Lab 1: JSON Manipulation

Due date: January 14, 11:59pm.

Lab Assignment

Assignment Preparation

This is an individual lab.

Dataset

You will be working with a JSON dataset that contains 10,000 records of liquor sales from the State of Iowa. State of Iowa requires documentation of all hard liquor sales to liquor stores, and makes the sales data available in a CSV format. The full data is available at (the link is available on the course web page as well)

https://data.iowa.gov/Economy/Iowa-Liquor-Sales/m3tr-qhgy

The entire dataset is millions of records and clocks at over 4G. For this lab, I converted the first 10,000 records to JSON and made it available to you in a pretty-printed, human readable format in a file `iowa.json`.

A sample JSON object from the dataset is shown below:

```
{
       "Invoice/Item Number": "S08102400013",
       "Date": "10/03/2012",
       "Store Number": 4202,
       "Store Name": "Fareway Stores #829 / Sioux City",
       "Address": "4267 SERGEANT RD",
       "City": "SIOUX CITY",
       "Zip Code": "51106",
       "County Number": 97.0,
    "County": "Woodbury",
    "Store Location": "4267 SERGEANT RD\nSIOUX CITY 51106\n(42.451213, -96.356879)",
    "Category": 1031080,
    "Category Name": "VODKA 80 PROOF",
    "Vendor Number": 297,
    "Vendor Name": "Laird And Company",
    "Item Number": 35916,
    "Item Description": "Five O'clock Vodka",
    "Pack": 12,
    "Bottle Volume (ml)": 750,
    "State Bottle Cost": "$3.31",
    "State Bottle Retail": "$4.96",
    "Bottles Sold": 24,
    "Sale (Dollars)": "$119.04",
    "Volume Sold (Liters)": 18.0,
    "Volume Sold (Gallons)": 4.76
}
```

Most of the fields are more or less self-explanatory. The `Invoice/Item Number` field is a unique ID of the particular sale. `"Store Number"` is the unique Id of the store responsible for the sale. Note that there are two sets of fields documenting the address and geolocation of the store. `Category` field is the unique Id of the type of liquor that was sold, and `Category Name` is its name. `Vendor Number` is the unique Id of the wholesale vendor supplying the liquor to the liquor store (the liquor store is the buyer in this dataset), `Vendor Name` is the name of the wholesale vendor. `Item Number` is the unique id of the specific type of alcohol sold, `Item Description` is the text string specifying the name of the liquor. The remaining fields describe the sale volume - number of bottles, size of bottle, price, total sale, total volume, etc.

Python Assignment

Write five Python programs. Each program shall take `iowa.json` file as input and produce a specific output based on your analysis of the data contained in the file.

liquorAggregate.py: Your first program, `liquorAggregate.py` shall compute the following information:

- Total number of bottles sold
- Total volume of alcohol sold in liters
- Overall revenue derived from all the sales
- Average price per bottle

This information has to be computed over **the entire dataset**, and reported back as a single JSON object with the structure (the field names are self-explanatory):

```
 {
   bottles: <number of bottles>,
   volume : <volume of alcohol in liters>,
   revenue: <overall revenue>,
   averagePrice: <average price per bottle>
}
```

storeAccounting.py: We are interested in grouping some basic information about purchases made by each store into a single JSON document. Write a program `storeAccounting.py` which produces **one JSON object per each individual store**. The object shall have the following format:

```
 { store: <number>,
   storeName: <store name>,
   purchases: [ {invoice: <invoice #>,
                 amount: <total dollar amount of purchase>}, ...
              ]
   dates: [ <date1>, <date2>,...]
}
```

Here, `store` and `storeName` are the `Store Number` and `Store Name` values from the original JSON documents; `purchases` contains an array of objects that document the invoice number and the `Sale (Dollars)` for each sale for that store, and `dates` is an array that stores **without duplication** all dates on which purchases were made by the given store, *sorted in chronological order.*

distribution.py We are interested in figuring out the distribution network for the wholesale alcohol sales. Write a program `distribution.py` that for each distributor (vendor) lists all stores to which it delivered. The format of the output is

```
{
 vendorId : <vendor number>,
 vendorName: <vendor name>,
 stores: [ {storeId: <store number>,
            store: <store name>,
            county: <county>
           }, ...
          ]
}
```

Please make sure that each store shows up exactly once in the array.

vodkas.py: We would like to find out all varieties of the category `"VODKA 80 PROOF"` that are being sold in the State of Iowa. Analyze the input dataset to discover such varieties and output them in the following format:

```
{type: "VODKA 80 PROOF",
 id: <unique id of the type of alcohol>,
 description: <name of the alcohol>
}
```

for each type of sold vodka with no duplicate entries. Note that the value of the `type` field is kept constant in your output, while `id` comes from the `Item Number` field in your original data, and `description` comes from the `Item Description` field.

categorySales.py: For our last exercise in Python, we want to figure out how many individual sales (i.e., you are counting number of invoices) took place for each category of liquor sold in Iowa. Your output shall have the following format:

```
{
 catId: <category id>,
 category: <category name>,
 nSales: <number of sales>
}
```

The output shall contain one object per category, and shall be *sorted* in ascending order by category Id.

Java Assignment

Write two Java programs to manipulate the sales data.

leaderboard.java: We are interested in discovering the vendor that sold the largest quantity of alcohol (in terms of number of bottles) to the stores. The `leaderboard.java` program shall return a single JSON object with the following structure:

```
{
 vendorID: <vendor number>,
 vendor: <vendor name>,
 bottles: <total number of bottles sold>
}
```

with the identity of the vendor with the largest number of bottles sold.

countyStats.java: We are interested in figuring out if there is a correlation between total population in the county and the sales of alcohol to the stores in that county. To this extent, you shall write a Java program `countyStats.java` that takes **two inputs** provided to you: the `iowa.json` files with the liquor sales, and a provided for you file `counties.json` documenting the population in each county in Iowa. The records in `counties.json` file have very straightforward format, as illustrated in the example below:

```
  {
        "id": "5 ",
        "county": "Black Hawk County ",
        "population": "132,960"
    }
```

The records in the file are sorted in descending order by county poplulation, and `id` is essentially the rank of the county in terms of its population.

Your program shall compute for each county the total amount of alcohol sales to the stores in the county (in dollars), and the total volume (in liters). It then shall use the county population information to compute the *per capita* alcohol sales and *per capita* volume. The resulting information shall be returned in the following JSON format:

```
{
 county: <CountyName>,
 totalSales: <total sales in dollars>
 totalVolume: <total volime in liters>
 perCapSales: <per capita sales in dollars per person>
 perCapVolume: <per capita volume in liters per person>
}
```

(Please note, because this is an incomplete dataset, the information you compute is not actually a good indicator of any relationships... But your code shall produce the true per capita numbers when run on the full dataset, and that's what's important for this exercise.)

General Comments and submission

All your programs, both written in Python and in Java must produce valid JSON as output. This means that when your programs are producing multiple JSON documents as output, the output shall be a JSON array. For a single JSON object, there is no need to encapsulate it in an array.

All your programs shall take a file name of the input file as an input parameter (`countyStats.java` shall take two filenames as input parameters).

Each program/README file you submit must contain your name and cal poly email address.

Submit the seven programs you developed, and a `README` file containing any additional information I may need to know to compile and run your programs (e.g., the `javac` compilation command, if it incorporates some parameters I may not be immediately aware of).

You can develop Python code in Jupyter notebooks, but for this assignment, I want you to submit standalone Python programs.

Use Python 3 syntax.

Use `handin` to submit as follows:

```
$ handin dekhtyar lab01 <FILES>
```

Good Luck!