10.0 hours
Project Status Ready for Submission

Project 4

Store Inventory

Are you ready to have your project reviewed? Before you submit your project, make sure you have double-checked your work. We want you to pass the first time!

If you get your project back and it needs work, you'll have to wait 1 day before you can resubmit your project.

Submit for Review

- Instructions
- How you'll be graded

Use your knowledge of CSV and File I/O and database ORMs to build a console application that allows you to easily interact with data for a store's inventory. The data needs to be cleaned from the CSV before it is to be persisted in the database. All interactions with the records should use ORM methods for viewing records, creating records, and exporting a new CSV backup.

Before you start

To prepare for this project you'll need to make sure you complete and understand these steps.

3 steps

- **Review the policy on Reusing Code in a Techdegree project.**
- **Read the instructions carefully**

  Take your time and read through the instructions and the grading criteria section. We suggest twice, at least.

- **GitHub**

  - Create a new repo for this project.
  - Create a `README.md` file for your repo that explains what the project is and anything your user or fellow developers might need to know to use the project.

Project Instructions

To complete this project, follow the instructions below. If you get stuck, ask a question on Slack or in the Treehouse Community.

13 steps

- **Create a new virtual Python environment**

  Create a virtual environment in your project's directory by running the following command in your terminal:

  ```
  python3 -m venv env
  ```

  NOTE: The environment *(./env)* folder and files pertaining to your virtual environment do not need to be added to your github repo.

- **Activate your new virtual Python environment**

  Activate the virtual environment by running the following command in your terminal for students using macs:

  If using **Mac/Linux**:

  ```
  source ./env/bin/activate
  ```

  If using **Windows**:

  ```
  .\env\Scripts\activate
  ```

- **Install required dependencies into your Python environment**

  Install the project requirements from the provided `requirements.txt` file by running the following command in your terminal:

  ```
  pip3 install -r requirements.txt
  ```

- **Create your application file**

  Create a new file in your project directory called `app.py`. Be sure to import the appropriate Python and Peewee modules at the top of this file.

- **Read in the existing CSV data**

  Read the `inventory.csv` file into your program, and create a list that contains each product inside the csv file as a dictionary. Be sure to clean up the data before adding each product dictionary to your list:

  - the value for `product_quantity` will be stored as an integer
  - the value for `product_price` will be stored as an integer and converted to cents ($3.19 becomes 319, for example)
  - the value for `date_updated` will be stored as a date.
  - *Hint:* You'll need the `datetime` module for this.

- **Initialize your Sqlite database**

  Initialize an Sqlite database called `inventory.db`.

- **Create your Product model**

Create a model called Product that the Peewee ORM will use to build the database. The Product model should have five attributes: `product_id`, `product_name`, `product_quantity`, `product_price`. Use PeeWee's built in `primary_key` functionality for the `product_id` field, so that each product will have an automatically generated unique identifier.

- **Add the data from CSV into the database**

  Create a function that will add the products listed in the `inventory.csv` file to the database.

- **Create a Menu to make selections**

  Create a function to handle interaction with the user of your app. This function should prompt the user to enter `v` in order to view the details of a single product in the database, `a` to add a new product to the database, or `b` to make a backup of the entire contents of the database.

- **Displaying a product by its ID**

  Create a function to handle getting and displaying a product by its `product_id`.

- **Adding a new product to the database**

  Create a function to handle adding a new product to the database. This function should prompt the user to enter the product's name, quantity, and price. The function must process the user provided value for price from a string to an int. Be sure the value you stored for the price field to the database is converted to cents ($2.99 becomes 299, for example).

- **Backup the database (Export new CSV)**

  Create a function to handle making a backup of the database. The backup should be written to a `.csv` file.

- **Connect the database and create tables**

  In your dunder main method:

  1. Ensure you are connected to the database you created/initialized
  2. Ensure you load the CSV products data into the created table
  3. Run the application so the user can make menu choices and interact with the application.

Extra Credit

To get an "exceeds" rating, complete all of the steps below:

5 steps

- **Avoiding duplicate products**

When adding product data into the database, add an additional check before that product is added so that if a duplicate product name is found, the app will save the data that was most recently updated for that existing record.

- **Menu validation**

  When entering input for the menu selection feature, add extra validation that ensures if the User enters any other character besides `v`, `a`, or `b` it will notify them of their error and re-prompts them again for their menu input.

- **Menu option: v**

  When selecting option `v`, If the User enters a product ID that does not exist, a human readable error message will be displayed, and the user will be prompted to try again.

- **Menu option: a**

  When selecting option `a`, if a duplicate product name is found while the product is attempting to be added to the database, the app will check to see which product entry was most recently updated and only save that data.

- **Menu option: b**

  When selecting option `b`, the backup CSV outpu file should contain a single header row with all the appropriate field titles.

- **NOTE: Getting an "Exceed Expectations" grade.**
  - See the rubric in the "**How You'll Be Graded**" tab above for details on what you need to receive an "Exceed Expectations" grade.
  - Passing grades are final. If you try for the "Exceeds Expectations" grade, but miss an item and receive a “Meets Expectations” grade, you won’t get a second chance. Exceptions can be made for items that have been misgraded in review.
  - Always mention in the comments of your submission or any resubmission, what grade you are going for. Some students want their project to be rejected if they do not meet all Exceeds Expectations Requirements, others will try for all the "exceeds" requirement but do not mind if they pass with a Meets Expectations grade. Leaving a comment in your submission will help the reviewer understand which grade you are specifically going for

Download files

Zip file

Project Resources

Workshop

Python File I/O

Workshop

CSV and JSON in Python

Course

Using Databases in Python

Need Help?

Have questions about this project? Start a discussion with the community and Treehouse staff.

Get Help