Quick content review

- can create a new table with **SELECT** statements

```
SELECT "Ben" AS first, "Bitdiddle" AS last;
```

↳ creates a new table with two columns: first & last and a single entry (first = Ben, last = Bitdiddle)

first	last
Ben	Bitdiddle

- can combine rows of 2 tables (if they have same # of cols) with **UNION**

```
SELECT "Ben" AS first, "Bitdiddle" AS last UNION
SELECT "Louis",        "Reasoner";
```

↳ combines these 2 rows to create larger table

first	last
Ben	Bitdiddle
Louis	Reasoner

- can save a table using **CREATE TABLE**

```
CREATE TABLE records AS
  SELECT "Ben" AS first, "Bitdiddle" AS last
```

↳ stores the table under the name 'records'

- can select specific values from an existing table using **FROM**

```
SELECT first FROM records
```

↳ gets the column 'first' from the table 'records'

- **SELECT *** will select all the columns from a table
  - ↳ good way to print the contents of a table

- to select specific columns from a table use **WHERE** and **ORDER BY**

```
SELECT [columns] FROM [tables] WHERE [condition] ORDER BY [criteria]
```

- all valid SQL statements must be terminated by a semicolon (;)

PRACTICE PROBLEMS

page 8 (#5.1) (warm-up/check-in)

5.1 **Tutorial:** This short question is meant to help refresh your memory of topics covered in lecture and lab this week before tackling more challenging problems.

Table A has 5 rows and 4 columns, and Table B has 3 rows and 2 columns. If we join table A and table B, how many rows will the resulting table contain?

Table A:

	col1	col2	col3	col4
row1				
row2				
row3				
row4				
row5				

Table B:

	col1	col2
row1		
row2		
row3		

A join yields all combinations of a row from table A and a row from table B.

row1, row1 row2, row1 row3, row1 row4, row1 row5, row1
row1, row2 row2, row2 row3, row2 row4, row2 row5, row2
row1, row3 row2, row3 row3, row3 row4, row3 row5, row3

since there are 15 possible combos of a row from A & a row from B, the resulting table will have 15 rows.

Page 8 (#5.1)

5.1 Create a table called `num_taught` that contains three columns: `professor`, the `course` they taught, and the number of `times` they taught each course.

Hint: For this problem, it may help to `GROUP BY` multiple columns. Multiple columns and full expressions can appear in the group by clause, and groups will be formed for every unique combination of values that result.

courses

Professor	Course	Semester
John DeNero	CS 61C	Sp20
John DeNero	CS 61A	Fa19
Dan Garcia	CS 61C	Sp19
John DeNero	CS 61A	Fa18
Dan Garcia	CS 10	Fa18
Josh Hug	CS 61B	Sp18
John DeNero	CS 61A	Sp18
John DeNero	CS 61A	Fa17
Paul Hilfinger	CS 61A	Fa17
Paul Hilfinger	CS 61A	Sp17
John DeNero	Data 8	Sp17
Josh Hug	CS 61B	Sp17
Satish Rao	CS 70	Sp17
Nicholas Weaver	CS 61C	Sp17
Gerald Friedland	CS 61C	Sp17
⋮	⋮	⋮

(#) will share collabedit link in chat
↳ please write answer/ideas there!

SOLUTION:

```
CREATE TABLE num_taught AS
    SELECT professor AS professor, course AS course,
    COUNT(*) AS times FROM courses GROUP BY professor, course;
```

- num_taught → name of the table
- professor AS professor, course AS course → which column to look at/what to do in 'courses' table to get correct cols for our table; name of columns in new table
- GROUP BY professor, course; → creates the aggregate table from which we select our cols (we group by professor & course so that we can call count(*) to get # of times a given prof, course pair appears in the table)

Page 8 (#5.2)

5.2 Write a query that outputs two professors and a course if they have taught that course the same number of times. You may use the num_taught table you created in the previous question.

→ SELECT professor AS professor, course AS course, COUNT(*) AS times FROM courses GROUP BY professor, course

professor	course	times
⋮	⋮	⋮

courses

Professor	Course	Semester
John DeNero	CS 61C	Sp20
John DeNero	CS 61A	Fa19
Dan Garcia	CS 61C	Sp19
John DeNero	CS 61A	Fa18
Dan Garcia	CS 10	Fa18
Josh Hug	CS 61B	Sp18
John DeNero	CS 61A	Sp18
John DeNero	CS 61A	Fa17
Paul Hilfinger	CS 61A	Fa17
Paul Hilfinger	CS 61A	Sp17
John DeNero	Data 8	Sp17
Josh Hug	CS 61B	Sp17
Satish Rao	CS 70	Sp17
Nicholas Weaver	CS 61C	Sp17
Gerald Friedland	CS 61C	Sp17
⋮	⋮	⋮

A: B:

professor	course	times
a b ⋮	⋮	⋮

professor	course	times
a b ⋮	⋮	⋮

(A) will share collabedit link in chat
↳ please write answer/ideas there!

SOLUTION:

```
SELECT a.professor, b.professor, a.course
    FROM num_taught AS a, num_taught AS b
    WHERE a.professor > b.professor
        AND a.course = b.course
        AND a.times = b.times;
```

→ these are the 3 things we want to output (2 professor names & a course)

→ this is how we create 2 instances of the num_taught table to compare professors

→ this makes sure that we only compare to other profs. below us in table (to avoid repeats)

conditions that must be true for us to satisfy the problem & thus return

Page 9 (#5.3)

5.3 Write a query that outputs two professors if they co-taught (taught the same course at the same time) the same course more than once.

courses

Professor	Course	Semester
John DeNero	CS 61C	Sp20
John DeNero	CS 61A	Fa19
Dan Garcia	CS 61C	Sp19
John DeNero	CS 61A	Fa18
Dan Garcia	CS 10	Fa18
Josh Hug	CS 61B	Sp18
John DeNero	CS 61A	Sp18
John DeNero	CS 61A	Fa17
Paul Hilfinger	CS 61A	Fa17
Paul Hilfinger	CS 61A	Sp17
John DeNero	Data 8	Sp17
Josh Hug	CS 61B	Sp17
Satish Rao	CS 70	Sp17
Nicholas Weaver	CS 61C	Sp17
Gerald Friedland	CS 61C	Sp17
⋮	⋮	⋮

@ will share collabedit link in chat
↳ please write answer/ideas there!

SOLUTION:

```
SELECT a.professor, b.professor
FROM courses AS a, courses AS b
WHERE a.professor < b.professor
  AND a.semester = b.semester
  AND a.course = b.course
GROUP BY a.course, a.professor, b.professor HAVING COUNT(*) > 1;
```

→ the things we want to output
→ creates two instances of courses table to be used
→ makes sure we don't have repeats
→ conditions to be satisfied (taught same course in same semester)
→ we are only interested in the (profA, profB, course) entries that have a count of more than one (keeps only the profs who co-taught a class for more than one semester)