**Week#4: List of Assignments**

**PHP 1**

**Lecture:**

* **Review of Week#4 Assignment solutions**

**MySQL**

* **Introduction to MySQL (Lectures#5)**1. Databases(overview).pptx

2. LectureNotes#5.PPT (Intro. To MySQL & Movie Database - Part 1)

**Required Assignments: “Working with PHP and MySQL”**

WEEK#4: “The Movie database” PART 1

* **wk4\_assign\_ABC\_php1.doc   
   FILES:**
  + **connectvars.php**
  + **createMovieTable.php**
  + **movieForm.html**

3%

* + **movieInsert.php**
  + **displayMovie.php**

**~~~ Note: All assignment solutions should be posted to the college server,   
Tested, and Marked in lab “when working”. ~~~**

QUIZ :

* **Complete the Online Quiz posted in the Week#4 assignments folder.**

READINGS: **READ Chapter 4 of the textbook “Introduction to MySQL”**

**WEEKLY REVIEW QUESTIONS:**

**Complete the questions starting on PAGE 2 of this document**

**WEEKLY REVIEW QUESTIONS (WEEK#4):**

**Answer the following questions. Show your answers to the instructor in-class for making along with your Weekly Assignment solutions.**

**MySQL offers you the opportunity to dive deeply into advanced databases and database applications, while still being free of charge. It is worthwhile and important to understand the SQL language.**

**One of the first SQL commands you need to master is the SELECT statement.**

1. A r\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ database consists of one or more **tables**, where each **table** consists of 0 or more **records**, or **rows**, of data.

*Answer:*

**relational**

The SQL **SELECT** statement is what we use to choose, or select, the data that we want returned from the database to our application. It is the language we use to formulate our question, or query, that we want answered by the database.

A SQL **SELECT** statement can be broken down into numerous elements, each beginning with a keyword. Although it is not necessary, common convention is to write these keywords in all capital letters. In the following questions, you will focus on the most fundamental and common elements of a **SELECT** statement, namely

* **SELECT**
* **FROM**
* **WHERE**
* **ORDER BY**

*Use the following database to answer the questions below:* ***Table name: Comics***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CharacterID* | *Name* | *RealName* | *Abilities* | *FirstAppearance* |
| ***1*** | ***Iron Man*** | ***Tony Stark*** | ***Intelligent*** | ***1963*** |
| ***2*** | ***Spider-man*** | ***Peter Parker*** | ***Scientist*** | ***1962*** |
| ***3*** | ***Hulk*** | ***Robert Bruce Banner*** | ***Physical Strength*** | ***1962*** |
| ***4*** | ***Superman*** | ***Clark Kent*** | ***Super Strength, Flight*** | ***1938*** |
| ***5*** | ***Batman*** | ***Bruce Wayne*** | ***Rich*** | ***1939*** |
| ***6*** | ***Wonder Woman*** | ***Diana*** | ***Super Strength, Flight*** | ***1941*** |

1. If we want to retrieve all of the information about all of the characters in the Comics table, we could use the asterisk (\*) as a shortcut for all of the columns.   
   **Write the query to retrieve all columns for all records in the table.** *Answer:*

**$query=”SELECT \* FROM Comics”;**

i.e.

**name**

*Use the following database to answer the questions below:* ***Table name: Comics***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CharacterID* | *Name* | *RealName* | *Abilities* | *FirstAppearance* |
| ***1*** | ***Iron Man*** | ***Tony Stark*** | ***Intelligent*** | ***1963*** |
| ***2*** | ***Spider-man*** | ***Peter Parker*** | ***Scientist*** | ***1962*** |
| ***3*** | ***Hulk*** | ***Robert Bruce Banner*** | ***Physical Strength*** | ***1962*** |
| ***4*** | ***Superman*** | ***Clark Kent*** | ***Super Strength, Flight*** | ***1938*** |
| ***5*** | ***Batman*** | ***Bruce Wayne*** | ***Rich*** | ***1939*** |
| ***6*** | ***Wonder Woman*** | ***Diana*** | ***Super Strength, Flight*** | ***1941*** |

1. If we want only specific columns, we should explicitly specify them in a comma-separated list.  
   **Write the query to return the CharacterID, Name and RealName columns for all records in the table.**

*Answer:*

**$query = “SELECT CharacterID, Name, RealName FROM Comics”;**

*Use the following database to answer the questions below:* ***Table name: Comics***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CharacterID* | *Name* | *RealName* | *Abilities* | *FirstAppearance* |
| ***1*** | ***Iron Man*** | ***Tony Stark*** | ***Intelligent*** | ***1963*** |
| ***2*** | ***Spider-man*** | ***Peter Parker*** | ***Scientist*** | ***1962*** |
| ***3*** | ***Hulk*** | ***Robert Bruce Banner*** | ***Physical Strength*** | ***1962*** |
| ***4*** | ***Superman*** | ***Clark Kent*** | ***Super Strength, Flight*** | ***1938*** |
| ***5*** | ***Batman*** | ***Bruce Wayne*** | ***Rich*** | ***1939*** |
| ***6*** | ***Wonder Woman*** | ***Diana*** | ***Super Strength, Flight*** | ***1941*** |

1. Using the WHERE clause:   
   The next thing we want to do is to start limiting, or filtering, the data we fetch from the database. By adding a **WHERE** clause to the **SELECT** statement, we add one (or more) conditions that must be met by the selected data. This will limit the number of rows that answer the query and are fetched. We can continue with our previous query, and limit it to only those employees living in London:

**Write the query to limit only those characters who first appeared before 1960. Note, display the Name column only.**

*Answer:*

*What Names are returned?*

***Superman***

***Batman***

***Wonder Woman***

*Use the following database to answer the questions below:* ***Table name: Comics***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CharacterID* | *Name* | *RealName* | *Abilities* | *FirstAppearance* |
| ***1*** | ***Iron Man*** | ***Tony Stark*** | ***Intelligent*** | ***1963*** |
| ***2*** | ***Spider-man*** | ***Peter Parker*** | ***Scientist*** | ***1962*** |
| ***3*** | ***Hulk*** | ***Robert Bruce Banner*** | ***Physical Strength*** | ***1962*** |
| ***4*** | ***Superman*** | ***Clark Kent*** | ***Super Strength, Flight*** | ***1938*** |
| ***5*** | ***Batman*** | ***Bruce Wayne*** | ***Rich*** | ***1939*** |
| ***6*** | ***Wonder Woman*** | ***Diana*** | ***Super Strength, Flight*** | ***1941*** |

1. Using the WHERE clause:   
   Let us enhance the **WHERE** clause by looking at a slightly more sophisticated, comparison operator:

**Write the query to limit only those characters who first appeared in either 1962 or 1963.   
Note, display the Name column only.**

*Answer:*

*Use the following database to answer the questions below:* ***Table name: Comics***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CharacterID* | *Name* | *RealName* | *Abilities* | *FirstAppearance* |
| ***1*** | ***Iron Man*** | ***Tony Stark*** | ***Intelligent*** | ***1963*** |
| ***2*** | ***Spider-man*** | ***Peter Parker*** | ***Scientist*** | ***1962*** |
| ***3*** | ***Hulk*** | ***Robert Bruce Banner*** | ***Physical Strength*** | ***1962*** |
| ***4*** | ***Superman*** | ***Clark Kent*** | ***Super Strength, Flight*** | ***1938*** |
| ***5*** | ***Batman*** | ***Bruce Wayne*** | ***Rich*** | ***1939*** |
| ***6*** | ***Wonder Woman*** | ***Diana*** | ***Super Strength, Flight*** | ***1941*** |

1. Using the **LIKE** operator in the WHERE clause:   
   The **LIKE** operator allows us to perform basic pattern-matching using wildcard characters.  
   The **%** wildcard character – matches a string of one or more characters.

**Write the query to limit only those characters whose real name contains ‘Bruce’ anywhere in the name. Display the Name column only.**

*Answer:*

*What Names are returned:* ***Hulk****,* ***Batman***

*Use the following database to answer the questions below:* ***Table name: Comics***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CharacterID* | *Name* | *RealName* | *Abilities* | *FirstAppearance* |
| ***1*** | ***Iron Man*** | ***Tony Stark*** | ***Intelligent*** | ***1963*** |
| ***2*** | ***Spider-man*** | ***Peter Parker*** | ***Scientist*** | ***1962*** |
| ***3*** | ***Hulk*** | ***Robert Bruce Banner*** | ***Physical Strength*** | ***1962*** |
| ***4*** | ***Superman*** | ***Clark Kent*** | ***Super Strength, Flight*** | ***1938*** |
| ***5*** | ***Batman*** | ***Bruce Wayne*** | ***Rich*** | ***1939*** |
| ***6*** | ***Wonder Woman*** | ***Diana*** | ***Super Strength, Flight*** | ***1941*** |

1. Using the **LIKE** operator in the WHERE clause:   
   The **LIKE** operator allows us to perform basic pattern-matching using wildcard characters.  
   The **%** wildcard character – matches a string of one or more characters.

**Write the query to limit only those characters whose real name starts with ‘Tony’.   
Display the Name column only.**

*Answer:*

*What Names are returned:* ***Iron Man***

### The ORDER BY Clause

Once we have determined which columns and rows will be included in the results of our **SELECT** query, we may want to control the order in which the rows appear—sorting the data.

To sort the data rows, we include the **ORDER BY** clause. The **ORDER BY** clause includes one or more column names that specify the sort order.

Note: The default sort order is **ASC**. To specify descending order use **DESC**.

1. (a) Using the **ORDER BY** clause in the WHERE clause:

**Write the query to SORT the records in ascending order (lowest to highest).   
Display the Name column only.**

*Answer:*

*What Names are returned in what order:*

***Batman****,* ***Hulk****,* ***Iron Man, Spider-man, Superman, Wonder Woman***

**(b) To reverse the above order, change the query to:**

*Answer:*

*What Names are returned in what order:*

***Wonder Woman, Superman, Spider-man, Iron Man, Hulk, Batman***

*WEEK#4 TASK LIST\_php1.docx*