Administrative Notes – February 2, 2023

- Feb 3: Assignment 2 (individual) due

- Feb 17: Assignment 3 (pairs if you want) due
  - Make sure you sign up in pairs beforehand as we need time to configure Canvas to allow both partners to view the submission and subsequent feedback
    - Everyone who has signed up as of last night has been put into a group on Canvas

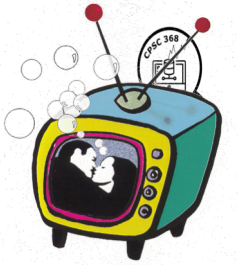
- Feb 20 – 24: Reading Break! (yay!)
  - No lectures, tutorials, or office hours during this week

CPSC 368
Databases in Data Science

Structured Query Language (SQL)

Textbook Reference
Database Management Systems: Chapter 5

Databases: The Continuing Saga



When last we left databases…

- We had decided they were great things
- We knew how to conceptually model them in ER diagrams
- We knew how to logically model them in the relational model

- Now: how do most people write queries? SQL!

Learning Goals

- Given the schemas of a relation, create SQL queries using: SELECT, FROM, WHERE, EXISTS, NOT EXISTS, UNIQUE, NOT UNIQUE, ANY, ALL, DISTINCT, GROUP BY and HAVING.
- Show that there are alternative ways of coding SQL queries to yield the same result. Determine whether or not two SQL queries are equivalent.
- Given a SQL query and table schemas and instances, compute the query result.
- Translate a query between SQL and RA.
- Comment on the relative expressive power of SQL and RA.
- Explain the purpose of NULL values and justify their use. Also describe the difficulties added by having nulls.
- Create and modify table schemas and views in SQL.
- Explain the role and advantages of embedding SQL in application programs.
- Write SQL for a small-to-medium sized programming application that requires database access.
- Identify the pros and cons of using general table constraints (e.g., CONSTRAINT, CHECK) and triggers in databases.

Coming up in SQL…

- Data Definition Language (reminder)
- Basic Structure
- Set Operations
- Aggregate Functions
- Null Values
- Nested Subqueries
- Modification of the Database
- Views
- Integrity Constraints
- Putting SQL to work in an application

The SQL Query Language

- Need for a standard since relational queries are used by many vendors
- Consists of several parts:
  - Data Definition Language (DDL)
    (a blast from the past (Chapter 3))
  - Data Manipulation Language (DML)
    - Data Query
    - Data Modification

Creating Tables in SQL(DDL) Revisited

- A SQL relation is defined using the **create table** command:

  create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

- *Integrity constraints can be:*
  - *primary and candidate keys*
  - *foreign keys*
- Example:

```
CREATE TABLE Student
        (sid     CHAR(20),
         name    CHAR(20),
         address CHAR(20),
         phone   CHAR(8),
         major   CHAR(4),
         PRIMARY KEY (sid))
```

Domain Types in SQL Reference Sheet

- **char(n).** Fixed length character string with length *n.*
- **varchar(n).** Variable length character strings, with maximum length *n.*
- **int.** Integer (machine-dependent).
- **smallint.** Small integer (machine-dependent).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

- Null values are allowed in all the domain types.
  To prohibit null values declare attribute to be **not null**
- **create domain** in SQL-92 and 99 creates user-defined domain types

  create domain *person-name* **char**(20) **not null**

Date/Time Types in SQL
Reference Sheet

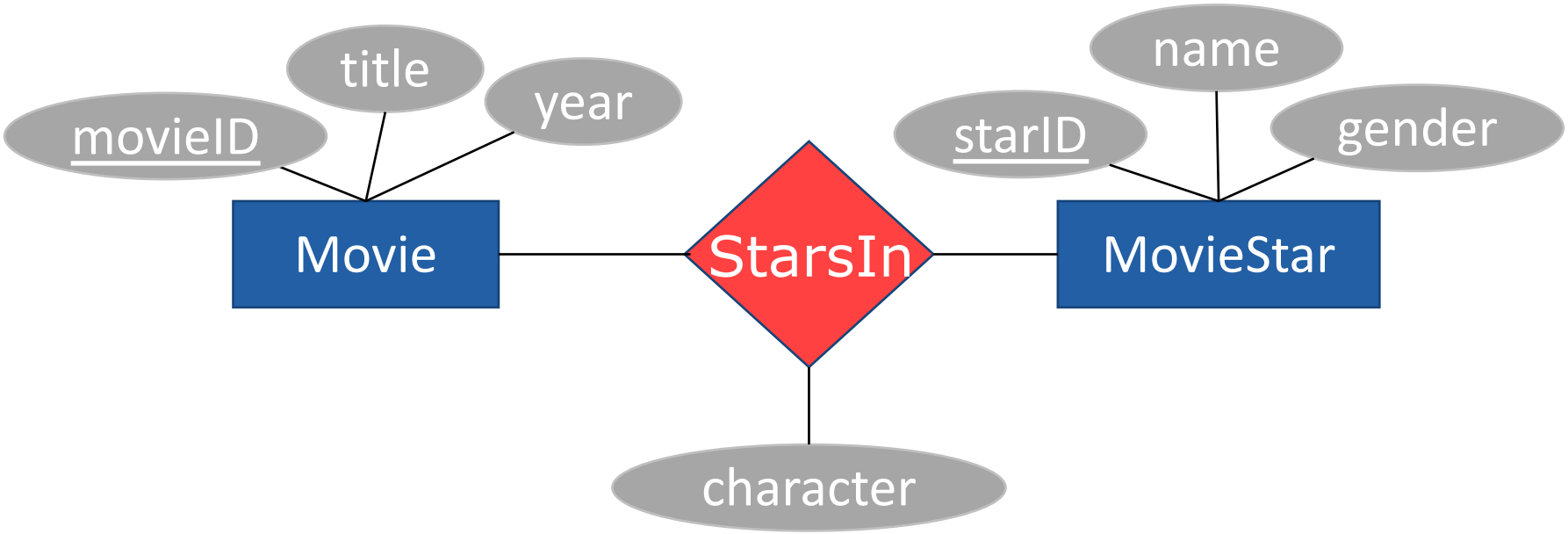
- **date.** Dates, containing a (4 digit) year, month and date
  - E.g. **date** '2001-7-27'
- **time.** Time of day, in hours, minutes and seconds.
  - E.g. **time** '09:00:30' **time** '09:00:30.75'
- **timestamp**: date plus time of day
  - E.g. **timestamp** '2001-7-27 09:00:30.75'
- **Interval**: period of time
  - E.g. Interval '1' day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values
- Relational DBMS offer a variety of functions to
  - extract values of individual fields from date/time/timestamp
  - convert strings to dates and vice versa
  - For instance in Oracle (date is a timestamp):
    - TO_CHAR(date, format)
    - TO_DATE(string, format)
    - format looks like: 'DD-Mon-YY HH:MI.SS'

Running Example (should look familiar)

Movie(MovieID, Title, Year)

StarsIn(MovieID, StarID, Character)

MovieStar(StarID, Name, Gender)



Basic SQL Query

- SQL is based on set and relational operations
- A typical SQL query has the form:

  SELECT $A_1, A_2, \ldots, A_n$
  FROM $r_1, r_2, \ldots, r_m$
  WHERE P

  - A_is represent attributes
  - r_is represent relations
  - P is a predicate.

SELECT	*target-list*
FROM	*relation-list*
WHERE	*qualification*

- The result of a SQL query is a table (relation)
- By default, duplicates are not eliminated in SQL relations, which are bags or multisets and not sets

Basic SQL Query

SELECT $A_1, A_2, \ldots, A_n$
FROM $r_1, r_2, \ldots, r_m$

SELECT	*target-list*
FROM	*relation-list*

- This is where we specify which column(s) we want in our answer
- If you want all the columns, use * (`SELECT *`)

Basic SQL Query

SELECT $A_1, A_2, \ldots, A_n$
FROM $r_1, r_2, \ldots, r_m$

SELECT	*target-list*
FROM	*relation-list*

- This is where we specify which table(s) you want to query
- Separate each table with a comma

Clicker Question: SQL projection

What is the result of:

```
SELECT Score1,Score2
FROM Scores
```

Which of the following rows is in the answer?

A. (1,2)
B. (5,3)
C. (8,6)
D. All are in the answer
E. None are in the answer

Scores

Team1	Team2	Score1	Score2
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12

Clicker Question: SQL projection

What is the result of:

```
SELECT Score1,Score2
FROM Scores
```

Which of the following rows is in the answer?

A. (1,2)
B. (5,3)
C. (8,6)
D. All are in the answer
E. None are in the answer

Scores

Team1	Team2	Score1	Score2
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12

clickerprojection.sql

Basic SQL Query

SELECT $A_1, A_2, \ldots, A_n$
FROM $r_1, r_2, \ldots, r_m$
WHERE P

SELECT	*target-list*
FROM	*relation-list*
WHERE	*qualification*

- What conditions do the tuples in your result have to adhere to?
  - ALL tuples in the result must meet the conditions listed in the WHERE
- Another way to think about it: what conditions will you use to rule out the tuples you don't want?

WHERE Clause

```
SELECT  *
FROM    Movie
WHERE   Year > 1939
```

You can use:

- attribute names of the relation(s) used in the FROM.
- comparison operators: =, <>, <, >, <=, >=
- apply arithmetic operations: rating*2
- operations on strings (e.g., "||" for concatenation).
- Lexicographic order on strings.
    - Pattern matching: s LIKE p
    - Special stuff for comparing dates and times.

Clicker Question: Selection

```
SELECT *
FROM Scores
WHERE RunsFor > 5
```

Which tuple is in the result?

A. (Swallows, Carp, 6, 4)
B. (Swallows, Carp, 4)
C. (12)
D. (*)

Scores

Team	Opponent	Runs For	Runs Against
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12
Tigers	Dragons	3	5
Swallows	Carp	6	4
Giants	Bay Stars	1	2
Hawks	Marines	3	5
Buffaloes	Ham Fighters	6	1
Golden Eagles	Lions	12	8

Clicker Question: Selection

```
SELECT *
FROM Scores
WHERE RunsFor > 5
```

Which tuple is in the result?

A. (Swallows, Carp, 6, 4)
B. (Swallows, Carp, 4)
C. (12)
D. (*)

clickerselection.sql

Scores

Team	Opponent	Runs For	Runs Against
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12
Tigers	Dragons	3	5
Swallows	Carp	6	4
Giants	Bay Stars	1	2
Hawks	Marines	3	5
Buffaloes	Ham Fighters	6	1
Golden Eagles	Lions	12	8

SELECT, FROM, WHERE

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

We can put these together:

- What are the names of female movie stars?

- What are the titles of movies from prior to 1939?

SELECT, FROM, WHERE

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

We can put these together:

- What are the names of female movie stars?

```
SELECT name
FROM MovieStar
WHERE Gender = 'female'
```

- What are the titles of movies from prior to 1939?

```
SELECT title
FROM Movie
WHERE year < 1939
```

Selection Example (Dates)

reserves

SID	BID	Day
22	101	2010-10-10
22	102	2010-10-10
22	103	2010-10-08
22	104	2010-07-10
31	102	2010-11-10
31	103	2010-11-06
31	104	2010-11-12
58	102	2010-11-08
58	103	2010-11-12

```
SELECT *
FROM reserves
WHERE day < DATE'2010-11-01'
```

SID	BID	Day
22	101	2010-10-10
22	102	2010-10-10
22	103	2010-10-08
22	104	2010-07-10

Joins in SQL

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

```
SELECT   Character, Name
FROM     StarsIn s, MovieStar m
WHERE    s.StarID = m.StarID
```

- Can alias relations (e.g., “StarsIn s”)
- Conditions specified in WHERE clause
- Joins are cross products

Joins in SQL

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

```
SELECT   Character, Name
FROM     StarsIn s, MovieStar m
WHERE    s.StarID = m.StarID
```

- Can alias relations (e.g., “StarsIn s”)
- Conditions specified in WHERE clause
- Joins are cross products

Cross Product (x)

- Produces every possible combination of tuples

R

A	B
0	0
0	1
1	0
1	1

S

C	D
0	0
0	1

R x S

A	B	C	D
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	0
0	0	0	1
0	1	0	1
1	0	0	1
1	1	0	1

Clicker Question: Simple Joins

```
SELECT R.a, R.b, S.a, S.b
FROM R, S
WHERE R.b = S.a
```

R

a	b
0	0
0	1
1	0
1	1

S

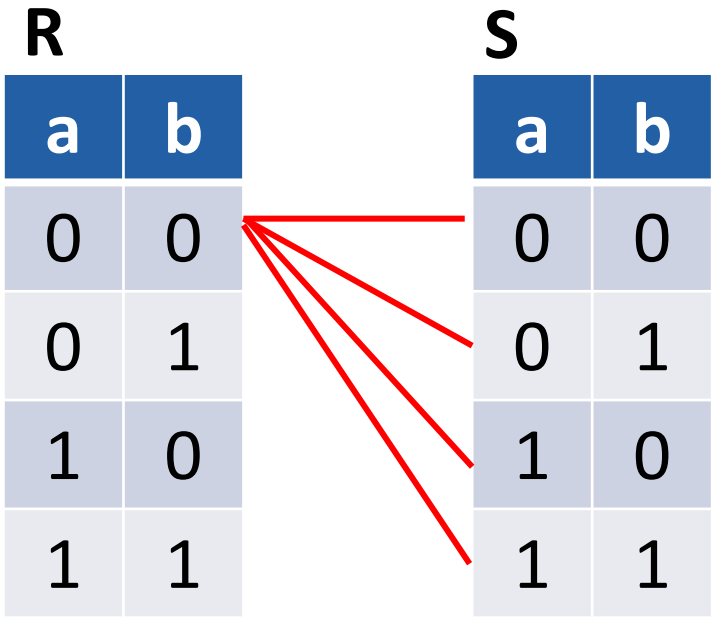
a	b
0	0
0	1
1	0
1	1

Which statement is true?

A. (0,1,1,0) appears twice.
B. (1,1,0,1) appears once.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

Clicker Question: Simple Joins

```
SELECT R.a, R.b, S.a, S.b
FROM R, S
WHERE R.b = S.a
```

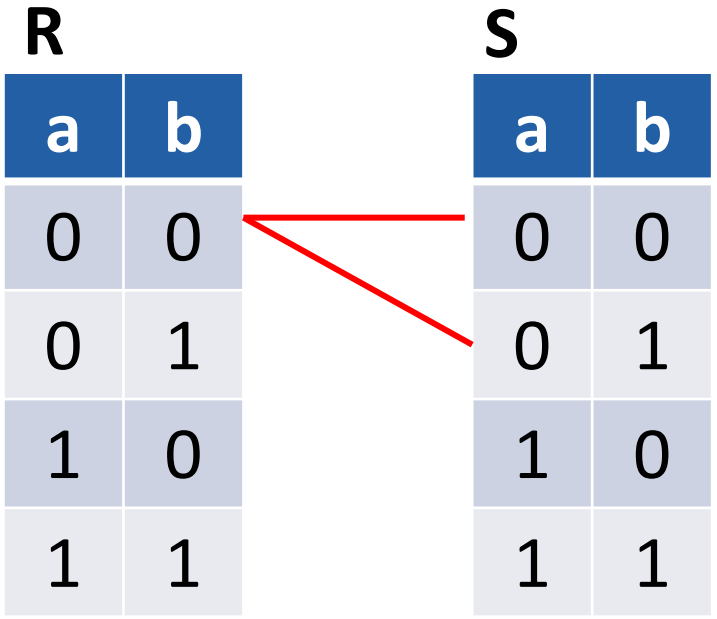


Which statement is true?

A. (0,1,1,0) appears twice.
B. (1,1,0,1) appears once.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

Clicker Question: Simple Joins

```
SELECT R.a, R.b, S.a, S.b
FROM R, S
WHERE R.b = S.a
```



Which statement is true?

A. (0,1,1,0) appears twice.
B. (1,1,0,1) appears once.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

Repeat this process for all tuples in R

Simple Joins Results

R.a	R.b	S.a	S.B
0	0	0	0
0	0	0	1
1	0	0	0
1	0	0	1
0	1	1	0
0	1	1	1
1	1	1	0
1	1	1	1

Clicker Question: Simple Joins

```
SELECT R.a, R.b, S.a, S.b
FROM R, S
WHERE R.b = S.a
```

R

a	b
0	0
0	1
1	0
1	1

S

a	b
0	0
0	1
1	0
1	1

Which statement is true?

A. (0,1,1,0) appears twice.

B. (1,1,0,1) appears once.

C. (1,1,1,0) appears once.

D. All are true

E. None are true

clickerjoineasy.sql

Clicker Question: Joins

```
SELECT R.a, R.b, S.b, T.b
  FROM R, S, T
  WHERE R.b = S.a AND S.b <> T.b
```

(note: <> == 'not equals')

Which of the following is true:

A. (0,1,1,0) appears twice.
B. (1,1,0,1) does not appear.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

R

a	b
0	0
0	1
1	0
1	1

S

a	b
0	0
0	1
1	0
1	1

T

a	b
0	0
0	1
1	0
1	1

R.a	R.b	S.b	T.b
0	0	0	1
0	0	0	1
0	0	1	0
0	0	1	0
0	1	0	1
0	1	0	1
0	1	1	0
0	1	1	0
1	0	0	1
1	0	0	1
1	0	1	0
1	0	1	0
1	1	0	1
1	1	0	1
1	1	1	0
1	1	1	0

Clicker Question: Joins

```
SELECT R.a, R.b, S.b, T.b
  FROM R, S, T
  WHERE R.b = S.a AND S.b <> T.b
```

(note: <> == 'not equals')

Which of the following is true:

A. (0,1,1,0) appears twice.
B. (1,1,0,1) does not appear.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

R

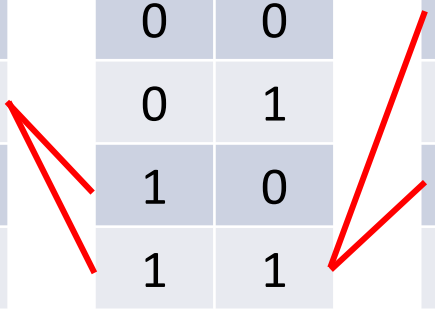
a	b
0	0
0	1
1	0
1	1

S

a	b
0	0
0	1
1	0
1	1

T

a	b
0	0
0	1
1	0
1	1



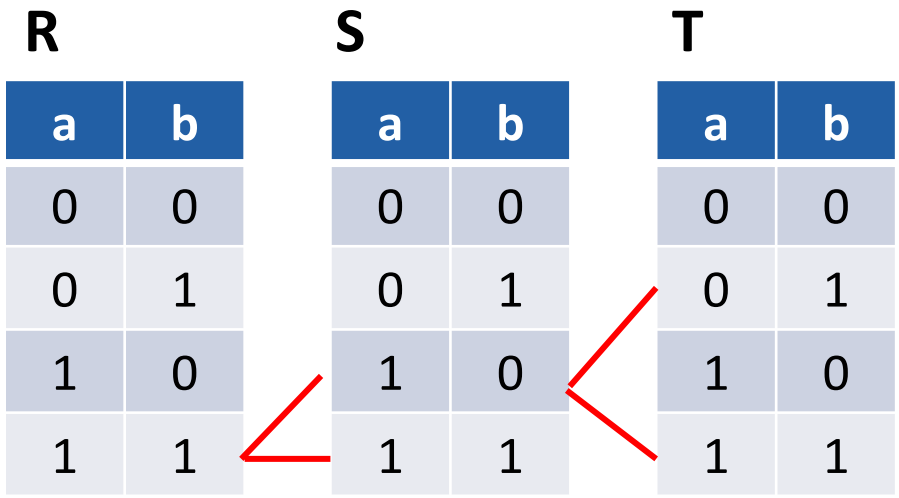
Clicker Question: Joins

```
SELECT R.a, R.b, S.b, T.b
  FROM R, S, T
  WHERE R.b = S.a AND S.b <> T.b
```

(note: <> == 'not equals')

Which of the following is true:

A. (0,1,1,0) appears twice.
B. (1,1,0,1) does not appear.
C. (1,1,1,0) appears once.
D. All are true
E. None are true



Clicker Question: Joins

```
SELECT R.a, R.b, S.b, T.b
  FROM R, S, T
  WHERE R.b = S.a AND S.b <> T.b
```

(note: <> == 'not equals')

Which of the following is true:

A. (0,1,1,0) appears twice.
B. (1,1,0,1) does not appear.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

R

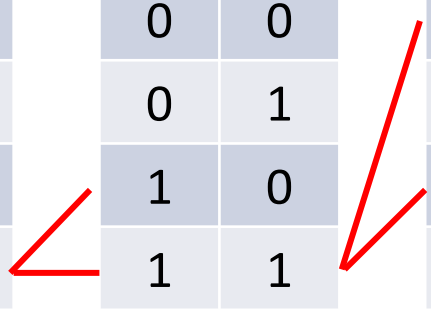
a	b
0	0
0	1
1	0
1	1

S

a	b
0	0
0	1
1	0
1	1

T

a	b
0	0
0	1
1	0
1	1



Clicker Question: Joins

```
SELECT R.a, R.b, S.b, T.b
  FROM R, S, T
  WHERE R.b = S.a AND S.b <> T.b
```

(note: <> == 'not equals')

Which of the following is true:

A. (0,1,1,0) appears twice.
B. (1,1,0,1) does not appear.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

R

a	b
0	0
0	1
1	0
1	1

S

a	b
0	0
0	1
1	0
1	1

T

a	b
0	0
0	1
1	0
1	1

Using DISTINCT

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

- Find the names of actors who have been in at least one movie

```
SELECT DISTINCT Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

- Would removing DISTINCT from this query make a difference?
- Note: on the exams, if we ask for a general question like “find all the names”, **we expect duplicates to be removed**.

Clicker Question

Consider the relation Scores(Team, Opponent, RunsFor, RunsAgainst) and the query:

```
SELECT DISTINCT Team,
                RunsFor
FROM Scores
```

Which is true:

A. 1 appears once
B. 5 appears twice
C. 6 appears 4 times
D. All are true
E. None are true

Team	Opponent	Runs For	Runs Against
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12
Tigers	Dragons	3	5
Swallows	Carp	6	4
Giants	Bay Stars	1	2
Hawks	Marines	3	5
Buffaloes	Ham Fighters	6	1
Golden Eagles	Lions	12	8

Clicker Question

clickerdistinction.sql

Consider the relation Scores(Team, Opponent, RunsFor, RunsAgainst) and the query:

```
SELECT DISTINCT Team,
                RunsFor
FROM Scores
```

Which is true:

A. 1 appears once
B. 5 appears twice
C. 6 appears 4 times
D. All are true
E. None are true

Team	Opponent	Runs For	Runs Against
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12
Tigers	Dragons	3	5
Swallows	Carp	6	4
Giants	Bay Stars	1	2
Hawks	Marines	3	5
Buffaloes	Ham Fighters	6	1
Golden Eagles	Lions	12	8

Join Example

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

- Find the names of all movie stars who have been in a movie

```
SELECT Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

Is this totally correct?

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

Harrison Ford will appear twice

Join Example

- Find the names of all movie stars who have been in a movie

```
SELECT Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

Is this totally correct?

- What if I run the following query?

```
SELECT DISTINCT Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

What if two movie stars had the same name?

```
SELECT DISTINCT StarID, Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

Error: Column StarID is ambiguous

Select Project Join Example

What are all the titles of movies with female actors?

```
SELECT DISTINCT Title
FROM Movie m, StarsIn s, MovieStar st
WHERE m.MovieID = s.MovieID and
      s.StarID = st.StarID and
      gender = 'female'
```

Renaming Attributes in Result

- SQL allows renaming relations and attributes using the **as** clause:

  old-name **as** *new-name*

- Example: Find the title of movies and the IDs of all actors in them, and rename "StarID" to "ID"

```
SELECT Title, StarID AS ID
FROM StarsIn S, Movie M
WHERE M.MovieID = S.MovieID
```

Congratulations: You know select-project-join queries

- Very common subset to talk about
  - You saw (or will see) it in tutorial
- Can do many (but not all) useful things

SQL is *declarative*, not procedural
how do we know? Lets see what procedural would look like…

In-Class Exercise (SQL 1)

- Canvas → Modules → In Class Exercises
- You can work on it with other people around you. If you work with others, you must **write their names on your submission to acknowledge the collaboration**.
  - Everyone must submit to Canvas
- Reminder: no late submissions accepted