**Homework: MySQL and PHP Fundamentals**

This document defines the homework assignments for ["PHP HYPERLINK "https://softuni.bg/trainings/2163/php-web-development-basics-september2018"Web Development Basic HYPERLINK "https://softuni.bg/trainings/2163/php-web-development-basics-september2018"" Course @ Software University](https://softuni.bg/trainings/2163/php-web-development-basics-september2018).

You can check your solutions here: <https://judge.softuni.bg/Contests/1284/Database-Relations-and-Design-Exercise>.

Get familiar with the **gringotts** database. You will use it in the assignments below.

* **Problem 1. Records’ Count**

Import the database and send the **total count of records** to Mr. Bodrog. Make sure nothing got lost.

**Example:**

|  |
| --- |
| **count** |
| **162** |

* **Problem 2. Longest Magic Wand**

Select the size of the **longest magic wand**. Rename the new column appropriately.

**Example:**

|  |
| --- |
| **longest\_magic\_wand** |
| **31** |

* **Problem 3. Longest Magic Wand per Deposit Groups**

For wizards in each deposit group show the longest magic wand. **Sort result by longest magic** wand for each deposit group **in increasing order**, then by **deposit\_group** alphabetically. Rename the new column appropriately.

**Example:**

|  |  |
| --- | --- |
| **deposit\_group** | **longest\_magic\_wand** |
| Human Pride | 30 |
| … | … |

* **Problem 4.\* Smallest Deposit Group per Magic Wand Size**

Select the deposit group with the **lowest** average wand size.

**Example:**

|  |
| --- |
| **deposit\_group** |
| Troll Chest |

* **Problem 5. Deposits Sum**

Select all deposit groups and its **total deposit sum**. Sort result by **total\_sum** in **increasing order**.

**Example:**

|  |  |
| --- | --- |
| **deposit\_group** | **total\_sum** |
| Blue Phoenix | 819598.73 |
| … | … |

Import **restaurant** database.

* **Problem 6. Appetizers Count**

Write a query to retrieve the count of all **appetizers** (**category id = 2**) with price **higher than 8**. Submit your queries with the **MySQL prepare DB & run queries** strategy.



* **Problem 7. Menu Prices**

Write a query to retrieve information about the prices of each category. The output should consist of:

* **Category\_id**
* **Average Price**
* **Cheapest Product**
* **Most Expensive Product**

See the **examples** for more information. **Round** the results to **2 digits after the decimal point**. Submit your queries with the **MySQL prepare DB & run queries** strategy.

**Example**

|  |  |  |  |
| --- | --- | --- | --- |
| **category\_id** | **Average Price** | **Cheapest Product** | **Most Expensive Product** |
| **1** | **7.49** | **6.79** | **8.79** |
| **2** | **10.93** | **7.99** | **14.89** |
| **3** | **7.90** | **6.90** | **8.89** |
| **4** | **12.79** | **11.69** | **13.69** |
| **5** | **5.37** | **4.90** | **5.60** |

Import **soft\_uni** database.

* **Problem 8. Employee Address**

Write a query that selects:

* **employee\_id**
* **job\_title**
* **address\_id**
* **address\_text**

Return the first 5 rows sorted by **address\_id in ascending order.**

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **employee\_id** | **job\_title** | **address\_id** | **address\_text** |
| 142 | Production Technician | 1 | 108 Lakeside Court |
| 30 | Human Resources Manager | 2 | 1341 Prospect St |
| … | … | … | … |

* **Problem 9. Employee Departments**

Write a query that selects:

* **employee\_id**
* **first\_name**
* **salary**
* **department\_name**

Filter only **employees** with **salary** higher than 15000. Return the first 5 rows sorted by **department\_id** **in descending order.**

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **employee\_id** | **first\_name** | **salary** | **department\_name** |
| 109 | Ken | 125500.00 | Executive |
| 140 | Laura | 60100.00 | Executive |
| … | … | … | … |

* **Problem 10. Employees Without Project**

Write a query that selects:

* **employee\_id**
* **first\_name**

Filter only **employees** without a project. Return the first 3 rows sorted by **employee\_id in descending order.**

**Example:**

|  |  |
| --- | --- |
| **employee\_id** | **first\_name** |
| 293 | George |
| 292 | Martin |
| … | … |

* **Problem 11. \*Employee 24**

Write a query that selects:

* **employee\_id**
* **first\_name**
* **project\_name**

Filter all the **projects** of employees with **id 24**. If the project has started after **2005 inclusively** the return value should be **NULL**. Sort result by **project\_name alphabetically.**

**Example**

|  |  |  |
| --- | --- | --- |
| **employee\_id** | **first\_name** | **project\_name** |
| 24 | David | NULL |
| 24 | David | NULL |
| … | … | … |

* **Problem 12. \*Employee Manager**

Write a query that selects:

* **employee\_id**
* **first\_name**
* **manager\_id**
* **manager\_name**

Filter all **employees** with a manager who has **id equals to 3 or 7**. Return the all rows sorted by **employee first\_name** **in ascending order.**

**Example**

|  |  |  |  |
| --- | --- | --- | --- |
| **employee\_id** | **first\_name** | **manager\_id** | **manager\_name** |
| 122 | Bryan | 7 | JoLynn |
| 158 | Dylan | 3 | Roberto |
| … | … | … | … |

* **Problem 13. \*Employee Summary**

Write a query that selects:

* **employee\_id**
* **employee\_name**
* **manager\_name**
* **department\_name**

Show first 5 **employees** (only for employees who has a manager) with their **managers** and the **departments** which they are in (show the departments of the **employees**). Order by **employee\_id.**

**Example**

|  |  |  |  |
| --- | --- | --- | --- |
| **employee\_id** | **employee\_name** | **manager\_name** | **department\_name** |
| 1 | Guy Gilbert | Jo Brown | Production |
| 2 | Kevin Brown | David Bradley | Marketing |
| … | … | … | … |

**Optional**

Create CRUD Application using mysqli.

* **Problem1. List Posts from MySQL**

**Solution:**

$mysqli = new mysqli('localhost', 'root', '', 'blog');

$mysqli->set\_charset("utf8");

if ($mysqli->connect\_errno) die('Cannot connect to MySQL');

$result = $mysqli->query('SELECT \* FROM posts ORDER BY date');

if (!$result) die('Cannot read `posts` table from MySQL');

echo "<table>\n";

echo "<tr><th>Title</th><th>Content</th><th>Date</th></tr>\n";

while ($row = $result->fetch\_assoc()) {

$title = htmlspecialchars($row['title']);

$body = htmlspecialchars($row['content']);

$date = (new DateTime($row['date']))->format('d.m.Y');

echo "<tr><td>$title</td><td>$body</td><td>$date</td></tr>\n";

}

echo "</table>\n";

* **Problem 2. Create New Post in MySQL**
* **Write PHP script to create a new post in MySQL database**
* **Use a HTML form holding title + content (empty author)**

**<form>**

**<div>Title</div>**

**<input type="text" name="title">**

**<div>Content</div>**

**<textarea name="content"></textarea>**

**<div><input type="submit" value="Post"></div>**

**</form>**

***<!-- TODO: process the form here ->***

**if (isset($\_GET['title'])) {**

**$mysqli = new mysqli('localhost', 'root', '', 'blog');**

**$mysqli->set\_charset("utf8");**

**$stmt = $mysqli->prepare(**

**"INSERT INTO posts(title, content) VALUES (?, ?)");**

**$stmt->bind\_param("ss", $\_GET['title'], $\_GET['content']);**

**$stmt->execute();**

**if ($stmt->affected\_rows == 1)**

**echo "Post created.";**

**else die("Insert post failed.");**

**}**

* **Problem 3.Edit Existing Post from MySQL**
* **Write PHP script to edit existing post from MySQL database**
* **Problem 4. Delete Existing Post from MySQL**
* **Write PHP script to delete existing post from MySQL database**

**$mysqli = new mysqli('localhost', 'root', '', 'blog');**

**$mysqli->set\_charset("utf8");**

**if (isset($\_GET['id'])) {**

**$st = $mysqli->prepare("DELETE FROM posts WHERE id = ?");**

**$st->bind\_param("i", $\_GET['id']);**

**$st->execute();**

**if ($st->affected\_rows == 1) echo "Post deleted.";**

**}**

**$result = $mysqli->query('SELECT id, title FROM posts');**

**while ($row = $result->fetch\_assoc()) {**

**$title = htmlspecialchars($row['title']);**

**$delLink = 'delete-post.php?id=' . $row['id'];**

**echo "<p><a href='$delLink'>Delete post '$title'.</a></p>";**

**}**

**Part I. Related Tables**

In this section you will solve problems by which you will exercise **table relations** (Primary and Foreign Keys). Every problem is a **task for an HR Application**. You will need the **geographic\_database.sql** to work with. Download it from the web page and section for this exercise:

**10. PHP-MySQL-and-PHP-Fundamentals-Exercise-Databases.zip**

You can load **geographic\_database.sql** into the server as shown on the screenshot.

* **Problem 5. Database for HR Application**

Janette works as a HR (Human Resource) in a company called BestWishComeTrue.com with about 5000 employees from people all over the world. She needs to keep track all of the employees. Write a simple application which can save in a database three different things:

* Employee **id**, **First Name**, **Middle Name**, **Last name**, **department**, **position**, **passport ID** (the number on the employee card issued by the state). The employee can work only at one department and have only one position until he/she is moved to another department and position. The combination of names and passport ID is unique. Janette uses the simple **id** (**primary key** and **auto increment**) not Passport ID.
* Employee email addresses. It is usual for employees to have at least two different emails. Each email address has the actual **email**, **type of email (work or personal).**
* Employee phone number. One can have **at least one phone number**: personal phone, work phone number and a second, and third work phone number.

**Example**

|  |  |  |
| --- | --- | --- |
| Employee | Email | Phone |
| John Avenger Ausini, Marketing, Junior Marketing Specialist, Pass.ID 643972016689 | Work: [ausini@marketing.bwct.com](mailto:ausini@marketing.bwct.com)  Personal: [john\_ausini@bestmail.top](mailto:john_ausini@bestmail.top) | Personal: +357123456789,  Work: +44357123456784,  Work: +45357123456783, |
| Pragubta Brahaman Kishivara, Programming, Senior Front-end developer, Pass. ID 43IN2432467545465 | Personal: [keepcalmwhenucode@abv.bg](mailto:keepcalmwhenucode@abv.bg)  Work: [pragubta@programming.bwct.com](mailto:pragubta@programming.bwct.com),  Work: [summer\_vacations@programming.bwct.com](mailto:summer_vacations@programming.bwct.com) | Work: + (410)293-5762 |
| Hoordvig Van Derholdt,  Construction, Construction Worker,  Pass. ID H654364366546 | Work: [hoordvig@construction.bwct.com](mailto:hoordvig@construction.bwct.com)  Personal: [john\_ausini@bestmail.top](mailto:john_ausini@bestmail.top) | Work: + (31)293576245 |

**Todo**

1.Create the structure of the **new tables** in **the geography database** because geographical data will be used further in the application.

2. Create three separate tables with proper names (like **employee, employee\_emails, employee\_phones**) with a primary key called **id** for each table.

3. Let the new tables have columns to relate them to one another. Table **employee\_email** needs an **employee\_id** foreign key. What about **employee\_phones**?

* **Problem 6. HR Application Insert Employee**

Janette needs first to insert the main employee data. Write the HR application as CLI with the following input and output:

|  |  |
| --- | --- |
| **Input** | **Ouptut** |
| John, Avenger, Ausini, Marketing, Junior Marketing Specialist, Pass 643972016689 | New employee John Avenger Ausini saved. |
| Pragubta, Brahaman, Kishivara, Programming, Senior Front-end developer, Pass 43IN2432467545465 | New employee Pragubt, Brahaman Kishivara saved. |
| Hoordvig, Van, Derholdt, Construction, Construction Worker, Pass H654364366546 | New employee Hoordvig Van Derholdt saved. |
| Hoordvig, Van, Derholdt, Construction Worker | Error: Please, check your input syntax. |

**Todo**

1. Encapsulate the application in a class Employee.

|  |
| --- |
| **File Employee.php:**  **<?php class Employee {  *//Todo* }** |

2. Use a **db\_config.php** file which will hold your **database credentials** and initialize a connection using PDO kept in the **$db** variable. Set the error attribute to throw exceptions for MySQL errors. Do not catch exceptions!

|  |
| --- |
| **File db\_config.php:**  **<?php $db\_host = "localhost"; $db\_name = "geography"; $db\_user = "root"; $db\_password = 'pass4yoursql'; $db = new PDO("mysql:dbname=$db\_name;host=$db\_host", $db\_user, $db\_password,  array(PDO::*ATTR\_ERRMODE* => PDO::*ERRMODE\_EXCEPTION*));** |

3.The entry point of your application should be **app.php**:

|  |
| --- |
| **File app.php:**  **<?php include "db\_config.php"; include "Employee.php"; *// Todo*** |

4. Pass **$db** wherever you need it **(like in the Employee class). Use Prepare / execute!**

5. Code the functionality to insert a new employee in the Employee class. Use an appropriate name of the method like **insertEmployee()**.

7. **Insert the data** for the four employees in the **example** by **using your application** to test it.

* **Problem 7. Insert Email**

Now Janette needs to insert the emails of an employee. To do that she enters the three names of the employee. But there might be two employees with same names like Ivan Ivanov Ivanov:

|  |  |
| --- | --- |
| **Input** | **Ouptut** |
| Ivan, Ivanov, Ivanov, **emails**, **personal:** [ivanov@abv.bg](mailto:ivanov@abv.bg), **work:** [ivanov@bwct.com](mailto:ivanov@bwct.com), **work:** [summer\_deals@bwct.com](mailto:summer_deals@bwct.com) | Employees with this name: 123, 678 |
| **678**, Ivan, Ivanov, Ivanov, **emails**, **personal:** [ivanov@abv.bg](mailto:ivanov@abv.bg), **work:** [ivanov@bwct.com](mailto:ivanov@bwct.com), **work:** [summer\_deals@bwct.com](mailto:summer_deals@bwct.com) | Emails of Ivan Ivanov Ivanov saved. |
| Hoordvig, Van, Derholdt, **emails**, +7654656465655 | Error: Please, check your input email syntax. |

**Todo**

1. Write a method to check if the person name is unique in the database. If the name is unique return an empty array else return their **IDs in an array.**

|  |
| --- |
| **// Unique: []**  **// Not unique: [123,456, 789,1234]** |

2.Use the method from point 1 and write a method to insert the emails**.**

If the employee is **not unique** print:

|  |
| --- |
| **Employees with this name: 123,456,789,1234** |

Else print:

|  |
| --- |
| **Emails of Ivan Ivanov Ivanov saved.** |

3. Somewhere in the Employee class you should check the syntax. You may use a **main** or **validate** method to do that.

4.**Insert the emails** for the four employees from the example

* **Problem 8. \*Insert Phones**

To insert phone numbers we use pretty much the same like for the emails. The methodology for the uniqueness of the person should be used again. Try to implement the 3 steps from problem 3 by yourself to have the following input and output:

|  |  |
| --- | --- |
| **Input** | **Ouptut** |
| Ivan, Ivanov, Ivanov, **phones**, **personal:** +7654656465655, **work:** +7654656656565, **work:** +7654656656599 | Employees with this name: 123, 678 |
| Ivan, Ivanov, Ivanov, **phones**, **personal:** +7654656465655, **work:** +7654656656565, **work:** +7654656656599 | Phones of Ivan Ivanov Ivanov saved. |
| Hoordvig, Van, Derholdt, **phones**, work: derholdt@bwct.com | Error: Please, check your input phone number syntax. |

**Todo**

Insert the phone numbers for the four employees from the example! Now in your database you should have complete data for four employees.