Introduction to MySQL & PHP

Samar Abdelghani

An Overview of SQL

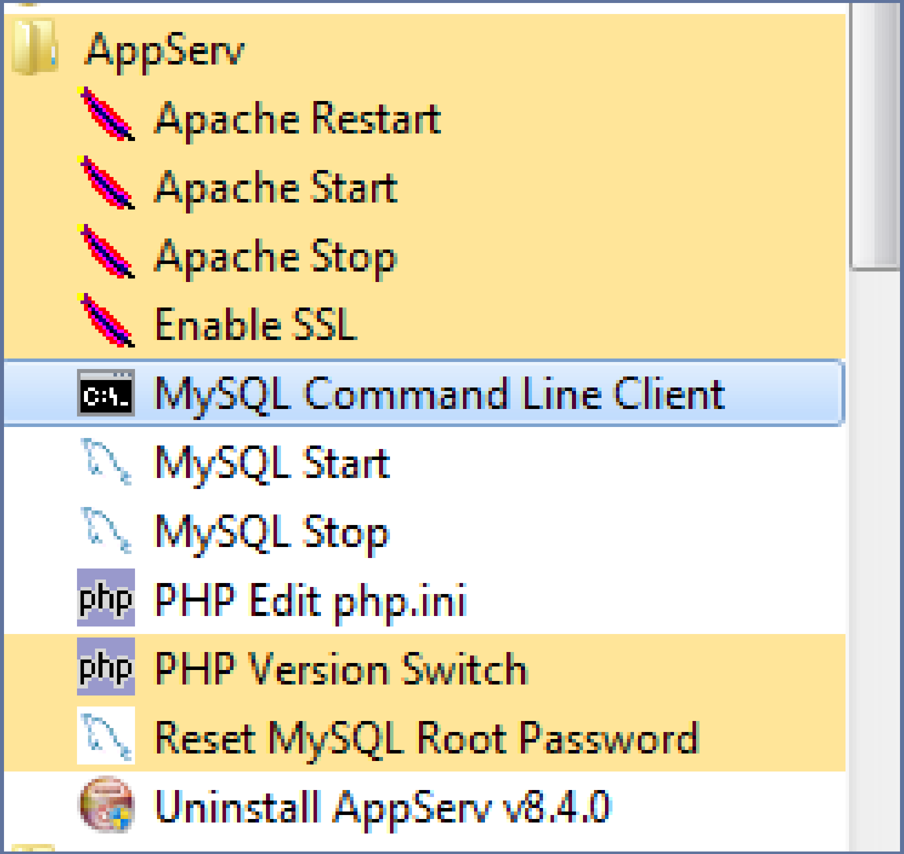
- SQL stands for Structured Query Language.
- Communicate with databases
- Used to created and edit databases.
- Also used to create queries, forms, and reports

- You can create, populate, modify, update and delete a mysql database directly from the command line.

- There is an alternative method, and that is to use a GUI, such as phpMyAdmin. phpMyAdmin allows you to maintain MYSQL databases from your browser, with more clicking and less typing.

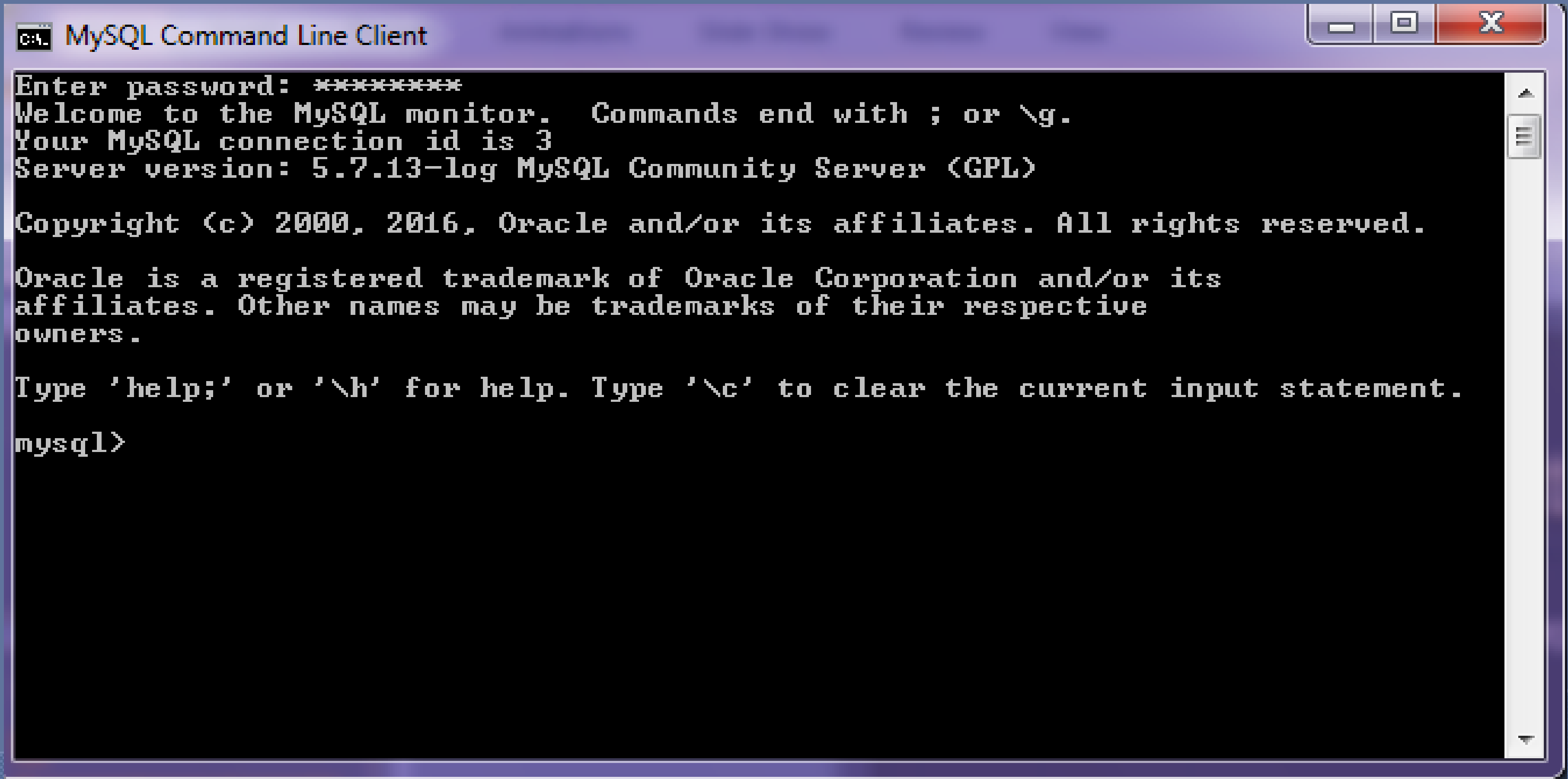
MySQL Monitor

- Let's open MySQL monitor from the **Start** menu on Windows.



MySQL Monitor

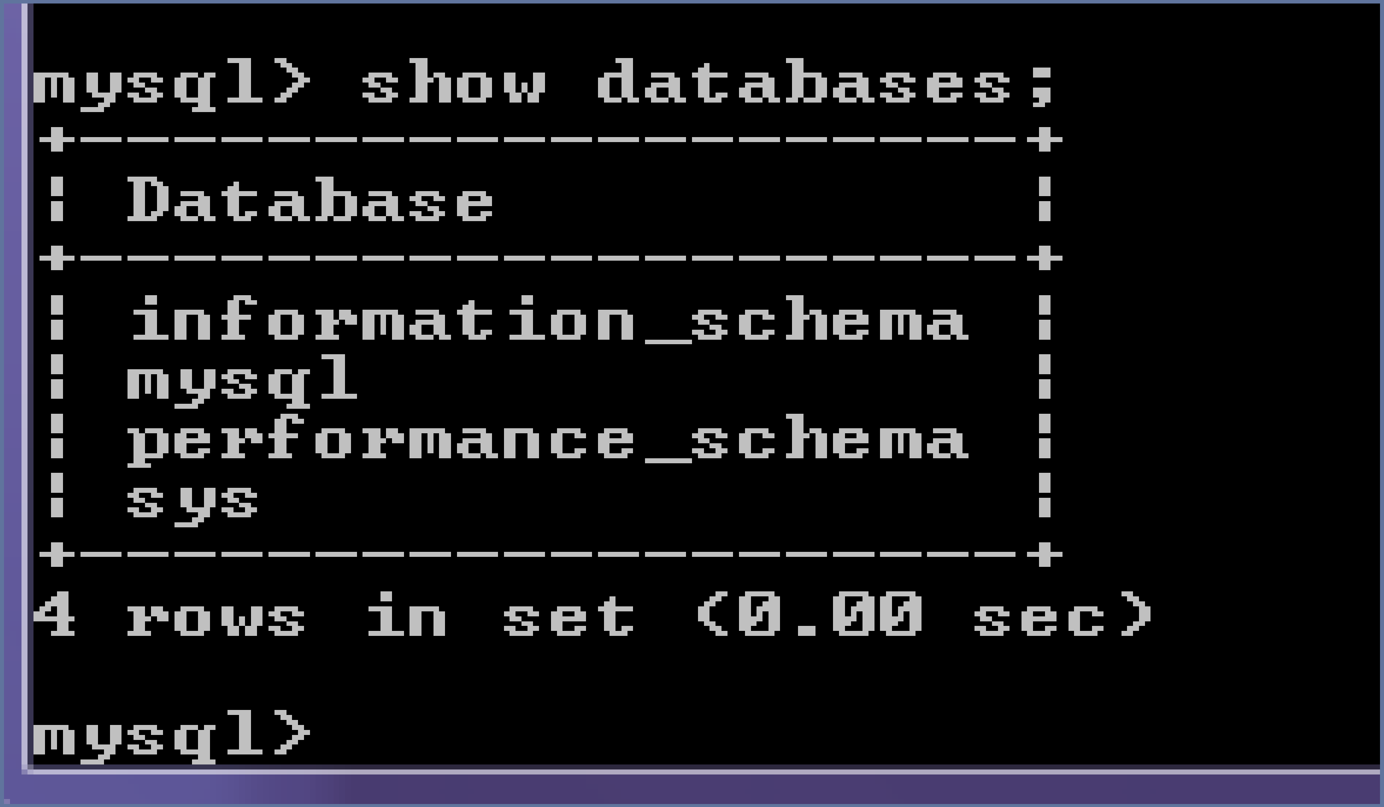
- After type in the password, MySQL prompt window should look like this.



```
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.13-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

MySQL Monitor

- The name of all databases on the MySQL server can be displayed with the **show databases**;

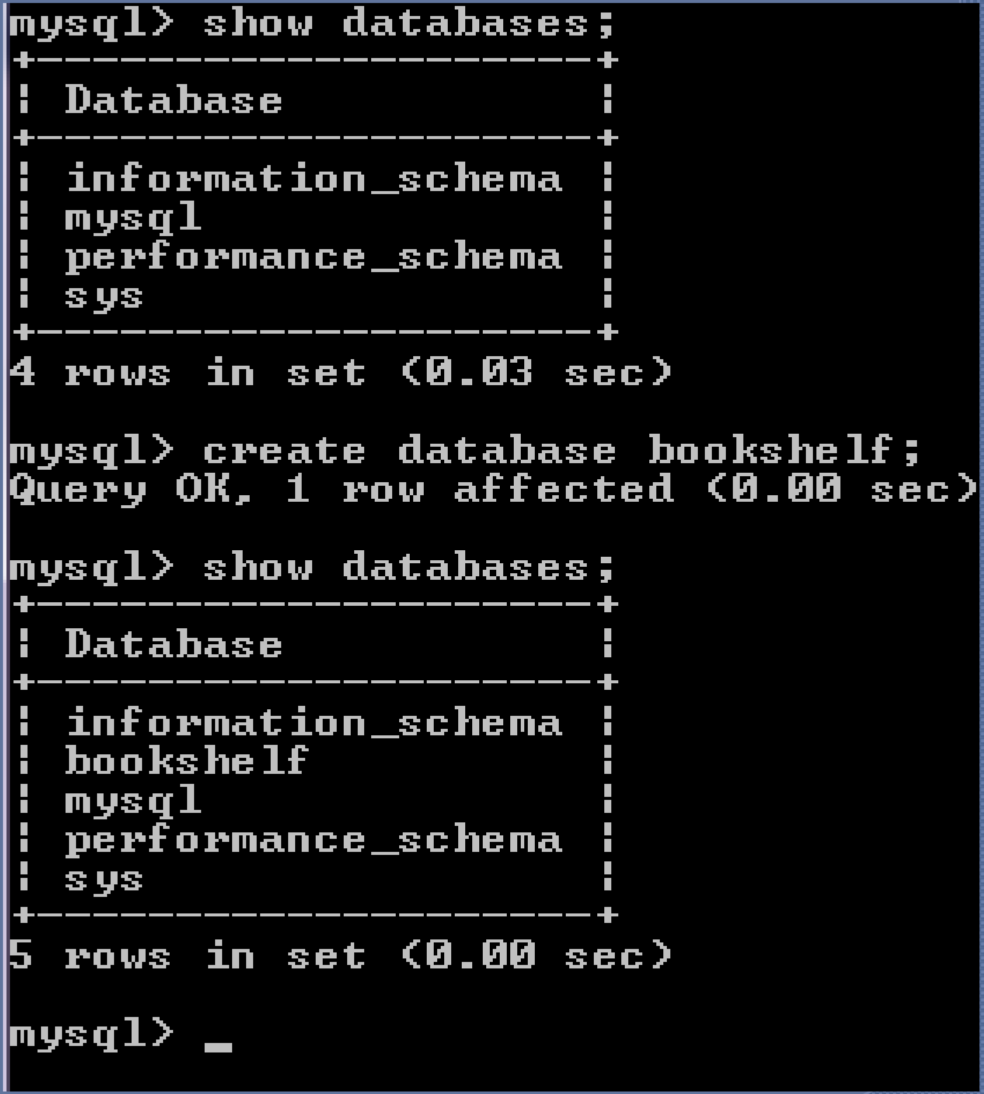


```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

Creating a Database

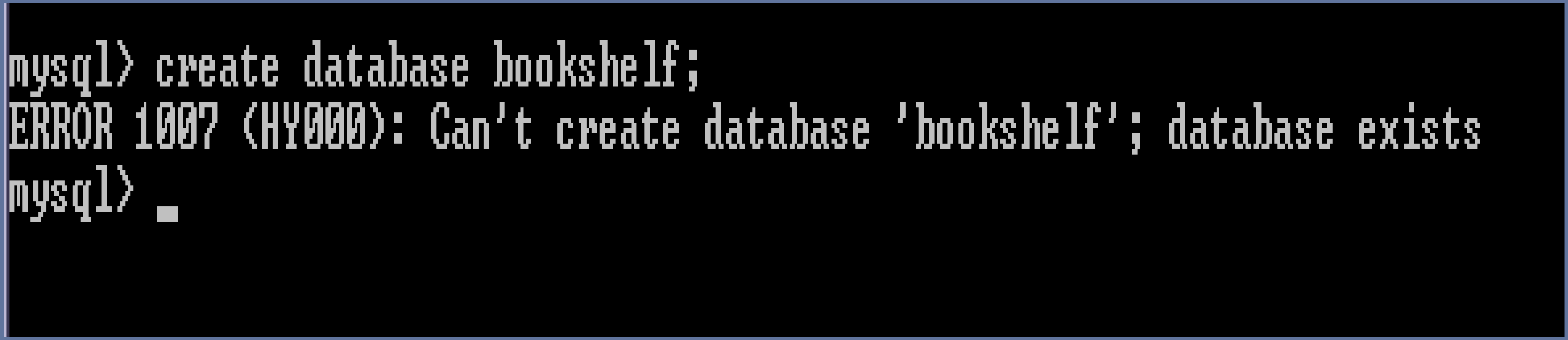
To begin, you must first CREATE a database using the following SQL statement:

```
CREATE DATABASE database_name
```



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.03 sec)

mysql> create database bookshelf;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bookshelf          |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> _
```

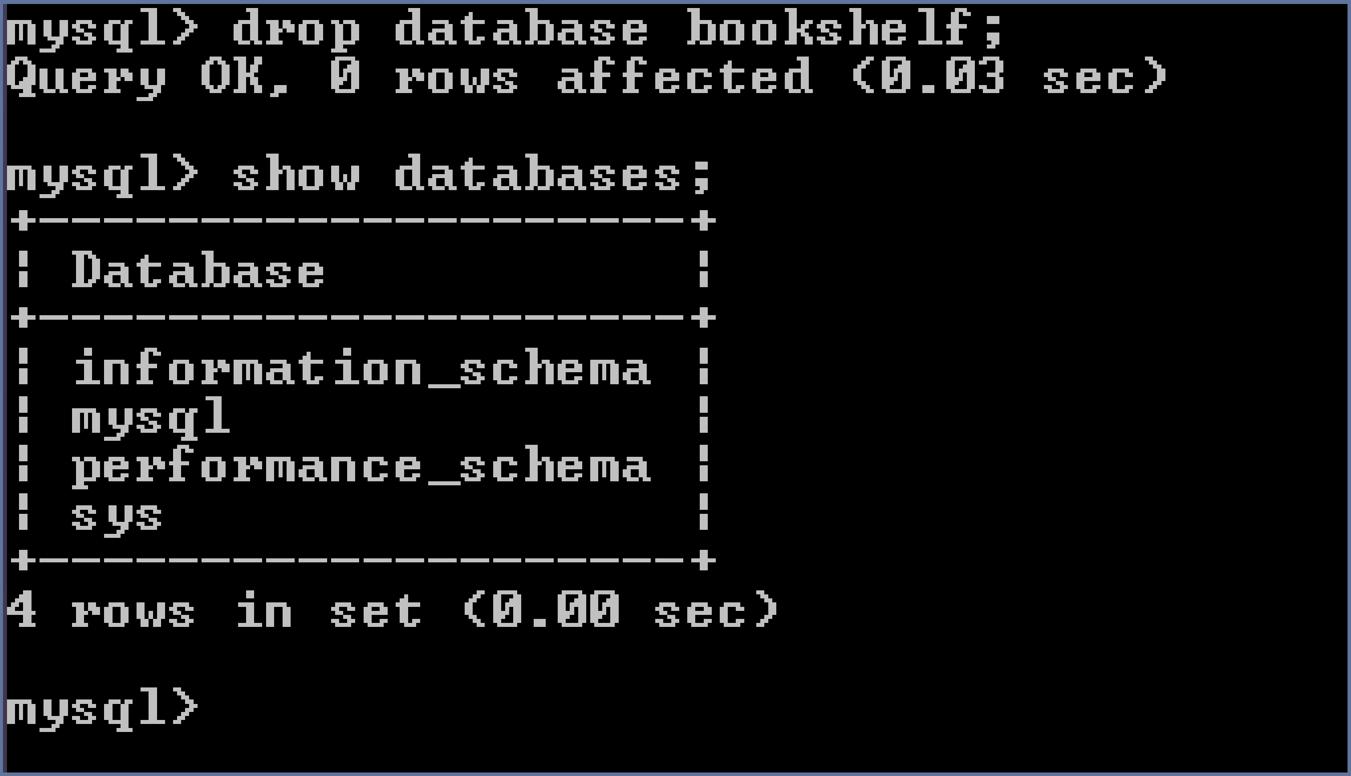
- If a database already exists when we try to create the database, it will give us the error message:



```
mysql> create database bookshelf;
ERROR 1007 (HY000): Can't create database 'bookshelf'; database exists
mysql> _
```

Dropping(Deleting) a Database

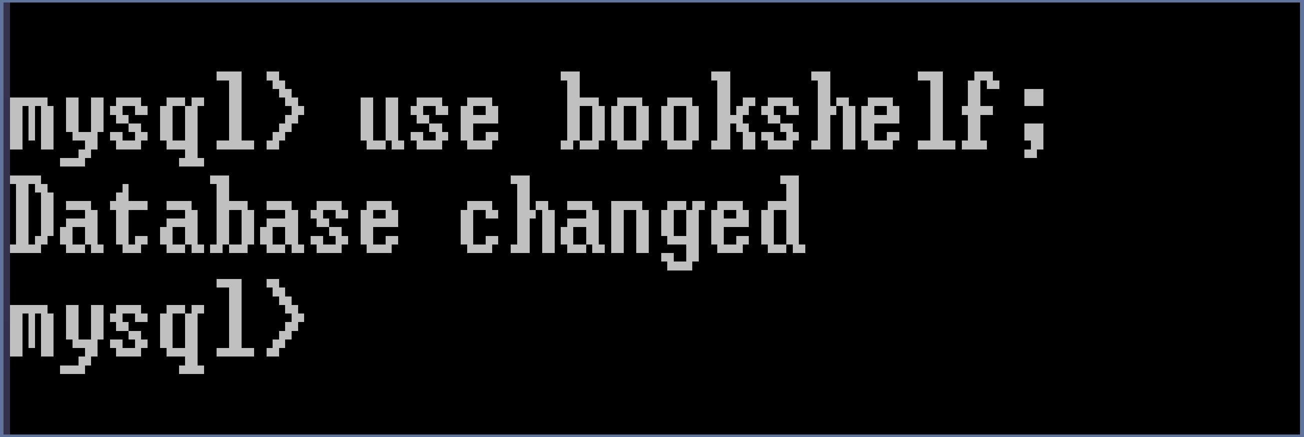
- To delete/remove a database, we use **drop** command. So, use **drop database bookshelf;** to remove **bookshelf** from the database.



```
mysql> drop database bookshelf;
Query OK, 0 rows affected (0.03 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

Creating a Database Table

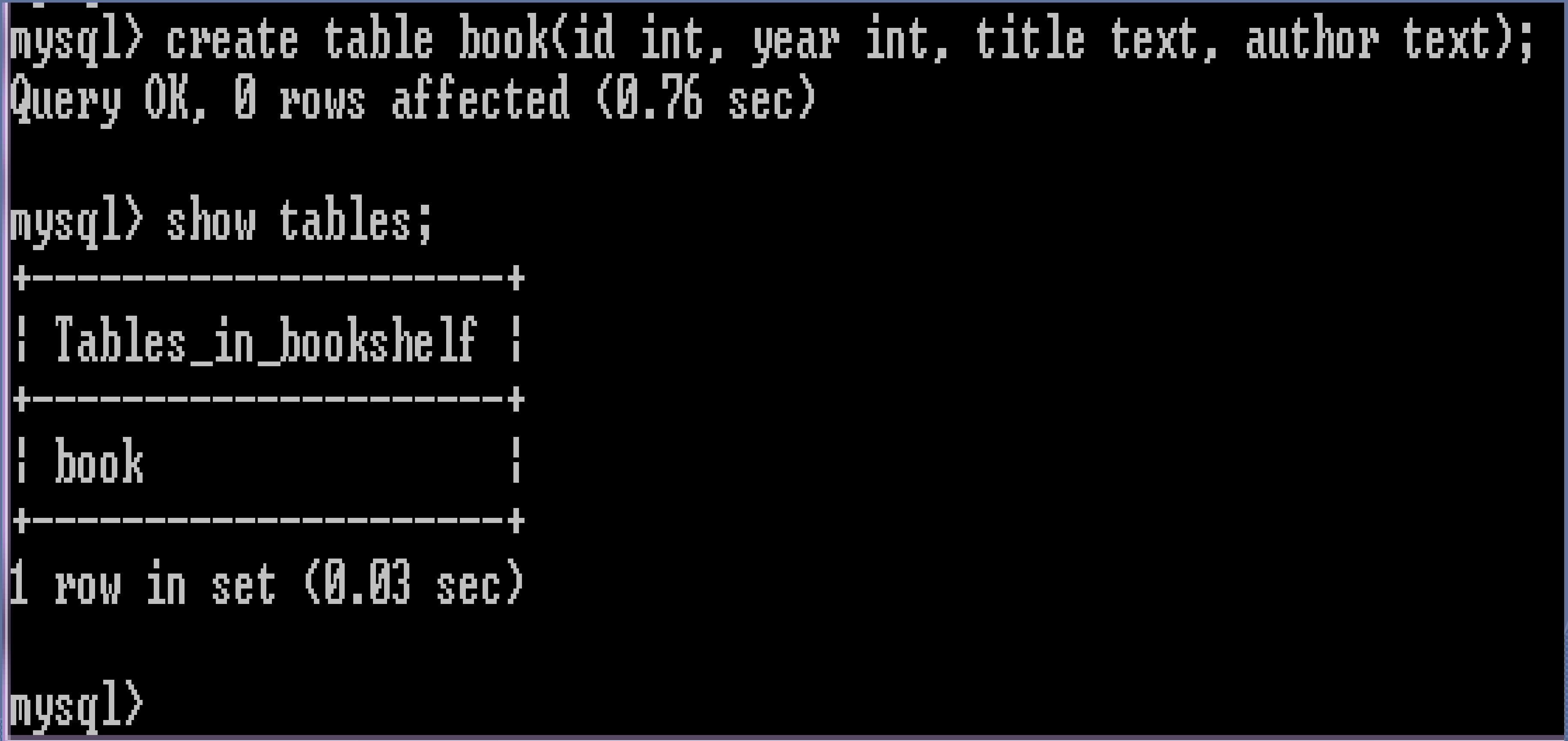
- Before we create a database table we need to select the database to which it is to be added. The command for this is **use *database_name*;**. This requires one of the existing database names.



```
mysql> use bookshelf;
Database changed
mysql>
```

Creating a Database Table

- We create a new table using **create table *table_name*;** with a comma-separated list of column names.



```
mysql> create table book(id int, year int, title text, author text);
Query OK, 0 rows affected (0.76 sec)

mysql> show tables;
+---------------------+
| Tables_in_bookshelf |
+---------------------+
| book                |
+---------------------+
1 row in set (0.03 sec)

mysql>
```

Data Types

TABLE 7.4 Some Common SQL Data Types

DATA TYPE	FORMAT	COMMENTS
Numeric	NUMBER(L,D)	The declaration NUMBER(7,2) indicates numbers that will be stored with two decimal places and may be up to six digits long, including the sign and the decimal place. Examples: 12.32, -134.99.
	INTEGER	May be abbreviated as INT. Integers are (whole) counting numbers, so they cannot be used if you want to store numbers that require decimal places.
	SMALLINT	Like INTEGER, but limited to integer values up to six digits. If your integer values are relatively small, use SMALLINT instead of INT.
	DECIMAL(L,D)	Like the NUMBER specification, but the storage length is a *minimum* specification. That is, greater lengths are acceptable, but smaller ones are not. DECIMAL(9,2), DECIMAL(9), and DECIMAL are all acceptable.
Character	CHAR(L)	Fixed-length character data for up to 255 characters. If you store strings that are not as long as the CHAR parameter value, the remaining spaces are left unused. Therefore, if you specify CHAR(25), strings such as "Smith" and "Katzenjammer" are each stored as 25 characters. However, a U.S. area code is always three digits long, so CHAR(3) would be appropriate if you wanted to store such codes.
	VARCHAR(L) or VARCHAR2(L)	Variable-length character data. The designation VARCHAR2(25) will let you store characters up to 25 characters long. However, VARCHAR will not leave unused spaces. Oracle users may use VARCHAR2 as well as VARCHAR.
Date	DATE	Stores dates in the Julian date format.

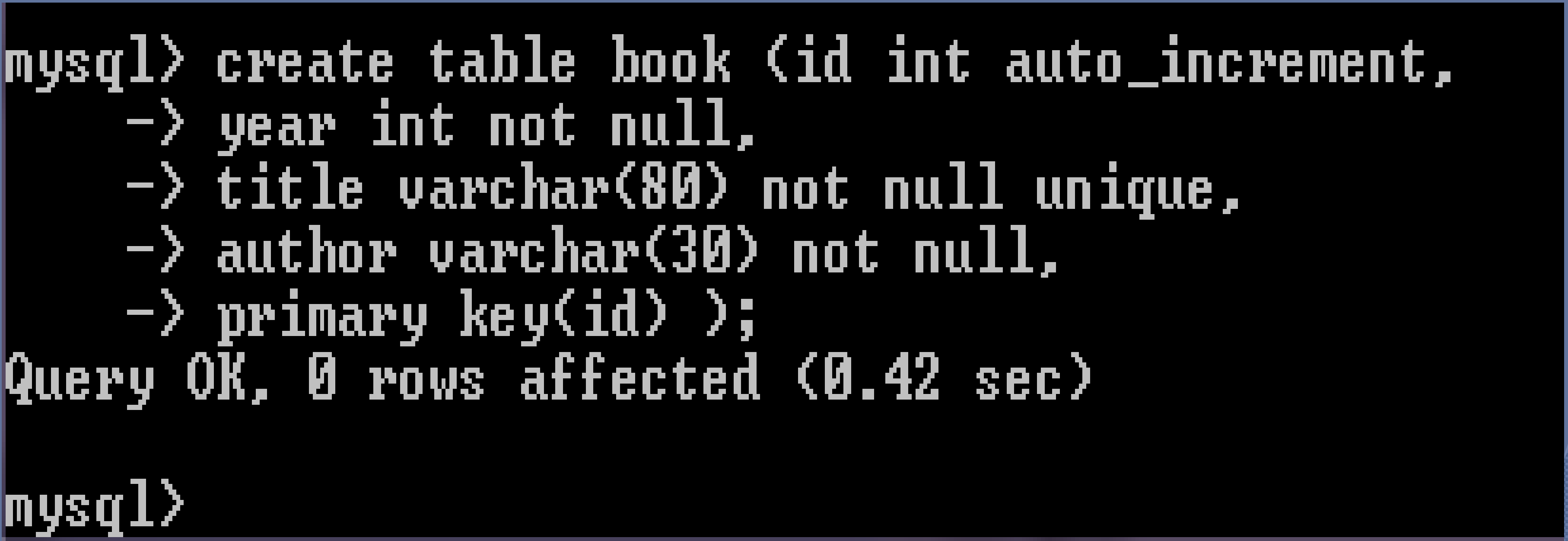
Column Modifier

- The modifiers in the table below can be selected to control how a column should be used.

Modifier	Description
not null	Each record should include data entry in this column
unique	Records may not duplicate any entry in this column
auto_increment	A variable only for numeric columns to automatically generate a number that is one more than the previous value in that column
primary key()	Specifies as its argument the name of the column to be used as the primary key for that table, i.e., primary key(id).

Creating a Database Table

The create table table_name command displayed in the example below can automatically number the primary key id column. Each record should include data in the year, title, and author columns. No duplicate entries are permitted in the title column.



```
mysql> create table book (id int auto_increment,
    -> year int not null,
    -> title varchar(80) not null unique,
    -> author varchar(30) not null,
    -> primary key(id) );
Query OK, 0 rows affected (0.42 sec)

mysql>
```

Table Data - Primary Key

- A **PRIMARY KEY** is a **constraint** that is applied to a column to uniquely identify each row of that database table. It ensures that the values in each row of that column are unique and never change, so those values can be used to reference any specified row.
- By setting the PRIMARY KEY constraint it is possible to manipulate data on specific rows of the database table.

- Any column can be set as the PRIMARY KEY but is often the first column that is used to provide a unique identifying number.

Table Data - Primary Key

- Any column set as the PRIMARY KEY must meet the following criteria:

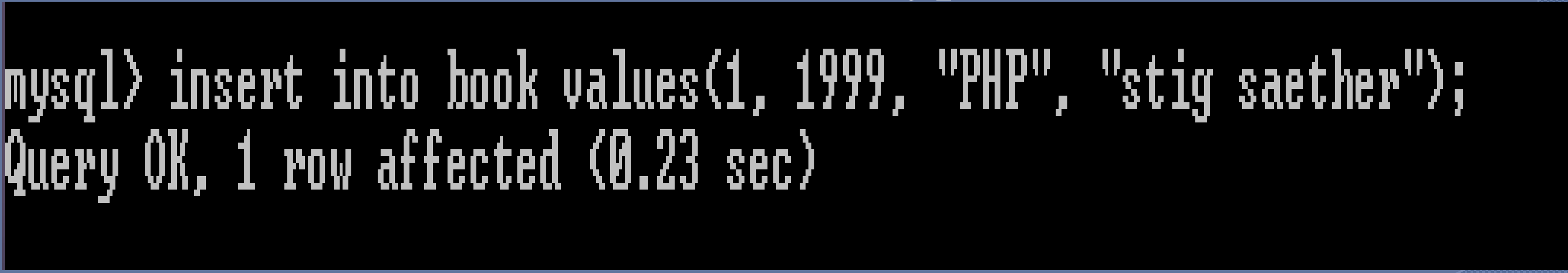
1. Each field in that column must have a value - it should not be empty or have a NULL value.
2. Each value in that column must be unique - there must be no duplications.
3. Each value in that column never be modified or updated.
4. Each value in that column cannot be re-used - when a row is deleted its PRIMARY KEY value cannot be re-assigned as the PRIMARY KEY value of a new row.

Table Data - Inserting

- After a table created, data can be entered into it with **insert into** command:

```
insert into table_name values(value1, value2, value3);
```

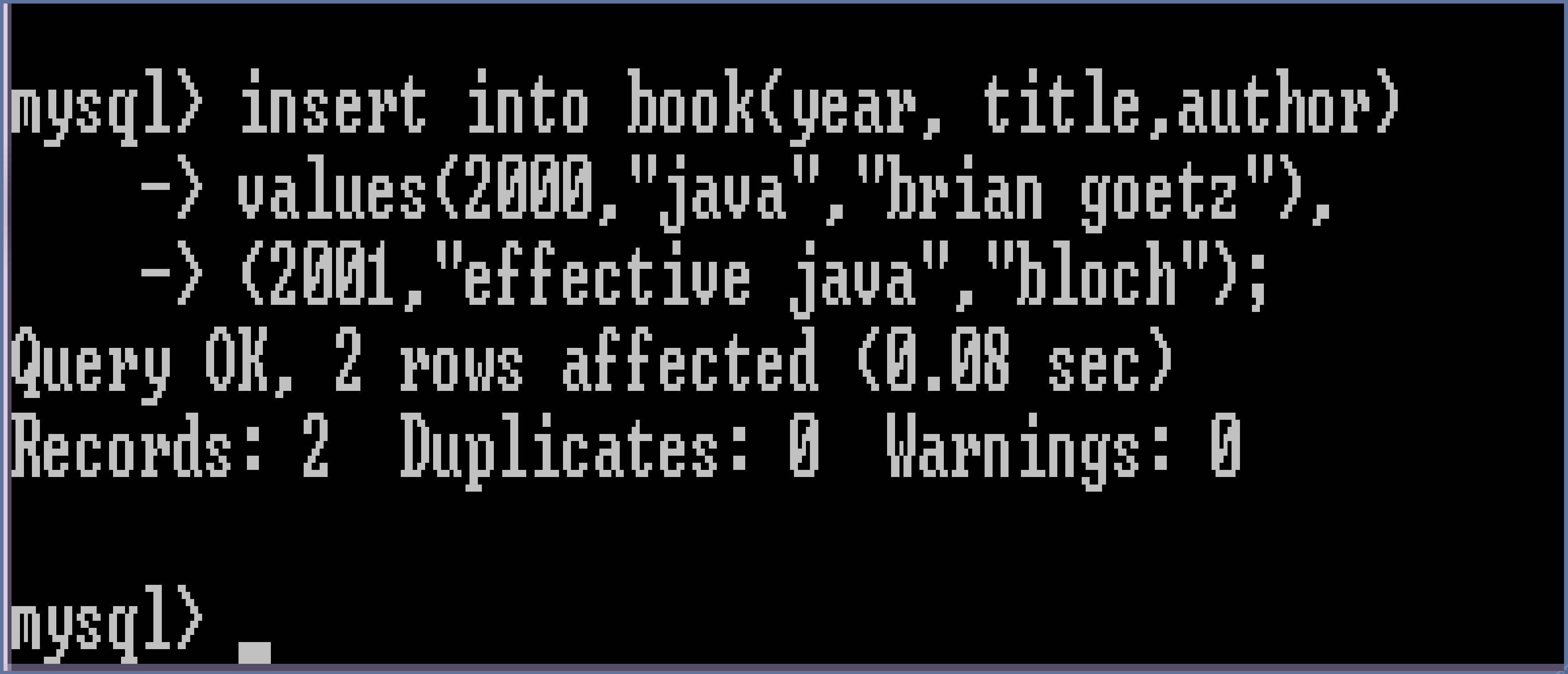
- The data values are entered as comma-separated arguments to the value() function; the list must correspond to the number of table columns and each value must be of the correct data type.



```
mysql> insert into book values(1, 1999, "PHP", "stig saether");
Query OK, 1 row affected (0.23 sec)
```

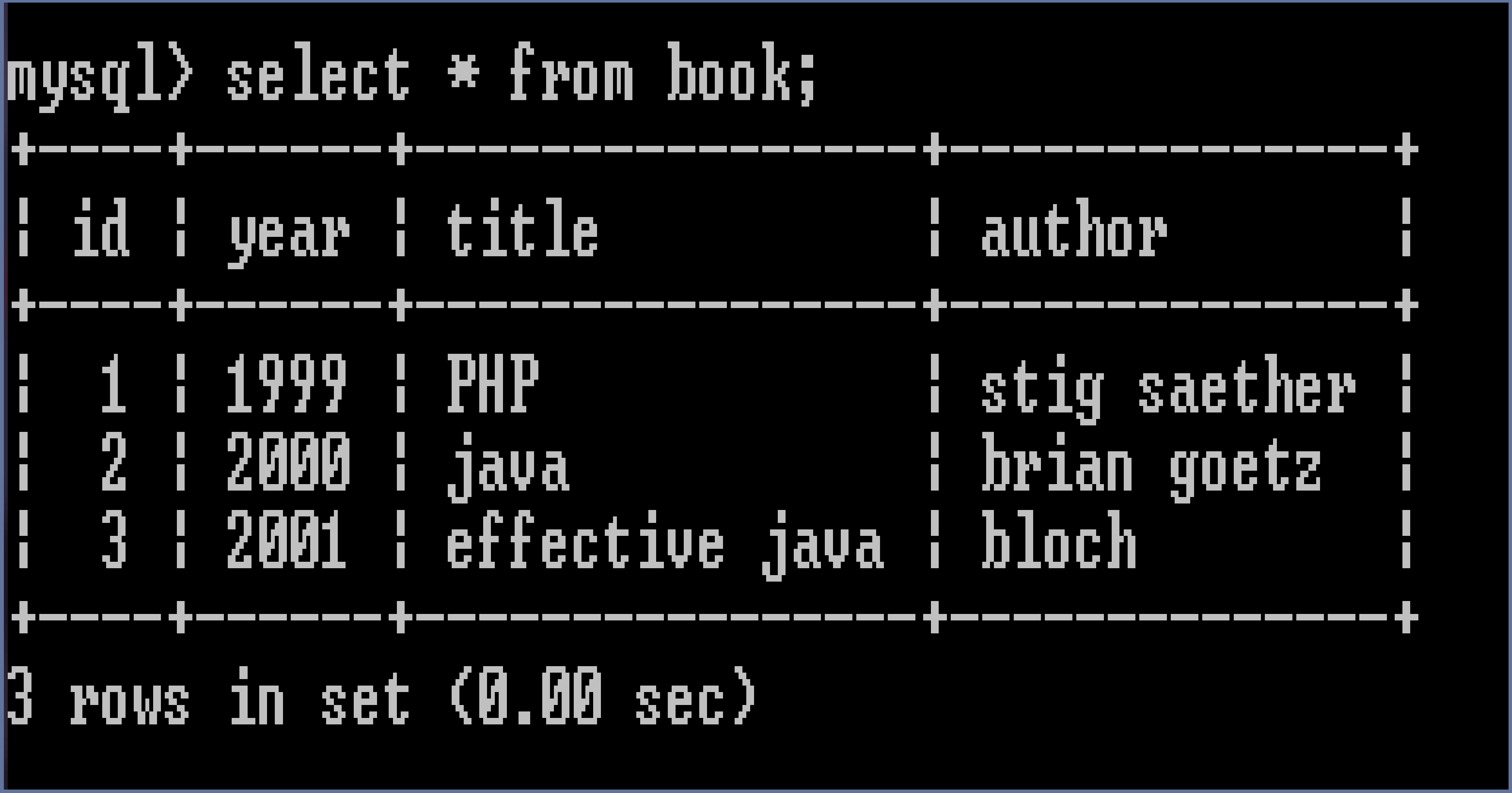
Table Data - Inserting

- Another way to insert data into a table is to specify the names of the columns where the data is to be added.



```
mysql> insert into book(year, title,author)
    -> values(2000,"java","brian goetz"),
    -> (2001,"effective java","bloch");
Query OK, 2 rows affected (0.08 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> _
```

- Entire table can be viewed with
- **select * from** table_name;

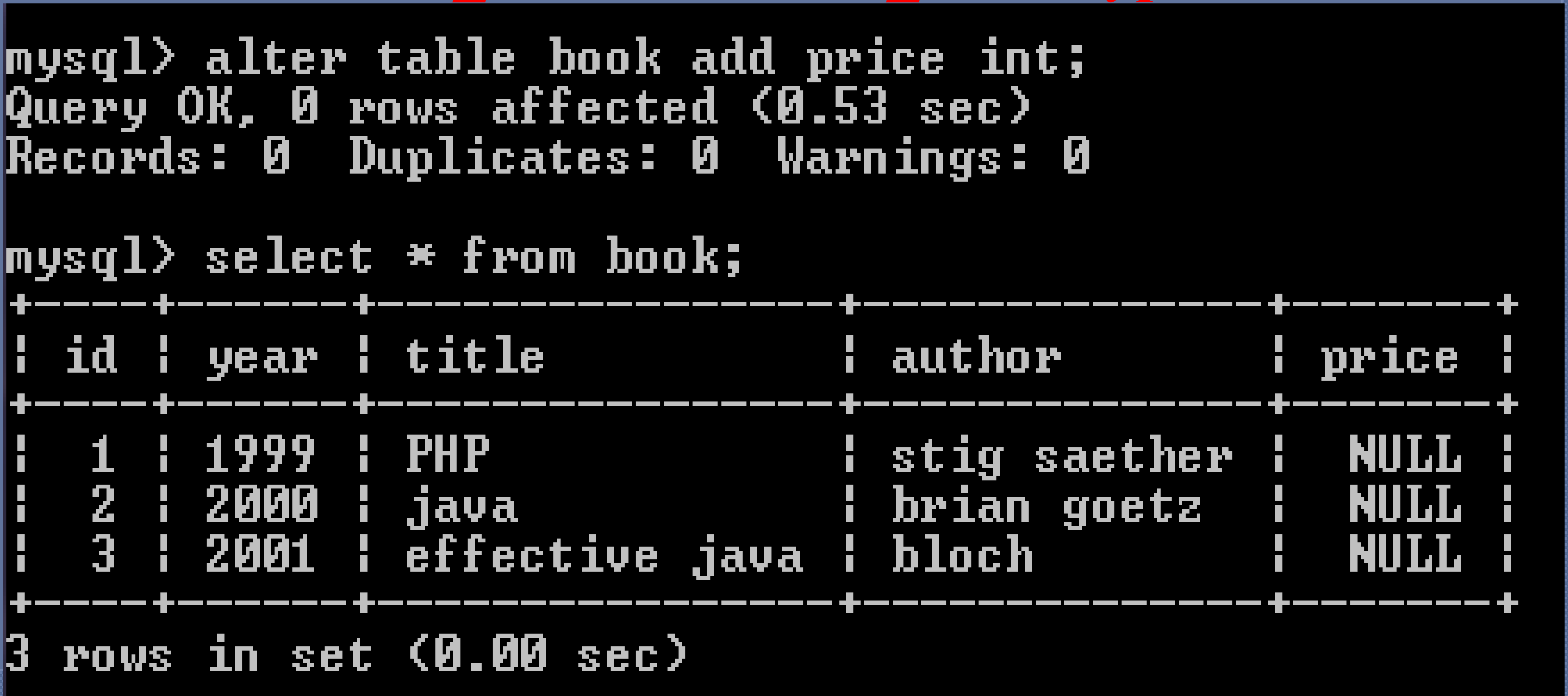


```
mysql> select * from book;
+----+------+-------------------+-----------------+
| id | year | title             | author          |
+----+------+-------------------+-----------------+
|  1 | 1999 | PHP               | stig saether    |
|  2 | 2000 | java              | brian goetz     |
|  3 | 2001 | effective java    | bloch           |
+----+------+-------------------+-----------------+
3 rows in set (0.00 sec)
```

Table Data - Altering

- New columns can be added to an existing table using **alter table** and **add**.

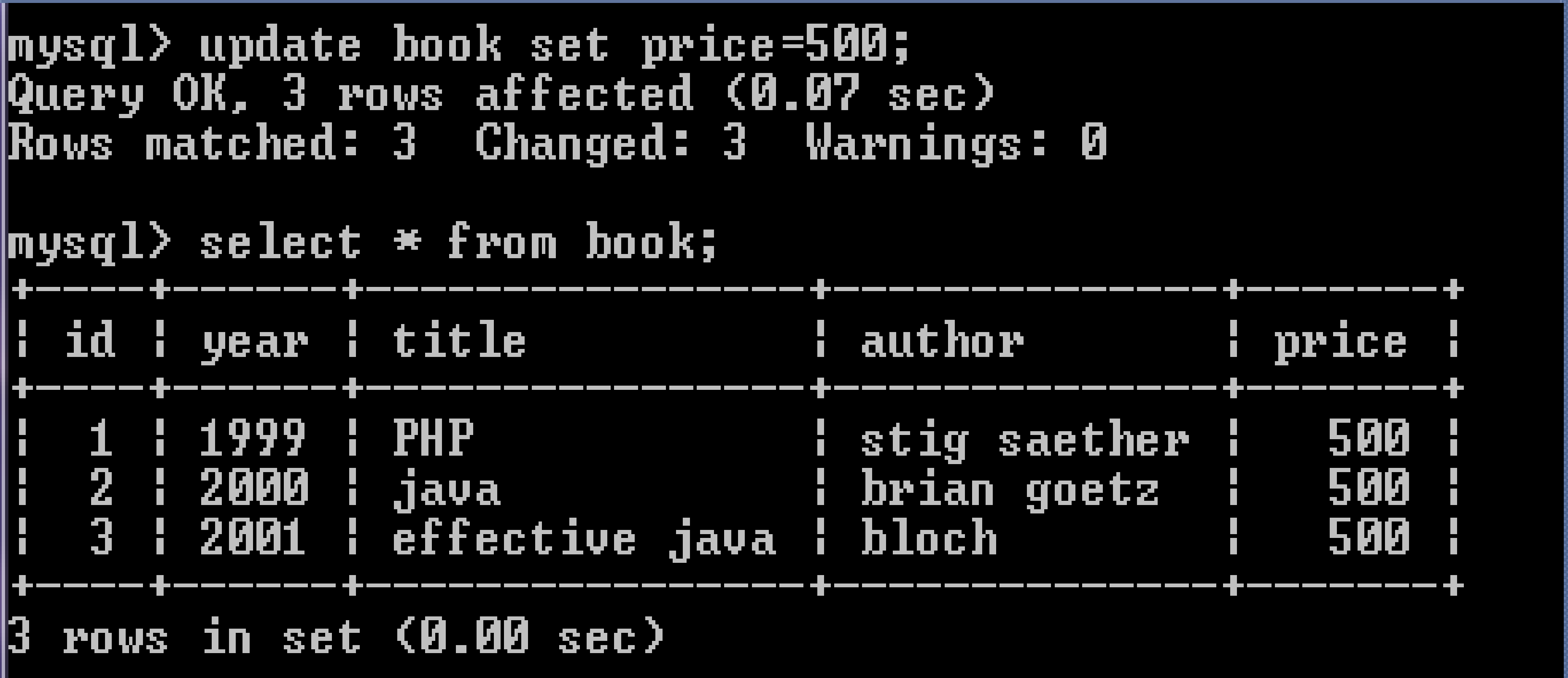
alter table *table_name* add *field_name type modifier*



```
mysql> alter table book add price int;
Query OK, 0 rows affected (0.53 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from book;
+----+------+------------------+-----------------+-------+
| id | year | title            | author          | price |
+----+------+------------------+-----------------+-------+
|  1 | 1999 | PHP              | stig saether    |  NULL |
|  2 | 2000 | java             | brian goetz     |  NULL |
|  3 | 2001 | effective java   | bloch           |  NULL |
+----+------+------------------+-----------------+-------+
3 rows in set (0.00 sec)
```

Table Data - Updating

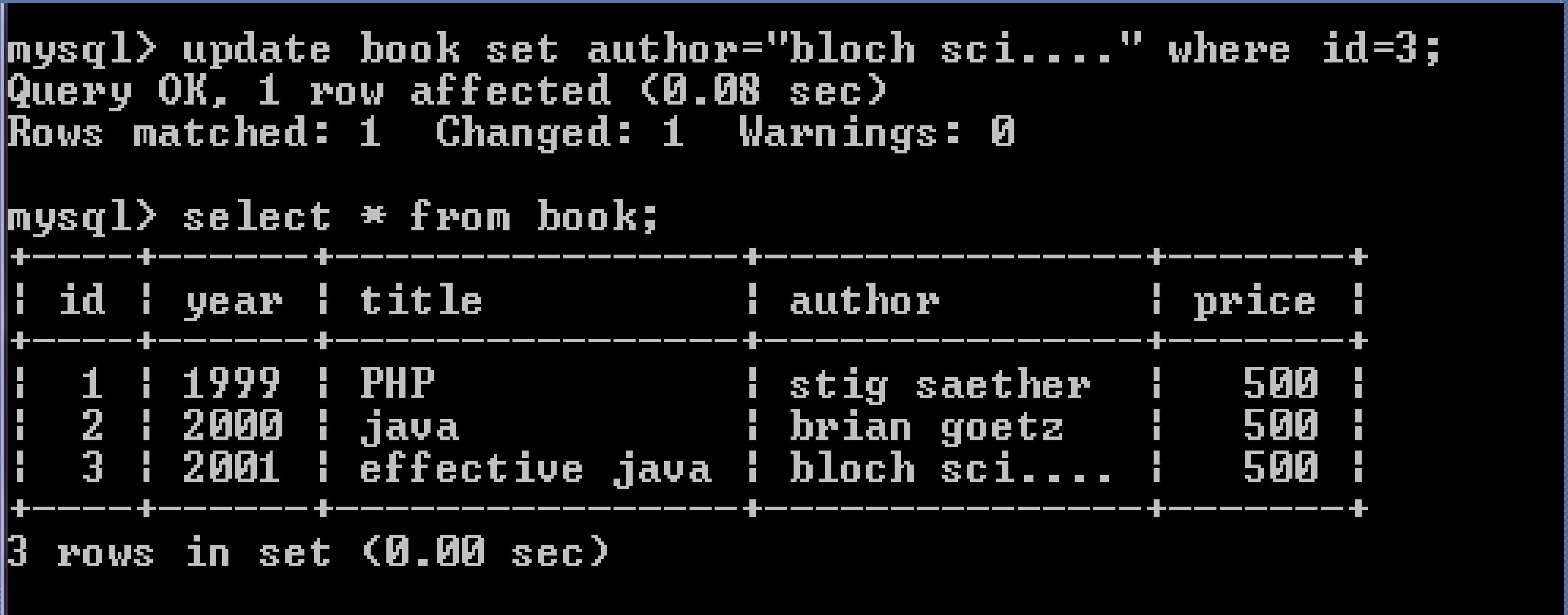
- all data values in a table column can be changed using **update** command with **set**:
  ***update** table_name **set** field_name = new_value;*



```
mysql> update book set price=500;
Query OK, 3 rows affected (0.07 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> select * from book;
+----+------+----------------+---------------+-------+
| id | year | title          | author        | price |
+----+------+----------------+---------------+-------+
|  1 | 1999 | PHP            | stig saether  |   500 |
|  2 | 2000 | java           | brian goetz   |   500 |
|  3 | 2001 | effective java | bloch         |   500 |
+----+------+----------------+---------------+-------+
3 rows in set (0.00 sec)
```

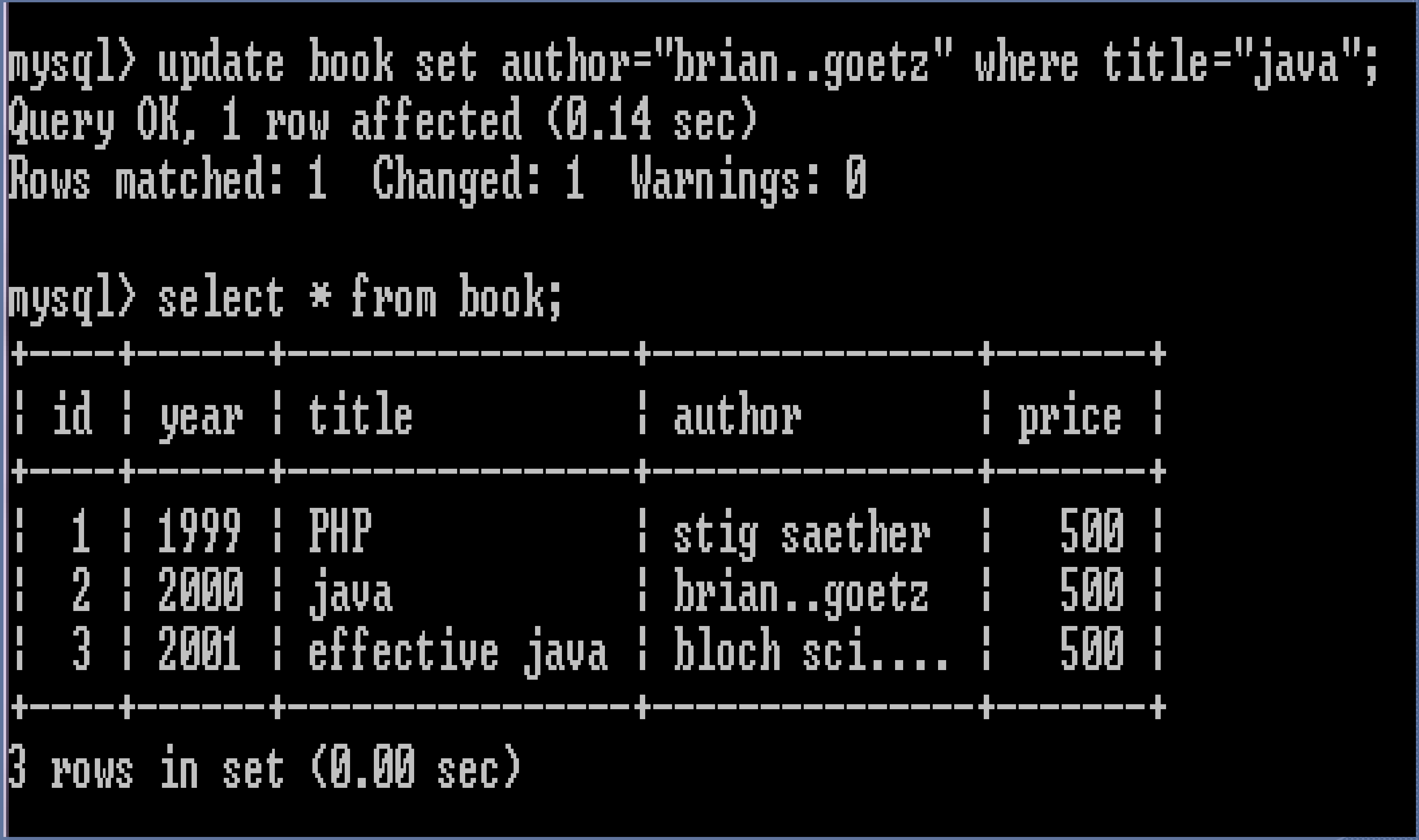
Table Data - Updating

- Individual column values can be changed by adding a qualifier to the syntax with **where**:
  update *table_name* set *field_name = new_value* where *id = int;*



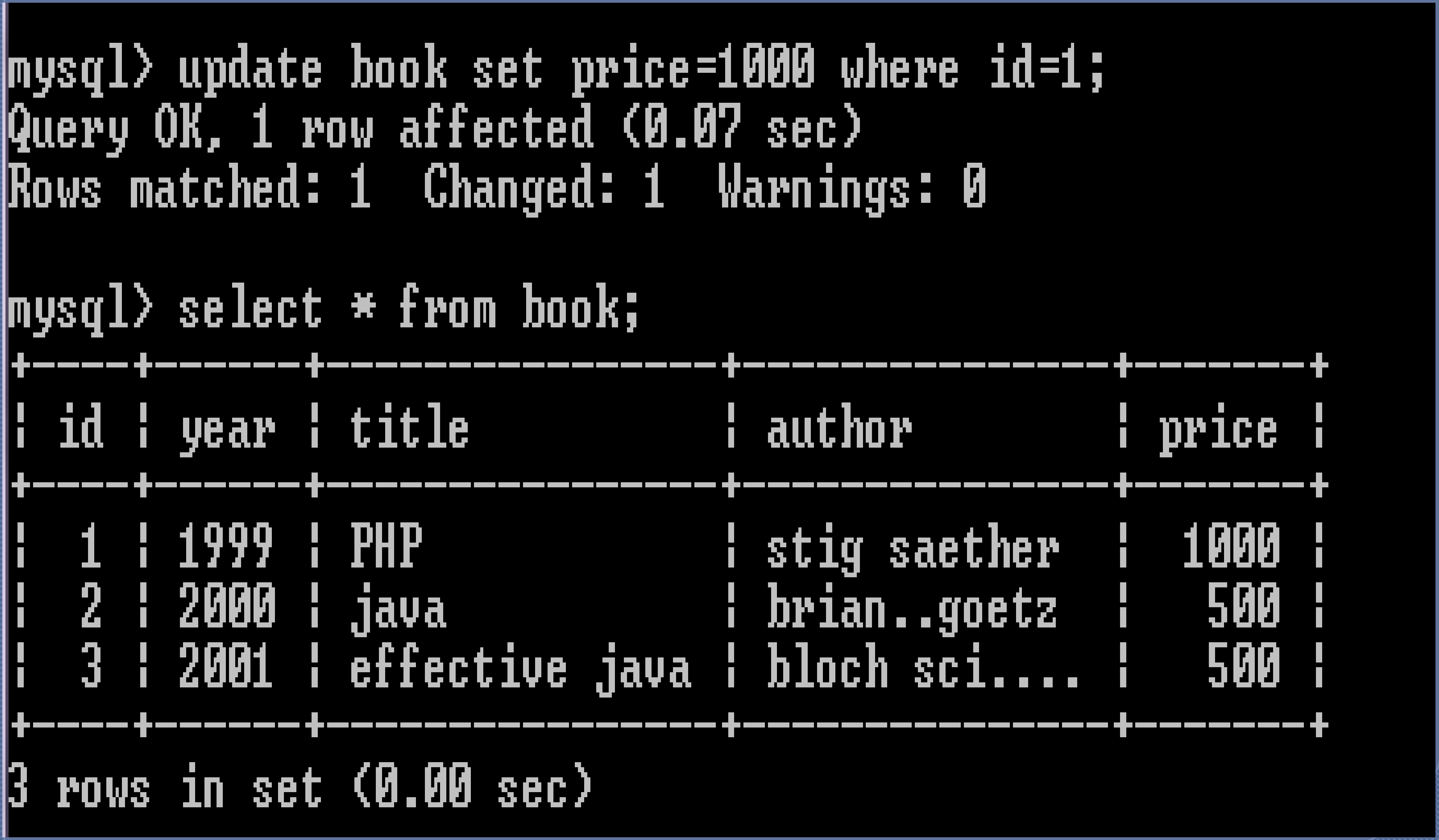
```
mysql> update book set author="bloch sci...." where id=3;
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from book;
+----+------+------------------+----------------+-------+
| id | year | title            | author         | price |
+----+------+------------------+----------------+-------+
|  1 | 1999 | PHP              | stig saether   |   500 |
|  2 | 2000 | java             | brian goetz    |   500 |
|  3 | 2001 | effective java   | bloch sci....  |   500 |
+----+------+------------------+----------------+-------+
3 rows in set (0.00 sec)
```

Table Data - Updating



```
mysql> update book set author="brian..goetz" where title="java";
Query OK, 1 row affected (0.14 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from book;
+----+------+----------------+---------------+-------+
| id | year | title          | author        | price |
+----+------+----------------+---------------+-------+
|  1 | 1999 | PHP            | stig saether  |   500 |
|  2 | 2000 | java           | brian..goetz  |   500 |
|  3 | 2001 | effective java | bloch sci.... |   500 |
+----+------+----------------+---------------+-------+
3 rows in set (0.00 sec)
```

Table Data - Updating



```
mysql> update book set price=1000 where id=1;
Query OK, 1 row affected (0.07 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from book;
+----+------+-----------------+----------------+-------+
| id | year | title           | author         | price |
+----+------+-----------------+----------------+-------+
|  1 | 1999 | PHP             | stig saether   |  1000 |
|  2 | 2000 | java            | brian..goetz   |   500 |
|  3 | 2001 | effective java  | bloch sci....  |   500 |
+----+------+-----------------+----------------+-------+
3 rows in set (0.00 sec)
```

Table Data - Query

```
SELECT  <attributes>
FROM    <one or more relations>
WHERE   <conditions>;
```

Simple SQL Query(select all columns)

```
SELECT  *
FROM    book;
```

book

id	author	title	year
1	Orson Scott card	Ender's game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

id	author	title	year
1	Orson Scott card	Ender's game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

Simple SQL Query

book

id	author	title	year
1	Orson Scott card	Ender’s game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT  *
FROM    book
WHERE   author=‘Orson Scott
card’;
```

id	author	title	year
1	Orson Scott card	Ender’s game	1985
6	Orson Scott card	land	1961

“selection”

Simple SQL Query

book

id	author	title	year
1	Orson Scott card	Ender's game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT  id, author, title
FROM    book
WHERE  year > 1960;
```

id	author	title
1	Orson Scott card	Ender's game
2	Frank Herbert	dune
4	Douglas Adams	galaxy
6	Orson Scott card	land

Selecting Rows with Conditional Restrictions

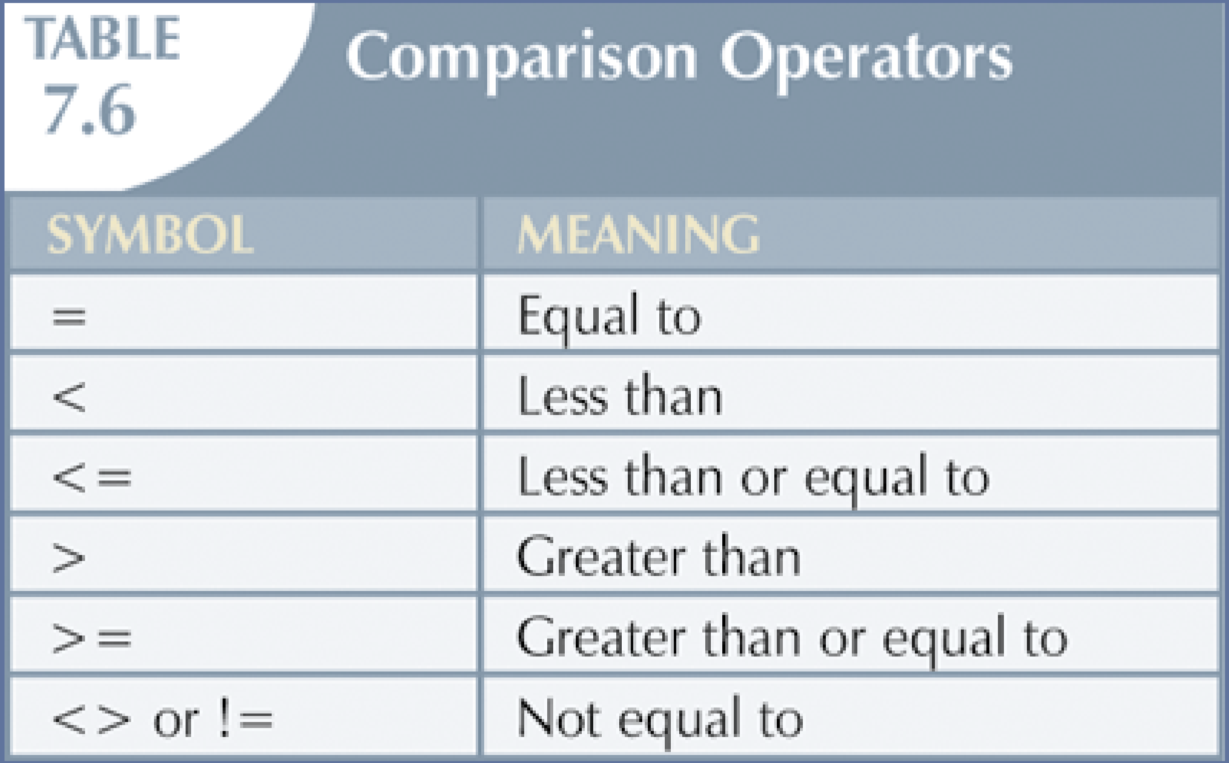


TABLE 7.6 Comparison Operators

SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to

Simple SQL Query

book

id	author	title	year
1	Orson Scott card	Ender's game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT   author, title,year
FROM     book
order by year;
```

author	title	year
George Orwell	1984	1949
Isaac Asimov	foundation	1951
Orson Scott card	land	1961
Frank Herbert	dune	1965
Douglas Adams	galaxy	1979
Orson Scott card	Ender's game	1985

Simple SQL Query

book

id	author	title	year
1	Orson Scott card	Ender's game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT   author, title,year
FROM     book
order by year desc;
```

author	title	year
Orson Scott card	Ender's game	1985
Douglas Adams	galaxy	1979
Frank Herbert	dune	1965
Orson Scott card	land	1961
Isaac Asimov	foundation	1951
George Orwell	1984	1949

Simple SQL Query

book

id	author	title	year
1	Orson Scott card	Ender’s game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT   id, author, title
FROM     book
WHERE  year between 1970 and 1990;
```

id	author	title
1	Orson Scott card	Ender’s game
4	Douglas Adams	galaxy

Simple SQL Query

book

id	author	title	year
1	Orson Scott card	Ender’s game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT   id, author, title
FROM     book
WHERE   year in (1965, 1979, 1985);
```

id	author	title
1	Orson Scott card	Ender’s game
2	Frank Herbert	dune
4	Douglas Adams	galaxy

Values Matching or Not Matching

book

id	author	title	year
1	Orson Scott card	Ender’s game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT   id, author, title
FROM     book
WHERE author like "%Orson%";
```

id	author	title
1	Orson Scott card	Ender’s game
6	Orson Scott card	land

Values Matching or Not Matching

book

id	author	title	year
1	Orson Scott card	Ender's game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT   id, author, title
FROM     book
WHERE author like "%Or%";
```

id	author	title
1	Orson Scott card	Ender's game
5	George Orwell	1984
6	Orson Scott card	land

Values Matching or Not Matching

book

id	author	title	year
1	Orson Scott card	Ender’s game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT  id, author, title
FROM    book
WHERE title like "%____%";
```

id	author	title
2	Frank Herbert	dune
5	George Orwell	1984
6	Orson Scott card	land

Values Matching or Not Matching

book

id	author	title	year
1	Orson Scott card	Ender’s game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
5	George Orwell	1984	1949
6	Orson Scott card	land	1961

```
SELECT  id, author, title
FROM    book
WHERE author !="Orson Scott card";
```

id	author	title
2	Frank Herbert	dune
3	Isaac Asimov	foundation
4	Douglas Adams	galaxy
5	George Orwell	1984

Table Data - Deleting

Records can be deleted from a table using **delete from** command followed by the table name. So, the command **delete from book;** would remove all the records from the **book** table.

```
Delete FROM    book
WHERE id=5;
```

id	author	title	year
1	Orson Scott card	Ender's game	1985
2	Frank Herbert	dune	1965
3	Isaac Asimov	foundation	1951
4	Douglas Adams	galaxy	1979
6	Orson Scott card	land	1961

Table Data - Deleting

Specific columns can be deleted from a table using alter table command with drop keyword.

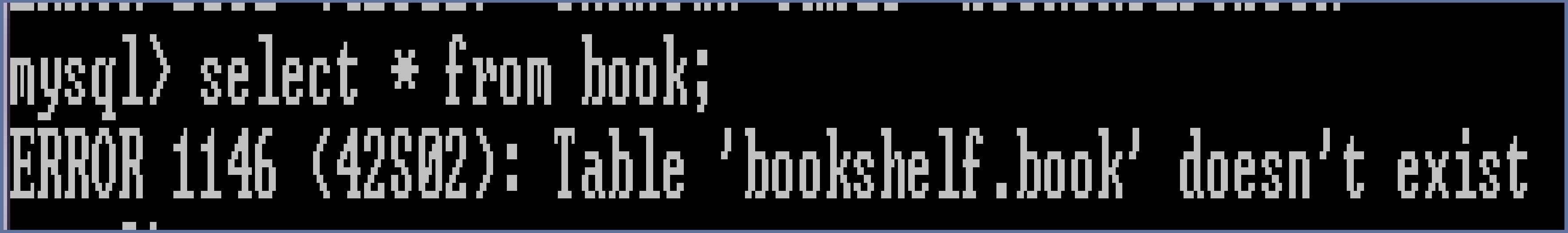
```
Alter table book
drop year;
```

id	author	title
1	Orson Scott card	Ender’s game
2	Frank Herbert	dune
3	Isaac Asimov	foundation
4	Douglas Adams	galaxy
6	Orson Scott card	land

Table - Deleting

The whole table can be deleted from a database using **drop table** command.

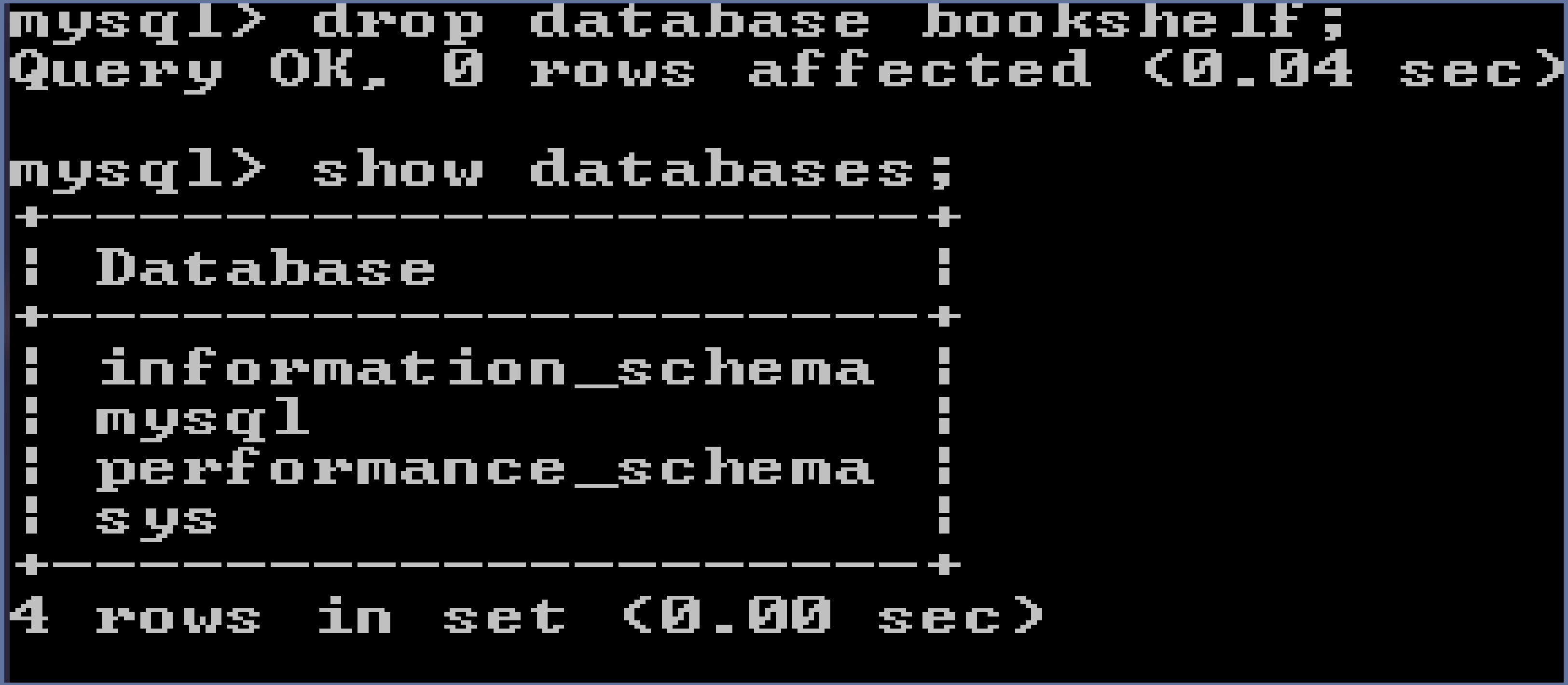
```
drop table book;
```



```
mysql> select * from book;
ERROR 1146 (42S02): Table 'bookshelf.book' doesn't exist
```

database - Deleting

```
drop database bookshelf;
```



```
mysql> drop database bookshelf;
Query OK, 0 rows affected (0.04 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

Setting UP Users and Privileges

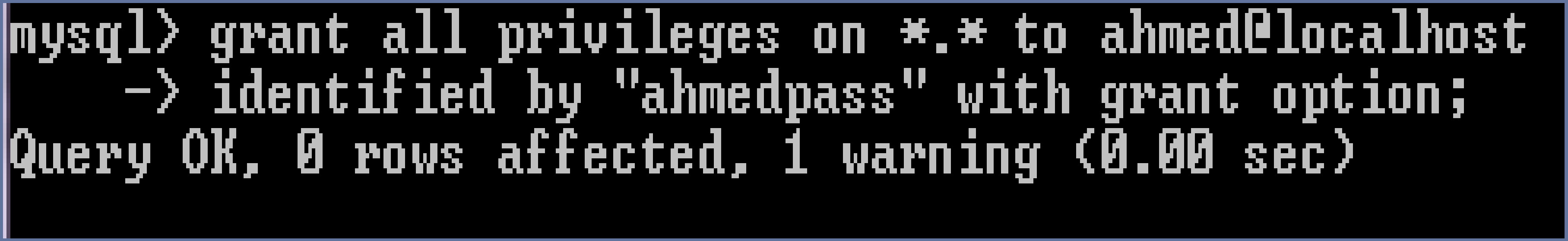
- A MySQL system can have many users. The **root** should be used.

- One of the best features of MySQL is that it supports a sophisticated privilege system. Aprivilege is the right to perform a particular action on a particular object and is associated with a particular user. The concept of privilege is similar to file permission. When we create a user within MySQL, we grant the user a set of privileges to specify what the user can and cannot do within the system. for administration purposes only for security reasons.

Setting UP Users and Privileges

- The **GRANT** and **REVOKE** commands enable you to give rights to and take them from MySQL users at these four levels of privilege: (Global, Database, Table, Column).

- We can add a user by entering a **grant** statement into MySQL monitor as the root user.



```
mysql> grant all privileges on *.* to ahmed@localhost
    -> identified by "ahmedpass" with grant option;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

This will create a new user with **root** status. The new user **ahmed** was created in the **localhost** domain with a password **ahmedpass**.

Privileges for Users

Privilege	Column	Context
CREATE	Create_priv	databases, tables, or indexes
DROP	Drop_priv	databases, tables, or views
GRANT OPTION	Grant_priv	databases, tables, or stored routines
REFERENCES	References_priv	databases or tables
ALTER	Alter_priv	tables
DELETE	Delete_priv	tables
Index	Index_priv	tables
INSERT	Insert_priv	tables or columns
SELECT	Select_priv	tables or columns
UPDATE	Update_priv	tables or columns

Privileges for Users

Privilege	Column	Context
CREATE TEMPORARY TABLES	Create_tmp_table_priv	tables
LOCK TABLES	Lock_tables_priv	tables
CREATE VIEW	Create_view_priv	views
SHOW VIEW	Show_view_priv	views
ALTER ROUTINE	Alter_routine_priv	stored routines
CREATE ROUTINE	Create_routine_priv	stored routines
EXECUTE	Execute_priv	stored routines
FILE	File_priv	file access on server host
CREATE USER	Create_user_priv	server administration
PROCESS	Process_priv	server administration

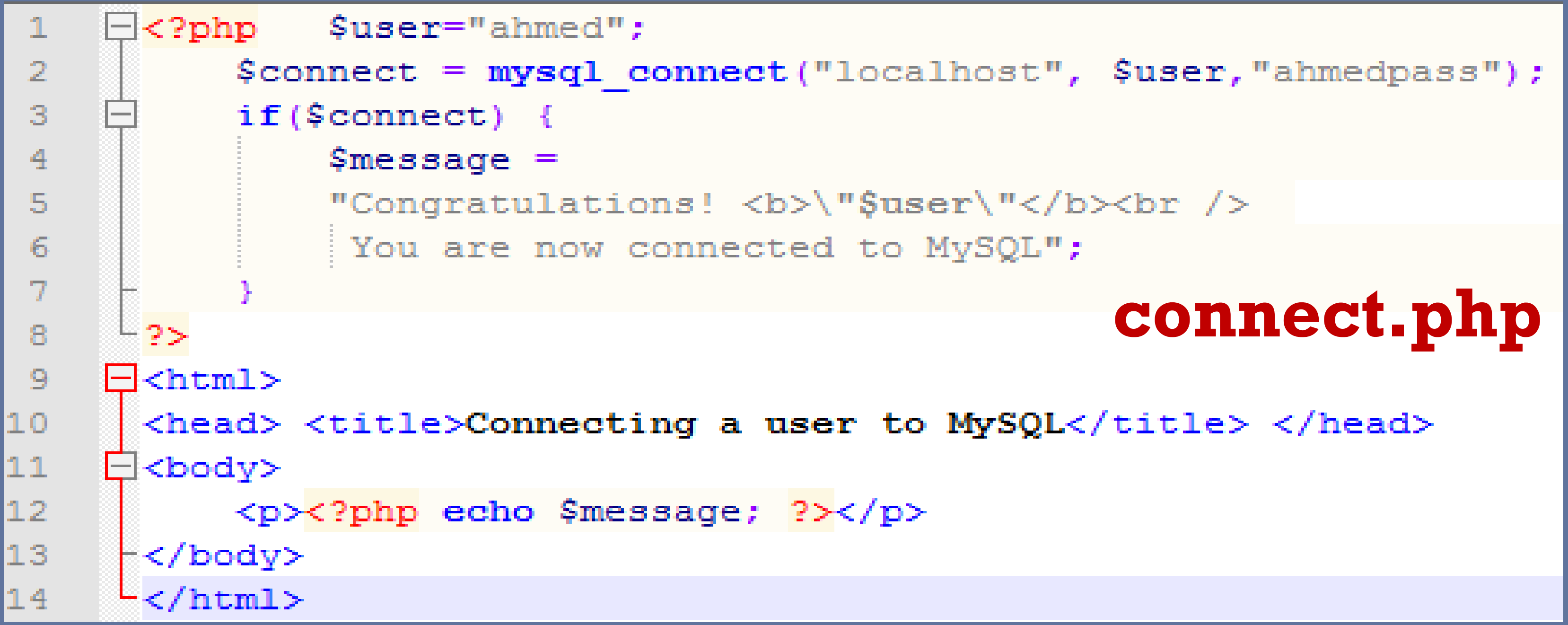
Privileges for Users

Privilege	Column	Context
RELOAD	Reload_priv	server administration
REPLICATION CLIENT	Repl_client_priv	server administration
REPLICATION SLAVE	Repl_slave_priv	server administration
SHOW DATABASES	Show_db_priv	server administration
SHUTDOWN	Shutdown_priv	server administration
SUPER	Super_priv	server administration
ALL[PRIVILEGES]	-	server administration
USAGE	-	server administration
RELOAD	Reload_priv	server administration

Connection to MySQL

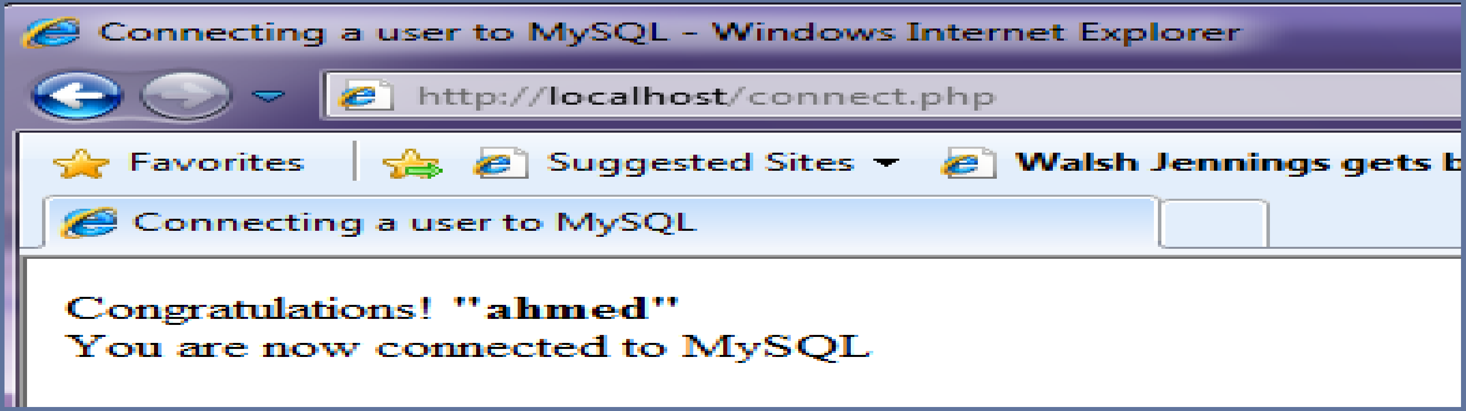
- To connect to MySQL, we use **mysql_connect()** function. This function requires three arguments: domain name, user name, and password. It returns true at successful connection. The example below shows how to write a confirmation after the successful connection.

Connection to MySQL



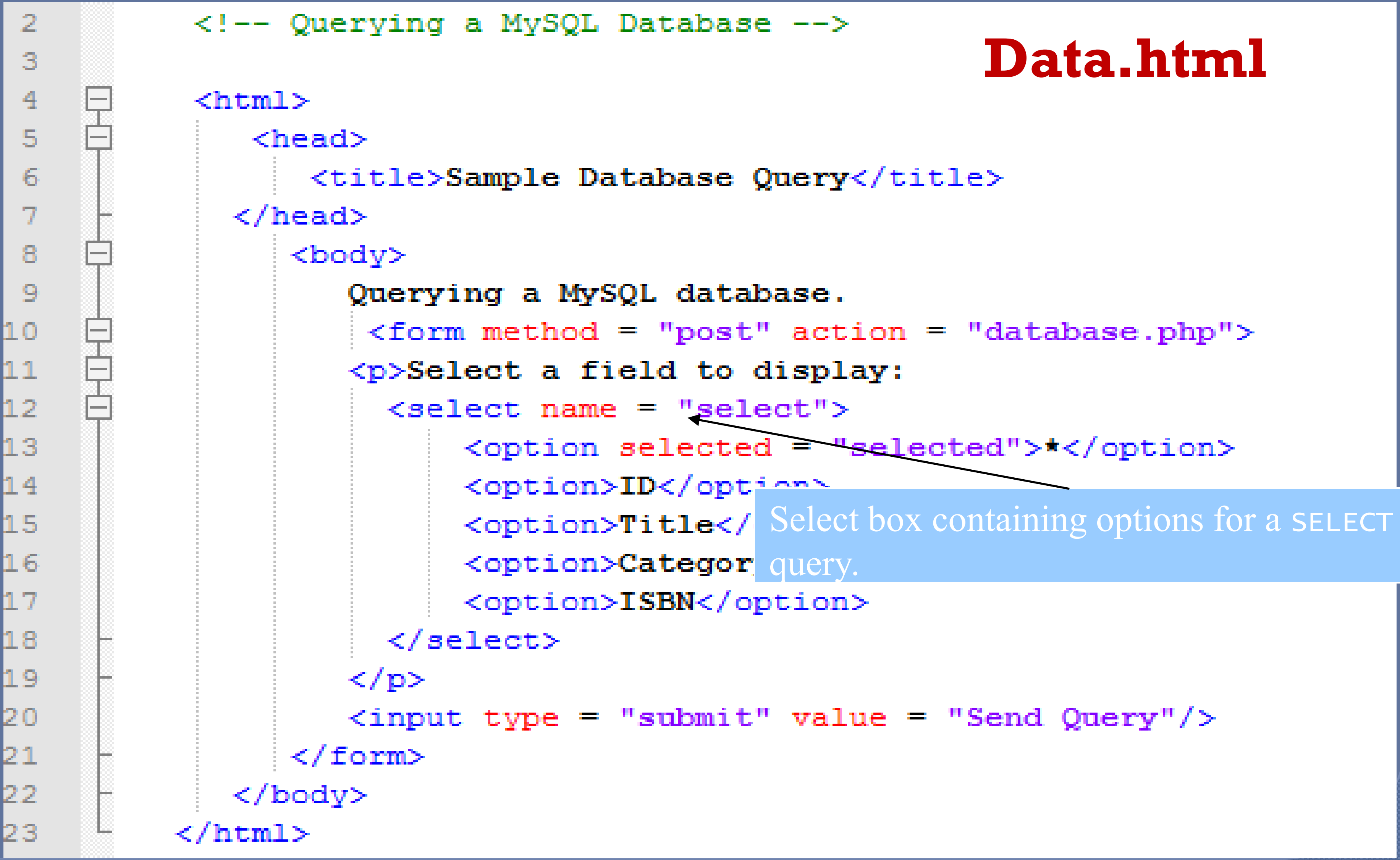
connect.php

```
<?php    $user="ahmed";
    $connect = mysql_connect("localhost", $user,"ahmedpass");
    if($connect) {
        $message =
        "Congratulations! <b>\"$user\"</b><br />
         You are now connected to MySQL";
    }
?>
<html>
<head> <title>Connecting a user to MySQL</title> </head>
<body>
    <p><?php echo $message; ?></p>
</body>
</html>
```


Connecting a user to MySQL - Windows Internet Explorer
http://localhost/connect.php
Favorites
Suggested Sites
Walsh Jennings gets b
Connecting a user to MySQL

```
Congratulations! "ahmed"
You are now connected to MySQL
```

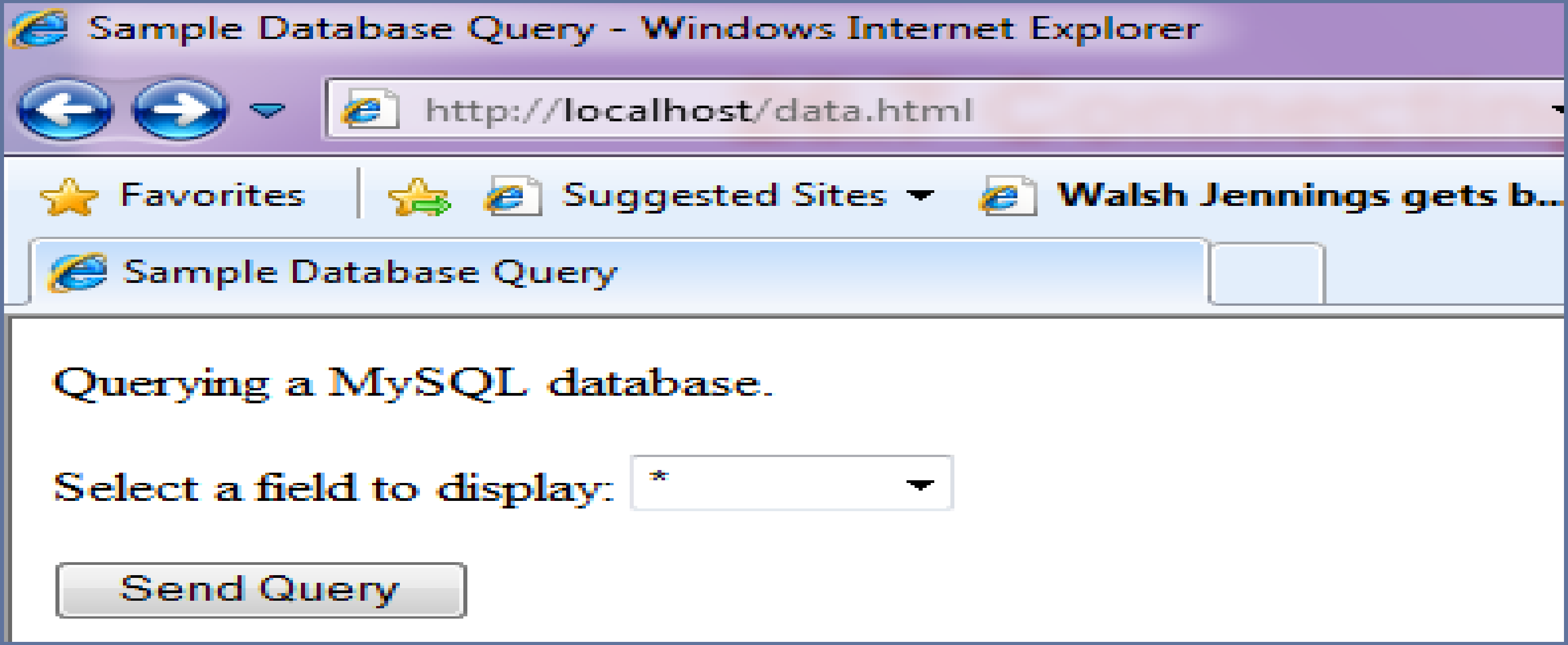
PHP & SQL



Data.html

```
<!-- Querying a MySQL Database -->

<html>
   <head>
      <title>Sample Database Query</title>
  </head>
    <body>
       Querying a MySQL database.
        <form method = "post" action = "database.php">
       <p>Select a field to display:
         <select name = "select">
             <option selected = "selected">*</option>
             <option>ID</opti
             <option>Title</
             <option>Categor
             <option>ISBN</option>
         </select>
       </p>
       <input type = "submit" value = "Send Query"/>
    </form>
  </body>
</html>
```

Select box containing options for a SELECT query.


Sample Database Query - Windows Internet Explorer
http://localhost/data.html
Favorites
Suggested Sites
Walsh Jennings gets b...
Sample Database Query

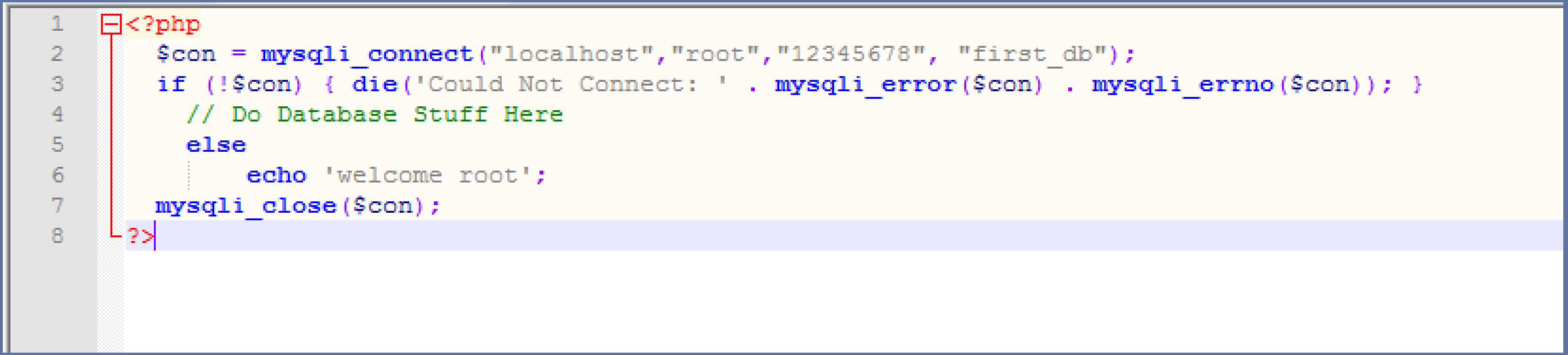
Querying a MySQL database.

Select a field to display: *

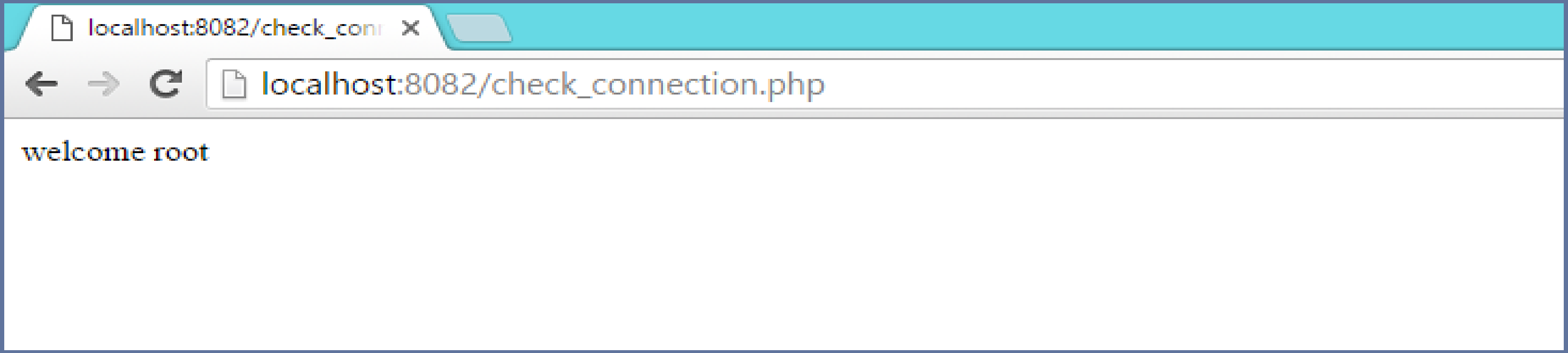
Function	Description
mysqli_affected_rows()	Returns the number of affected rows in the previous MySQL operation
mysqli_close()	Closes a previously opened database connection
mysqli_connect()	Opens a new connection to the MySQL server
mysqli_errno()	Returns the last error code for the most recent function call
mysqli_error()	Returns the last error description for the most recent function call
mysqli_fetch_all()	Fetches all result rows as an associative array, a numeric array, or both
mysqli_fetch_array()	Fetches a result row as an associative, a numeric array, or both
mysqli_fetch_assoc()	Fetches a result row as an associative array
mysqli_fetch_row()	Fetches one row from a result-set and returns it as an enumerated array
mysqli_free_result()	Frees the memory associated with a result
mysqli_num_rows()	Returns the number of rows in a result set
mysqli_query()	Performs a query against the database
mysqli_real_escape_string()	Escapes special characters in a string for use in an SQL statement
mysqli_select_db()	Changes the default database for the connection

MYSQL Connect & Close

- The **mysqli_connect()** function is used to connect. It requires four parameters, in the following order:
  mysqli_connect(servername, username, password, databasename)



```php
<?php
  $con = mysqli_connect("localhost","root","12345678", "first_db");
  if (!$con) { die('Could Not Connect: ' . mysqli_error($con) . mysqli_errno($con)); }
    // Do Database Stuff Here
    else
        echo 'welcome root';
  mysqli_close($con);
?>
```

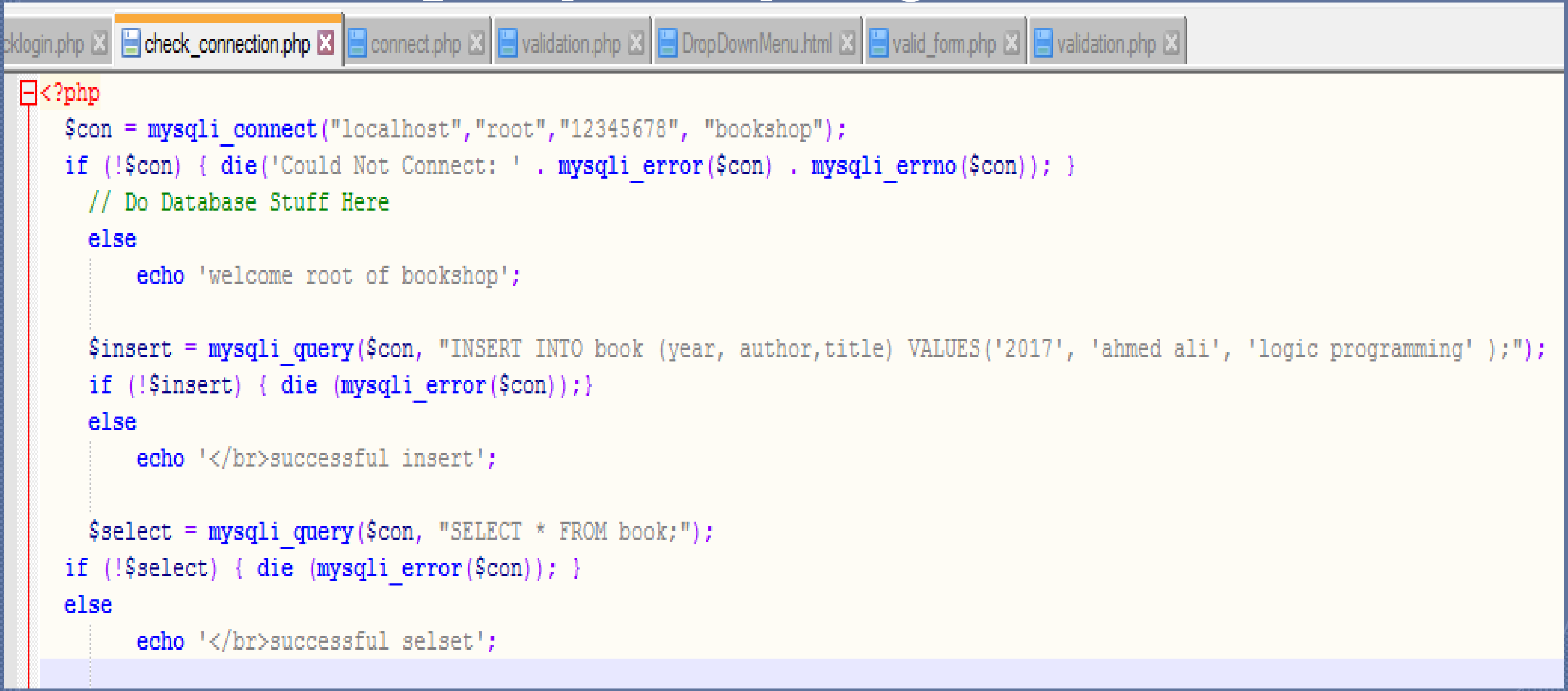


localhost:8082/check_connection.php

```
welcome root
```

Running MYSQL Queries In PHP

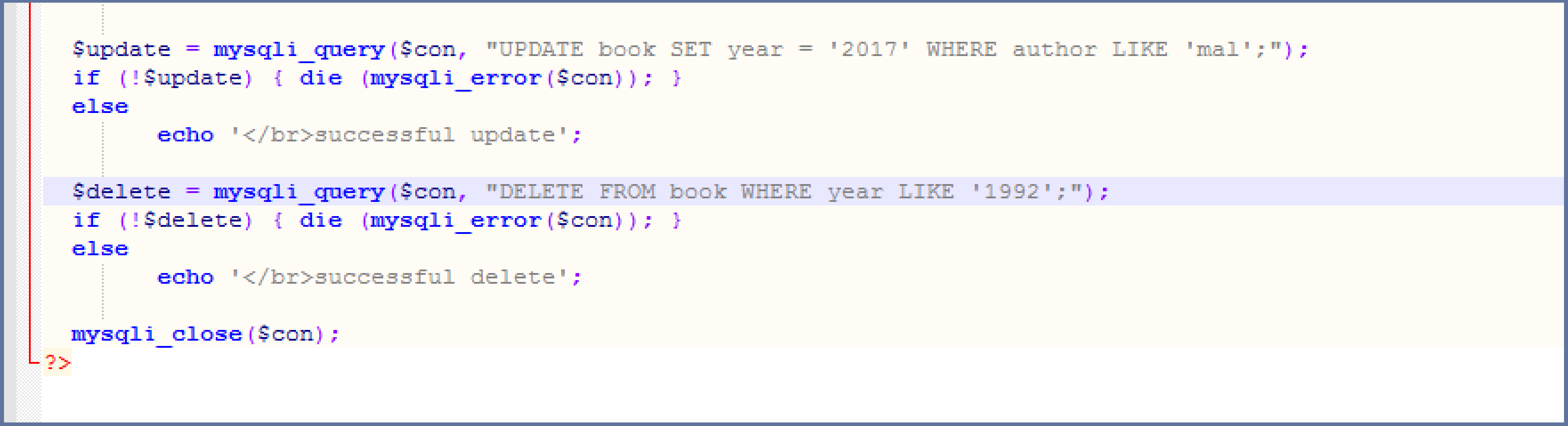
- **mysqli_query()** function can run any MYSQL query that you give it



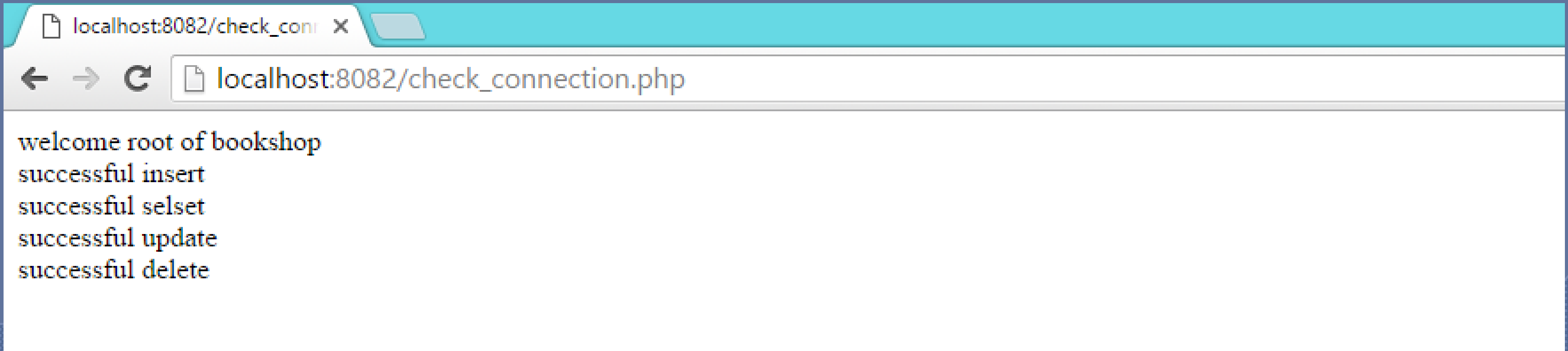
```
<?php
  $con = mysqli_connect("localhost","root","12345678", "bookshop");
  if (!$con) { die('Could Not Connect: ' . mysqli_error($con) . mysqli_errno($con)); }
    // Do Database Stuff Here
    else
        echo 'welcome root of bookshop';

    $insert = mysqli_query($con, "INSERT INTO book (year, author,title) VALUES('2017', 'ahmed ali', 'logic programming' );");
    if (!$insert) { die (mysqli_error($con));}
    else
        echo '</br>successful insert';

    $select = mysqli_query($con, "SELECT * FROM book;");
  if (!$select) { die (mysqli_error($con)); }
  else
        echo '</br>successful selset';
```

Running MYSQL Queries In PHP

- **Die()** print an error message and exit if errors occurred during execution of the statement.



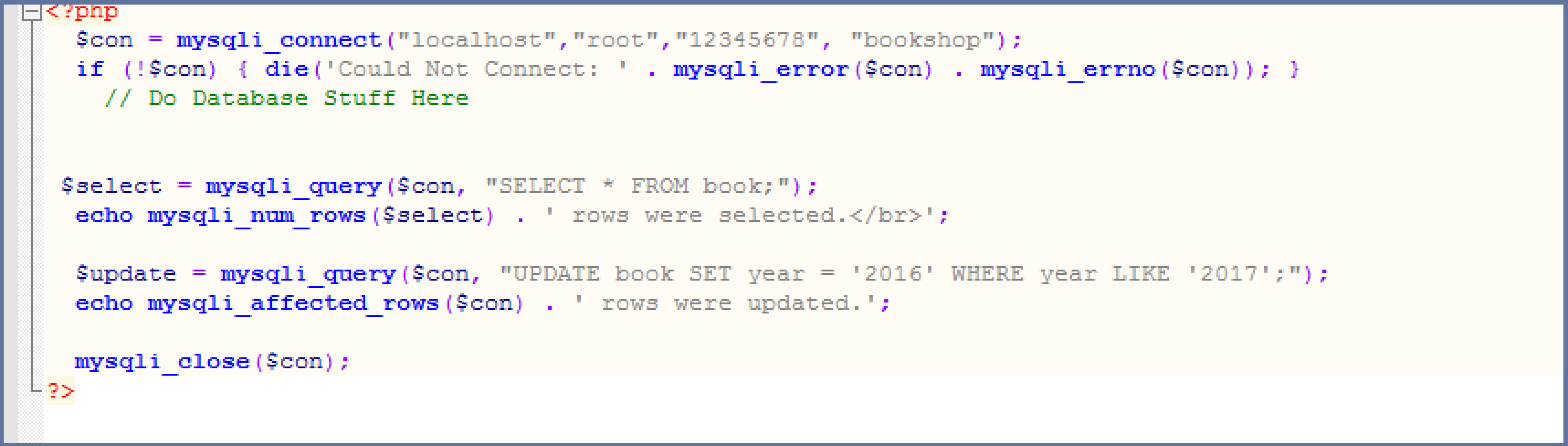
```
$update = mysqli_query($con, "UPDATE book SET year = '2017' WHERE author LIKE 'mal';");
if (!$update) { die (mysqli_error($con)); }
else
     echo '</br>successful update';

$delete = mysqli_query($con, "DELETE FROM book WHERE year LIKE '1992';");
if (!$delete) { die (mysqli_error($con)); }
else
     echo '</br>successful delete';

mysqli_close($con);
?>
```



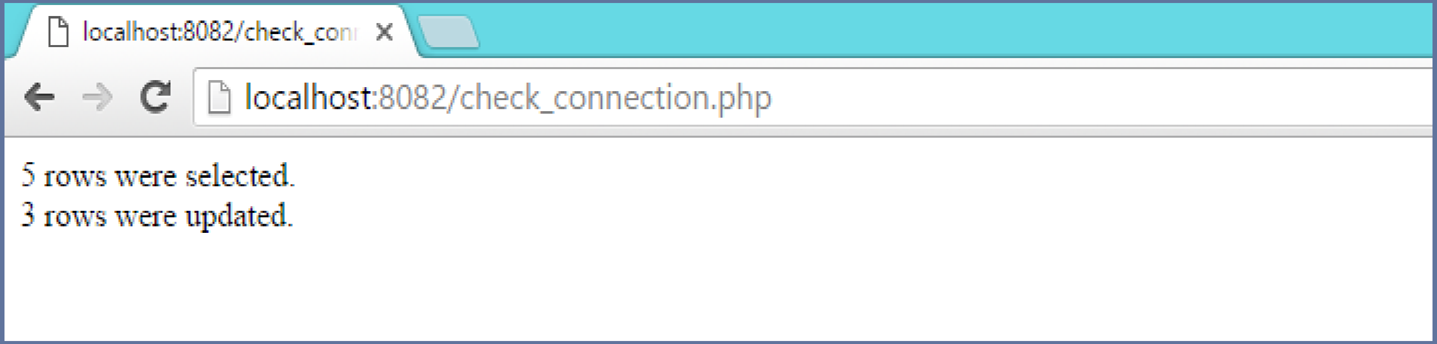
```
welcome root of bookshop
successful insert
successful selset
successful update
successful delete
```

Handling MYSQL Query Results In PHP

- **mysqli_num_rows()** return the number of rows that will be returned of SELECT or SHOW statement.
- **mysqli_affected_rows()** return how many rows were affected by the execution of the last statement of *INSERT*, *UPDATE*, *REPLACE* or *DELETE* query . For SELECT statements it work as **mysqli_num_rows().**



```
<?php
  $con = mysqli_connect("localhost","root","12345678", "bookshop");
  if (!$con) { die('Could Not Connect: ' . mysqli_error($con) . mysqli_errno($con)); }
    // Do Database Stuff Here


 $select = mysqli_query($con, "SELECT * FROM book;");
  echo mysqli_num_rows($select) . ' rows were selected.</br>';

  $update = mysqli_query($con, "UPDATE book SET year = '2016' WHERE year LIKE '2017';");
  echo mysqli_affected_rows($con) . ' rows were updated.';

  mysqli_close($con);
?>
```

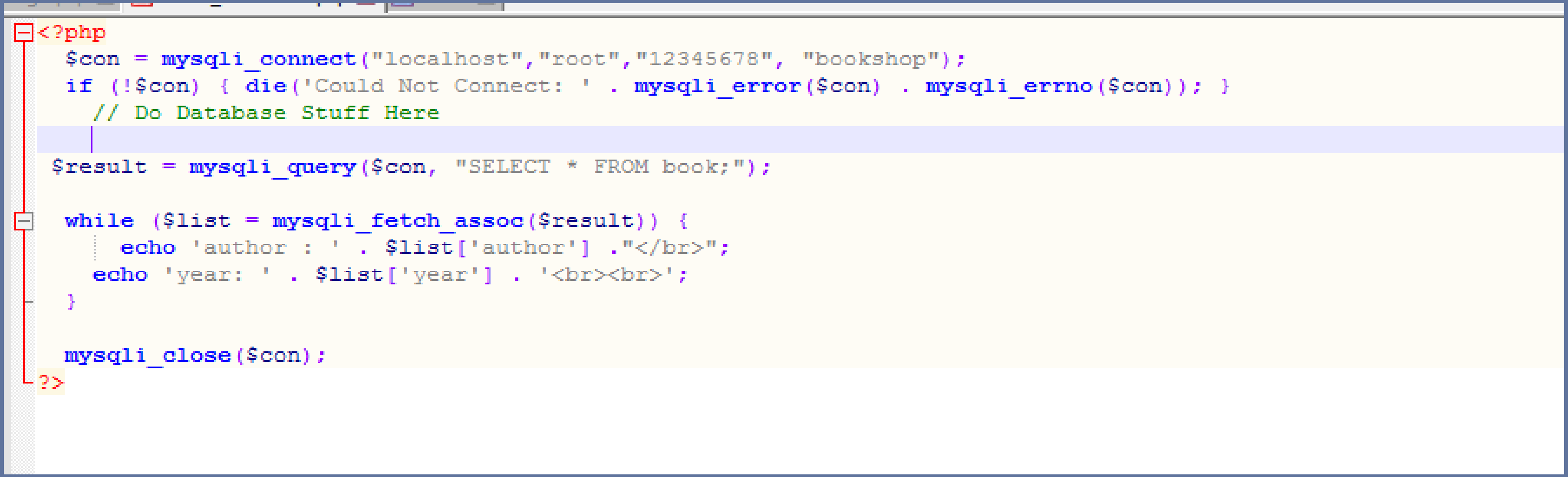


localhost:8082/check_conn
localhost:8082/check_connection.php

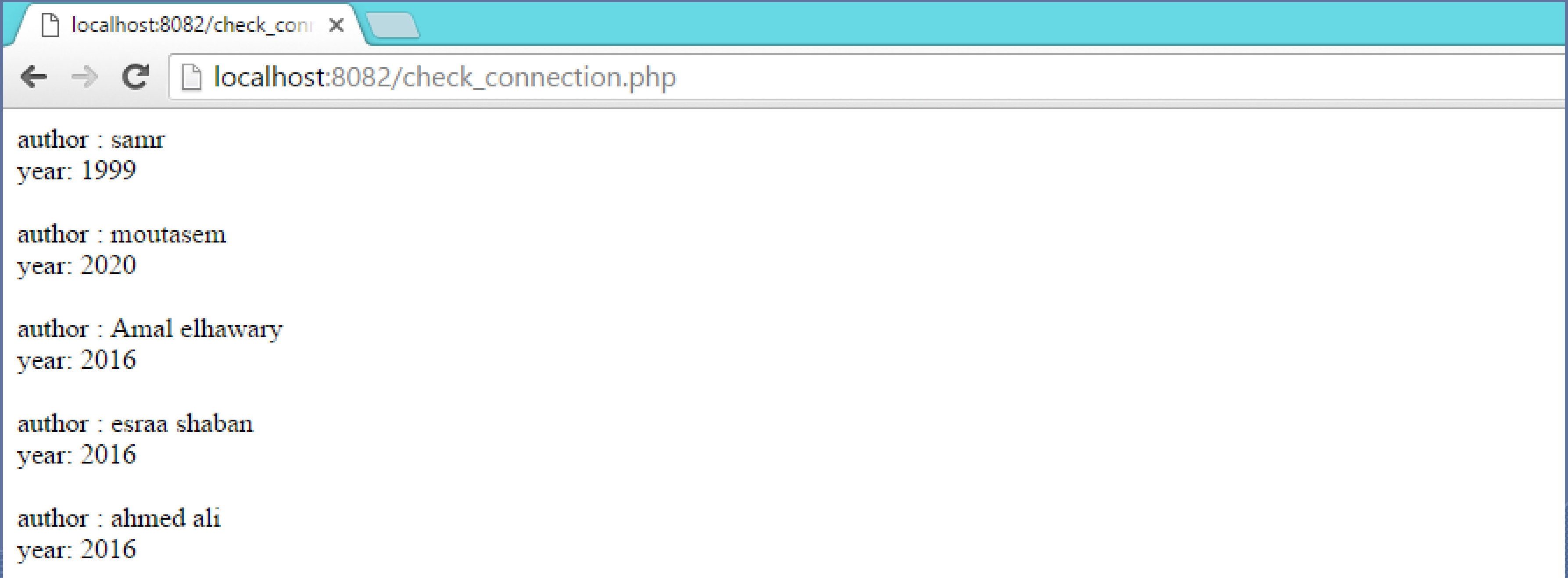
```
5 rows were selected.
3 rows were updated.
```

- mysqli_fetch_assoc() function, return an array which is 'associated' with the name of each column of the database.

- **mysqli_fetch_array()** , return an array which is 'associative' or numberic



```php
<?php
  $con = mysqli_connect("localhost","root","12345678", "bookshop");
  if (!$con) { die('Could Not Connect: ' . mysqli_error($con) . mysqli_errno($con)); }
    // Do Database Stuff Here

 $result = mysqli_query($con, "SELECT * FROM book;");

  while ($list = mysqli_fetch_assoc($result)) {
      echo 'author : ' . $list['author'] ."</br>";
    echo 'year: ' . $list['year'] . '<br><br>';
  }

  mysqli_close($con);
?>
```

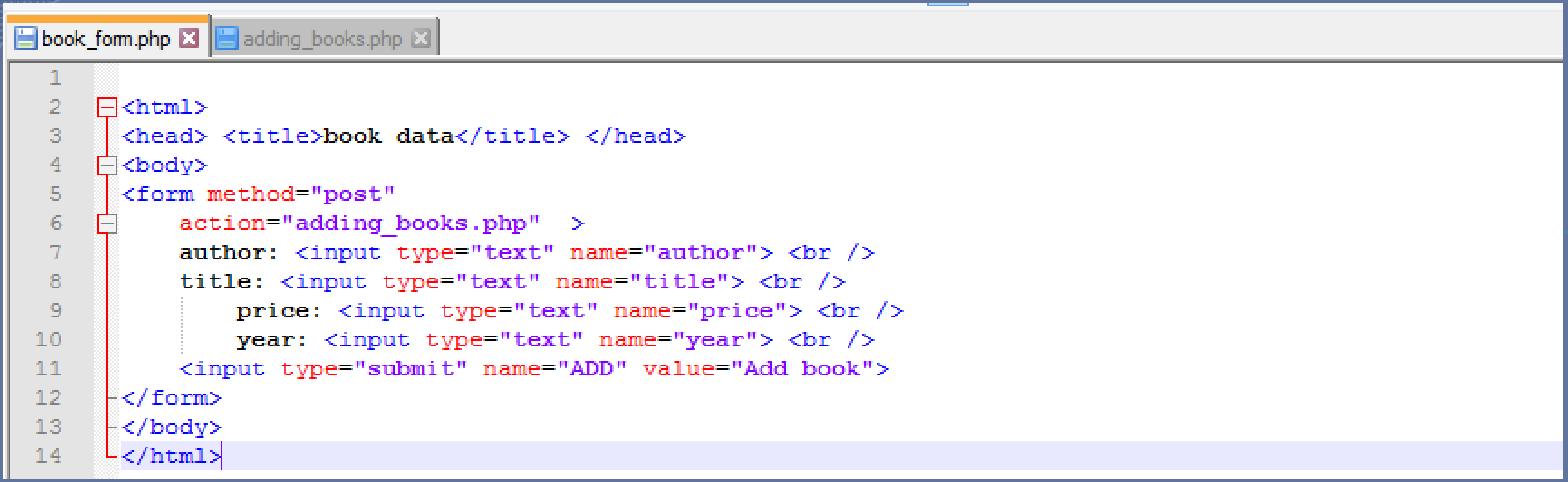


localhost:8082/check_connection.php

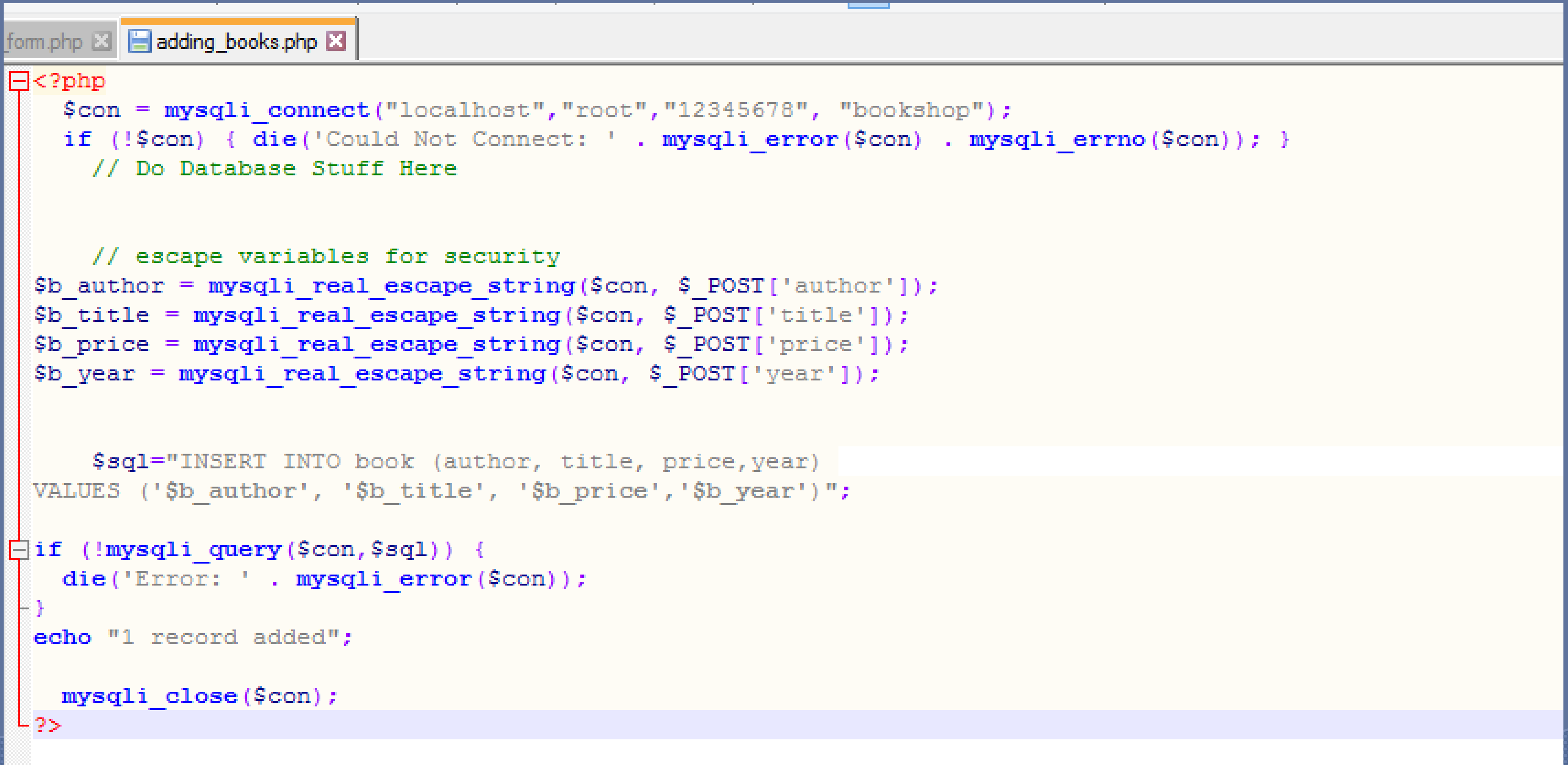
```
author : samr
year: 1999

author : moutasem
year: 2020

author : Amal elhawary
year: 2016

author : esraa shaban
year: 2016

author : ahmed ali
year: 2016
```

MYSQL Security & Handling User Input

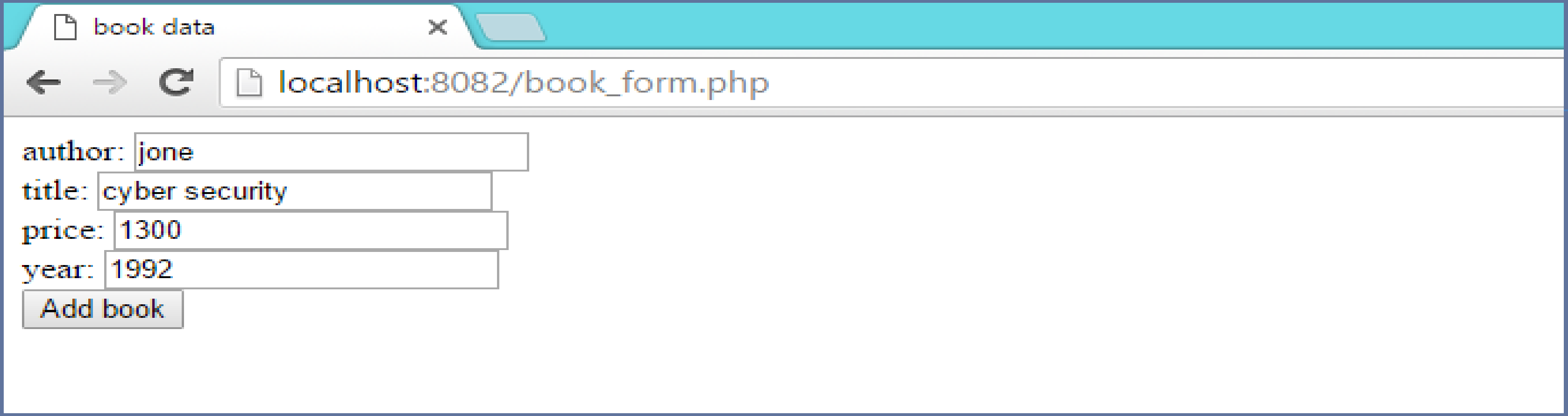
- **mysqli_real_escape_string()** function should always be used when entering data into a MYSQL query.
- escape any characters that may cause the query to be used maliciously.
  - → **'SQL injection attack'**


book_form.php
adding_books.php

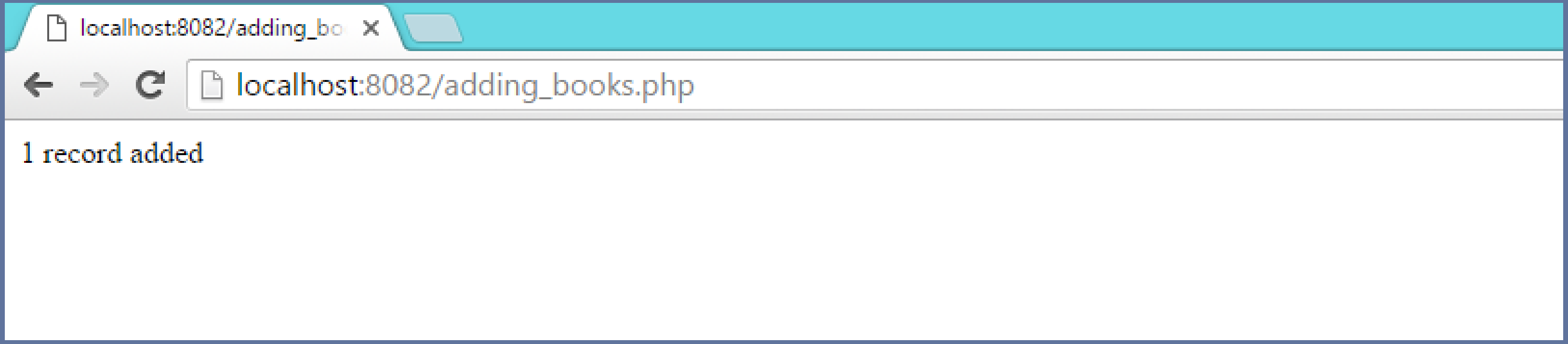
```
<html>
<head> <title>book data</title> </head>
<body>
<form method="post"
    action="adding_books.php"  >
    author: <input type="text" name="author"> <br />
    title: <input type="text" name="title"> <br />
        price: <input type="text" name="price"> <br />
        year: <input type="text" name="year"> <br />
    <input type="submit" name="ADD" value="Add book">
</form>
</body>
</html>
```


form.php
adding_books.php

```
<?php
  $con = mysqli_connect("localhost","root","12345678", "bookshop");
  if (!$con) { die('Could Not Connect: ' . mysqli_error($con) . mysqli_errno($con)); }
    // Do Database Stuff Here


    // escape variables for security
$b_author = mysqli_real_escape_string($con, $_POST['author']);
$b_title = mysqli_real_escape_string($con, $_POST['title']);
$b_price = mysqli_real_escape_string($con, $_POST['price']);
$b_year = mysqli_real_escape_string($con, $_POST['year']);


    $sql="INSERT INTO book (author, title, price,year)
VALUES ('$b_author', '$b_title', '$b_price','$b_year')";

if (!mysqli_query($con,$sql)) {
  die('Error: ' . mysqli_error($con));
}
echo "1 record added";

  mysqli_close($con);
?>
```



author: jone
title: cyber security
price: 1300
year: 1992
Add book



```
1 record added
```