 **CS 341: Web Technologies**

LAB EXERCISE 9a (Wednesday, 7th November 2018)

**Instructions:** This lab aims to help you to revise your SQL from last year. It also highlights some important points about how to connect to MySQL. **DO THIS LAB INDIVIDUALLY.**

**TASK:** Review SQL, create a database, and query the database using PHP.

**Activity A: Review SQL (Selecting Data in Database)**

SQL is a standard language for accessing and manipulating databases. The **select** statement is used to query the *database* and retrieve selected data that *match* the *criteria* that you specify. The *column names* that follow the *select* keyword determine which columns will be returned in the results. You can select as many column names that you would like, or you can use a “**\***” to select *all columns*.

The *table name* that follows the keyword **from** specifies the table that will be queried to retrieve the desired results.

The **where** clause (optional) specifies which data values or rows will be returned or displayed, based on the criteria described after the keyword **where**. *Conditional selections* used in the **where** clause:

|  |  |
| --- | --- |
| = | Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| <> | Not equal to |
| LIKE | \*See note below |
|  |  |

The **LIKE** *pattern matching operator* can also be used in the conditional selection of the *where clause*. Like is a very powerful operator that allows you to select only rows that are “**like**” what you specify. The percent sign “**%**” can be used as a **wild card** to match any possible character that might appear before or after the characters specified. Can you identify any other wild card? Take note of how they are used. For example:

select first, last, city

from empinfo

where first LIKE ‘Er%’;

This SQL statement will match any *first names* that start with ‘Er’. **Strings must be in single quotes.** Or you can specify,

select first, last

from empinfo

where last LIKE ‘%s’;

This statement will match any *last names* that end in a ‘s’.

select \* from empinfo

where first = ‘Eric’;

This will only select rows where the *first name* equals ‘Eric’ exactly.

Combine these wildcards in complex patterns to perform more advanced queries. For example, suppose you need to construct a list of all of your employees who have names that begin with a letter from the first half of the alphabet but do not end with a vowel. You could use the following query:

select \*  
 from empinfo  
 where last\_name LIKE '[a-m]%[^aeiou]'

**Activity B: Create a Database with MySQL/MariaDB**

Create a database (**AshesiEmployee**) or table (refer to table below) for the employee information for Ashesi University:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Database/Table: empinfo** | | | | | |
| **first** | **last** | **id** | **age** | **city** | **region** |
| John | Clottey | 99980 | 45 | Accra | Greater Accra |
| Mary | Azuri | 99982 | 25 | Bolgatanga | Upper East |
| Eric | Asante | 88232 | 32 | Kumasi | Ashanti |
| Mary Ann | Edwards | 88233 | 32 | Tamale | Northern |
| Ginger | Agbesei | 98002 | 42 | Ho | Volta |
| Sebastian | Smith | 92001 | 23 | Bolgatanga | Upper East |
| Gus | Osei | 22322 | 35 | Obuasi | Ashanti |
| Mary Ann | May | 32326 | 52 | Tamale | Northern |
| Erica | Williams | 32327 | 60 | Koforidua | Eastern |
| Leroy | Brownei | 32380 | 22 | Takoradi | Western |
| Elroy | Bart-Plange | 32382 | 42 | Tamale | Northern |

1. Submit queries and screenshots of database and table.
2. Which column do you think will be a good **PRIMARY KEY**?
3. Consider the queries below and provide your expected set of data from the above table. Also, show the screenshots of your results:
4. select first, last, city from empinfo;
5. select last, city, age from empinfo

where age > 30;

1. select first, last, city, region from empinfo

where first LIKE ‘J%’;

1. select \* from empinfo;
2. select first, last, from empinfo

where last LIKE ‘%s’;

1. select first, last, age from empinfo

where last LIKE ‘%illia%’;

1. select \* from empinfo where first = ‘Eric’;
2. Can you create a second table called “**department**” where employee“**id**” is a **FOREIGN KEY.** Create any ‘random’ **5 rows** with **3 columns** (i.e. “deptid”, “id”, “deptname”). Show at least **one query** and **result screenshot** that obtains data from both tables based on the relationship created by the FOREIGN KEY.

**Activity C: Select statement exercises**

Enter select statements to:

1. Display the **first name** and **age** for everyone that is in the table.
2. Display the **first name**, **last name**, and **city** for everyone that is not from **Tamale**.
3. Display **all** columns for everyone that is **over** **40 years old** and from **Northern Region**.
4. Display the **first** and **last names** for everyone whose last name ends in an “**ei**”.
5. Display **all** columns for everyone whose **first name** **equals** “**Mary**”.
6. Display **all** columns for everyone whose **first name** **contains** “**Mary**”.
7. Display the **number** of employees from **Ashanti Region**.

**Activity D: PHP connection to MySQL (MariaDB)**

PHP 5 and later can work with a MySQL database using:

* **MySQLi extension** (the "i" stands for improved)
* **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

* PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases. So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
* Both are object-oriented, but MySQLi also offers a procedural API.
* Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

This class will use PDO to connect with MariaDB. To install PDO, go to ‘*C:\xampp\php*’ and open ‘*php.ini*’. In this file, add the following to the end:

[PDO]

extension=php\_pdo.dll

extension=php\_pdo\_firebird.dll

extension=php\_pdo\_informix.dll

extension=php\_pdo\_mssql.dll

extension=php\_pdo\_mysql.dll

extension=php\_pdo\_oci.dll

extension=php\_pdo\_oci8.dll

extension=php\_pdo\_odbc.dll

extension=php\_pdo\_pgsql.dll

extension=php\_pdo\_sqlite.dll

Alternatively, follow the instructions at <http://php.net/manual/en/pdo.installation.php>

### Example (PDO): Do you understand the code below? If not, note them down and we will discuss them during our lecture on Thursday.

<?php  
 $servername = "localhost";  
 $username = "username";  
 $password = "password";  
 $dbname = "myDBPDO";

$sql = "SELECT \* FROM MyGuests";

// connection  
 try {  
     $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
     // set the PDO error mode to exception  
     $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
  
     // sql to create table  
     $sql = "CREATE TABLE MyGuests (  
     id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,   
    firstname VARCHAR(30) NOT NULL,  
    lastname VARCHAR(30) NOT NULL,  
    email VARCHAR(50),  
    reg\_date TIMESTAMP  
    )";  
  
     // use exec() because no results are returned  
    $conn->exec($sql);  
     echo "Table MyGuests created successfully";  
    }  
 catch(PDOException $e)  
     {

if($sql = "SELECT \* FROM MyGuests"){

echo "Connection was not established. Find reason below:<br><br>";

echo $sql . "<br>" . $e->getMessage();

}  
       
    }  
  
 $conn = null;  
?>