BZAN 6354

Lecture 10

March 25, 2024

Dr. Mark Grimes, Ph.D.
gmgrimes@bauer.uh.edu

UNIVERSITY of HOUSTON
C. T. BAUER COLLEGE of BUSINESS
Department of Decision & Information Sciences

Agenda

- Administration
  - Assignment 3
  - SQL Project
- Quick Overview of last week
  - Unary SQL Queries
- Assignment 3 Demo
- Module 12.2: Binary SQL Queries
- Break
- Module 12.3: SQL Subqueries

Assignment 3

- A3 has been posted
  - You need the content in modules 12.1 – 12.2
  - **As with the other assignments there is a walkthrough video**

- Due April 1

- Ten SQL queries to write in Access
  - **If you do not have access** you can find the same tables in Oracle, but you will not be able to do the insert and update queries – just write the needed queries in the file you submit.
  - You could also use the computers in the Melcher computer lab on the second floor

- You must write the SQL
  - Don’t use the “Query by example” wizard

- We’ll do a demo today

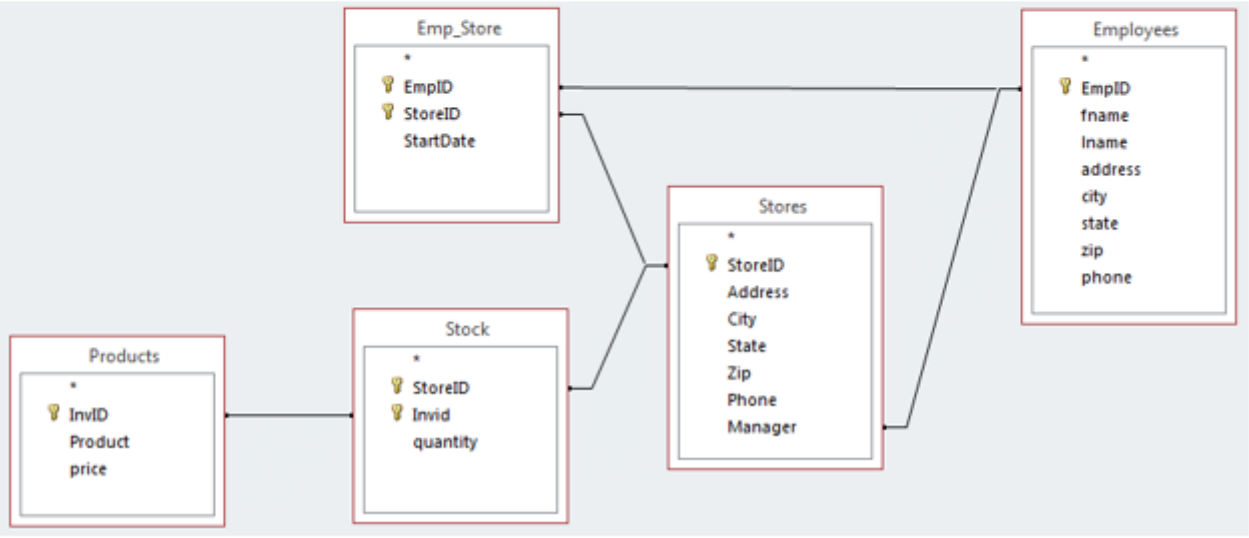
Assignment 3

- A3 has been posted
  - You need the content in modules 12.1 – 12.2
  - **As with the other assignments there is a walkthrough video**

Assignment 3: Using Databases

In this assignment, you will run basic SQL queries to query and update store, product, and employee information.

Attempt this assignment yourself, then when you feel either **Lost** or **Successful**, watch this walkthrough video and correct any mistakes you might have made: https://bit.ly/DB1-A3



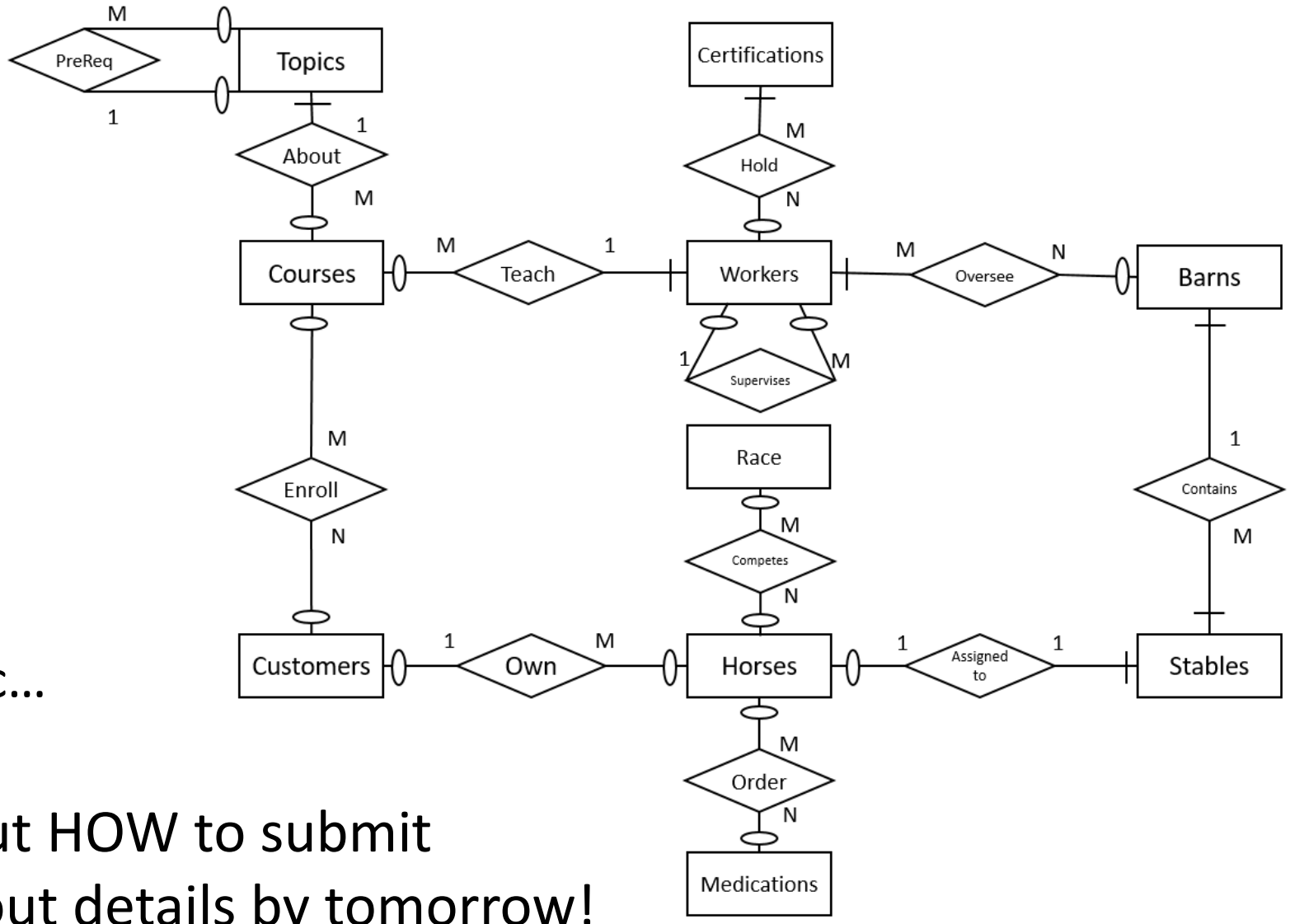
The answers and HOW to come up with the answers are in this video

This link is in the assignment file

If you want additional feedback on your submission please let me know!

SQL Project

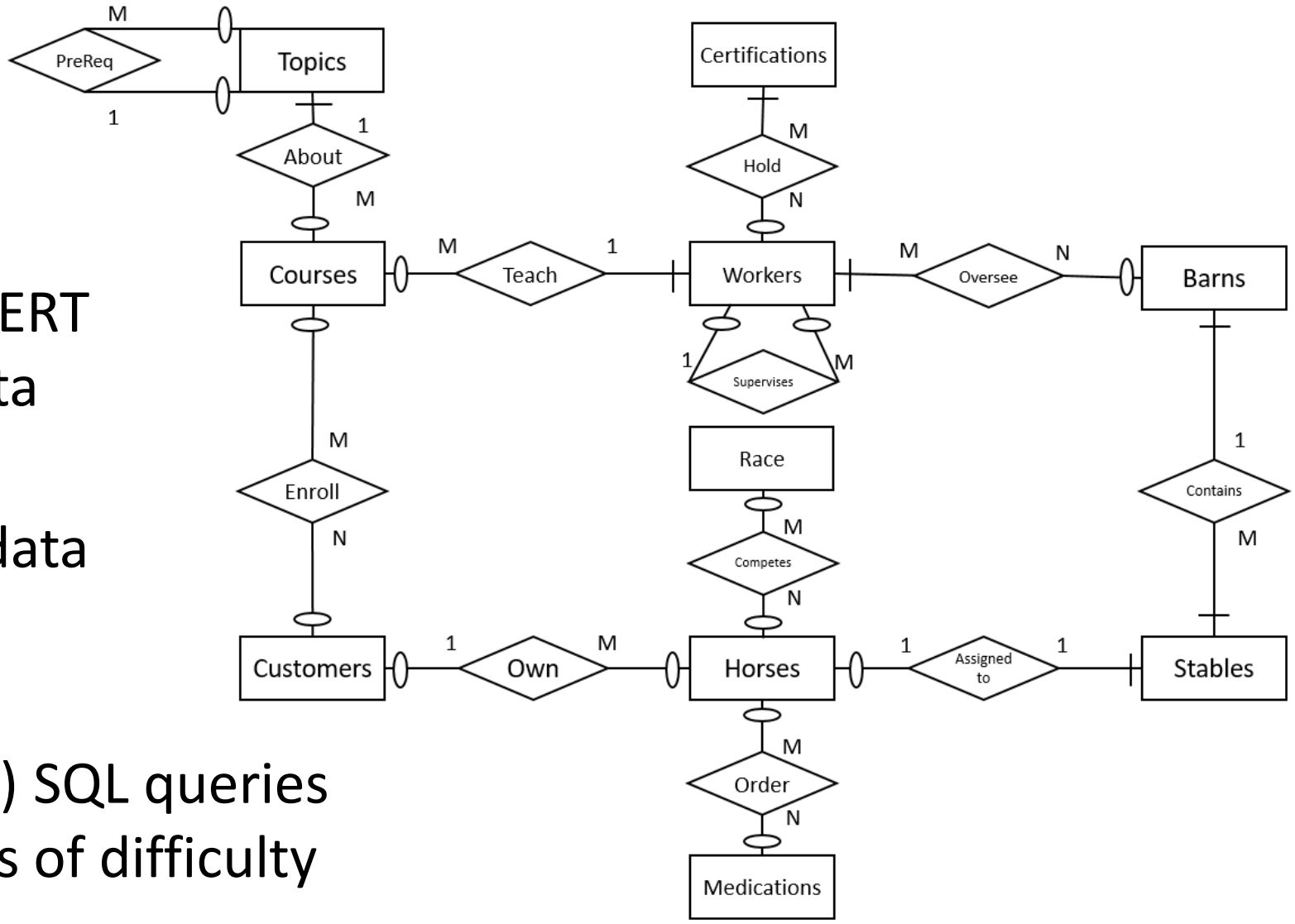
- The database for the SQL project is compete!
  - 15 Tables
  - 163 Horses
  - 55 Customers
  - 205 Stables
  - 6 Barns
  - 20 Workers
  - Certifications, courses, Etc…



- I am finalizing details about HOW to submit your answers – will send out details by tomorrow!

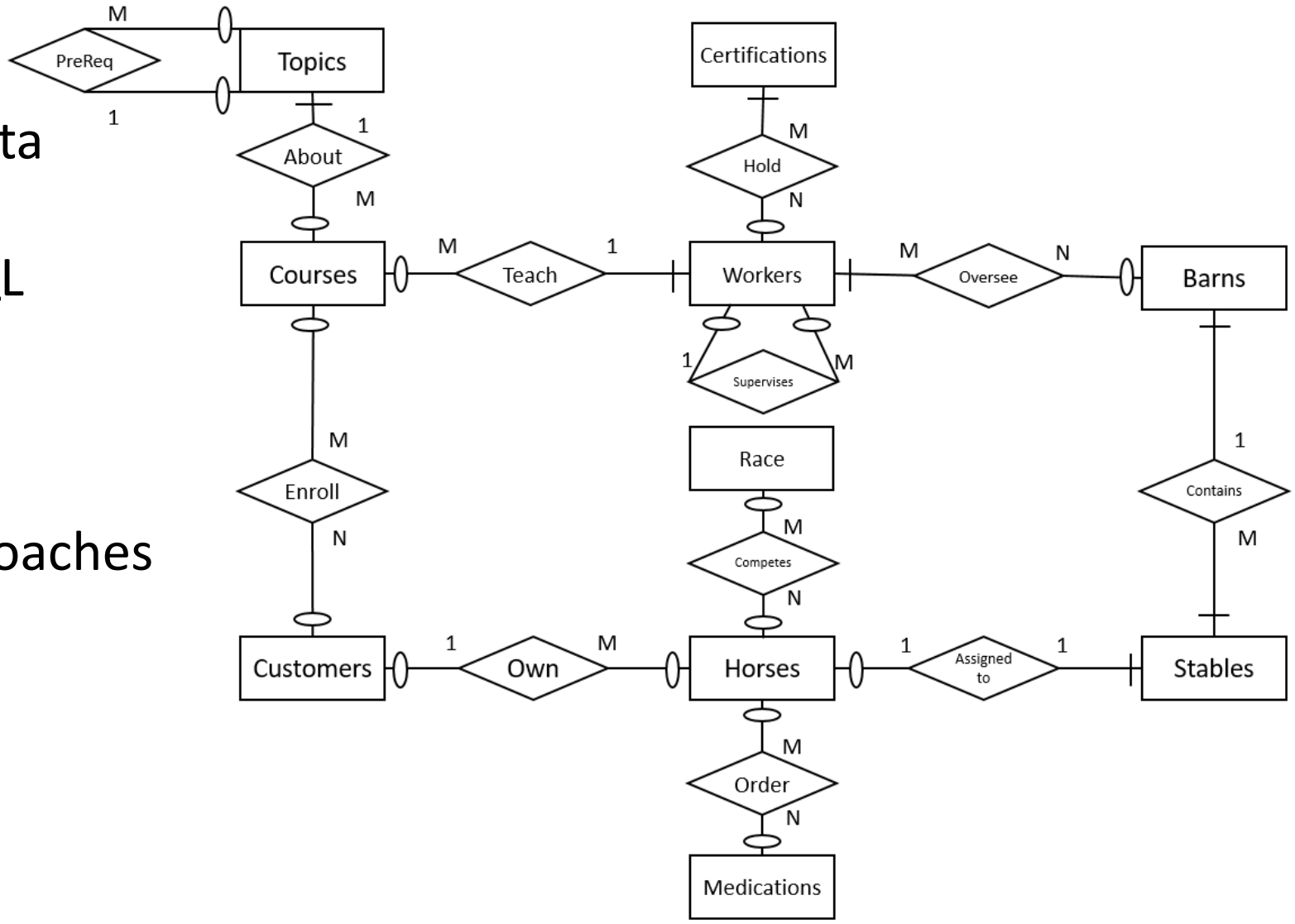
SQL Project

- I will provide you with the DDL to create this →
- I will provide you with INSERT statements to load the data
- You will add a little bit of data about yourself
- You will have many (40-50) SQL queries to write with varying levels of difficulty



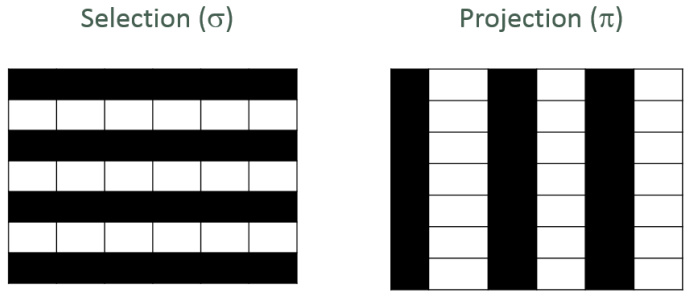
SQL Project

- Grading outline (subject to change)
  - 10%: Loading provided data
  - 10%: Inserting new data
  - 20%: Basic single table SQL
  - 20%: Aggregate Functions
  - 20%: Binary Queries
  - 10%: String Manipulation
  - 10%: Using Multiple Approaches
  - ------
  - 100%
  - +10%: Bonus Queries
  - -----
  - 110% total possible score



Last time we talked about unary operators

- Select: $\sigma_{\text{<section condition>}}R$
- Project: $\pi_{\text{<attribute list>}}R$



- The basic “Select-From-Where” block:

SELECT <attributes>
FROM <table>
WHERE <condition>

SELECT * FROM students;

SELECT empid, fname, lname FROM employees WHERE state=‘TX’;

Comparison operations

- Simple ones:
  - >, <, >=, <=, <>, =
    - SELECT * FROM employees WHERE salary > 50000;

- The LIKE operator allows for wildcards
  - % for many characters (* in MS Access)
    - SELECT * FROM employees WHERE lname LIKE ‘G%’;
  - _ for a single character (? In MS Access)
    - SELECT * FROM employees WHERE fname LIKE ‘_I_A’;

- The IN operator is used to identify values in a list or a sub-query
  - SELECT * FROM employees WHERE lname IN (‘Smith’, ‘Jones’, ‘Han’);

- All can be negated with NOT

NULL Values

- NULL is the absence of any value at all
  - Doesn’t make sense to use normal comparison operators
  - Must use the IS operator:

```
SELECT * FROM employees WHERE fired_date IS NULL;

SELECT * FROM employees WHERE fired_date IS NOT NULL;
```

Additions to select-from-where

- Group by
  - Groups row based on a value of a particular attribute
  - Necessary for aggregate functions (count, min, max, avg, etc…)
    - SELECT college, AVG(hrs) FROM course
      GROUP BY college;

- Having
  - Similar to WHERE, but used for aggregate functions
    - SELECT college, AVG(hrs) FROM course
      GROUP BY college HAVING AVG(hrs) > 3;

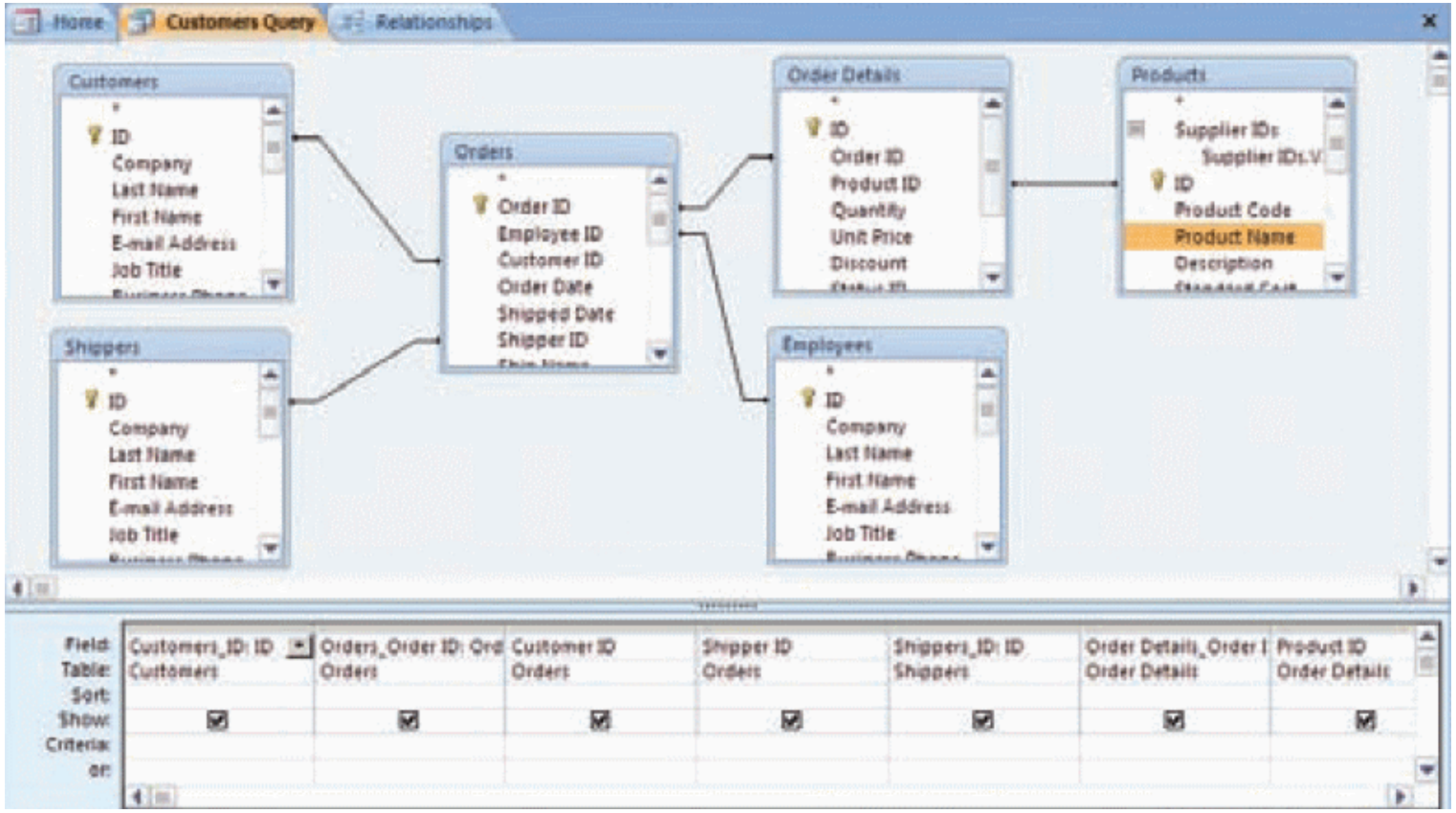
- Order by
  - Changes the order in which tuples are returned
    - SELECT college, AVG(hrs) FROM course
      GROUP BY college HAVING AVG(hrs) > 3 ORDER BY AVG(hrs);

Assignment 3 Demo

- If you have Access, feel free to follow along – you can download the .aacdb file from the content section on Blackboard - it is the same file you will use for assignment 3

Query By Example

- QBE provides a graphical interface to define what information you want to see.
- More “user friendly” but far less powerful than writing in SQL!



Access and SQL

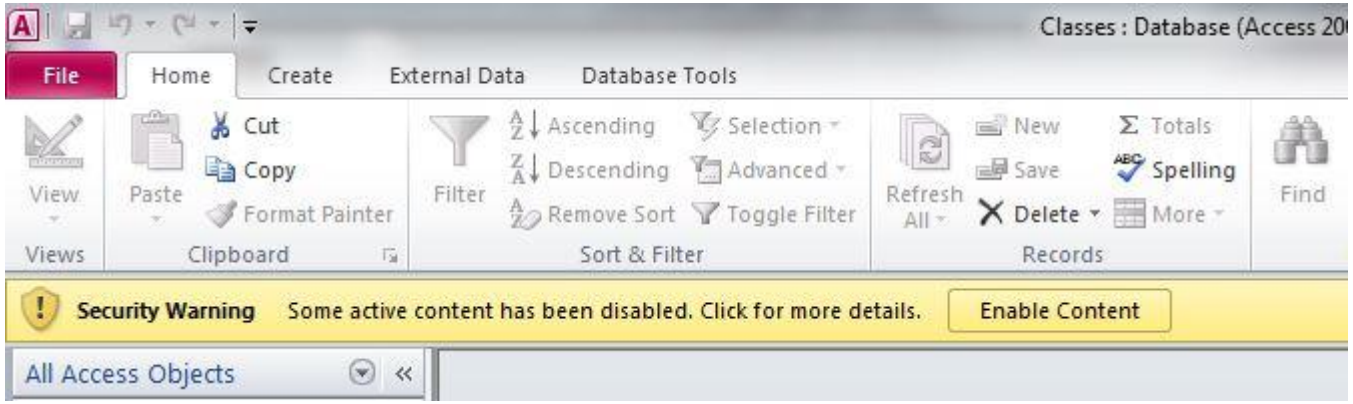
- A few things to note:
  - Wildcard characters are different in Access and other DBMS
    - Access uses * and ?
    - Everyone else uses % and _
  - Do not copy and paste from this presentation - the quote characters get messed up and the queries will not work. Better to retype rather than cut and paste

Access and SQL

Open the database

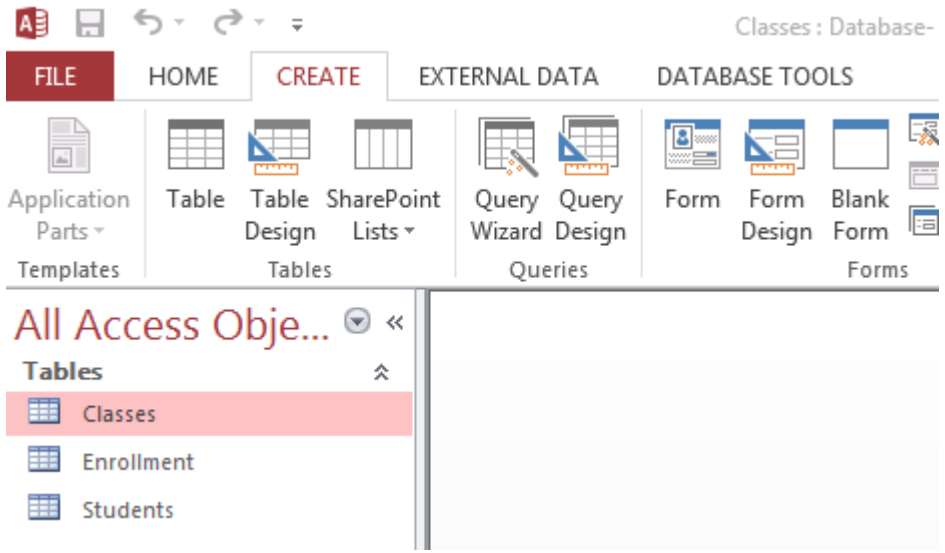
MDB (older format) or ACCDB (newer format) file

If you get a message like this, click Enable Content!



Access and SQL

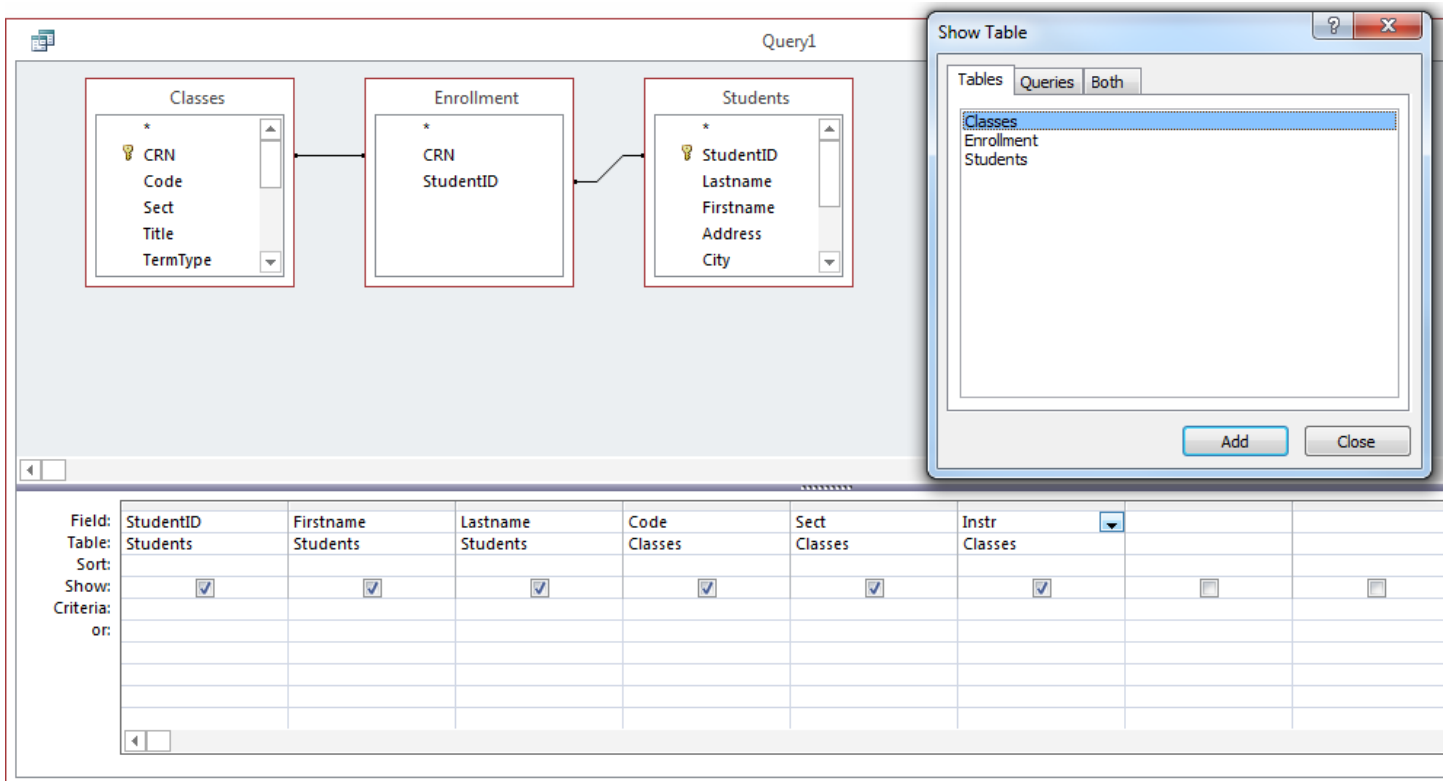
- You can look at the data by double clicking the table (for example, the Classes table)



- To write SQL queries, click the Create tab, then click query design

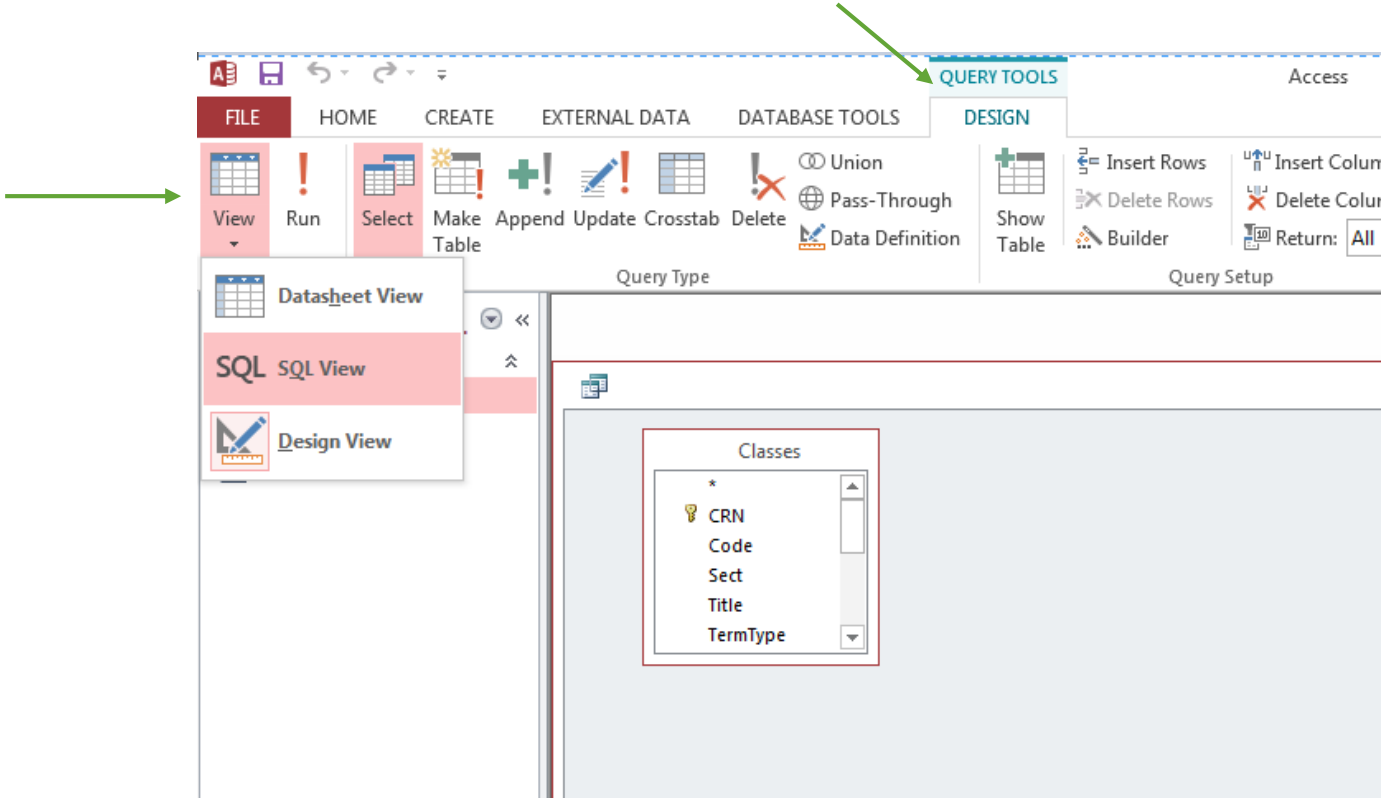
Access and SQL

- Access has a Query By Example (QBE) system where you can drag and drop elements to create queries – don't use this.



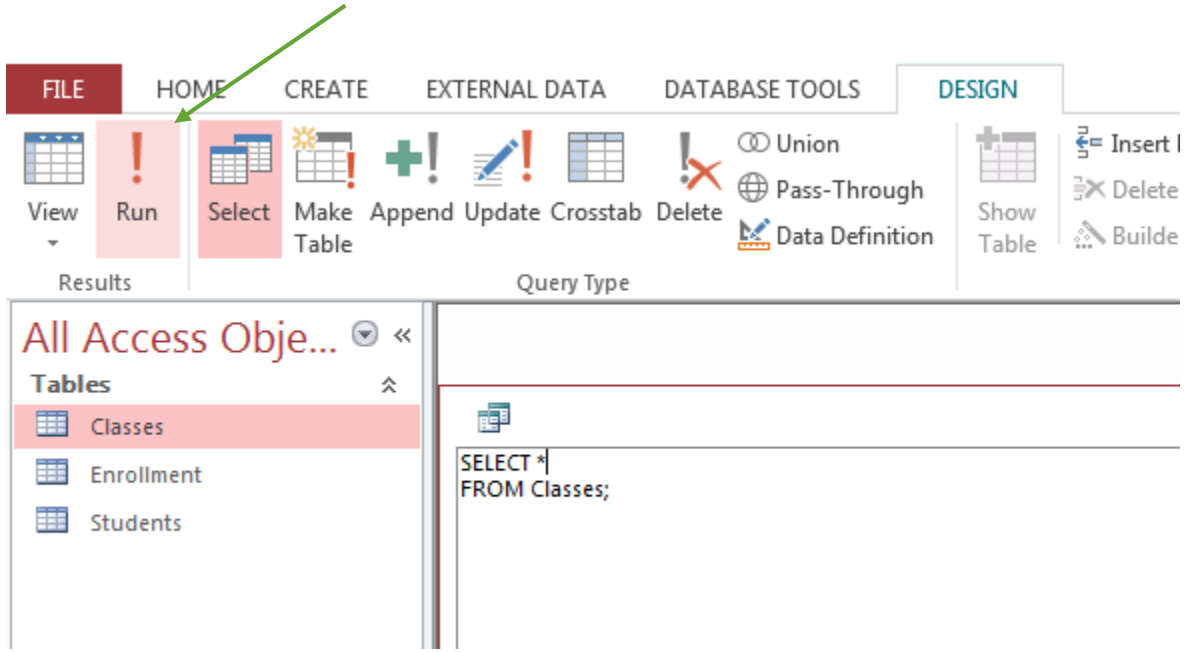
Access and SQL

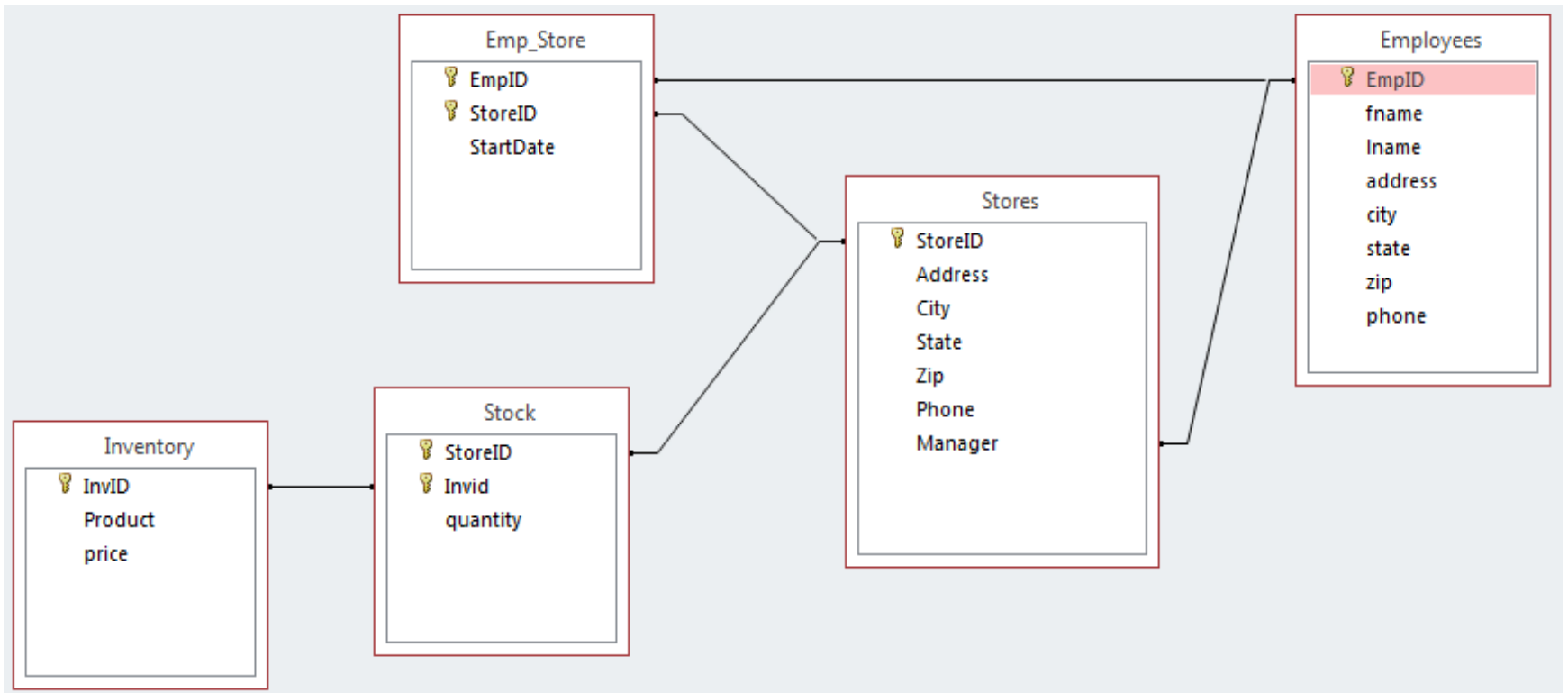
- Change to the more powerful SQL view by clicking the small arrow at the bottom of the view button in the top left (make sure you are on the DESIGN tab)



Access and SQL

- Now we can type in SQL queries and hit Run to execute them




Emp_Store
EmpID
StoreID
StartDate
Employees
EmpID
fname
lname
address
city
state
zip
phone
Stores
StoreID
Address
City
State
Zip
Phone
Manager
Stock
StoreID
Invid
quantity
Inventory
InvID
Product
price

- Select all attributes for all employees and order them alphabetically by last name:

- SELECT * FROM Employees ORDER BY lname

EmpID	fname	lname	address	city	state	zip	phone
1016	Lacy	Acosta	953-8683 Nec, S	Oxford	MS	38655	555-821-9800
1005	Clarke	Delgado	174-7411 Suspe	Cordova	TN	38103	555-764-4532
1017	Bert	Frye	Ap #207-8835 N	Oxford	MS	38655	555-507-9610
1007	Tina	Goff	1583 Augue St.	Marana	AZ	85373	555-843-7095
1004	Clayton	Hale	434 Amet St.	Memphis	TN	38103	555-499-9374
1001	Jack	Hatfield	395-6365 Aliqu	Nashville	TN	37212	555-980-5688
1013	Bert	Hensley	P.O. Box 984, 5	Tupelo	MS	38804	555-619-8184
1014	Troy	Kane	578-7454 Egest	Tupelo	MS	38804	555-228-5267
1015	David	Landry	Ap #551-6192 R	Oxford	MS	38655	555-895-7552
1009	Brandon	Leon	6109 Elementu	Tucson	AZ	85373	555-248-9721
1006	Kermit	Levine	P.O. Box 283, 4	Memphis	TN	38103	555-434-9263
1011	Castor	Mcknight	Ap #575-8716 A	Tupelo	MS	38804	555-570-9445
1008	Seth	Mercer	8928 Quisque S	Tucson	AZ	85373	555-799-6954
1003	Jesse	Noel	773 Id Rd.	Nashville	TN	37212	555-436-7282
1002	Duncan	Oneal	5344 Netus Rd.	Nashville	TN	37212	555-656-6212
1010	Wyatt	Talley	8251 Sed Aven	Oro Valley	AZ	85373	555-317-7170
1012	Brett	Weber	7864 Metus. Rd	Tupelo	MS	38804	555-575-9659

- Select all attributes for all employees whose last name starts with M

- SELECT * FROM Employees WHERE lname LIKE ‘m*’

EmpID	fname	lname	address	city	state	zip	phone
1008	Seth	Mercer	8928 Quisque S	Tucson	AZ	85373	555-799-6954
1011	Castor	Mcknight	Ap #575-8716 A	Tupelo	MS	38804	555-570-9445
0							

- Select the first name, last name, city, state and phone number for all employees living in Tennessee (TN), ordered by city then by last name.

- SELECT fname, lname, city, state, phone FROM employees
  WHERE state = ' TN'
  ORDER BY city, lname

fname	lname	city	state	phone
Clarke	Delgado	Cordova	TN	555-764-4532
Clayton	Hale	Memphis	TN	555-499-9374
Kermit	Levine	Memphis	TN	555-434-9263
Jack	Hatfield	Nashville	TN	555-980-5688
Jesse	Noel	Nashville	TN	555-436-7282
Duncan	Oneal	Nashville	TN	555-656-6212
*				

- Select all attributes about stores for stores that are in TN or AZ
- SELECT * FROM stores WHERE state in ('TN', 'AZ')

Or

- SELECT * FROM stores WHERE state = 'TN' or state='AZ'

StoreID	Address	City	State	Zip	Phone	Manager
1	1718 Belmont A	Nashville	TN	37212	615-555-1234	1002
2	125 South Mair	Memphis	TN	38103	901-555-6363	1004
3	325 Oracle	Tucson	AZ	85737	520-444-2355	1009
0						0

- Display all attributes about products with a price between $10 and $50 in order by price from high to low

- SELECT *
  FROM products
  WHERE price BETWEEN 10 AND 50
  ORDER BY price DESC

Alternatively:

- SELECT *
  FROM products
  WHERE price >= 10 AND price <= 50
  ORDER BY price DESC

InvID	Product	price
118	Coffee Pot	45
104	Hat Rack	42
101	Lava Lamp	40
100	Floor Lamp	35
108	Rice Cooker	34
121	Water Filter	22
105	Ironing Board	22
114	Skillet	15
102	Magazine Rack	15
116	Clock	14
124	Sunglasses	11
* 0		0

- Display all attributes about products with a price that is NOT between $10 and $50 in order by price from high to low

- SELECT *
  FROM products
  WHERE price NOT BETWEEN 10 AND 50
  ORDER BY price DESC

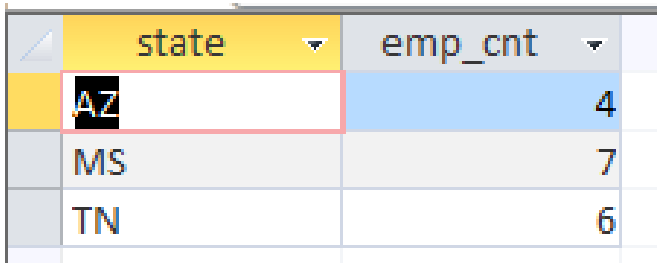
Alternatively:

- SELECT *
  FROM products
  WHERE price < 10 **OR** price > 50
  ORDER BY price DESC

InvID	Product	price
111	Digital Camera	240
113	Monitor	140
106	Knife Set	125
117	Radio	86
112	Watch	60
107	DVD Player	60
103	Throw Rug	60
120	Picture Frame	9
125	Book light	8
109	Dog Water Bow	7
119	Stapler	6
115	Coffee Mug	5
123	Dust pan	4
110	Fly Swatter	1
*	0	0

- Display how many employees are in each state. Name the column with the count “emp_cnt”

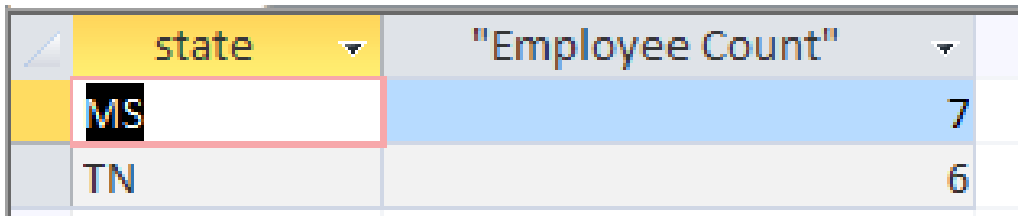
- SELECT state, count(*) as emp_cnt
  FROM employees
  GROUP BY state



state	emp_cnt
AZ	4
MS	7
TN	6

- Display how many employees are in each state that has over five employees. Name the column with the count “Employee Count”

- SELECT state, count(*) as “Employee Count”
  FROM employees
  GROUP BY state
  HAVING count(*) > 5



state	"Employee Count"
MS	7
TN	6

Module 12.2
Binary SQL operations

- Cartesian product (X)
- Joins
  - Equijoin (⋈)
  - Natural Join (*)
  - Left outer join (⟕)
  - Right outer join (⟖)
  - Full outer join (⟗)
- Set Theory Operators
  - Union (∪)
  - Intersection (∩)
  - Difference (-)

Some syntax does NOT work in MS Access – I have denoted this with the following symbol where I know there are issues, but Google is your friend when things don't work as expected

Quick note on table aliases

- Just like we can use the “AS” operator to give an attribute a different name…
  - SELECT college, count(*) AS clg_count FROM courses GROUP BY college;

- …we can follow a table name with an “alias” to make it easier to refer to it in other parts of the query

  - SELECT * FROM course C, department D
    WHERE C.dcode = D.dcode;

<u>exactly the same as doing:</u>

  - SELECT * FROM course, department
    WHERE course.dcode = department.dcode;

Reminder about our data for these examples:

- Note: For the upcoming examples we have created two new (simplified) relations using the project operation
  - H = $\pi_{(Name, Color, Weight, Owner)}$Horses
  - C = $\pi_{(Username, Fname, Lname)}$Customers

  - I have also removed "ssimpson" as the owner for Shamrock – If you want to follow along you can update your table with:

```
UPDATE Horses SET owner=NULL where Name='Shamrock';
```

- H has 14 tuples
- C has 5 tuples

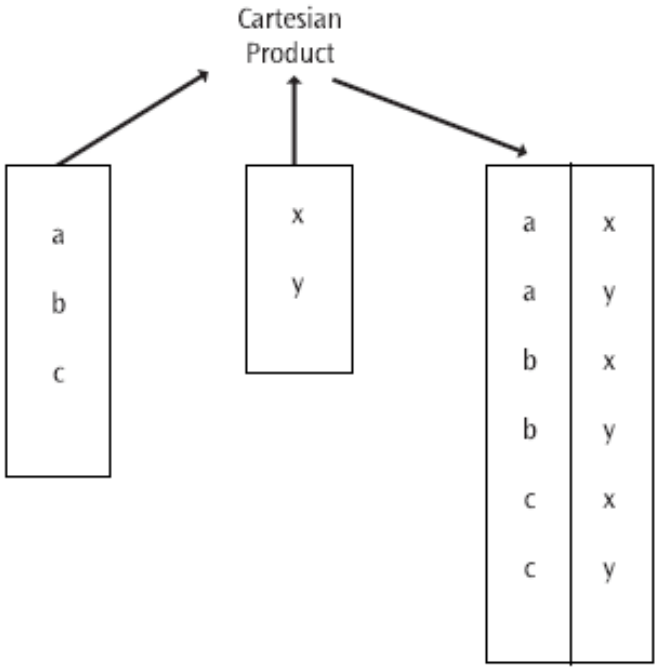
- How many tuples will the Cartesian product (H X C) have?

	NAME	COLOR	WEIGHT	OWNER
1	Sam	Brown	1,500	mgrimes
2	Erica	Yellow	920	canderson
3	John	Grey	1,800	mgrimes
4	Trotty	Brown	1,300	mgrimes
5	Rio	Grey	1,700	tswift
6	Robin	Yellow	1,100	jisbell
7	Katy	Brown	1,200	jisbell
8	Pegasus	Brown	1,750	mgrimes
9	Sammy	Black	2,200	mgrimes
10	Pinky	Red	1,050	tswift
11	Hulk	Grey	2,050	mgrimes
12	Pat	White	1,400	mgrimes
13	Betty	White	1,250	tswift
14	Shamrock	Black	1,400	[NULL]

	USERNAME	FNAME	LNAME
1	mgrimes	Marvin	Grimes
2	canderson	Christine	Anderson
3	tswift	Tina	Swift
4	jisbell	Jason	Isbell
5	ssimpson	Sam	Simpson

Cartesian Product

- All combinations of tuples from two relations
  - Relational Algebra Syntax: R1 X R2
- As we mentioned previously – not very useful by itself



Cartesian Product

- σ H X C
- SELECT * FROM H CROSS JOIN C;

	NAME	COLOR	WEIGHT	OWNER
1	Sam	Brown	1,500	mgrimes
2	Erica	Yellow	920	canderson
3	John	Grey	1,800	mgrimes
4	Trotty	Brown	1,300	mgrimes
5	Rio	Grey	1,700	tswift
6	Robin	Yellow	1,100	jisbell
7	Katy	Brown	1,200	jisbell
8	Pegasus	Brown	1,750	mgrimes
9	Sammy	Black	2,200	mgrimes
10	Pinky	Red	1,050	tswift
11	Hulk	Grey	2,050	mgrimes
12	Pat	White	1,400	mgrimes
13	Betty	White	1,250	tswift
14	Shamrock	Black	1,400	[NULL]

	USERNAME	FNAME	LNAME
1	mgrimes	Marvin	Grimes
2	canderson	Christine	Anderson
3	tswift	Tina	Swift
4	jisbell	Jason	Isbell
5	ssimpson	Sam	Simpson

	NAME	COLOR	WEIGHT	OWNER	USERNAME	FNAME	LNAME
1	Sam	Brown	1,500	mgrimes	mgrimes	Marvin	Grimes
2	Erica	Yellow	920	canderson	mgrimes	Marvin	Grimes
3	John	Grey	1,800	mgrimes	mgrimes	Marvin	Grimes
4	Trotty	Brown	1,300	mgrimes	mgrimes	Marvin	Grimes
5	Rio	Grey	1,700	tswift	mgrimes	Marvin	Grimes
6	Robin	Yellow	1,100	jisbell	mgrimes	Marvin	Grimes
7	Katy	Brown	1,200	jisbell	mgrimes	Marvin	Grimes
8	Pegasus	Brown	1,750	mgrimes	mgrimes	Marvin	Grimes
9	Sammy	Black	2,200	mgrimes	mgrimes	Marvin	Grimes
10	Pinky	Red	1,050	tswift	mgrimes	Marvin	Grimes
11	Hulk	Grey	2,050	mgrimes	mgrimes	Marvin	Grimes
12	Pat	White	1,400	mgrimes	mgrimes	Marvin	Grimes
13	Betty	White	1,250	tswift	mgrimes	Marvin	Grimes
14	Shamrock	Black	1,400	[NULL]	mgrimes	Marvin	Grimes
15	Sam	Brown	1,500	mgrimes	canderson	Christine	Anderson
16	Erica	Yellow	920	canderson	canderson	Christine	Anderson
17	John	Grey	1,800	mgrimes	canderson	Christine	Anderson
18	Trotty	Brown	1,300	mgrimes	canderson	Christine	Anderson
19	Rio	Grey	1,700	tswift	canderson	Christine	Anderson
20	Robin	Yellow	1,100	jisbell	canderson	Christine	Anderson
21	Katy	Brown	1,200	jisbell	canderson	Christine	Anderson
22	Pegasus	Brown	1,750	mgrimes	canderson	Christine	Anderson
23	Sammy	Black	2,200	mgrimes	canderson	Christine	Anderson
24	Pinky	Red	1,050	tswift	canderson	Christine	Anderson
25	Hulk	Grey	2,050	mgrimes	canderson	Christine	Anderson
26	Pat	White	1,400	mgrimes	canderson	Christine	Anderson
27	Betty	White	1,250	tswift	canderson	Christine	Anderson
28	Shamrock	Black	1,400	[NULL]	canderson	Christine	Anderson

Cartesian Product

- IF you want to do a Cartesian product (which should be rare) in Access or any other DBMS, the following will produce the same result:
  - SELECT * FROM *table1, table2*;

Inner Join - Equijoin

- When we add some evaluation criteria to the Cartesian product, it becomes equivalent to an Inner Join, which is expressed as: $H \bowtie_{(H.Owner=C.Username)} C$

- In SQL, we use the INNER JOIN operator and use the ON operator to specify the join condition(s):
  - SELECT * FROM H INNER JOIN C ON H.owner = C.username

	NAME	COLOR	WEIGHT	OWNER
1	Sam	Brown	1,500	mgrimes
2	Erica	Yellow	920	canderson
3	John	Grey	1,800	mgrimes
4	Trotty	Brown	1,300	mgrimes
5	Rio	Grey	1,700	tswift
6	Robin	Yellow	1,100	jisbell
7	Katy	Brown	1,200	jisbell
8	Pegasus	Brown	1,750	mgrimes
9	Sammy	Black	2,200	mgrimes
10	Pinky	Red	1,050	tswift
11	Hulk	Grey	2,050	mgrimes
12	Pat	White	1,400	mgrimes
13	Betty	White	1,250	tswift
14	Shamrock	Black	1,400	[NULL]

	USERNAME	FNAME	LNAME
1	mgrimes	Marvin	Grimes
2	canderson	Christine	Anderson
3	tswift	Tina	Swift
4	jisbell	Jason	Isbell
5	ssimpson	Sam	Simpson

	NAME	COLOR	WEIGHT	OWNER	USERNAME	FNAME	LNAME
1	Sam	Brown	1,500	mgrimes	mgrimes	Marvin	Grimes
2	Erica	Yellow	920	canderson	canderson	Christine	Anderson
3	John	Grey	1,800	mgrimes	mgrimes	Marvin	Grimes
4	Trotty	Brown	1,300	mgrimes	mgrimes	Marvin	Grimes
5	Rio	Grey	1,700	tswift	tswift	Tina	Swift
6	Robin	Yellow	1,100	jisbell	jisbell	Jason	Isbell
7	Katy	Brown	1,200	jisbell	jisbell	Jason	Isbell
8	Pegasus	Brown	1,750	mgrimes	mgrimes	Marvin	Grimes
9	Sammy	Black	2,200	mgrimes	mgrimes	Marvin	Grimes
10	Pinky	Red	1,050	tswift	tswift	Tina	Swift
11	Hulk	Grey	2,050	mgrimes	mgrimes	Marvin	Grimes
12	Pat	White	1,400	mgrimes	mgrimes	Marvin	Grimes
13	Betty	White	1,250	tswift	tswift	Tina	Swift

Inner Join - Equijoin

- Quick note:
  - "INNER JOIN" and "JOIN" are equivalent terms so this:
    - SELECT * FROM H INNER JOIN C ON H.owner = C.username;

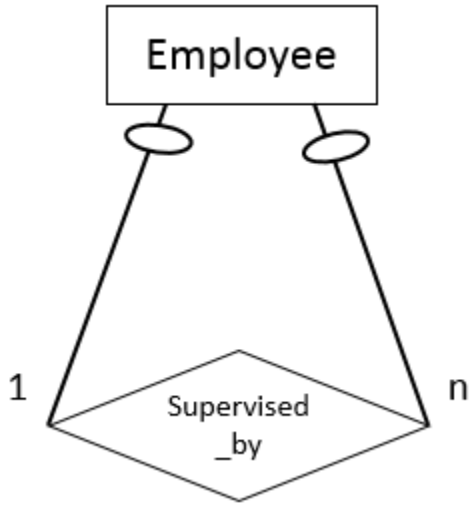
      …is the same as:

    - SELECT * FROM H JOIN C ON H.owner = C.username;
- Using INNER makes it a little more clear what you are doing
- NOTE: MS Access DOES REQUIRE you to use the word "INNER"

	NAME	COLOR	WEIGHT	OWNER	USERNAME	FNAME	LNAME
1	Sam	Brown	1,500	mgrimes	mgrimes	Marvin	Grimes
2	Erica	Yellow	920	canderson	canderson	Christine	Anderson
3	John	Grey	1,800	mgrimes	mgrimes	Marvin	Grimes
4	Trotty	Brown	1,300	mgrimes	mgrimes	Marvin	Grimes
5	Rio	Grey	1,700	tswift	tswift	Tina	Swift
6	Robin	Yellow	1,100	jisbell	jisbell	Jason	Isbell
7	Katy	Brown	1,200	jisbell	jisbell	Jason	Isbell
8	Pegasus	Brown	1,750	mgrimes	mgrimes	Marvin	Grimes
9	Sammy	Black	2,200	mgrimes	mgrimes	Marvin	Grimes
10	Pinky	Red	1,050	tswift	tswift	Tina	Swift
11	Hulk	Grey	2,050	mgrimes	mgrimes	Marvin	Grimes
12	Pat	White	1,400	mgrimes	mgrimes	Marvin	Grimes
13	Betty	White	1,250	tswift	tswift	Tina	Swift

Inner Join - Joining a Relation With Itself

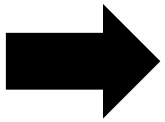
- Used for recursive (unary) relationships
  - It is necessary to use table aliases if using the same table twice (or more) in one query
- SELECT *attributes*
  FROM *table1 alias1*
  INNER JOIN *table1 alias2*
  ON *alias1.attr = alias2.attr*



- SELECT E.EmpID, E.Name AS EmpName, M.Name AS ManagerName
  FROM Employee E
  INNER JOIN Employee M
  ON E.MgrID = M.EmpID;

Employee

EmpID	Name	MgrID
1	Mark	
2	Jill	1
3	Jane	1
4	Molly	3
5	Gus	3



EmpID	EmpName	ManagerName
2	Jill	Mark
3	Jane	Mark
4	Molly	Jane
5	Gus	Jane

An alternate syntax for inner joins

- Inner join RA syntax is: $H \bowtie_{(H.Owner=C.Username)} C$
- However, we also said we can do a Cartesian product and Select only the rows where the joining attribute matches:

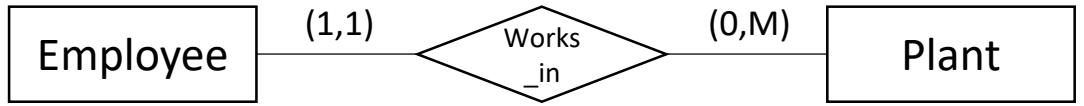
  $\sigma_{(H.Owner=C.Username)} H \times C$
- SELECT * FROM *table1*, *table2*
  WHERE *table1.attribute* = *table2.attribute*
- This implies a 1:M relationship
  - The table with the FK is the “child”

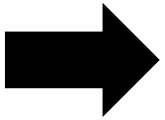
An alternate syntax for inner joins

- SELECT * FROM Employee E, Plant P
  WHERE E.PlantID = P.PlantID;
  - Note the use of aliases (E and P) to make the joining operation more concise
  - This implies a 1:M relationship – Plant is the parent

Employee		
EmpID	Name	PlantID
1	Mark	10
2	Jill	10
3	Jane	11
4	Molly	11
5	Gus	12

Plant	
PlantID	PLName
10	River Oaks
11	Oaks Forest
12	Garden Oaks
13	Heights





EmpID	Name	E.PlantID	P.PlantID	PLName
1	Mark	10	10	River Oaks
2	Jill	10	10	River Oaks
3	Jane	11	11	Oaks Forest
4	Molly	11	11	Oaks Forest
5	Gus	12	12	Garden Oaks

Inner Join – More than two tables

- You can join any number of tables together by nesting the join operations:

- SELECT *attributes* FROM
  (*table1* JOIN *table2* ON *table1.attribute* = *table2.attribute*)
  JOIN *table3* ON *table2.attribute* = *table3.attribute*

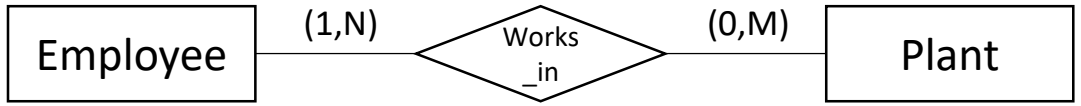
- What attribute (and tables) you use for the ON part of the join will depend on what you are trying to do

- See Page 606 for another example.

Inner Join – More than two tables

- If we had a M:N relationship which had been decomposed with a gerund we would need to join all three tables…

- Decomposes to



Employee

EmpID	Name
1	Mark
2	Jill
3	Jane
4	Molly
5	Gus

Works_in

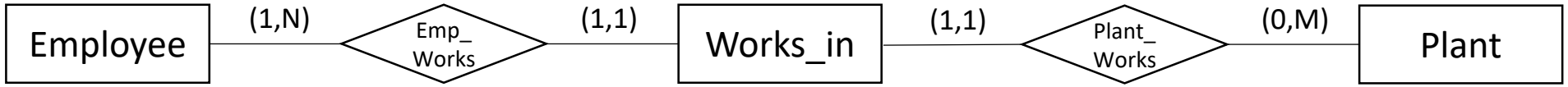
EmpID	PlantID
1	10
2	10
2	13
3	11
4	11
4	13
5	12

Plant

PlantID	PLName
10	River Oaks
11	Oaks Forest
12	Garden Oaks
13	Heights

Inner Join – More than two tables

SELECT * FROM (Employee E INNER JOIN Works_in W ON E.EmpID = W.EmpID)
INNER JOIN Plant P on P.PlantID = W.PlantID;



Employee

EmpID	Name
1	Mark
2	Jill
3	Jane
4	Molly
5	Gus

Works_in

EmpID	PlantID
1	10
2	10
2	13
3	11
4	11
4	13
5	12

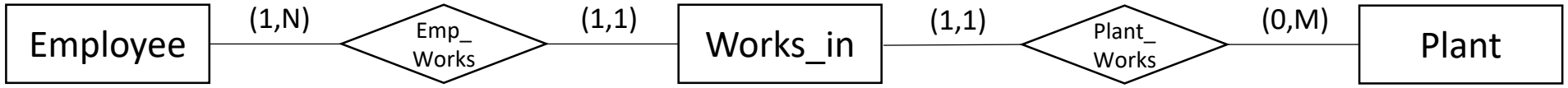
Plant

PlantID	PLName
10	River Oaks
11	Oaks Forest
12	Garden Oaks
13	Heights

E.EmpID	Name	W.EmpID	W.PlantID	P.PlantID	PLName
1	Mark	1	10	10	River Oaks
2	Jill	2	10	10	River Oaks
2	Jill	2	13	13	Heights
3	Jane	3	11	11	Oaks Forest
4	Molly	4	11	11	Oaks Forest
4	Molly	4	13	13	Heights
5	Gus	5	12	12	Garden Oaks

An alternate syntax for inner joins with more than two tables – the SQL?

```
SELECT * FROM Employee E, Plant P, Works_in W
WHERE E.EmpID = W.EmpID AND P.PlantID = W.PlantID
```



Employee

EmpID	Name
1	Mark
2	Jill
3	Jane
4	Molly
5	Gus

Works_in

EmpID	PlantID
1	10
2	10
2	13
3	11
4	11
4	13
5	12

Plant

PlantID	PLName
10	River Oaks
11	Oaks Forest
12	Garden Oaks
13	Heights

E.EmpID	Name	W.EmpID	W.PlantID	P.PlantID	PLName
1	Mark	1	10	10	River Oaks
2	Jill	2	10	10	River Oaks
2	Jill	2	13	13	Heights
3	Jane	3	11	11	Oaks Forest
4	Molly	4	11	11	Oaks Forest
4	Molly	4	13	13	Heights
5	Gus	5	12	12	Garden Oaks

Outer Joins

- In Inner Join operations, tuples without a matching (or related) tuple are eliminated from the Join result.
- Tuples with null values in the join attributes are also eliminated.
- A set of operations, called Outer Joins, can be used when we want to keep all the tuples in R, or those in S, or those in both relations in the result of the Join, whether or not they have matching tuples in the other relation.

Left Outer Join

- $H \bowtie_{(H.Owner=C.Username)} C$

- SELECT * FROM H LEFT OUTER JOIN C ON H.Owner = C.username;

	NAME	COLOR	WEIGHT	OWNER
1	Sam	Brown	1,500	mgrimes
2	Erica	Yellow	920	canderson
3	John	Grey	1,800	mgrimes
4	Trotty	Brown	1,300	mgrimes
5	Rio	Grey	1,700	tswift
6	Robin	Yellow	1,100	jisbell
7	Katy	Brown	1,200	jisbell
8	Pegasus	Brown	1,750	mgrimes
9	Sammy	Black	2,200	mgrimes
10	Pinky	Red	1,050	tswift
11	Hulk	Grey	2,050	mgrimes
12	Pat	White	1,400	mgrimes
13	Betty	White	1,250	tswift
14	Shamrock	Black	1,400	[NULL]

	USERNAME	FNAME	LNAME
1	mgrimes	Marvin	Grimes
2	canderson	Christine	Anderson
3	tswift	Tina	Swift
4	jisbell	Jason	Isbell
5	ssimpson	Sam	Simpson

	NAME	COLOR	WEIGHT	OWNER	USERNAME	FNAME	LNAME
1	Pat	White	1,400	mgrimes	mgrimes	Marvin	Grimes
2	Hulk	Grey	2,050	mgrimes	mgrimes	Marvin	Grimes
3	Sammy	Black	2,200	mgrimes	mgrimes	Marvin	Grimes
4	Pegasus	Brown	1,750	mgrimes	mgrimes	Marvin	Grimes
5	Trotty	Brown	1,300	mgrimes	mgrimes	Marvin	Grimes
6	John	Grey	1,800	mgrimes	mgrimes	Marvin	Grimes
7	Sam	Brown	1,500	mgrimes	mgrimes	Marvin	Grimes
8	Erica	Yellow	920	canderson	canderson	Christine	Anderson
9	Betty	White	1,250	tswift	tswift	Tina	Swift
10	Pinky	Red	1,050	tswift	tswift	Tina	Swift
11	Rio	Grey	1,700	tswift	tswift	Tina	Swift
12	Katy	Brown	1,200	jisbell	jisbell	Jason	Isbell
13	Robin	Yellow	1,100	jisbell	jisbell	Jason	Isbell
14	Shamrock	Black	1,400	[NULL]	[NULL]	[NULL]	[NULL]

Right Outer Join

- H ⋈ $_{(H.Owner=C.Username)}$ C

- SELECT * FROM H RIGHT OUTER JOIN C ON H.Owner = C.username;

	NAME	COLOR	WEIGHT	OWNER
1	Sam	Brown	1,500	mgrimes
2	Erica	Yellow	920	canderson
3	John	Grey	1,800	mgrimes
4	Trotty	Brown	1,300	mgrimes
5	Rio	Grey	1,700	tswift
6	Robin	Yellow	1,100	jisbell
7	Katy	Brown	1,200	jisbell
8	Pegasus	Brown	1,750	mgrimes
9	Sammy	Black	2,200	mgrimes
10	Pinky	Red	1,050	tswift
11	Hulk	Grey	2,050	mgrimes
12	Pat	White	1,400	mgrimes
13	Betty	White	1,250	tswift
14	Shamrock	Black	1,400	[NULL]

	USERNAME	FNAME	LNAME
1	mgrimes	Marvin	Grimes
2	canderson	Christine	Anderson
3	tswift	Tina	Swift
4	jisbell	Jason	Isbell
5	ssimpson	Sam	Simpson

	NAME	COLOR	WEIGHT	OWNER	USERNAME	FNAME	LNAME
1	Sam	Brown	1,500	mgrimes	mgrimes	Marvin	Grimes
2	Erica	Yellow	920	canderson	canderson	Christine	Anderson
3	John	Grey	1,800	mgrimes	mgrimes	Marvin	Grimes
4	Trotty	Brown	1,300	mgrimes	mgrimes	Marvin	Grimes
5	Rio	Grey	1,700	tswift	tswift	Tina	Swift
6	Robin	Yellow	1,100	jisbell	jisbell	Jason	Isbell
7	Katy	Brown	1,200	jisbell	jisbell	Jason	Isbell
8	Pegasus	Brown	1,750	mgrimes	mgrimes	Marvin	Grimes
9	Sammy	Black	2,200	mgrimes	mgrimes	Marvin	Grimes
10	Pinky	Red	1,050	tswift	tswift	Tina	Swift
11	Hulk	Grey	2,050	mgrimes	mgrimes	Marvin	Grimes
12	Pat	White	1,400	mgrimes	mgrimes	Marvin	Grimes
13	Betty	White	1,250	tswift	tswift	Tina	Swift
14	[NULL]	[NULL]	[NULL]	[NULL]	ssimpson	Sam	Simpson

Full Outer Join

- $H \bowtie_{(H.Owner=C.Username)} C$

- SELECT * FROM H FULL OUTER JOIN C ON H.Owner = C.username;

	NAME	COLOR	WEIGHT	OWNER
1	Sam	Brown	1,500	mgrimes
2	Erica	Yellow	920	canderson
3	John	Grey	1,800	mgrimes
4	Trotty	Brown	1,300	mgrimes
5	Rio	Grey	1,700	tswift
6	Robin	Yellow	1,100	jisbell
7	Katy	Brown	1,200	jisbell
8	Pegasus	Brown	1,750	mgrimes
9	Sammy	Black	2,200	mgrimes
10	Pinky	Red	1,050	tswift
11	Hulk	Grey	2,050	mgrimes
12	Pat	White	1,400	mgrimes
13	Betty	White	1,250	tswift
14	Shamrock	Black	1,400	[NULL]

	USERNAME	FNAME	LNAME
1	mgrimes	Marvin	Grimes
2	canderson	Christine	Anderson
3	tswift	Tina	Swift
4	jisbell	Jason	Isbell
5	ssimpson	Sam	Simpson

	NAME	COLOR	WEIGHT	OWNER	USERNAME	FNAME	LNAME
1	Sam	Brown	1,500	mgrimes	mgrimes	Marvin	Grimes
2	Erica	Yellow	920	canderson	canderson	Christine	Anderson
3	John	Grey	1,800	mgrimes	mgrimes	Marvin	Grimes
4	Trotty	Brown	1,300	mgrimes	mgrimes	Marvin	Grimes
5	Rio	Grey	1,700	tswift	tswift	Tina	Swift
6	Robin	Yellow	1,100	jisbell	jisbell	Jason	Isbell
7	Katy	Brown	1,200	jisbell	jisbell	Jason	Isbell
8	Pegasus	Brown	1,750	mgrimes	mgrimes	Marvin	Grimes
9	Sammy	Black	2,200	mgrimes	mgrimes	Marvin	Grimes
10	Pinky	Red	1,050	tswift	tswift	Tina	Swift
11	Hulk	Grey	2,050	mgrimes	mgrimes	Marvin	Grimes
12	Pat	White	1,400	mgrimes	mgrimes	Marvin	Grimes
13	Betty	White	1,250	tswift	tswift	Tina	Swift
14	Shamrock	Black	1,400	[NULL]	[NULL]	[NULL]	[NULL]
15	[NULL]	[NULL]	[NULL]	[NULL]	ssimpson	Sam	Simpson

Full Outer Join

- Note that a full outer join is equivalent to the union of a left outer join and a right outer join

- $H \overset{(H.Owner=C.Username)}{\bowtie} C$

```
SELECT * FROM H FULL OUTER JOIN C ON H.owner = C.username;
```

Is equivalent to

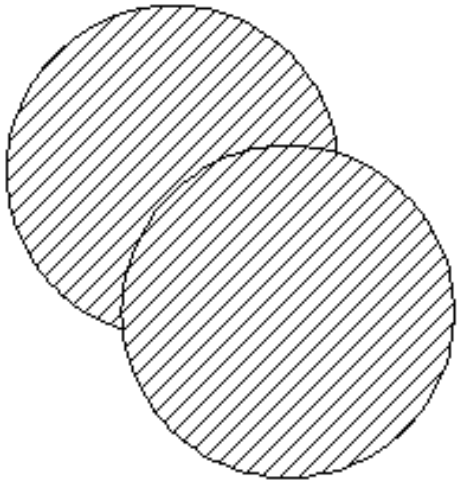
- $H \overset{(H.Owner=C.Username)}{\bowtie} C \cup H \overset{(H.Owner=C.Username)}{\bowtie} C$

```
SELECT * FROM H LEFT OUTER JOIN C ON H.owner = C.username
UNION
SELECT * FROM H RIGHT OUTER JOIN C ON H.owner = C.username;
```

Remember, set theory operators do not work in Access

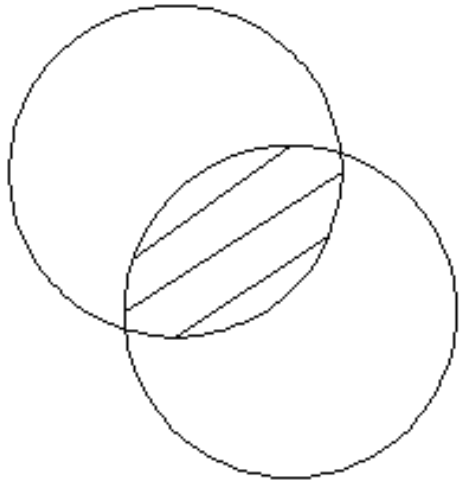
Set Theory Operators

Union



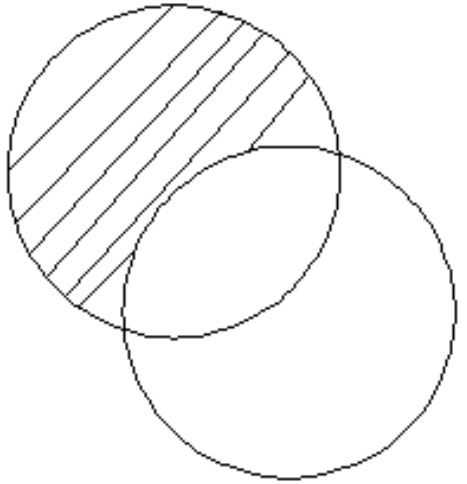
“OR”

Intersection



“AND”

Difference



“NOT”

Set Operations

- Two relations (R and S) are “union compatible” if they:
  - Have the same degree (number of attributes)
  - Pairs of attributes from R and S have the same domain

- Union compatibility is a requirement for all set operations (union, intersection, and difference)

Relation F: Horses that are Female

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sam	Brown	No	F	1,500
2	Erica	Yellow	Yes	F	920
3	Rio	Grey	No	F	1,700
4	Katy	Brown	No	F	1,200
5	Pat	White	No	F	1,400
6	Betty	White	Yes	F	1,250

Relation S: Horses that have spots

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Erica	Yellow	Yes	F	920
2	Trotty	Brown	Yes	M	1,300
3	Sammy	Black	Yes	M	2,200
4	Betty	White	Yes	F	1,250

Union

- Horses that are EITHER Female OR have spots

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sam	Brown	No	F	1,500
2	Erica	Yellow	Yes	F	920
3	Rio	Grey	No	F	1,700
4	Katy	Brown	No	F	1,200
5	Pat	White	No	F	1,400
6	Betty	White	Yes	F	1,250

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Erica	Yellow	Yes	F	920
2	Trotty	Brown	Yes	M	1,300
3	Sammy	Black	Yes	M	2,200
4	Betty	White	Yes	F	1,250

F ∪ S

SELECT * FROM F UNION SELECT * FROM S;

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Betty	White	Yes	F	1,250
2	Erica	Yellow	Yes	F	920
3	Katy	Brown	No	F	1,200
4	Pat	White	No	F	1,400
5	Rio	Grey	No	F	1,700
6	Sam	Brown	No	F	1,500
7	Sammy	Black	Yes	M	2,200
8	Trotty	Brown	Yes	M	1,300

- Note that only unique tuples are returned

Intersection

- Horses that are Female AND have spots

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sam	Brown	No	F	1,500
2	Erica	Yellow	Yes	F	920
3	Rio	Grey	No	F	1,700
4	Katy	Brown	No	F	1,200
5	Pat	White	No	F	1,400
6	Betty	White	Yes	F	1,250

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Erica	Yellow	Yes	F	920
2	Trotty	Brown	Yes	M	1,300
3	Sammy	Black	Yes	M	2,200
4	Betty	White	Yes	F	1,250

$F \cap S$

SELECT * FROM F INTERSECT SELECT * FROM S;

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Betty	White	Yes	F	1,250
2	Erica	Yellow	Yes	F	920

Difference

- Horses that are female but do NOT have spots

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sam	Brown	No	F	1,500
2	Erica	Yellow	Yes	F	920
3	Rio	Grey	No	F	1,700
4	Katy	Brown	No	F	1,200
5	Pat	White	No	F	1,400
6	Betty	White	Yes	F	1,250

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Erica	Yellow	Yes	F	920
2	Trotty	Brown	Yes	M	1,300
3	Sammy	Black	Yes	M	2,200
4	Betty	White	Yes	F	1,250

F – S

SELECT * FROM F MINUS SELECT * FROM S;

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Katy	Brown	No	F	1,200
2	Pat	White	No	F	1,400
3	Rio	Grey	No	F	1,700
4	Sam	Brown	No	F	1,500

Difference

- Horses that have spots but are NOT female

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sam	Brown	No	F	1,500
2	Erica	Yellow	Yes	F	920
3	Rio	Grey	No	F	1,700
4	Katy	Brown	No	F	1,200
5	Pat	White	No	F	1,400
6	Betty	White	Yes	F	1,250

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Erica	Yellow	Yes	F	920
2	Trotty	Brown	Yes	M	1,300
3	Sammy	Black	Yes	M	2,200
4	Betty	White	Yes	F	1,250

S – F

SELECT * FROM S MINUS SELECT * FROM F;

	NAME	COLOR	SPOTS	SEX	WEIGHT
1	Sammy	Black	Yes	M	2,200
2	Trotty	Brown	Yes	M	1,300

Demo

SELECT * FROM SET1;

	NAME
1	Adam
2	Ben
3	Claire
4	Dana

SELECT * FROM SET2;

	NAME
1	Ben
2	Dana
3	Eugene

Difference: Note

- The keyword “MINUS” is used in Oracle, whereas the keyword “EXCEPT” is used in most other DBMS

Oracle: SELECT * FROM S MINUS SELECT * FROM R;

Others: SELECT * FROM S EXCEPT SELECT * FROM R;

Progress Quiz Time!

- The Progress Quiz is available in Canvas
  - You MUST complete the quiz on Canvas by 5:00 on Friday – This in-class activity does not count for points!
  - Each week we will discuss the questions, so for those of you that are in class and keeping up with things, you'll have an extra easy time with it!
- Go to http://kahoot.it and we'll get started momentarily!

Break

Module 12.3
Subqueries

Module 12.3: Subqueries

- A complete SELECT statement embedded within another SELECT statement
  - One or more “inner” queries nested within an “outer” query

- Either uncorrelated or correlated
  - Uncorrelated: Executed once and results used by outer query
  - Correlated: Executed once for each row in the outer query

Uncorrelated Subqueries

- The subquery is executed first and passes one or more values to the outer query
  - Three operators may be used: IN, ANY, ALL
  - May be negated with the NOT operator

- The IN operator evaluates if rows processed by the outer query are equal to any of the values returned by the subquery (i.e., it creates an OR condition).

Uncorrelated Subqueries (IN)

- SELECT * FROM horses
  WHERE name IN (SELECT ord_Horsename FROM orders);

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1,800	mgrimes
4	Trotty	Brown	Yes	M	1,300	mgrimes
5	Rio	Grey	No	F	1,700	tswift
6	Robin	Yellow	No	M	1,100	jisbell
7	Katy	Brown	No	F	1,200	jisbell
8	Pegasus	Brown	No	M	1,750	mgrimes
9	Sammy	Black	Yes	M	2,200	mgrimes
10	Pinky	Red	No	M	1,050	tswift
11	Hulk	Grey	No	M	2,050	mgrimes
12	Pat	White	No	F	1,400	mgrimes
13	Betty	White	Yes	F	1,250	tswift
14	Shamrock	Black	No	M	1,400	[NULL]

	RXNUM	ORD_MEDCODE	ORD_HORSENAME	DOSE	FREQUENCY
1	108	Dexa	Betty	2	1
2	107	Dexa	Shamrock	2	1
3	105	Doxy	Trotty	1	3
4	110	Enro	Erica	3	2
5	101	Flux	Sam	1	2
6	104	Iver	Trotty	2	1
7	103	Iver	John	2	1
8	102	Pen	Sam	1	1
9	106	Pres	Shamrock	1	1
10	109	Xyl	Hulk	1	1

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	John	Grey	No	M	1,800	mgrimes
3	Trotty	Brown	Yes	M	1,300	mgrimes
4	Shamrock	Black	No	M	1,400	[NULL]
5	Betty	White	Yes	F	1,250	tswift
6	Hulk	Grey	No	M	2,050	mgrimes
7	Erica	Yellow	Yes	F	920	canderson

…So this query is effectively the same as:

```
SELECT * FROM horses WHERE name IN ('Betty', 'Shamrock', 'Trotty',
'Erica', 'Sam', 'Trotty', 'John', 'Sam', 'Shamrock', 'Hulk');
```

But even better, since list of names is created dynamically based on the subquery!

Uncorrelated Subqueries (NOT IN)

- SELECT * FROM horses
  WHERE name NOT IN (SELECT ord_Horsename FROM orders);

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1,500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1,800	mgrimes
4	Trotty	Brown	Yes	M	1,300	mgrimes
5	Rio	Grey	No	F	1,700	tswift
6	Robin	Yellow	No	M	1,100	jisbell
7	Katy	Brown	No	F	1,200	jisbell
8	Pegasus	Brown	No	M	1,750	mgrimes
9	Sammy	Black	Yes	M	2,200	mgrimes
10	Pinky	Red	No	M	1,050	tswift
11	Hulk	Grey	No	M	2,050	mgrimes
12	Pat	White	No	F	1,400	mgrimes
13	Betty	White	Yes	F	1,250	tswift
14	Shamrock	Black	No	M	1,400	[NULL]

	RXNUM	ORD_MEDCODE	ORD_HORSENAME	DOSE	FREQUENCY
1	108	Dexa	Betty	2	1
2	107	Dexa	Shamrock	2	1
3	105	Doxy	Trotty	1	3
4	110	Enro	Erica	3	2
5	101	Flux	Sam	1	2
6	104	Iver	Trotty	2	1
7	103	Iver	John	2	1
8	102	Pen	Sam	1	1
9	106	Pres	Shamrock	1	1
10	109	Xyl	Hulk	1	1

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Pinky	Red	No	M	1050	tswift
2	Pegasus	Brown	No	M	1750	mgrimes
3	Sammy	Black	Yes	M	2200	mgrimes
4	Katy	Brown	No	F	1200	jisbell
5	Rio	Grey	No	F	1700	tswift
6	Pat	White	No	F	1400	mgrimes
7	Robin	Yellow	No	M	1100	jisbell

Use of ALL and ANY With Subqueries

- The ALL and ANY operators can be combined with other comparison operators to treat the results of a subquery as a set of values, rather than as individual values.

Operator	Description
> ALL	Greater than the highest value returned by the subquery
< ALL	Less than the lowest value returned by the subquery
< ANY	Less than the highest value returned by the subquery
> ANY	Greater than the lowest value returned by the subquery
= ANY	Equal to any value returned by the subquery (equivalent to the IN operator)

Use of > ALL in a Subquery

	123 EMPID	ABC NAME	ABC TITLE	123 SALARY
1	402	Daniel Taylor	Director of Education	140000
2	102	Matthew Martinez	Director of Facilities	125000
3	103	Kimberly Hall	Director of Racing	160000
4	852	Richard Davis	Equine Specialist	57000
5	750	Charles Wilson	Equine Specialist	60000
6	987	David Jones	Equine Specialist	58000
7	955	Anthony Thompson	Financial Advisor	185000
8	599	Thomas Moore	Health and Nutrition Specialist	135000
9	959	Mary Garcia	Nutritionist	70000
10	977	William Brown	Nutritionist	82000
11	812	Robert Williams	Nutritionist	85000
12	101	James Smith	Owner	250000
13	923	Jennifer Anderson	Race Coordinator	80000
14	414	Michael Johnson	Race Coordinator	82000
15	557	Karen Lewis	Race Coordinator	78000
16	978	Joseph Miller	Ranch Hand	57000
17	973	Lisa Martinez	Ranch Hand	45000
18	670	Christopher Anderson	Ranch Hand	52000
19	436	Sarah Rodriguez	Ranch Hand	55000
20	735	Angela Young	Ranch Hand	62000

	ABC NAME	123 SALARY	ABC TITLE
1	Matthew Martinez	125000	Director of Facilities
2	Thomas Moore	135000	Health and Nutrition Specialist
3	Daniel Taylor	140000	Director of Education
4	Kimberly Hall	160000	Director of Racing
5	Anthony Thompson	185000	Financial Advisor
6	James Smith	250000	Owner

```
SELECT name, salary, title FROM workers WHERE salary > ALL
     (SELECT salary FROM workers WHERE title='Nutritionist');
```

Use of >= ALL in a Subquery

	123 EMPID	ABC NAME	ABC TITLE	123 SALARY
1	402	Daniel Taylor	Director of Education	140000
2	102	Matthew Martinez	Director of Facilities	125000
3	103	Kimberly Hall	Director of Racing	160000
4	852	Richard Davis	Equine Specialist	57000
5	750	Charles Wilson	Equine Specialist	60000
6	987	David Jones	Equine Specialist	58000
7	955	Anthony Thompson	Financial Advisor	185000
8	599	Thomas Moore	Health and Nutrition Specialist	135000
9	959	Mary Garcia	Nutritionist	70000
10	977	William Brown	Nutritionist	82000
11	812	Robert Williams	Nutritionist	85000
12	101	James Smith	Owner	250000
13	923	Jennifer Anderson	Race Coordinator	80000
14	414	Michael Johnson	Race Coordinator	82000
15	557	Karen Lewis	Race Coordinator	78000
16	978	Joseph Miller	Ranch Hand	57000
17	973	Lisa Martinez	Ranch Hand	45000
18	670	Christopher Anderson	Ranch Hand	52000
19	436	Sarah Rodriguez	Ranch Hand	55000
20	735	Angela Young	Ranch Hand	62000

	ABC NAME	123 SALARY	ABC TITLE
1	Robert Williams	85000	Nutritionist
2	Matthew Martinez	125000	Director of Facilities
3	Thomas Moore	135000	Health and Nutrition Specialist
4	Daniel Taylor	140000	Director of Education
5	Kimberly Hall	160000	Director of Racing
6	Anthony Thompson	185000	Financial Advisor
7	James Smith	250000	Owner

```
SELECT name, salary FROM workers WHERE salary >= ALL
     (SELECT salary FROM workers WHERE title='Nutritionist');
```

Same results with an aggregate function

	123 EMPID	ABC NAME	ABC TITLE	123 SALARY
1	402	Daniel Taylor	Director of Education	140000
2	102	Matthew Martinez	Director of Facilities	125000
3	103	Kimberly Hall	Director of Racing	160000
4	852	Richard Davis	Equine Specialist	57000
5	750	Charles Wilson	Equine Specialist	60000
6	987	David Jones	Equine Specialist	58000
7	955	Anthony Thompson	Financial Advisor	185000
8	599	Thomas Moore	Health and Nutrition Specialist	135000
9	959	Mary Garcia	Nutritionist	70000
10	977	William Brown	Nutritionist	82000
11	812	Robert Williams	Nutritionist	85000
12	101	James Smith	Owner	250000
13	923	Jennifer Anderson	Race Coordinator	80000
14	414	Michael Johnson	Race Coordinator	82000
15	557	Karen Lewis	Race Coordinator	78000
16	978	Joseph Miller	Ranch Hand	57000
17	973	Lisa Martinez	Ranch Hand	45000
18	670	Christopher Anderson	Ranch Hand	52000
19	436	Sarah Rodriguez	Ranch Hand	55000
20	735	Angela Young	Ranch Hand	62000

	ABC NAME	123 SALARY	ABC TITLE
1	Matthew Martinez	125000	Director of Facilities
2	Thomas Moore	135000	Health and Nutrition Specialist
3	Daniel Taylor	140000	Director of Education
4	Kimberly Hall	160000	Director of Racing
5	Anthony Thompson	185000	Financial Advisor
6	James Smith	250000	Owner

```
SELECT name, salary FROM workers WHERE salary >
   (SELECT max(salary) FROM workers WHERE title='Nutritionist');
```

Use of > ANY in a Subquery

	123 EMPID	ABC NAME	ABC TITLE	123 SALARY
1	402	Daniel Taylor	Director of Education	140000
2	102	Matthew Martinez	Director of Facilities	125000
3	103	Kimberly Hall	Director of Racing	160000
4	852	Richard Davis	Equine Specialist	57000
5	750	Charles Wilson	Equine Specialist	60000
6	987	David Jones	Equine Specialist	58000
7	955	Anthony Thompson	Financial Advisor	185000
8	599	Thomas Moore	Health and Nutrition Specialist	135000
9	959	Mary Garcia	Nutritionist	70000
10	977	William Brown	Nutritionist	82000
11	812	Robert Williams	Nutritionist	85000
12	101	James Smith	Owner	250000
13	923	Jennifer Anderson	Race Coordinator	80000
14	414	Michael Johnson	Race Coordinator	82000
15	557	Karen Lewis	Race Coordinator	78000
16	978	Joseph Miller	Ranch Hand	57000
17	973	Lisa Martinez	Ranch Hand	45000
18	670	Christopher Anderson	Ranch Hand	52000
19	436	Sarah Rodriguez	Ranch Hand	55000
20	735	Angela Young	Ranch Hand	62000

	ABC NAME	123 SALARY	ABC TITLE
1	Daniel Taylor	140000	Director of Education
2	Matthew Martinez	125000	Director of Facilities
3	Kimberly Hall	160000	Director of Racing
4	Anthony Thompson	185000	Financial Advisor
5	Thomas Moore	135000	Health and Nutrition Specialist
6	Robert Williams	85000	Nutritionist
7	William Brown	82000	Nutritionist
8	James Smith	250000	Owner
9	Michael Johnson	82000	Race Coordinator
10	Jennifer Anderson	80000	Race Coordinator
11	Karen Lewis	78000	Race Coordinator

```
SELECT name, salary, title FROM workers WHERE salary > ANY
     (SELECT salary FROM workers WHERE title='Nutritionist');
```

Will return records where salary is greater than ANY salary for Nutritionists (70,000)

Can you write this using an aggregate function in the subquery?

```
SELECT name, salary, title FROM workers WHERE salary > (SELECT min(salary) FROM workers WHERE title='Nutritionist');
```

Subqueries in FROM Clause (“inline view”)

```
SELECT a.name, a.title, a.salary, b.title_avg
FROM workers a JOIN
(SELECT title, round(avg(salary),0) AS Title_Avg
 FROM workers GROUP BY title) b
ON a.title = b.title
WHERE a.salary > b.title_avg;
```

	TITLE	TITLE_AVG
1	Ranch Hand	54200
2	Health and Nutrition Specialist	135000
3	Owner	250000
4	Race Coordinator	80000
5	Nutritionist	79000
6	Director of Education	140000
7	Director of Facilities	125000
8	Director of Racing	160000
9	Financial Advisor	185000
10	Equine Specialist	58333

	EMPID	NAME	TITLE	SALARY
1	402	Daniel Taylor	Director of Education	140000
2	102	Matthew Martinez	Director of Facilities	125000
3	103	Kimberly Hall	Director of Racing	160000
4	852	Richard Davis	Equine Specialist	57000
5	750	Charles Wilson	Equine Specialist	60000
6	987	David Jones	Equine Specialist	58000
7	955	Anthony Thompson	Financial Advisor	185000
8	599	Thomas Moore	Health and Nutrition Specialist	135000
9	959	Mary Garcia	Nutritionist	70000
10	977	William Brown	Nutritionist	82000
11	812	Robert Williams	Nutritionist	85000
12	101	James Smith	Owner	250000
13	923	Jennifer Anderson	Race Coordinator	80000
14	414	Michael Johnson	Race Coordinator	82000
15	557	Karen Lewis	Race Coordinator	78000
16	978	Joseph Miller	Ranch Hand	57000
17	973	Lisa Martinez	Ranch Hand	45000
18	670	Christopher Anderson	Ranch Hand	52000
19	436	Sarah Rodriguez	Ranch Hand	55000
20	735	Angela Young	Ranch Hand	62000

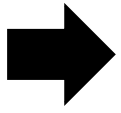
	NAME	TITLE	SALARY	TITLE_AVG
1	Michael Johnson	Race Coordinator	82000	80000
2	Sarah Rodriguez	Ranch Hand	55000	54200
3	Angela Young	Ranch Hand	62000	54200
4	Charles Wilson	Equine Specialist	60000	58333
5	Robert Williams	Nutritionist	85000	79000
6	William Brown	Nutritionist	82000	79000
7	Joseph Miller	Ranch Hand	57000	54200

Subqueries in SELECT Clause

```
SELECT a.name, a.title, a.salary,
ROUND((SELECT AVG(B.SALARY) FROM workers B WHERE A.title = B.title
GROUP BY B.title),0) AS title_Avg
FROM workers A
WHERE a.salary > title_Avg;
```

Same result, but much slower because the subquery executes for every professor rather that executing once, then just joining the relations – this is a CORRELATED SUBQUERY

	123 EMPID	ABC NAME	ABC TITLE	123 SALARY
1	402	Daniel Taylor	Director of Education	140000
2	102	Matthew Martinez	Director of Facilities	125000
3	103	Kimberly Hall	Director of Racing	160000
4	852	Richard Davis	Equine Specialist	57000
5	750	Charles Wilson	Equine Specialist	60000
6	987	David Jones	Equine Specialist	58000
7	955	Anthony Thompson	Financial Advisor	185000
8	599	Thomas Moore	Health and Nutrition Specialist	135000
9	959	Mary Garcia	Nutritionist	70000
10	977	William Brown	Nutritionist	82000
11	812	Robert Williams	Nutritionist	85000
12	101	James Smith	Owner	250000
13	923	Jennifer Anderson	Race Coordinator	80000
14	414	Michael Johnson	Race Coordinator	82000
15	557	Karen Lewis	Race Coordinator	78000
16	978	Joseph Miller	Ranch Hand	57000
17	973	Lisa Martinez	Ranch Hand	45000
18	670	Christopher Anderson	Ranch Hand	52000
19	436	Sarah Rodriguez	Ranch Hand	55000
20	735	Angela Young	Ranch Hand	62000



	ABC NAME	ABC TITLE	123 SALARY	123 TITLE_AVG
1	Michael Johnson	Race Coordinator	82000	80000
2	Sarah Rodriguez	Ranch Hand	55000	54200
3	Angela Young	Ranch Hand	62000	54200
4	Charles Wilson	Equine Specialist	60000	58333
5	Robert Williams	Nutritionist	85000	79000
6	William Brown	Nutritionist	82000	79000
7	Joseph Miller	Ranch Hand	57000	54200

Correlated Subqueries

- The subquery is executed once for every row in the outer query
- A correlated subquery can be used if it is necessary to check if a nested subquery returns no rows, using the EXISTS operator which returns the value of true if a set is non-empty.
- In a correlated subquery, the subquery is executed once for each row in the outer query. In addition, the execution of the subquery stops and the EXISTS condition of the main query is declared true for a given row should the condition in the subquery be true.
- Correlated subquery (using EXISTS) are generally not as efficient (or easy to understand) as uncorrelated queries – I would typically avoid them!

Use of EXISTS in a Query

Using a correlated subquery:

Display the names of customers who own at least one horse.

```
SELECT username
FROM customers
WHERE EXISTS
   (SELECT * FROM horses
    WHERE customers.username = horses.owner);
```

This is not a very efficient query, because the inner query is executed once for every instance of customer!

	USERNAME
1	canderson
2	jisbell
3	mgrimes
4	tswift

	USERNAME	FNAME	LNAME	PHONE
1	mgrimes	Marvin	Grimes	(218) 330-8004
2	canderson	Christine	Anderson	(555) 523-9989
3	tswift	Tina	Swift	(555) 424-1313
4	jisbell	Jason	Isbell	(615) 555-5555
5	ssimpson	Sam	Simpson	(615) 387-9682

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1800	mgrimes
4	Trotty	Brown	Yes	M	1300	mgrimes
5	Rio	Grey	No	F	1700	tswift
6	Robin	Yellow	No	M	1100	jisbell
7	Katy	Brown	No	F	1200	jisbell
8	Pegasus	Brown	No	M	1750	mgrimes
9	Sammy	Black	Yes	M	2200	mgrimes
10	Pinky	Red	No	M	1050	tswift
11	Hulk	Grey	No	M	2050	mgrimes
12	Pat	White	No	F	1400	mgrimes
13	Betty	White	Yes	F	1250	tswift
14	Shamrock	Black	No	M	1400	[NULL]

Same question answered using IN

<u>Using an uncorrelated subquery</u>:

Display the names of customers who own at least one horse.

```
SELECT username
FROM customers_old
WHERE username IN (SELECT owner FROM horses_old);
```

In this case the inner query executes only once, returns a list of usernames, then the outer query uses those values!

	USERNAME
1	canderson
2	jisbell
3	mgrimes
4	tswift

	USERNAME	FNAME	LNAME	PHONE
1	mgrimes	Marvin	Grimes	(218) 330-8004
2	canderson	Christine	Anderson	(555) 523-9989
3	tswift	Tina	Swift	(555) 424-1313
4	jisbell	Jason	Isbell	(615) 555-5555
5	ssimpson	Sam	Simpson	(615) 387-9682

	NAME	COLOR	SPOTS	SEX	WEIGHT	OWNER
1	Sam	Brown	No	F	1500	mgrimes
2	Erica	Yellow	Yes	F	920	canderson
3	John	Grey	No	M	1800	mgrimes
4	Trotty	Brown	Yes	M	1300	mgrimes
5	Rio	Grey	No	F	1700	tswift
6	Robin	Yellow	No	M	1100	jisbell
7	Katy	Brown	No	F	1200	jisbell
8	Pegasus	Brown	No	M	1750	mgrimes
9	Sammy	Black	Yes	M	2200	mgrimes
10	Pinky	Red	No	M	1050	tswift
11	Hulk	Grey	No	M	2050	mgrimes
12	Pat	White	No	F	1400	mgrimes
13	Betty	White	Yes	F	1250	tswift
14	Shamrock	Black	No	M	1400	[NULL]

The point of today…

- There are many ways to do the same thing
  - Sometimes there is no benefit to one way over another
  - Sometimes knowing multiple ways will save you

- Generally, relational algebra expressions can be translated into any number of other equivalent expressions

- Much like we did when creating ERDs – DECOMPOSE the query into each subquery or operation and UNDERSTAND what each piece is doing

Next week

- Next week we pause our discussion of SQL and start talking about normalization
- You will, of course, continue using SQL for the SQL project
- We will resume our discussion of SQL in two weeks
- Remember – no class on April 8!

Go forth and do great things

You have to practice, practice, practice:
cramming at exam time will not work!