PROYECTO FINAL DE FUNDAMENTOS DE PROGRAMACIÓN CON PYTHON

Proyecto 1 – Reporte

Propuesta para: Gerencia de Ventas de LifeStore

Autor: José Angel Conde Francisco

Fecha: 13 de febrero de 2022

Repositorio: https://github.com/angelc29/proyecto01-emtech

Índice

1. Objetivo

Poner en práctica las bases de programación en Python para análisis y clasificación de datos mediante la creación de programas de entrada de usuario y validaciones, uso y definición de variables y listas, operadores lógicos y condicionales para la clasificación de información.

2. Desarrollo

2.1 Autenticación del usuario

Para esta primera etapa, se realizó un inicio de sesión básico, y gran parte de poder comenzar con esto fueron las sesiones de tutoría donde vimos a detalle como realizar esta parte del proyecto.

```
1  usuarioAccedio = False
2  intentos = 0
3
4  # Bienvenida!
5  mensaje_bienvenida = 'Bienvenido al sistema!\nAccede con tus credenciales'
6  print(mensaje_bienvenida)
7
8  # Recibo constantemente sus intentos
9  while not usuarioAccedio:
10     # Primero ingresa Credenciales
11     usuario = input('Usuario: ')
12     contras = input('Contraseña: ')
13     intentos += 1
14     # Reviso si el par coincide
15     if usuario == 'angelC' and contras == 'admindb321':
16         usuarioAccedio = True
17         print('\nHola de nuevo',usuario,'!')
18     else:
19         print(f'Tienes {3 - intentos} intentos restantes')
20         if usuario == 'angel':
21             print('Te equivocaste en la contraseña')
22         else:
23             print(f'El usuario: "{usuario}" no esta registrado')
24
25         if intentos == 3:
26             exit()
```

Segmento de código 1. Login de usuario

Vemos como se comienza con una variable que guarda un booleano y este únicamente cambia en el momento que el usuario ingresa sus credenciales correctamente. En otro caso no se le dará acceso al usuario al programa y terminará repentinamente.
Hay un total de 3 intentos para que el usuario ingrese sus datos correctamente y en caso de que logre ingresar al programa lo siguiente será mostrar un mensaje de bienvenida junto a menú principal.

2.2 Productos más vendidos

Para conocer los productos más vendidos primero se dividió el problema en tareas más simples, es por eso que primero se desarrolló la función ventasSorted() la cual se encargó de obtener la cantidad de ventas de cada producto y concatenar este valor en una copia de la lista original de productos.

La idea de hacer esto es contener todos los atributos de un producto en una sola tupla para facilitar su ordenamiento sin perder orden en sus otras propiedades de interés.

```
  def ventasSorted():
1     # Una copia de los productos para concatenar como columna al numero de
2 ventas, copia profunda
3     ventas = [row[:] for row in lifestore_products]
4     # Recorre la lista de productos
5     for i,product in enumerate(lifestore_products):
6         # Obtiene id de un producto y se crea la variable para sumar sus ventas
7         id_prod=product[0]
8         numVentas = 0
9 
10        # Se recorren las ventas buscando al producto en cuestion y se suma el
11 numero de apariciones
12        for sale in lifestore_sales:
13            if id_prod == sale[1] and sale[4]==0:
14                numVentas+=1
15        # Se concatena el numero de ventas con el producto analizado
16        ventas[i].append(numVentas)
17    # Devuelve la lista con ventas
      return ventas
```

Segmento de código 2. Función ventasSorted()

En este caso se empieza copiando el contenido de lifestore_products para luego iterar en esta lista y obtener la id_prod que es la llave para encontrar sus ventas en lifestore_sales. Se recorre en la lista de ventas y se acumulan las ventas por producto sin considerar a los productos devueltos, seguidamente se concatena este valor en la tupla del producto analizado. Al final solo devuelve la lista con estos valores concatenados en su correspondiente tupla por cada producto.

El siguiente paso es ordenar esta lista de acuerdo a sus ventas, esto lo hace la función sortList() mostrada en el segmento de codigo 3.

```
1 # Ordena y devuelve una lista de listas l de acuerdo a
2 # un parametro en la posicion i
3 def sortList(l,i,h_to_l):
4     return sorted(l,reverse=h_to_l,key=lambda x:x[i])
```

Segmento de código 3. Función sortList()

Utiliza una expresión lambda para ordenar de acuerdo al índice que se le indique como parámetro, también recibe la lista que se va a ordenar y un booleano que indica si el orden será ascendente o descendente, finalmente devuelve la lista dada.

Por último, se muestran los productos con mas ventas utilizando ciclos for, en este caso, para el segmento de código 4 se muestran solo los 5 productos con mayores ventas.

```
for i,v in enumerate(ventas[:5],start=1):
    print(f"    {i}.- Las ventas del producto {v[1]} fueron {v[5]}")
```

Segmento de código 4. Mostrando los 5 más vendidos

2.3 Productos más buscados

Para hallar los productos mas buscados el proceso fue muy parecido al realizado para encontrar las ventas. Primero se generó una copia de la lista de productos que indicará con un atributo adicional en cada tupla el numero de búsquedas, de esto se encargó la función busquedasSorted()

```
def busquedasSorted():
    # Copia la lista de productos para concatenarle el numero de busquedas
    busquedas = [row[:] for row in lifestore_products]
    # Recorre a la lista de productos y obtiene id_prod
    for i,product in enumerate(lifestore_products):
        id_prod = product[0]
        numBusq = 0
        # Recorre la lista de busquedas encontrando coincidencias
        for search in lifestore_searches:
            if id_prod == search[1]:
                numBusq+=1
        # Se concatena el numero de busquedas con la informacion del producto
        busquedas[i].append(numBusq)
    # Devuelve la lista con busquedas
    return busquedas
```

Segmento de código 6. Función busquedasSorted()

Esta función hace un trabajo parecido a la del segmento de código 2, itera sobre la lista de productos obteniendo al id_prod, llave necesaria para iterar sobre lifestore_searches y acumular las coincidencias que se encuentren aquí, después concatena este valor de búsquedas en la tupla correspondiente. El resultado final nos lleva a devolver una lista con una lista de productos que tiene como atributo el numero de veces que se buscó dicho producto.

Realizado esto, el siguiente paso es ordenar la lista, esto lo hacemos con la función sortList() creada anteriormente y mostrada en el segmento de código 3.

Para mostrar los productos más buscados se realiza un ciclo for como del segmento de código 7 donde se muestran los 10 productos más buscados.

```
for i,s in enumerate(busquedas[:10],start=1):
    print(f"    {i}.- Las busquedas del producto {s[1]} fueron {s[5]}")
```

Segmento de código 7. Mostrando los 10 productos más buscados

2.4 Productos menos vendidos por categoría

Para mostrar una cierta cantidad de productos de una cierta categoría fue necesario primero categorizar estos productos. Es por esto que se realizó la función categorizar().

Esta función se ejecuta después de ordenar la lista de productos con sortList() con ventas del punto 3.2 ya que de esta manera nos ahorramos la tarea de ordenamiento después de categorizarlas.

```
def categorizar(lista_productos):
    # Diccionario para agrupar productos por categoria, primero solo nombres
    categorias = {}
    categorias = {prod[3] : [] for prod in lista_productos if prod[3] not in categorias}
    # Se rellena este diccionario con la informacion de la lista dada
    for prod in lista_productos:
        for cat in categorias.keys():
            # Recorre los productos y va almacenandolosen el diccionario creado
            if prod[3] == cat:
                categorias[cat].append(prod)
    return categorias
```

Segmento de código 8. Función categorizar()

Esta función recibe como único parámetro a la lista que se va a categorizar, en este caso se le pasa a las ventas.

Lo que realiza es crear un diccionario que contiene como llaves el nombre de cada categoría y dentro de sus valores a cada uno de los productos correspondientes a esa categoría, recalcando que en cada tupla de producto se encuentran el numero de ventas que tuvo dicho producto.

Un paso antes de mostrar los productos menos vendidos por cada categoría es filtrar de aquellos que no tuvieron ninguna venta ya que solo nos interesa saber de aquellos que tuvieron menos ventas, de esto se encarga la función limpiarCeros() dada a continuación.

```
# Limpia un diccionario de tuplas con ceros en el lugar index
def limpiarCeros(dict,index):
    for key in dict:
        dict[key] = [v for v in dict[key] if v[index]!=0]
    return dict
```

Segmento de código 9. Función limpiarCeros()

Esta función itera sobre cada conjunto generado en el diccionario utilizando la llave, y a su vez actúa en cada conjunto eliminando las tuplas que contengan campos con 0 en el lugar index. Por último, regresa el diccionario dado como parámetro, pero con las modificaciones realizadas.

Hecho esto se filtra obteniendo hasta 5 productos por cada categoría y estos son mostrados al usuario con las líneas del segmento de código 10.

```
# Se limpia el diccionario para dejar solo 5 productos por categoria y con menos ventas
menosVendidos = {cat:list(reversed(categorias[cat]))[:5] for cat in categorias}
# Se muestran hasta los 5 productos menos vendidos por categoria
for cat in menosVendidos:
    print(f"Menos vendidos de la categoria {cat}:")
    for i,v in enumerate(menosVendidos[cat],start=1):
        print(f"    {i}.- {v[1]} ----------- {v[5]} ventas")
```

Segmento de código 10. Función limpiarCeros()

2.5 Productos menos buscados por categoría

Para mostrar los productos menos buscados por categoría se realizan pasos muy parecidos al punto 2.4, comenzando con el hecho de que la lista de búsquedas debe ser ordenada antes de reutilizar la función categorizar(), dándole como parámetro a la lista de búsquedas generada en el punto 2.3.

Seguidamente de esto el diccionario es filtrado de productos sin búsquedas con la función limpiarCeros() y finalmente los resultados son mostrados en unas cuantas líneas de código, esto está resumido en el segmento de código 11 mostrado a continuación.

```
# Se ordenan las busquedas
busquedas=sortList(busquedas,5,True)
# Con la informacion de busquedas se obtiene un diccionario parecido al que se genero anteriormente
categorias = categorizar(busquedas)
# Se limpian los renglones de productos que no tuvieron busquedas
categorias=limpiarCeros(categorias,5)
# Se guarda una nueva lista con al menos 10 productos con menores busquedas por categoria
menosBuscados = {cat:list(reversed(categorias[cat]))[:10] for cat in categorias}
# Se muestran los productos con menores busquedas por categoria
for cat in menosBuscados:
    if len(menosBuscados[cat]) != 0:
        print(f"Menos buscados de la categoria {cat}:")
        for i,b in enumerate(menosBuscados[cat],start=1):
            print(f"    {i}.- {b[1] } ----------- {b[5]} busquedas")
    else:
        print("Sin busquedas para la categoria",cat)
```

Segmento de código 11. Muestra productos con menos búsquedas por categoría

En caso de que no existan búsquedas para alguna categoría, se muestra un mensaje indicando esto.

2.6 Productos con peor y mejor reseña

Para este paso se realizó un trabajo parecido al del punto 2.2 y 2.3 donde generamos una copia de las lista principal de productos y le concatenamos en cada tupla nuestro valor de interés. En este caso el valor nos interesaba fue el de reseña general por producto.

El principal problema para obtener lista de reseñas generales por cada producto es que cada uno de estos tiene una reseña basada en sus ventas y el promedio obtenido de todas estas. Es por esto que el problema principal se dividió en dos problemas más simples: obtener la reseña de un solo producto y obtener el promedio de todos los productos.

Para obtener la reseña promedio de un solo producto, se hizo la función getResProm().

```
# Obtiene una lista de productos con su correspondiente reseña concatenada
def reseniasProds():
    # Crea una copia de la lista principal de productos
    reseniasProds = [row[:] for row in lifestore_products]
    # Recorre la lista productos, obtiene id_prod y corre la funcion getResProm()
    # Almacena esta reseña general en la lista creada correspondiendo al producto
    for idx,prod in enumerate(lifestore_products):
        id_prod = prod[0]
        res = getResProm(id_prod)
        reseniasProds[idx].append(res)
    return reseniasProds
```

Segmento de código 12. Función getResProm()

Esta función recibe como parámetro un id_prod con el cual se ubican las ventas de este producto en la lista lifestore_sales. Se guardan las reseñas de estas ventas encontradas y luego se suman los valores de la lista, seguidamente, si es 0 se obvia la división para no generar un error o en caso contrario se obtiene un promedio con los valores y esta función termina devolviendo la reseña general de ese producto.

Para obtener la reseña promedio de todos los productos se hizo la función reseniasProds().

```
# Obtiene la reseña promedio de un solo producto dada su id_prod
def getResProm(id_prod):
    # Crea una lista que almacena todas las reseñas del producto solicitado
    resenias = [v[2] for v in lifestore_sales if id_prod == v[1]]
    prom = sum(resenias)
    # Se obtiene un promedio de los numeros en la lista resenias
    # y en caso de no tener reseñas/ventas no forzar una division entre 0
    if(prom != 0):
        prom = prom/len(resenias)
    # Finalmente se devuelve la reseña promedio redondeada a 2 decimales
    return round(prom,2)
```

Segmento de código 13. Función reseniasProds()

Esta función reseniasProds() primero crea una copia de los productos para concatenarles su correspondiente reseña general, utiliza la función getResProm() y le pasa como parámetro el id_prod obteniendo así una lista con productos y su correspondiente reseña cada uno.

Con las funciones anteriores ejecutadas solo resta limpiar de productos sin reseña por falta de ventas. Lo siguiente es ordenar la lista de manera descendente con ya que es un paso importante antes de mostrar a los peores y mejores productos de acuerdo a la reseña. Esto se resume en el segmento de código 14.

```
resProductosCpy = [p_res for p_res in resProductos if p_res[5]!=0]
# Se ordena la lista con reseñas de acuerdo a esta de mayor a menor
resProductosCpy = sortList(resProductosCpy,5,True)
# Se muestran los 5 productos con mejor reseña
for i,p_res in enumerate(resProductosCpy[:5],start=1):
    print(f"    {i}.- {p_res[1]} tiene una reseña de {p_res[5]}")
print("Top 5 peores productos de acuerdo a su reseña")
# Se muestran los 5 productos con peor reseña
for i,p_res in enumerate(list(reversed(resProductosCpy))[:5],start=1):
    print(f"    {i}.- {p_res[1]} tiene una reseña de {p_res[5]}")
```

Segmento de código 14. Mostrando 5 mejores y peores productos

2.7 Análisis de las ventas

El principal problema al analizar las ventas fue el de ordenarlas de acuerdo a su mes en que se hizo la venta. Para resolver esto se utilizó la función ventasMeses() mostrada en el segmento de código 15.

```python
from datetime import datetime as dt

# Formato correspondiente a las ventas de lifestore_sales
formato = "%d/%m/%Y"

# La siguiente funcion agrupa las ventas por meses en un diccionario
def ventasMeses():
    # Se genera una copia de lifestore_sales
    aux_ventas = [row[:] for row in lifestore_sales]
    # Se le concatena el precio de cada producto en las ventas
    for prod in lifestore_products:
        for idx,venta in enumerate(aux_ventas):
            if prod[0]==venta[1]:
                aux_ventas[idx].append(prod[2])

    # Crea un diccionario {mesNum: [v1,v2,v3...], mesNum [v1,v2,v3...]...}
    ventas_meses = {dt.strptime(sale[3],formato).strftime("%m") : [] for sale in aux_ventas}

    # Inserta las ventas en su correspondiente mes sin considerar a las devoluciones
    for mes in ventas_meses.keys():
        for sale in aux_ventas:
            if dt.strptime(sale[3],formato).strftime("%m") == mes and sale[4]==0:
                ventas_meses[mes].append(sale)
    return ventas_meses
```

Segmento de código 15. Función ventasMeses()

Esta función se encarga de ordenar las ventas por meses, para esto genera un diccionario para almacenar como llave al mes en numero de 2 dígitos y como valores a las ventas realizadas en dicho mes. Para leer las fechas de cada venta se hace uso de la librería datetime que nos facilita la conversión de datos tipo fecha a cadenas o viceversa.

Después de agrupar las ventar por meses lo que sigue es ordenarlas, esto lo hace la función ordenaMeses() vista en el segmento de código 16. Utiliza un diccionario para ordenar y sustituir las llaves (en dígitos) a cadenas (letras), esto es importante para el análisis mas adelante.

```
dict_meses={ "Enero" : 1,
             "Febrero" : 2,
             "Marzo" : 3,
             "Abril" : 4,
             "Mayo" : 5,
             "Junio" : 6,
             "Julio" : 7,
             "Agosto" : 8,
             "Septiembre" : 9 ,
             "Octubre" : 10,
             "Noviembre" : 11,
             "Diciembre" : 12
             }

# Esta funcion ordena el diccionario de ventas por meses
# Cambia la llave para identificarla por un string y los ordena como calendario
def ordenaMeses(ventas_meses):
    # Crea una copia con las llaves del diccionario en numeros
    aux_meses = list(ventas_meses.keys())
    # Se ordena el diccionario por meses y se convierte a cadena el numero de este
    for mesStr,mDig in dict_meses.items():
        for mes in aux_meses:
            if int(mes) == mDig:
                ventas_meses[mesStr] = ventas_meses[mes]
                del ventas_meses[mes]
                break
    return ventas_meses
```

Segmento de código 16. Función ordenaMeses()

El paso siguiente es mostrar estos resultados y para mostrar los ingresos mensuales se analizan las ventas en cada mes, se suman los totales y se entrega el resultado en un total mensual. Podemos ver este proceso en el segmento de código 17.

```
# Se crea una lista para contener los ingresos mensuales
ingresos_meses=[]
cantVentas_meses = []
print('Ingresos Mensuales\n')
for mes in dict_meses:
    totalMensual = 0
    notVentas = False
    try:
        for venta in ventas_meses[mes]:
            totalMensual+=venta[5]
        if totalMensual == 0:
            notVentas = True
            print('    Hubo devoluciones.',end=' ')
    except:
        notVentas = True
    if notVentas:
        print('    El mes',mes,'no tuvo ventas')
        cantVentas_meses.append(0)
    else:
        print(f'    {mes} generó {totalMensual}.00 MXN de ingresos y hubo
{len(ventas_meses[mes])} ventas')
        cantVentas_meses.append(len(ventas_meses[mes]))
    ingresos_meses.append(totalMensual)
```

Segmento de código 17. Mostrando ingresos mensuales

Se pueden obtener los otros datos de interés en este punto con los datos obtenidos, esto es el total anual, promedio mensual o meses con mas ventas, se muestra el procedimiento en el segmento de código 18.

```
# Se obtiene el promedio de ingresos mensuales
promMensual=sum(ingresos_meses)/len(ingresos_meses) #Considerando 12 meses
print('\nEl promedio general de ingresos de ventas mensuales fue
de',promMensual,'MXN')
print('\nEl total de ingresos generados en 2020 fué de',sum(ingresos_meses),'.00
MXN')
print('\nMeses con mas ventas')
cantVentas_meses = {key : cantVentas_meses[idx] for idx,key in
enumerate(dict_meses)}
for i,it in enumerate(sorted(cantVentas_meses.items(),key=lambda x:
x[1],reverse=True)[:5],start=1):
    print(f'    {i}.- {it[0]} tuvo {it[1]} ventas')
```

Segmento de código 18. Mostrando otros datos de interés

3. Resultados

3.1 Inicio de sesión

```
Bienvenido al sistema!
Accede con tus credenciales
Usuario: angelC
Contraseña: admindb321

Hola de nuevo angelC !


Menu principal
 1. Consultar los 5 productos más vendidos
 2. Consultar los 10 productos más buscados
 3. Ver los productos menos vendidos por categoria
 4. Ver los productos menos buscados por categoria
 5. Consultar los productos con peor y mejor reseña
 6. Ver el informe de ventas
 7. Estrategia Sugerida
 0. Salir
Ingresa una opción:
```

Figura 1. Inicio de sesión y menú principal

3.2 Los 5 productos más vendidos

```
Ingresa una opción: 1

Hubo un total de 274 ventas satisfactorias

Top de los 5 productos más vendidos
    1.- Las ventas del producto SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm fueron 49
    2.- Las ventas del producto Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth fueron 42
    3.- Las ventas del producto Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) fueron 20
    4.- Las ventas del producto Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD fueron 18
    5.- Las ventas del producto SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm fueron 15

Esperando Enter para continuar...
```

Figura 2. Consultar los 5 productos mas vendidos

3.3 Los 10 productos más buscados

```
Ingresa una opción: 2

Hubo un total de 1033 busquedas

Top de los 10 productos más buscados
    1.- Las busquedas del producto SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm fueron 263
    2.- Las busquedas del producto SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm fueron 107
    3.- Las busquedas del producto Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD fueron 60
    4.- Las busquedas del producto Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth fueron 55
    5.- Las busquedas del producto Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire fueron 41
    6.- Las busquedas del producto Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul fueron 35
    7.- Las busquedas del producto TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro fueron 32
    8.- Las busquedas del producto Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) fueron 31
    9.- Las busquedas del producto Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) fueron 30
    10.- Las busquedas del producto SSD XPG SX8200 Pro, 256GB, PCI Express, M.2 fueron 30
```

Figura 3. Consultar los 10 productos mas buscados

3.4 Productos menos vendidos por categoría

```
Productos menos vendidos de 8 categorias (mostrando hasta 5):

Menos vendidos de la categoria discos duros:
    1.- SSD Crucial MX500, 1TB, SATA III, M.2 ----------- 1 ventas
    2.- SSD Western Digital WD Blue 3D NAND, 2TB, M.2 ----------- 2 ventas
    3.- SSD Kingston UV500, 480GB, SATA III, mSATA ----------- 3 ventas
    4.- Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm ----------- 3 ventas
    5.- SSD Kingston A2000 NVMe, 1TB, PCI Express 3.0, M2 ----------- 9 ventas
Menos vendidos de la categoria procesadores:
    1.- Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache ----------- 2 ventas
    2.- Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake) ----------- 3 ventas
    3.- Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake) ----------- 4 ventas
    4.- Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) ----------- 7 ventas
    5.- Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth ----------- 12 ventas
Menos vendidos de la categoria tarjetas madre:
    1.- Tarjeta Madre Gigabyte XL-ATX TRX40 Designare, S-sTRX4, AMD TRX40, 256GB DDR4 para AMD ----------- 1 ventas
    2.- Tarjeta Madre ASUS ATX PRIME Z390-A, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel  ----------- 2 ventas
    3.- Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD ----------- 3 ventas
    4.- Tarjeta Madre MSI ATX B450 TOMAHAWK MAX, S-AM4, AMD B450, 64GB DDR4 para AMD ----------- 6 ventas
    5.- Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD ----------- 13 ventas
Menos vendidos de la categoria tarjetas de video:
    1.- Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti, 6GB 192-bit GDDR6, PCI Express x16 3.0 ----------- 1 ventas
    2.- Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC, 4GB 128-bit GDDR5, PCI Express x16 3.0 ----------- 1 ventas
    3.- Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0 ----------- 1 ventas
    4.- MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0 ----------- 1 ventas
    5.- Tarjeta de Video Sapphire AMD Pulse Radeon RX 5500 XT Gaming, 8GB 128-bit GDDR6, PCI Express 4.0 ----------- 2 ventas
Menos vendidos de la categoria bocinas:
    1.- Logitech Bocinas para Computadora con Subwoofer G560, Bluetooth, Inalámbrico, 2.1, 120W RMS, USB, negro ----------- 2 ventas
Menos vendidos de la categoria audifonos:
    1.- HyperX Audífonos Gamer Cloud Flight para PC/PS4/PS4 Pro, Inalámbrico, USB, 3.5mm, Negro ----------- 1 ventas
    2.- Cougar Audífonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro. ----------- 1 ventas
    3.- Logitech Audífonos Gamer G332, Alámbrico, 2 Metros, 3.5mm, Negro/Rojo ----------- 1 ventas
    4.- Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul ----------- 2 ventas
Menos vendidos de la categoria memorias usb:
    1.- Kit Memoria RAM Corsair Dominator Platinum DDR4, 3200MHz, 16GB (2x 8GB), Non-ECC, CL16, XMP ----------- 1 ventas
Menos vendidos de la categoria pantallas:
    1.- TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro ----------- 1 ventas
    2.- TCL Smart TV LED 55S425 54.6, 4K Ultra HD, Widescreen, Negro ----------- 1 ventas
```

Figura 4. Ver los productos menos buscados por categoría

3.5 Productos menos buscados por categoría

```
Productos menos buscados de 8 categorias (mostrando hasta 10):

Menos buscados de la categoria discos duros:
    1.- SSD Samsung 860 EVO, 1TB, SATA III, M.2 ---------- 1 busquedas
    2.- SSD para Servidor Lenovo Thinksystem S4500, 480GB, SATA III, 3.5'', 7mm ---------- 2 busquedas
    3.- SSD Western Digital WD Blue 3D NAND, 2TB, M.2 ---------- 5 busquedas
    4.- SSD Crucial MX500, 1TB, SATA III, M.2 ---------- 7 busquedas
    5.- Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm ---------- 10 busquedas
    6.- SSD Kingston UV500, 480GB, SATA III, mSATA ---------- 11 busquedas
    7.- SSD Kingston A2000 NVMe, 1TB, PCI Express 3.0, M2 ---------- 27 busquedas
    8.- SSD XPG SX8200 Pro, 256GB, PCI Express, M.2 ---------- 30 busquedas
    9.- SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm ---------- 107 busquedas
    10.- SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm ---------- 263 busquedas
Menos buscados de la categoria tarjetas madre:
    1.- Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel ---------- 1 busquedas
    2.- Tarjeta Madre Gigabyte micro ATX Z390 M GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel  ---------- 1 busquedas
    3.- ASUS T. Madre uATX M4A88T-M, S-AM3, DDR3 para Phenom II/Athlon II/Sempron 100 ---------- 3 busquedas
    4.- Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel ---------- 4 busquedas
    5.- Tarjeta Madre Gigabyte XL-ATX TRX40 Designare, S-sTRX4, AMD TRX40, 256GB DDR4 para AMD ---------- 10 busquedas
    6.- Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD ---------- 10 busquedas
    7.- Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD ---------- 23 busquedas
    8.- Tarjeta Madre MSI ATX B450 TOMAHAWK MAX, S-AM4, AMD B450, 64GB DDR4 para AMD ---------- 25 busquedas
    9.- Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD ---------- 60 busquedas
Menos buscados de la categoria procesadores:
    1.- Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generación - Coffee Lake) ---------- 1 busquedas
    2.- Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake) ---------- 10 busquedas
    3.- Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache ---------- 10 busquedas
    4.- Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake) ---------- 20 busquedas
    5.- Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth ---------- 24 busquedas
    6.- Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) ---------- 30 busquedas
    7.- Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) ---------- 31 busquedas
    8.- Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire ---------- 41 busquedas
    9.- Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth ---------- 55 busquedas
Menos buscados de la categoria audifonos:
    1.- Ginga Audífonos con Micrófono GI18ADJ01BT-RO, Bluetooth, Alámbrico/Inalámbrico, 3.5mm, Rojo ---------- 1 busquedas
    2.- Genius GHP-400S Audífonos, Alámbrico, 1.5 Metros, Rosa ---------- 2 busquedas
    3.- Iogear Audífonos Gamer GHG601, Alámbrico, 1.2 Metros, 3.5mm, Negro ---------- 3 busquedas
    4.- HyperX Audífonos Gamer Cloud Flight para PC/PS4/PS4 Pro, Inalámbrico, USB, 3.5mm, Negro ---------- 6 busquedas
    5.- Cougar Audífonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro. ---------- 7 busquedas
    6.- Logitech Audífonos Gamer G332, Alámbrico, 2 Metros, 3.5mm, Negro/Rojo ---------- 10 busquedas
    7.- Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul ---------- 35 busquedas
Menos buscados de la categoria pantallas:
    1.- Samsung Smart TV LED 43, Full HD, Widescreen, Negro ---------- 1 busquedas
    2.- Samsung Smart TV LED UN55TU7000FXZX 55, 4K Ultra HD, Widescreen, Negro/Gris ---------- 4 busquedas
    3.- Seiki TV LED SC-39HS950N 38.5, HD, Widescreen, Negro ---------- 4 busquedas
    4.- TCL Smart TV LED 55S425 54.6, 4K Ultra HD, Widescreen, Negro ---------- 15 busquedas
    5.- TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro ---------- 32 busquedas
Menos buscados de la categoria tarjetas de video:
    1.- Tarjeta de Video VisionTek AMD Radeon HD5450, 2GB GDDR3, PCI Express x16 ---------- 1 busquedas
    2.- MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0 ---------- 1 busquedas
    3.- Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0 ---------- 2 busquedas
    4.- Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0 ---------- 3 busquedas
    5.- Tarjeta de Video EVGA NVIDIA GeForce GTX 1660 Ti SC Ultra Gaming, 6GB 192-bit GDDR6, PCI 3.0 ---------- 4 busquedas
    6.- Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti, 6GB 192-bit GDDR6, PCI Express x16 3.0 ---------- 5 busquedas
    7.- Tarjeta de Video VisionTek AMD Radeon HD 5450, 1GB DDR3, PCI Express x16 2.1 ---------- 5 busquedas
    8.- Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC, 4GB 128-bit GDDR5, PCI Express x16 3.0 ---------- 5 busquedas
    9.- Tarjeta de Video ASUS AMD Radeon RX 570, 4GB 256-bit GDDR5, PCI Express 3.0 ---------- 5 busquedas
    10.- Tarjeta de Video Sapphire AMD Pulse Radeon RX 5500 XT Gaming, 8GB 128-bit GDDR6, PCI Express 4.0 ---------- 10 busquedas
 Menos buscados de la categoria bocinas:
     1.- Ghia Bocina Portátil BX800, Bluetooth, Inalámbrico, 2.1 Canales, 31W, USB, Negro ---------- 1 busquedas
     2.- Acteck Bocina con Subwoofer AXF-290, Bluetooth, Inalámbrico, 2.1, 18W RMS, 180W PMPO, USB, Negro ---------- 2 busquedas
     3.- Logitech Bocinas para Computadora con Subwoofer G560, Bluetooth, Inalámbrico, 2.1, 120W RMS, USB, negro ---------- 6 busquedas
 Sin busquedas para la categoria memorias usb
```

Figura 5. Ver los productos menos buscados por categoría

3.6 Mejores y peores productos según su reseña

```
La cantidad de productos reseñados fue de 42

Top 5 mejores productos de acuerdo a su reseña
    1.- Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache tiene una reseña de 5.0
    2.- Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake) tiene una reseña de 5.0
    3.- Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) tiene una reseña de 5.0
    4.- Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake) tiene una reseña de 5.0
    5.- Tarjeta de Video ASUS AMD Radeon RX 570, 4GB 256-bit GDDR5, PCI Express 3.0 tiene una reseña de 5.0
Top 5 peores productos de acuerdo a su reseña
    1.- Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel tiene una reseña de 1.0
    2.- Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0 tiene una reseña de 1.0
    3.- Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD tiene una reseña de 1.83
    4.- Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel tiene una reseña de 2.0
    5.- Cougar Audífonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro. tiene una reseña de 3.0
```

Figura 6. Consultar los productos con peor y mejor reseña

3.7 Informe de ventas

Para el análisis de las ventas se tomó en cuenta generar un informe con los siguientes datos:

- Ingresos netos mensuales
- Promedio de ingresos generados al mes
- Total, de ingresos anuales generados
- Meses con más ventas

```
Informe general de las ventas de LifeStore

Ingresos Mensuales

    Enero generó 117738.00 MXN de ingresos y hubo 52 ventas
    Febrero generó 107270.00 MXN de ingresos y hubo 40 ventas
    Marzo generó 162931.00 MXN de ingresos y hubo 49 ventas
    Abril generó 191066.00 MXN de ingresos y hubo 74 ventas
    Mayo generó 91936.00 MXN de ingresos y hubo 34 ventas
    Junio generó 36949.00 MXN de ingresos y hubo 11 ventas
    Julio generó 26949.00 MXN de ingresos y hubo 11 ventas
    Agosto generó 3077.00 MXN de ingresos y hubo 3 ventas
    Hubo devoluciones.     El mes Septiembre no tuvo ventas
    El mes Octubre no tuvo ventas
    Hubo devoluciones.     El mes Noviembre no tuvo ventas
    El mes Diciembre no tuvo ventas

El promedio general de ingresos de ventas mensuales fue de 61493.0 MXN

El total de ingresos generados en 2020 fué de 737916 .00 MXN

Meses con mas ventas
    1.- Abril tuvo 74 ventas
    2.- Enero tuvo 52 ventas
    3.- Marzo tuvo 49 ventas
    4.- Febrero tuvo 40 ventas
    5.- Mayo tuvo 34 ventas
```

Figura 7. Informe de ventas

3.8 Estrategia Sugerida

Para generar una estrategia de retiro de productos del mercado se tomó en cuenta a los productos que no generaban ganancias y tampoco eran relevantes para los consumidores, esto quiere decir que filtrar a los productos sin búsquedas y sin ventas es un buen indicador para hallar productos rezagados. A su vez, de estos productos se tomó en cuenta un ordenamiento basado en espacio que utilizan en inventario.

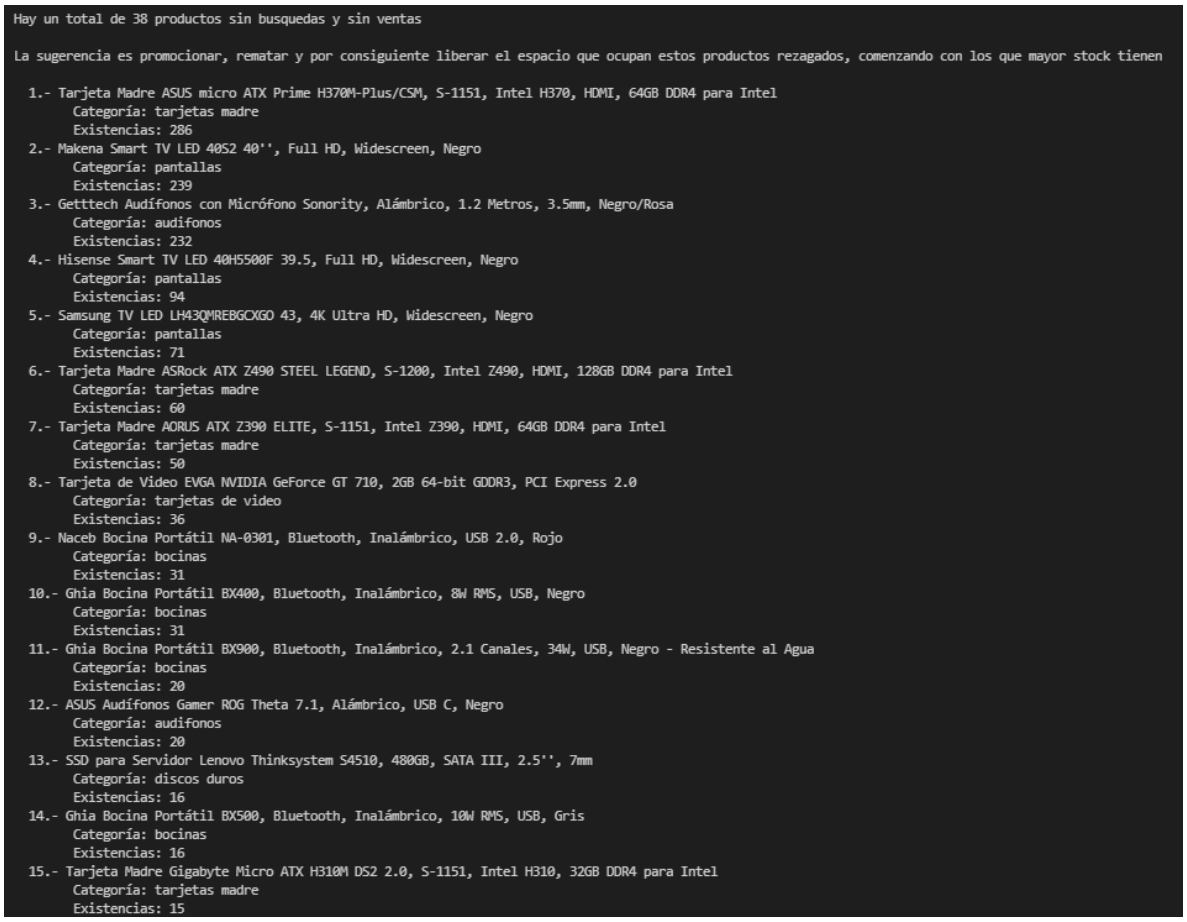

Figura 8. Informe de ventas

4. Conclusión

Este proyecto me pareció muy entretenido y una forma muy didáctica de adentrarse a Python ya que entre mas se avanza, mas crece la necesidad de explorar funciones nativas de Python que nos faciliten las tareas que queremos resolver. En mi opinión fue muy satisfactorio poder resolver todos los problemas de análisis del proyecto usando Python puro y no más que una librería para facilitar el manejo de fechas, esto me ayudó mucho a practicar este lenguaje y creo que gran parte de la importancia para adentrarse en otras librerías lo primero es tener unos buenos cimientos de programación con el lenguaje. También opino que fue interesante resolver los problemas dados ya que son indispensables en el análisis de datos y es un buen comienzo para comprender problemas más complejos que pueden requerir un manejo más amplio de otras funciones o librerías.

5. Anexos

Codigo de main.py

```
from lifestore_file import lifestore_searches, lifestore_sales, lifestore_products
from datetime import datetime as dt
#import funciones #De esta manera hay que anteponer 'funciones' antes de usar cada una
from funciones import *
"""
La info de lifestore_file:

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]
"""

"""
Login
credenciales:

usuario:
    angelC
contrasenia:
    admindb321
"""

usuarioAccedio = False
intentos = 0

# Bienvenida!
mensaje_bienvenida = 'Bienvenido al sistema!\nAccede con tus credenciales'
print(mensaje_bienvenida)

# Recibo constantemente sus intentos
while not usuarioAccedio:
    # Primero ingresa Credenciales
    usuario = input('Usuario: ')
    contras = input('Contraseña: ')
    intentos += 1
    # Reviso si el par coincide
    if usuario == 'angelC' and contras == 'admindb321':
        usuarioAccedio = True
        print('\nHola de nuevo',usuario,'!')
    else:
        print(f'Tienes {3 - intentos} intentos restantes')
        if usuario == 'angel':
            print('Te equivocaste en la contraseña')
        else:
            print(f'El usuario: "{usuario}" no esta registrado')

        if intentos == 3:
```

```
            exit()

# Ejecuta la funcion principal para contar las ventas
ventas = ventasSorted()
sinVentas = [venta for venta in ventas if venta[5]==0]
# Ejecuta la funcion principal para contar las busquedas
busquedas=busquedasSorted()
sinBusquedas = [bus for bus in busquedas if bus[5] == 0]
#Se obtiene una lista de productos con su correspondiente reseña general
resProductos = reseniasProds()

def printMenu():
    print("\n")
    print('Menu principal')
    print(' 1. Consultar los 5 productos más vendidos')
    print(' 2. Consultar los 10 productos más buscados')
    print(' 3. Ver los productos menos vendidos por categoria')
    print(' 4. Ver los productos menos buscados por categoria')
    print(' 5. Consultar los productos con peor y mejor reseña')
    print(' 6. Ver el informe de ventas')
    print(' 7. Estrategia Sugerida')
    print(' 0. Salir')

# Menu principal
printMenu()
menuMain = int(input("Ingresa una opción: "))
while menuMain !=0:
    if menuMain == 1:
        # Resuelve el punto 1.1 del PUNTO 1
        # Realiza un calculo del numero de ventas total sin considerar devoluciones
        totalSales = 0
        for ven in ventas:
            totalSales+=ven[5]
        print("\nHubo un total de",totalSales,"ventas satisfactorias")
        print("\nTop de los 5 productos más vendidos")
        # Se ordena de acuerdo al numero de ventas de cada producto (mayor a menor)
        ventas = sortList(ventas,5,True)
        for i,v in enumerate(ventas[:5],start=1):
            print(f"    {i}.- Las ventas del producto {v[1]} fueron {v[5]}")
    elif menuMain == 2:
        # Resuelve el punto 1.2 del PUNTO 1
        print("\nHubo un total de",len(lifestore_searches),"busquedas")
        print("\nTop de los 10 productos más buscados")
        # Se ordena la lista de mayor a menor segun su numero de busquedas
        busquedas=sortList(busquedas,5,True)
        for i,s in enumerate(busquedas[:10],start=1):
            print(f"    {i}.- Las busquedas del producto {s[1]} fueron {s[5]}")
    elif menuMain == 3:
        # # Se resuelve el punto 2.1 del PUNTO 1
        # Se ordenan las ventas
        ventas = sortList(ventas,5,True)

```

```
        # Con la informacion de ventas se obtiene un diccionario que corresponde al tipo
{categoria1: [venta1,venta2...], categoria2: [venta1,venta2...]...}
        categorias = categorizar(ventas)
        print("\nProductos menos vendidos de",len(categorias),"categorias (mostrando hasta 5):\n")
        # Usa la funcion para limpiar de productos sin ventas
        categorias=limpiarCeros(categorias,5)
        # Se limpia el diccionario para dejar solo 5 productos por categoria y con menos ventas
        menosVendidos = {cat:list(reversed(categorias[cat]))[:5] for cat in categorias}
        # Se muestran hasta los 5 productos menos vendidos por categoria
        for cat in menosVendidos:
            print(f"Menos vendidos de la categoria {cat}:")
            for i,v in enumerate(menosVendidos[cat],start=1):
                print(f"    {i}.- {v[1]} ----------- {v[5]} ventas")
    elif menuMain == 4:
        # # Se resuelve el punto 2.2 del PUNTO 1
        # Se ordenan las busquedas
        busquedas=sortList(busquedas,5,True)
        # Con la informacion de busquedas se obtiene un diccionario parecido al que se genero
anteriormente
        categorias = categorizar(busquedas)
        print("\nProductos menos buscados de",len(categorias),"categorias (mostrando hasta
10):\n")
        # Se limpian los renglones de productos que no tuvieron busquedas
        categorias=limpiarCeros(categorias,5)
        # Se guarda una nueva lista con al menos 10 productos con menores busquedas por categoria
        menosBuscados = {cat:list(reversed(categorias[cat]))[:10] for cat in categorias}
        # Se muestran los productos con menores busquedas por categoria
        for cat in menosBuscados:
            if len(menosBuscados[cat]) != 0:
                print(f"Menos buscados de la categoria {cat}:")
                for i,b in enumerate(menosBuscados[cat],start=1):
                    print(f"    {i}.- {b[1] } ----------- {b[5]} busquedas")
            else:
                print("Sin busquedas para la categoria",cat)
    elif menuMain == 5:
        # # Resuelve el PUNTO 2
        # Elimina los producto que no obtuvieron reseñas generales por no tener ninguna venta y
devuelve esta lista limpia
        resProductosCpy = [p_res for p_res in resProductos if p_res[5]!=0]
        print("\nLa cantidad de productos reseñados fue de",len(resProductosCpy))
        # Se ordena la lista con reseñas de acuerdo a esta de mayor a menor
        resProductosCpy = sortList(resProductosCpy,5,True)
        print("\nTop 5 mejores productos de acuerdo a su reseña")
        # Se muestran los 5 productos con mejor reseña
        for i,p_res in enumerate(resProductosCpy[:5],start=1):
            print(f"    {i}.- {p_res[1]} tiene una reseña de {p_res[5]}")
        print("Top 5 peores productos de acuerdo a su reseña")
        # Se muestran los 5 productos con peor reseña
        for i,p_res in enumerate(list(reversed(resProductosCpy))[:5],start=1):
            print(f"    {i}.- {p_res[1]} tiene una reseña de {p_res[5]}")
    elif menuMain == 6:
        # Resuelve el PUNTO 3
```

```
        print("\nInforme general de las ventas de LifeStore\n")
        # La funcion ventasMeses obtiene un diccionario donde estan agrupadas las ventas por mes y
se ordena con la funcion ordenaMeses
        ventas_meses = ordenaMeses(ventasMeses())
        # Se crea una lista para contener los ingresos mensuales
        ingresos_meses=[]
        cantVentas_meses = []
        print('Ingresos Mensuales\n')
        for mes in dict_meses:
            totalMensual = 0
            notVentas = False
            try:
                for venta in ventas_meses[mes]:
                    totalMensual+=venta[5]
                if totalMensual == 0:
                    notVentas = True
                    print('    Hubo devoluciones.',end=' ')
            except:
                notVentas = True
            if notVentas:
                print('    El mes',mes,'no tuvo ventas')
                cantVentas_meses.append(0)
            else:
                print(f'    {mes} generó {totalMensual}.00 MXN de ingresos y hubo
{len(ventas_meses[mes])} ventas')
                cantVentas_meses.append(len(ventas_meses[mes]))
            ingresos_meses.append(totalMensual)

        # Se obtiene el promedio de ingresos mensuales
        promMensual=sum(ingresos_meses)/len(ingresos_meses) #Considerando 12 meses
        print('\nEl promedio general de ingresos de ventas mensuales fue de',promMensual,'MXN')
        print('\nEl total de ingresos generados en 2020 fué de',sum(ingresos_meses),'.00 MXN')
        print('\nMeses con mas ventas')
        cantVentas_meses = {key : cantVentas_meses[idx] for idx,key in enumerate(dict_meses)}
        for i,it in enumerate(sorted(cantVentas_meses.items(),key=lambda x:
x[1],reverse=True)[:5],start=1):
            print(f'    {i}.- {it[0]} tuvo {it[1]} ventas')
    elif menuMain == 7:
        # Se realiza una interseccion entre productos sin ventas y sin busquedas para saber que
productos estan mas olvidados
        rezagados = [prod for prod in sinVentas if prod in sinBusquedas]
        # Se ordenan de acuerdo a su stock
        rezagados = sortList(rezagados,4,True)
        print('\nHay un total de',len(rezagados),'productos sin busquedas y sin ventas')
        print('\nLa sugerencia es promocionar, rematar y por consiguiente liberar el espacio que
ocupan estos productos rezagados, comenzando con los que mayor stock tienen\n')
        for i,r in enumerate(rezagados,start=1):
            print(f'  {i}.- {r[1]}\n        Categoría: {r[3]}\n        Existencias: {r[4]}')
    else:
        print('Ingrese una opción valida')
    input('\nEsperando Enter para continuar...')
    printMenu()
```

```
menuMain = int(input("Ingresa una opción: "))
```

Código de funciones.py

```
from lifestore_file import lifestore_searches, lifestore_sales, lifestore_products
from datetime import datetime as dt

# Ordena y devuelve una lista de listas l de acuerdo a
# un parametro en la posicion i
def sortList(l,i,h_to_l):
    return sorted(l,reverse=h_to_l,key=lambda x:x[i])

# Funcion para mostrar un diccionario
def showDict(dict):
    total=0
    for key in dict:
        total+=len(dict[key])
        print(key, f" {len(dict[key])} values")
        for value in dict[key]:
            print("      ",value)
    print("Total values:",total)

# Limpia un diccionario de tuplas con ceros en el lugar index
def limpiarCeros(dict,index):
    for key in dict:
        dict[key] = [v for v in dict[key] if v[index]!=0]
    return dict

# Obtiene el numero de busquedas por producto y concatena este valor en una copia de la lista
productos
def busquedasSorted():
    # Una copia de la lista de productos para concatenarle el numero de busquedas, copia profunda
    busquedas = [row[:] for row in lifestore_products]
    # Recorre a la lista de productos y obtiene id_prod
    for i,product in enumerate(lifestore_products):
        id_prod = product[0]
        numBusq = 0
        # Recorre la lista de busquedas encontrando coincidencias con el producto en cuestion
        for search in lifestore_searches:
```

```
            if id_prod == search[1]:
                numBusq+=1
        # Se concatena el numero de busquedas con la informacion del producto
        busquedas[i].append(numBusq)
    # Devuelve la lista con busquedas
    return busquedas

# Obtiene el numero de ventas por producto y concatena este valor en una copia de la lista de
productos
def ventasSorted():
    # Una copia de los productos para concatenar como columna al numero de ventas, copia profunda
    ventas = [row[:] for row in lifestore_products]
    # Recorre la lista de productos
    for i,product in enumerate(lifestore_products):
        # Obtiene id de un producto y se crea la variable para sumar sus ventas
        id_prod=product[0]
        numVentas = 0

        # Se recorren las ventas buscando al producto en cuestion y se suma el numero de
apariciones
        for sale in lifestore_sales:
            if id_prod == sale[1] and sale[4]==0:
                numVentas+=1
        # Se concatena el numero de ventas con el producto analizado
        ventas[i].append(numVentas)
    # Devuelve la lista con ventas
    return ventas

# Categoriza la lista de productos (utiliza el indice 3 para generar grupos) y regresa un
diccionario
def categorizar(lista_productos):
    # Se crea un diccionario para agrupar productos por categoria, en un inicio solo guardamos los
nombres
    categorias = {}
    categorias = {prod[3] : [] for prod in lista_productos if prod[3] not in categorias}
    # Se rellena este diccionario con la informacion de la lista dada
    for prod in lista_productos:
        for cat in categorias.keys():
            # Se recorre sobre los productos y se van almacenando cada una de estos en el
diccionario de categorias
            if prod[3] == cat:
                categorias[cat].append(prod)
    return categorias

# Obtiene una lista de productos con su correspondiente reseña concatenada como una columna
def reseniasProds():
    # Crea una copia de la lista principal de productos
    reseniasProds = [row[:] for row in lifestore_products]
    # Recorre la lista productos para obtener la id_prod y correr la funcion getResProm() por cada
producto,
    # despues almacena esta reseña general en la lista recien creada correspondiendo al producto
    for idx,prod in enumerate(lifestore_products):
```

```
            id_prod = prod[0]
            res = getResProm(id_prod)
            reseniasProds[idx].append(res)
        return reseniasProds

    # Obtiene la reseña promedio de un solo producto dada su id_prod
    def getResProm(id_prod):
        # Crea una lista que almacena todas las reseñas del producto solicitado
        resenias = [v[2] for v in lifestore_sales if id_prod == v[1]]
        prom = sum(resenias)
        # Se obtiene un promedio de los numeros en la lista resenias
        # y en caso de no tener reseñas/ventas no forzar una division entre 0
        if(prom != 0):
            prom = prom/len(resenias)
        # Finalmente se devuelve la reseña promedio redondeada a 2 decimales
        return round(prom,2)

    # Formato correspondiente a las ventas de lifestore_sales
    formato = "%d/%m/%Y"
    # Se crea un diccionario del tipo mes : mes_numero que ayudará a ordenar el diccionario de ventas
    por meses
    dict_meses={ "Enero" : 1,
                 "Febrero" : 2,
                 "Marzo" : 3,
                 "Abril" : 4,
                 "Mayo" : 5,
                 "Junio" : 6,
                 "Julio" : 7,
                 "Agosto" : 8,
                 "Septiembre" : 9 ,
                 "Octubre" : 10,
                 "Noviembre" : 11,
                 "Diciembre" : 12
                 }

    # La siguiente funcion agrupa las ventas por meses en un diccionario
    def ventasMeses():
        # Se genera una copia de lifestore_sales
        aux_ventas = [row[:] for row in lifestore_sales]
        # Se le concatena el precio de cada producto en las ventas
        for prod in lifestore_products:
            for idx,venta in enumerate(aux_ventas):
                if prod[0]==venta[1]:
                    aux_ventas[idx].append(prod[2])

        # Crea un diccionario de ventas por meses de la forma {mesNum: [v1,v2,v3...], mesNum
    [v1,v2,v3...]...}
        ventas_meses = {dt.strptime(sale[3],formato).strftime("%m") : [] for sale in aux_ventas}

        # Se inserta cada una de las ventas en su correspondiente mes sin considerar a las
    devoluciones
        for mes in ventas_meses.keys():
```

```
140             for sale in aux_ventas:
141                 if dt.strptime(sale[3],formato).strftime("%m") == mes and sale[4]==0:
142                     ventas_meses[mes].append(sale)
143         return ventas_meses
144 
145     # Esta funcion ordena el diccionario de ventas por meses
146     # Cambia la llave para identificarla por un string y los ordena como calendario
147     def ordenaMeses(ventas_meses):
        # Crea una copia con las llaves del diccionario en numeros, se usará para ordenar estos y
    cambiarlos a cadenas
        aux_meses = list(ventas_meses.keys())
        # A continuacion se ordena el diccionario por meses y a su vez se convierte a cadena el numero
    de mes (llaves)
        for mesStr,mDig in dict_meses.items():
            for mes in aux_meses:
                if int(mes) == mDig:
                    ventas_meses[mesStr] = ventas_meses[mes]
                    del ventas_meses[mes]
                    break
        return ventas_meses
```