Task 1

For each dataset that you profile, you will output a JSON file with a dataset specification as explained below. Make sure that you have a valid JSON file![1]

```
{
   "dataset_name": the name of the dataset, which would be the name
   of the dataset file (type: string)
   "columns": a list of the dataset columns (type: array) -- see the
   column specification below
   "key_column_candidates": a list of column names that are candidates
   for being the key of the dataset (type: array)
}
```

The column specification can be found below. For the **`data_types`** attribute, only use the data types you found for that specific column; no need to have a JSON object for `INTEGER (LONG)` if there are no values with that type, for instance.

```
{
   "column_name": the name of the column (type: string)
   "number_non_empty_cells": the number of non-empty cells (type:
   integer)
   "number_empty_cells": the number of empty cells, i.e., cells with
   no value (type: integer)
   "number_distinct_values": the number of distinct values in the
   column (type: integer)
   "frequent_values": a list with the top-5 most frequent values of
   this columns, in descending order of frequency (type: array)
   "data_types": [
      {
         "type": "INTEGER (LONG)"
         "count": the number of values of type INTEGER (LONG) in the
         column (type: integer)
         "max_value": the maximum value among the values of type
         INTEGER (LONG) (type: integer)
         "min_value": the minimum value among the values of type
         INTEGER (LONG) (type: integer)
         "mean": the mean of the values of type INTEGER (LONG) (type:
         float)
```

[1] You can use resources such as https://jsonformatter.curiousconcept.com/ to check the validity of your json output.

```
      "stddev": the standard deviation of the values of type
      INTEGER (LONG) (type: float)
    },
    {
      "type": "REAL"
      "count": the number of values of type REAL in the column
      (type: integer)
      "max_value": the maximum value among the values of type REAL
      (type: float)
      "min_value": the minimum value among the values of type REAL
      (type: float)
      "mean": the mean of the values of type REAL (type: float)
      "stddev": the standard deviation of the values of type REAL
      (type: float)
    },
    {
      "type": "DATE/TIME"
      "count": the number of values of type DATE/TIME in the column
      (type: integer)
      "max_value": the maximum value among the values of type
      DATE/TIME (type: string)
      "min_value": the minimum value among the values of type
      DATE/TIME (type: string)
    },
    {
      "type": "TEXT"
      "count": the number of values of type TEXT in the column
      (type: integer)
      "shortest_values": a list with the top-5 shortest values
      (i.e.: values with shortest length / number of characters),
      in ascending order of length (type: array)
      "longest_values": a list with the top-5 longest values (i.e.:
      values with longest length / number of characters), in
      descending order of length (type: array)
      "average_length": the average value length (type: float)
    }
  ],
    ...
  ]
}
```

Task 2

The JSON format for the columns you will work with for Task2 is as follows.

```
{
    "column_name": the name of the column (type: string)
    "semantic_types": [
        {
           "semantic_type": label of the semantic type choosing from
           the list provided below (type: string)
           "label": semantic type is other, provide a label of the
           semantic type (type: string)
           "count": the number of instances in the column that
           belong to that semantic types (type: integer)
        },
        ...
    ]
}
```

```
Label list
[person_name, business_name, phone_number, address, street_name,
city, neighborhood, lat_lon_cord, zip_code, borough, school_name,
color, car_make, city_agency, area_of_study, subject_in_school,
school_level, college_name, website, building_classification,
vehicle_type, location_type, park_playground, other]
```