

# SQL: Structured Query Language

INFO/CS 2300:  
Intermediate Web Design and  
Programming

# HW 1

Due this Tuesday 5pm.

Note that [evaluation.pdf](#) must be uploaded to CMS by the deadline as your signal to us that your work is complete and ready for grading.

# jQuery example: sticky menu

```
$(document).scroll(function() {  
    var y = $(document).scrollTop(), //get page y value  
    nav = $("#primary_nav"),  
    headerHeight = $('#cu-identity').outerHeight(),  
    adminBarHeight = $('#wpadminbar').outerHeight();  
    aboveHeight = headerHeight + adminBarHeight;  
  
    if((y >= aboveHeight) && screen.width > 939) {  
        nav.css({position: "fixed", "top" : adminBarHeight + "px"});  
    } else {  
        nav.css({position: "relative", "top" : "0"});  
    }  
});
```

<http://music.cornell.edu/people/faculty/>

<http://www.smashingmagazine.com/2012/09/11/sticky-menus-are-quicker-to-navigate/>

<http://jsfiddle.net/mariusc23/s6mLJ/31/>



# jQuery example: slideToggle

```
jQuery(document).ready(function($) {  
    //Respond to all-works-head.click  
    $(".all-works-head").click(function() {  
        //Toggle the display of the links  
        $(".all-works-body").slideToggle();  
  
    })  
  
    $(".all-works-body").hide();  
});
```



# jQuery example: CSS transition

```
$("#article.piece").on( 'mouseenter', function () {  
    $(this).addClass('hover');  
});  
$("#article.piece").on( 'mouseleave', function () {  
    $(this).removeClass('hover');  
});
```

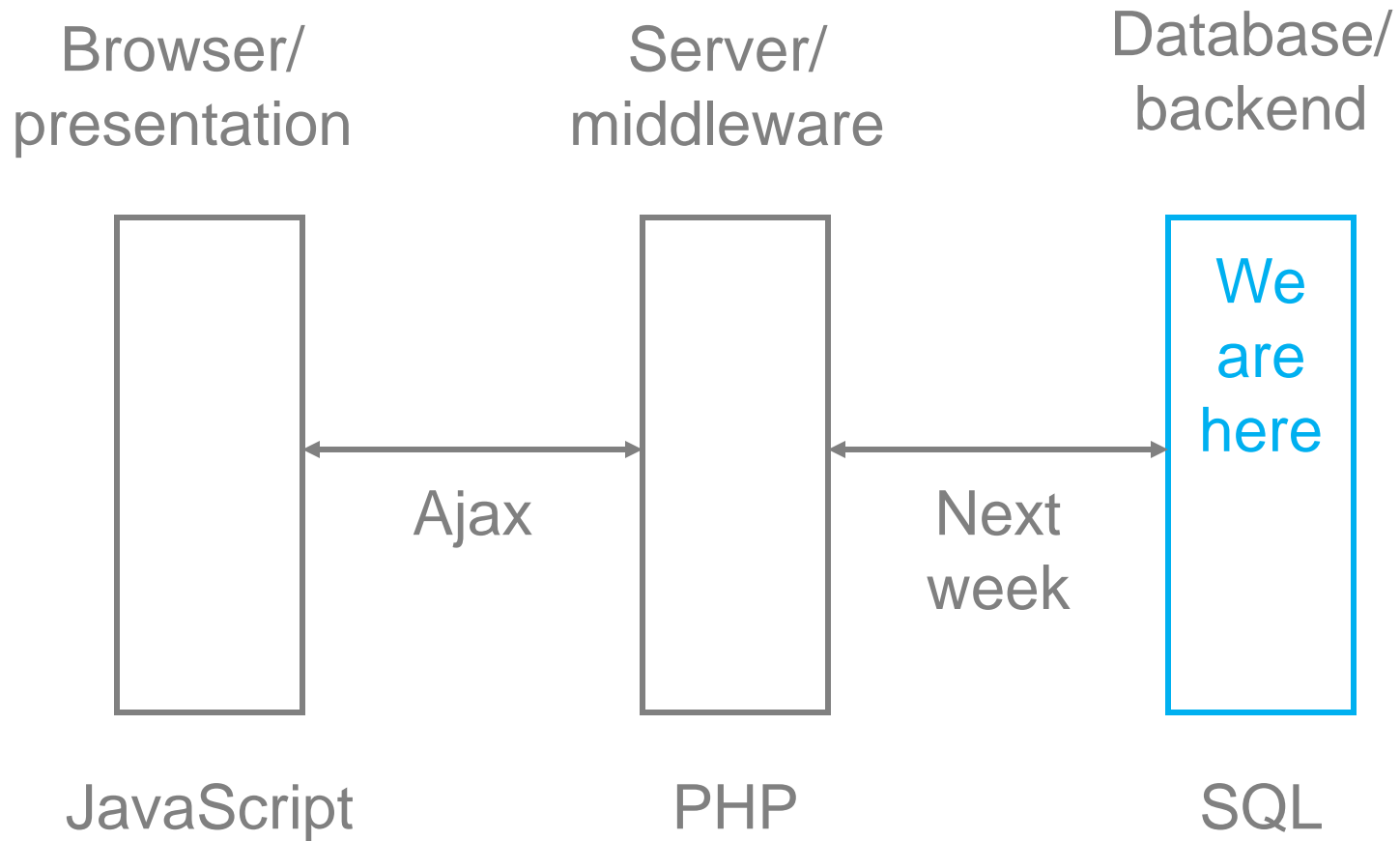
```
.hover .overlay {  
    height: 100%;  
    opacity: .8;  
}  
.overlay {  
    opacity: 0;  
    height: 0;  
    transition: all 0.25s;  
}
```

Even better: no JavaScript

```
parent:hover .overlay {  
    height: 100%;  
    opacity: .8;  
}  
.overlay {  
    opacity: 0;  
    height: 0;  
    transition: all 0.25s;  
}
```



# Course Overview



# Choosing primary keys

Students( NetID, FirstName, LastName)

Courses( Dept, Number, Time, Semester)

Registrations( NetID, Dept, Number, Semester)



Natural Keys

Alternatively

Students( NetID, FirstName, LastName)

Courses( CourseID, Dept, Number, Time, Semester)

Registrations( NetID, CourseID)



Surrogate Key

# SQL basics: selection and projection



## Movies

Title	Year	Length
Sleepless in Seattle	1993	105
Holiday	1938	95
The Philadelphia Story	1940	112
Sabrina	1954	113

Show me movies that are shorter than 100 minutes

```
SELECT *  
FROM MOVIES  
WHERE Length < 100
```

Title	Year	Length
Holiday	1938	95 mins.

# The basic SQL statement

```
SELECT Fields  
FROM Table  
WHERE Condition
```

```
SELECT *  
FROM MOVIES  
WHERE Length < 100
```



The \* means all fields

# Conditions

We can use the following conditions in the “WHERE” clause:

= (equals)

not == like in PHP

< (less than)

> (greater than)

<= (less than or equal)

>= (greater than or equal)

<> (not equal)

not != like in PHP

Conditions can use ( ) as well as “AND”, “OR”, and “NOT”.

# Selection

Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

E.g. `SELECT *`  
`FROM MOVIES`  
`WHERE Length > 150`

Title	Year	Length
Gladiator	2000	155
The Return of the King	2003	201

Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Show me movies that are longer than 150 minutes  
and were made before 2002

```
SELECT *
FROM MOVIES
WHERE (Length > 150)
      AND (Year < 2002)
```

Title	Year	Length
Gladiator	2000	155

Title	Year	Length
The Lion King	1994	89
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Show me movies that have “King” in the title

```
SELECT *
FROM MOVIES
WHERE (Title LIKE '%King%')
```

Title	Year	Length
The Lion King	1994	89
The Return of the King	2003	201

# Simple wildcards

LIKE uses special characters

% - percent matches 0 or more characters

\_ - underscore matches any single character

```
SELECT *  
FROM Movies  
WHERE Title LIKE '%King%'
```

# Regular expressions

Some variants of SQL (such as MySQL) allow regular expression matching.

```
SELECT *  
FROM Movies  
WHERE Title REGEXP '[aeou]t';
```

Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132



# Projection

Student( NetID, firstName, lastName, year,  
address, dateOfBirth, gpa, ... )

Give me just the NetID and last name of  
students named “Steve”

```
SELECT NetID, lastName  
FROM Student  
WHERE firstName = 'Steve'
```

When the query specifies only some of the  
fields it is called a **projection**.

# Distinct

Name	Title	Year
Russell Crowe	Gladiator	2000
Russell Crowe	A Beautiful Mind	2001
Viggo Mortensen	Return of the King	2003
Hillary Swank	Million Dollar Baby	2004

What actors are in my table?

```
SELECT Name  
FROM StarsIn
```

## Name

Russell Crowe
Russell Crowe
Viggo Mortensen
Hillary Swank

```
SELECT DISTINCT Name  
FROM StarsIn
```

## Name

Russell Crowe
Viggo Mortensen
Hillary Swank

Click In!

# Click In!

Which of the following is not a valid SQL statement?

- A. `SELECT * FROM movies  
WHERE year = 2002`
- B. `SELECT title WHERE year > 2002`
- C. `SELECT * FROM movies`
- D. `SELECT year, length FROM movies  
WHERE length <> 145`
- E. `SELECT title FROM movies WHERE  
title LIKE 'The_'`

# Click In!

Which of the following is not a valid SQL statement?

A. `SELECT * FROM movies  
WHERE year = 2002`

B. `SELECT title WHERE year > 2002`

C. `SELECT * FROM movies`

D. `SELECT year, length FROM movies  
WHERE length <> 145`

E. `SELECT title FROM movies WHERE  
title LIKE 'The_'`

# Click In!

id	first_name	last_name	age	subject
100	Rahul	Sharma	10	Science
101	Anjali	Bhagwat	12	Math
102	Stephen	Fleming	9	Science
103	Shekar	Gowda	18	Math
104	Priya	Chandra	15	Economics

What is the last name and subject of students 10 and older

SELECT \_\_\_\_\_ FROM students \_\_\_\_\_

A	last_name, subject	WHERE age >= 10
B	*	if age >= 10
C	last_name, subject	WHERE age > 10
D	*	age >=10
E	last_name, subject	if age >= 10

# Click In!

id	first_name	last_name	age	subject
100	Rahul	Sharma	10	Science
101	Anjali	Bhagwat	12	Math
102	Stephen	Fleming	9	Science
103	Shekar	Gowda	18	Math
104	Priya	Chandra	15	Economics

What is the last name and subject of students 10 and older

SELECT \_\_\_\_\_ FROM students \_\_\_\_\_

A	last_name, subject	WHERE age >= 10
B	*	if age >= 10
C	last_name, subject	WHERE age > 10
D	*	age >=10
E	last_name, subject	if age >= 10

More than one table



## Movies

## StarsIn

Title	Year
Gladiator	2000
A Beautiful Mind	2001

Name	Title
Russell Crowe	Gladiator
Viggo Mortensen	Return of the King
Hillary Swank	Million-Dollar Baby

```
SELECT *
FROM MOVIES, StarsIn
```

Movies.Title	Year	Name	StarsIn.Title
Gladiator	2000	Russell Crowe	Gladiator
A Beautiful Mind	2001	Russell Crowe	Gladiator
Gladiator	2000	Viggo Mortensen	Return of the King
A Beautiful Mind	2001	Viggo Mortensen	Return of the King
Gladiator	2000	Hillary Swank	Million-Dollar Baby
A Beautiful Mind	2001	Hillary Swank	Million-Dollar Baby

# INNER JOIN

Students( [NetID](#), FirstName, LastName)

Courses( CourseID, Dept, Number, Time, Semester)

Registrations( [NetID](#), CourseID)

How do we connect Students and Registrations so that only related data is in the result set?

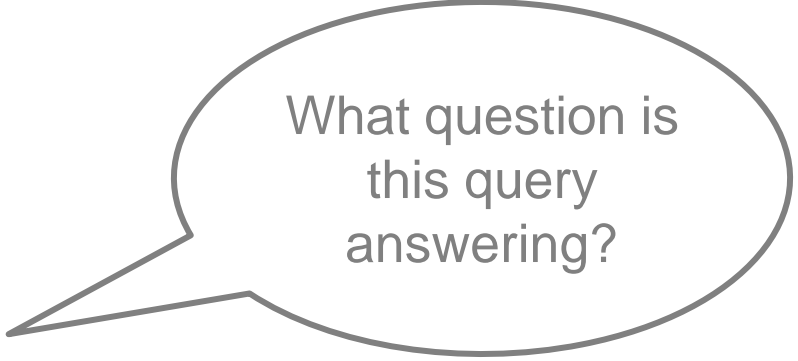
# INNER JOIN

Students( NetID, FirstName, LastName)

Courses( CourseID, Dept, Number, Time, Semester)

Registrations( NetID, CourseID)

```
SELECT
    Students.NetID,
    Students.FirstName
FROM
    Students INNER JOIN Registrations
        ON Students.NetID = Registrations.NetID
WHERE CourseID = 12345;
```



What question is  
this query  
answering?

# Inner join 2 fields

Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Name	Title	Year
Russell Crowe	Gladiator	2000
Russell Crowe	A Beautiful Mind	2001
Viggo Mortensen	Return of the King	2003
Hillary Swank	Million Dollar Baby	2004

```
SELECT *  
FROM  
    Movies INNER JOIN StarsIn  
        ON Movies.Title = StarsIn.Title  
        AND Movies.Year = StarsIn.Year  
WHERE  
    Length > 150;
```

# Inner join results

Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201

Name	Title	Year
Russell Crowe	Gladiator	2000
Russell Crowe	A Beautiful Mind	2001
Viggo Mortensen	Return of the King	2003

Movies.Title	Movies.Year	Length	Name	StarsIn.Title	StarsIn.Year
Gladiator	2000	155	Russell Crowe	Gladiator	2000
A Beautiful Mind	2001	135	Russell Crowe	A Beautiful Mind	2001
The Return of the King	2003	201	Viggo Mortensen	Return of the King	2003

# INNER JOIN 3 tables

students( NetID, FirstName, LastName)

courses( CourseID, Dept, Number, Time, Semester)

registrations( NetID, CourseID)

```
SELECT
```

```
    students.NetID,  
    students.FirstName,  
    courses.Dept
```

```
FROM registrations
```

```
    INNER JOIN students
```

```
        ON registrations.NetID = students.NetID
```

```
    INNER JOIN courses
```

```
        ON registrations.CourseID = courses.CourseID
```

```
WHERE courses.Dept = 12345
```

# Alias

```
SELECT
    s.NetID,
    s.FirstName
FROM registrations r
    INNER JOIN students s
        ON r.NetID = s.NetID
WHERE r.CourseID = 12345
```

The “s” is an **alias** for Students and “r” for Registrations

Steve’s opinion: usually makes the SQL harder to read;  
we have to translate in our minds

# Aggregation

We can aggregate results of a given field across all records of a table.

**SUM** – sums a field with numerical value

**AVG** – averages a field with numerical values

**MIN, MAX** – produces minimum, maximum of a field (either for numbers or strings)

**COUNT** – counts the number of records



Title	Year	Length
Gladiator	2000	155
A Beautiful Mind	2001	135
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
SELECT MAX (Length)
FROM Movies;
```

**MAX(Length)**

201

```
SELECT AVG (Length) as Average
FROM Movies;
```

**Average**

147.2

```
SELECT COUNT (*)
FROM Movies
WHERE Year >= 2002;
```

**Count(\*)**

3

Alias for field

# Distinct

Name	Title	Year
Russell Crowe	Gladiator	2000
Russell Crowe	A Beautiful Mind	2001
Viggo Mortensen	Return of the King	2003
Hillary Swank	Million Dollar Baby	2004

Remember this?

```
SELECT DISTINCT Name  
FROM StarsIn
```

Name
Russell Crowe
Viggo Mortensen
Hillary Swank

```
SELECT COUNT(DISTINCT Name)  
FROM StarsIn
```

**Count(DISTINCT Name)**

3

Now you try...

## List all hotel names

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

## List all hotel names

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

```
SELECT hotelName  
FROM Hotel;
```

List the names of all hotels that have single rooms with a price below \$100

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

List the names of all hotels that have single rooms with a price below \$100

Hotel (hotelNo, **hotelName**, city)

Room (roomNo, hotelNo, **type**, **price**)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

SELECT

**hotelName**

FROM

WHERE

**type = 'single'**

**AND price < 100;**

List the names of all hotels that have single rooms with a price below \$100

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

SELECT

hotelName

FROM

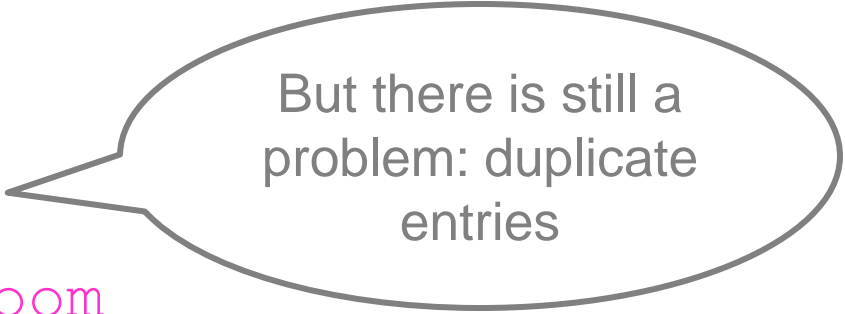
Hotel INNER JOIN Room

ON Hotel.hotelNo = Room.hotelNo

WHERE

type = 'single'

AND price < 100;



But there is still a  
problem: duplicate  
entries



List the names of all hotels that have single rooms with a price below \$100

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

```
SELECT DISTINCT
```

```
    hotelName
```

```
FROM
```

```
    Hotel INNER JOIN Room
```

```
        ON Hotel.hotelNo = Room.hotelNo
```

```
WHERE
```

```
    type = 'single'
```

```
    AND price < 100;
```



Eliminate duplicate entries

# What is the average price of a room?

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom,  
dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

# What is the average price of a room?

```
SELECT
    AVG( price )
FROM
    Room;
```

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom,  
dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

List the room number, price and type of all rooms at the Statler

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom,  
dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

# List the room number, price and type of all rooms at the Statler

Hotel (hotelNo, hotelName, city)  
Room (roomNo, hotelNo, type, price)  
Booking (hotelNo, guestNo, dateFrom,  
dateTo, roomNo)  
Guest (guestNo, guestName, guestAddress)

SELECT

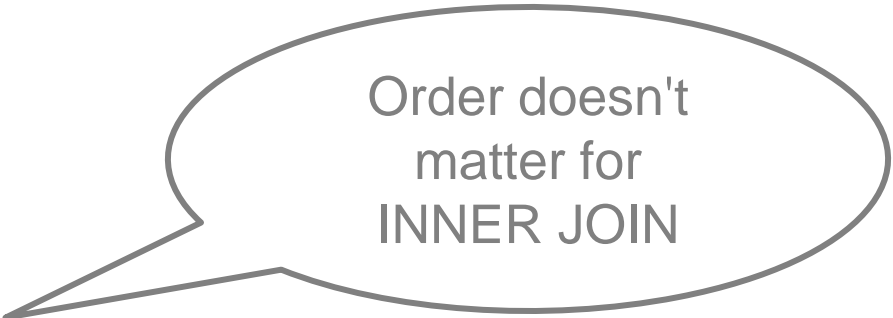
Room.roomNo,  
Room.price,  
Room.type

FROM

Hotel INNER JOIN Room  
ON Hotel.hotelNo = Room.hotelNo

WHERE

Hotel.hotelName = 'Statler'



Order doesn't  
matter for  
INNER JOIN

# List all guests currently staying at the Statler

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom,  
dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

# List all guests currently staying at the Statler

```
SELECT      Hotel (hotelNo, hotelName, city)
            Room (roomNo, hotelNo, type, price)
            Booking (hotelNo, guestNo, dateFrom,
                    dateTo, roomNo)
FROM        Guest (guestNo, guestName, guestAddress)

            Booking

            INNER JOIN Hotel
                ON Booking.hotelNo = Hotel.hotelNo
            INNER JOIN Guest
                ON Booking.guestNo = Guest.guestNo
WHERE
            Hotel.hotelName = 'Statler'
            AND Booking.dateFrom <= CURDATE()
            AND Booking.dateTo >= CURDATE()
```

# Review

- SQL is the standard language for asking queries in a relational DBMS.

HW1 Due Tuesday 5 pm – Remember CMS