Sign up for cloud based IDE - Text lecture

Section 1, Lecture 5

To create an account at cloud9, go to c9.io and sign-up

Select php, apache as your workspace type

To install MySQL, type the following in your workspace directory and hit return -> mysql-ctl install  
To start the MySQL server, type the following and hit return -> mysql-ctl start  
To open a command line interface with direct access to your mysql database type in the following and hit return -> mysql-ctl cli

To see the hostname, type in the following and hit return -> select @@hostname;  
\*\*\*\* Don't forget the semi-colon at the end! \*\*\*\*

To show databases -> show databases;

To exit the mysql cli -> exit

Command line crash course online -> <http://cli.learncodethehardway.org/book/>

Create a Database/Schema - Text lecture

Section 2, Lecture 8

Database Server: Service running on a machine (server or desktop), it houses many databases.

Database: Contains all of the persistent information for your applications

Schema: In MySQL, this is synonymous with database and can be used interchangeably, but shouldn’t because SQL Server and Oracle treat them differently.   
SQL Server/Oracle - Schema is a namespace inside the database with different permissions associated with it.

To clear screen from mysql cli -> \! clear

Open MySQL ->  
Type : mysql-ctl cli

Some commands ->

show databases;

show schemas;

SQL statements end with “;”  
If you don’t put the “;”, MySQL will wait to execute until you ad it in.

To create a new database ->  
create database project1;

MySQL is case sensitive.  
Don’t put spaces in the database name.  While possible, it will just cause issues later..  Spaces generally cause annoyances in names of databases, tables, columns and everything.

create schema project1; this statement would accomplish the same thing as the command above to create database

Create another database ->  
create database mashrur1;

show databases;

To delete (or drop) the database mashrur1 (warning: this will immediately delete it) ->  
drop database mashrur1;

To use a specific database for queries we'll be running and impacting (let's say we want this to be project1):  
use project1;

Now when we create and do things like create objects or call objects, database “project1” is assumed.  
Note: up arrow recalls previous commands.

Working with tables - Text lecture

Section 2, Lecture 11

- Tables are just like Excel Spreadsheets

- Columns have headers and are for certain types of data like strings, dates, numbers etc.  
- Rows are the actual data which all fit inside the column data types.

- Structure matters. Forgetting a comma or a line or a space can break things.

To create a table called people (The capitalization of CREATE TABLE below is to separate out the sql code from the names of the tables, you can also just say create table, capitalization matters in column names):  
CREATE TABLE people ( PersonID int  
                     ,first\_name varchar(100)  
                     ,last\_name  varchar(100)  
);

To show database in use, type in select database();  
show tables;

CREATE TABLE people2 ( PersonID INT NOT NULL AUTO\_INCREMENT  
                      ,FIRST\_NAME VARCHAR(100) NULL  
                      ,LAST\_NAME VARCHAR(100) NULL  
                      ,PRIMARY KEY (PersonID));

Primary Key is a unique value that each row will have, auto\_increment makes it easy by automatically assigning them

If we want to remove a table, use DROP TABLE (or drop table)  
show tables;

To add a column, first see the columns that are there by typing in -> show columns from people2;

Notice there are 3 columns;

To add a column called DOB:  
alter table people2 add column DOB date NULL;

Similarly, to drop a column, type in:  
alter table people2 drop column DOB;     
show columns from people2;

To delete tables:  
drop table people;  
drop table people2;

This will get rid of both tables

Homework exercise: Create two tables - actors and movies  
actors will have columns ActorID (as primary key), first\_name, last\_name and specify primary key  
movies will have columns MovieID (as primary key), title, release\_year, rating and specify primary key

Inserting rows of data - Text lecture

Section 2, Lecture 13

To add data to tables:

INSERT INTO actors ( FIRST\_NAME , LAST\_NAME )  
VALUES ('Ben','Stiller');

We’ll discuss select further in the next lecture, but to show all data in the table:  
SELECT \*  
FROM project1.actors;

To insert multiple values into actors table:  
INSERT INTO project1.actors ( FIRST\_NAME , LAST\_NAME )  
VALUES ('Owen','Wilson')  
      ,('Christine','Taylor')  
      ,('Will','Ferrell')  
      ,('Milla','Jovovich')  
      ,('Jerry','Stiller')  
      ,('David','Duchovny')  
      ,('Jon','Voight')  
      ,('Nathan','Graham');

Now that we have Zoolander, lets add new actors from Zoolander 2  
INSERT INTO actors ( FIRST\_NAME , LAST\_NAME )  
VALUES ('Kristen','Wiig')  
      ,('Penélope','Cruz')  
      ,('Lenny','Kravitz')  
      ,('Macaulay','Culkin')  
      ,('Justin','Bieber')  
      ,('Cyrus','Arnold');  
     
Now let's add data to the movies table:  
INSERT INTO movies ( Title , ReleaseYear, Rating )  
VALUES ('Zoolander',2001,'PG-13')  
      ,('Zoolander2',2016,'PG-13')  
      ,('Night at the Museum: Secret of the Tomb',2014,'PG')  
      ,('Night at the Museum: Battle of the Smithsonian',2009,'PG')  
      ,('Night at the Museum',2005,'PG')  
      ,('National Treasure',2004,'PG')  
      ,('Tropic Thunder',2008,'R');

SELECT \* FROM movies; -> This will display all data from the movies table

Select - Text lecture

Section 2, Lecture 15

SELECT \* FROM actors;

SELECT \* FROM movies;

-> SELECT section lists the results to be returned.  
\* Means everything  
From is where you list tables.  
We will add lots of extra details but the SELECT \* FROM is the general structure of how we retrieve any and all data from the database.

Listing the columns returns the same results as \*  
SELECT ActorID  
      ,FIRST\_NAME  
      ,LAST\_NAME  
FROM actors;   
    
SELECT FIRST\_NAME  
      ,LAST\_NAME  
FROM actors;     
Returns same number of rows, but only the columns that were requested (FIRST\_NAME AND LAST\_NAME)

Concat and substring - Text lecture

Section 2, Lecture 17

CONCAT - Use to add columns together for display

SELECT CONCAT(LAST\_NAME, ', ', FIRST\_NAME)  
FROM actors;

Returns same number of rows, but with our calculated row.

SQL lets you rename columns for display as well:  
SELECT CONCAT(LAST\_NAME, ', ', FIRST\_NAME) AS ACTOR\_NAME  
FROM actors;

To grab certain data from the rows under a column (instead of all the data) you can use substring:  
SELECT LAST\_NAME, SUBSTRING(LAST\_NAME,1,3)  
FROM actors;

A useful query to generate user name from first\_name and last\_name columns:  
SELECT FIRST\_NAME  
      ,LAST\_NAME  
      ,SUBSTRING(FIRST\_NAME,1,1)  
      ,CONCAT(SUBSTRING(FIRST\_NAME,1,1),LAST\_NAME) AS USER\_NAME  
FROM actors;

Select allows you to name columns with spaces using “”, you can create new columns, and calculate columns.  You can reuse columns for basically anything.

SELECT CONCAT(LAST\_NAME, ', ', FIRST\_NAME) AS "Actor's Names"  
      ,LAST\_NAME AS LNAME  
      ,FIRST\_NAME AS "First Name"  
      ,CONCAT(SUBSTRING(FIRST\_NAME,1,1),LAST\_NAME) AS USER\_NAME  
      ,'Any Text' AS Anything  
      ,1+2        AS Math  
FROM actors;

Homework challenge - Text version

Section 2, Lecture 19

Phase 1:

Create a table which should have the following columns:  
Primary Key (int), Fname, Lname, Game1, Game2, Game3, Game4

Fill in data for the table with your and 3 of your friends names and 4 game scores (remember id for each row, primary key, should be assigned automatically)

Phase 2:  
Create a report from the data in the table:  
Exclude ID column  
- Each column should have a custom header/title, atleast 1 should have a space in it  
- A column with player initials  
- A column with last name, first name separated by comma  
- Return each game's score  
- Column with total score of all four games for each player  
- Column with average score for each player

IMPORTANT: Please post your code to the Q & A section for this lecture (or the prior video lecture)

Solution - Text

Section 2, Lecture 21

CREATE TABLE players1 (BowlerID INT NOT NULL AUTO\_INCREMENT

                     ,FNAME varchar(50) DEFAULT NULL  
                     ,LNAME varchar(50) DEFAULT NULL  
                     ,Game1 int DEFAULT NULL   
                     ,Game2 int DEFAULT NULL   
                     ,Game3 int DEFAULT NULL   
                     ,Game4 int DEFAULT NULL                        
                     ,PRIMARY KEY (BowlerID));   
INSERT INTO players1 (FNAME, LNAME,Game1,Game2,Game3,Game4)  
VALUES ('Mashrur', 'Hossain',121,87,115,124)  
      ,('Matt', 'Berstein',111,99,135,105)  
      ,('Anastasia', 'Ivanov',75,99,125,141)  
      ,('Mark', 'Futre',115,128,101,84);  
        
SELECT \*  
FROM players1;

SELECT CONCAT(LNAME,', ',FNAME) AS Player  
      ,CONCAT(SUBSTRING(FNAME,1,1),SUBSTRING(LNAME,1,1)) AS Initials  
      ,GAME1 AS G1  
      ,GAME2 AS G2  
      ,GAME3 AS G3  
      ,GAME4 AS G4  
      ,GAME1 + GAME2 + GAME3 + GAME4 AS "Tournament Total"  
      ,(GAME1 + GAME2 + GAME3 + GAME4)/4 AS "Tournament Average"  
FROM players1;

Alternate method of running queries - script - Text version

Section 2, Lecture 23

To run a sql script, create a file (example: test1.sql) in your working directory and save it. Add all the sql code you want to run in there.

Start a SQL cli session and use the source keyword to execute it. If you wanted to run test1.sql type in:  
source test1.sql

Limit, order by and distinct - Text lecture

Section 2, Lecture 26

Large datasets where you don’t want to tax the system returning millions of rows use limit:

SELECT \*  
FROM actors  
LIMIT 10;

This query above will return the first 10 results from the actors table instead of all the rows.

Order by can be used to sort the data that is returned in the query result:  
SELECT \*  
FROM actors  
ORDER BY 3;

This query above will return the results sorted by the 3rd column of the table in ascending order by default

Which is the same as   
SELECT \*  
FROM actors  
ORDER BY LAST\_NAME ASC;

The ASC is for ascending which is implied.  The opposite is descending DESC

SELECT \*  
FROM actors  
ORDER BY 3 DESC;

We can also sort by multiple columns at the same time and by name  
SELECT \*  
FROM actors  
ORDER BY LAST\_NAME ASC, FIRST\_NAME DESC;  
Notice the order the Jerry and Ben are in.

And you can combine LIMIT and ORDER BY  
SELECT \*  
FROM project1.actors  
ORDER BY LAST\_NAME ASC  
        ,FIRST\_NAME ASC  
LIMIT 10;

DISTINCT

Let's say you want to know all of the ratings for the movies   
SELECT RATING  
FROM movies;

But don’t like the duplicate rows and want to only display unique ratings.  
SELECT DISTINCT RATING  
FROM movies;

Distinct only works when the whole row returned is unique. In the query below, because each movie has a unique ID, none of the rows are removed.  
SELECT DISTINCT MovieID  
               ,RATING  
FROM movies;

Count, like and group by - Text lecture

Section 2, Lecture 28

Count can be used as follows:

SELECT COUNT(\*) AS cnt  
FROM movies;

This will return the number of movies in the table

Try the following to get the number of movies for each rating and you'll get incorrect results:  
SELECT RATING  
      ,COUNT(\*) AS cnt  
FROM movies;  
\*Incorrect Results

You can use GROUP BY (column name) to get correct results in such cases:  
SELECT RATING  
      ,COUNT(\*) AS MOVIES  
FROM movies  
GROUP BY RATING;

Like is used all the time for search when an entire description is not known, it's used with a % -> LIKE %  
SELECT \*  
FROM movies  
WHERE TITLE like 'Night at the Museum%';

Name the return columns with spaces using:  
SELECT COUNT(\*) AS "Night at the Museum Movies"  
FROM movies  
WHERE TITLE like 'Night at the Museum%';

You can put % before the word as well so like '%Museum%' and it will return all results that have museum in the title

Min and max - Text lecture

Section 2, Lecture 30

SELECT MAX(ReleaseYear)

FROM movies;

They work exactly how you would expect.

SELECT Title  
      ,MAX(ReleaseYear)  
      ,RATING  
FROM movies;  
\*Incorrect results

SELECT MIN(ReleaseYear)  
      ,RATING  
FROM movies  
WHERE Title like 'Night at the Museum%'  
GROUP BY Title  
        ,RATING;

SELECT RATING  
      ,MIN(ReleaseYear)  
FROM movies  
WHERE Title like 'Night at the Museum%'  
GROUP BY RATING;

SELECT RATING  
      ,MIN(ReleaseYear)  
FROM movies  
GROUP BY RATING;

SELECT RATING  
      ,MAX(ReleaseYear)  
FROM movies  
GROUP BY RATING;

MySQL is more “flexible” than other SQL versions.  It doesn’t require the GROUP BY but it should.  It is likely you will get unhelpful or incorrect results when you do not use a group by.

All result sets should either be part of an aggregate function, or in the Group By.

Code challenge project - Text version

Section 2, Lecture 32

CREATE TABLE bowlResults (BowlResultID INT NOT NULL AUTO\_INCREMENT

                         ,FNAME varchar(50) DEFAULT NULL  
                         ,LNAME varchar(50) DEFAULT NULL  
                         ,Game\_Num int DEFAULT NULL   
                         ,Game\_Score int DEFAULT NULL   
                         ,PRIMARY KEY (BowlResultID));

INSERT INTO bowlResults (FNAME, LNAME,Game\_Num, Game\_Score)  
VALUES ('Mashrur', 'Hossain',1,121)  
      ,('Mashrur', 'Hossain',2,87)  
      ,('Mashrur', 'Hossain',3,115)  
      ,('Mashrur', 'Hossain',4,124)  
      ,('Matt', 'Berstein',1,111)  
      ,('Matt', 'Berstein',2,99)  
      ,('Matt', 'Berstein',3,135)  
      ,('Matt', 'Berstein',4,105)  
      ,('Anastasia', 'Ivanov',1,75)  
      ,('Anastasia', 'Ivanov',2,99)  
      ,('Anastasia', 'Ivanov',3,125)  
      ,('Anastasia', 'Ivanov',4,141)  
      ,('Mark', 'Futre',1,115)  
      ,('Mark', 'Futre',2,128)  
      ,('Mark', 'Futre',3,101)  
      ,('Mark', 'Futre',4,84);

SELECT \*  
FROM bowlResults;

SELECT CONCAT(FNAME,' ',LNAME)  AS Player  
      ,SUM(Game\_Score)          AS "Tournament Total"  
      ,SUM(Game\_Score)/COUNT(\*) AS "Tournament Ave"  
FROM  bowlResults       
GROUP BY FNAME, LNAME     
ORDER BY  SUM(Game\_Score)/COUNT(\*) DESC  
LIMIT 3;

SELECT CONCAT(FNAME,' ',LNAME)  AS Player  
      ,MAX(Game\_Score)          AS "Best Game"  
FROM  bowlResults       
GROUP BY FNAME, LNAME     
ORDER BY  MAX(Game\_Score) DESC;

SELECT DISTINCT FNAME AS "First Name"  
               ,LNAME AS "Last Name"  
FROM  bowlResults;

Data types in MySQL - Text lecture

Section 3, Lecture 34

https://en.wikipedia.org/wiki/SQL#Data\_types

Each column has a data type. They define what type of data is allowed in each field. Understanding them unlocks tons of functions and features and avoids pitfalls that cause a lot of heartache. These can be slightly different between SQL environments, but they all accomplish the same things.

VARCHAR(n) where N is the maximum length of the string (65,535 characters)  
CHAR(n) fixed length of the string regardless of contents. It will be padded with spaces. Anything longer than a few characters should be a VARCHAR as it generally takes up less space in the database.

Read more: http://dev.mysql.com/doc/refman/5.7/en/char.html

Each data type has a different size and takes different amounts of time to process.  Strings are handy in that they don't care what they are holding, dates, numbers, true/false, but a database that relies on strings when they could use other data types are bigger, slower, inefficient and miss out on all of the features built into MySQL.

Date - no consideration of time.  Usually not the one you pick.  
Time - also not the one you usually want because of how limited it is.  Think of how you might handle midnight  
Timestamp - likely always the one you want to define.

Int - Integer is plus or minus -2,147,483,648 so it can handle basically anything. There are other types, but int is usually good enough in most cases.

Boolean -  unlike other database environments, MySQL doesn’t currently have a boolean (True/False). Instead we use tinyint(1).  When creating a table, you can use “boolean”, and MySQL will use tinyint(1) instead.  0 = “False”, and 1=”True”.

Decimal - Exact Values like money and measurements.  Decimals are defined as follows DECIMAL(A,B) where A is the total number of digits, and B is the number of digtis after the period.  For example DECIMAL(5,2) could handle anything from -999.99 to 999.99.  It could not handle 1000, or 123.456.

Where and not equal - Text lecture

Section 3, Lecture 36

In section 2 we looked at where but didn’t dive very far into it. Now we will explore more options of the Where Clause….

Where is how we control which rows are returned.

Example:  
SELECT Title  
FROM movies  
WHERE 1=1;

Always true, so return every title for all movies(rows) in the table.

SELECT 1 As Result  
FROM movies  
WHERE 1=1;

SELECT 1 As Result  
FROM movies  
WHERE Rating=’PG’;

It’s important to see that WHERE only controls which rows are returned, but has no other effect on what data actually returns.

SELECT \*  
FROM movies  
WHERE RATING = "PG"   
 AND Title LIKE "Night%";

You can string as may conditions together as you need using AND

SELECT \*  
FROM movies  
WHERE RATING = "PG"   
 AND Title NOT LIKE "Night%";

NOT allows you to define what you don’t want displayed.

Comparison operators - Text lecture

Section 3, Lecture 38

Greater than and less than are pretty straight forward in their use and meaning:

SELECT title  
    ,release\_year  
FROM movies  
WHERE release\_year > 2005;

Notice that Night at the Museum and National Treasure are not in the list.

SELECT title  
    ,release\_year  
FROM movies  
WHERE release\_year > 2005  
 AND  release\_year < 2016;

We can combine greater than and less than to limit to a specific range.

We can include the limits of the range using >= or <= (Greater Than or Equal To/Less Than or Equal To)  
SELECT title  
    ,release\_year  
FROM movies  
WHERE release\_year >= 2005  
 AND  release\_year <= 2016;

While this is a great way to accomplish this, it’s also perfectly acceptable to use BETWEEN

SELECT title  
    ,release\_year  
FROM movies  
WHERE release\_year BETWEEN 2005 AND 2016;

Exact Same results that include 2005 and 2016

Combination of both Greater Than and Less Than is <> which means Not Equal.    
SELECT title  
      ,release\_year  
FROM movies  
WHERE release\_year <> 2016;

Notice that the only value missing is the 2016

Not Equal also works well with other data types like strings.  
SELECT title  
      ,release\_year  
FROM movies  
WHERE title <> 'Zoolander';

Just like = and <>, we can use IN to include or exclude multiple values at the same time  
SELECT title  
      ,release\_year  
FROM movies  
WHERE release\_year IN (2005,2009,2010);

In order to exclude items, we can use NOT IN  
SELECT title  
      ,release\_year  
FROM movies  
WHERE release\_year NOT IN (2005,2009,2010);

This also works well for Strings  
SELECT title  
      ,release\_year  
FROM movies  
WHERE title IN ('Zoolander','Tropic Thunder');

Or - Text lecture

Section 3, Lecture 40

OR Will allow you to have distinct conditions that may otherwise conflict

example:  
SELECT \*  
FROM movies  
WHERE RATING = "R"   
  OR Title LIKE "Zoo%";

Both of these accomplish the same thing but ()s and spacing help avoid mistakes.

SELECT \*  
FROM movies  
WHERE ( RATING = "R"   
        OR Title LIKE "Zoo%");

Null and blanks - Text lecture

Section 3, Lecture 42

Let's add more data to help explore the issue of null and empty string:

INSERT INTO movies (title, rating)  
VALUES  
  ('DodgeBall: A True Underdog Story' ,'PG-13')  
  ,('Along Came Polly','PG-13')  
  ,('Anchorman:The Legend of Ron Burgundy','PG-13');

INSERT INTO movies (title, release\_year)  
VALUES  
  ('Anchorman 2: The Legend Continues' ,2013)  
  ,('Megamind',2010);

INSERT INTO movies (title, rating)  
VALUES('Fight Club','');

NULL Means there is no data.  This happens all the time depending on the data you are dealing with. The issue comes is when you are trying come compare a non-existent value (null) to anything

SELECT \*   
FROM movies  
WHERE NOT rating = 'PG-13'  
 AND  NOT rating = 'PG'  
 AND  NOT rating = 'R';

Where did Anchorman 2 and Megamind go?

SELECT \*   
FROM movies  
WHERE rating IS NULL;

The issue is we were comparing NULL (a non existent value) to a value. Non-existent values don't equal anything. “IS NULL” and IS NOT NULL” compares the field in a way that will return content in these scenarios.

SELECT \*   
FROM movies  
WHERE rating = 'R'  
      OR rating IS NULL;

We can compare a string to an empty string.  
SELECT \*   
FROM movies  
WHERE rating = 'R'  
      OR rating = ‘’;

SELECT \*   
FROM movies  
WHERE rating = 'R'  
      OR IFNULL(rating,'') = '';

IFNULL(A, B) where A is the field to be compared and B is the default value to be used when NULL is found.  Oracle uses NVL and SQL Server uses ISNULL

Case statements - Text lecture

Section 3, Lecture 44

SELECT title  
      ,rating  
FROM movies;

SELECT title  
      ,rating  
      ,CASE  
         WHEN RATING = 'PG'    THEN 'Bring the Kids!'  
         WHEN RATING = 'PG-13' THEN 'Older Kids'  
         WHEN RATING = 'R'     THEN 'Not for Kids'  
         ELSE 'No Information'  
       END AS AUDIENCE  
FROM movies;     
     
Case statements allow a lot of flexibility. They evaluate the first WHEN, and if it passes, it uses the THEN for the return value and skips to the next row.

Else option and allows for a catch-all value like “No Information”  
The different 'whens' can be inspecting different values, and the CASE Statment can be used in the SELECT or the WHERE.    
Let revisit the NULL query from earlier.

SELECT \*  
FROM movies  
WHERE CASE   
        WHEN rating IS NULL THEN 1  
        WHEN rating = ''    THEN 1  
        WHEN release\_year <= 2007 THEN 1  
      END = 1;

Dates and times - Text lecture

Section 3, Lecture 46

Some time functions:

SELECT NOW()  
      ,CURDATE()  
      ,CURTIME();

More information on time functons: http://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html

MySQL has prebuilt functions that offer some handy functionality when dealing with dates and times.

NOW() returns the system date and time of the server    
CURDATE() returns the system date without the time  
CURTIME() returns the system time without the date

NOW() is useful for adding a timestamp when a row is being added to the database, but let’s use it to explore some other data functions

SELECT NOW()  
      ,CURDATE()  
      ,CURTIME()  
      ,YEAR(NOW())  
      ,YEAR(CURDATE());

We can have just the year pulled out.  This function could easily be used to help group rows by years

SELECT NOW()  
      ,MONTH(NOW())  
      ,MONTHNAME(Now());

MySQL allows you to pull just the month by number or name

SELECT NOW()  
      ,DAY(NOW())  
      ,DAYNAME(Now())  
      ,DAYOFMONTH(NOW())  
      ,DAYOFWEEK(NOW())  
      ,DAYOFYEAR(NOW());

DAY gives the day of the month  
DAYNAME gives the name of the day of the week  
DAYOFMONTH is the same as DAY  
DAYOFWEEK gives the number corresponding to the day. Sunday is 1 and Saturday is 7  
DAYOFYEAR gives the day if we were counting from Jan 1st as 1,

You may have noticed that your cloud 9 Environment is not in your current timezone. Instead the System time is UTC.   
Find your timezone here:  
https://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones

Then you can set your timezone:  
SET time\_zone = '-7:00';  
SET GLOBAL time\_zone = '-7:00';

Then you can check the time difference:  
select @@global.time\_zone, @@session.time\_zone;

You can then check your newly set time:  
select NOW();

Date, time and math - Text lecture

Section 3, Lecture 48

Let’s do some math with dates  
Start with a table  
CREATE TABLE person (personID INT NOT NULL AUTO\_INCREMENT  
                    ,FNAME varchar(50) DEFAULT NULL  
                    ,LNAME varchar(50) DEFAULT NULL  
                    ,dob date DEFAULT NULL   
                    ,PRIMARY KEY (personID));

Insert a couple of rows  
INSERT INTO person (FNAME, LNAME,dob)  
VALUES ('Mashrur', 'Hossain','1981-12-25')  
      ,('Mark', 'Futre','1985-01-03');

Let’s find ages, so we start with  
SELECT dob AS Birthdays  
      ,CURDATE()   AS Today  
FROM person;

MySQL needs to know that you want to do date math and not just regular math.  There are rules for dates like 12 months in a year, and leap years and days in a week. We need to use date specific functions to apply these rules like TIMESTAMPDIFF()

SELECT dob AS "Birthday"  
      ,CURDATE()   AS Today  
      ,TIMESTAMPDIFF(YEAR,dob,CURDATE()) AS AGE  
FROM person;

Let’s find when when the next birthday will be.  In this case we are taking the year only from the current date, attaching it to the month/day of the dob and displaying it as 'Birthday This Year'. Similarly, we are taking the current date, adding a year, and then attaching the dob month/day to calculate what the dob will be next year. The issue is we don’t know if it already happened this year or if it’s going to take place next year.

First let's start with printing out birthday this year and birthday next year  
SELECT FNAME  
      ,dob AS Birthday  
      ,CURDATE()   AS Today  
      ,STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') AS "Birthday This Year"  
      ,DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y'), INTERVAL 1 YEAR ) AS "Birthday Next Year"  
FROM person;

Now, we can use a CASE Statement to help decide which birthday to use:  
SELECT FNAME  
      ,dob AS Birthday  
      ,CURDATE()   AS Today  
      ,STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') AS "Birthday This Year"  
      ,DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y'), INTERVAL 1 YEAR ) AS "Birthday Next Year"  
      ,CASE   
         WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') >= CURDATE()  
           THEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y')  
         ELSE DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y'), INTERVAL 1 YEAR )  
       END AS "Next Birthday"  
FROM person;

Text solution to final project on birthday reporting

Section 3, Lecture 50

Section 3 final project!

Create a table with your four best friends first names, last names and their birthdays.  
Create a report which will return the following:  
1. First and last name together in one column  
2. Current Age  
3. Create a column - if birthday is today, return ‘Call Today’, else if birthday is less than 14 days, display ‘Send Card”, else display Birthday is in (name of month).  
4. Only return friends that are within the next 6 months.  
5. Test your report by replacing current date with various hard coded dates.

First create the table and fill in data:  
CREATE TABLE Friends (FriendID INT NOT NULL AUTO\_INCREMENT  
                     ,FNAME varchar(50) DEFAULT NULL  
                     ,LNAME varchar(50) DEFAULT NULL  
                     ,dob date DEFAULT NULL   
                     ,PRIMARY KEY (FriendID));

INSERT INTO Friends (FNAME, LNAME,dob)  
VALUES ('Mashrur', 'Hossain','1982-12-01')  
      ,('Matt', 'Berstein','1980-08-05')  
      ,('Anastasia', 'Ivanov','1989-04-01')  
      ,('Mark', 'Futre','1989-07-04');

SELECT \*  
FROM Friends;

Its best to build these in phases. Return the individual parts before you incorporate them into case statements  
SELECT CONCAT(FNAME,' ',LNAME)  AS Friend  
      ,TIMESTAMPDIFF(YEAR,dob,CURDATE()) AS AGE  
      ,dob  
      ,CURDATE()  
      ,STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') AS BDAY\_THIS\_YR  
      ,DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y'), INTERVAL 1 YEAR ) AS BDAY\_NEXT\_YR  
      ,DATE\_ADD(CURDATE(), INTERVAL 6 MONTH) AS 6Months  
      ,MONTHNAME(dob) AS "Birth Month"  
FROM Friends  
WHERE CASE   
         WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') >= CURDATE()  
           THEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y')  
         ELSE DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y'), INTERVAL 1 YEAR )  
       END BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 6 MONTH)  
       ;

Now let's add in an update:  
SELECT CONCAT(FNAME,' ',LNAME)  AS Friend  
      ,TIMESTAMPDIFF(YEAR,dob,CURDATE()) AS AGE  
      ,CASE  
         WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') = CURDATE()  
           THEN 'Call Today'  
         WHEN CASE   
                WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') >= CURDATE()  
                  THEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y')  
                ELSE DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y'), INTERVAL 1 YEAR )  
              END <= DATE\_ADD(CURDATE(), INTERVAL 14 DAY)    
           THEN 'Send a card'  
         ELSE concat('Birthday is in ',MONTHNAME(dob))  
       END AS ToDo  
FROM Friends  
WHERE CASE   
         WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y') >= CURDATE()  
           THEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y')  
         ELSE DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR(CURDATE())),'%m/%d/%Y'), INTERVAL 1 YEAR )  
       END BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 6 MONTH)  
       ;

That's it! now let's test it with hardcoding different dates:  
Find and replace works wonders.

SELECT CONCAT(FNAME,' ',LNAME)  AS Friend  
      ,TIMESTAMPDIFF(YEAR,dob,'2016-08-05') AS AGE  
      ,CASE  
         WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR('2016-08-05')),'%m/%d/%Y') = '2016-08-05'  
           THEN 'Call Today'  
         WHEN CASE   
                WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR('2016-08-05')),'%m/%d/%Y') >= '2016-08-05'  
                  THEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR('2016-08-05')),'%m/%d/%Y')  
                ELSE DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR('2016-08-05')),'%m/%d/%Y'), INTERVAL 1 YEAR )  
              END <= DATE\_ADD('2016-08-05', INTERVAL 14 DAY)    
           THEN 'Send a card'  
         ELSE concat('Birthday is in ',MONTHNAME(dob))  
       END AS ToDo  
FROM Friends  
WHERE CASE   
         WHEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR('2016-08-05')),'%m/%d/%Y') >= '2016-08-05'  
           THEN STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR('2016-08-05')),'%m/%d/%Y')  
         ELSE DATE\_ADD( STR\_TO\_DATE(CONCAT(MONTH(dob),'/',DAY(dob),'/',YEAR('2016-08-05')),'%m/%d/%Y'), INTERVAL 1 YEAR )  
       END BETWEEN '2016-08-05' AND DATE\_ADD('2016-08-05', INTERVAL 6 MONTH)  
       ;

Update - Text lecture

Section 4, Lecture 52

In previous lectures we learned about select like below:

SELECT title  
      ,release\_year  
FROM movies  
WHERE release\_year IS NULL;

If we want to update or change the data, we use UPDATE instead of select:  
Use a select to find what to update  
SELECT \*  
FROM movies  
WHERE Title = 'Fight Club';

Then run an update on the same selection criteria:  
UPDATE movies  
SET ReleaseYear = 1999  
WHERE Title = 'Fight Club';

We can update multiple rows and multiple columns at the same time:  
SELECT \*  
FROM movies  
WHERE title IN ('DodgeBall: A True Underdog Story'  
               ,'Along Came Polly'  
               ,'Anchorman:The Legend of Ron Burgundy');

Rows matched:3 Changed:3  
If you run this again, it will say matched: 3, Changed: 0

Use SELECT \* to verify your WHERE before updating a table.

Update based on comparison - Text lecture

Section 4, Lecture 54

Let’s create some data to work with:   
CREATE TABLE CustBal (CustID int NOT NULL AUTO\_INCREMENT  
            ,CheckingBalance DECIMAL(8,2)  
            ,SavingsBalance DECIMAL(8,2)  
            ,CreditCardBalance DECIMAL(8,2)  
            ,CreditCardTotalCredit DECIMAL(8,2)  
            ,CreditCardAvailableCredit DECIMAL(8,2)  
            ,TotalBalance DECIMAL(8,2)  
            ,ActiveStatus boolean  
            ,primary key (CustID));

INSERT INTO CustBal ( CheckingBalance  
                     ,SavingsBalance  
                     ,CreditCardBalance  
                     ,CreditCardTotalCredit  
                     ,CreditCardAvailableCredit  
                     ,TotalBalance  
                     ,ActiveStatus)  
VALUES (835.27 , 2223.01 ,   123.09 ,  1000.00 , NULL , NULL , 1)  
      ,(165.27 , 1223.01 ,     0.00 ,  1000.00 , NULL , NULL , 1)  
      ,(899.27 , 3223.01 ,  9125.09 , 20000.00 , NULL , NULL , 1)  
      ,(  0.00 ,    0.00 ,    00.00 ,     0.00 , NULL , NULL , 0)        
      ,(123.27 , 4223.01 ,   123.00 ,   800.00 , NULL , NULL , 1)  
      ,( 65.00 , 6223.01 , 21123.09 , 20000.00 , NULL , NULL , 1)  
      ,(315.90 , 7223.01 ,     0.00 , 21000.00 , NULL , NULL , 1)  
      ,(992.22 , 1923.01 , 20000.00 , 20000.00 , NULL , NULL , 1)  
      ,( 75.21 , 4223.01 ,  1123.51 , 15000.00 , NULL , NULL , 1)  
      ,(165.22 ,    0.01 , 15123.08 , 16000.00 , NULL , NULL , 0)  
      ,(455.23 ,    0.00 ,  4123.00 ,  5000.00 , NULL , NULL , 1);

Let's say we wanted a process to update the customer accounts  
1) Use Select Statements to plan out what you want to do  
SELECT CustID  
      ,CreditCardTotalCredit  
      ,CreditCardBalance   
      ,CreditCardAvailableCredit                 AS OldCreditCardAvailableCredit  
      ,CreditCardTotalCredit - CreditCardBalance AS NewCreditCardAvailableCredit  
FROM CustBal   
WHERE ActiveStatus = 1;  
        
SELECT CustID  
      ,CheckingBalance  
      ,SavingsBalance  
      ,CreditCardBalance  
      ,TotalBalance                                          AS OldTotalBalance   
      ,CheckingBalance + SavingsBalance - CreditCardBalance  AS NewTotalBalance   
FROM CustBal   
WHERE ActiveStatus = 1;

We could have combined them into one larger select, but breaking it apart makes it easier to read in the terminal window.

2) Put it together into an update statement  
UPDATE CustBal  
SET CreditCardAvailableCredit = CreditCardTotalCredit - CreditCardBalance  
   ,TotalBalance              = CheckingBalance + SavingsBalance - CreditCardBalance  
WHERE ActiveStatus = 1;

Delete - Text lecture

Section 4, Lecture 56

Delete is just like Select but instead of returning rows with information, it removes them from the table. As a rule, always use a select statement to verify what you are about to remove.

Example, remove all of the inactive accounts from our CustBal table:

First, identify the inactive accounts:  
SELECT \*  
FROM CustBal   
WHERE ActiveStatus = 0;

To delete, replace SELECT \* with DELETE and execute;  
DELETE   
FROM CustBal   
WHERE ActiveStatus = 0;

Primary and foreign keys - Text lecture

Section 5, Lecture 58

Primary Keys and Foreign Keys

While not always, most tables use Primary Keys to uniquely identify each row. In our exercises, we’ve had an auto incrementing integer that keeps the rows unique. Think about what would happen if you had lots of rows that were identical, how do you keep them separate or unique to identify them? Understanding keys is essential when we start exploring joining tables.  
A primary key keeps every row unique. If you were going to join two tables together, you can use the primary key from one table to reference another

In the examples in the video, we have 2 tables. Persons and Movies. Each has an integer as their primary key. Notice that the Persons Table uses the Primary key from Movies in the Favorite Movie column. This is an example of a foreign key. The Favorite Movie column contains they key of another table. Notice that rows in the Movies table can be referenced several times or not at all.

Another example in the video is another table ‘MoviesPeopleHaveSeen’. Both PersonID and MovieID are foreign keys, but together they make the primary key of this table. This means, no two rows can have the exact same PersonID and MovieID combination. Setting up tables like this allows us to have details about the movie and details about the people recorded only once. If a detail about a move changes, we change it on the movies table in one place which saves space and simplifies updates, otherwise we would have to update every table that had movies in them.

Data prep for joins - Text lecture

Section 5, Lecture 60

Lets prepare some tables for Joins.  
Generally, you don’t want to rename tables, but let's expand the definition of our actors table and name it as people

RENAME TABLE actors TO people;

Now let's add some more people:

INSERT INTO people (FIRST\_NAME , LAST\_NAME)  
VALUES ('Steven','Spielberg')  
    ,('Shawn','Levy')   
    ,('Jon','Turteltaub')   
    ,('Rawson','Thurber')   
    ,('John','Hamburg')   
    ,('Adam','McKay')   
    ,('Tom','McGrath')  
    ,('David','Fincher');

Now let's add a director column to the movies table which will be a foreign key with the primary key of the people table:

ALTER TABLE movies   
ADD COLUMN  Director INT NULL;

Now let's set directors for each row in the movies table:  
UPDATE movies  
SET Director = 1  
WHERE MovieId IN (1,2,7);

UPDATE movies  
SET Director = 17  
WHERE MovieId IN (3,4,5);

UPDATE movies  
SET Director = 18  
WHERE MovieId = 6;

UPDATE movies  
SET Director = 19  
WHERE MovieId = 8;

UPDATE movies  
SET Director = 20  
WHERE MovieId = 9;

UPDATE movies  
SET Director = 21  
WHERE MovieId IN (10,11);

UPDATE movies  
SET Director = 22  
WHERE MovieId = 12;

UPDATE movies  
SET Director = 23  
WHERE MovieId = 13;

Let's create a characters table:

CREATE TABLE characters ( characterID INT NOT NULL AUTO\_INCREMENT  
                         ,MovieID INT NULL   
                         ,ActorID INT NULL  
                         ,CharacterName VARCHAR(100) NULL  
                         ,PRIMARY KEY (characterID));

Let's add some characters to it with the foreign keys of MovieID and ActorID, along with Character Name in the movie:  
INSERT INTO characters (MovieID , ActorID, CharacterName) VALUES  
 (1 , 1 , 'Derek Zoolander')  
,(1 , 2 , 'Hansel')  
,(1 , 3 , 'Matilda Jeffries')  
,(1 , 4 , 'Mugatu')  
,(1 , 5 , 'Katinka')  
,(1 , 6 , 'Maury Ballstein')  
,(1 , 7 , 'J.P. Prewitt')  
,(1 , 8 , 'Larry Zoolander')  
,(1 , 9 , 'Todd')  
,(2 , 1 , 'Derek Zoolander')  
,(2 , 2 , 'Hansel')  
,(2 , 3 , 'Matilda Jeffries')  
,(2 , 4 , 'Mugatu')  
,(2 , 5 , 'Katinka')  
,(2 , 9 , 'Todd')  
,(2 , 10, 'Alexanya Atoz')  
,(2 , 11, 'Valentina Valencia')  
,(2 , 12, 'Lenny Kravitz')  
,(2 , 14, 'Justin Bieber')  
,(2 , 15, 'Derek Jr.')  
,(5 , 1 , 'Larry Daley')  
,(5 , 2 , 'Jedediah')  
,(4 , 1 , 'Larry Daley')  
,(4 , 2 , 'Jedediah')  
,(3 , 1 , 'Larry Daley')  
,(3 , 2 , 'Jedediah')  
,(3 , 1 , 'Laaa')  
,(6 , 8 , 'Patrick Gates')  
,(7 , 1 , 'Tugg Speedman')  
,(7 , 3 , 'Rebecca')  
,(7 , 8 , 'Jon Voight')  
,(8 , 3 , 'Kate Veatch')  
,(8 , 1 , 'White Goodman')  
,(9 , 1 , 'Reuben Feffer');

Joins and aliases - Text lecture

Section 5, Lecture 62

Join:

SELECT \*   
FROM movies  
    ,people  
WHERE movies.DIRECTOR = people.ActorID ;

We simply list the tables with a comma to separate them and we define how the two tables relate to each other in the where clause.

When dealing with more than one table, we should always specify the table. Often times tables have columns with the same names and MySQL won’t know which one you want.

Let's select some specific columns instead of all of them:    
SELECT movies.TITLE  
      ,movies.RELEASE\_YEAR  
      ,movies.RATING  
      ,CONCAT(people.FIRST\_NAME,' ',people.LAST\_NAME) AS Director  
FROM movies  
    ,people  
WHERE movies.DIRECTOR = people.ActorID ;

While specifying the table is necessary in many situations, and always a good habit, it causes a Lot of typing and can make the code look overwhelming. Which is why we use aliases

Aliases

SELECT A.TITLE  
      ,A.RELEASE\_YEAR  
      ,A.RATING  
      ,CONCAT(B.FIRST\_NAME,' ',B.LAST\_NAME) AS Director  
FROM movies A  
    ,people B  
WHERE A.DIRECTOR = B.ActorID ;

You can use anything as the alias, but I have found that simply using from letters A-Z for each table is the best way. It’s clean and straightforward. All of the examples and code going forward will incorporate this type of aliasing.

More about joins - Text lecture

Section 5, Lecture 64

Let’s Join Movies with Characters

SELECT A.TITLE  
      ,A.RELEASE\_YEAR  
      ,A.RATING  
      ,B.CharacterName  
      ,B.ActorID  
FROM movies A  
    ,characters B  
WHERE A.MovieID = B.MovieID;

We can add additional tables:  
SELECT A.TITLE  
      ,A.RELEASE\_YEAR  
      ,A.RATING  
      ,B.CharacterName  
      ,CONCAT(C.FIRST\_NAME,' ',C.LAST\_NAME) AS Actor  
FROM movies A  
    ,characters B  
    ,people C  
WHERE A.MovieID = B.MovieID   
 AND  B.ActorID  = C.ActorID;

We can reference the same table multiple times to pull information for various reasons. In the case below, we are using the People table to provide the name of both the directors and the actors

SELECT A.TITLE  
      ,CONCAT(D.FIRST\_NAME,' ',D.LAST\_NAME) AS Director  
      ,A.RELEASE\_YEAR  
      ,A.RATING  
      ,B.CharacterName  
      ,CONCAT(C.FIRST\_NAME,' ',C.LAST\_NAME) AS Actor  
FROM movies A  
    ,characters B  
    ,people C  
    ,people D  
WHERE A.MovieID  = B.MovieID   
 AND  B.ActorID  = C.ActorID  
 AND  A.DIRECTOR = D.ActorID  
 AND  A.TITLE LIKE 'Zoo%';

This can be written a different way. Listing tables in the FROM and joining them in the WHERE is perfectly acceptable, you can also join tables by putting everything in the FROM:

SELECT A.TITLE  
      ,CONCAT(D.FIRST\_NAME,' ',D.LAST\_NAME) AS Director  
      ,A.RELEASE\_YEAR  
      ,A.RATING  
      ,B.CharacterName  
      ,CONCAT(C.FIRST\_NAME,' ',C.LAST\_NAME) AS Actor  
FROM movies A  
       INNER JOIN characters B  
         ON A.MovieID  = B.MovieID   
       INNER JOIN people C  
         ON B.ActorID  = C.ActorID    
       INNER JOIN people D  
         ON A.DIRECTOR = D.ActorID  
WHERE A.TITLE LIKE 'Zoo%';

Both of these joins are Inner Joins. They mean that all of the rows in all of the tables need to exist. If no actors were listed for a movie, the movie won't show on the list. If no movies are associated with that actor, the actor won’t be returned.

Inner, Left, Outer and Right joins - Text lecture

Section 5, Lecture 66

Before we go any further with Joins, let's get the main types straight: INNER, LEFT OUTER, RIGHT OUTER and FULL OUTER.

There are a few ways to specify these types in SQL, but for the purposes of this course, we will stick to these terms: LEFT OUTER JOIN is often called a LEFT JOIN, and an INNER JOIN may be referred to as a JOIN. A FULL OUTER JOIN can be called an OUTER JOIN. etc.

Imagine we have two tables:

Table A  
 ColA  
  1  
  2  
  3  
  4

Table B   
 ColB  
  3  
  4  
  5  
  6

If we were to join these tables together, the number of rows returned would depend on the join we use. Notice 1 and 2 are unique to Table A, 5 and 6 are unique to table B. 3 and 4 are common to both tables.

SELECT \*  
FROM A INNER JOIN B ON A.ColA = B.ColB;

This would return:  
ColA | ColB  
 3   |  3  
 4   |  4  
INNER JOIN requires that the value be on both tables

SELECT \*  
FROM A LEFT OUTER JOIN B ON A.ColA = B.ColB;

This would return:  
ColA | ColB  
 1   | NULL  
 2   | NULL  
 3   |  3  
 4   |  4  
LEFT OUTER JOIN returns everything from the first table (left) whether or not the second table has a corresponding row

SELECT \*  
FROM A RIGHT OUTER JOIN B ON A.ColA = B.ColB;

This would return:  
ColA | ColB  
 3   |  3  
 4   |  4  
 NULL|  5  
 NULL|  6  
RIGHT OUTER JOIN returns everything from the second table (right) whether or not the first table has a corresponding row

SELECT \*  
FROM A FULL OUTER JOIN B ON A.ColA = B.ColB;

This would return:  
ColA | ColB  
 1   | NULL  
 2   | NULL  
 3   |  3  
 4   |  4  
 NULL|  5  
 NULL|  6  
FULL OUTER JOIN returns everything from both tables whether or not there are corresponding rows

Left outer joins - Text lecture

Section 5, Lecture 68

Let's say we want to report all of the actors and the movies that they directed:

SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
      ,B.TITLE AS Directed  
      ,B.RELEASE\_YEAR  
FROM people A  
      INNER JOIN movies B  
        ON B.DIRECTOR = A.ActorID;

If I want to have rows from the people table returned whether or not they directed a movie, I can use a left outer join:

SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
      ,B.TITLE AS Directed  
      ,B.RELEASE\_YEAR  
FROM people A  
      LEFT OUTER JOIN movies B  
        ON B.DIRECTOR = A.ActorID;

We can couple this concept with functions we discussed earlier like Min and Max

SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
      ,MIN(C.RELEASE\_YEAR) AS FirstDirected  
      ,MAX(B.RELEASE\_YEAR) AS LatestRelease  
FROM people A  
     LEFT OUTER JOIN movies B  
      ON B.DIRECTOR = A.ActorID  
     LEFT OUTER JOIN movies C  
      ON C.DIRECTOR = A.ActorID  
GROUP BY A.FIRST\_NAME ,A.LAST\_NAME  
ORDER BY COUNT(B.MovieID) DESC;

When your joining tables, you can join them to the primary table or any table, but it is important to keep them in a logical order so that you can troubleshoot if you run into unexpected results.

SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
      ,IFNULL(MIN(C.ReleaseYear),'N/A') AS FirstDirected  
      ,IFNULL(MAX(B.ReleaseYear),'N/A') AS LatestRelease  
FROM people A  
      LEFT OUTER JOIN movies B  
       ON B.director = A.ActorID  
      LEFT OUTER JOIN movies C  
       ON C.director = B.director     
GROUP BY A.FIRST\_NAME  
        ,A.LAST\_NAME;

Fix a bad join challenge - Text lecture

Section 5, Lecture 70

You have been given the following query to produce the selected columns but the join is not working correctly. Run the following query and then fix it:  
SELECT A.FNAME  
      ,A.dob  
      ,B.Game1  
      ,B.Game2  
      ,B.Game3  
      ,B.Game4  
FROM Friends A  
    ,bowlers B;

Solution:  
Pick any join you want, but for my example, we will use INNER. Also, depending on which order you put the same fiends in, you can’t rely on the bowlerID and FriendID matching so I joined on names. If none of your friends and bowlers match up, insert a few friends that do.  
SELECT A.FNAME  
      ,A.dob  
      ,B.Game1  
      ,B.Game2  
      ,B.Game3  
      ,B.Game4  
FROM Friends A  
     INNER JOIN bowlers B  
       ON A.FNAME = B.FNAME  
       AND A.LNAME = B.LNAME;

Subselects - Text lecture

Section 5, Lecture 72

SQL allows queries inside of other quiries which can be extermely useful. Take our example from the previous lecture. If we want to know how many movies each director directed, a subselect can easily do the job.

SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
      ,IFNULL(MIN(C.Release\_Year),'N/A') AS FirstDirected  
      ,IFNULL(MAX(B.Release\_Year),'N/A') AS LatestRelease  
      ,D.cnt AS MoviesDirected  
FROM people A  
     LEFT OUTER JOIN movies B  
      ON B.director = A.ActorID  
     LEFT OUTER JOIN movies C  
      ON C.director = B.director     
     LEFT OUTER JOIN (SELECT Z.Director AS ActorID  
                            ,COUNT(Z.MovieID) AS cnt  
                      FROM movies Z  
                      GROUP BY Z.Director) D  
      ON D.ActorID = A.ActorID                        
GROUP BY A.FIRST\_NAME  
        ,A.LAST\_NAME  
        ,D.cnt  
ORDER BY D.cnt DESC;

While there are performance hits from subselects in the SELECT section, it also is an option:  
SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
      ,IFNULL(MIN(C.Release\_Year),'N/A') AS FirstDirected  
      ,IFNULL(MAX(B.Release\_Year),'N/A') AS LatestRelease  
      ,(SELECT COUNT(Z.MovieID)   
        FROM movies Z  
        WHERE Z.Director = A.ActorID) AS MoviesDirected  
FROM people A  
     LEFT OUTER JOIN movies B  
      ON B.director = A.ActorID  
     LEFT OUTER JOIN movies C  
      ON C.director = B.director     
GROUP BY A.FIRST\_NAME  
        ,A.LAST\_NAME  
ORDER BY MoviesDirected DESC;

More subselects - Text lecture

Section 5, Lecture 74

SUBSELECTS are also useful when you want to refer to a table but don’t plan on returning results from that additional table.

Let’s try to pull a list of people who have directed movies, but we don’t care to display anything from the movies.

SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
FROM people A  
WHERE EXISTS (SELECT 1  
              FROM movies Z  
              WHERE Z.Director = A.ActorID)   
GROUP BY A.FIRST\_NAME  
        ,A.LAST\_NAME;

Notice that they SELECT is set to return 1.  You need to have a select statement, but there is no point in specifying anything there as it will not be used.

Another alternative to accomplish the same thing is using IN.  The statement Z.Director = A.ActorID is removed and replaced by putting 1 and only one column in the SELECT statement (Z.Director) and replacing ‘EXISTS’ with ‘A.ActorID IN’  
SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
FROM people A  
WHERE A.ActorID IN (SELECT Z.Director  
                    FROM movies Z)   
GROUP BY A.FIRST\_NAME  
        ,A.LAST\_NAME;

We can use NOT to find anyone on the People table who has never directed a movie  
SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
FROM people A  
WHERE NOT EXISTS (SELECT 1  
                  FROM movies Z  
                  WHERE Z.Director = A.ActorID)   
GROUP BY A.FIRST\_NAME  
        ,A.LAST\_NAME;

Similarly, we can use NOT IN to find the same information:

SELECT A.FIRST\_NAME  
      ,A.LAST\_NAME  
FROM people A  
WHERE A.ActorID NOT IN (SELECT Z.Director  
                        FROM movies Z)   
GROUP BY A.FIRST\_NAME  
        ,A.LAST\_NAME;

Final Project Part 1 - Text solution

Section 5, Lecture 76

Challenge: Update with Joins

Objective: Alter the bowlers table to not keep the individual game scores.  Instead we will have a Running Average score (int) based on the results from bowlresults, ensure the names of the bowlers match in both tables (FNAME and LNAME)

You will need need some new functions for this:   
AVE() to pull the average of a grouped value   
ROUND() will round to an integer \* use google to find and see examples of these functions

Step 1, drop the unneeded columns from bowlers table:  
ALTER TABLE bowlers   
 DROP COLUMN Game1  
,DROP COLUMN Game2  
,DROP COLUMN Game3  
,DROP COLUMN Game4;

Step 2: Add new column to bowlers:  
ALTER TABLE bowlers  
ADD COLUMN AveScore int DEFAULT NULL;

Step 3: Plan the Update  
SELECT A.FNAME  
      ,A.LNAME  
      ,AVG(B.Game\_Score) AS RawAveScore  
      ,ROUND(AVG(B.Game\_Score)) AS AveScore  
FROM  bowlers A  
     ,bowlResults B  
WHERE A.FNAME = B.FNAME  
 AND  A.LNAME = B.LNAME  
GROUP BY A.FNAME  
        ,A.LNAME;

Step 4: Create the update statement from the select

UPDATE bowlers A  
SET AveScore = (SELECT ROUND(AVG(Game\_Score)) AS AveScore  
                FROM bowlResults B   
                WHERE B.FNAME = A.FNAME   
                 AND  B.LNAME = A.LNAME);   
SELECT \* FROM bowlers;

CONGRATULATIONS!

Final Project Part 2 - Text solution

Section 5, Lecture 78

Challenge: Delete With Joins

Objective: Delete all people from the people table that have never been a director.

Step 1: Plan it out using a select Statement  
SELECT COUNT(\*) FROM people;   
SELECT COUNT(\*)  
FROM people A  
WHERE NOT EXISTS (SELECT 1  
                  FROM movies Z  
                  WHERE Z.Director = A.ActorID);  
Sometimes it’s helpful to view what you are going to be deleting in terms of row count

SELECT \*  
FROM people A  
WHERE NOT EXISTS (SELECT 1  
                  FROM movies Z  
                  WHERE Z.Director = A.ActorID);     
                   
Step 2: Update the Select into a Delete and Execute  
DELETE  
FROM people   
WHERE NOT EXISTS (SELECT 1  
                  FROM movies Z  
                  WHERE Z.Director = ActorID);

Step 3: Confirm your results  
SELECT \* FROM people;

HTML and CSS - Text lecture

Section 6, Lecture 81

Use this link to learn/build a site using HTML/CSS -> http://learn.shayhowe.com/html-css/

Assignment: Once complete you can either paste images of this completed site to the Q & A section, or you can create your own static site and post pictures of that. Good luck!

Project start - Text lecture

Section 6, Lecture 83

Right click on workspace folder and select the option to create a new folder   
Name the folder FormCollector

Right click on the FormCollector folder and select ‘New File’, name it OptIn.html  
We are going to put static HTML form in this file to collect information like name and email address. This could be used as part of a larger website that is asking people to join the mailing list.

HTML is fairly easy to read.  Everything has “tags” which are words or phrases inside of <> and </> to show the beginning and end of that tag. For Example, at the beginning of an HTML document, we have the tag <HTML> meaning anything after this and prior to </HTML> is html code. Similarly, we have a Header section of the HTML marked by <HEAD> and </HEAD>  
http://www.w3schools.com/html/ is a great resource when referencing HTML, along with http://learn.shayhowe.com/html-css/  
Something handy to know: Commenting stuff out is done by adding <!-- add comment here -->

Open OptIn.html, paste the following  
<HTML>  
  <HEAD>  
  </HEAD>   
  <BODY>  
    <h1>Sign Up</h1>  
  </BODY>  
</HTML>  
Save and close.  
Run it by clicking on run project and going to the screen.

Notice that “Sign Up” is in a pretty big font due to H1 tag.

Now let's add a tag for title.  
Add the title tag between the HEAD tags  
<TITLE>OptIn</TITLE>

Now add the form tag  
Place the following below the <h1>Sign Up</h1>  
      <form action="receiver.php" method = "POST" >  
        <input type="submit" class='button'>          
      </form>

We haven’t created it yet, but when the form is submitted, it will send the contents of the form to the form action page. In this case “reciever.php”  
Test it out.

Notice that we have a method of POST.  This can be thought of as we are submitting/posting information to the other page. There is another method called GET, but for our purposes, POST works well.

Notice that the submit button is generated by this:  
<input type="submit" class='button'>

All of the information we send in the form needs to be part of an input.

To collect the information, we add Inputs for First Name, Last Name, and Email address prior to the submit button.  
<p>First Name:  <input type="text" name="FNAME" maxlength='100'  required/></p>  
<p>Last Name:  <input type="text" name="LNAME" maxlength='100'  required/></p>        
<p>Email: <input type="text" name="EMAIL" maxlength='100'  required/></p>

The <p> tag is for paragraph and it simply creates some space between the inputs.  
“First Name: “, “Last Name: “, and “Email: “ are simply labels  
“required” is an HTML5 setting that requires the fields to be filled out before the form will submit.  
“maxlength” prevents users from entering too much data. Depending on what we plan to do with the input, this length will vary.  
The name of “FNAME” or “LNAME” is what the input will be named when the form is submitted.

Type sets the type of input. There are a number of types, test types, submit types and lots of others.  
Another type is “hidden” which simply is an input that the user won’t see, but will be sent as part of the form.  
Test this out.

Let's add some hidden inputs for use later on in the project:  
<input type="hidden" name="formID" value="OptIn" />  
<input type="hidden" name="successredirecturl" value="Success.html">  
<input type="hidden" name="rejectredirecturl" value="Fail.html">

At this point, the full file should be as follows:  
<HTML>  
  <HEAD>  
    <TITLE>OptIns</TITLE>  
  </HEAD>   
  <BODY>  
    <h1>Sign Up</h1>  
      <form action="receiver.php" method = "POST">  
        <input type="hidden" name="formID" value="OptIn" />  
        <input type="hidden" name="successredirecturl" value="Success.html">  
        <input type="hidden" name="rejectredirecturl" value="Fail.html">  
        <p>First Name:  <input type="text" name="FNAME" maxlength='100'  required/></p>  
        <p>Last Name:  <input type="text" name="LNAME" maxlength='100'  required/></p>          
        <p>Email: <input type="text" name="EMAIL" maxlength='100'  required/></p>  
        <input type="submit" class='button'>          
      </form>  
  </BODY>  
</HTML>

The plan for the successredirecturl and rejectredirecturl is that our application will send the user to the predefined page depending on the outcome of the form submission.

Add styling to form - Text follow-up

Section 6, Lecture 85

Add the following to OptIn.html directly under the TITLE tags

 <link href="styles.css" rel="stylesheet" type="text/css">

Something else we need to add to our HTML are div tags and classes. This is simply a way to connect the CSS styles to the HTML code. Div is simply a section to apply the settings corresponding to the class. Update the OptIn.HTML to the following

<HTML>  
  <HEAD>  
    <TITLE>OptIns</TITLE>  
    <link href="styles.css" rel="stylesheet" type="text/css">   
  </HEAD>   
  <BODY>  
    <div class="backgrounds">  
      <div class="main-content">  
        <h1>Sign Up</h1>  
        <form action="receiver.php" method = "POST">  
          <input type="hidden" name="formID" value="OptIn" />  
          <input type="hidden" name="successredirecturl" value="Success.html">  
          <input type="hidden" name="rejectredirecturl" value="Fail.html">  
          <p>First Name:  <input type="text" name="FNAME" maxlength='100'  required/></p>  
          <p>Last Name:  <input type="text" name="LNAME" maxlength='100'  required/></p>          
          <p>Email: <input type="text" name="EMAIL" maxlength='100'  required/></p>  
          <input type="submit" class='button'>          
        </form>  
      </div><!--main-content-->  
    </div><!--background-->   
  </BODY>  
</HTML>

Create a new file named styles.css to add the styling  
This file is where you specify colors, sizes, fonts and most things related to styling HTML.

We are going to use this style sheet to set a background color, border, make the button look cool, etc.

Check out this page to generate custom HTML and CSS code for buttons:  
http://livetools.uiparade.com/button-builder.html

Here are the full contents of the CSS file:   
\*\*\* Caution. Your browser sometimes stores a version of the CSS and reuses that instead of pulling in updates. You may have to clear your “Cached images and files” to see the changes.

body  
{  
background-color:#000000;  
background-repeat:repeat-x;  
background-position:center top;  
}

.backgrounds {  
  position: relative;  
  margin: auto;  
  border-style: solid;  
   border-color: red;  
   padding: 1% 1% 1% 1%; /\*top right bottom left \*/  
  /\*background-image:url(images/background.png); \*/  
  background-repeat: no-repeat;  
  background-size: contain;  
  background-color: #4d4d4d;  
  max-width: 1345px;  
  background-position: top;   
  vertical-align: text-top;     
  height: auto;  
}

.main-content {  
   position:relative;   
   border-style: solid;  
  border-color:red;  
  background-color:rgba(0, 0, 0, 0.8);  
  height: auto;  
  text-align: center;  
  color: #ffffff;  
  font-size: 150%;  
  padding: 5px 5px 5px 5px;  
}

.button {  
   border: 1px solid #401818;  
   background: #9c3e3e;  
   background: -webkit-gradient(linear, left top, left bottom, from(#d66565), to(#9c3e3e));  
   background: -webkit-linear-gradient(top, #d66565, #9c3e3e);  
   background: -moz-linear-gradient(top, #d66565, #9c3e3e);  
   background: -ms-linear-gradient(top, #d66565, #9c3e3e);  
   background: -o-linear-gradient(top, #d66565, #9c3e3e);  
   background-image: -ms-linear-gradient(top, #d66565 0%, #9c3e3e 100%);  
   padding: 10px 20px;  
   -webkit-border-radius: 11px;  
   -moz-border-radius: 11px;  
   border-radius: 11px;  
   -webkit-box-shadow: rgba(255,255,255,0.4) 0 1px 0, inset rgba(255,255,255,0.4) 0 1px 0;  
   -moz-box-shadow: rgba(255,255,255,0.4) 0 1px 0, inset rgba(255,255,255,0.4) 0 1px 0;  
   box-shadow: rgba(255,255,255,0.4) 0 1px 0, inset rgba(255,255,255,0.4) 0 1px 0;  
   text-shadow: #bd7f7f 0 1px 0;  
   color: #ffffff;  
   font-size: 17px;  
   font-family: helvetica, serif;  
   text-decoration: none;  
   vertical-align: middle;  
   }  
.button:hover {  
   border: 1px solid #401818;  
   text-shadow: #571e1e 0 1px 0;  
   background: #9c3e3e;  
   background: -webkit-gradient(linear, left top, left bottom, from(#d66565), to(#9c3e3e));  
   background: -webkit-linear-gradient(top, #d66565, #9c3e3e);  
   background: -moz-linear-gradient(top, #d66565, #9c3e3e);  
   background: -ms-linear-gradient(top, #d66565, #9c3e3e);  
   background: -o-linear-gradient(top, #d66565, #9c3e3e);  
   background-image: -ms-linear-gradient(top, #d66565 0%, #9c3e3e 100%);  
   color: #fff;  
   }  
.button:active {  
   text-shadow: #571e1e 0 1px 0;  
   border: 1px solid #401818;  
   background: #d66565;  
   background: -webkit-gradient(linear, left top, left bottom, from(#9c3e3e), to(#9c3e3e));  
   background: -webkit-linear-gradient(top, #9c3e3e, #d66565);  
   background: -moz-linear-gradient(top, #9c3e3e, #d66565);  
   background: -ms-linear-gradient(top, #9c3e3e, #d66565);  
   background: -o-linear-gradient(top, #9c3e3e, #d66565);  
   background-image: -ms-linear-gradient(top, #9c3e3e 0%, #d66565 100%);  
   color: #fff;  
   }

Paste the code into styles.css, save it and close.

Now test it by visiting the page!

The plan now is to have the user fill out the form, then hit Submit. The HTML form will reach out to the receiver.php file with all the information from the form, with an IP, and what to do in the case of a success or failure.  If it’s successful, reciever.php will redirect the user to the success page, and if there is an issue, the user will be directed to the Fail.html page.

To create the success and fail pages:  
Right click on FormCollector folder and create both Success.html and Fail.html files.  
Inside Fail.html, simply type the word “Fail!”, then save, test it in browser and close  
Inside Success.html, type the word “Success!”, then save, test it in browser and close

Create table - Text follow-up

Section 6, Lecture 87

Create a database, use it and then create a table from mysql cli:

1) CREATE DATABASE formcollector;

2) use formcollector;

3) CREATE TABLE OptIn ( OptInID INT NOT NULL AUTO\_INCREMENT  
                    ,formID varchar(50) DEFAULT NULL  
                    ,NAME varchar(100) DEFAULT NULL   
                    ,EMAIL varchar(100) DEFAULT NULL   
                    ,ENTRY\_TimeStamp datetime DEFAULT NULL  
                    ,SOURCE\_IP char(15) DEFAULT NULL  
                    ,successredirecturl char(200) DEFAULT NULL  
                    ,rejectredirecturl char(200) DEFAULT NULL  
                    ,PRIMARY KEY (OptInID));

Our PHP scripts will need to connect to the database, so the easiest way to maintain the connection information is using a config file. So create a config.php file for the application connection information under the formcollector folder

Config and other php - Text follow-up

Section 6, Lecture 89

Update config.php with the following code:  
<?php  
/\*Configuration Settings\*/

define('DB\_HOST', localhost'); /\*Database Server\*/  
define('DB\_NAME', 'formcollector'); /\*Database Name\*/  
define('DB\_USER', 'root'); /\*Database Username\*/  
define('DB\_PWD',  ''); /\*Database Password\*/

?>

This information could have been stored with the rest of the application, but separating it out into it’s own file simplifies things when moving the application to a new location.

Create another file named receiver.php  
Put the following contents in it:  
<?php  
/\*Bring in our custom library of php and functions, we will create this next\*/  
include('SQLFunctions.php');

/\*First we are going to handle what is being submitted in the form by displaying it, so we display what is in the \_POST\*/  
/\*echo simply means write the following out to the page, adding this description and the line breaks will make the output legible \*/  
/\*The echo statements aren’t actually needed, but they are really helpful in troubleshooting\*/  
  echo '<br>Display full contents of the \_POST: <br>';  
/\*var\_dump is a function that write all of names and values from the post\*/  
  var\_dump($\_POST);

/\*We need to esatblish a connection with the database  
   f\_sqlConnect() is a function we will create in SLQFunctions.php and will to the database based on config.php file settings. $link is a variable where we store the connection so we can reference it\*/  
  $link = f\_sqlConnect();  
    
/\*We are going to pull out the destination table for this data\*/  
/\*isset() is a function that checks whether or not anything is stored under the name formID from $\_POST, if so, we are storing that as the table name\*/  
    if(isset($\_POST['formID'])){ $table = $\_POST['formID']; }   
    echo '<br>Destination Table: ' . $table;

/\*We are going to divide the values from the form in an array that we will use later\*/    
/\*implode a function that reads through \_POST and divides the names into an array, we will call it keys, and then we use it again for the \_POST values, put them into an array call values.\*/  
  $keys = implode(", ", (array\_keys($\_POST)));  
  echo '<br>Parsed Key :'.$keys;  
  $values = implode("', '", (array\_values($\_POST)));  
  echo '<br>Parsed Values :'.$values;

/\*We now want to record the source IP as well as date/time   
PHP offers some really cool features like detecting the IP that the \_POST came from and recording the time that the transaction happened.\*/      
  $x\_fields = 'ENTRY\_TimeStamp, SOURCE\_IP';  
  $x\_values = date("Y-m-d H:i:s") . "', '" . f\_getIP();  /\*f\_getIP() is a custom function from SQLFunctions.php\*/  
  echo '<br>x\_values :'.$x\_values;

/\*Then we want to confirm that the destination table exists prior to actually attempting to insert the data.  We use another custom function from SQLFunctions.php, f\_tableExists\*/  
/\*check to see if the table exists\*/  
  if (!f\_tableExists($link, $table, DB\_NAME) ) {  
    die('<br>Destination Table Does Not Exist:'.$table);  
  }

/\*Grab the success and reject URLs from the \_POST and store them in variables that we can use later.\*/  
  if(isset($\_POST['rejectredirecturl'])){   
    $rejectredirecturl = $\_POST['rejectredirecturl'];   
    echo '<br>rejectredirecturl :'.$rejectredirecturl;  
  }   
  if(isset($\_POST['successredirecturl'])){   
    $successredirecturl = $\_POST['successredirecturl'];   
    echo '<br>successredirecturl :'.$successredirecturl;  
  }

/\*We now assemble the SQL that will insert our values into the database.\*/  
  $sql="INSERT INTO $table ($keys, $x\_fields) VALUES ('$values', '$x\_values')";  
  echo '<br>sql :'.$sql;

/\*We attempt the SQL Insert, mysqli\_query() actually tries to execute the sql against the database connection provided. mysqli\_error() is a function that stores any MySQL errors that occurred when the script was run\*/  
  if (!mysqli\_query($link,$sql)) {  
    echo '<br>Error: ' . mysqli\_error($link);  
    if (!empty ($rejectredirecturl)) {  
                  /\*header Location with a url will redirect the user to the url specified.  
                    We have this row commented out so we can review the troubleshooting   
                    text written above\*/  
                  /\*header("Location: $rejectredirecturl?msg=1");\*/  
               }  
  }else if (!empty ($rejectredirecturl)) {  
                 /\*header("Location: $successredirecturl?msg=1");\*/  
               }

/\*Lastly - It’s always good practice to close the database connection\*/  
  mysqli\_close ($link);  
?>

Complete receiver - Text follow-up

Section 6, Lecture 91

All the code for this file was provided in the text lecture for the prior video, please reference that for comparisons.

Start SQL Functions - Text lecture

Section 6, Lecture 93

Create another file named SQLFunctions.php

Put the following in SQLFunctions.php  
<?php  
include('config.php');

function f\_sqlConnect() {  
  /\*mysqli is a php function that requires the database hostname, database name, username and password in order to create a database connection\*/  
  $link = new mysqli(DB\_HOST, DB\_USER, DB\_PWD, DB\_NAME);  
  if ($link->connect\_error) {  
               /\*if an error occurs while establishing the connection, stop processing and   
               write out an error message\*/  
    die("Connection failed: " . $link->connect\_error);  
  }   
  echo "<br>Connected successfully to the database<br><br>";

/\*return defines what gets sent back when the function completes.  In this case, a database connection variable named $link\*/  
  return $link;  
}

/\*This function is to check if the IP identified is in the correct format and not in a non-routable range.\*/  
function f\_validIP($ip) {  
       /\*!empty($ip) checks if $ip is populated\*/  
       /\*ip2long($ip) is a form of validation on the input\*/  
  if (!empty($ip) && ip2long($ip)!=-1) {  
               /\*Create an array of arrays standard non routable IPs so we can filter   
               out IPs that aren’t useful\*/  
    $reserved\_ips = array (  
      array('0.0.0.0','2.255.255.255'),  
      array('10.0.0.0','10.255.255.255'),  
                 array('127.0.0.0','127.255.255.255'),  
                 array('169.254.0.0','169.254.255.255'),  
                 array('172.16.0.0','172.31.255.255'),  
                 array('192.0.2.0','192.0.2.255'),  
                 array('192.168.0.0','192.168.255.255'),  
                 array('255.255.255.0','255.255.255.255')  
    );  
               /\*Compare the IP to each array and return a false if the IP is within any   
               of the ranges\*/  
    foreach ($reserved\_ips as $r) {  
      $min = ip2long($r[0]);  
      $max = ip2long($r[1]);  
      if ((ip2long($ip) >= $min) && (ip2long($ip) <= $max)) return false;  
    }  
               /\*if the ip is populated and isn’t in the non routable range, return true\*/  
    return true;   
  } else {  
      return false;       
  }  
}  
/\*This function attempts various methods to get a valid IP, checking each with the f\_validIP function.  \*/  
function f\_getIP() {  
  if (f\_validIP($\_SERVER["HTTP\_CLIENT\_IP"])) {  
    return $\_SERVER["HTTP\_CLIENT\_IP"];  
  }  
  foreach (explode(",",$\_SERVER["HTTP\_X\_FORWARDED\_FOR"]) as $ip) {  
    if (f\_validIP(trim($ip))) {  
      return $ip;  
      }  
    }  
  if (f\_validIP($SERVER["HTTP\_X\_FORWARDED"])) {  
    return $\_SERVER["HTTP\_X\_FORWARDED"];  
  } elseif (f\_validIP($\_SERVER["HTTP\_FORWARDED\_FOR"])) {  
    return $\_SERVER["HTTP\_FORWARDED\_FOR"];  
  } elseif (f\_validIP($\_SERVER["HTTP\_FORWARDED"])) {  
    return $\_SERVER["HTTP\_FORWARDED"];  
  } elseif (f\_validIP($\_SERVER["HTTP\_X\_FORWARDED"])) {  
    return $\_SERVER["HTTP\_X\_FORWARDED"];  
  } else {  
    return $\_SERVER["REMOTE\_ADDR"];  
  }  
}

/\*Check if table exists mysqli\_query is a standard php function that requires the following inputs:connection, query, and resultmode\*/    
/\*with $database = false, we create a new variable named $database and set it to false\*/  
function f\_tableExists(mysqli $link, $tablename, $database = false) {  
  if(!$database) {  
    $res = mysqli\_query($link, "SELECT DATABASE()");  
    $database = mysql\_result($res, 0);  
  }  
  $res = mysqli\_query($link," SELECT \*  
                                    FROM information\_schema.tables  
                                    WHERE table\_schema = '$database'  
                                    AND table\_name = '$tablename'"  
            );  
  echo '<br>Table Exist:'.($res->num\_rows);  
  return $res->num\_rows == 1;  
}

?>

Preview of web app - Text follow-up

Section 6, Lecture 95

Run the project go to OptIn page and try out some entries

You may run into an error if the FNAME and LNAME fields aren't set up in your db. Your challenge is to drop the OptIn table and create a new one, with the update FNAME and LNAME in place of NAME. Then it should work, if not, check the syntax for errors in receiver.php and SQLFunctions.php and the naming.

You may also want to update SELECT \* to SELECT COUNT(\*) as count in the f\_tableExists function of the SQLFunctions.php file

Add browser display - Text follow-up

Section 6, Lecture 97

Create a new file titled OptInDisplay.php

<?php  
include('SQLFunctions.php');

/\*Open the database connection based on config.php file settings\*/  
  $link = f\_sqlConnect();  
    
/\*Set Source Table\*/    
  $table = 'OptIn';  
  echo '<br>Source Table: ' . $table;  
    
/\*check to see if the table exists\*/  
  if (!f\_tableExists($link, $table, DB\_NAME) ) {  
    die('<br>Destination Table Does Not Exist:'.$table);  
  }

/\*Query contents of source table\*/  
  $sql="SELECT \* FROM $table ";  
  echo '<br>sql :'.$sql;  
    if ($result = mysqli\_query($link,$sql)){  
      echo "<table border=''1'' style=''width:100%''>";  
        //header  
        echo "<tr>";  
          echo "<td>OptInID</td>";  
          echo "<td>formID</td>";  
          echo "<td>First Name</td>";  
          echo "<td>Last Name</td>";  
          echo "<td>Email Addess</td>";  
          echo "<td>Date/Time</td>";  
          echo "<td>Source IP</td>";  
          echo "<td>SuccessUrl</td>";  
          echo "<td>RejectUrl</td>";            
        echo "</tr>";  
        
      //for each row returned in the query, separate them into <td> tags  
      while ($row = mysqli\_fetch\_array($result))  {  
        echo "<tr>";  
          echo "<td>{$row[0]}</td>";  
          echo "<td>{$row[1]}</td>";  
          echo "<td>{$row[2]}</td>";  
          echo "<td>{$row[3]}</td>";  
          echo "<td>{$row[4]}</td>";  
          echo "<td>{$row[5]}</td>";  
          echo "<td>{$row[6]}</td>";  
          echo "<td>{$row[7]}</td>";  
          echo "<td>{$row[8]}</td>";            
        echo "</tr>";  
      }   
      echo "</table>";  
    }  
    /\*mysqli\_free\_result() clears the $result variable\*/  
    mysqli\_free\_result($result);

/\*display any sql error encountered \*/  
  if (mysqli\_error($link)) {  
    echo '<br>Error: ' . mysqli\_error($link);  
  }else echo '<br>Success';

/\*Close database connection\*/  
  mysqli\_close ( $link );  
    
?>

Project - Solution text

Section 6, Lecture 99

Copy contents of OptIn.html into a new file, I used PoliticalPoll.html

<!DOCTYPE html>  
<HTML>  
  <HEAD>  
    <TITLE>Political Poll</TITLE>  
    <META http-equiv=Content-Type content="text/html; charset=utf-8">  
    <link href="styles.css" rel="stylesheet" type="text/css">    
<script type="text/javascript">   
function validateForm(){  
  return true;  
};               
</script>  
  </HEAD>   
  <BODY>  
      <div class="backgroundimage" >  
      <div id="main-content" class="main-content" >  
    <h1>Politics</h1>  
      <form action="receiver.php" method = "POST" onsubmit='return validateForm()' />  
        <input type="hidden" name="formID" value="PoliticalPoll" />  
        <input type="hidden" name="successredirecturl" value="Success.html">  
        <input type="hidden" name="rejectredirecturl" value="Fail.html">  
        <p>Gender:   
          <select name="GENDER" required>  
            <option value="">Please Select</option>  
            <option value="Male">Male</option>  
            <option value="Female">Female</option>  
            <option value="Other" >Other</option>  
          </select>  
        </p>  
        <p>Age:  <input type="number" name="AGE" min="0" max="110"  required/></p>  
        <p>How often do you vote:    
          <select name="VOTE\_FREQ" required>  
            <option value="">Please Select</option>  
            <option value="Always">Always</option>  
            <option value="Sometimes">Sometimes</option>  
            <option value="Never" >Never</option>  
          </select>  
        </p>  
        <p>Political Party:    
          <select name="PARTY" required>  
            <option value="">Please Select</option>  
            <option value="Republican">Republican</option>  
            <option value="Democrat">Democrat</option>  
            <option value="Libertarian">Libertarian</option>  
            <option value="Not Listed">Not Listed</option>  
            <option value="Not Affiliated">Not Affiliated</option>              
          </select>  
        </p>  
        <p>Who will you vote for:   
          <select name="CANDIDATE" required>  
            <option value="">Please Select</option>  
            <option value="Bernie">Bernie</option>  
            <option value="Hillary">Hillary</option>  
            <option value="Trump">Trump</option>  
            <option value="Cruz">Cruz</option>  
            <option value="Rubio">Rubio</option>              
          </select>  
        </p>          
        <br>  
        <input type="submit" class='button'>          
      </form>  
    </div>  
  </div>  
  </BODY>  
</HTML>

Create the Table:  
CREATE TABLE PoliticalPoll ( PoliticalPollID INT NOT NULL AUTO\_INCREMENT  
                            ,formID varchar(20) DEFAULT NULL  
                            ,GENDER varchar(20) DEFAULT NULL                              
                            ,AGE varchar(20) DEFAULT NULL  
                            ,VOTE\_FREQ varchar(20) DEFAULT NULL  
                            ,PARTY varchar(20) DEFAULT NULL  
                            ,CANDIDATE varchar(20) DEFAULT NULL  
                            ,ENTRY\_TimeStamp datetime DEFAULT NULL  
                            ,SOURCE\_IP char(15) DEFAULT NULL  
                            ,successredirecturl char(200) DEFAULT NULL  
                            ,rejectredirecturl char(200) DEFAULT NULL  
                            ,PRIMARY KEY (PoliticalPollID));

For browser viewing of table data copy over OptInDisplay.php and make necessary updates to column and row data  
OptInDisplayPoll.php  
<?php  
include('SQLFunctions.php');

/\*Open the database connection based on config.php file settings\*/  
  $link = f\_sqlConnect();  
    
/\*Set Source Table\*/    
    $table = 'PoliticalPoll';  
  echo '<br>Source Table: ' . $table;  
    
/\*check to see if the table exists\*/  
  if (!f\_tableExists($link, $table, DB\_NAME) ) {  
    die('<br>Destination Table Does Not Exist:'.$table);  
  }

/\*Query contents of source table\*/  
  $sql="SELECT \* FROM $table ";  
  echo '<br>sql :'.$sql;  
    if ($result = mysqli\_query($link,$sql)){  
      echo "<table border=''1'' style=''width:100%''>";  
        //header  
        echo "<tr>";  
          echo "<td>PoliticalPollID</td>";  
          echo "<td>formID</td>";  
          echo "<td>Gender</td>";  
          echo "<td>Age</td>";  
          echo "<td>Voting Frequency</td>";  
          echo "<td>Party</td>";            
          echo "<td>Cadidate</td>";  
          echo "<td>EntryTimeStamp</td>";             
          echo "<td>Source IP</td>";  
          echo "<td>SuccessUrl</td>";  
          echo "<td>RejectUrl</td>";            
        echo "</tr>";  
        
      //data    
      while ($row = mysqli\_fetch\_array($result))  {  
        echo "<tr>";  
          echo "<td>{$row[0]}</td>";  
          echo "<td>{$row[1]}</td>";  
          echo "<td>{$row[2]}</td>";  
          echo "<td>{$row[3]}</td>";  
          echo "<td>{$row[4]}</td>";  
          echo "<td>{$row[5]}</td>";  
          echo "<td>{$row[6]}</td>";  
          echo "<td>{$row[7]}</td>";  
          echo "<td>{$row[8]}</td>";            
          echo "<td>{$row[9]}</td>";               
          echo "<td>{$row[10]}</td>";                         
        echo "</tr>";  
      }   
      echo "</table>";  
    }  
    mysqli\_free\_result($result);

  if (mysqli\_error($link)) {  
    echo '<br>Error: ' . mysqli\_error($link);  
  }else echo '<br>Success';

/\*Close database connection\*/  
  mysqli\_close ( $link );  
    
?>

Questions

Top of Form



3 questions in this lecture

Bottom of Form

https://udemy-images.udemy.com/user/50x50/anonymous.png

Political Poll

[image]I only asked about what party they were in.  I don't recall too much of my HTML work, so I modified the receiver.php to check for legitimate parties.  Which brought up a question that I have.  Is it better to do checking of user inputs on the HTML side or on the server side? Or does it make a difference?Here is my new receiver.php.<?php/\*Bring in our custom library of php and functions, we will create this next\*/include('SQLFunctions.php'); /\*First we are going to handle what is being submitted in the form by displaying it, so we display what is in the \_POST\*//\*echo simply means write the following out to the page, adding this description and the line breaks will make the output legible \*//\*The echo statements aren’t actually needed, but they are really helpful in troubleshooting\*/  echo '<br>Display full contents of the \_POST: <br>';/\*var\_dump is a function that write all of names and values from the post\*/  var\_dump($\_POST); /\*We need to esatblish a connection with the database   f\_sqlConnect() is a function we will create in SLQFunctions.php and will to the database based on config.php file settings. $link is a variable where we store the connection so we can reference it\*/  $link = f\_sqlConnect();  /\*We are going to pull out the destination table for this data\*//\*isset() is a function that checks whether or not anything is stored under the name formID from $\_POST, if so, we are storing that as the table name\*/    if(isset($\_POST['formID'])){ $table = $\_POST['formID']; }     echo '<br>Destination Table: ' . $table;        if (isset($\_POST['PARTY'])){ $party = $\_POST['PARTY']; }     $partynum = 0;    if ($party == 'Republican') {$partynum = 1;}    if ($party == 'Democrat') {$partynum = 2;}    if ($party == 'Green') {$partynum = 3;}    if ($party == 'Communist') {$partynum = 4;}    if ($party == 'Libertarian') {$partynum = 5;}        if ($partynum == 0)    {       echo '<br>Not an allowed party: '.$partynum;       $nopartyredirecturl = $\_POST['nopartyredirecturl'];        echo '<br>Redirect to: '.$nopartyredirecturl;       header("Location: $nopartyredirecturl?msg=1");    }    else    {       /\*We are going to divide the values from the form in an array that we will use later\*/        /\*implode a function that reads through \_POST and divides the names into an array, we will call it keys, and then we use it again for the \_POST values, put them into an array call values.\*/        $keys = implode(", ", (array\_keys($\_POST)));        echo '<br>Parsed Key :'.$keys;        $values = implode("', '", (array\_values($\_POST)));        echo '<br>Parsed Values :'.$values;            /\*We now want to record the source IP as well as date/time       PHP offers some really cool features like detecting the IP that the \_POST came from and recording the time that the transaction happened.\*/            $x\_fields = 'ENTRY\_TimeStamp, SOURCE\_IP';        $x\_values = date("Y-m-d H:i:s") . "', '" . f\_getIP();  /\*f\_getIP() is a custom function from SQLFunctions.php\*/        echo '<br>x\_values :'.$x\_values;            /\*Then we want to confirm that the destination table exists prior to actually attempting to insert the data.  We use another custom function from SQLFunctions.php, f\_tableExists\*/      /\*check to see if the table exists\*/        if (!f\_tableExists($link, $table, DB\_NAME) ) {          die('<br>Destination Table Does Not Exist:'.$table);        }            /\*Grab the success and reject URLs from the \_POST and store them in variables that we can use later.\*/        if(isset($\_POST['rejectredirecturl'])){           $rejectredirecturl = $\_POST['rejectredirecturl'];           echo '<br>rejectredirecturl :'.$rejectredirecturl;        }         if(isset($\_POST['successredirecturl'])){           $successredirecturl = $\_POST['successredirecturl'];           echo '<br>successredirecturl :'.$successredirecturl;        }               /\*We now assemble the SQL that will insert our values into the database.\*/        $sql="INSERT INTO $table ($keys, $x\_fields) VALUES ('$values', '$x\_values')";        echo '<br>sql :'.$sql;            /\*We attempt the SQL Insert, mysqli\_query() actually tries to execute the sql against the database connection provided. mysqli\_error() is a function that stores any MySQL errors that occurred when the script was run\*/        if (!mysqli\_query($link,$sql)) {          echo '<br>Error: ' . mysqli\_error($link);          if (!empty ($rejectredirecturl)) {                        /\*header Location with a url will redirect the user to the url specified.                          We have this row commented out so we can review the troubleshooting                           text written above\*/                       /\*header("Location: $rejectredirecturl?msg=1");\*/                     }        }else if (!empty ($rejectredirecturl)) {                       /\*header("Location: $successredirecturl?msg=1");\*/                     }    }      /\*Lastly - It’s always good practice to close the database connection\*/    mysqli\_close ($link);?>

https://udemy-images.udemy.com/user/50x50/anonymous.png

Political Poll Assignment

[image]<div class="backgrounds">        <div class="main-content">            <h1>Political Poll</h1>                        <form action="receiver.php" method="POST">                <input type="hidden" name="formID" value="PoliticalPoll"/>                <input type="hidden" name="successredirecturl" value="Success.html"/>                <input type="hidden" name="rejectredirecturl" value="Fail.html"/>                <p>Gender:                     <select name="gender" required>                        <option value="">Please Select</option>                                                <option value="female">Female</option>                        <option value="male">Male</option>                        <option value="dna">Do not want to answer</option>                    </select>                </p>                <p>Age: <input type="number" name="age" min="0" max="110" required/></p>                <p>How often do you vote?                    <select name="vote\_freq" required>                        <option value="">Please Select</option>                                                <option value="always">Always</option>                        <option value="sometime">Sometimes</option>                        <option value="never">Never</option>                        <option value="dna">Do not want to answer</option>                    </select>                </p>                <p>What party do you vote?                    <select name="party" required>                        <option value="">Please Select</option>                        <option value="democat">Democat</option>                        <option value="indepentent">Independent</option>                        <option value="rebublican">Republican</option>                        <option value="other">Other</option>                        <option value="dna">Do not want to answer</option>                    </select>                </p>                <p>Which canditate will you vote for?                    <select name="candidate" required>                        <option value="">Please Select</option>                        <option value="leonardo">Leonardo</option>                        <option value="donatello">Donatello</option>                        <option value="raphael">Raphael</option>                        <option value="michelangelo">Michelangelo</option>                        <option value="dna">Do not want to answer</option>                    </select>                                </p>                                <h3>Thank you for your time!</h3>                <input type="submit" class="button" value="Submit"/>            </form>                         </div><!--/main-content-->    </div><!--/backgrounds-->

https://udemy-images.udemy.com/user/50x50/18016430_c1d7.jpg

My Political Poll (check the picture)

[image] TABLE: CREATE TABLE PoliticalPoll  (PollID INT NOT NULL AUTO\_INCREMENT,                    formID VARCHAR(50) DEFAULT NULL,                    GENDER VARCHAR(20) DEFAULT NULL,                    AGE VARCHAR(20) DEFAULT NULL,                    FREQUENCY VARCHAR(20) DEFAULT NULL,                    POLITICS VARCHAR(20) DEFAULT NULL,                    PARTY VARCHAR(20) DEFAULT NULL,                    CANDIDATE VARCHAR(20) DEFAULT NULL,                    ENTRY\_TimeStamp datetime DEFAULT NULL,                    SOURCE\_IP CHAR(15) DEFAULT NULL,                    successredirecturl CHAR(200) DEFAULT NULL,                    rejectredirecturl CHAR(200) DEFAULT NULL,                    PRIMARY KEY (PollID)); FORM: <form action="receiver.php" method="POST">                    <input type="hidden" name="formID" value="PoliticalPoll">                    <input type="hidden" name="successredirecturl" value="Success.html">                    <input type="hidden" name="rejectredirecturl" value="Fail.html">                    <p>Gender : <select name="Gender" required>                                <option value="">Please Select</option>                                <option value="Male">Male</option>                                <option value="Female">Female</option>                                <option value="Not disclosed">Prefer not to disclose</option>                              </select></p>                     <p>Age : <select name="Age" required>                                <option value="">Please Select</option>                                <option value="25 or less">25 os less</option>                                <option value="Between 26 and 35">Between 26 and 35</option>                                <option value="Between 36 and 45">Between 36 and 45</option>                                <option value="Between 46 and 59">Between 46 and 59</option>                                <option value="60 or more">60 or more</option>                                <option value="Not disclosed">Prefer not to disclose</option>                              </select></p>                     <p>How often do you vote? <select name="Frequency" required>                                <option value="">Please Select</option>                                <option value="Always">Always</option>                                <option value="Almost always">Almost always</option>                                <option value="Sometimes">Sometimes</option>                                <option value="Almost never">Almost never</option>                                <option value="Never">Never</option>                                <option value="Not disclosed">Prefer not to disclose</option>                              </select></p>                     <p>Do you follow politics? <select name="Politics" required>                                <option value="">Please Select</option>                                <option value="Always">Always</option>                                <option value="Almost always">Almost always</option>                                <option value="Sometimes">Sometimes</option>                                <option value="Almost never">Almost never</option>                                <option value="Never">Never</option>                                <option value="Not disclosed">Prefer not to disclose</option>                              </select></p>                     <p>Political party : <select name="Party" required>                                <option value="">Please Select</option>                                <option value="Republican">Republican</option>                                <option value="Democrat">Democrat</option>                                <option value="Libertarian">Libertarian</option>                                <option value="Not Listed">Not Listed</option>                                <option value="Not Affilliated">Not Affilliated</option>                                <option value="Not disclosed">Prefer not to disclose</option>                              </select></p>                     <p>Who will you vote for? <select name="Candidate" required>                                <option value="">Please Select</option>                                <option value="Bernie">Bernie</option>                                <option value="Trump">Trump</option>                                <option value="Hilary">Hilary</option>                                <option value="Cruz">Cruz</option>                                <option value="Rubio">Rubio</option>                                <option value="Not disclosed">Prefer not to disclose</option>                              </select></p>                     <input type="submit" class="button">                                    </form>

CRUD - Intro - Text follow-up

Section 7, Lecture 101

Create Read Update Delete

First in mysql cli, create a database and table for the project “To Do List”  
CREATE DATABASE CRUD;

Then use the database  
USE CRUD;

Create the table  
CREATE TABLE ToDos ( ToDoID INT NOT NULL AUTO\_INCREMENT  
                    ,ToDoTitle varchar(50) DEFAULT NULL  
                    ,ToDoDescription varchar(1000) DEFAULT NULL  
                    ,Complete boolean DEFAULT NULL   
                    ,ToDueDate datetime DEFAULT NULL   
                    ,EntryTS datetime DEFAULT NULL  
                    ,UpdateTS datetime DEFAULT NULL  
                    ,CompleteTS datetime DEFAULT NULL  
                    ,PRIMARY KEY (ToDoID));

Make a test entry:  
INSERT INTO ToDos (ToDoTitle , ToDoDescription)   
VALUES ('Laundry' , 'Do the Laundry');

Create new folder ToDoCrud under workspace

Create a new file titled ToDoApp.php

Fill the file will the following content:  
<?php

/\*Connect to CRUD Database  mysqli(Server,User,Password,Database)\*/  
$link = new mysqli('localhost', 'root', '', 'CRUD');

/\*Write error if they exist, otherwise, write success\*/  
if ($link->connect\_error) {  
  die("Connection failed: " . $link->connect\_error);  
} else {echo "<br>Connected successfully"; }

/\*Close database connection\*/  
  mysqli\_close ( $link );  
?>  
Click Run Project

Open the url from Apache

Open the ToDoCrud folder

Then open ToDoApp.php

The page should say “Connected successfully” confirming that you application was able to find the database and connect to it.

Next step, insert rows

Every time you access the page in your browser, it inserts a row into the table due to the code below that will be added now.    
<?php

/\*Connect to CRUD Database  mysqli(Server,User,Password,Database)\*/  
$link = new mysqli('localhost', 'root', '', 'CRUD');

if ($link->connect\_error) {  
  die("Connection failed: " . $link->connect\_error);  
} else {echo "<br>Connected successfully"; }  
?>

//adding basic CSS color and border to the table, th, and td tags  
<style>  
table, th, td { border: 1px solid black;  
                border-collapse: collapse; }  
table th { background-color: black;  
           color: white; }      
table tr:nth-child(even) { background-color: #eee; }  
table tr:nth-child(odd)  { background-color: #fff; }  
</style>

<body>  
    
<?php

/\*Set Up the SQL statement\*/  
$sql = "INSERT INTO ToDos (ToDoTitle, ToDoDescription) VALUES ('Test', 'TestDescription');";

/\*Exectute the statement, and write the results\*/  
if (mysqli\_query($link, $sql)) {  
    echo "<br>New record created successfully";  
} else {  
    echo "Error: " . $sql . "<br>" . mysqli\_error($link);  
}  
    
if (mysqli\_error($link)) {  
    echo '<br>Error: ' . mysqli\_error($link);  
    }else echo '<br>Success';

/\*Close database connection\*/  
  mysqli\_close ( $link );  
?>

To reduce the amount of code we need to write and make it a bit more versatile going forward, let's make a config file and a functions file.

Create a new file titled config.php with the following content:  
<?php  
/\*Configuration Settings\*/

define('DB\_HOST', 'localhost'); /\*Database Server\*/  
define('DB\_NAME', 'CRUD'); /\*Database Name\*/  
define('DB\_USER', 'root'); /\*Database Username\*/  
define('DB\_PWD',  ''); /\*Database Password\*/  
?>

Then make an SQLFunctions.php file with the following content:  
<?php  
include('config.php');

/\*Opens connection to database with credentials\*/  
function connectDB() {  
    $link = new mysqli(DB\_HOST, DB\_USER, DB\_PWD, DB\_NAME);  
  if ($link->connect\_error) {  
    die("Connection failed: " . $link->connect\_error);  
  }   
  /\*echo "<br>Connected successfully"; \*/  
  return $link;  
}  
?>

We’ll use these going forward to connect to the database.

Create action - Text lecture

Section 7, Lecture 103

Create a new file titled CreateToDo.php, this will be our form to create new to-do's from the browser  
<HTML>  
  <HEAD>  
    <TITLE>CreateToDo</TITLE>  
    <META http-equiv=Content-Type content="text/html; charset=utf-8">  
    <script type="text/javascript">   
      function validateForm(){  
        //this is just a placeholder incase we wanted add additional javascript type validations.  
        return true;  
      };               
    </script>  
  </HEAD>   
  <BODY>  
    <h1>New To-do</h1>  
      <form action="CreateToDoSubmit.php" method="POST" onsubmit='return validateForm()' />  
        <p>To-do Title:  <input type="text" name="ToDoTitle" maxlength='50' required/></p>  
        <p>To-Due Date:  <input type="date" name="ToDueDate"></p>  
        <p>Description:<br> <textarea cols="100" rows="5" name="ToDoDesription" maxlength='1000' required>      </textarea></p>    
        <input type="submit">         
      </form>  
  </BODY>  
</HTML>

Test it out by visiting the page.

Now that we are using an html form to collect the data, it will submit it to another file we need to create called CreateToDoSubmit.php

Create the file CreateToDoSubmit.php

This file will receive the data from the html form and insert it into the database and then return the user back to the ToDoApp.php page

Insert the following into CreateToDoSubmit.php

<?php  
include('SQLFunctions.php');

if ( !empty($\_POST)) {  
  // Store data from html form POST action into variables  
  $tdTitle = $\_POST['ToDoTitle'];  
  $tdDate  = $\_POST['ToDueDate'];  
  $tdDescr = $\_POST['ToDoDescription'];  
           
/\*Open the database connection based on config.php file settings\*/  
  $link = connectDB();

  /\*Prepare the SQL INSERT Statement\*/  
  $sql = "INSERT INTO ToDos (ToDoTitle, ToDoDescription, ToDueDate, EntryTS) VALUES ('".$tdTitle."','".$tdDescr."','".$tdDate."', NOW());";  
  /\*Insert values into the database\*/  
  if (mysqli\_query($link, $sql)) {  
  /\*    echo "<br>New record created successfully";\*/  
  } else {  
      echo  "<br>Error: " . $sql . "<br>" . mysqli\_error($link);  
  }

/\*Close database connection\*/  
mysqli\_close ( $link );

/\*Forward User Back to Main View\*/    
header("Location: ToDoApp.php");

}

?>

Read - Text lecture

Section 7, Lecture 105

Lets update ToDoApp.php to display all of the rows on the table

<?php  
include('SQLFunctions.php');  
?>  
<html>  
<!--The Style tag allows us to put some basic css shading and borders to make the table a little easier to look at. Table, th and td are elements of an html table.-->  
<head>  
<style>  
  table, th, td { border: 1px solid black;  
                  border-collapse: collapse; }  
  table th { background-color: black;  
             color: white; }      
  table tr:nth-child(even) { background-color: #eee; }  
  table tr:nth-child(odd)  { background-color: #fff; }  
</style>  
</head>    
<body>  
      <h1>To-do Main View</h1>  
      <a href="CreateToDo.php"><button>New To-do</button></a>  
<?php        
  /\*Create the SQL Statement, selecting the four columns were are interested in\*/  
  /\*format the date to display it easier\*/  
  $sql="SELECT ToDoTitle  
                ,ToDoDescription  
                ,DATE\_FORMAT(ToDueDate,'%m-%d-%Y')    
                ,ToDoID  
          FROM ToDos;";    
  echo '<br>sql :'.$sql.'<br>Comment this out, after testing<br><br>';  
   
   /\*Open the database connection based on config.php file settings\*/  
  $link = connectDB();

  /\*Execute the sql and if there is a result, write out the table headers, then rows\*/  
  if ($result = mysqli\_query($link,$sql)){  
      echo "<table>";  
        //header  
        echo "<tr>";  
          echo "<th>Title</td>";  
          echo "<th>Description</td>";  
          echo "<th>DueDate</td>";  
          echo "<th>Action</td>";     
        echo "</tr>";  
        
      //rows, use a while loop to write out each field in the result set.  
      //mysqli\_fetch\_array() separates the results into an array named  $row so   
      //that each field can be referrenced using $row[x]  
      while ($row = mysqli\_fetch\_array($result))  {  
        echo "<tr>";  
          echo "<td>{$row[0]}</td>";  
          echo "<td>{$row[1]}</td>";  
          echo "<td>{$row[2]}</td>";  
          echo "<td>Link To Update Page for <br> ToDoID {$row[3]}</td>";  
        echo "</tr>";  
      }   
      echo "</table>";  
    }  
      
  /\*Close database connection\*/  
  mysqli\_close ( $link );  
?>  
</body>  
</html>

We have now Completed “READ”

Update action - Text lecture

Section 7, Lecture 107

Update

Create a new file titled UpdateToDo.php and put the following in it  
<?php        
  include('SQLFunctions.php');

   /\*The Read page ToDoApp.php is going to link to this page by sending an html form POST   
   with the ToDoID as the only input. We could name it anything. In this case, we named it "q" \*/  
   $q=$\_POST["q"];  
     
   /\*Open the database connection based on config.php file settings\*/  
   $link = connectDB();  
     
    /\*Create the Sql Statement\*/  
    $sql = "SELECT ToDoID, ToDoTitle, ToDoDescription, ToDueDate FROM ToDos WHERE ToDoID = 7";     
    /\*$sql = "SELECT ToDoID, ToDoTitle, ToDoDescription, ToDueDate FROM ToDos WHERE ToDoID =".$q;\*/  
    /\*We will use the hard-coded sql statement for now\*/  
    echo '<br>sql :'.$sql.'<br>Comment this out, after testing<br><br>';  
      
      
    /\*If the $sql passes validation, exectute it\*/  
    if($stmt = $link->prepare($sql))  
    {  
        $stmt->execute();  
        /\*Assign the results into their respective php variables\*/  
        $stmt->bind\_result($ToDoID, $ToDoTitle, $ToDoDescription, $ToDueDate);  
        while ($stmt->fetch())  
        {  
          /\*reformat the date to html\*/  
          $newToDueDate = date("Y-m-d", strtotime($ToDueDate));  
          echo "<BODY>";  
          echo "  <div>";  
          echo "    <div>";  
          echo "    <h1>Update To-do</h1>";  
          /\*Create and prepopulate an html form with the values pulled from the database.\*/  
          echo "    <form action='UpdateToDoCommit.php' method = 'POST' onsubmit='' />";  
          echo "      <input type='hidden' name='ToDoId' value='".$ToDoID."'>";  
          echo "      <p>To-do Title:  <input text='text' name='ToDoTitle' maxlength='50'  required value='".$ToDoTitle."'/></p>";  
          echo "      <p>To-Due Date:  <input type='date' name='ToDueDate' value='".$newToDueDate."''></p>";  
          echo "      <p>Description:<br> <textarea cols='100' rows='5' name='ToDoDesr' maxlength='1000'  required>".$ToDoDescription."</textarea></p>";  
          echo "      <input type='submit'> ";  
          echo "    </form>";  
          echo "    <a href='ToDoApp.php'><button>Cancel</button></a>";  
          echo "    </div>";  
          echo "  </div>";  
          echo "</BODY>";      
        }  
    }  
    else  {   
      echo 'Unable to connect';   
      exit();  
    }  
?>

This should look almost exactly like the Create upon testing in the browser

The plan is to query the database, then populate these fields when the file is loaded.  
Then when submit is clicked, the row is updated in the database.

To update the database, create a new file titled UpdateToDoCommit.php  
<?php        
include('SQLFunctions.php');

// If there is anything in the POST, store the data from the form into variables  
if ( !empty($\_POST)) {  
  $tdID = $\_POST['ToDoId'];  
  $tdTitle = $\_POST['ToDoTitle'];  
  $tdDate  = $\_POST['ToDueDate'];  
  $tdDescr = $\_POST['ToDoDesr'];  
           
  /\*Open the database connection based on config.php file settings\*/  
  $link = connectDB();

  /\*Prepare the SQL INSERT Statement\*/  
  $sql = "UPDATE ToDos  
          SET ToDoTitle = '".$tdTitle."'  
             ,ToDoDescription = '".$tdDescr."'  
             ,ToDueDate ='".$tdDate."'  
             ,UpdateTS =  NOW()  
          WHERE ToDoId = ".$tdID.";";  
  echo $sql."<br>Comment this out, once tested";   
    
  /\*Insert values into the database\*/  
  if (mysqli\_query($link, $sql)) {  
      echo "<br>Update record successfully";  
  } else {  
      echo  "<br>Error: " . $sql . "<br>" . mysqli\_error($link);  
  }

/\*Close database connection\*/  
mysqli\_close ( $link );

/\*Forwarded User Back to Main View\*/    
/\*header("Location: ToDoApp.php");  Uncomment this after testing \*/

}

?>

Test it out by submitting an update from UpdateToDo.php

Update ToDoApp.php as follows:  
Replace the last echo td with this:  
          echo "<td><form action='UpdateToDo.php' method = 'POST' onsubmit='' /> <input type='hidden' name='q' value='".$row[3]."' /><input type='Submit' value='Update'></form></td>";

Also, comment out the echo $SQL

<?php  
include('SQLFunctions.php');  
?>  
<html>  
<!--The Style tag allows us to put some basic css shading and borders to make the table a little easier to look at.  Table, th and td are elements of an html table.-->  
<style>  
  table, th, td { border: 1px solid black;  
                  border-collapse: collapse; }  
  table th { background-color: black;  
             color: white; }      
  table tr:nth-child(even) { background-color: #eee; }  
  table tr:nth-child(odd)  { background-color: #fff; }  
</style>    
  <body>  
      <h1>To-do Main View</h1>  
      <a href="CreateToDo.php"><button>New To-do</button></a>  
<?php        
  /\*Create the SQL Statement, selecting the four columns were are interested in\*/  
  /\*format the date to display it easier\*/  
  $sql="SELECT ToDoTitle  
                ,ToDoDescription  
                ,DATE\_FORMAT(ToDueDate,'%m-%d-%Y')    
                ,ToDoID  
          FROM ToDos;";    
  /\*echo '<br>sql :'.$sql.'<br>Comment this out, after testing<br><br>';\*/  
   
   /\*Open the database connection based on config.php file settings\*/  
  $link = connectDB();

  /\*Execute the sql and if there is a result, write out the table headers, then rows\*/  
  if ($result = mysqli\_query($link,$sql)){  
      echo "<table>";  
        //header  
        echo "<tr>";  
          echo "<th>Title</td>";  
          echo "<th>Description</td>";  
          echo "<th>DueDate</td>";  
          echo "<th>Action</td>";     
        echo "</tr>";  
        
      //rows, use a while loop to write out each field in the result set.  
      //mysqli\_fetch\_array() separates the results into an array named  $row so   
      //that each field can be referrenced using $row[x]  
      while ($row = mysqli\_fetch\_array($result))  {  
        echo "<tr>";  
          echo "<td>{$row[0]}</td>";  
          echo "<td>{$row[1]}</td>";  
          echo "<td>{$row[2]}</td>";  
          echo "<td><form action='UpdateToDo.php' method = 'POST' onsubmit='' /> <input type='hidden' name='q' value='".$row[3]."' /><input type='Submit' value='Update'></form></td>";  
        echo "</tr>";  
      }   
      echo "</table>";  
    }  
      
  /\*Close database connection\*/  
  mysqli\_close ( $link );  
?>

  </body>  
</html>

Back on UpdateToDo.php  
Comment out the hard coded  $sql row, and uncomment the dynamic one.  
Test it out by updating a few existing ToDo records.

Delete action - Text lecture

Section 7, Lecture 109

Delete ->

Create a new field titled DeleteToDo.php  
This file will take the input from a POST, and delete the corresponding ToDo row.  
<?php  
  include('SQLFunctions.php');

/\*if anything is in the Post, assign the $tdID variable with the ID from the post\*/  
if ( !empty($\_POST)) {  
  $tdID = $\_POST['q'];  
   /\*Open the database connection based on config.php file settings\*/  
  $link = connectDB();

  /\*Prepare the SQL Delete Statement using the ID from the POST\*/  
  $sql = "DELETE  
          FROM ToDos  
          WHERE ToDoId = ".$tdID.";";  
  echo "sql:".$sql." Comment this out after testing";   
    
  /\*Attempt Delete\*/  
  if (mysqli\_query($link, $sql)) {  
      echo "<br>Delete record successfully";  
  } else {  
      echo  "<br>Error: " . $sql . "<br>" . mysqli\_error($link);  
  }

/\*Close database connection\*/  
mysqli\_close ( $link );

/\*Forwared User Back to Main View\*/   
/\*header("Location: ToDoApp.php");  uncomment this after testing\*/  
}  
?>

To Test it, we will need to update ToDoApp.php  
Update the while statement on ToDoApp.php and add a delete button under update, ensure you've moved the </td> to the end of the delete button from the update  
      while ($row = mysqli\_fetch\_array($result))  {  
        echo "<tr>";  
          echo "<td>{$row[0]}</td>";  
          echo "<td>{$row[1]}</td>";  
          echo "<td>{$row[2]}</td>";  
          echo "<td><form action='UpdateToDo.php' method = '\_POST' onsubmit='' /> <input type='hidden' name='q' value='".$row[3]."' /><input type='Submit' value='Update'></form>";  
          echo "<form action='DeleteToDo.php' method = '\_POST' onsubmit='' /> <input type='hidden' name='q' value='".$row[3]."' /><input type='Submit' value='Delete'></form></td>";  
        echo "</tr>";  
      }

Now test it out.  You will quickly find that you can delete rows with the touch of a button.

Once tested, Update the DeleteToDo.php  
Comment out the echo sql  
Uncomment the header Location ToDoApp.php at the end. Test out the functions of the ToDo app and enjoy!

Intro to authentication - Text lecture

Section 8, Lecture 111

We want our application to allow users to log in using a username and password. The application will check that the user exists in the database and confirm that their password is correct.

We also want to force users to log in. If they try to access pages when they aren’t logged in, we need to reroute them to the login page.

Php offers a way to store information that can be accessed across multiple pages call sessions. A session is unlike a cookie in that the information is stored on the server instead of the user’s machine.    
http://www.w3schools.com/php/php\_sessions.asp  
Well put sessions to work after we create the MySQL table.

Start by creating a User Definition table  
CREATE TABLE User\_Dfn ( User\_ID INT NOT NULL AUTO\_INCREMENT  
                       ,username varchar(20) NOT NULL  
                       ,pwd varchar(40) NOT NULL  
                       ,PRIMARY KEY (User\_ID));

Now create a page to add new users. Create a new file named AddUser.php  
This page will simply collect the user name and password that we want to use, and send it in a post action to AddUserSubmit.php.

<?php        
/\* begin our session \*/  
session\_start();  
?>

<html>  
<head>  
  <title>Add New User</title>  
</head>

<body>  
<h2>Add New User</h2>  
<form action="AddUserSubmit.php" method="post">  
  <fieldset>  
    <p>  
      <label>Username</label>  
      <input type="text" name="username" value="" maxlength="20" required/>  
      <i>(4-20 characters)</i>  
    </p>  
    <p>  
      <label>Password</label>  
      <input type="password" name="pwd" value="" maxlength="20" required/>  
      <i>(4-20 characters)</i>  
    </p>  
    <p>   
      <input type="submit" value="Add User" />  
    </p>  
  </fieldset>  
</form>  
</body>  
</html>

Next, Add a new file titled AddUserSubmit.php

For filter\_var FILTER\_SANITIZE\_STRING function check out the link below:  
http://www.w3schools.com/php/filter\_sanitize\_string.asp  
This function removes special characters which can cause many issues including possible security risks.

<?php  
require\_once('SQLFunctions.php');  
session\_start();

/\* Check that username, and password are populated\*/  
if(!isset( $\_POST['username'], $\_POST['pwd']))  
{  
    $message = 'Please enter a valid username and password';  
}  
/\* Check username length is not more than 20 and not less than 4\*/  
elseif (strlen( $\_POST['username']) > 20 || strlen($\_POST['username']) < 4)  
{  
    $message = 'Incorrect Length for Username';  
}  
/\* Check password length is not more than 20 and not less than 4\*/  
elseif (strlen( $\_POST['pwd']) > 20 || strlen($\_POST['pwd']) < 4)  
{  
    $message = 'Incorrect Length for Password';  
}  
/\* Check the username for alpha numeric characters \*/  
elseif (ctype\_alnum($\_POST['username']) != true)  
{  
    $message = "Username must be alpha numeric";  
}  
/\* Check the password for alpha numeric characters \*/  
elseif (ctype\_alnum($\_POST['pwd']) != true)  
{  
        $message = "Password must be alpha numeric";  
}  
else  
{  
    /\* Store username and pwds as variable \*/  
    /\*Use filter\_var to remove special characters from the inputs\*/  
    $username = filter\_var($\_POST['username'], FILTER\_SANITIZE\_STRING);  
    $pwd = filter\_var($\_POST['pwd'], FILTER\_SANITIZE\_STRING);

    /\* Encrypt with password with sha1, a cryptographic hash function  \*/  
    /\* Never store plain text passwords in the database\*/  
    $pwd = sha1( $pwd );

We will continue building this in the next video

Complete user sign-up - Text lecture

Section 8, Lecture 113

Below is the full AddUserSubmit.php file:

<?php  
require\_once('SQLFunctions.php');  
session\_start();

/\* Check that username, and password are populated\*/  
if(!isset( $\_POST['username'], $\_POST['pwd']))  
{  
    $message = 'Please enter a valid username and password';  
}  
/\* Check username length is not more than 20 and not less than 4\*/  
elseif (strlen( $\_POST['username']) > 20 || strlen($\_POST['username']) < 4)  
{  
    $message = 'Incorrect Length for Username';  
}  
/\* Check password length is not more than 20 and not less than 4\*/  
elseif (strlen( $\_POST['pwd']) > 20 || strlen($\_POST['pwd']) < 4)  
{  
    $message = 'Incorrect Length for Password';  
}  
/\* Check the username for alpha numeric characters \*/  
elseif (ctype\_alnum($\_POST['username']) != true)  
{  
    $message = "Username must be alpha numeric";  
}  
/\* Check the password for alpha numeric characters \*/  
elseif (ctype\_alnum($\_POST['pwd']) != true)  
{  
        $message = "Password must be alpha numeric";  
}  
else  
{  
    /\* Store username and pwds as variable \*/  
    /\*Use filter\_var to remove special characters from the inputs\*/  
    $username = filter\_var($\_POST['username'], FILTER\_SANITIZE\_STRING);  
    $pwd = filter\_var($\_POST['pwd'], FILTER\_SANITIZE\_STRING);

    /\* Encrypt with password with sha1, a cryptographic hash function  \*/  
    /\* Never store plain text passwords in the database\*/  
    $pwd = sha1( $pwd );

    try  
    {  
         /\*Connect to CRUD Database  mysqli(Server,User,Password,Database)\*/  
        $link = connectDB();

        /\* Check that username does not already exist \*/  
        $sql = "SELECT 1 FROM User\_Dfn WHERE username = '".$username."'";  
        if($result=mysqli\_query($link,$sql))   
        {  
            if(mysqli\_num\_rows($result)>=1) {  
              $message = "Username already exists";  
            } else {  
              /\* Prepare the sql insert statement \*/  
              $sql = "INSERT INTO User\_Dfn (username, pwd ) VALUES ('".$username."', '".$pwd."')";  
              if (mysqli\_query($link, $sql)) {  
                $message = 'New user added';  
              } else { echo  "<br>Error: " . $sql . "<br>" . mysqli\_error($link);  }  
            }  
        }  
    }  
    catch(Exception $e)  
    {  
        $message = 'Unable to process request';  
    }  
}  
?>

<html>  
  <head>  
    <title>Add New User</title>  
  </head>  
  <body>  
    <!-- Message is a variable that was populated previously based on the php above  -->      
    <p><?php echo $message; ?>  
  </body>  
</html>

Try the form out.  
There are all sorts of good security practices that we could implement, but for now, this  will accomplish basic functionality.

Login users - Text lecture

Section 8, Lecture 115

Now lets create a Login page Login.php

<html>  
<head>  
<title>CRUD Login</title>  
</head>

<body>  
  <h2>CRUD Login</h2>  
  <form action="LoginSubmit.php" method="post">  
  <fieldset>  
    <p>  
      <label>Username</label>  
      <input type="text" name="username" value="" maxlength="20" />  
    </p>  
    <p>  
      <label>Password</label>  
      <input type="password"  name="pwd" value="" maxlength="20" />  
    </p>  
    <p>  
      <input type="submit" value="Login" />  
    </p>  
  </fieldset>  
  </form>  
</body>  
</html>

Create another file LoginSubmit.php  
<?php  
require\_once('SQLFunctions.php');  
session\_start();

/\* Check if the user is already logged in \*/  
if(isset( $\_SESSION['user\_id'] ))  
{  
    $message = 'User is already logged in';  
}  
/\* Check that username and password are populated \*/  
if(!isset( $\_POST['username'], $\_POST['pwd']))  
{  
    $message = 'Please enter a valid username and password';  
}  
/\* Check username length \*/  
elseif (strlen( $\_POST['username']) > 20 || strlen($\_POST['username']) < 4)  
{  
    $message = 'Incorrect Length for Username';  
}  
/\* Check password length \*/  
elseif (strlen( $\_POST['pwd']) > 20 || strlen($\_POST['pwd']) < 4)  
{  
    $message = 'Incorrect Length for Password';  
}  
/\* Check username for alpha numeric characters \*/  
elseif (ctype\_alnum($\_POST['username']) != true)  
{  
    $message = "Username must be alpha numeric";  
}  
/\* Check password for alpha numeric characters \*/  
elseif (ctype\_alnum($\_POST['pwd']) != true)  
{  
        $message = "Password must be alpha numeric";  
}  
else  
{  
    /\* Store username and pwds as variables\*/  
    $username = filter\_var($\_POST['username'], FILTER\_SANITIZE\_STRING);  
    $pwd = filter\_var($\_POST['pwd'], FILTER\_SANITIZE\_STRING);

    /\* Encrypt password with sha1\*/  
    $pwd = sha1( $pwd );  
      
    try  
    {  
         /\*Connect to CRUD Database  mysqli(Server,User,Password,Database)\*/  
        $link = connectDB();

        /\* Prep SQL statement which will compare the user credentials with what is stored in the database\*/  
        $sql = "SELECT User\_ID FROM User\_Dfn WHERE username = '".$username."' AND pwd = '".$pwd."'";  
        /\*echo $sql."<br>";\*/  
          
        /\*Run the query\*/  
        if($result=mysqli\_query($link,$sql))   
        {  
          /\*assign the User\_id from the database to the session user\_id\*/  
          while($row = mysqli\_fetch\_assoc($result)) {  
            $user\_id = $row['User\_ID'];  
            /\*echo "<br>user\_id=".$user\_id;\*/

            /\* Set the session user\_id parameter \*/  
            $\_SESSION['user\_id'] = $user\_id;  
            $\_SESSION['timeout'] = time();  
            /\*header("Location: ToDoApp.php"); UNCOMMENT this once tested there will be ins below\*/  
            $message = 'You are now logged in';  
          }          
        }  
          if($user\_id == false)  
          {  
            $message = 'Login Failed';  
          }  
    }      
    catch(Exception $e)  
    {  
        $message = 'Unable to process request';  
    }  
}  
?>

<html>  
<head>  
<title>LoginSubmit</title>  
</head>  
<body>  
<p><?php echo $message; ?>  
</body>  
</html>

Test this by attempting to log in intentionally fail the validations.  
Once tested, uncomment the header location row that forwards users to ToDoApp.php once they are logged in. Then you should be able to test logging in with correct credentials

Now lets create a logout page Logout.php  
<?php  
session\_start();

// Unset all of the session variables.    
session\_unset();

// Destroy the session.  
session\_destroy();  
?>  
<html>  
  <head>  
    <title>Logged Out</title>  
  </head>  
  <body>  
    <h1>You are now logged out.</h1>  
  </body>  
</html>

Test it out and ensure you have closed the session by logging out!

Enforce log in - Text lecture

Section 8, Lecture 117

Create a file titled TestLoginStatus.php

<?php        
require\_once('SQLFunctions.php');  
session\_start();

if(!isset($\_SESSION['user\_id']))  
{  
    $message = 'You must be logged in to access this page';  
}  
else  
{  
    /\*copy the session user\_id to a local variable\*/  
    $user\_id = $\_SESSION['user\_id'];  
    /\*echo "<br>user\_id=".$user\_id;\*/  
      
    try  
    {  
         /\*Connect to CRUD Database\*/  
        $link = connectDB();

        /\* Prep SQL statement to find the user name based on the user\_id \*/  
        $sql = "SELECT username FROM User\_Dfn WHERE User\_ID = ".$user\_id;  
        /\*echo "<br>".$sql."<br>";\*/  
          
        /\*execute the sql statement\*/  
        if($result=mysqli\_query($link,$sql))   
        {  
          /\*from the sql results, assign the username that returned to the $username variable\*/      
          while($row = mysqli\_fetch\_assoc($result)) {  
            $username = $row['username'];  
            /\*echo "<br>username=".$username;\*/  
          }          
        }

        /\* Return Status to User\*/  
        if($username == false)  
        {  
            $message = 'Access Error';  
        }  
        else  
        {  
            $message = 'Welcome '.$username;  
        }  
    }  
    /\*if something goes wrong, return the following error\*/  
    catch (Exception $e)  
    {  
        $message = 'Unable to process request.';  
    }  
}

?>

<html>  
<head>  
<title>Test Login Status</title>  
</head>  
<body>  
<h2><?php echo $message; ?></h2>  
</body>  
</html>

Now test out this page -> If you aren’t logged in, you will see a message that you must be logged in to access this page

If you are logged in, you will see a welcome message with your username

Now that we can log in and out of the site using session, we need to enforce this which we will start in the next video

Sessions - Text lecture

Section 8, Lecture 119

First, let's create a new page titled session.php

This page will load at the beginning of the other pages. It will redirect users to the login page if they haven’t yet authenticated.  
<?php    
session\_start();

if(!isset($\_SESSION['user\_id']))  
{  
    /\* Redirect If Not Logged In \*/  
    header("Location: Login.php");  
    exit; /\* prevent other code from being executed\*/  
} else {  
  /\*we are going to start tracking a new session variable we will call timeout.  
   by comparing the session timeout plus 600 seconds to the current time,   
   we can force users to the logout page when they attempt to access the page, after 10 mins of inaction\*/  
  if ($\_SESSION['timeout'] + 10 \* 60 < time()) {  
    /\* session timed out \*/  
    header("Location: Logout.php");  
  } else {  
    /\*if the user isn't timed out, update the session timeout variable to the current time.\*/  
     $\_SESSION['timeout'] = time();  
  }

Challenge section intro - text reference

Section 9, Lecture 121

https://docs.google.com/document/d/1QwOaVWsnJzY62G1iwbVe43UNHa8zH3xqKG32DBkkvXg/edit?usp=sharing

The whole point of this challenge section is to build the app yourself and share your experience/code

Please don't forget to post a paragraph of what you have learnt, found difficult/easy, fulfilling - anything at all in the Q & A for the day's video

Good luck!

Deploy to production - Text lecture

Section 10, Lecture 126

You can find instructions to deploy your apps to production using eco web hosting below:

https://docs.google.com/document/d/1ilAnWOcsqNKv8RYLTAWYXGulbp4zPgZT\_xNCsrUJXjI/edit?usp=sharing

Database permissions section doc

Section 11, Lecture 127

You can access and view the document here:

https://docs.google.com/document/d/1h02cEaO4leWgsB\_T4iGLZAWxw7j4f1nLJdg8TfXvv2Q/edit?usp=sharing

I would recommend saving to PDF so you can access offline as well. Good luck and enjoy!