# Web Worksheet 10: Modiy server.py to use a Database

## References:

* + <https://www.tutorialspoint.com/sqlite/sqlite_python.htm>
  + Using Python with Sqlite3: <https://docs.python.org/3/library/sqlite3.html>

**Objectives:**

* Create a database and table for storing products details
* Using Python code to retrieve data from database and display them on a web page
* Using Python code to insert new records into the database

**Pre-requistes for Lab:**

* Your completed solution for Worksheet 7.
* A copy of the completed solution is in the folder 10\_ProductApp

**Part 1 : Make a copy of the solutiion from Worksheet 7 and use the copy for the following tasks.**

**Part 2 Import the contents of Products.txt into a Database using DB Browser.**

1. Usng the DB Browser, create a database named **catalogue.db** and import the products.txt file from your solution to Worksheet 7 into the database.
2. The table name should be named **Products**
3. The attributes for the table should be **Name, Description, Price and Image**.

**Part 3: Read the product records from the database and render them using jinja template**

1. Open the Python file, server.py from the ProductsApp folder.
2. Comment out the lines that read from the Products.txt.
3. Add the following boiler-plate helper function code.

| 1  2  3  4  5  6 | ## DB boiler plates \*\*  **import** sqlite3  **def** open\_DB(db):  connection**=**sqlite3.connect(db)  connection.row\_factory **=** sqlite3.Row  **return** connection |
| --- | --- |

Line 5 will configure sqlite to return a record from the database as a **dictionary object** instead of a tuple object.

1. In the root function, comment out those lines that read from products.txt. Add the following lines of code to read from the catalogue.db database

| 1  2  3  4  5  6  7  8  9  10 | @app.route(**"/"**)  **def** root():  **#f = open("products.txt","r")**  **#products\_list =[ line.strip().split(",") for line in f]**  try:  con **=** open\_DB(**'catalogue.db'**)  cur**=**con.execute(**"SELECT \* FROM Products"**)  rows**=**cur.**fetchall**() ## returns a list of dictionary objects  finally:  con.close()  #file\_handler.close()  **return** render\_template(**"products.html"**,products**=**rows) |
| --- | --- |
|  |  |

In line 7, instead of using cur as an iterator to iterate over all the records returned from the SELECT statement, a fetchall() method is called.

This will return **all** the records in a list of dictionary objects.

1. Modify the products.html jinja template to render the product record.

row is now a **dictionary object**.

| **{% for row in products %}**  **<div class="panel">**  **<div class="text title">**  **{{ row["Name"]}}**  **</div>**  **<div class="text">**  **{{ row["Description"]}}**  **</div>**  **<div class="text price">**  **${{ row["Price"]}}**  **</div>**  **<img src="/static/images/{{row['Image']}}" />**  **</div>**  **{% endfor%}** |
| --- |

1. Test your landing page. You should be able to see all the product records from the Prodicts table render as html.
2. Next, modify the form\_submit function so that it will insert a new product into the Products table in the catalogue datbase instead of appending a line in the products.txt file