# Homework: MySQL and PHP Advanced

# Agregate Functions, Transactions, Views,

# SQL Functions and procedures, Triggers

This document defines the homework assignments for ["PHP Web Development Basic" Course @ Software University](https://softuni.bg/trainings/1746/php-web-developmentbasics-september-2017). Please submit ……………………………………………….???

All tasks with SQL Scripts must be sent as text files with .sql extension in your homework archive file. Dump your whole database in an SQL file in the homework.

Download the initial code **11.PHP-MySQL-and-PHP-Advanced-Initialcode.zip.**

# The Car Salesman

John Pavlov has a small business in Eastern Europe for selling old cars. His business keeps growing and he needs to track his sales. Help him out and create a **CLI app**. Wrap your functionality in a **class Carshop**.

Today we will solve a practical problem by creating an app for a **car salesman**. Every task is you will solve is a part of creating the application.

# Problem 1. Create a Database for Car Sales

You need to create a database called "cars" which will contain the following structure:

1. Table for used cars

* Id of car (primary key)
* Make
* Model
* Year of production

1. Table for sales

* Id of sale
* Id of car
* Id of customer
* Datetime of sale
* Amount

1. Table for customers

* Id of customer
* First Name
* Family Name

Export your SQL database structure in a file called **cars.sql.** Use HeidiSQL to do that or another tool.

# Problem 2. Add a Car and a Customer

First thing that John must do is to be able in one line to add both a car and his customer into the database. Because these two go together you need to use a **transaction in your SQL**. The input and output must be like these:

|  |  |
| --- | --- |
| **Input** | **Ouptut** |
| Audi, A4, 2004, Ivan, Ivanov, BGN 7000 | New sale entered 2017-01-24 10:13 |
| BMW, 116, 2010, Ilia, Petrov, BGN 24000 | New sale entered 2017-01-24 10:45 |
| Ford, Focus, 2004, Stoyan, Lazarov, BGN 3800 | New sale entered 2017-01-25 14:34 |

Wrap your code in SetSale() method.

# Hint

The idea is to be able to **rollback** what we have done if there is an error. We can use try / catch PHP functionality to catch the error and the rollback our SQL. Here is a representational code you can use for your transaction. It inserts the data only in the **car table. Write your own code** for the tables **customers** and **sales (or your table names)**

|  |
| --- |
| **protected function setSale($car, $person, $amount) {  try {  *// Insert into car* $this->db->beginTransaction();  $stmt = $this->db->prepare("  INSERT INTO `cars`  (`id`,`make`, `model`, `year`)  VALUES  (?, ?, ?, ?)");  $car\_id = "null";  $stmt->bindParam(1, $car\_id);  $stmt->bindParam(2, $car['make']);  $stmt->bindParam(3, $car['model']);  $stmt->bindParam(4, $car['year']);  $stmt->execute();  *// Insert into customers  // Todo  // Insert into sales  // Todo* $this->db->commit();  } catch (PDOException $e) {  $this->db->rollBack();  print "Error!: " . $e->getMessage() . "<br/>";  } }** |

# Todo

Insert three car sales from the example data into the database by using your code.

# Problem 3. Get All Sales

In every moment of time John wants to be able to see his sales. All he does is to enter Sales in his input and he will get a table of all his sales:

|  |  |
| --- | --- |
| **Input** | **Ouptut** |
| Sales | Audi, A4, 2017-01-24 10:13, 7000  BMW, 116, 2017-01-24 10:45, 24000  Ford, Focus, 2004, 2017-01-25 14:34, BGN 3800  ---------  Total: 34800 |

Use an aggregate function to calculate all sales. Put your code in a method getSales(). See additional information here:

<https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html>

<https://mariadb.com/kb/en/library/aggregate-functions/>

# Hint

Your code should look like this:

|  |
| --- |
| protected function getSales() {  try {  $this->db->beginTransaction();  $stmt = $this->db->prepare("  SELECT *SUM*(`amount`)   FROM `sales` ");  $stmt->execute();  $this->db->commit();  } catch (PDOException $e) {  $db->rollBack();  print "Error!: " . $e->getMessage() . "<br/>";  } } |

# Problem 4. Use MySQL Procedure

Because we want to be able to retrieve the sales in different ways we want to wrap that functionality in an SQL procedure and change it in the future. Create a new MySQL procedure GetSales:

|  |
| --- |
| DELIMITER //  **CREATE** **PROCEDURE** get\_sales(**OUT** amount\_total **FLOAT**)  **BEGIN**  **SELECT** **SUM**(`amount`) **INTO** amount\_total **FROM** `sale`;  **END**;  //  DELIMITER; |

Now go back to your code of the method getSales() and change it to use the new MySQL procedure instead.

# Hint

You can now use the procedure from PHP like this:

|  |
| --- |
| Inside class Carshop:  $result = $this->db->query('CALL get\_sales', PDO::*FETCH\_ASSOC*); foreach ($result as $row) {  // Todo } |

# Problem 5. Use MySQL Function

Create a MySQL function **get\_full\_name** which should return the full name of a person. The function receives two varchar parameters: **first**, **family** and returns the full name of the person. You choose the exact name of the variables.

# Todo

1. First check the official documentation:

<https://dev.mysql.com/doc/refman/5.7/en/create-function.html>

<https://mariadb.com/kb/en/library/create-function/>

2. Check in your database client (HeidiSQL, Datagrip or other) the new function

3. Test the new function with a particular name (like: Garry McIntosh)

4.Change getSales() PHP method to use **get\_full\_name** in the SQL queries

# Problem 6. View a Particular Deal

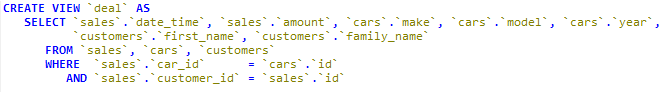
We have the information in three different tables (cars, customers, sales) but we would like to have it in one abstract table which is a combination of all. Create a new MySQL view called 'deal' which shows: Sale date\_time, sale amount, Car model, car year, customer first\_name, customer family\_name.

# Todo

Construct your code as an SQL query.

# Hint

Your query should be something like this:



**SELECT** sales . `date\_sale`, sales . `amount`, cars . `make`, cars . `model`, cars .`year\_of\_production`, customers.`first\_name`, customers .`family\_name`

**FROM** sales, cars, customers

**WHERE** sales.`car\_id` = cars.`car\_id`

**AND** sales.`customer\_id` = customers.`customer\_id`

After you refresh you should see your new view in your database client.

Additional information on views:

<https://dev.mysql.com/doc/refman/5.7/en/create-view.html>

<https://mariadb.com/kb/en/library/create-view/>

# Problem 7. Extend getSales()

In problem 3 we created a method getSales() but it shows only the data about the cars. You have already created a view so use your view called **'deal'** and **rewrite** getSales()to show **all data for all sales.**

See now how the input and output should look like:

|  |  |
| --- | --- |
| **Input** | **Ouptut** |
| Sales | Audi, A4, 2017-01-24 10:13  Sold to Ivan Ivanov for BGN 7000  ---  BMW, 116, 2017-01-24 10:45  Sold to Ilia Petrov for BGN 24000  ---  Ford, Focus, 2004, 2017-01-25 14:34  Sold to Stoyan Lazarov for BGN 3800  ---------  Total: BGN 34800 |

# Problem 8. Create a Trigger for Sales

Create a trigger for the sales table which will hold the total amount of sales. The trigger acts like an accumulator.

Your code should look like:

# Hint

Your query should be something like this:



Now change the code of your application to show the following input and output:

|  |  |
| --- | --- |
| **Input** | **Ouptut** |
| Audi, A4, 2004, Ivan, Ivanov, BGN 7000 | New sale entered 2017-01-24 10:13  ---  Total sales: 7000 |
| BMW, 116, 2010, Ilia, Petrov, BGN 24000 | New sale entered 2017-01-24 10:45  ---  Totals sales: 31000 |
| Ford, Focus, 2004, Stoyan, Lazarov, BGN 3800 | New sale entered 2017-01-25 14:34  ---  Total sales: 34000 |

# Todo

1.Change the code of **setSale()** to use the new trigger you have created. The trigger is used in the following way which you have to **implement in your PHP**:

|  |
| --- |
| SQL: SET @sum = 0;  SQL: INSERT INTO `sales` … VALUES (…)  SQL: SELECT @sum AS 'Total sales' |

2.As you can see you need to add two more queries in your PHP (in blue) to use the trigger.

Additional information: <https://dev.mysql.com/doc/refman/5.5/en/trigger-syntax.html>

# Problem 9. Get Sales for Particular Period\*

Rewrite the method getSales() to be able to deal with a particular start and end period of time and show sales for this period. Check online by doing a search on MySQL data functions.

|  |
| --- |
| **Input** |
| Sales, 2017-01, 2017-04 |

# Problem 10. Rewrite get\_sales MySQL Procedure\*\*

Rewrite the **get\_sales MySQL procedure** to receive two more input parameters: **date\_start** and **date\_end**. Change the method getSales() to use the new code.