PA02 - finance tracker - using SQL and pytest

Motivation

Many software projects use SQLite to manage their data and this problem set will give you the experience of building such an app. Another important process in software engineering is the design of automated tests. This assignment will ask you to develop a suite of tests for your app. There are other database and testing frameworks, but they are all similar in principle and this assignment will expose you to the core concepts and skills you'll need.

Learning Objectives -

* to write SQL queries to perform the CRUD operations (Create, Read, Update, Delete) and aggregation(with SQLite3)
* to develop automated testing (with pytest)

Steps:

1. **Download the starter code** - download the pa02 branch for the cs103a github site
2. **create a git repository** containing the pa02 code, which you then push to github
3. **create your team** add all members of the team as collaborators
4. **create a Python class Transaction** in a new file transactions.py which will store financial transactions with the fields. It should have an __init__ method where you pass in the filename for the database to be used (e.g. tracker.db) and each transaction should have the following fields stored in a SQL table called transactions.
   The transaction class should not do any printing!!
   `'item #','amount','category','date','description'`
5. **start adding features** to the user interface code, tracker.py, which will make calls to the Transaction class to add transactions, find transactions, delete transactions, summarize transactions, etc. Here is the list of all operations. The first 4 (0,1,2,3) have been implemenrted already, you need to complete the remaining actions. The tracker.py program should do all of its database operations by making calls to the Transaction class methods. You may want to add some additional features so that everyone gets an opportunity to implement a feature.
   The tracker.py program should not have any SQL calls.
   a. 0. quit
   b. 1. show categories
   c. 2. add category
   d. 3. modify category
   e. 4. show transactions
   f. 5. add transaction
   g. 6. delete transaction

        h. 7. summarize transactions by date
        i. 8. summarize transactions by month
        j. 9. summarize transactions by year
        k. 10. summarize transactions by category
        l. 11. print this menu
6. **Testing with pytest** -- start adding tests to a file test_transaction.py for each method in the Transaction class. It is a good idea to add a test each time you implement a feature. You are testing the Transaction class, not the tracker.py user interface code.
7. **Linting with pylint** -- use pylint to eliminate Python style errors.
   Use pylint with transactions.py and tracker.py.
8. **Collaborating with github** -- regularly commit your changes and push them to github, every team member should push some changes of their own
9. **Creating a transcript** - use the "script" command to create a transcript of your session as you demonstrate each of the features you have implemented. You may need to clone your project to tiara to be able to use the script feature.
10. **create a README.md file** which describes your app and contains
    a. a script of you running pylint, and
    b. then running pytest, and
    c. then running tracker.py and demonstrating all of the features you added

What to upload to mastery.cs.brandeis.edu (programs)

- a link to your github
- a reflection where you state what you did **personally** on the project (you should put your name in the comments of each method you personally write....). We can look at the github repository to see what each person did, but its easier if you just tell us!

Rubric

- This will be graded on the basis of a good faith effort.... if your code generally works, then you will get full credit, if not, then you will be asked to revise and resubmit.