Today

- Review SQL + Relational Algebra
  - HW2
  - Other examples
- Class Participation Exercise

HW2 – Exercise 1

```
CREATE TABLE Chef(
    idC INT PRIMARY KEY,
    nameC VARCHAR(255),
    nameDiner VARCHAR(255),
    cityDiner VARCHAR(255),
    nbStars  INT,
    CONSTRAINT ch_nbStars CHECK (nbStars >= 0 AND nbStars <=3)
);
```

```
CREATE TABLE Ingredient(
    idI INT PRIMARY KEY,
    nameI VARCHAR(255),
    typeI VARCHAR(100),
    unitI VARCHAR(80)
);
```

```
CREATE TABLE purchaseOrder(
    idC INT REFERENCES Chef(idC),
    idI INT REFERENCES Ingredient(idI),
    dateP DATE,
    quantityP INT,
    PRIMARY KEY(idC, idI, dateP),
    CONSTRAINT ch_quantity CHECK (quantityP > 0)
);
```

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

Q: $R \times S$

A: 25

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

S.B	S.C	S.D	R.A	R.B
x	0	3	1	x
x	0	3	3	x
y	2	1	2	y
y	3	3	2	y
y	2	0	2	y

Q: $R \bowtie S$

A: 5

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

S.B	S.C	S.D	R.A	R.B
x	0	3	3	x
y	2	1	1	x
y	3	3	3	x

Q: $R \bowtie_{A=D} S$

A: 3

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

S.B	S.C	S.D	R.A -> R.C	R.B
y	2	1	2	y
y	2	0	2	y

Q: $\rho_{C \leftarrow A}(R) \bowtie S$

A: 2

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

R.B
x
y
z
x
a

-

S.B
x
y
y

=

Result
z
a

Q: $\pi_B(R) - \pi_B(\sigma_{C<3}(S))$

A: 2

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

R.A
1
2
2
3
9

∩

S.D -> S.A
3
1
3
0
0

=

Result
1
3

Q: $\pi_A(R) \cap \rho_{A \leftarrow D}(\pi_D(S))$

A: 2

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

Q: $\pi_D(S) \bowtie S$

A: 5

S.B	S.C	S.D	S.D_2
x	0	3	3
x	0	3	3
y	2	1	1
y	3	3	3
y	3	3	3
w	3	0	0
w	3	0	0
y	2	0	0
y	2	0	0

HW2 – Exercise 2

S

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	2	0

R

A	B
1	x
2	y
2	z
3	x
9	a

Q: $\pi_D(S) \bowtie S$

A: 5

S.B	S.C	S.D	S.D_2
x	0	3	3
~~x~~	~~0~~	~~3~~	~~3~~
y	2	1	1
y	3	3	3
~~y~~	~~3~~	~~3~~	~~3~~
w	3	0	0
~~w~~	~~3~~	~~0~~	~~0~~
y	2	0	0
~~y~~	~~2~~	~~0~~	~~0~~

HW2 – Exercise 3

- VISITS(Drinker, Bar)
- SERVES(Bar, Wine)
- LIKES(Drinker, Wine)

- Q: Find the bars that serve a wine that Charlie likes
- A: $\pi_{BAR}(SERVES \bowtie \sigma_{(DRINKER='CHARLIE')}(LIKES))$

HW2 – Exercise 3

- VISITS(Drinker, Bar)
- SERVES(Bar, Wine)
- LIKES(Drinker, Wine)

- Q: Find the drinkers that visits at least a bar that serves a wine they like
- A: $\pi_{DRINKER}(VISITS \bowtie SERVES \bowtie LIKES)$

HW2 – Exercise 3

- VISITS(Drinker, Bar)
- SERVES(Bar, Wine)
- LIKES(Drinker, Wine)
- Q: Find the drinkers that visits only the bars that serve a wine they like (we assume that each drinker likes at least a wine and visits at least a bar)
- A: $\pi_{DRINKER}(VISITS) - \pi_{DRINKER}(VISITS - \pi_{DRINKER,BAR}(LIKES \bowtie SERVES))$
- Idea is we want to first find the drinkers that have visited a bar that serves a wine they do not like. Want to subtract that out from all the drinkers' visits
- Inner RHS is the bars that have at least one wine the drinkers likes
- Outer RHS, then, are the visits where the drinker didn't like any wine at the bar. Extract drinker from there. This is the list of drinkers that have visited a bar that serves wine they don't like.

HW2 – Exercise 3

- VISITS(Drinker, Bar)
- SERVES(Bar, Wine)
- LIKES(Drinker, Wine)

- Q: Find the drinkers that does not frequent any bar that serves a wine they like
- A:

$$(\pi_{DRINKER} VISITS) - (\pi_{DRINKER}(LIKES \bowtie SERVES \bowtie VISITS))$$

- Idea is we want to find the drinkers that have visited at least one bar that serves a wine they like. Then we subtract that from all the visits.

- RHS is from part2

More SQL Practice

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q: Find the names of all juniors (level = ‘JR’) who are enrolled in a class taught by ‘Bob’

More SQL Practice

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q: Find the names of all juniors (level = ‘JR’) who are enrolled in a class taught by ‘Bob’
- A:

```
SELECT DISTINCT S.Sname
    FROM Student S, Class C, Enrolled E, Faculty F
        WHERE S.snum = E.snum AND E.cname = C.name
          AND C.fid = F.fid AND f.name = ‘Bob’ AND S.level = ‘JR’
```

More SQL Practice

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q: Find the age of the oldest student who is either a History major or enrolled in a course taught by ‘Bob’

More SQL Practice

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q: Find the age of the oldest student who is either a History major or enrolled in a course taught by 'Bob'
- A:
```
SELECT MAX(S.age) FROM Student S
  WHERE (S.major = 'History')
     OR S.snum IN (SELECT E.snum FROM Class C, Enrolled E, Faculty F
                          WHERE E.cname = C.name AND C.fid = F.fid
                                AND f.name = 'Bob')
```

More SQL Practice

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q: Find the names of students enrolled in the maximum number of classes.

More SQL Practice

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q: Find the names of students enrolled in the maximum number of classes.
- A:

```
SELECT DISTINCT S.sname
  FROM Student S
  WHERE S.num IN (SELECT E.snum FROM Enrolled E
                  GROUP BY E.snum HAVING COUNT(*) >= ALL(SELECT COUNT(*)
                                                         FROM Enrolled E2
                                                         GROUP BY E2.snum))
```

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q1: Find the names of all classes that either meet in room R128 or have five or more students enrolled.
- Q2: Find the names of all students who are enrolled in two classes that meet at the same time
- Q3: For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

- Student(snum: integer, sname: string, major: string, level: string, age:integer
- Class(name: string, meets_at: time, room:string, fid: integer)
- Enrolled(snum: integer, cname: string)
- Faculty(fid: integer, fname: string, deptid: integer)

- Q1: Find the names of all classes that either meet in room R128 or have five or more students enrolled.
- Q2: Find the names of all students who are enrolled in two classes that meet at the same time
- Q3: For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).
  - Hint. Can use the form:
    - `SELECT S.age, S.level FROM Student S`
      `GROUP BY S.age, S.level HAVING S.level IN ( … correlated subquery …)`

```
A1: SELECT C.name
      FROM Class C
      WHERE C.room = 'R128'
        OR C.name IN (SELECT E.cname
                      FROM Enrolled E
                      GROUP BY E.cname HAVING COUNT(*) >= 5)
```

```
A2: SELECT DISTINCT S.sname
       FROM Student S
       WHERE S.snum IN (SELECT E1.snum
                        FROM Enrolled E1, Enrolled E2, Class C1, Class C2
                        WHERE E1.snum = E2.snum AND E1.cname <> E2.cname
                              AND E1.cname = C1.name
                              AND E2.cname = C2.name
                              AND C1.meets_at = C2.meets_at)
```

```
A3: SELECT S.age, S.level FROM Student S
      GROUP BY S.age, S.level HAVING S.level IN (
        SELECT S1.level
                    FROM Student S1
        WHERE S1.age = S.age
        GROUP BY S1.level, S1.age
        HAVING COUNT(*)  >= ALL(SELECT COUNT(*)
                               FROM Student S2 WHERE S1.age = S2.age GROUP BY S2.level, S2.age))
```