Assignment 1: Javascript

COMP 315: Cloud Computing for E-Commerce

February 2025

1 Introduction

A common task when backend programming is data cleaning, which is the process of taking an initial data set that may contain erroneous or incomplete data, and removing or fixing those elements. In this assignment, you will be tested on your knowledge of JavaScript by implementing a set of functions that perform data cleaning operations on a dataset.

2 Objectives

By the end of this assignment, you will:

- Gain proficiency in using JavaScript for data manipulation.
- Be able to implement various data cleaning procedures, and understand the significance of them.
- Have developed problem-solving skills through practical application.

3 Initial setup

On the canvas page for this assignment, there is a template file called *Data_Processing.js*. Upload this template file to Codegrade. Note that the function signatures and module export statements have been prepared for you. Codegrade hosts a series of unit tests that will auto mark your code, and will present you with your final mark. This allows for immediate feedback and removes any potential bias in marking. You have an unlimited number of submissions to Codegrade. For this assignment you cannot import any additional libraries, as only the File System (FS) library is allowed. The data you will be cleaning can be seen in Table 1.

Data name	Note
id	This is a unique integer identifier for each item.
name	Each product has a name, which for this data set will just be the type of product followed by a unique identifier integer.
price	This is a real value representing the cost of the product.
category	Each product has a general category such as Clothing or Electronics, and is saved as a string.
type	Each product's type is a string that represents what sort of item it is.
quantity	This is an integer value that shows how many of this item is currently in stock.
rating	This is a real value that is a multiple of 0.5, starting from 0 and being capped at 5. This value represents the number of stars users rate the item.
image_link	This is the link to the product in the database, and is saved in the format of "[category]/[type]/[name].jpg". Name is all in lowercase with the spaces replaced with underscores.

Table 1: The attributes that are stored for each product

Data name	Note
category	Each product has a general category such as Clothing or Electronics, and is saved as a string.
type	Each product's type is a string that represents what sort of item it is.
min_price	This number represents the minimum price for this type of product.
max_price	This number represents the maximum price for this type of product.

Table 2: The attributes that are stored for pricing each type of product

4 Data cleaning

Add the following functionality to each of the functions. Do not modify the function signature or return statement. The *file_name* variable is the string that corresponds to a JSON file name without the '.json' extension. For example if the file is called *test.json* then *file_name* would have *test* stored. The *entries* variable is an array of JSON objects corresponding to products. Each JSON object has the attributes listed in Table 1. The *products_prices* variable is an array of JSON objects corresponding to the minimum and maximum price for each type of product. Each JSON object has the attributes listed in Table 2.

4.1 load_JSON

This function should take the *file_name* variable and load the corresponding JSON file. Note that the '.json' file extension should be added. If the file cannot be found then an exception should be thrown that says "'[file_name].json' cannot be found", where [file_name] is replaced with the actual file name. The input text should be converted to an array of JSON objects. If this process cannot be done, then it should throw the error "'[file_name].json' is not a valid JSON file" where again [file_name] is replaced with the actual file name.

4.2 clean_name

This function should iterate through each of the products, and replace null name values with the correct name. If the name value is 'null' then the *image_link* variable should be checked. If *image_link* is not 'null' then the name should be extracted from that. It should replace the underscores with spaces, and each word should be capitalised. If *image_link* is 'null' then the name should be taken from the 'type' variable. Each name is unique, and is the *type* value with a unique integer identifier appended to it. This identifier corresponds to the number of that type of product that have occurred up to that point. For example if the product type is "Belt", and it is the third "Belt" in the product list then it's name should be "Belt 3".

4.3 clean_price

This function takes in the *entries* variable, as well as the *product_prices* variable. The price values should always be non-negative number, and should first be changed to positive. The corresponding *min_price* and *max_price* for this type of product should be found from *product_prices*. If the price for the product is less than the *min_price* value for that type of product, then it should be set as the *min_price* value. For example if the price of *Shoe*4 is 1, and the *min_price* value for *Shoe* is 3, then the price for *Shoe*4 should be set to 3. If the price for the product is more than the *max_price* value for that type of product, then it should be set as the *max_price* value.

4.4 clean_category

This function should check if the *category* is 'null', then it should use the *type* value to fill in the missing data. It should find other entries with the same *type* value, and then use this to fill in the corresponding *category* value.

4.5 clean_type

This function should check if the *type* is 'null', then it should check the *name* value. If the *name* is not 'null' then it should remove the unique number and use that value. If the *name* is 'null' then it should derive the name from the *image_link*.

4.6 clean_quantity

The *quantity* of a product must be a non-negative value, so if the value is negative then it should be set to positive. If the value is not an integer, then it should be rounded to the nearest integer using standard 'rounding half up' rounding.

4.7 clean_rating

If the *rating* for a product is set to 'null', then it should be set to 0. The *rating* of a product is given as a star value between 0 and 5 stars. It is possible to give a product half a star, therefore the possible values are: 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5. If a value does not correspond to one of these, then it should be rounded to the nearest possible value.

4.8 clean_image_link

This function populates 'null' *image_link* values using the other variables for the JSON object. This should be in the format of "[category]/[type]/[name].jpg". The *category* and *type* values are used as they are, however the *name* value should all be in lowercase with spaces replaced with underscores. The '.jpg' text should be appended to the end.

5 Marking

The marking will be carried out automatically using the CodeGrade marking platform. A series of unit tests will be ran, and the mark will correspond with how many of those unit tests were successfully executed. Your work will be submitted to an automatic plagiarism/collusion detection system, and those exceeding a threshold will be reported to the Academic Integrity Officer for investigation regarding adhesion to the university's policy https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/appendix_L_cop_assess.pdf.

6 Deadline

The deadline is 23:59 GMT Thursday the 20th of March 2025. Late submissions will have the typical 5% penalty applied for each day late, up to 5 days. Submissions after this time will not be marked. https://www.liverpool.ac.uk/aqsd/academic-codes-of-practice/code-of-practice-on-assessment/