Problem Statement

The goal is to read data from popular file formats (.csv, .txt, .json), to perform operations on this data, and to finally save this modified data into the original file formats.

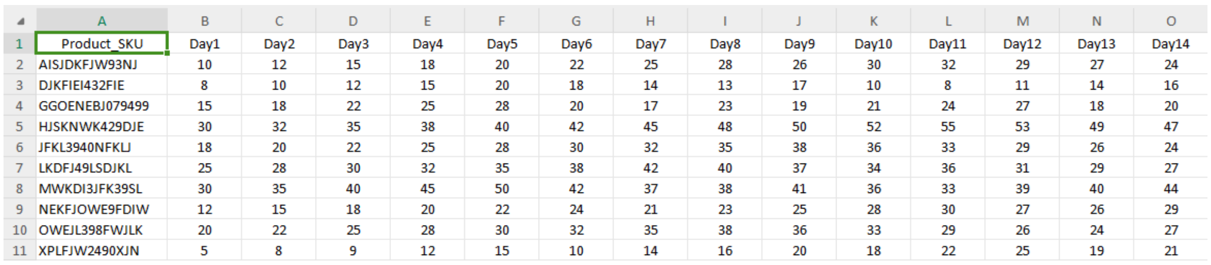
Understanding the Data

The dataset and exercise notebook for this project have been provided in the student folder which can be found here. The *mainfolder.zip* file contains data about products for which the unique ID provided is the stock-keeping unit, or SKU for short. The ZIP file contains a structured collection of sales data and product information organized into a main folder with three key components:

```
- mainfolder/
    - sales_data.csv
    - product_details/
        -  details_AISJDKFJW93NJ.json
        -  details_DJKFIEI432FIE.json
        ...
    - product_description/
        - description_AISJDKFJW93NJ.txt
        - description_DJKFIEI432FIE.txt
        ...
```

folder structure

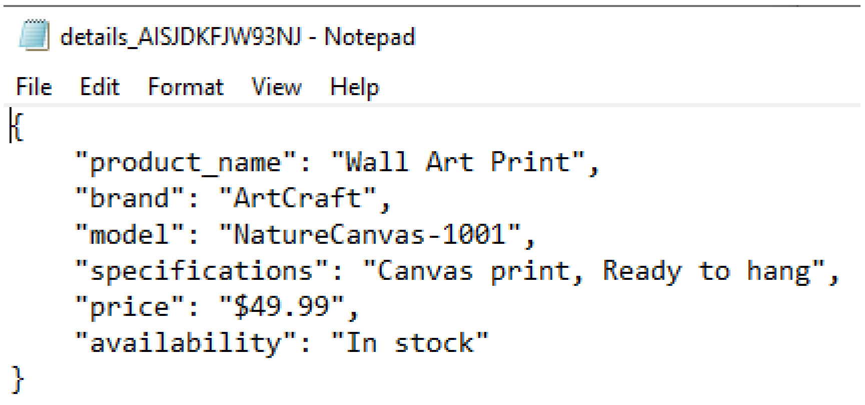
Sales Data *(sales_data.csv):* A CSV file that includes sales data for various products over a 14-day period. Each row corresponds to a different product, identified by a *Product_SKU*. The columns *Day1* through *Day14* represent the sales figures for each consecutive day.



Product_SKU	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
AISJDKFJW93NJ	10	12	15	18	20	22	25	28	26	30	32	29	27	24
DJKFIEI432FIE	8	10	12	15	20	18	14	13	17	10	8	11	14	16
GGOENEBJ079499	15	18	22	25	28	20	17	23	19	21	24	27	18	20
HJSKNWK429DJE	30	32	35	38	40	42	45	48	50	52	55	53	49	47
JFKL3940NFKLJ	18	20	22	25	28	30	32	35	38	36	33	29	26	24
LKDFJ49LSDJKL	25	28	30	32	35	38	42	40	37	34	36	31	29	27
MWKDI3JFK39SL	30	35	40	45	50	42	37	38	41	36	33	39	40	44
NEKFJOWE9FDIW	12	15	18	20	22	24	21	23	25	28	30	27	26	29
OWEJL398FWJLK	20	22	25	28	30	32	35	38	36	33	29	26	24	27
XPLFJW2490XJN	5	8	9	12	15	10	14	16	20	18	22	25	19	21

sales data

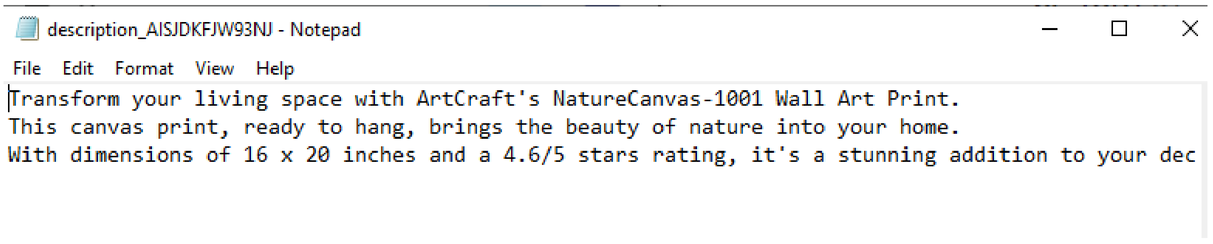
Product Details *(product_details* folder): This folder includes JSON files, again with filenames corresponding to product SKUs (e.g., *details_AISJDKFJW93NJ.json)*. These files contain detailed attributes of the products, such as the model, specifications, pricing, etc.



```
{
    "product_name": "Wall Art Print",
    "brand": "ArtCraft",
    "model": "NatureCanvas-1001",
    "specifications": "Canvas print, Ready to hang",
    "price": "$49.99",
    "availability": "In stock"
}
```

product details

Product Descriptions *(product_descriptions* folder): This folder contains text files, each corresponding to a specific product identified by the SKU in the filename (e.g., *description_AISJDKFJW93NJ.txt)*. These files provide descriptive information about the products.


description_AISJDKFJW93NJ - Notepad
File Edit Format View Help

```
Transform your living space with ArtCraft's NatureCanvas-1001 Wall Art Print.
This canvas print, ready to hang, brings the beauty of nature into your home.
With dimensions of 16 x 20 inches and a 4.6/5 stars rating, it's a stunning addition to your dec
```

product description

Your first task is to set up the environment for this project by loading the required packages and modules.

You will achieve all of this by completing the following sub-task:

1. Write code to import the following packages:
    - For navigating through files stored on your device or on Google Colaboratory: *os*
    - For working with JSON files: *json*
    - For working with CSV files: *csv*

Then, you will ensure that the necessary files are loaded into the working environment. This includes loading sales data from a CSV file, product details from JSON files, and product descriptions from text files.

We recommend that you either use Jupyter Notebook or Google Colab to build and execute your code. You will achieve all of this by completing the following:

1. In case you are using Google Colab,

    - upload the files directly from your PC to Colab or
    - import *drive* from *google.colab* and mount your Google Drive
2. In case you are using Jupyter Notebook, please make sure that your files and folders are all in the right place.
3. Use a function named *load_data()* to read data from the *sales_data.csv* file, JSON files in the product_details folder, and TXT files in the product_description folder and store them in three dictionaries called *sales_data, product_details* and *product_descriptions*.

Create functions that let the admin add sales data, product details, and a product description for a new product, or to update these for an existing product.

The code should help the admin accomplish the following tasks:

1. Use a function named *update_sales_data()* to add the following sales data:
    - Product SKU: *CMWKCILOP27KF*
    - Sales data for the past 14 days: *[8, 14, 16, 7, 15, 21, 14, 16, 32, 29, 26, 30, 25, 22]*
2. Use a function named *update_product_details()* to add the following product details to the same SKU as above:
    - Product name: *Pokemon Card*
    - Brand: *GameFreak*
    - Model: *ScarletViolet151*
    - Specifications: *Genuine, TCG, English*
    - Price: *$1.99*
    - Availability: *In stock*
3. Use a function named *update_product_description()* to add the following product description to the same SKU as above:
    - Product Description: *Original Pokemon TCG Pikachu card*
4. Use a function named *update()* that calls the functions defined in 1, 2, and 3 to add product details, sales data, and product descriptions, allowing user inputs at each stage. The *update()* function should follow certain rules and perform internal checks to validate the provided information as follows:

- It prompts the user to input the SKU, which must be exactly 13 characters long. If the SKU does not meet this requirement, print an error message and terminate the function without updating any dictionaries.
- It prompts the user to enter sales data for the last 14 days, which must consist of exactly 14 whole numbers separated by spaces. If the input does not meet this criterion, print an error message and terminate the function without updating any dictionaries.
- It collects product details from the user, including name, brand, model, specifications, price, and availability. These inputs are required but not subject to specific validation criteria for this function.
- It prompts the user for a product description, which is also required for successful product registration.
- If all inputs are validated successfully, the function updates the *product_details, sales_data,* and *product_descriptions* dictionaries with the provided product information and prints a success message.
- The function returns a tuple containing the updated *product_details, sales_data,* and *product_descriptions* dictionari es in that order.

Having added and/or updated products, you want to now enable the admin to overwrite the existing files to reflect this new and/or updated data.

You will achieve this by building a *dump_data()* function which should do the following:

1. It should accept four parameters:
    - *sales_data* - A dictionary containing sales data, with SKU as keys and a list of sales quantities for the last 14 days as values.
    - *product_details* - A dictionary containing product details, with SKU as keys and details as values. Details include attributes like name, brand, model, specifications, price, and availability.
    - *product_descriptions* - A dictionary containing product descriptions, with SKU as keys and the textual description of the product as values.
    - *main_folder* - A string representing the location of the main folder, which should contain *product_details* and *product_descriptions* subfolders.
2. It should dump the data into a folder and files containing the same file structure as the data that was loaded, by overwriting the original files. It should:

- Dump *sales_data* into a CSV file named *sales_data.csv* located in the *main_folder*. Each row in the CSV file represents the sales data for a product, with fields for the SKU and sales quantities for each of the 14 days.
- Dump each entry in *product_details* into a separate JSON file within the *product_details* subfolder of the *main_folder*. Each file is named after the SKU (e.g. *details_ABCDEFGHIJKLM.json)* of the product and contains the details of that product in JSON format.
- Dump each entry in *product_descriptions* into a separate TXT file within the *product_descriptions* subfolder of the *main_folder*. Each file is named after the SKU of the product (e.g. *details_ABCDEFGHIJKLM.txt)* and contains the textual description of that product.
- If a folder or subfolder does not exist, the function should automatically create it.