# Problem 2 – SoftUni SQL

The students from **SoftUni** had an assignment for their **PHP Basics Team project** to design and implement a **Blog** / **Forum** / **Photo Album** / **Audio Album.** They had to use **MySQL Database**. And they did. Unfortunately one day before the public defense **MySQL stopped being open source** and all the commands it provided stopped working worldwide. The students were left with Databases and **no commands to access them**. Help them finish their projects in time. Your task is to **write PHP code** that takes as **input SQL commands** and makes them **work**.

You will receive **3 types** of commands**: INSERT INTO**, **UPDATE**, **DELETE FROM**.

Each **INSERT INTO** command has the **following parts**:

* **"INSERT INTO"**
* The name of the **table**
* So called **'fields'** - the names of the fields that will be filled with the command, enclosed in **brackets** and **separated by commas**
* So called **'values'** – the values of the fields in the corresponding order as the 'fields', enclosed in **brackets** and **separated by commas**

Example: **"INSERT INTO users (user\_id, login, gender, age) VALUES (1, yana, female, 20)"**

Each **UPDATE** command has the **following parts**:

* **"UPDATE"**
* The name of the **table**
* **"SET"**
* So called **'update field'** – the name of the field and its new value, enclosed in **brackets** and **separated by "="**
* **"WHERE"**
* So called **'update condition'** – a field name and a field value, enclosed in **brackets** and **separated by "="**

Example: **" UPDATE users SET (login = root) WHERE (user\_id = 1)"**

Note: The command **"UPDATE"** can change only **one field** in the database table. It can have only **one** **'update condition'** which will always be for **equality**. The field **'user\_id'** can **not** be updated.

Each **DELETE** command has the **following parts**:

* "**DELETE FROM**"
* The name of the **table**
* "**WHERE**"
* So called **'delete condition'** – a field name and a field value, enclosed in **brackets** and **separated by "="**

Example: **"DELETE FROM users WHERE (user\_id = 178)"**

Note: The command "**DELETE**" can erase **more than one entry** (row) from the database table. It can only have **one 'delete condition'** which will always be for **equality**.

A **command is not valid** (you have **error**) in the **following cases**:

"**INSERT INTO**" **errors**:

* The starting word **'INSERT' is missing**
* There is **no 'login' field**
* The **number** of 'fields' and 'values' is **different** (if they are equal, they will always be arranged correctly: user\_id, login, gender, age)

"**UPDATE**" errors:

* The starting word **"UPDATE" is missing**
* The **'update condition'** refers to non **existing 'field'**
* The **'update field'** is non **existing 'field'**
* The **'update condition'** is a combination of **existing 'field' – 'value'**

**"DELETE FROM"** errors:

* The starting word **"DELETE" is missing**
* When you try to **delete the field 'login'**
* The field from **'delete condition'** does **not exist**

Note: Command **"DELET FROM"** – when the **'delete condition' is not valid**, for example **"DELETE FROM users WHERE (user\_id = 1)"** if we **do not have 'field' 'user\_id'** with **'value' = '1'**, there is **no error,** just **nothing happens**. If on the same conditions the **table is empty**, **nothing happens** as well.

We have **'default values'** which we use when we do not have the corresponding 'field' and 'value' in the input. The 'default values' are the **following**:

For **'user\_id'** – we have **autoincrement** – we get the **biggest existing 'value'** for 'user\_id' and we **increment it by 1**. Example: If we first add user\_id = 0, then we add user\_id=12, then we do not have value for user\_id, the user\_id will be 13 automatically (12 + 1). If on the first record we do not have user\_id value, we apply user\_id=0

For **'age' and 'gender'** – we have default value string **'undefined'**

### Input

The input will be read from an **HTTP GET** **request**. The **list of commands** will be received as a **string array** from a **text** **input field with name 'commands'**. The **values** of the fields will **not** be enclosed in **quotes**.

### Output

Output consists of a **table** with **4 columns** – **user\_id, login, gender, age**. The table has a **<thead>, <tbody>, <tfoot>.** The **<thead>** consists of **4 cells** with the **field names**. The **<tbody>** consists of the **inserted** and **updated data** from the input. The **<tfoot>** consists of only **one cell** that contains the **number of errors**. If after all the commands are processed we have **no entries**, the output consists of a **simple sentence** **"You have {0} error/s"**, where {0} is the count of the errors, **{0} >= 0**. See example below.

### Constraints

* The **separator** of the **'fields', 'values'** will always be **', '** – exactly **one comma** and **one space**
* The **separator** of the **'update fields', 'update condition', 'delete condition'** will always be **'='** – **one symbol** with **no or many spaces** around it
* There are exactly 4 types of fields - **user\_id, login, gender, age.** They will **always** come **arranged** this way, but you may have **less than 4 per command**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | | **Output** |
| commands | ["UPDAtE users SET (age = 30) WHERE (user\_id = 1)", "INSERT INTO users (login, age, gender) VALUES (yana, 20, female)", "UPDATE users SET (age = 30) WHERE (user\_id = 1)", "INSERT INTO users (login, age) VALUES (yana, 20, female)", "INSERT INTO users (login, age, gender) VALUES (yana, 20, female)", "INSERT INTO users (user\_id, login, gender) VALUES (12, yana, female)", "INSERT INTO users (login, age, gender) VALUES (yana, 20, female)", "INSrRT INTO users (login, age, gender) VALUES (yana, 20, female)", "UPDATE users SET (age = 30) WHERE (user\_id = 1)", "UPDATE users SET (age = 30) WHERE (age = undefined)", "DELETE FROM users WHERE (age = 30)"]; | <table><thead><tr><th>user\_id</th><th>login</th><th>gender</th><th>age</th></tr></thead><tbody><tr><td>0</td><td>yana</td><td>female</td><td>20</td></tr><tr><td>13</td><td>yana</td><td>female</td><td>20</td></tr></tbody><tfoot><tr><td colspan="4">Errors=4</td></tr></tfoot></table> |
| **Comment** | | |
| We have array of strings that contains 11 commands. Command 1 gives error (doesn't start with "UPDATE"). Command 2 is valid (gets in the database). Command 3 gives error (we have only one entry and it has 'user\_id' = 0). Command 4 gives error (has more 'values' than 'fields'). Command 5 is valid – we have new entry with 'user\_id' = 1. Command 6 is valid – we have new entry with 'user\_id' = 12 and default value 'undefined' for 'age'. Command 7 is valid – we have new entry with 'user\_id' = 13. Command 8 gives error (doesn't start with "INSERT INTO"). Command 9 is valid and updates entry with 'user\_id = 1'. Command 10 is valid. Command 11 is valid. | | |

|  |  |  |
| --- | --- | --- |
| **Input** | | **Output** |
| commands | [INSERT INTO users (login, age, gender) VALUES (yana, 20, female)", "DELETE FROM users WHERE (age = 20)", "DELETE FROM users WHERE (age = 30)", "UPDATE users SET (age = 30) WHERE (user\_id = 1)"] | You have 1 error/s |
| **Comment** | | |
| We have array of strings that contains 4 commands. Command 1 is valid (gets in the database). Command 2 is valid and deletes the entry we have. Command 3 is valid and does nothing. Command 4 gives error. | | |