**The tutorial has three parts.**

* **Creating the example scripts from the document “Database Forms with PHP - Part 1”**
* **Answering the theory questions in part two (some of these will require additional research on the Internet).**
* **Developing the practical scripts in part three.**

**This tutorial includes practical and theoretical questions. It will require you to answer some questions based on existing knowledge, discussions with the teacher and research on selected web sites. The link below accesses the PDO section of the PHP manual.**

<http://php.net/manual/en/book.pdo.php>

**If the question is a written answer questions insert some space at the end of the question and include your answer in blue text. If the question requires a PHP script to be created, create the script and save it for later submission.**

**Part 1 – Practical Examples (exercises)**

Create each of the example programs in the document “**Database Forms with PHP - Part 1**”

**Part 2 - Theory Questions**

1. **What is PDO?**

PDO is an acronym for PHP Data Object. It is an extension which defines a lightweight, consistent API for PHP accessing database. PDO is more like a data access layer, but you can not perform any database functions by using PDO extension itself. You must use a database-specific PDO database driver to access a database. When using PDO extension, you can use the same function to execute queries and fetch data regardless of what database you are running.

1. **What details need to be provided by your script for PHP to make a connection to a MySQL database using PDO?**

When creating a connection using PDO, the connection is established by creating instance of new PDO class. The new PDO() method accepts 3 parameters: database source, username and password. For example:

$conn = new PDO(“mysql:host=localhost;dbname=test”, “username”, “password”);

1. **Name 4 PDO methods and describe the tasks they perform. (refer to the PHP)**

PDO::query() Executes an SQL statement, returning a result set as a PDOStatement object.

PDO::beginTransaction() Initiates a transaction, indicating the transaction is beginning.

PDO::commit() Commits a transaction, allowing any changes to database in a transaction to take effect.

PDO::rollBack() Rolls back a transaction, without doing any changes to database.

PDO::prepare() Prepares a SQL statement

PDO::exec() Executes a SQL statement and returns the number of affected rows.

1. **How do you make use of global variables inside a PHP function?**

Global variables are those variables that is always accessible in all scopes, in any function throughout the application programme. To use a global variable, you first need to define a variable outside all functions, and then use key word ‘global’ to indicate that variable is a global variable and you intend to use it. For example:

<?php

$value = “Hello World”; // declare a variable outside functions

function test() {

global $value;

echo $value; //output Hello World

}

?>

1. **How would you develop user-defined database functions to ensure they have the maximum potential for re-use in other web applications?**

We can create all relevant functions in one php page and this page is considered as a library. When someone intends to use functions in this library, all he needs to do is using require() or require\_once() method to import this php library page. These two methods enable us to include the content of another PHP page.

**Part 3- Practical Question**

1. **Create a new PHP page (in the PHP folder) that displays all of the Customers of Mavis in a table with the same format as the “View Branch” page. Use the existing database functions and CSS file.**