CS102: Big Data

Tools and Techniques, Discoveries and Pitfalls

Spring 2017

Ethan Chan, Lisa Wang

Lecture 4: Relational Databases and SQL

Announcements

All Office Hours held in Lathrop Tech Lounge

Recap

A SQL query walks into a bar and sees two tables. He walks up to them and asks,

"Can I join you?"

SQL Query

- Basic form (there are many many more bells and whistles)

```
SELECT <attributes>
FROM   <one or more relations>
WHERE  <conditions>
```

Call this a SFW query.

Adapted from CS145: Databases

Selecting all Columns (Select *)

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT *
FROM   Product
WHERE  Category = 'Gadgets'
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

Adapted from CS145: Databases

Selecting specific Columns

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT Pname, Price, Manufacturer
FROM   Product
WHERE  Category = 'Gadgets'
```

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks

Adapted from CS145: Databases

Find all regions that have both coastal and non-coastal states

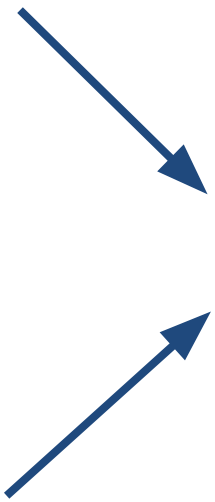
Cross Product : Select * from Regions R1, Regions R2

Region R1 (n rows)

City	Regions	Coastal
Maine	NE	Y
Vermont	NE	N
NYC	MA	Y
..	..	..
..	..	..

Region R2 (n rows)

City	Regions	Coastal
Maine	NE	Y
Vermont	NE	N
NYC	MA	Y
..	..	..
..	..	..



Result (n x n rows)

City	Regions	Coastal	City1	Regions 1	Coastal 1
Maine	NE	Y	Maine	NE	Y
Maine	NE	Y	Vermont	NE	N
Maine	NE	Y	NYC	MA	Y
Vermont	NE	N	Maine	NE	Y
Vermont	NE	N	Vermont	NE	N
Vermont	NE	N	NYC	MA	Y
..	..	..	..	..	..

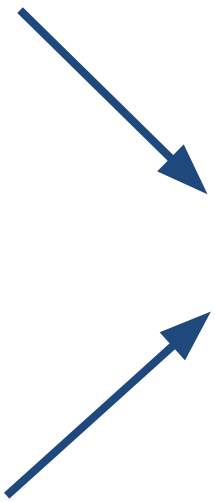
Query : Select * from Regions R1, Regions R2 Where R1.region = R2.region

Region R1 (n rows)

City	Regions	Coastal
Maine	NE	Y
Vermont	NE	N
NYC	MA	Y
..	..	..
..	..	..

Region R2 (n rows)

City	Regions	Coastal
Maine	NE	Y
Vermont	NE	N
NYC	MA	Y
..	..	..
..	..	..



Result

City	Regions	Coastal	City1	Regions 1	Coastal 1
Maine	NE	Y	Maine	NE	Y
Maine	NE	Y	Vermont	NE	N
Maine	NE	Y	NYC	MA	Y
Vermont	NE	N	Maine	NE	Y
Vermont	NE	N	Vermont	NE	N
Vermont	NE	N	NYC	MA	Y
..	..	..	..	..	..

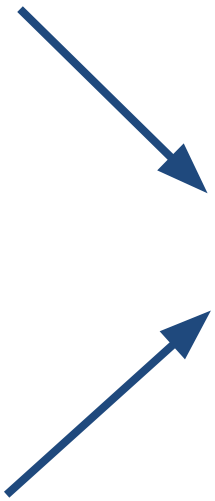
Query : Select * from Regions R1, Regions R2 Where R1.region = R2.region

Region R1 (n rows)

City	Regions	Coastal
Maine	NE	Y
Vermont	NE	N
NYC	MA	Y
..	..	..
..	..	..

Region R2 (n rows)

City	Regions	Coastal
Maine	NE	Y
Vermont	NE	N
NYC	MA	Y
..	..	..
..	..	..

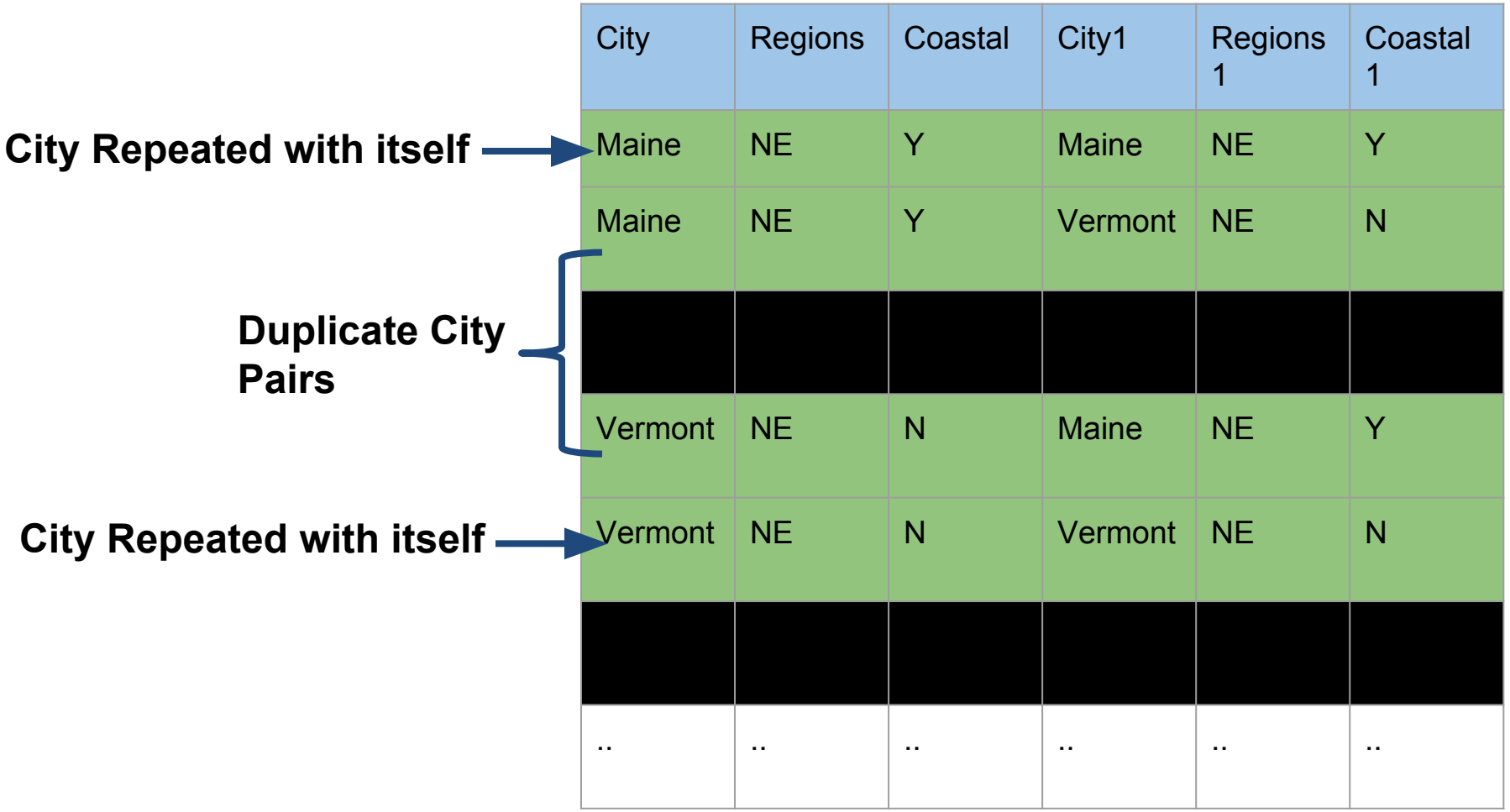


Result

City	Regions	Coastal	City1	Regions 1	Coastal 1
Maine	NE	Y	Maine	NE	Y
Maine	NE	Y	Vermont	NE	N
Vermont	NE	N	Maine	NE	Y
Vermont	NE	N	Vermont	NE	N
..	..	..	..	..	..

Query : Select * from Regions R1, Regions R2 Where R1.region = R2.region

Result



City	Regions	Coastal	City1	Regions 1	Coastal 1
Maine	NE	Y	Maine	NE	Y
Maine	NE	Y	Vermont	NE	N
Vermont	NE	N	Maine	NE	Y
Vermont	NE	N	Vermont	NE	N
..	..	..	..	..	..

Find all pairs of cities that are near each other, i.e., lat and lng are both less than 1.0 apart; return city pairs


McNeil
@Reflog_18
Follow
United Airlines is pleased to announce new seating on all domestic flights- in addition to United First and Economy Plus we introduce....
02 03 07 08 09 10 11 12 15 16 17 18 19 20 21 22 23 24
D
C
B
A
Fight Club
01 02 03 07 08 09 10 11 12 15 16 17 18 19 20 21 22 23 24
RETWEETS
36,464
LIKES
54,614
8:13 AM - 10 Apr 2017
© McNeil/Twitter

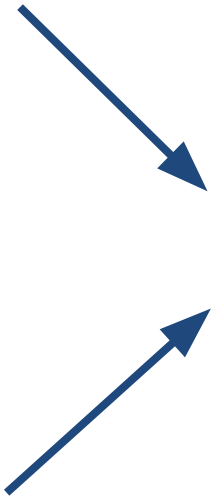
Query : Select * from CityTemps C1, CityTemps C2

City C1 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..

City C2 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..



Result (n x n rows)

City	Lat	Lng	City1	Lat	Lng
Mobile	31.2	88.5	Mobile	31.2	88.5
Mobile	31.2	88.5	Phoenix	33.6	112.5
Mobile	31.2	88.5	LA	34.3	118.7
Phoenix	33.6	112.5	Mobile	31.2	88.5
Phoenix	33.6	112.5	Phoenix	33.6	112.5
Phoenix	33.6	112.5	LA	34.3	118.7
..	..	..	..	..	..

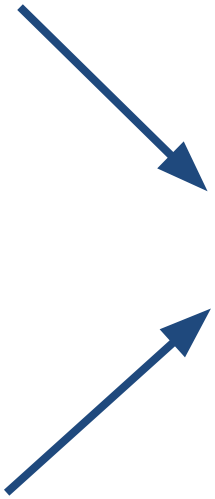
Query : Select * from CityTemps C1, CityTemps C2 Where C1.city = C2.city

City C1 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..

City C2 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..



Result (n x n rows)

City	Lat	Lng	City1	Lat	Lng
Mobile	31.2	88.5	Mobile	31.2	88.5
Mobile	31.2	88.5	Phoenix	33.6	112.5
Mobile	31.2	88.5	LA	34.3	118.7
Phoenix	33.6	112.5	Mobile	31.2	88.5
Phoenix	33.6	112.5	Phoenix	33.6	112.5
Phoenix	33.6	112.5	LA	34.3	118.7
..	..	..	..	..	..

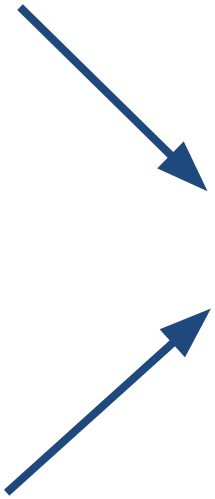
Query : Select * from CityTemps C1, CityTemps C2 Where C1.city = C2.city

City C1 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..

City C2 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..



Result (n x n rows)

City	Lat	Lng	City1	Lat	Lng
Mobile	31.2	88.5	Mobile	31.2	88.5
Phoenix	33.6	112.5	Phoenix	33.6	112.5
..	..	..	..	..	..

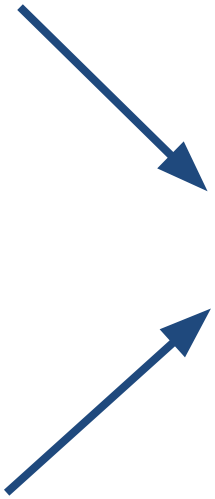
Query : Select * from CityTemps C1, CityTemps C2
Where C1.city <> C2.city

City C1 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..

City C2 (n rows)

City	Lat	Lng
Mobile	31.2	88.5
Phoenix	33.6	112.5
LA	34.3	118.7
..	..	..
..	..	..



Result (n x n rows)

City	Lat	Lng	City1	Lat	Lng
Mobile	31.2	88.5	Phoenix	33.6	112.5
Mobile	31.2	88.5	LA	34.3	118.7
Phoenix	33.6	112.5	Mobile	31.2	88.5
Phoenix	33.6	112.5	LA	34.3	118.7
..	..	..	..	..	..

Query : Select * from CityTemps C1, CityTemps C2

Where C1.city <> C2.city

Result (n x n rows)

City	Lat	Lng	City1	Lat	Lng
Mobile	31.2	88.5	Phoenix	33.6	112.5
Mobile	31.2	88.5	LA	34.3	118.7
Phoenix	33.6	112.5	Mobile	31.2	88.5
Phoenix	33.6	112.5	LA	34.3	118.7
..	..	..	..	..	..

Duplicate pair

Query : Select * from CityTemps C1, CityTemps C2
Where C1.city < C2.city

Result (n x n rows)

Ensures that the left City's alphabet is always lexographically less than the right City's alphabet

City	Lat	Lng	City1	Lat	Lng
Mobile	31.2	88.5	**P**hoenix	33.6	112.5
..	..	..	..	..	..

Subqueries in Where Clause (Not Exists)

Find the southernmost city

The Exists function

EXISTS returns TRUE

- When the query returns 1 or more rows
- Else FALSE

Example

EXISTS (Select * from CityTemps where lat > 99999)

- Returns FALSE (no cities have such high latitudes)

EXISTS (Select * from CityTemps where lat > 0)

- Returns TRUE

Adding a NOT before EXISTS will negate it

Find the southernmost city (1)

Note: the higher the latitude, the further north a city is

```
Select city
From CityTemps C1
Where Not Exists
    (Select * from CityTemps C2 where C1.lat > C2.lat)
```

"Select the city where there does not exist another city that is more south than it."

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

City C1

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

City C2

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

City C1

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

City C2

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

EXISTS returns TRUE

- When the query returns 1 or more rows
- Else FALSE

City C1

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

33.6 > 25

↓

Subquery returns [Key West, 25] row

↓

EXISTS (...) = TRUE

↓

NOT EXISTS (...) = FALSE

City C2

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

City C1

City	Lat
Key West	25
LA	34.3
..	..
..	..

33.6 > 25

↓

Subquery returns [Key West, 25] row

↓

EXISTS (...) = TRUE

↓

NOT EXISTS (...) = FALSE

↓

Phoenix won't be returned

City C2

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

City C1

City	Lat
Key West	25
LA	34.3
..	..
..	..

No rows returned from subquery

City C2

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

City C1

City	Lat
Key West	25
LA	34.3
..	..
..	..

No rows returned from subquery

↓

EXISTS (...) = FALSE

↓

NOT EXISTS (...) = TRUE

↓

Main query returns [Key West] row

City C2

City	Lat
..	..
..	..

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

City C1

City	Lat
Key West	25
..	..
..	..

34.3 > 25

↓

Subquery returns [Key West, 25] row

↓

EXISTS (...) = TRUE

↓

NOT EXISTS (...) = FALSE

↓

LA is not returned

City C2

City	Lat
Phoenix	33.6
Key West	25
LA	34.3
..	..
..	..

```
Select city
From CityTemps C1
Where Not Exists (Select *
                  from
                  CityTemps C2
                  where C1.lat > C2.lat)
```

City C1

City	Lat
Key West	25
..	..
..	..

We now have the southernmost city!

"Select the city where there does not exist another city that is more south than it."

Find the southernmost city (2)

Note: the higher the latitude, the further north a city is

```
Select city
From CityTemps C1
Where C1.lat = (Select min(lat) from CityTemps C2)
```

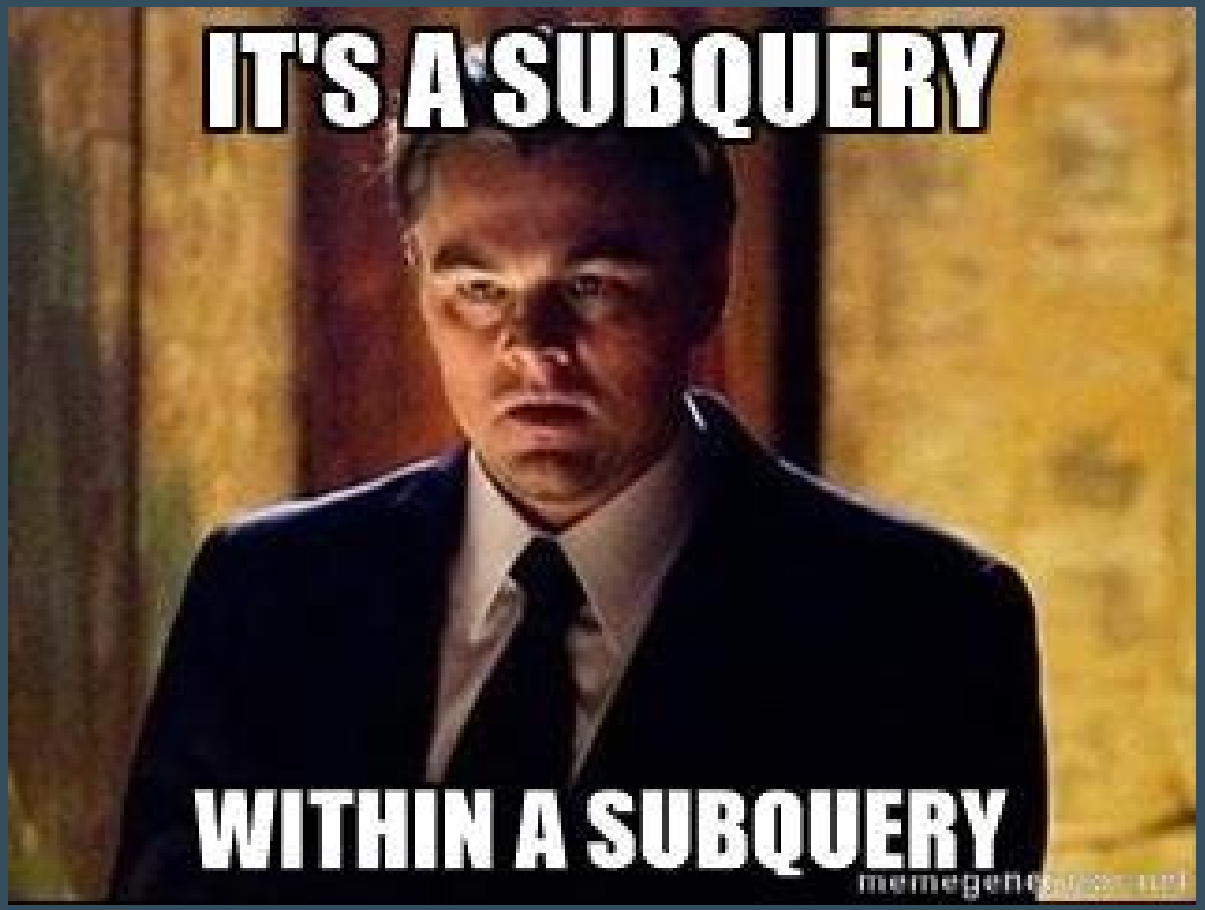
"Select the city that equals to the minimum latitude"

Find the southernmost city (wrong)

Note: the higher the latitude, the further north a city is

```
Select city
From CityTemps C1
Where C1.lat = min(lat)
```

DOES NOT WORK!!


IT'S A SUBQUERY
WITHIN A SUBQUERY

Advanced: Having Clause

Advanced: Having Clause

Find all states with at least three cities with temp > 30

```
Select state
   From CityTemps
   Where temp > 30
   Group By state
   Having count(*) >= 3
```

WHERE is a condition on CityTemps

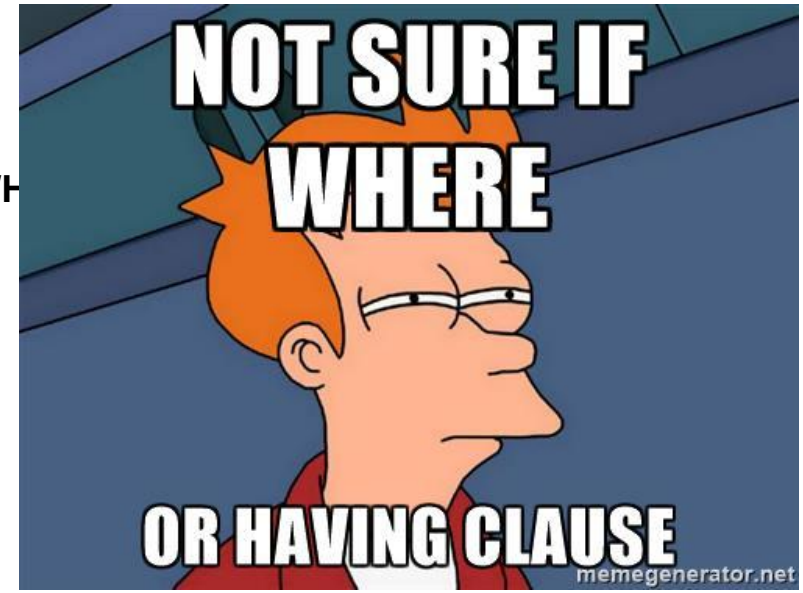
HAVING is a condition on the Group By

Advanced: Having Clause

Find all states with at least three cities with temp > 30

```
Select state
   From CityTemps
   Where temp > 30
   Group By state
   Having count(*) >= 3
```

WH



Advanced: Subqueries in Select clause

Find all regions that have both coastal and non-coastal states; return region and number of coastal and non-coastal states

```
Select Distinct R1.region,

  (Select count(*) From Regions R3
   Where R3.region = R1.region And R3.coastal = 'Y') as numcoastal,

  (Select count(*) From Regions R3
   Where R3.region = R1.region And R3.coastal = 'N') as numnot

From Regions R1, Regions R2

Where R1.coastal = 'Y' And R2.coastal = 'N' And R1.region=R2.region
```

Advanced: Subqueries in From clause

Find all regions that have both coastal and non-coastal states; return region and number of coastal and non-coastal states

```
Select C.region, numcoastal, numnot

From (Select region, count(*) as numcoastal
      From Regions Where coastal = 'Y' Group By region) C,

     (Select region, count(*) as numnot
      From Regions Where coastal = 'N' Group By region) NC

Where C.region = NC.region And numcoastal > 0 And numnot > 0
```

SQL Features not Covered

- Set Operators
  - Union, Intersect, Except
- Keys
  - Designated column that must have unique value in each row
  - Or designated set of columns
- Null values
  - Special value usually denoting unknown or undefined
  - Not included in aggregations, =, <, etc.
  - Example: ... where temp <= 10 or temp > 10
- Outer joins