Septiembre 2021

LIFESTORE

Reporte de caso práctico (LifeStore)

Proyecto 1. Introducción a Python

Autor: Gabriela del Carmen
Barron Orozco
baorz.gab@gmail.com
GitHub: baorz-gab

Índice

1. Introducción

La tienda virtual LifeStore tiene una amplia variedad de artículos a la venta, pero, recientemente se han disminuido las ventas y por ende, se ha generado una acumulación de inventario considerable; por lo que, la administración quiere una estrategia de productos a retirar del mercado y de igual forma sugerencias para reducir la acumulación de inventario.

Para hallar una solución a este problema se decidió generar un análisis de la rotación de productos, el cual parte de clasificar y analizar los datos que se tienen en el historial de ventas y búsqueda de productos, mediante un programa escrito en Python se generaron nuevas listas que nos ayudaran a visualizar mejor la información, para posteriormente dar solución a la problemática presente; a continuación se muestra el proceso que se siguió.

2. Definición del código

Este proyecto o caso practico consta de 3 archivos, el archivo *lifestore_file.py* nos fue proporcionado y contiene 3 listas, cada una donde almacena datos específicos sobre cada producto, ventas y búsquedas. Estas listas tienen los siguientes atributos:

```
"""
lifestore_searches = [id_search, id product]
lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund
    (1 for true or 0 to false)]
lifestore_products = [id_product, name, price, category, stock]
"""
```

Listing 1: LifeStore data

El segundo archivo, *lifestore_consignas.py* contiene los siguientes procesos:

- Total ventas de cada producto.
- Total búsquedas de cada producto.
- Búsquedas y ventas por categoría.
- Reseña o *score* por producto.
- Ingresos mensuales y anuales.

Antes de comenzar a explicar el código de este archivo, es importante aclarar que el objetivo de este caso practico era hacer uso de los fundamentos de la programación en Python (uso de operadores lógicos, listas, bucles, etc); teniendo esto en cuenta todo este archivo esta escrito siguiendo ese objetivo.

Para comenzar, debemos importar las librerías correspondientes:

```
#importamos las listas de donde sacaremos los datos
from lifestore_file import lifestore_searches
from lifestore_file import lifestore_sales
from lifestore_file import lifestore_products
#libreria para manipular fechas
from datetime import datetime
#interactuar con el sistema operativo
```

```
8 import os
```

Listing 2: Librerías

2.1. Total ventas de cada producto

Para generar la lista de los productos mas vendidos, se debe contar cuantas veces se vende cada producto y posteriormente ordenar la lista de mayor a menor. Para eso, debemos recorrer la lista *'lifestore_products'* con un bucle *for*, dentro de cada iteración recorreremos la lista *'lifestore_sales'* con un bucle *for* para hacer el conteo de ventas de cada producto, para finalmente llenar una lista con el total de ventas por producto.

```
1 lista_top_ventas = [] #definimos una lista para las ventas
2 for producto in lifestore_products: #vamos producto por producto
3     veces = 0 #iniciamos contador del producto actual
4     for venta in lifestore_sales: #recorremos para contar ventas por
    producto
5         if producto[0] == venta[1]: #verificar si id_product in producto =
     id_product in venta
6             veces += 1 #actualizamos contador de ventas
7     #agregamos valores -> lista_top_ventas = [id_product, name, total
    ventas]
8     lista_top_ventas.append([producto[0],producto[1][0:producto[1].index('
    ,')],veces])
```

Listing 3: Lista top ventas

2.2. Total búsquedas de cada productos

El proceso para generar los productos mas buscados es bastante similar al anterior, solo que en vez de recorrer la lista *'lifesotre_sales'*, se recorre la lista *'lifestore_searches'*

```
1 lista_top_busqueda = [] #Definimos una lista para las busquedas
2 for producto in lifestore_products: #vamos producto por producto
3     veces = 0 #iniciamos contador del producto actual
4     for busqueda in lifestore_searches: #recorremos para contar busquedas
    por producto
5         if producto[0] == busqueda[1]: #verificar si id_product in
    producto = id_product in busqueda
6             veces += 1 #actualizamos contador de busquedas
7     #agregamos valores -> lista_top_busqueda = [id_product, name, total
    busquedas]
8     lista_top_busqueda.append([producto[0], producto[1][0:producto[1].
    index(',')], veces])
```

Listing 4: Lista top búsquedas

2.3. Búsquedas y ventas por categoría

Para generar la lista con los productos separados por categoría, lo primero que debemos hacer es poner en una lista los productos existentes junto con el número total de ventas y

búsquedas, gracias a que ya obtuvimos una lista para el total de ventas y de búsquedas esto sera mas sencillo.

```
lista_ord_top_ventas = sorted(lista_top_ventas, key = lambda x: x[0]) #ordenamos tomando id_product como parametro de ordenamiento
lista_ord_top_busqueda = sorted(lista_top_busqueda, key = lambda x: x[0]) #ordenamos tomando id_product como parametro de ordenamiento
lista_ventas_busqueda = [] #Definimos una lista para las ventas y busquedas totales de cada producto.
for producto in lista_ord_top_ventas: #recorremos para llenar nuestra nueva lista
    #agregamos valores -> lista_ventas_busqueda = [id_product, name, total ventas, total busquedas]
    lista_ventas_busqueda.append([producto[0],producto[1], producto[2], lista_ord_top_busqueda[producto[0]-1][2]])
```

Listing 5: Lista ventas y busqueda

Ahora, para separar por categoría, es necesario saber cuantas categorías existen, por lo que recorreremos la lista ordenada *'lifestore_products'* tomando la frecuencia con la que se repite la palabra en el atributo "*category*" como auxiliar para saber cuantos índices debemos saltar para pasar a la siguiente categoría; mientras recorremos iremos agregando datos a la lista final.

```
j = 0 #inicializamos un contador
lista_categorias = []  #lista para las categorias
lista_ord_prod = sorted(lifestore_products, key = lambda x: x[3]) #ordenamos conforme al atributo category
lista_top_categoria = [] #lista total de ventas y busquedas por categora
while j <= len(lista_ord_prod)-1: #Bucle para recorrer la lista ya ordenada
    lista_prod_in_cat = [] #lista para productos por categoria
    busqueda_categoria = 0 #Inicializamos un contador para las busquedas totales por categoria
    ventas_categoria = 0 #Inicializamos un contador para las ventas totales por categoria
    frecuencia = sum(x.count(lista_ord_prod[j][3]) for x in lista_ord_prod) #cuantas veces se repite la categoria
    for i in range(frecuencia): #Bucle para recorrer cada producto que esta en la categoria actual.
        #agregamos valores -> lista_prod_in_cat=[id_product, name,total ventas, total busquedas, stock]
        lista_prod_in_cat.append([lifestore_products[lista_ord_prod[j][0]-1][0], lista_ventas_busqueda[lista_ord_prod[j][0]-1][1], lista_ventas_busqueda[lista_ord_prod[j][0]-1][2],lista_ventas_busqueda[lista_ord_prod[j][0]-1][3],lifestore_products[lista_ord_prod[j][0]-1][4] ])
        busqueda_categoria += lista_ventas_busqueda[lista_ord_prod[j][0]-1][3] #Actualizamos conteo
        ventas_categoria += lista_ventas_busqueda[lista_ord_prod[j][0]-1][2] #Actualizamos conteo
        j += 1 #Actualizamos contador
        lista_prod_in_cat.sort(key = lambda x: (x[2],x[3]), reverse = False) #ordenamos de forma ascendente tomando como parametro total de busquedas
    #agregamos categoria a la lista
```

```
18      lista_categorias.append(lista_ord_prod[j-1][3])
19      #agregamos valores -> lista_busqueda_categoria =[categoria,
        lista_prod_in_cat,total ventas por categoria,total busquedas por
        categoria]
20      lista_top_categoria.append([lista_ord_prod[j-1][3], lista_prod_in_cat,
        ventas_categoria, busqueda_categoria])
```

Listing 6: Lista productos por categoría

2.4. Reseña o *score* por producto

Para generar la lista con la reseña de cada producto, debemos obtener el promedio de la suma total del atributo *'score'* en la lista *'lifesotre_sales'*. Para hallar ese promedio el proceso es el mismo que el que se realizo en la sección 2.1, solo que aquí también contaremos el *'score'* para posteriormente sacar el promedio de la suma, como dato adicional guardaremos el total de devoluciones.

```
1  lista_top_resena = [] #lista para el promedio de score
2  for producto in lifestore_products: #Bucle para recorrer producto a
       producto
3      veces = 0 #Inicializamos contador de ventas
4      suma = 0 #Inicializamos contador del score
5      devoluciones = 0 #Inicializamos contador de devoluciones
6      for resena in lifestore_sales: #Bucle para tomar los datos de scrore y
       refund
7          if producto[0] == resena[1]: #Condicion, si id_product de
       lifestore_products = id_product de lifestore_sales
8              veces += 1 #En caso de que se cumpla la condicion,
       actualizamos  contador
9              suma += resena[2] #Actualizamos conteo
10             devoluciones += resena[4] #Actualizamos conteo
11     if veces == 0: #Cuando se termina de recorrer la lista para el
       id_producto actual hacemos una condicion ya que no tiene sentido
       rankear por score productos que no tienen ventas
12         continue
13     else: #En otro caso hacemos la operacion del promedio
14         promedio = round(suma/veces,2)
15 #agregamos valores -> lista_top_resena = [id_product, name, promedio score
       , total ventas, total devoluciones]
16     lista_top_resena.append([producto[0],producto[1][0:producto[1].index('
       ,')], promedio, veces, devoluciones])
```

Listing 7: Lista productos por reseña

2.5. Ingresos mensuales y anuales

Para saber el ingreso mensual y anual, lo primero que debemos hacer es ordenar la lista *'lifestore_sales'* conforme a las fechas de menor a mayor, para obtener el año de la primer venta y el de la última.

```
1  lista_anio = sorted(lifestore_sales,key=lambda date: datetime.strptime(
       date[3], '%d/%m/%Y')) #creamos una lista con las fechas ordenadas de
       forma ascendente
```

```
anio = int(lista_anio[0][3][6:10]) #Definimos anio que ira recorriendo la lista, en este caso sera anio que esta en el primer elemento de la lista
anio_fin = int(lista_anio[len(lista_anio)-1][3][6:10]) #Definimos el anio en el que termina la lista
ventas_anio = [] #Definimos lista para las ventas anuales y mensuales
lista_anios = [] #Definimos lista para guardar los anios
total_ingresos = 0 #Definimos la variable que tendra el total de ingresos
```

Listing 8: definiendo año de inicio y fin

Una vez que tenemos el año de inicio y de fin, con un bucle *while* iremos avanzando de año para ir haciendo el conteo de ingresos anual, para el conteo mensual haremos uso de un bucle *for*.

```
while anio <= anio_fin: #Bucle, mientras anio sea menor o igual al anio final
    ventas_mes = [] #Definimos lista para ventas mensuales por anio
    sum_anio_venta = 0 #Inicializamos contador suma de ventas anuales
    for mes in range(1,13): #Segundo bucle que recorrera mes por mes
        sum_mes_venta = 0 #Inicializamos nuestro contador para la suma de ventas mensual
        fecha = str(mes) + '/' + str(anio) #Definimos la cadena de fecha a buscar en la lista lifestore_sales
        for producto in lifestore_sales: #Con la fecha definida, recorremos la lista de ventas para encontrar las ventas mensuales y anuales.
            if fecha in producto[3] and producto[4] == 0: #Condicion, si valor de fecha esta en indice de la lista y no hay devolucion
                sum_mes_venta += lifestore_products[producto[1]-1][2] #Actualizamos conteo
        ventas_mes.append([mes, sum_mes_venta]) #Agregamos valores -> ventas_mes = [mes, ingreso mensual]
        sum_anio_venta += sum_mes_venta #actualizamos la suma de ventas anuales

    lista_anios.append(anio) #Agregamos el anio actual a la lista de anios.
    ventas_anio.append([anio, sum_anio_venta, ventas_mes]) #Agregamos valores -> ventas_anio = [anio, ingreso anual, ventas_mes]
    anio += 1 #actualizamos nuestro indice para el anio
    total_ingresos += sum_anio_venta #Actualizamos el total de ingresos
```

Listing 9: Lista ingresos mensuales y anuales

2.6. Interfaz gráfica

El tercer archivo, *'PROYECTO-01-BARRON-GABRIELA.py'*, es el archivo que desplegará la interfaz gráfica, la cual nos dejara visualizar en forma de tablas los procesos que se realizaron en el archivo *'lifestore_consignas.py'*.

Lo primero es importar las librerias que vamos a ocupar, para desplegar la GUI usaremos la paquetería *tkinter*.

```
from tkinter import *
import tkinter as tk
```

```
from tkinter import ttk
#importamos  del archivo lifestore_consignas las variables que usaremos
from lifestore_consignas import lista_top_ventas, lista_top_busqueda,
    lista_top_resena, lista_top_categoria, lista_categorias, lista_anios,
    ventas_anio
```

Listing 10: Librerías y variables a usar.

2.6.1. Ventana inicio de sesión

Para desplegar la ventana de inicio de sesión, primero debemos definir una función que valide el llenado de las celdas.

```
#definimos la funcion de login
def login():
    #tomamos los valores que se ingresaron de usuario y contrasena
    uname=username.get()
    pwd=password.get()
    #validaciones
    if uname=='' or pwd=='': #si no se llenaron los espacios mandamos
    mensaje
        message.set("Llena el espacio vacio")
    else:
      if uname=="usuario" and pwd=="contrasena": #si el usuario y
    contrasena son los correctos
        reporte() #llamamos la funcion que abrira la siguiente ventana
      else: #si no, mandamos mensaje
       message.set("Usuario y/o contrasena incorrectos")
```

Listing 11: Función Login

Lo siguiente es la función que despliega la ventana de inicio de sesión (cabe aclarar que aun no la mandamos llamar, eso es hasta el final), esta función da diseño a la ventana, define el posicionamiento de las etiquetas y elementos, además de que manda a llamar a la función login para verificar los datos ingresados.

```
#funcion ventana de login
def Loginform():
    global login_screen #definimos variable global para una ventana
    login_screen = tk.Tk()
    #Titulo o nombre de la ventana
    login_screen.title("Inicio Sesion")
    #Definimos el tamano de la ventana
    login_screen.geometry("300x250")
    #declaramos las varaibles a usar
    global  message;
    global username
    global password
    username = StringVar() #definimos entradas
    password = StringVar()
    message=StringVar()
    #Diseno de la ventana
    tk.Label(login_screen,width="300", text="Ingrese su usuario y
    contrasena").pack()
    tk.Label(login_screen, text="Usuario * ").place(x=20,y=40)#Etiqueta de
    usuario, def posicion
```

```
19      tk.Entry(login_screen, textvariable=username).place(x=90,y=42) #Caja
    de texto donde se escribe el usuario
20      tk.Label(login_screen, text="Contrasena * ").place(x=20,y=80) #
    Etiqueta de contrasena, def posicion
21      tk.Entry(login_screen, textvariable=password ,show="*").place(x=90,y
    =82) #Caja de texto donde se escribe la contrasena
22      tk.Label(login_screen, text="",textvariable=message).place(x=95,y=100)
     #Etiqueta, despliega la validacion de los datos ingresados
23      tk.Button(login_screen, text="Aceptar", width=10, height=1,command=
    login).place(x=105,y=130) #Boton para validar login, def posicion
24      login_screen.mainloop()
```

Listing 12: Ventana Login

La interfaz gráfica de la ventana de inicio de sesión luce de la siguiente forma:

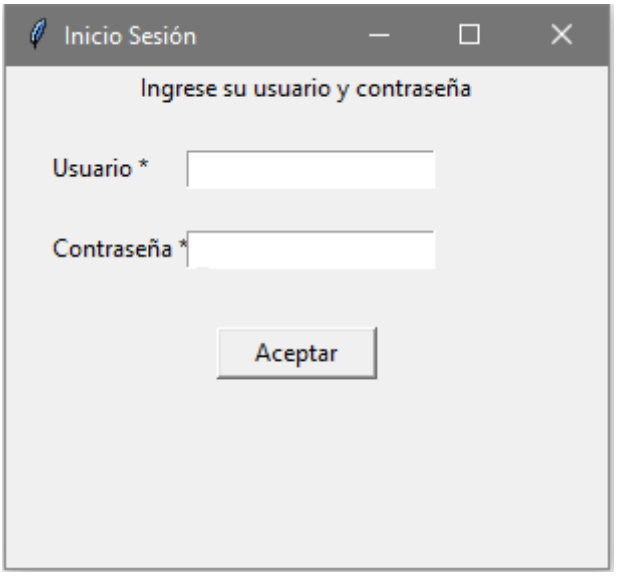

Figura 1: Ventana de Inicio de Sesión

2.6.2. Ventana reporte de productos

Esta ventana se despliega cuando el login es exitoso, para esa ventana definimos una sola función *reporte()*, dentro de la función están las funciones para los comandos que realizara cada opción del menú; ya que, la función es bastante grande, primero mostraremos el código de las funciones que están dentro de esta función.

La primer función es la que desplegará la tabla con los 20 productos mas vendidos, en este función llamamos a la lista *'lista_top_ventas'* que obtuvimos en el archivo *'lifestore_consignas.py'*.

```
1 #Funciones que desplegaran las tablas a visualizar
2     def top_ventas(): #funcion top ventas
3         lista_top_ventas.sort(key = lambda x: x[2], reverse= True) #
    ordenamos nuestra lista en orden descendente conforme al atributo total
     ventas
```

```
        tk.Label(ventana, text="   PRODUCTOS MAS VENDIDOS    ", font=("Arial",15)).grid(row=0, columnspan=3) #Etiqueta, def posicion
        cols = (['',40], ['Id',40],['Nombre',600], ['Total ventas',100]) #Definimos el tamano que tendra cada columna
        listBox = ttk.Treeview(ventana, columns=[x[0] for x in cols], show='headings', heigh = 20)  #Definimos la tabla que se desplegara, def tamano
        for i, (id_, name, score) in enumerate(lista_top_ventas, start=1): #Llenado de tabla
                listBox.insert("", "end", values=(i, id_, name, score))
                if i == 10: #condicion de paro
                  break
        for col in cols:  #Imprime tabla
            listBox.heading(col[0], text=col[0])
            listBox.column(col[0], width=col[1])
        listBox.grid(row=1, column=0, columnspan=2)
```

Listing 13: Función top ventas (GUI)

La segunda función despliega la tabla con los 20 productos mas buscados, para poder desplegar esta tabla debemos llamar a la función *'lista_top_busqueda'* del archivo *'lifestore_consignas.py'*.

```
def top_busquedas(): #funcion top busquedas
        lista_top_busqueda.sort(key = lambda x: x[2], reverse= True) #ordenamos nuestra lista en orden descendente conforme al atributo total busquedas
        tk.Label(ventana, text="   PRODUCTOS MAS BUSCADOS    ", font=("Arial",15)).grid(row=0, columnspan=3) #Etiqueta, def posicion
        cols = (['',40], ['Id',40],['Nombre',500], ['Total busquedas',200]) #Definimos el tamano que tendra cada columna
        listBox = ttk.Treeview(ventana, columns=[x[0] for x in cols], show='headings', heigh = 20) #Definimos la tabla que se desplegara, def tamano
        for i, (id_, name, score) in enumerate(lista_top_busqueda, start=1): #Llenado de tabla
            listBox.insert("", "end", values=(i, id_, name, score))
            if i == 10: #condicion de paro
              break
        for col in cols:  #Imprime tabla
            listBox.heading(col[0], text=col[0])
            listBox.column(col[0], width=col[1])
        listBox.grid(row=1, column=0, columnspan=2)
```

Listing 14: Función top búsquedas (GUI)

Las siguientes dos funciones son muy parecidas, lo único que cambia es el ordenamiento en la lista *'lista_top_reseña'*, una la ordena de menor a mayor, mientras que la otra de mayor a menor.

```
def top_resena(): #funcion top resena (mayor)
        #ordenamos nuestra lista de forma descendente, tomando como primer parametro el promedio de score, despues por el numero de ventas
        lista_top_resena.sort(key = lambda x: (x[2],x[3]), reverse= True) #Etiqueta, def posicion
        tk.Label(ventana, text="PRODUCTOS CON MEJOR RESENA", font=("Arial",15)).grid(row=0, columnspan=3) #Etiqueta, def posicion
```

```
        cols = (['',30], ['Id',30],['Nombre',500], ['Score',60],['Ventas'
    ,60],['Devoluciones',100]) #Definimos el tamano que tendra cada columna
        listBox = ttk.Treeview(ventana, columns=[x[0] for x in cols], show
    ='headings', heigh = 20) #Definimos la tabla que se desplegara, def
    tamano
        for i, (id_, name, score,ventas, devoluciones) in enumerate(
    lista_top_resena, start=1): #Llenado de tabla
            listBox.insert("", "end", values=(i,id_, name, score,ventas,
    devoluciones))
            if i == 20: #condicion de paro
              break
        for col in cols: #Imprime tabla
            listBox.heading(col[0], text=col[0])
            listBox.column(col[0], width=col[1])
        listBox.grid(row=1, column=0, columnspan=2)

     def top_resena2(): #funcion top resena (menor)
        #ordenamos nuestra lista de forma ascendente, tomando como primer
    parametro el promedio de score, despues por el numero de ventas
        lista_top_resena.sort(key = lambda x: (x[2],x[3]), reverse= False)
        tk.Label(ventana, text="PRODUCTOS CON PEOR RESENA", font=("Arial"
    ,15)).grid(row=0, columnspan=3) #Etiqueta, def posicion
        cols = (['',30], ['Id',30],['Nombre',500], ['Score',60],['Ventas'
    ,60],['Devoluciones',100]) #Definimos el tamano que tendra cada columna
        listBox = ttk.Treeview(ventana, columns=[x[0] for x in cols], show
    ='headings', heigh = 20)  #Definimos la tabla que se desplegara, def
    tamano
        for i, (id_, name, score,ventas, devoluciones) in enumerate(
    lista_top_resena, start=1):  #Llenado de tabla
            listBox.insert("", "end", values=(i,id_, name, score,ventas,
    devoluciones))
            if i == 20: #condicion de paro
              break
        for col in cols: #Imprime tabla
            listBox.heading(col[0], text=col[0])
            listBox.column(col[0], width=col[1])
        listBox.grid(row=1, column=0, columnspan=2)
```

Listing 15: Función top reseña (GUI)

Para la función que desplegara el top por categoría, lo primero que se debe hacer es asignarle un valor a la opción que se escogió, para después tomar el índice en el que se ubica en la lista *'lista_top_categoria'* y así poder desplegar la tablar de dicha categoría.

```
def top_categoria(): #funcion top por categoria
      indice = 0 #variable para guardar el indice de la opcion que se
    escogio
      categoria = categoria_sel.get() #asignamos a la variable la opcion
    que se escogio
      for cat in lista_top_categoria: #bucle, para buscar el indice exacto
     en donde esta la opcion que se escogio
        if cat[0] == categoria: #condicion, si categoria en cat = a
    categoria que se ingreso
          indice = lista_top_categoria.index(cat) #asignamos el indice
    exacto en el que se encuentra
        else: #si no se cumple pasamos a la siguiente iteracion
```

```
            continue
        tk.Label(ventana, text="                          "+categoria+"
                ", font=("Arial",15)).grid(row=0, columnspan=3) #Etiqueta,
    def posicion
        cols = ([’’,30], [’Id’,30],[’Nombre’,510], [’Ventas’,70],[’Busquedas
    ’,70],[’Stock’,70]) #Definimos el tamano que tendra cada columna
        listBox = ttk.Treeview(ventana, columns=[x[0] for x in cols], show=’
    headings’, heigh = 20)#Definimos la tabla que se desplegara, def tamano
        for i, (id_, name, ventas, busquedas, stock) in enumerate(
    lista_top_categoria[indice][1], start=1): #Llenado de tabla
            listBox.insert("", "end", values=(i, id_, name,ventas, busquedas
    , stock))

        for col in cols: #Imprime tabla
            listBox.heading(col[0], text=col[0])
            listBox.column(col[0], width=col[1])
        listBox.grid(row=1, column=0, columnspan=2)
```

Listing 16: Función top por categoría (GUI)

La función de ingresos anuales es muy parecida a la anterior, tomamos el valor de la opción que se elige para hallar el índice en el que se encuentra dentro de la lista *‘ventas_año’* para posteriormente desplegar la lista correspondiente a los ingresos de ese año.

```
def ingreso_anual(): #funcion ingresos mensuales
        indice = 0 #variable para guardar el indice de la opcion que se
    escogio
        anio = anio_sel.get() #asignamos a la variable la opcion que se
    escogio
        for year in ventas_anio: #bucle, para buscar el indice exacto en
    donde esta la opcion que se escogio
            if str(year[0]) == str(anio): #condicion, si anio en year = a anio
     elegido por el usuario
                indice = ventas_anio.index(year) #asignamos el indice exacto en
    el que se encuentra
            else: #si no se cumple pasamos a la siguiente iteracion
                continue
     #definimos una lista para dar mejor visibilidad a los meses
        meses = [’Enero’, ’Febrero’, ’Marzo’, ’Abril’, ’Mayo’, ’Junio’, ’
    Julio’, ’Agosto’, ’Septiembre’, ’Octubre’, ’Noviembre’, ’Diciembre’]
        tk.Label(ventana, text="  INGRESO MENSUAL ANIO:  "+ anio , font=("
    Arial",15)).grid(row=0, columnspan=3) #Etiqueta, def posicion
        cols = ([’Mes’,390], [’Ingreso mensual’,390]) #Definimos el tamanio
    que tendra cada columna
        listBox = ttk.Treeview(ventana, columns=[x[0] for x in cols], show=’
    headings’, heigh = 20)  #Definimos la tabla que se desplegara, def
    tamanio
        for i, (mes, ingresos) in enumerate(ventas_anio[indice][2], start=1)
    : #Llenado de tabla
            listBox.insert("", "end", values=(meses[i-1], ingresos))

        for col in cols: #Imprime tabla
            listBox.heading(col[0], text=col[0])
            listBox.column(col[0], width=col[1])
        listBox.grid(row=1, column=0, columnspan=2)
        #tk.Label(ventana, text="Total anual: "+ str(ventas_anio[indice][1])
```

```
    , font=("Arial",15)).grid(row=22, columnspan=3)
```

Listing 17: Función ingresos mensuales por año (GUI)

A partir de aquí se define la ventana que desplegará la interfaz gráfica, la ubicación del menú y las funciones a las que llamara cada opción del menú.

```
#desplegando ventana con menu de opciones para desplegar tablas
    global ventana
    login_screen.state(newstate = "withdraw") #una vez que el login es exitoso, suspendemos la ventana login y mostramos la ventana de reporte
    ventana = tk.Tk() #asignamos la variable ventana
    ventana.title("Reporte Lifestore") #Asignamos titulo a ventana
    ventana.geometry("800x500") #tamano de ventana
```

Listing 18: Menú de opciones (GUI)

Ya que en nuestro menú tendremos submenus para la selección de categoria, top reseña y para ver los ingresos anuales, debemos definir variables que guardaran la selección.

```
    categoria_sel = tk.StringVar(ventana) #definimos una variable para la seleccion de categorias
    categoria_sel.set(lista_categorias[0]) #le damos un valor default

    resena_sel = tk.StringVar(ventana) #definimos una variable para la seleccion de categorias
    resena_sel.set(lista_categorias[0]) #le damos un valor default

    anio_sel = tk.StringVar(ventana) #definimos una variable para la seleccion de categorias
    anio_sel.set(lista_anios[0]) #le damos un valor default

    menu_lifestore = tk.Menu(ventana) #asignamos a una variable al menu de opciones
```

Listing 19: Variables que guardaran la selección de submenus (GUI)

Ya que definimos esas variables, debemos ponerle nombre a las etiquetas con las opciones para cada menú tipo cascada

```
    categoria_menu = tk.Menu(menu_lifestore, tearoff = 0) #menu tipo cascada o submenu para la lista de categorias
    for categoria in lista_categorias: #asignamos las etiquetas al submenu de categorias
      categoria_menu.add_radiobutton(label = categoria, value = categoria, variable = categoria_sel, command = top_categoria )

    menu_resena = tk.Menu(menu_lifestore, tearoff = 0) #menu tipo cascada o submenu para las opciones top resena
    #asignamos las etiquedas al submenu de resena
    menu_resena.add_radiobutton(label = 'Mejor resena', value = 'Mejor resena', variable = resena_sel, command = top_resena)
    menu_resena.add_radiobutton(label = 'Peor resena', value = 'Peor resena', variable = resena_sel, command = top_resena2)

    anio_menu = tk.Menu(menu_lifestore, tearoff = 0) #menu tipo cascada o submenu para los anios
    for anio in lista_anios: #asignamos las etiquedas al submenu de anio
```

```
12      anio_menu.add_radiobutton(label = anio, value = anio, variable =
    anio_sel, command = ingreso_anual )
```

Listing 20: Etiquetas de submenus (GUI)

Así luce la ventana con el menú y submenus.

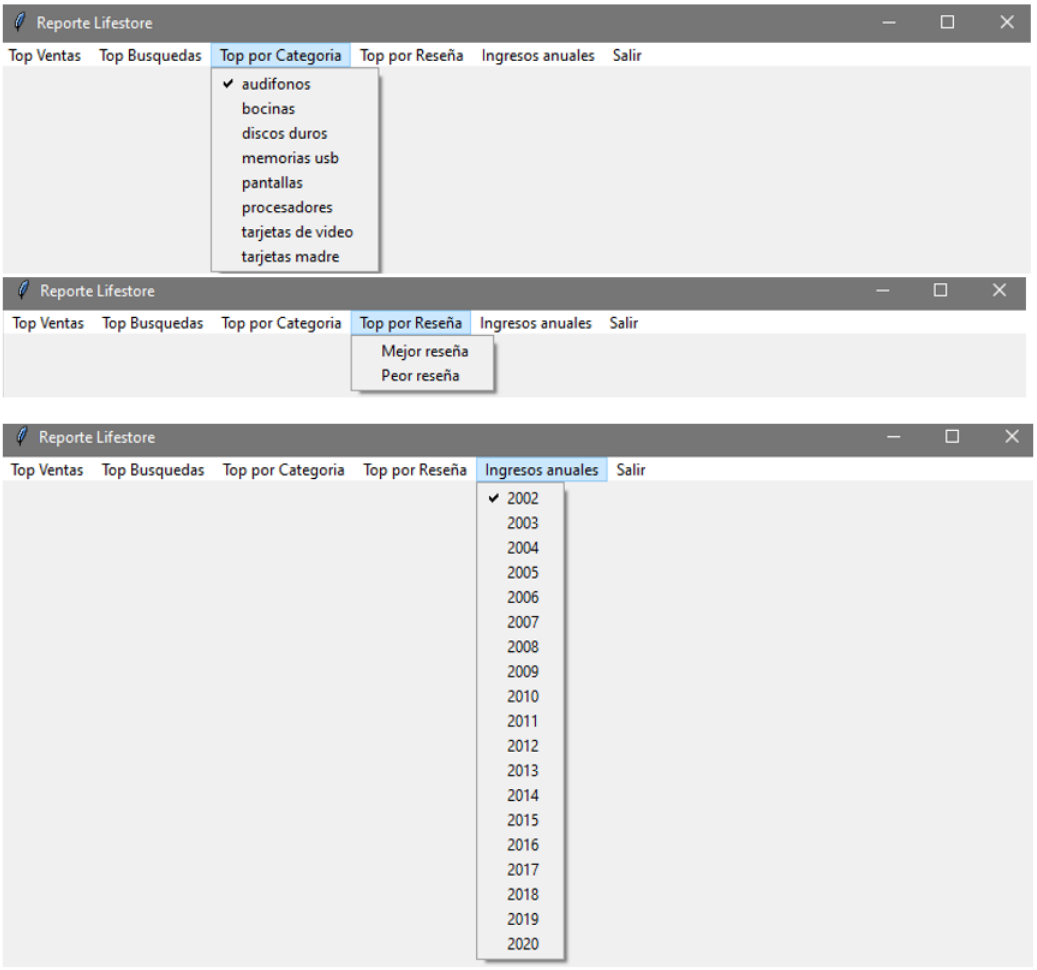

Figura 2: Ventana de Reporte

Por ultimo, le damos forma al menú, que opciones tendrá y las funciones o variables que mandara a llamar cada opción.

```
1     ventana.config(menu = menu_lifestore) #condiguramos la ventana y le
    asignamos el menu
2
3     #comandos que realiza cada pestana del menu, aqui se llama a las
    funciones que definimos arriba
4     menu_lifestore.add_command(label = 'Top Ventas', command=top_ventas )
5     menu_lifestore.add_command(label = 'Top Busquedas', command=
    top_busquedas)
6     menu_lifestore.add_cascade(label = 'Top por Categoria', menu =
    categoria_menu)
7     menu_lifestore.add_cascade(label = 'Top por Resena', menu =
    menu_resena)
```

```
8     menu_lifestore.add_cascade(label = 'Ingresos anuales', menu =
      anio_menu)
9     menu_lifestore.add_command(label = 'Salir', command = exit)
10    ventana.mainloop()
```

Listing 21: Etiquetas de menú (GUI)

Para finalizar, y que el programa despliegue la interfaz gráfica debemos mandar a llamar a la función *Loginform()* ya que es la función que empieza todo, es decir, despliega la ventana de inicio de sesión y posteriormente la ventana del reporte.

```
1     def reporte():
2     .
3     .
4     .
5     #aqui va las funciones y proceso para el menu que arriba explicamos.
6     .
7     .
8
9     #llamamos a la funcion loginform()
10    Loginform()
```

Listing 22: Desplegamos ventana de inicio de sesión (GUI)

Los archivos con el código que acabamos de explicar se encuentran en el siguiente url: `https://github.com/baorz-gab/CasoPractico1-Python/`GitHub

3. Solución al problema

Vamos a dar algunas propuestas para las siguientes problematicas:

1. Productos mas vendidos y productos rezagados
2. Productos por reseña
3. Estrategia de productos a retirar

No vamos a mostrar las listas completas que el programa despliega, solo para mostrar los productos por reseña, para no generar un reporte largo y que se pierda la información, pero si el lector gusta ir leyendo este reporte y viendo las listas del programa seria lo recomendable, ya que, en algunos puntos haremos uso de otras listas del menú.

3.1. Productos más vendidos y productos rezagados

3.1.1. Productos más vendidos

Viendo las tablas de productos mas vendidos y buscados, podemos decir que el producto estrella de la tienda es el disco duro **SSD Kingston A400** , pero el producto que mejor porcentaje de efectividad a la hora de buscarlo y comprarlo sería el **procesador AMD Ryzen 5 2600** de 55 búsquedas que tuvo, se completaron 42 ventas, también podemos notar que la **tarjeta de vídeo ASUS NVIDIA GeForce GTX 1660** no esta en el top de mas buscados ya que tiene un total de 15 búsquedas pero tiene 9 ventas, lo mismo pasa con los productos con el **id 42, 2 y 12**, cada uno tiene 23,24 y 15 búsquedas respectivamente; entonces, el top de productos mas vendidos queda de la siguiente manera.

Id	Producto	Ventas	Búsquedas
54	SSD Kingston A400	50	263
3	Procesador AMD Ryzen 5 2600	42	55
5	Procesador Intel Core i3-9100F	20	30
42	Tarjeta Madre ASRock Micro ATX B450M Steel Legend	18	23
57	SSD Adata Ultimate SU800	15	107
29	Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING	14	60
2	Procesador AMD Ryzen 5 3600	13	24
4	Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8	13	41
47	SSD XPG SX8200 Pro	13	30
12	Tarjeta de Video ASUS NVIDIA GeForce GTX 1660 SUPER EVO OC	11	15

Cuadro 1: Productos mas vendidos

Teniendo esto en mente seria bueno resurtir al producto con el **Id 42**, ya que, ya no cuenta con *stock* y es uno de los productos con mas ventas y mejor *score*, también se debería pedir mas piezas de los productos con **Id 12, 47 y 56**, ya que, son productos bien rankeados,tienen buenas ventas y también bastantes búsquedas, sería malo que un producto que tiene buenas búsquedas no tenga *stock*.

3.1.2. Productos rezagados

Para los productos rezagados se nos pidió ordenar los productos por categoría con menores ventas y búsquedas. Analizando por categoría tenemos los siguientes productos rezagados.

Para la categoría **audífonos**:

Id	Producto	Ventas	Búsquedas	Stock
92	Getttech Audífonos con Micrófono Sonority	0	0	232
93	Ginga Audífonos con Micrófono GI18ADJ01BT-RO	0	1	139
86	ASUS Audífonos Gamer ROG Theta	0	0	20
88	Audífonos Gamer Balam Rush Orphix RGB 7.1	0	0	15

Cuadro 2: Audífonos: Productos rezagados

A pesar de que hay otros productos con ninguna venta, no hay tanto *stock*, los demás están por debajo de 17 de *stock*, aunque hay una excepción, para el producto **Logitech**

Audífonos Gamer G332, ya que, aunque tiene 83 de *stock* y una venta, tiene 10 búsquedas y un *score* de 5.0 por lo que en un futuro puede que ese *stock* se convierta en ingresos.

Para la categoría **bocinas**:

Id	Producto	Ventas	Búsquedas	Stock
79	Naceb Bocina Portátil NA-0301	0	0	31
82	Ghia Bocina Portátil BX400	0	0	31
81	Ghia Bocina Portátil BX900	0	0	20

Cuadro 3: Bocinas: Productos rezagados

En general esta categoría tiene bastante merma de productos, solo tiene dos ventas, y en total 8 búsquedas; pero los 3 productos del cuadro son los que tienen mas *stock* ya que ni una búsqueda han tenido.

Para la categoría **discos duros**:

Id	Producto	Ventas	Búsquedas	Stock
58	SSD para Servidor Lenovo Thinksystem S4510	0	0	16
55	SSD para Servidor Supermicro SSD-DM128-SMCMVN1	0	0	310
56	SSD Samsung 860 EVO	0	1	10

Cuadro 4: Discos duro: Productos rezagados

Podemos notar que los disco duros para servidor son los que menos se venden y tienen un *stock* considerable, también en la tabla que despliega el programa, podemos ver que ya no hay *stock* para el disco duro con **id 51**, seria bueno resurtir, no es el producto con mas ventas pero tiene 11 búsquedas y su *score* es de 4.55, algo que si podemos agregar es que esta categoría genera bastantes ventas tres de sus productos rebasan las 10 ventas.

Para la categoría **memorias usb**:

Id	Producto	Ventas	Búsquedas	Stock
61	Kit Memoria RAM Corsair Dominator Platinum DDR4	0	0	5
60	Kit Memoria RAM Corsair Vengeance LPX DDR4	1	0	10

Cuadro 5: Memorias usb: Productos rezagados

Algo curioso pasa en esta categoría, solo hay una venta, pero, no hay búsquedas, no podemos decir que hay merma en esta categoría ya que no hay tanto *stock* pero si sería bueno que empezara a haber mas ventas, ya que, la única venta que se hizo de esta categoría tuvo un *score* de 5.0

Para la categoría: **pantallas**

Id	Producto	Ventas	Búsquedas	Stock
68	Kit Makena Smart TV LED 40S2 40"	0	0	239
69	Hisense Smart TV LED 40H5500F 39.5	0	0	94
64	Samsung TV LED LH43QMREBGCXGO 43	0	0	71
63	Seiki TV LED SC-39HS950N 38.5 43	0	4	146

Cuadro 6: Pantallas: Productos rezagados

En general esta categoría no tiene ventas, solo tiene dos ventas, y solo 2 de sus productos tienen buenas búsquedas, pero las pantallas mencionadas en el cuadro tienen bastante merma, aunque, a pesar de que hay bastante *stock* en el producto con **Id 63**, tiene 4 búsquedas que en un futuro con la gestión correcta se pueden transformar en ventas.

Para la categoría **procesadores**:

Id	Producto	Ventas	Búsquedas	Stock
9	Procesador Intel Core i3-8100	0	1	35

Cuadro 7: Procesadores: Productos rezagados

Esta categoría es la que mejor ventas en todos sus productos tiene, ya que solo uno no tiene ventas ni búsquedas (solo tiene una búsqueda), por lo que, podríamos decir que es el único producto rezagado; además de que todos los demás procesadores además de tener buenos números en ventas y búsquedas tienen buenas reseñas.

Para la categoría **tarjetas de video**:

Id	Producto	Ventas	Búsquedas	Stock
26	Tarjeta de Video VisionTek AMD Radeon HD 5450	0	5	180
27	Tarjeta de Video VisionTek AMD Radeon HD5450	0	1	43
14	Tarjeta de Video EVGA NVIDIA GeForce GT 710	0	0	36

Cuadro 8: Tarjetas de video: Productos rezagados

En esta categoría hay bastante variedad de productos, lamentablemente la mitad no tiene ventas y la otra mitad si, de igual forma los mas vendidos o que tienen mas de dos ventas son los que tienen mas búsquedas, pero si hay una merma considerable en los 3 productos del cuadro anterior, ya que, el *stock* en los demás productos no es tan grande, algo también a tener en cuenta es resurtir los productos con el **id 12, 18, 5 y 21** ya que tiene buenas reseñas, y números buenos en las búsquedas.

Para la categoría **tarjetas madre**:

Id	Producto	Ventas	Búsquedas	Stock
51	Tarjeta Madre ASUS micro ATX Prime H370M-Plus/CSM	0	0	286
50	ASUS T. Madre uATX M4A88T-M	0	3	98
48	Tarjeta Madre ASRock ATX Z490 STEEL LEGEND	0	0	60
43	Tarjeta Madre AORUS ATX Z390 ELITE	0	0	50

Cuadro 9: Tarjetas madre: Productos rezagados

Al igual que en la categoría anterior, la mitad de sus productos no tienen ni ventas ni búsquedas y la otra mirad tiene ventas y búsquedas, pero, si hay una merma considerable en los productos antes mencionados, ya que, el resto no pasa de las 30 piezas en *stock*; también algo a considerar es resurtir el producto con el **Id 44**, ya que, tiene 6 ventas, 25 búsquedas y un *score* de 4.67.

Estrategia para revertir la situación:

Una buena estrategia para estos productos rezagados seria ponerlos en oferta, hacer un cronograma para las ofertas, una semana ofertar ciertos productos, en la siguiente otros y así empezar a rotar productos; las ofertas no solo generaran mas visitas si no que empezaran también a comprar otros productos que tal vez no estén en oferta. Algo que puede complementar lo anterior, es implementar en la pagina la sección "también podría interesarte", cuando alguien entre a ver un producto o lo busque aparezca esta pestaña; y como ultimo recurso esta poner una promoción, en donde el cliente tenga que gastar cierta cantidad y de regalo se llevan un producto de los rezagados.

Esto se debe hacer con el seguimiento debido, se debe estar observando el comportamiento de las ventas, búsquedas y que tanto les gusta el producto a los clientes.

3.2. Productos por reseña en el servicio

3.2.1. Lista de productos con Mejor reseña

PRODUCTOS CON MEJOR RESEÑA

	Id	Nombre	Score	Ventas	Devoluciones
1	7	Procesador Intel Core i7-9700K	5.0	7	0
2	8	Procesador Intel Core i5-9600K	5.0	4	0
3	6	Procesador Intel Core i9-9900K	5.0	3	0
4	11	Tarjeta de Video ASUS AMD Radeon RX 570	5.0	3	0
5	49	Kit SSD Kingston KC600	5.0	3	0
6	1	Procesador AMD Ryzen 3 3300X S-AM4	5.0	2	0
7	21	Tarjeta de Video MSI AMD Mech Radeon RX 5500 XT MECH Gaming OC	5.0	2	0
8	25	Tarjeta de Video Sapphire AMD Pulse Radeon RX 5500 XT Gaming	5.0	2	0
9	52	SSD Western Digital WD Blue 3D NAND	5.0	2	0
10	85	Logitech Audífonos Gamer G635 7.1	5.0	2	0
11	22	Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC	5.0	1	0
12	28	Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti	5.0	1	0
13	40	Tarjeta Madre Gigabyte XL-ATX TRX40 Designare	5.0	1	0
14	50	SSD Crucial MX500	5.0	1	0
15	60	Kit Memoria RAM Corsair Dominator Platinum DDR4	5.0	1	0
16	66	TCL Smart TV LED 55S425 54.6	5.0	1	0
17	67	TV Monitor LED 24TL520S-PU 24	5.0	1	0
18	84	Logitech Audífonos Gamer G332	5.0	1	0
19	57	SSD Adata Ultimate SU800	4.87	15	0
20	3	Procesador AMD Ryzen 5 2600	4.81	42	0

Figura 3: Productos con mejor reseña

Ninguno de los 20 productos anteriores tiene devolución, por lo que podemos considerarlos como los 20 productos con mejor reseña, por ahora, seria bueno ver a un futuro las reseñas de aquellos productos que tienen 1 o 2 ventas, también hay que considerar resurtir los productos con **id 7, 22, 40** ya que tienen de 0 a 1 productos en *stock*.

3.3. Estrategia de productos a retirar

Para ver que productos vamos a retirar, revisaremos el top de productos por peor reseña:

¡PRODUCTOS CON PEOR RESEÑA!

	Id	Nombre	Score	Ventas	Devoluciones
1	17	Tarjeta de Video Gigabyte AMD Radeon R7 370 OC	1.0	1	1
2	45	Tarjeta Madre ASRock ATX H110 Pro BTC+	1.0	1	1
3	31	Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0)	1.83	6	3
4	46	Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2	2.0	1	1
5	89	Cougar Audífonos Gamer Phontum Essential	3.0	1	0
6	10	MSI GeForce 210	4.0	1	0
7	13	Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix	4.0	1	0
8	94	HyperX Audífonos Gamer Cloud Flight para PC/PS4/PS4 Pro	4.0	1	0
9	29	Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING	4.14	14	1
10	2	Procesador AMD Ryzen 5 3600	4.23	13	1
11	18	Tarjeta de Video Gigabyte NVIDIA GeForce GT 1030	4.4	5	0
12	4	Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8	4.46	13	0
13	33	Tarjeta Madre ASUS ATX PRIME Z390-A	4.5	2	0
14	74	Logitech Bocinas para Computadora con Subwoofer G560	4.5	2	0
15	47	SSD XPG SX8200 Pro	4.55	11	0
16	42	Tarjeta Madre ASRock Micro ATX B450M Steel Legend	4.56	18	0
17	51	SSD Kingston UV500	4.67	3	0
18	44	Tarjeta Madre MSI ATX B450 TOMAHAWK MAX	4.67	6	0
19	48	SSD Kingston A2000 NVMe	4.67	9	0
20	5	Procesador Intel Core i3-9100F	4.7	20	0

Figura 4: Productos con peor reseña

En total hay 9 devoluciones de todas las compras que hay, por lo que, podemos decir que la tienda cuenta con productos de calidad, a excepción de 4, como vemos en la figura anterior son los productos con el **id 17, 45 31, 46**; tienen un *score* bastante bajo y del total en ventas tienen 100 % de devoluciones o el 50 %, también hay que tener en cuenta el producto con **id 89**, tomando en cuenta esto, seria conveniente retirar esos 4 productos inmediatamente antes de que generen mas perdidas que ganancias.

3.4. Ingresos

INGRESO MENSUAL AÑO: 2002

Mes	Ingreso mensual
Enero	0
Febrero	0
Marzo	0
Abril	0
Mayo	259
Junio	0
Julio	0
Agosto	0
Septiembre	0
Octubre	0
Noviembre	0
Diciembre	0

Figura 5: Ingresos del año 2002

INGRESO MENSUAL AÑO: 2020

Mes	Ingreso mensual
Enero	117738
Febrero	107270
Marzo	162931
Abril	191066
Mayo	91677
Junio	36949
Julio	26949
Agosto	3077
Septiembre	0
Octubre	0
Noviembre	0
Diciembre	0

Figura 6: Ingresos del año 2020

Viendo los ingresos mensuales, los únicos ingresos del año 2002 son en el mes de Mayo, después tenemos ingresos hasta el año 2020, pero, el ultimo trimestre no tuvo ventas. Los meses con mas ventas fueron Abril y Marzo, también notemos que el mes de Septiembre tiene una caída bástate abrupta en comparación con los ingresos del mes de Agosto.

Para volver a tener ventas en la tienda lo ideal sería hacer una campaña de marketing, definir el publico objetivo, pagar por tener publicidad, para, así tener otra vez posicionamiento entre las tiendas *online* de tecnología.

4. Conclusión

Para empezar a reactivar las ventas en la tienda, se debe tomar en cuenta que las categorías con mas ventas son las de **procesadores, tarjetas de vídeo y tarjetas madre**, este dato podría ayudar a definir el publico meta. Algo que podría ayudar a tener buenas métricas en un futuro seria hacer análisis de estadística multivariada como: Análisis de componentes principales (PCA), Regresión Lineal, Clustering, etc, incluso realizar alguna serie temporal para tener estimaciones de los ingresos futuros.

Como conclusión personal, este caso practico fue un reto en la parte de programación, fue la primera vez que programe una interfaz gráfica pero fue una buena experiencia y me dejo bastantes aprendizajes, además de que afiance mis conocimientos de programación básica.