INFSCI 2710
Database Management

Today's Plan

- More SQL
- Aggregate functions
- Even More SQL
- Working with string data types
- Joining multiple tables with SQL
- More SQL…

Homework #1

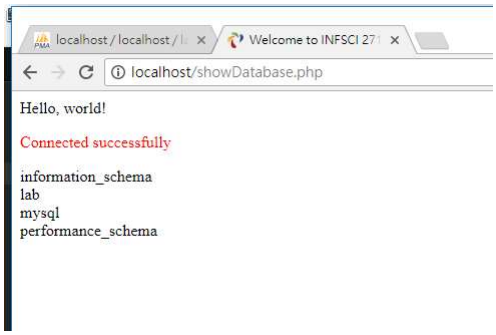
- Q1: you need to add new primary key.
- Q2: I accept almost all answer with explanation.
- Q3: you will be deducted some points due to
  - No primary key in your table.
  - No meaning relation (e.g., a 1:m relation with no key, an FK with no corresponding attributes in other tables)
  - No meaning table (e.g., list a PK but no other columns)
  - Cross reference (e.g., PK/FK refer to each other, which is a unique-key violation) -> combine PH/FK in one attribute is not a right way, but I didn't deduct the points.
  - No field/attribute type (e.g., Int(20))
  - "Many to many" issue: you cannon simply connect two tables with m:n relation, check this post for more information.

Homework Grading

- Homework: 20%
  - Homework 1: 4%
  - *Homework 2: 6%
  - Homework 3: 4%
  - Homework 4: 4%
  - *Homework 5: 6%
- You will have 4% extra credit to make up.

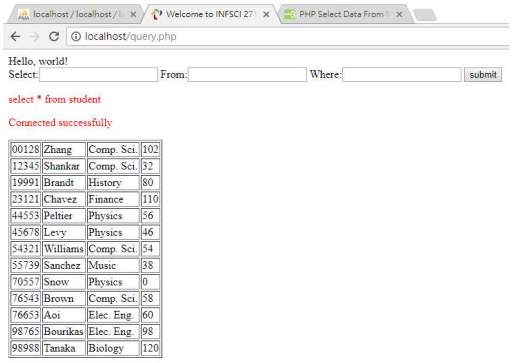
Show database using PHP

- Create showDatabase.php
- Check http://localhost/showDatabase.php for testing
  - All files/URLs are case-sensitive



Select Database using PHP

- Create 1) query.php ; 2) queryForm.html
- Check http://localhost/queryForm.html for testing



00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Homework#2

- Complete homework posted on CourseWeb
  - The lab session
  - Select student record
  - Add (Insert) student record
  - Delete student record
  - Edit (Update) student record
- Due on Feb 6

Delete	Edit	00128	Zhang	Comp. Sci.	102
Delete	Edit	12345	Shankar	Comp. Sci.	32
Delete	Edit	19991	Brandt	History	80
Delete	Edit	23121	Chavez	Finance	110
Delete	Edit	44553	Peltier	Physics	56
Delete	Edit	45678	Levy	Physics	46
Delete	Edit	54321	Williams	Comp. Sci.	54
Delete	Edit	55739	Sanchez	Music	38
Delete	Edit	70557	Snow	Physics	0
Delete	Edit	76543	Brown	Comp. Sci.	58
Delete	Edit	76653	Aoi	Elec. Eng.	60
Delete	Edit	98765	Bourikas	Elec. Eng.	98
Delete	Edit	98988	Tanaka	Biology	120

Homework#2

- #1881 - Operation not allowed when innodb_forced_recovery > 0.
  - Update your my.cnf file to solve this issue.
- Questions?

Query Clauses

Clauses - constituent components of statements and queries.

- **FROM**
- **WHERE**
- GROUP BY
- HAVING
- ORDER BY
- LIMIT

ORDER OF CLAUSES

- CLAUSES must appear in the following order
  - **FROM**
  - **WHERE**
  - **GROUP BY**
  - **HAVING**
  - **ORDER BY**
  - **LIMIT**
- Not all clauses must appear in a query – **FROM** clause is the only one that's required

FROM

- Indicates the table(s) from which data is to be retrieved.
- The FROM clause can include optional JOIN subclauses to specify the rules for joining tables.

SELECT * **FROM Employees**

WHERE

- Includes a comparison predicate, which restricts the rows returned by the query.
- The WHERE clause eliminates all rows from the result set for which the comparison predicate does not evaluate to True.

SELECT * FROM Employees
WHERE lastName = 'Smith'

Operators

Operator	Description	Example
=	Equal to	Author = 'Alcott'
<>	Not equal to (most DBMS also accept != instead of <>)	Dept <> 'Sales'
>	Greater than	Hire_Date > '2012-01-31'
<	Less than	Bonus < 50000.00
>=	Greater than or equal	Dependents >= 2
<=	Less than or equal	Rate <= 0.05
BETWEEN	Between an inclusive range	Cost BETWEEN 100.00 AND 500.00
LIKE	Match a character pattern	First_Name LIKE 'Will%'
IN	Equal to one of multiple possible values	DeptCode IN (101, 103, 209)
IS *or* IS NOT	Compare to null (missing data)	Address IS NOT NULL

LIKE + WILDCARDS

- LIKE statement allows you to search for matches within character fields.
- % (percent) is a wildcard

SELECT * FROM Employees
WHERE lastName **LIKE '%Sm'**;

LIKE + WILDCARDS

- WHERE lastName **LIKE 'Sm%'** – find all records where the value of lastName **begins** with **sm**
- WHERE lastName **LIKE '%th'** – find all records where the value of lastName **ends** with **th**
- WHERE lastName **LIKE '%sm%'** – find all records where the value of lastName **contains** character sequence **sm** anywhere in value

LIMIT

- Limits the number of records (table rows) returns by an SQL query
- Always the last clause in the query
- Note that LIMIT is specific to MySQL and Oracle – might not work with other database systems

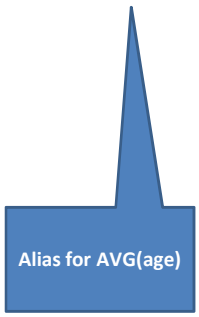
SELECT * FROM Employees WHERE lastName = 'Smith'
LIMIT 5;

Aggregate Functions

- An aggregate function performs a calculation on a set of values and returns a single value.
- Most common MySQL aggregate functions are
  - AVG
  - COUNT
  - SUM
  - MIN
  - MAX

AVG(expression)

SELECT AVG(age) averagePatientAge **FROM** Patients



http://www.mysqltutorial.org/mysql-avg/

COUNT Function

```
SELECT COUNT(*) patientCount
FROM Patients
WHERE patientAge > 10
```

http://www.mysqltutorial.org/mysql-count/

SUM Function

```
SELECT SUM(medicationPrice)
FROM Prescription p JOIN Medication m
ON p.medicationID = m.medicationID
WHERE patientID = 5
```

http://www.mysqltutorial.org/mysql-sum/

MAX Function

SELECT MAX(medicationPrice)
FROM Medication

http://www.mysqltutorial.org/mysql-max-function/

MIN Function

SELECT MIN(medicationPrice)
FROM Medication

http://www.mysqltutorial.org/mysql-min/

SQL Joins

Aggregate functions review

GROUP BY Clause

HAVING clause

GROUP BY

- Used to combine rows having common values into a smaller set of rows.
- GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a resultset.
- The WHERE clause is applied before the GROUP BY clause.

Table: patient_visit

visitID (pk)	fk_patientID (fk)	weight	BPS	BPD	OSAT
1	1	150	140	90	98
2	3	178	127	75	94
3	3	170	125	70	97
4	3	140	130	81	92
5	7	220	160	100	99
6	1	148	148	95	96
7	3	165	125	72	94
8	7	148	161	98	98
9	1	152	143	88	96

I want to know the average weight of each patient

fk_patientID	AVERAGE(weight)
1	150
3	163.25
7	184

I want to know the average weight of each patient

fk_patientID	AVERAGE(weight)
1	150
3	163.25
7	184

SELECT fk_patientID, **AVG(weight)**
FROM patient_visit
GROUP BY fk_patientID

SQL Joins
Aggregate functions review
GROUP BY Clause
HAVING clause

HAVING

- Used to filter rows resulting from the GROUP BY clause.
- Limits results returned by the GROUP BY clause
- Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.

Table: patient_visit

visitID (pk)	fk_patientID (fk)	weight	BPS	BPD	OSAT
1	1	150	140	90	98
2	3	178	127	75	94
3	3	170	125	70	97
4	3	140	130	81	92
5	7	220	160	100	99
6	1	148	148	95	96
7	3	165	125	72	94
8	7	148	161	98	98
9	1	152	143	88	96

I want to know the average weight of each patient whose average weight is greater than 150 lb

fk_patientID	AVERAGE(weight)
3	163.25
7	184

I want to know the average weight of each patient

fk_patientID	AVERAGE(weight)
3	163.25
7	184

SELECT fk_patientID, **AVG(weight)**
FROM patient_visit
GROUP BY fk_patientID
HAVING AVG(weight) > 150

String Functions

- Most DBMS provide useful functions that allow us to manipulate strings
- These functions may vary from DBMS to DBMS
- MySQL – specific string functions: http://dev.mysql.com/doc/refman/5.0/en/string-functions.html

CONCAT()

- Combines multiple strings into one

SELECT **CONCAT**('My', ' name', ' is', ' John');

Will return **My name is John**

CONCAT()

SELECT **CONCAT**(firstName, ' ', lastName)
FROM employees;

will return the value of firstName field followed by a space, followed by the value of lastName field

LCASE() and LOWER()

- LOWER(str) - returns the string str with all characters changed to lowercase

SELECT LOWER('HI, HOW ARE YOU?')

returns hi, how are you?

LCASE() and LOWER()

SELECT LOWER(firstName)
FROM employees;

returns the value of firstName field with all characters changed to lowercase

UCASE() and UPPER()

- UPPER(str) - returns the string str with all characters changed to uppercase

SELECT UPPER('hi, how are you?')
returns HI, HOW ARE YOU?

UCASE() and UPPER()

SELECT UPPER(firstName)
FROM employees;

returns the value of firstName field with all characters changed to uppercase

TRIM(), LTRIM(), RTRIM()

- TRIM() – removes preceding and trailing spaces from a string
- LTRIM() – removes preceding spaces from a string
- RTRIM() – removes trailing spaces from a string

REPLACE()

- REPLACE(str,from_str,to_str) - returns the string **str** with all occurrences of the string **from_str** replaced by the string **to_str**.
- REPLACE() performs a **case-sensitive** match when searching for from_str.

SELECT **REPLACE**('www.mysql.com', 'w', 'Ww');
returns 'WwWwWw.mysql.com'

SUBSTRING()

- SUBSTRING(str,pos) – returns a section of string **str** starting from position **pos** and going to the end of string **str**.

SELECT **SUBSTRING**('This is a test', 5);
returns **'is a test'**

SUBSTRING()

- SUBSTRING(str,pos,len) – returns a section of string **str** starting from position **pos** and going for a number of characters specified by **len**.

SELECT **SUBSTRING**('This is a test', 5, 4);

returns **'is a'**

SQL JOINS

Users
userID **(pk)**
lastName
firstName
dateOfBirth
ssn

Address
addressID **(pk)**
fk_userID **(fk)**
streetAddress
city
state
zip

SQL JOINS

userID	lastName	firstName	dateOfBirth	ssn
1	Doe	John	04/01/2001	111-11-1111
2	Brown	Michael	01/02/1986	222-22-2222
3	Green	Evelyn	03/14/1976	333-33-3333

addressID	fk_userID	streetAddress	city	state	zip
1335235	1	101 Phillips Avenue	Pittsburgh	PA	15217
5436543	1	325 Hobart Street	Pittsburgh	PA	15217
3675476	3	722 Darlington Avenue	Pittsburgh	PA	15217

SQL JOINS

lastName	firstName	streetAddress	city	state	zip
Doe	John	101 Phillips Avenue	Pittsburgh	PA	15217
Doe	John	325 Hobart Street	Pittsburgh	PA	15217
Green	Evelyn	722 Darlington Avenue	Pittsburgh	PA	15217

SQL JOINS

userID	lastName	firstName	dateOfBirth	ssn
1	Doe	John	04/01/2001	111-11-1111
2	Brown	Michael	01/02/1986	222-22-2222
3	Green	Evelyn	03/14/1976	333-33-3333

addressID	fk_userID	streetAddress	city	state	zip
1335235	1	101 Phillips Avenue	Pittsburgh	PA	15217
5436543	1	325 Hobart Street	Pittsburgh	PA	15217
3675476	3	722 Darlington Avenue	Pittsburgh	PA	15217

SQL JOINS

userID (pk)	lastName	firstName	dateOfBirth	ssn
1	Doe	John	04/01/2001	111-11-1111
2	Brown	Michael	01/02/1986	222-22-2222
3	Green	Evelyn	03/14/1976	333-33-3333

addressID	fk_userID (fk)	streetAddress	city	state	zip
1335235	1	101 Phillips Avenue	Pittsburgh	PA	15217
5436543	1	325 Hobart Street	Pittsburgh	PA	15217
3675476	3	722 Darlington Avenue	Pittsburgh	PA	15217

JOIN – Concatenating Data From Multiple Tables

Fields from ***users*** table

Fields from ***addresses*** table

```
SELECT lastName, firstName, streetAddress, city, state, zip
FROM emr.Users JOIN emr.Address
ON userID = fk_userID;
```

Primary key from **users** table

Foreign key from **addresses** table

Prefixing table name with the schema name

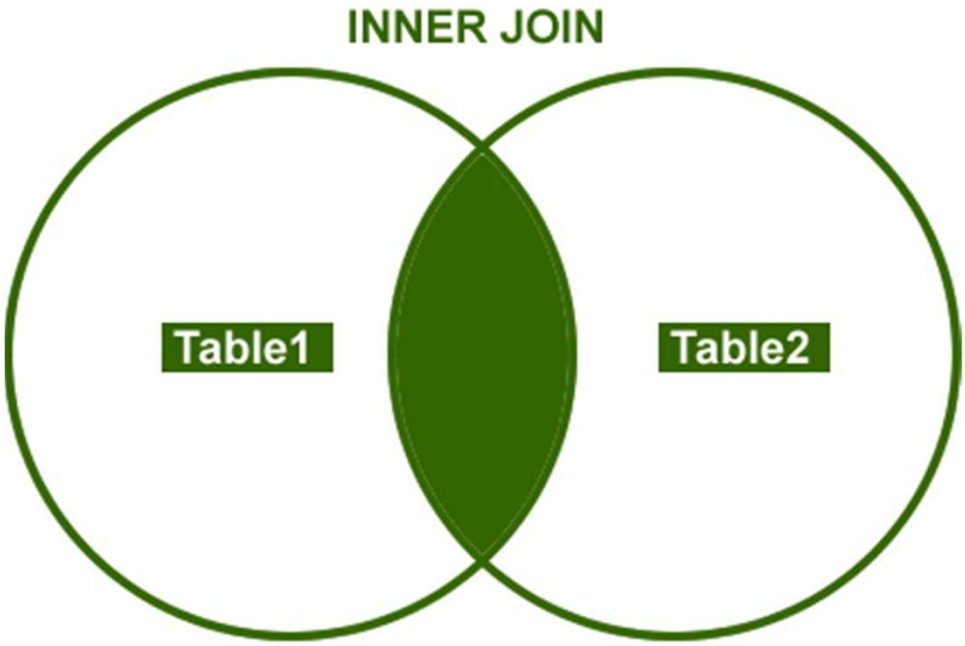
INNER JOIN

- This join returns rows when there is at least one match in both the tables.
- Inner join is the default join in SQL language.

userID	lastName	firstName	dateOfBirth	ssn
1	Doe	John	04/01/2001	111-11-1111
2	Brown	Michael	01/02/1986	222-22-2222
3	Green	Evelyn	03/14/1976	333-33-3333

addressID	fk_userID	streetAddress	city	state	zip
1335235	1	101 Phillips Avenue	Pittsburgh	PA	15217
5436543	1	325 Hobart Street	Pittsburgh	PA	15217
3675476	3	722 Darlington Avenue	Pittsburgh	PA	15217

http://blog.sqlauthority.com/2009/04/13/sql-server-introduction-to-joins-basic-of-joins/



SQL JOINS

Users
userID (pk)
lastName
firstName
dateOfBirth
ssn

Address
addressID
userID (fk)
streetAddress
city
state
zip

TABLE ALIASES

SELECT lastName, firstName, streetAddress, city, state, zip
FROM emr.Users INNER JOIN emr.Address
ON userID = userID;

Note that these two columns have same names. How would SQL interpreter know which column belongs to which table?

TABLE ALIASES

SELECT lastName, firstName, streetAddress, city, state, zip
FROM emr.Users INNER JOIN emr.Address
ON **Users**.userID = **Address**.userID;

We can prefix columns that have identical names with names of the tables to which they belong.

TABLE ALIASES

SELECT lastName, firstName, streetAddress, city, state, zip
FROM emr.Users **u** INNER JOIN emr.Address **a**
ON **u**.userID = **a**.userID;

Or, because typing sucks, we can create aliases.

TABLE ALIASES

Note that you can use table aliases in SELECT clause

SELECT **u.**lastName, **u.**firstName,
a.streetAddress, **a**.city, **a**.state, zip
FROM emr.Users **u** INNER JOIN emr.Address **a**
ON **u**.userID = **a**.userID;

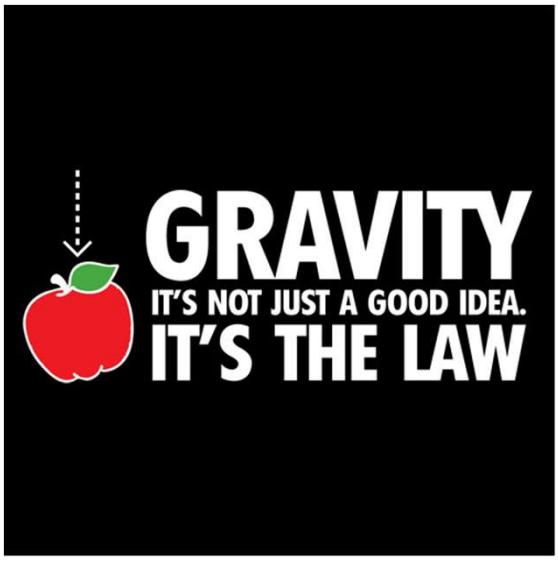
AIN'T NO DIFFERENCE

```
SELECT lastName, firstName, streetAddress, city, state, zip
FROM Users INNER JOIN Address
ON userID = fk_userID;

SELECT lastName, firstName, streetAddress, city, state, zip
FROM Users JOIN Address
ON userID = fk_userID;
```

AIN'T NO DIFFERENCE

```
SELECT lastName, firstName, streetAddress, city, state, zip
FROM Users JOIN Address
ON userID = fk_userID;

SELECT lastName, firstName, streetAddress, city, state, zip
FROM Users, Address
WHERE userID = fk_userID;
```

FOLLOW THE RULES



JOIN RULES

- You **should** join table on primary/foreign key relationships
- Fields on which you are joining must be of the same data type
- When joining on character fields, they do not necessarily have to be of the same length

TABLE ALIASES VS COLUMN ALIASES

```
SELECT CONCAT(lastName, ‘, ‘, firstName) AS userName,
streetAddress AS userAddress, city, state, zip
FROM emr.Users u JOIN emr.Address a
ON u.userID = a.userID;
```

Column alias

Table alias

SQL Practice

- There are three buildings in the Lab DB; please create a SQL statement to show…
  - the total student capacity in each building.
  - The total student capacity of the building has more than 100 seats
  - the number of offering courses in each building.
- String functions
  - Query the students whose name contains ‘an’ and order by grade
  - As Above, but replace ‘an’ as ‘xx’
  - As above, but convert the name as uppercase
  - As above, but contact the name with @ dept_name

SQL Practice & Visual Analytic

- Show the score distribution of students
  - As above, but with student name + @ + dept name
- Show the average score distribution of departments
  - As above, but with low performance with average score < 50
- Show the average (mentee) score distribution of advisors
  - As above, but with the ratio of sum score and their salary.

Sample Code and SQLs

- https://github.com/hymanct/infsci2710

Project

- You are to analyze the requirements for, design, implement, document and demonstrate a database system that could automate the functions of a sales company with E-Commerce.
- Through the project, you will gain hands-on experience in designing and implementing a *Data-Intensive* Application.
- **The project is done in groups of 3 students and is documented by a written group report.**
- More details will be provided next week.

Quiz

- ~30 mins, In class (Feb 6)
- Closed-book
- What's covered?
  - Lecture 3 & 4
  - All mentioned SQL statements.
  - Be prepared to **write** down your SQL.

Questions? ☺