**Exercise 1: Getting Started with MySQL**

**Scenario**

In this exercise you will be required recognize the key steps for setting up MySQL on Linux Ubuntu and creating a new database.

**Exercise 1: Getting Started with MySQL**

| **Task list** | |
| --- | --- |
| **Number** | **Instructions** |
| **1** | In the terminal window, perform a system update before proceeding with the MySQL install. Next enter the command to install MySQL. |
| **2** | In the terminal window, start and confirm that the MySQL service is installed and running. |
| **3** | Assuming that the root user has already been created, launch the MySQL shell. |
| **4** | In the MySQL shell, create and show a new sample database named "cars". |
| **5** | In the database create and show the columns for a table named "carInventory". The table should contain the following columns:   * id – auto incremented primary key * carName - name of car * carMake – car make/manufacturer * reg\_date – timestamp for when the data was entered |
| **6** | In the table named "carInventory" insert a new record containing the following:   * VW * Jetta   Then show the data from the table. |
| **7** | Assume you have created a folder to house your MySQL application and want to use Node.js. Use npm to install the Express connector for MySQL. |

**Exercise 1: Getting Started with MySQL**

**Solution**

| **Checkpoints** | |
| --- | --- |
| **Number** | **Details** |
| **1** | In the terminal window, to perform to a system update and then to install MySQL we use the following commands:   * sudo apt-get update * sudo apt-get install mysql-server |
| **2** | In the terminal window, to start and confirm that the MySQL service is installed and running we use the following commands:   * sudo systemctl start mysql * sudo service mysql status |
| **3** | Assuming that the root user has already been created, to launch the MySQL shell we use:   * /usr/bin/mysql -u root –p |
| **4** | In the MySQL shell, to create and show a new sample database named "cars" we use the following commands:   * CREATE DATABASE cars; * SHOW DATABASES; |
| **5** | In the database to create table with the required columns we use the following commands:   * use cars; * CREATE TABLE carInventory(  id INT(4) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  carName VARCHAR(30) NOT NULL,  carMake VARCHAR(30) NOT NULL,  reg\_date TIMESTAMP );   Then to show the columns from the table we use the following commands   * show tables; * SHOW COLUMNS FROM carInventory; |
| **6** | To Insert and show the new record we use the following commands:   * INSERT INTO carInventory(carMake, carModel) VALUES ('VW', 'Golf'); * SELECT \* FROM carInventory; |
| **7** | To install the mysql driver for Node.js, in the terminal window use:   * npm install express mysql |

**Exercise 2: Working with jQuery and AJAX**

**Scenario**

In this exercise you will be required to complete and analyze the following jQuery code snippets to validate a form.

**jqScripts.js**

**…**

$(function() {

$("#GetRecords").click(function(){

***MISSING\_CODE1*** ({

***MISSING\_CODE2***:"http://localhost:8000/getweights",

***MISSING\_CODE3,***

success: function(result){

displayData(result);

},

error:function(result){

console.log(result);

}

})

});

});

**…**

**teamrecords.html**

<!DOCTYPE HTML>

<html>

<head>

<title>SS Weight Tracker</title>

<meta http-equiv="content-type" content="text/html; charset=UTF-8" />

<link rel="stylesheet" type="text/css" href="styles/ssStyles.css" />

<script ***MISSING\_CODE4***>  
 </script>

<script src="scripts/scripts.js"></script>

<script src="scripts/jqscripts.js"></script>

</head>

<body>

<div id="header">

<h1><a href="index.html">SS Weight Tracker</a></h1>

</div>

<div id="navigation">

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="enterdata.html">Enter Data</a></li>

<li><a href="singlerecord.html">My Records</a></li>

<li><a href="teamrecords.html">Team Records</a></li>

<li><a href="#">Winner/Loser</a></li>

</ul>

</div>

<div id="section">

<h2>Team Records</h2>

<div id="id01"></div>

<div><button id="GetRecords">Get Data</button></div>

</div>

</body>

</html>

**Exercise 2: Working with jQuery and AJAX**

| **Task list** | |
| --- | --- |
| **Number** | **Instructions** |
| **1** | In jqScripts.js, complete the code to begin our method to perform an asynchronous request using AJAX |
| **2** | In jqScripts.js, complete the code to specify where to send the request to. |
| **3** | In jqScripts.js, complete the code to set the request type to GET. |
| **4** | In teamrecords.html, complete the code to add the Google jQuery library to the page. |

**Exercise 2: Working with jQuery and AJAX**

**Solution**

| **Checkpoints** | |
| --- | --- |
| **Number** | **Details** |
| **1** | The code to begin our method to perform an asynchronous request using AJAX is  $.ajax |
| **2** | The code to specify where to send the request to is:  url |
| **3** | The code to set the request type to GET is:  type:"GET" |
| **4** | The code to add the jQuery library to the page is:  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"> </script> |

**Exercise 3:** **Working with AngularJS**

**Scenario**

In this exercise you will be required to complete and analyze the following jQuery code snippets to validate a form.

**scripts.js**

**…**

let xmlhttp = new XMLHttpRequest();

let url = "http://localhost:8000/getweights";

let app = angular.module("ssApp", []);

***MISSING\_CODE1*** ('Weights', function($scope, $http) {

$http.get(url).

then(function(response) {

$scope.allWeights = response.data;

});

$scope.frmSubmit = function (){

$http({

method : 'POST',

url : 'http://localhost:8000/putweights',

headers : {'Content-Type': 'application/x-www-form-urlencoded;  
 charset=utf-8'},

data : 'empName='+$scope.empName + '&empWeight='+$scope.empWeight

})

};

});

**…**

**teamrecords.html**

<!DOCTYPE HTML>

<html>

<head>

<title>SS Weight Tracker</title>

<meta http-equiv="content-type" content="text/html; charset=UTF-8" />

<link rel="stylesheet" type="text/css" href="styles/ssStyles.css" />

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">

</script>

***MISSING\_CODE2***   
 </script>

<script src="scripts/scripts.js"></script>

<script src="scripts/jqscripts.js"></script> </head>

<body>

…

</div>

<div id="section" ng-controller="Weights">

<h2>Team Records</h2>

<p ***MISSING\_CODE3*** ="emp in allWeights">

{{emp.empName}} weighed in at {{emp.empWeight}}

</p>

<div id="id01"></div>

<div><button id="GetRecords">Get Data</button></div>

</div>

</body>

</html>

**Exercise 3: Working with AngularJS**

| **Task list** | |
| --- | --- |
| **Number** | **Instructions** |
| **1** | In scripts.js, complete the code to begin our method to create a controller for "Weights". |
| **2** | In teamrecords.html, complete the code to include the Google AngularJS API in the page. |
| **3** | In teamrecords.html, complete the code to include the proper directive to display the data. |

**Exercise 3: Working with AngularJS**

**Solution**

| **Checkpoints** | |
| --- | --- |
| **Number** | **Details** |
| **1** | The code to begin our method to create a controller for "weights" is:  app.controller |
| **2** | The code to include AngularJS in the page is:  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script> |
| **3** | To display the data we need to use the following directive:  ng-repeat  This essentially allows us to iterate over the data in a collection, and then display it in an easily readable format. |