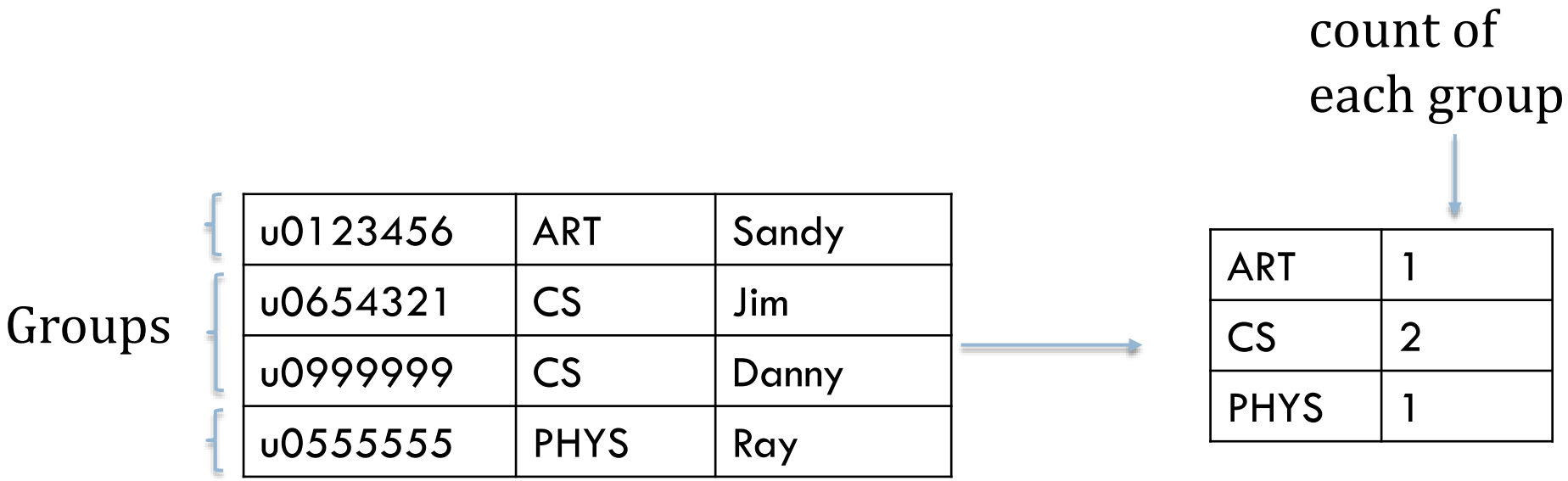
```
SELECT Major, COUNT(*) FROM Students
GROUP BY Major;
```

- `COUNT(*)` is applied to each group separately

GROUP BY

- `GROUP BY` usually used in conjunction with an aggregate function

```
SELECT Major, COUNT(*) FROM Students
GROUP BY Major;
```



GROUP BY

- `GROUP BY` usually used in conjunction with an aggregate function

```
SELECT Major, COUNT(*) FROM Students
GROUP BY Major;
```

ART	1
CS	2
PHYS	1

```
SELECT Major, AVG(gpa) FROM Students
GROUP BY Major;
```

ART	4.0
CS	3.4
PHYS	3.6

Quiz

- Find number of students in each class
- Find number of classes for each student

Students

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

Enrolled

sID	cID	Grd
1	3500	A
1	3810	A-
1	5530	A
2	3810	A
2	5530	B
3	3500	C
3	3810	B
4	3500	C

Courses

cID	Name
3500	SW Practice
3810	Architecture
5530	Databases

HAVING

- Filter on groups (instead of rows)

```
select Major, count(*) from Students
group by Major
HAVING count(*) = 2;
```

CS	2

HAVING

- Filter on groups (instead of rows)

```
select Major, count(*) from Students
group by Major
HAVING count(*) = 2;
```

- Better

```
select Major, count(*) as c
from Students group by Major
HAVING c = 2;
```

WHERE vs. HAVING

- Where is used before `GROUP BY` to filter rows
- Having is used after `GROUP BY` to filter groups
  - Or on the result of an aggregate function

WHERE vs. HAVING

- Find students with 2 or more A's

```
select sID, count(*) as c
from Enrolled e
where e.Grade='A'
group by sID
having c >= 2;
```

Aggregate Functions + Tuples

- Remember: aggregate functions don't return a row
  - They aggregate a group into one value
- Find the name of the cheapest product from each Dept

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

Aggregate Functions + Tuples

- Find the name of the cheapest product from each Dept

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select Dept, min(Price), Name
from Items group by Dept;
```

Aggregate Functions + Tuples

- Find the name of the cheapest product from each Dept

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select Dept, min(Price), Name
from Items group by Dept;
```

Dept	min(Price)	Name
produce	.70	Carrots
bakery	2	Cake

Aggregate Functions + Tuples

- Find the name of the cheapest product from each Dept

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select Dept, min(Price), Name
from Items group by Dept;
```

Dept	min(Price)	Name
produce	.70	Carrots
bakery	2	Cake

Aggregate Functions + Tuples

- Which tuple would we associate with `AVG`?

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select Dept, AVG(Price), Name
from Items group by Dept;
```

Dept	avg(Price)	Name
produce	.925	???
bakery	6	???

Aggregate Functions + Tuples

- Which tuple would we associate with `AVG`?

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select Dept, AVG(Price), Name
from Items group by Dept;
```

Dept	avg(Price)	Name
produce	.925	Carrots
bakery	6	Cake

Just pick some representative from the group

Aggregate Functions + Tuples

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select * from Items group by Dept;
```

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

Aggregate Functions + Tuples

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select Dept, min(Price), Name from Items
group by Dept;
```

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

Dept	min(P)	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	2	Cake
bakery	2	Bread

Aggregate Functions + Tuples

- Find the Name of the cheapest item from each Dept

Dept	Price	Name
produce	.70	Carrots
produce	1.15	Peaches
bakery	10	Cake
bakery	2	Bread

```
select Items.Name, mins.Dept from Items

join (select Dept, min(Price) as p from
     Items group by Dept) as mins

where Items.Price = mins.p and
Items.Dept=mins.Dept;
```

Aggregate of Aggregates?

- What if we want the average number of students per major

```
SELECT Major, COUNT(*) FROM Students
GROUP BY Major;
```

ART	1
CS	2
PHYS	1

Aggregate of Aggregates?

- What if we want the average number of students per major

```
SELECT Major, COUNT(*) FROM Students
GROUP BY Major;
```

ART	1
CS	2
PHYS	1

Select AVG(count(1))

```
SELECT AVG(counts.c) FROM
   (SELECT COUNT(*) AS c FROM Students
   GROUP BY Major) AS counts;
```

1.33

Dates

- `DATE` is a compound type, containing:
  - Year
  - Month
  - Day

Dates

- `DATE` is a compound type, containing:
  - Year
  - Month
  - Day
- But we can still do comparisons on them:

```
SELECT … WHERE DOB > '1990-01-01';
```

Dates

- `DATE` is a compound type, containing:
  - Year
  - Month
  - Day

11:15:20

- But we can still do comparisons on them:

```
SELECT … WHERE DOB > '1990-01-01';
```

Automatic conversion from string

Dates

- We can extract one component from a date:

```
select YEAR(DOB) from Students;
select MONTH(DOB) from Students;
select DAY(DOB) from Students;
```

Date Arithmetic?

```
select (“2020-02-10" - DOB) … ;
```

Date Arithmetic?

```
select (DATE("2019-02-11") - DOB) … ;
```

- Doesn't really do what you want…
  - And what about leap years?

Example

- Find age in years of all students

Example

- Find age in years of all students

```
select (2019 - year(DOB)) from Students;
```

- Better, but still some problems
  - (2019) doesn't account for today's month and day
  - Would have to re-write query every day

TIMESTAMPDIFF

- Gives number of years, months, days, or seconds between two dates

```
SELECT TIMESTAMPDIFF(unit, date1, date2) …
```

- … and it accounts for leap years

CURDATE()

- We don't want to have to rewrite this query every day:

```
SELECT TIMESTAMPDIFF(unit, date1,
DATE("2019-02-11")) …
```

- CURDATE gives us the server's date

```
SELECT TIMESTAMPDIFF(unit, date1,
CURDATE()) …
```

Dates

- Lots of built-in date functions:

```
dayname(date)
monthname(date)
dayofmonth(date)
dayofweek(date)
quarter(date)
from_unixtime(uint)
```

- …many more