MELBOURNE BUSINESS SCHOOL (The University of Melbourne) MBUSA90500 Programming Assignment V1.0

Initial submission before 9:00am 5 July.
Revised submission and tinkering allowed until 9:00am 7th July.
Peer review due 11:59pm 8th July.

1 Objectives

This project is designed to give your syndicate experience in software engineering, Python programming and processing data from different sources. In particular, you should experience the following.

- Python programming - every member of the syndicate must contribute code to the final solution.
- Software design - the process of breaking a task into modules and assigning each module to a programmer in such a way that the pieces can be easily assembled.
- The frustration of unclean data sets.
- Software testing - each module should be tested prior to integration, and then the whole tested once integrated.

2 The problem

You have been hired as a Data Scientist by Ye Olde Web Shoppe (YOWS) to improve their profits. YOWS was one of the first online retailers, and has been selling products online for 16 years. Their proprietor, Caesar, cannot understand this new-fangled analytics thing, but some of his younger staff have convinced him that profits could be improved if some analysis of their customer flow was undertaken.
The company has the following 3 data sets:

1. a Web page (in HTML) that contains their products and prices;
2. a Web-analytics file (in JSON) that contains information about customer interactions with the web site (clicks, hover time, viewing time, customer ID); and
3. a SQL database of stock, customer purchases and shipping expenses (as a CSV file).

Your syndicate has two tasks: create a data set for YOWS; and perform analysis on your own dataset and the datasets of all other syndicates. These are now described in detail.

3 Creation of a dataset

Your data set must contain the following three files.

3.1 A web page

A file called `index.html` that is a valid web page written in HTML (that is, it can be opened and viewed in a standard web browser). This file must contain the id-number and price of 32 products. Each product is surrounded by a `<div class="product" id="x">...</div>`, where `x` is the product number from 1 to 32. Within the `div` there must be exactly one `<span class="price">...</span>` that contains the price of the item (and nothing else), and one link to more information on the item. There should only be one link (`<a..>...</a>`) tag pair in the div. There should also be some descriptive text that is enclosed in a `<span class="info">...</span>` tag pair. The price may, or may not, have a dollar sign and/or cents. For example,

```
<html>
...
    <div class="product" id="13">
        <img src="bike.png"><!-- optional -->
        <span class="info">
            This Penny Farthing bike was used by William Bligh in a drunken rampage in Sydney in 1808.
        </span>
        <a href="#13">Click here</a> for more information.
        <span class="price">  250  </span>
        <p>Note the irony!</p><!-- optional -->
    </div>
...
```

The required tags can occur in any order in the `<div>`.
If your Web page includes local images, make sure their file size is small (less than 50K, say) and include them in your zip file.

3.2 Web analytics file (JSON)

This file must be named `web.json` and it contains information about customers interactions with the WWW site. It is a text file of the form

```
{"all":[
    {"date":"dd/mm/yyyy",
     "events": [
        {"id":login-name,
         event-type:product-id,
         "duration":seconds},
        ...
    ]},
    ...
]}
```

where `dd/mm/yyyy` is a valid date (after 31/12/1999), `login-name` is a string (the users login name), `product-id` is an integer between 1 and 32, `seconds` is an integer greater than 0; and `event-type` can be one of: `"hover"`, where the mouse was held over the product; or `"read"`, where the user clicked on the more-info link. They can only purchase the product after reading the more-info link; that is, after a `read` event.
For example

```
{"all":[
    {"date":"22/01/2000",
     "events": [
        {"id":"baileyj",
         "hover":13,
         "duration":14},
        {"id":"bjpop",
         "hover":1,
         "duration":6},
        {"id":"baileyj",
         "read":13,
         "duration":61}
    ]},
    {"date":"23/01/2000",
     "events": [
        {"id":"bjpop",
         "read":1,
         "duration":46},
     ]},
    ...
]}
```

3.3 Purchase database (CSV)

The YOWS database has many tables, but only one important here is `Purchase`.

It has 5 columns:

Id	Customers login id
Product	Integer in range 1 to 32 that is product number purchased
Quantity	The number of product purchased
Date	Date of the purchase in dd/mm/yyyy format
Shipping	Cost of shipping as a float

You should create a CSV file called `Purchase.csv` (note capitalisation) that has 5 columns, each with the heading as per the table definition. Entries in the table are rows in the CSV file, and should be consistent with the web analytics JSON file you create. For example, using the JSON file above, we could have

```
Id,Product,Quantity,Date,Shipping
baileyj,13,2,22/1/2000,52.00
bjpop,1,1,23/1/2000,5
```

In order to make your JSON and CSV consistent, you should write a python program to generate the two files that simulates many customers (at least 200 a year with at least 50 sales a year, say) over 16 years. You should put some signal in your data that might be interesting (or difficult) to find with the analysis. For example, you might like to build some probabilities of certain events occurring conditioned on others and generate your data using that model.

4 Analysis

We will have 12 datasets (one per syndicate including your own), and each dataset will have 3 files: `index.html`, `web.json` and `Purchase.csv`.
You should write Python code that analyses a given input dataset by outputting:

1. the highest priced product, giving the product id and the price;
2. the mean number of words in the 32 `info` spans;
3. the product that has the most hover time, giving the product id and the total number of seconds;
4. the product that has the most read time, giving the product id and the total number of seconds;
5. the product that has the most sales, giving the product id and the number of units sold;
6. the product with the highest total shipping cost, giving the product id and the total cost;
7. the product with the highest profit (price times quantity less shipping), giving the product and profit;
8. the product with the lowest profit, giving the product and profit;
9. the product with the lowest hover-to-profit ratio, giving the product and ratio;
10. the product with the highest hover-to-profit ratio, giving the product and ratio;
11. the product with the highest read-to-profit ratio, giving the product and ratio; and
12. the product with the lowest read-to-profit ratio, giving the product and ratio.
13. the product with the highest info-to-profit ratio, giving the product and ratio.
14. the product with the lowest info-to-profit ratio, giving the product and ratio.

Your analysis should allow for reads without hovers.

The hover and read-to-profit ratio is defined as the total time for hover/read of a product divided by the total profit (quantity sold times price less the total shipping cost). The info-to-profit ratio is defined as the total number of words in the `info` span for the product product divided by the total profit (quantity sold times price less the total shipping cost). A word is any contiguous sequence as defined by the default functioning of python `split()`.

If there are two or more products with the highest profit/shipping cost/etc, return results for the product with the lowest ID number from amongst the ties.

The 14 pieces of data required should be printed one line per item, with spaces separating values. Do not add any extra punctuation, words, and so on as the answers will be automatically compared across syndicates. All 14 answers rounded to one decimal place (except the product id's - no decimal places).

5 Assessment

5.1 Within-syndicate responsibilities

We expect that you will structure the Python code so that it is split into modules/functions, and different members of the syndicate will write different parts of the code. Naturally you can collaborate, but it is in the best interests of the less experienced coders that they be given a coding task to complete. The temptation for the stronger coders to do the lot is high, but there will be coding questions about this assignment on the exam that each individual must answer. We suggest the stronger coders concentrate on the design, and splitting the tasks up amongst the syndicate, aiming to allot the tasks based on a syndicate member's abilities.

5.2 Submitting data/code/description

You will make two submissions of data/code/description.

- An initial submission of data/code/description
- A revised submission of data/code/description

Both your initial and revised submissions of data/code/desciption should have the following format.

- The submission should be a single zip file (**not** tar, rar, xz, ...) for the syndicate
- The zip file should contain your three data files
- The zip file should contain your python code that is used to generate the JSON and CSV and also do the analysis to answer the 14 questions.
- The zip file should contain a pdf document (not Word) as follows.
  - A description of how your Python is structured. What are the main modules/files/functions and what do they do? There is no need to describe trivial functions, just the main idea of your software. Perhaps a diagram? Also include instructions on how we should run it.
  - A table showing which syndicate members wrote which python code.
  - A table showing the output of your code applied to your 3 data files (i.e. your output for the 14 questions on your 3 data files)

Both initial and revised Submissions will be via Canvas.

Your initial submission should be before 9am 5th July.

The data files of each syndicate will be made available on 5th July. You may test your code on the other syndicate's data, make changes to your Python and re-submit up until 9am 7th July, after which no more submissions will be allowed. The results of a syndicate's code on their own data will be taken as the ground truth for comparing results.

Your revised submission will be the one that is used for marking purposes.

5.3 Peer Review

There is a small amount of marks allocated to peer reviewing two other Syndicate's submissions. The peer review should critique the readability of code and logic, praise clever things, and offer alternates for not so clever things. About 6 bullet points are expected per review. While the feedback will be anonymised, with only 12 syndicates there is little chance that it will be truly anonymous, so be polite. All people will be able to see all feedback. You will be marked on your feedback as per below. Again, less experienced coders in particular should read and learn from the other syndicate's submissions.
Peer review is due by 11:59pm 8th July via email to baileyj@unimelb.edu.au. Please use the subject line: "MBUS90500 Peer Reviews by Syndicate X". In your email, make it clear which two syndicates you are reviewing. I will anonymise the reviews and post them on Canvas.

5.4 Timeline

Initial submission of data and code	before 9:00am 5 July	via Canvas
Code tinkering and revised submissions allowed	before 9:00am 7th July	via Canvas
Code released by James for peer review	09:30am 7th July	via Canvas
Peer reviews submitted	before 11:59pm 8th July	via Email
Peer reviews released by James	9th July	via Canvas

5.5 Marking Rubric for Syndicate

Component	Property	Mark
Approach to data generation	Genuine link between JSON and CSV	/3
(12 marks)	Complexity of the data	/3
	Code is Clean, Commented Readable/Maintainable	/4
	Some evidence of code testing*	/2
Approach to data analysis	Parsing of HTML	/2
(12 marks)	Parsing of JSON	/1
	Parsing of CSV	/1
	Combination of data	/2
	Code is Clean, Commented Readable/Maintainable	/4
	Some evidence of code testing*	/2
Correctness	All syndicate's data correct	5
(5 marks)	Most syndicate's data correct	3
	Not many syndicate's data correct	1
	Doesn't work	0
Peer review 1	Critique of approach/logic	/1
(3 marks)	Critique of readability	/1
	Critique of code	/1
Peer review 2	Critique of approach/logic	/1
(3 marks)	Critique of readability	/1
	Critique of code	/1
	Total	/35

* For example, functions called `test_xxx()` or even a separate testing modules `test.py`.