



Informe Solemne N°3

Inventariado de tarjetas de video





Integrantes del grupo:

* Alan Silva
* Vincent Bustamante
* Lucas Brown

Nombre de docente: Rodrigo Barria

Fecha: martes 21 de noviembre de 2023

Índice

[1. Problemática 3](#_Toc153154983)

[2. Metodología 4](#_Toc153154984)

[3. Resultados 12](#_Toc153154985)

[4. Conclusiones 18](#_Toc153154986)

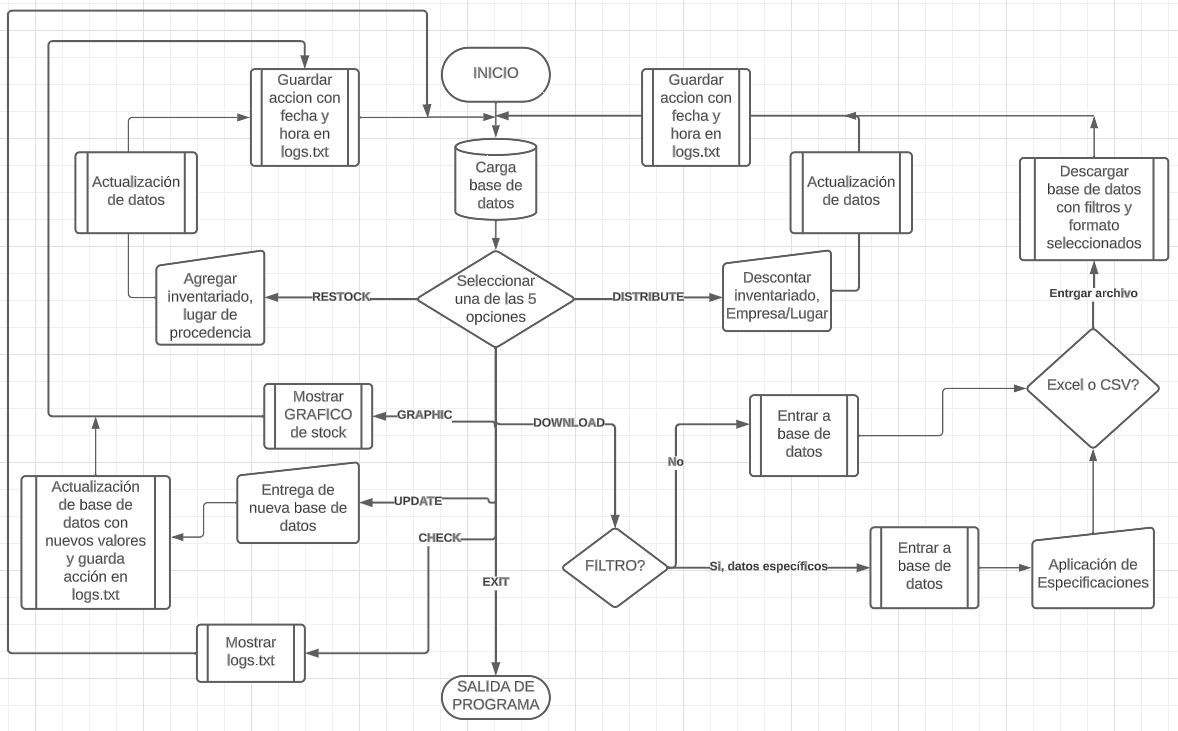
[5. Referencias 19](#_Toc153154987)

[6. Software e Implementaciones utilizadas 19](#_Toc153154988)

# Problemática

Debido a la complejidad de las especificaciones del hardware de las tarjetas gráficas y la gran cantidad de modelos disponibles, la distribuidora enfrenta la problemática de una "Gestión Ineficiente del Inventario". La diversidad de especificaciones, como velocidades de reloj, capacidades de memoria y características exclusivas, complica el seguimiento y control preciso de cada tarjeta gráfica en el inventario. Esto ha llevado a desafíos en la identificación rápida de productos, la actualización efectiva de especificaciones y la gestión óptima de la distribución, afectando la eficiencia operativa y generando posibles pérdidas económicas debido a errores en el manejo de inventario.

# Metodología



(Favor revisar GitHub para obtener una mejor visualización del diagrama de procesos)

Compatibilidad y requisitos:

Es necesario tener instalada una versión de Python 3.10 o superior.

Para el buen funcionamiento del programa es necesario la importación de ciertas bibliotecas, como “pandas”, “matplotlib”, “openpyxl”, donde un “Batch File” llamado “setup.bat” se encargará de instalar las dependencias necesarias, especificadas en un archivo de texto llamado “requirements.txt”.

Lógica y funciones

Se inicializa la base de datos mediante las funciones disponibles en la biblioteca “pandas”, guardándola como un Dataframe en una variable “df”.

A continuación, se activa la función “menu”, que se encarga de mostrar las funciones disponibles en el programa, en este caso "Distribute","Restock","Check", "Filter", “Graphic”, “Update” o "Exit", y preguntar al usuario que opción (“option”) desea escoger y así ingresar a la función señalada.

Distribute:

La función “Distribute” se encarga, como su nombre indica, de enviar las órdenes de actualización del stock, al descontar un cierto producto que fue distribuido a otra tienda.

Se importa la función “check” y su subfunción “sabe\_current\_action”.

from functions.check import save\_current\_action

Este método recibe el Dataframe “df” y de acuerdo al “id” del producto ingresado por el usuario filtra, con la funcion “.filter” de “pandas”, en el Dataframe y crea uno nuevo llamado “product”.

def distribute(df):  
 id = int(input("Enter the id of the product you want to distribute: "))

Luego usa el Dataframe “product” para extraer el nombre y el stock del producto distribuido, con la función “.iloc” de “pandas”, almacenando en las variables “gpu\_name” y “current\_stock” respectivamente, y de esta manera informar mediante un “print” al usuario sobre el producto consultado.

product = df.filter(items=[id], axis=0)  
gpu\_name = product.iloc[0]["gpu\_name"]  
current\_stock = product.iloc[0]["stock"]  
  
print(f"The product (id: {id}) {gpu\_name} currently has {current\_stock} units available.")

Posteriormente se le consulta al usuario sobre la ubicación de distribución y sobre la cantidad de producto, guardando este último dato en la variable llamada “distribution\_amount”, que desea distribuir en base a la información señalada, advirtiendo sobre si la cantidad solicitada es mayor a la de stock, para de esta manera descontar la cantidad de producto solicitada. Si cumple las condiciones se guardan los cambios realizados.

distribution\_location = input("\nEnter the location where the product will be distributed: ")  
distribution\_amount = int(input("Enter the amount of units to distribute to the location: "))  
  
if distribution\_amount > current\_stock:  
 print("ERROR: Distribution amount exceeds current stock.")  
 return  
  
df.at[id, 'stock'] = current\_stock - distribution\_amount

Finalmente, se informa al usuario sobre los cambios realizados mediante un “print”, y además se registra la acción mediante la importación e instancia del método “check”.

print(f"{distribution\_amount} units of {gpu\_name} distributed to {distribution\_location} successfully.")  
  
save\_current\_action(f"[DISTRIBUTE] The product (id: {id}) {gpu\_name} has been distributed. {distribution\_amount} units were sent to {distribution\_location}. Remaining stock: {current\_stock - distribution\_amount}")  
df.to\_csv("database\\techpowerup\_gpus.csv")

Restock:

Para no entrar en redundancias, este método funciona casi igual que el metodo “Distribute”, con la única salvedad de que, en vez de descontar el valor de stock ingresado por el usuario, agrega este monto al stock.

origin\_location = input("Enter the location where the product came from: ")  
new\_stock = int(input("Enter the amount of units to restock: "))  
  
df.at[id, 'stock'] = current\_stock + new\_stock  
print("Product stock updated succesfully")  
  
save\_current\_action(f"[RESTOCK] The product (id: {id}) {gpu\_name} has been restocked. {new\_stock} new units were received from {origin\_location}. Now it has {current\_stock+new\_stock} units")  
df.to\_csv("database\\techpowerup\_gpus.csv")

Check:

Este método importa la biblioteca “datetime” y su función “datetime”. También importa “getpass”.

from datetime import datetime  
import getpass

Funciona en base a 2 funciones; “save\_current\_action”, la cual registra la hora y fecha de una determinada acción (“action”) gracias a la bilbioteca “datetime”, además de guardar estas acciones en un archivo de texto plano llamado “logs”.

def save\_current\_action(action: str):  
 now = datetime.now()  
 time = now.strftime("%d/%m/%Y %H:%M:%S")  
 username = getpass.getuser()  
  
 with open("logs\\logs.txt", "a") as f:  
 f.write(f"{time} - User: {username} - Action: {action}\n")

También está la función “print\_logs”, que se encarga de mostrar en consola lo guardado. Esto lo hace mostrando las últimas 7 líneas que se encuentren en logs.txt. Además de un sistema de navegación usando ‘+’ para subir, ‘-’ para bajar y ‘q’ para salir del modo.

Lo primero que hace es verificar si el código se está ejecutando en un sistema operativo de Windows. Si es así, importará la liberia msvcrt.

    if os.name == 'nt':

        import msvcrt

Luego se define una función que mostrará las líneas en pantalla

    def display\_text(lines, start\_line, num\_lines):

        for i in range(start\_line, min(start\_line + num\_lines, len(lines))):

            print(lines[i].rstrip())

Otra para obtener las teclas presionadas del usuario

    def get\_user\_input():

        if os.name == 'nt':

            key = msvcrt.getch()

            return key

        else:

            user\_input = input("Press 'q' to exit the mode. + to move up. - to move down: ")

            return user\_input

Y finalmente la función principal que junta las funciones anteriores. Primero abre el archivo logs.txt. Seguido se asignan variables a la línea que se mostrará al comienzo, junto con cuantas líneas mostrará en total. Luego entra en un ciclo el cual limpia la consola cada vez para simular el efecto que sube y baja el texto, si el comando está siendo ejecutado desde Windows, ejecuta ‘cls’ en caso contrario, ejecuta ‘clear’. Luego usa la función ‘display\_text’ para mostrar el texto. Finalmente detecta la tecla presionada por el usuario y le suma o le añade un uno a la línea inicial. Este ciclo continua hasta que el usuario ponga la letra ‘q’. Cuando esto ocurre, se cierra el modo y vuelve al menú.

    def show\_logs():

        with open('logs\\logs.txt', 'r') as file:

            lines = file.readlines()

        start\_line = len(lines) - 1

        num\_lines\_displayed = 7

        while True:

            os.system('cls' if os.name == 'nt' else 'clear')

            display\_text(lines, start\_line, num\_lines\_displayed)

            print("Press 'q' to exit the mode. '+' to move up. '-' to move down")

            key = get\_user\_input()

            if key == b'+' and start\_line > 0:

                start\_line -= 1

            elif key == b'-' and start\_line < len(lines) - num\_lines\_displayed:

                start\_line += 1

            elif key.lower() == b'q':

                break

Esta función es la que es llamada al momento de elegir ‘check’ en el menú principal.

Filter/Download:

Probablemente la función más complicada de todo el código, por ello la siguiente explicación será simplificada para efectos prácticos, si desea ver más a detalle el código estará el enlace a GitHub al final del informe. Se importa la base de datos en forma de DataFrame y está compuesta por 7 subfunciones.

Inicia con la subfunción “applied\_filters” en donde se inicializa la función de las marcas “br”, en donde se consulta por el nombre de alguna de las 5 marcas disponibles, “Nvidia”,”Amd”,”Intel”, “Matrox” o “Ati”.

De acuerdo a lo anterior se ingresa a la subfunción “locate\_brand” que tiene por parámetros “df” y “brand”, entregando un nuevo DateFrame filtrado por la marca seleccionada guardándolo en “filtered\_df”.

Luego se inicializa la subfunción “filters”, en donde se define una variable global “selected\_filters”, la cuál es un diccionario de los siguientes 3 posibles filtros; Serie(“series”), generación(“gen”) o ram(“ram”). Se consulta en base a si quiere o no (“Yes /No”) utilizar el filtro actual.

Dependiendo de lo seleccionado se ingresa a ciertas funciones “gen”, “series” “ram”, pero todo dependiendo de la marca seleccionada, ya que ingresa a la clase correspondiente a cada marca, en la carpeta “graphics\_cards”, donde se realizan los filtros de Serie y generación, mientras que el filtro de ram se produce en la misma función “filter”.

Al final de todo el proceso se pregunta si el DateFrame filtrado lo desea en formato Csv o Excel, por medio de la subfunción Download. Si el formato es Csv se guarda el archivo en la misma carpeta del proyecto con un nombre ingresado por el usuario. En cuanto al formato Excel, este ingresa a la subfunción “csv\_to\_excel” donde se consulta sobre el nombre con el que se desea guardar y el nombre del libro principal, si no es ingresado ningún nombre de libro se asigna uno predeterminado “Sheet1”. Finalmente se descarga el archivo en la misma carpeta del proyecto.

Graphic:

Esta función crea un gráfico de torta en base al stock de las marcas disponibles.

Se importan las dependencias necesarias, como “pandas” (pd), “matplotlib” (plt) y “numpy” (np), y a partir de matplotlib se importa el método withStroke. También se utiliza la función “Check”

from functions.check import save\_current\_action  
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib.patheffects import withStroke

Se inicializa el Dateframe de acuerdo a la base de datos, guardándolo en una variable “df”.

df = pd.read\_csv('database\\techpowerup\_gpus.csv')  
df.set\_index("id", inplace=True)

Se define inicialmente una subfunción “brand\_pie\_chart” en donde se determinarán los parámetros del gráfico.

def brand\_pie\_chart():

Se enlistan las marcas disponibles en una variable “brands” para posteriormente usar esta lista para hallar el stock de cada marca y guardarlo en una variable “brand\_stocks”

brand\_stocks = [df[df["gpu\_name"].str.contains(f"{brand} ", case=False)]["stock"].sum() for brand in brands]

Luego se determina cuanto se van a expandir cada una de las partes correspondientes a cada una de las marcas con una variable “explode”, que es una lista inmutable con valor de tipo flotante.

Se realiza un procedimiento similar con cada uno de los aspectos necesarios como los colores “colors”, y las propiedades de las porciones “wp”.

# Explotar gráfica  
explode = (0.1, 0.0, 0.2, 0.3, 0.0)  
  
# Creating color parameters  
colors = ("yellowgreen", "crimson", "turquoise", "mediumblue", "black")  
  
# Wedge properties  
wp = {'linewidth': 1, 'edgecolor': "green"}

Se crea una función “func” que se encarga de crear las etiquetas automáticas son sus respectivos cálculos necesarios.

def func(pct, allvalues):  
 absolute = int(pct / 100. \* np.sum(allvalues))  
 return "{:.1f}%\n({:d} units)".format(pct, absolute)

En cuanto a la siguiente sección del código, este se encarga de la creación del gráfico gracias a la función “ax.pie”.

# Creating plot  
fig, ax = plt.subplots(figsize=(10, 7))  
wedges, texts, autotexts = ax.pie(brand\_stocks,  
 autopct=lambda pct: func(pct, brand\_stocks),  
 explode=explode,  
 labels=brands,  
 shadow=True,  
 colors=colors,  
 startangle=90,  
 wedgeprops=wp,  
 textprops=dict(color="white", path\_effects=[withStroke(linewidth=2.5, foreground='black')]))

A continuación, se agregan las leyendas del gráfico.

# Adding legend  
ax.legend(wedges, brands,  
 title="Brands",  
 loc="upper left",  
 bbox\_to\_anchor=(1, 0, 0.5, 1))

Por último se agregan ciertas personalización como el tamaño y “peso” de las etiquetas y el título del gráfico.

plt.setp(autotexts, size=8, weight="bold")  
ax.set\_title("Stock Distribution by Brand")

Se muestra el gráfico y se guarda la acción realizada.

plt.show()  
  
save\_current\_action(f"[GRAPHIC] A graphical representation of the stock has been acquired")

Update:

Esta función importa pandas y la función “check”.

import pandas as pd  
from functions.check import save\_current\_action

Define la subfunción principal “update” que recibe la base de datos en formato DateFrame “df”.

def update(df):

Se pide al usuario ingresar el directorio del DataFrame nuevo de formato csv para combinarlo con el anterior DateFrame, y también se inicializa el DateFrame anterior.

new\_df\_location = input("Please input the new CSV's folder location (path): ")

    df = pd.read\_csv("database\\techpowerup\_gpus.csv")

Posteriormente, utilizando la función try y except, se lee el nuevo DateFrame “new\_df”, y se posicionan las columnas de “id” de ambos. También se manejan ciertos posibles errores de compatibilidad:

new\_df = pd.read\_csv(new\_df\_location)  
if df is None or 'id' not in df.columns or 'id' not in new\_df.columns:  
 raise ValueError("Both DataFrames must have the 'id' column.")  
df.set\_index("id", inplace=True)  
new\_df.set\_index("id", inplace=True)

Se utiliza la función “update” y “concat” para la concatenación de las filas de cada DataFrame. También restablece los “id” del dataframe.

df.update(new\_df)

df = pd.concat([df, new\_df[~new\_df.index.isin(df.index)]])

df.reset\_index(inplace=True)

Luego se almacena el DateFrame actualizado en el mismo directorio de la Base de Datos y se avisa al usuario sobre el cambio realizado.

df.to\_csv("database\\techpowerup\_gpus.csv", index=False)

print("The Database has been updated")

Por último se guarda en el logs la actualización realizada.

save\_current\_action("[UPDATE] The working database has been updated")

Tener en consideración que se maneja en el caso de que halla algún error en el proceso.

except Exception as e:

        print(f"Error: {e}")

Exit:

Esta función es bastante simple, se encarga de terminar el proceso en el que se corre el programa mediante un exit(0), al ingresar al case “Exit”.

case "Exit":  
 exit(0)

# Resultados

Restock:

En este ejemplo llegan 5 unidades de Intel HD Graphics desde Rusia.

Texto

Descripción generada automáticamente

En el archivo logs.txt, se registra el siguiente mensaje:

10/12/2023 23:48:27 - User: alanb - Action: [RESTOCK] The product (id: 43) Intel HD Graphics has been restocked. 5 new units were received from Russia. Now it has 8 units

Distribute:

En este ejemplo se distribuyeron 2 unidades de Intel HD Graphics hacia China.

Texto

Descripción generada automáticamente

En el archivo logs.txt, se registra el siguiente mensaje:

10/12/2023 23:37:30 - User: alanb - Action: [DISTRIBUTE] The product (id: 43) Intel HD Graphics has been distributed. 2 units were sent to China. Remaining stock: 1

Check:

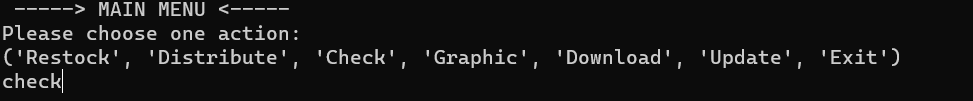


Imagen que contiene exterior, firmar, calle, grande

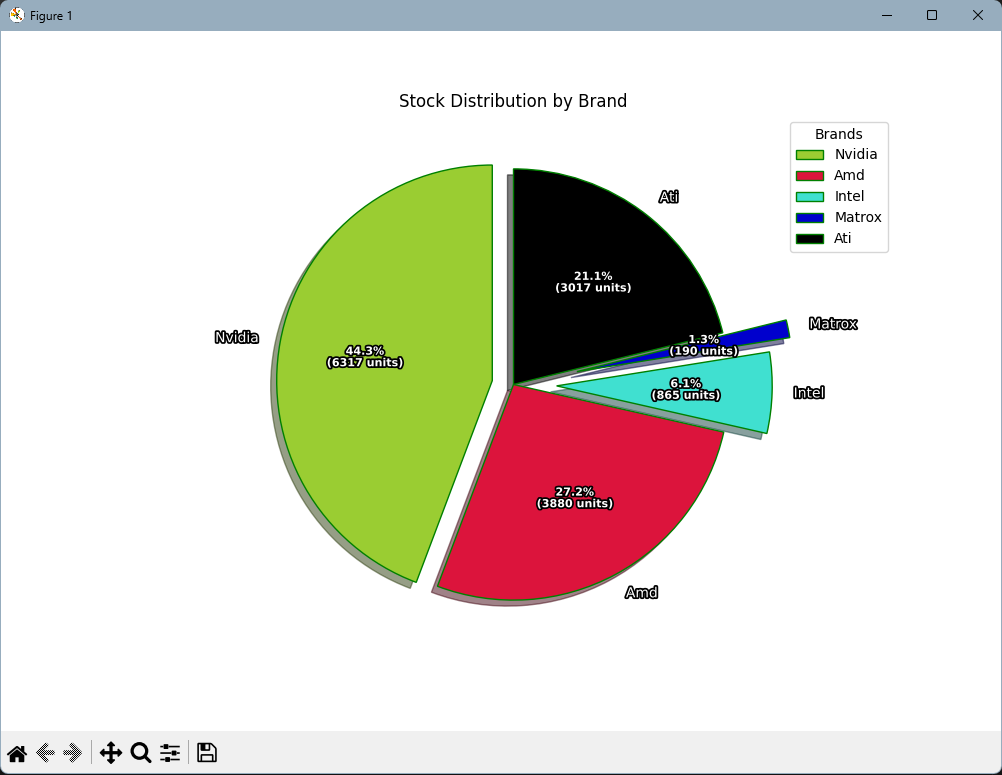
Descripción generada automáticamente

Al entrar a la función check, se mostrarán las últimas 7 lineas del archivo logs.txt. Como dice abajo del todo, para navegar por este texto, se usa ‘+’ para subir, ‘-’ para bajar y ‘q’ para salir del modo de vista.

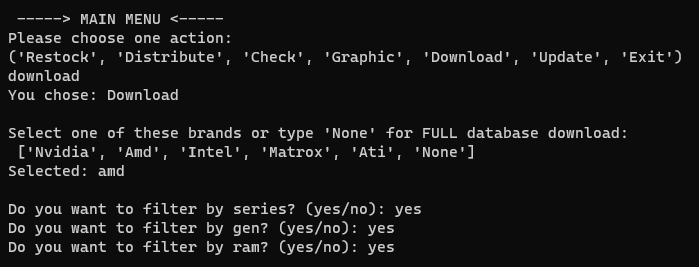
Graphic:

Texto

Descripción generada automáticamente

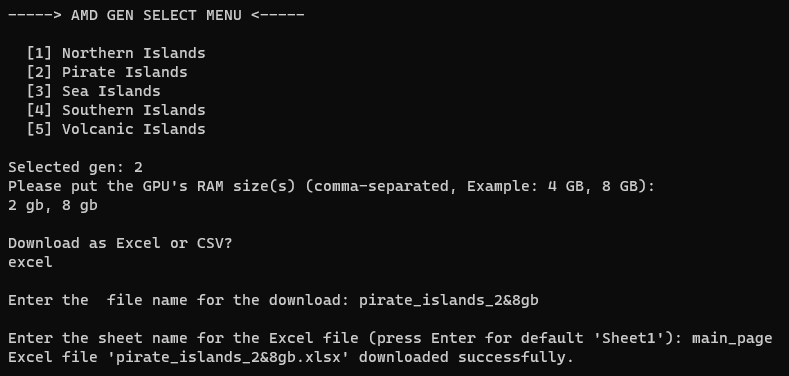


Download:



Texto

Descripción generada automáticamente



Update:

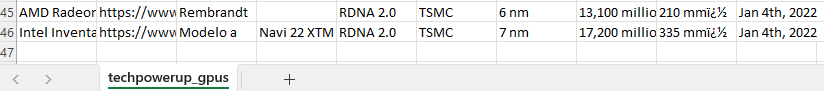
En el siguiente ejemplo se actualizará la base de datos a partir de una nueva llamada “new”, que tiene una fila y columna, llamada verify\_stock, más.

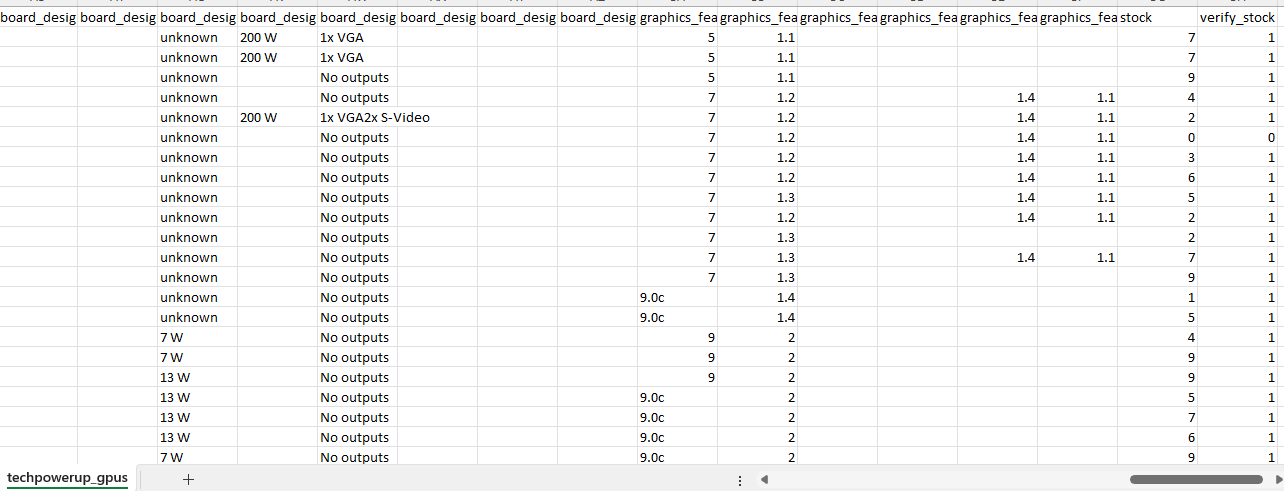


Imagen que contiene muebles, gabinete, luz, calle

Descripción generada automáticamente

Y el resultado es el siguiente:





# Conclusiones

Como conclusión podemos afirmar que los procesos de inventariado son más complicados de lo que aparentan a primera vista en aspectos de código. Se necesita una logística fuerte para que todos los procesos correspondientes a cada área del código funcionen bien de manera conjunta y den los resultados esperados.

Las funciones como “Distribute” o “Restock”, no supusieron una gran dificultad su implementación debido a la simplicidad del proceso. Lo mismo sucede con “Check”, “Update” o “Exit”, ya que se necesitó un poco de investigación de las funciones de las bibliotecas utilizadas para saber cómo utilizarlas adecuadamente y obtener los resultados esperados.

“Graphic” es un caso especial, ya que se requiere de una estructura fija para poder llevar a cabo el proceso de creación de los gráficos, por lo tanto, al saber esta estructura no conlleva problemas mucho más allá de la personalización que cada uno quiera agregar al resultado final.

“Filter” resultó ser lo más complicado al momento de programarlo, ya que poder llegar al resultado deseado fue más difícil de lo esperado. La logística dentro de cada una de las subfunciones y sus relaciones entre ellas debían ser llevadas a cabo con cuidado, por que cualquier detalle podía llevar a un escenario inesperado. También el funcionamiento interno de las funciones y los problemas con la estructura o redacción del Csv usado como base de datos llevaron a obstáculos para lograr el objetivo final.

En resumen, aún hay mucho que probablemente sea mejorable u optimizable, pero por cuestiones de tiempo no fue posible llevarlo a una mayor profundidad, y para proyectos profesionales es necesario ser cauteloso con cada uno de los posibles cambios a realizar o la logística de la programación.

Si desea ver el código más a detalle está disponible en el link “GitHub” de las referencias.

# Referencias

<https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-working-with-csv-files-in-python-with-pandas/>

<https://www.geeksforgeeks.org/update-column-value-of-csv-in-python/>

<https://www.geeksforgeeks.org/plot-a-pie-chart-in-python-using-matplotlib/>

<https://matplotlib.org/stable/gallery/color/named_colors.html>

# Software e Implementaciones utilizadas

Lucidchart - <https://lucid.app/>

GitHub - <https://github.com/LBrownI/Evaluacion3_Equipo1>  
Python - [Welcome to Python.org](https://www.python.org/)

pandas - [pandas - Python Data Analysis Library (pydata.org)](https://pandas.pydata.org/)

Replit - <https://replit.com/>

Visual Studio Code - [Visual Studio Code - Code Editing. Redefined](https://code.visualstudio.com/)  
MatPlotLib - <https://matplotlib.org/stable/index.html>

OpenPyXl - <https://openpyxl.readthedocs.io/en/stable/>