UNIVERSITÉ
Concordia
UNIVERSITY

CEWP 459

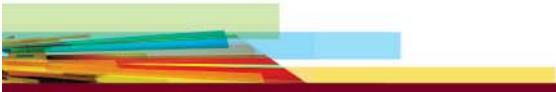PHP Programming with MySQL – Level I
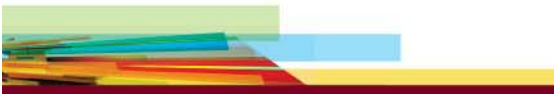Regular Expressions

SQL Review

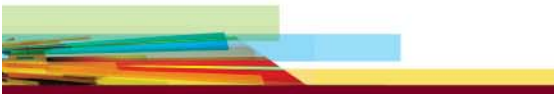# Tutorial on basic SQL commands

MySQL Workbench

DEMO CONNECTION and Setup

Sakila database

1. Install structure.
2. load data.
3. Complete tutorial

Option 2 (Just SQL, no PHP)

- http://sqlfiddle.com

SQL Fiddle Setup

- Create a table with the following attributes

Student

ID: integer, auto-increment, not null

FirstName: varchar(45) not null

LastName: varchar(45) not null

Age: integer not null
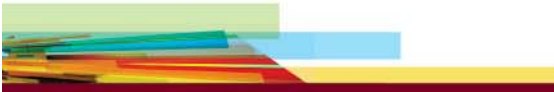
Data

John Doe, 45

Mary Smith, 44

Glen Quagmire, 49

Roger Waters, 33

David Gilmore, 41
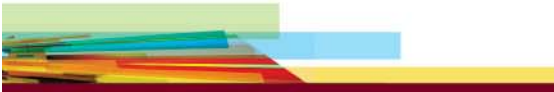
Query revisit

Retrieve:

1. The youngest student.
2. The average student age.
3. Names ending with "re".
4. The number of records.
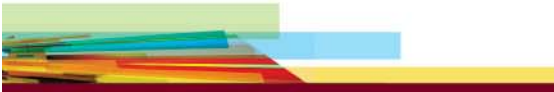
Primary Key

- Always unique
- In every table you always will have a key
- Singular or Compound*
- Naming: Generic key – ID, or use the proper name.
  Eg: Social Insurance Number = SIN
- Compound – Made up of 2 columns to make the
  value unique

Foreign Key

- When a PRIMARY key is placed in another table, it is called a "Foreign Key".
- Eg:
- **STUDENT_TYPE (ID, Type).**
- **STUDENT (ID, Name, Student_Type_ID).**
    - Student Type (F, 'Full Time')
    - Student Type (P, 'Part Time')
    - Student (1, Brendan, 'P')
    - Student (2, Mary, 'F')

Concordia
UNIVERSITÉ
UNIVERSITY

Example of a Junction Table (Many to many)

Artists
- ArtistID
- ArtistName
- ArtistBiography
- ArtistPicture

Recordings
- AlbumID
- ArtistID
- GenreID
- YearOfRelease
- NumberOfTracks
- Price
- AlbumCover

Genre
- GenreID
- Genre
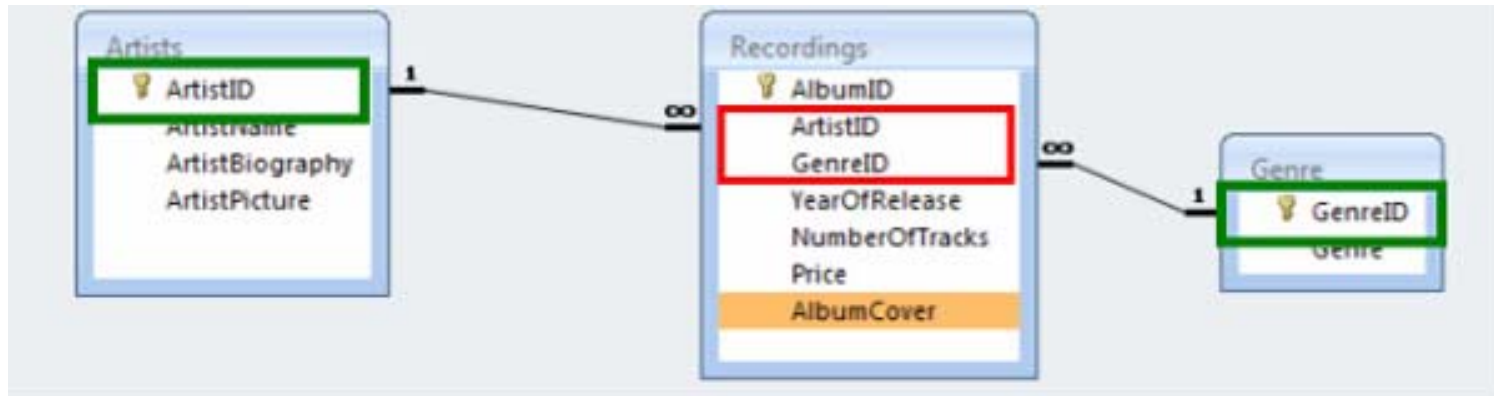
☐ = primary key

☐ = foreign key
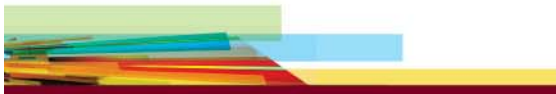
Setup Tables (Database = CorpDB)

- Table: Department
  - DepartmentID
  - DepartmentName

- Table: Employee
  - EmployeeID
  - LastName
  - DepartmentID

Concordia
UNIVERSITÉ
UNIVERSITY
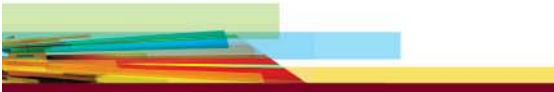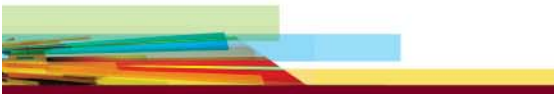
Employee

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL
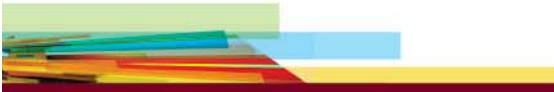
Department

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Inner Join Notation 1

SELECT *

FROM employee INNER JOIN department

  ON employee.DepartmentID =
department.DepartmentID;
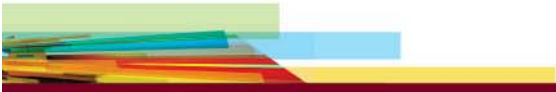
Inner Join Notation 2

SELECT *

FROM employee, department

WHERE employee.DepartmentID = department.DepartmentID;
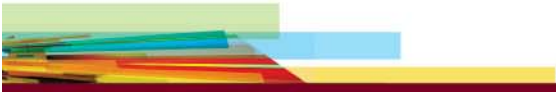
Natural Join

- This is a type of join that joins tables based on column names, automatically.

- In the case of employee and department, both have DepartmentID fields, so this will be the join field.

LEFT outer join

SELECT *

FROM employee LEFT OUTER JOIN department

ON employee.DepartmentID =
department.DepartmentID;

Right Outer Join

SELECT *

FROM employee RIGHT OUTER JOIN department

 ON employee.DepartmentID =
department.DepartmentID;
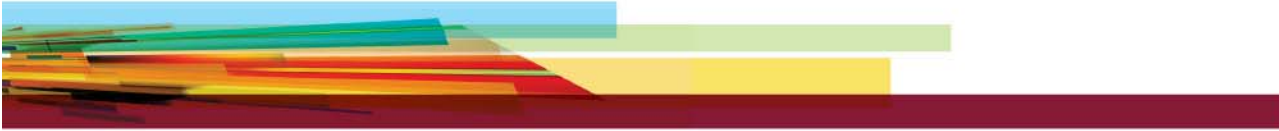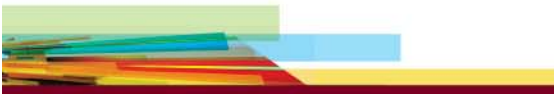
MySQLi

Connect

- mysqli_connect command is used to connect to the DB.
- Format:
  mysqli_connect(*host,username,password,dbname*);

Parameter	Description
host	Either a host name or an IP address -> localhost:port
username	The MySQL user name -> root
password	The password to log in with -> mysql
dbname	The default database to be used when performing queries -> Database name you created for this assignment. (or your course DB).

Connect Example

```php
<?php
// Create connection
$con=mysqli_connect("localhost",
"cewp459s00","abc123","cewp559s00");

// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }
?>
```
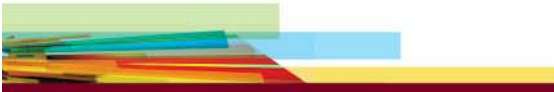
Error reporting

- Usually it's a good idea to enable PHP for detecting ALL SQL errors. By default it doesn't.
- mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
- This throws a mysqli exception when an error is detected.

Close Connection

- Please note, connection is automatically closed at the end of your script. This is to close a connection EARLY.

```php
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");

// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " .
mysqli_connect_error();
  }

mysqli_close($con);
?>
```
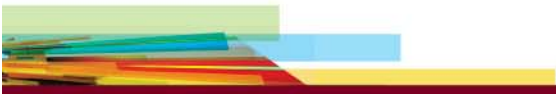
Query + Error Checking

```
// Perform a query, check for error
if (!mysqli_query($con,"INSERT INTO
Persons (FirstName) VALUES ('Glenn')"))
  {
  echo("Error description: " .
mysqli_error($con));
  }
```

MySQL Base commands

Function	Description
mysqli_query()	Performs a query against the database
mysqli_affected_rows()	Returns the number of affected rows in the previous MySQL operation
mysqli_connect_errno()	Returns the error code from the last connection error
mysqli_connect_error()	Returns the error description from the last connection error
mysqli_fetch_array()	Fetches a result row as an associative, a numeric array, or both
mysqli_fetch_all()	Fetches all result rows as an associative array, a numeric array, or both

MySQL Base commands

Function	Description
mysqli_info()	Returns information about the most recently executed query
mysqli_more_results()	Checks if there are more results from a multi query
mysqli_multi_query()	Performs one or more queries on the database
mysqli_sqlstate()	Returns the SQLSTATE error code for the last MySQL operation
mysqli_commit()	Commits the current transaction
mysqli_rollback()	Rolls back the current transaction for the database

Concordia
UNIVERSITÉ
UNIVERSITY

Connection Steps

- Define connection
- Connect
- Check the connection
- Build query
- Execute query
- Check if successful, unsuccessful
- Report result

General SQL Statements (no result)

- mysqli_query(connection, sql)

Create Table
Don't forget error checking level.

```php
$con=mysqli_connect("xxx","xxx","xxx","xxx");
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);

// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

// Create table
$sql="CREATE TABLE Persons(FirstName CHAR(30),LastName CHAR(30),Age INT)";

// Execute query
if (mysqli_query($con,$sql))
        echo "Table persons created successfully"
else
        echo "Error creating table: " . mysqli_error($con);
```

INSERT INTO

```php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

mysqli_query($con,"INSERT INTO Persons (SIN, FirstName, LastName, Age)
VALUES (12345, 'Peter', 'Griffin',35)");

mysqli_query($con,"INSERT INTO Persons (SIN, FirstName, LastName, Age)
VALUES (23456, 'Glenn', 'Quagmire',33)");

mysqli_close($con);
```
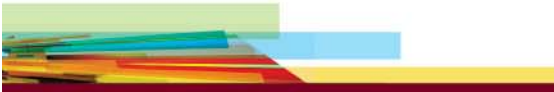
Exercise – Insert (15 mins)

- Insert <u>3</u> rows into your student table using PHP.
- Specify column names and leave out the key column (since it's auto-filled … auto-increment).
- **INSERT INTO Persons (FirstName, LastName, Age)**
  VALUES ('Peter', 'Griffin',35)");

PID INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY(PID),
FirstName CHAR(15),
LastName CHAR(15),
Age INT

Mysqli SELECT row by row

- Define the query with **mysqli_query** and place the result into a pointer variable.

- Use **mysqli_fetch_array(pointer)**, until there are no more rows and place the result into a variable.

- How to test if more rows? The result of **mysqli_fetch_array(pointer)** will be positive.

SELECT

1. Open your DB, connect, evaluate, etc…
2. Perform command (Select)
3. Step through results
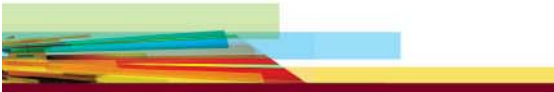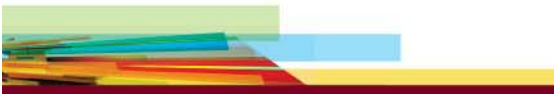4. Close db.

-
```
$result = mysqli_query($con,"SELECT * FROM
Persons");

while($row = mysqli_fetch_array($result))
{
  echo $row['FirstName'] . " " .
$row['LastName'];
  echo "<br>";
}
```

Select – Notes

- We can use WHERE in the select, and even JOIN's. The format is up to you. Also ORDER BY counts.
- You can also build your SELECT dynamically .. Example, sometimes we want ORDER BY DESCENDING, sometimes ORDER BY ASCENDING.
- If the SELECT is COMPLEX, test your select statement in HeidiSQL or in TOAD before putting it into your PHP

- **Important**: In the real world, best practice we would put our SELECT statements in "include" files.
- **Also Important**: Another good practice is to read you entries from the db into OBJECTS from a class.

Exercise – SELECT (10 mins)

- Select all rows from the table you have created.
- Use the mysqli_fetch_array statement inside a WHILE loop.
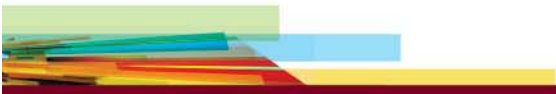
Ex – Display results in HTML table (15 mins)

- Use the same logic as EX3.
- But factor in the HTML aspect.

```
echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";
while($row = mysqli_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";
```

Ex – Select statement built dynamically.

Perform 2 selects
#1, ORDER BY LastName Ascending
#2, ORDER BY LastName Descending

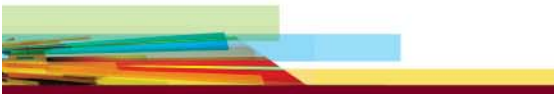Make sure your select statement is common for both.
Order by is CONCATINATED
Eg:

```
$a="Select * from TABLE{space}";
$b="order by X descending;";
$result = mysqli_query($con, $a . $b);
```

MySQLi fetch all

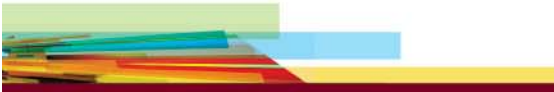- $con=mysqli_connect("localhost","my_user","my_password","my_db");
  // Check connection
  if (mysqli_connect_errno())
    {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
    }

  $sql="SELECT Lastname,Age FROM Persons ORDER BY Lastname";
  $result=mysqli_query($con,$sql);

  // Fetch all
  mysqli_fetch_all($result,***MYSQLI_ASSOC***);
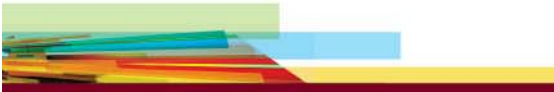
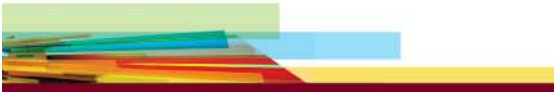 // Free result set
  mysqli_free_result($result);

  mysqli_close($con);

- MYSQLI_ASSOC
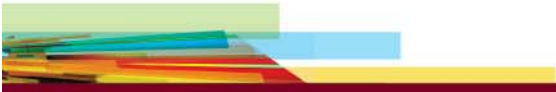- MYSQLI_NUM
- MYSQLI_BOTH

Update

```
$newage = 41;


mysqli_query($con,"UPDATE Persons
SET Age=" . $newage . " WHERE
FirstName='Peter' AND LastName='Griffin'");
```
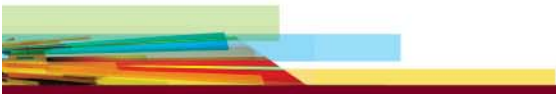
DELETE

- `mysqli_query($con,"DELETE FROM Persons WHERE LastName='Griffin'");`

Exercise – Update + Delete

- Update and Delete a row you have created in your table.
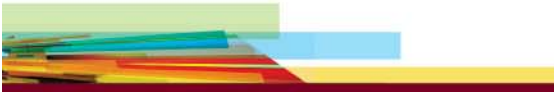
- What you update and what you delete are up to you.

ODBC

- ODBC allows you to connect to a data source (e.g. an MSSQL database).
- Use odbc_connect() to connect to the DB.
- Use odbc_exec() to run an SQL statement.
- Use odbc_fetch_row() to fetch a record from the DB.
- Use odbc_result() to use a field from the record.
- odbc_close() will close the connection.

ODBC Example

```php
$conn=odbc_connect('northwind','','');
if (!$conn)
  {exit("Connection Failed: " . $conn);}
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
if (!$rs)
  {exit("Error in SQL");}
echo "<table><tr>";
echo "<th>Companyname</th>";
echo "<th>Contactname</th></tr>";
while (odbc_fetch_row($rs))
  {
  $compname=odbc_result($rs,"CompanyName");
  $conname=odbc_result($rs,"ContactName");
  echo "<tr><td>$compname</td>";
  echo "<td>$conname</td></tr>";
  }
odbc_close($conn);
echo "</table>";
```
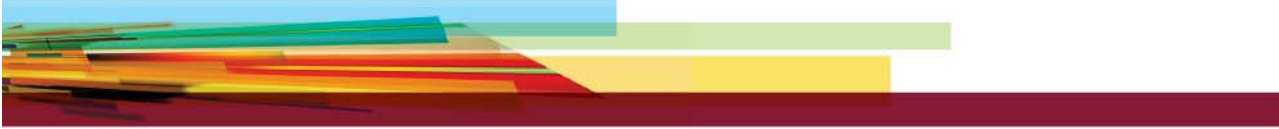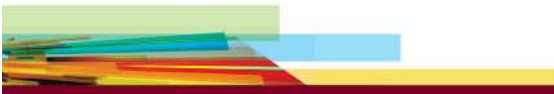
PDO

Intro to PDO

```
$db = new
PDO('mysql:host=localhost;dbname=testdb;c
harset=utf8', 'username', 'password');
```

- Error handling; different from standard Mysqli format.

```
try
{
    $db->query('hi'); //invalid query!
}
catch(PDOException $ex)
{
    echo "An Error occured!";
    echo $ex->getMessage());
}
```
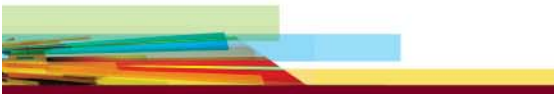
Exercise Setup

- Create a table or two for testing.
- Suggestions
- Employee Table (ID, Name, Role)
- SKU Table (SKU, Name, Price)

- Add about 3 rows of data for each

Fetching rows from a table

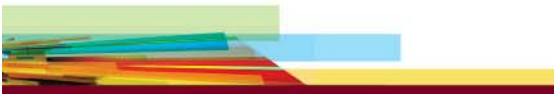- To fetch rows using basic PDO, we can do this.

```php
foreach($db->query('SELECT * FROM table')
as $row) {
    echo $row['field1'].'
'.$row['field2']; //etc...
}
```

Concordia
UNIVERSITÉ
UNIVERSITY

Rewrite previous slide's code

- Separate the query and the for-each, to see more accurately how each component works.
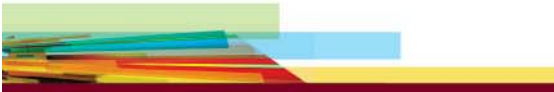
Fetching rows from a table.

- This is more object oriented.

```
$stmt = $db->query('SELECT * FROM table');

while($row = $stmt->fetch(PDO::FETCH_ASSOC))
{
        echo $row['field1'].' '.$row['field2'];
}
```

- **PDO::FETCH_ASSOC:** returns an array indexed by column name
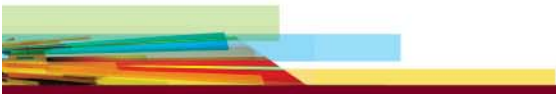- **PDO::FETCH_BOTH** (default): returns an array indexed by both column name and number
- **PDO::FETCH_BOUND**: Assigns the values of your columns to the variables set with the ->bindColumn() method
- **PDO::FETCH_CLASS**: Assigns the values of your columns to properties of the named class. It will create the properties if matching properties do not exist
- **PDO::FETCH_INTO**: Updates an existing instance of the named class
- **PDO::FETCH_LAZY**: Combines PDO::FETCH_BOTH/PDO::FETCH_OBJ, creating the object variable names as they are used
- **PDO::FETCH_NUM**: returns an array indexed by column number
- **PDO::FETCH_OBJ**: returns an anonymous object with property names that correspond to the column names
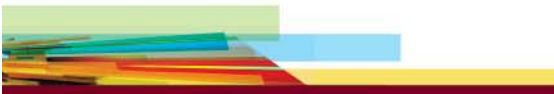
FetchAll

- Fetch-all into a $results array.

```php
$db = new
PDO('mysql:host=localhost;dbname=cewp559g01;c
harset=utf8', 'root', 'xxxxxx');
$stmt = $db->prepare('SELECT * FROM test1');
$stmt->execute();
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);
print_r($results);
```
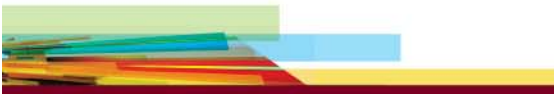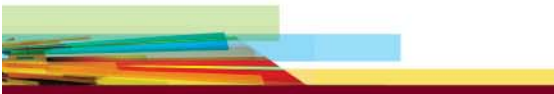
Row count for fetched rows

- This will put the value of the count of rows into $row_count.

```
$stmt = $db->query('SELECT * FROM
table');
$row_count = $stmt->rowCount();
echo $row_count.' rows selected';
```
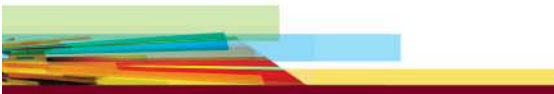
Concordia
UNIVERSITÉ
UNIVERSITY

Last record ID

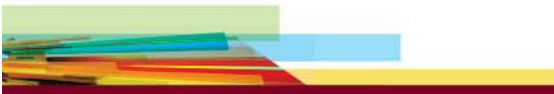- To get the ID of the last record inserted (very useful).

```
$result = $db->exec("INSERT INTO
table(firstname, lastname) VALUES('John',
'Doe')");
$insertId = $db->lastInsertId();
```

Updates / Inserts

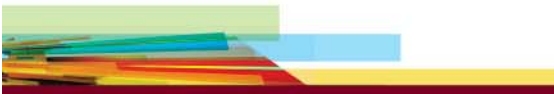- Updates and inserts are done as follows:

```
$affected_rows = $db->exec("UPDATE table
SET field='value'");
echo $affected_rows.' were affected'
```
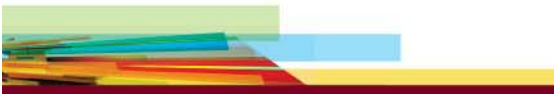
Parameters

- We can pass variable values to the statements;

```
$stmt = $db->prepare("SELECT * FROM table
WHERE id=? AND name=?");
$stmt->execute(array($id, $name));
$rows = $stmt-
>fetchAll(PDO::FETCH_ASSOC);
```
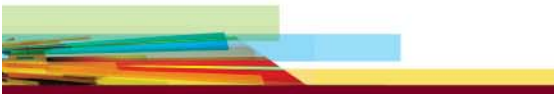
Named Placeholders

```php
$stmt = $db->prepare("SELECT * FROM table
WHERE id=:id AND name=:name");

$stmt->bindValue(':id',      ,
PDO::PARAM_INT);

$stmt->bindValue(':name', $name,
PDO::PARAM_STR);

$stmt->execute();

$rows = $stmt-
>fetchAll(PDO::FETCH_ASSOC);
```

Named Placeholders 2

```php
$stmt = $db->prepare("SELECT * FROM table
WHERE id=:id AND name=:name");

$stmt->execute(array(':name' => $name,
':id' => $id));

$rows = $stmt-
>fetchAll(PDO::FETCH_ASSOC);
```
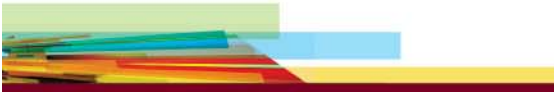
Insert Statement (Prepared)

```
$stmt = $db->prepare("INSERT INTO
table(field1,field2,field3,field4,field5)
VALUES(:field1,:field2,:field3,:field4,:f
ield5)");

$stmt->execute(array(':field1' =>
$field1, ':field2' => $field2, ':field3'
=> $field3, ':field4' => $field4,
':field5' => $field5));

$affected_rows = $stmt->rowCount();
```
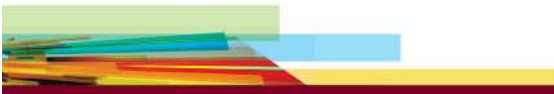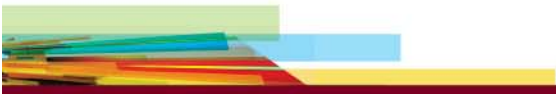
Delete from Table

```php
$stmt = $db->prepare("DELETE FROM table WHERE id=:id");
$stmt->bindValue(':id', $id, PDO::PARAM_STR);
$stmt->execute();
$affected_rows = $stmt->rowCount();
```

Update

```
$stmt = $db->prepare("UPDATE table SET
name=? WHERE id=?");

$stmt->execute(array($name, $id));

$affected_rows = $stmt->rowCount();
```
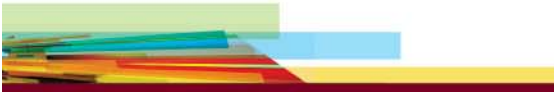
Multiple Execution Example

```
# Prepare the query ONCE
$stmt = $conn->prepare('INSERT INTO tablename
VALUES(:name)');
$stmt->bindParam(':name', $name);

# First insert
$name = 'Joe';
$stmt->execute();

# Second insert
$name = 'Larry';
$stmt->execute();
```
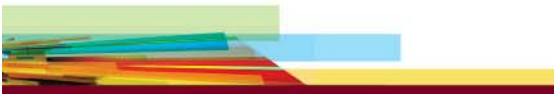
Concordia
UNIVERSITÉ
UNIVERSITY

PDO Exercise 1
15 minutes.

- Use an existing table in your db, or create a new one.
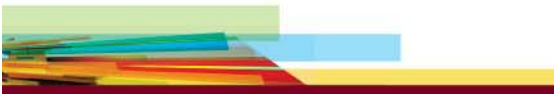- Insert more than one row using the technique above into the table.

PDO with objects (Intro)

- Step 1: Define a class matching a database table.
- Step 2: Set fetch mode to FETCH_CLASS.
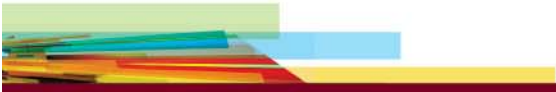- Step 3: Use the result object the same way as a class object.

Review Fetch Modes

- **PDO::FETCH_ASSOC**: Returns an array.
- **PDO::FETCH_BOTH**: Returns an array, indexed by both column-name, and 0-indexed.
- **PDO::FETCH_BOUND**: Returns TRUE and assigns the values of the columns in your result set to the PHP variables to which they were bound.
- **PDO::FETCH_CLASS**: Returns a new instance of the specified class.
- **PDO::FETCH_OBJ**: Returns an anonymous object, with property names that correspond to the columns.

Class Example

```php
class User {
  public $first_name;
  public $last_name;

  public function full_name()
  {
      return $this->first_name . ' ' .
          $this->last_name;
  }
}
```
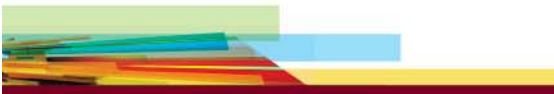
Fetch Example

```php
try {
  $pdo = new PDO('mysql:host=localhost;dbname=x', $uname, $passwd);
  $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $result = $pdo->query('SELECT * FROM someTable');

  # Map results to object
  $result->setFetchMode(PDO::FETCH_CLASS, 'User');

  while($user = $result->fetch()) {
    # Call our custom full_name method
    echo $user->full_name();
  }
} catch(PDOException $e) {
  echo 'Error: ' . $e->getMessage();
}
```
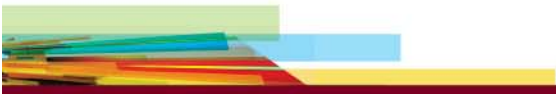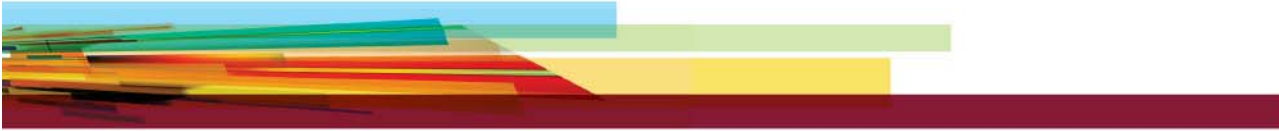
PDO Class Ex. 1
30 Minutes

- Using the table you created in the multiple-insert example:

- Write a class-based fetch to read the data into objects.
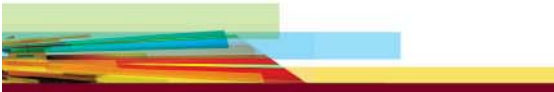

- Display the object contents to the screen.

SQL Injection

SQL Injection

- SQL injection is the most common software exploit
- It affects not just PHP but other languages also.
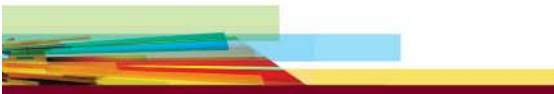- Affects dynamically constructed queries.

Example

- txtUserId = getRequestString("UserId");
- txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;

- Most common example:
  - Add 1=1 to the end of the statement.

    Eg:

    txtUserId = " 5 or 1=1";
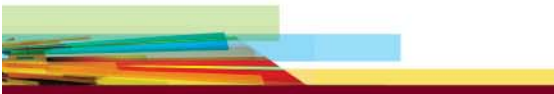
Exercise 1a

- Create a database table in your MySQL account

- Name: Users

- Attributes: UserId (integer), Name, Password

- Fill in a few rows of test data. For the password you can use a plain password for this example, or you can make a program to encrypt the values if you have more time.

Exercise 1b

- Create a form that accepts an ID, and then displays the record(s) on the next form in a table.
- It should build the query like the previous example.

- Step 1 : on the resulting page, JUST display the query and observe it.
- Step 2 : Make the real form that displays the data, and try to put the SQL injection hack into it.

More advanced: You can create a web service environment (refer to the BASIC web service I showed in previous classes).
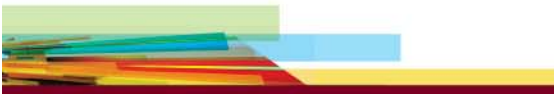
SQL Injection Example 2

- By inserting : " or ""="
- We do almost the same thing as previously.

uName = getRequestString("UserName");
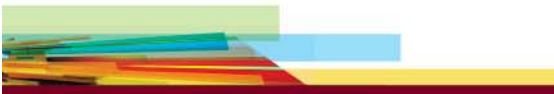uPass = getRequestString("UserPass");

sql = "SELECT * FROM Users WHERE Name ='" +
uName + "' AND Pass ='" + uPass + ""'

Exercise 2

- Create a form that accepts a username and password just like the previous example, and then displays the record(s) on the next form in a table.

- Step 1 : on the resulting page, JUST display the query and observe it.
- Step 2 : Make the real form that displays the data, and try to put the SQL injection hack into it.

Batched SQL Injection

SIMILAR TO ORIGINAL EXAMPLE

- txtUserId = getRequestString("UserId");
- txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;

- Example:
  - Add an extra statement to the end of the line.

  Eg:

  txtUserId = " 5; delete from tablename;";
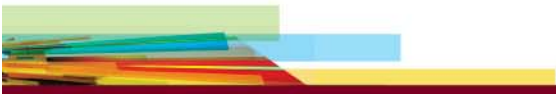
How to solve:

- Very easy: Use parameters like we learned in the PDO class:

:parameter1 :parameter2

This way the parameters are parsed independently and you cannot "construct" an SQL command.

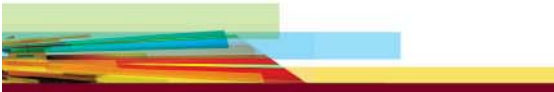Remember to set your DB connection to a "Real" parameterized mode:

$dbConnection->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);

$dbConnection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
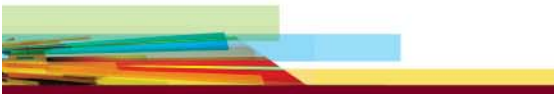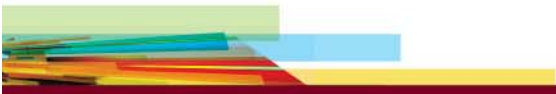
Example of a basic DB class

- DB.class
  - Methods
    - InsertBook(array)
    - UpdateBook(array)
    - DeleteBook(array)
    - InsertAuthor(array)
    - UpdateAuthor(array)
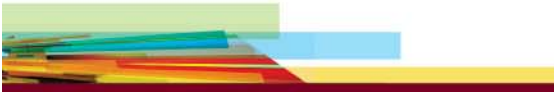   - DeleteAuthor(array)

DB Class More in depth

- Global Values
  - DB connection

- Private Methods
  - Connect
  - Disconnect

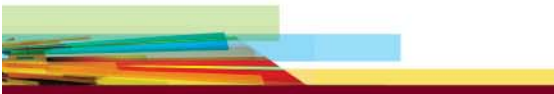# Create Structure for an application for testing your db.
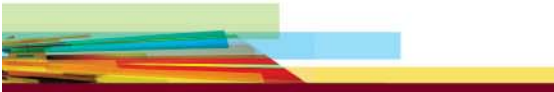
1. Main Menu
   - Manage books

Authors

1. Authors
   - List authors
   - For each book, include a delete link and an update link
   - Include an "Add" link at the bottom with appropriate fields to add a record.

Books

1. Books
   - List books
   - For each book, include details, a delete link and an update link
   - Include an "Add" link at the bottom
   - For the Add link, create appropriate fields to add a record.