# LifeStore

Por: Griselda Gutiérrez Silvestre

Análisis de Datos | Propuesta a LifeStore

EmTech

# Contenido

## | 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.

## | Descripción del caso

LifeStore es una tienda virtual que maneja una amplia gama de artículos, recientemente, la Gerencia de ventas, se percató que la empresa tiene una importante acumulación de inventario. Asimismo, se ha identificado una reducción en las búsquedas de un grupo importante de productos, lo que ha redundado en una disminución sustancial de sus ventas del último trimestre.

## | Consigna

Derivado de la situación, la Gerencia de Ventas te solicita que realices un análisis de la rotación de productos identificando los siguientes elementos:

1) Productos más vendidos y productos rezagados a partir del análisis de las categorías con menores ventas y categorías con menores búsquedas.
2) Productos por reseña en el servicio a partir del análisis de categorías con mayores ventas y categorías con mayores búsquedas.
3) Sugerir una estrategia de productos a retirar del mercado, así como sugerencia de cómo reducir la acumulación de inventario considerando los datos de ingresos y ventas mensuales.

# | Desarrollo

## → Productos más vendidos

```
#-----------------------------------------------------------------------------------------------
""" Identificar los 5 productos con mayor venta """

#Paso 1: obtener los identificadores de los productos vendidos de la lista lifestore_sales
vendidos=[]
for i in lifestore_sales:
    vendidos.append(i[1])

#Paso 2: obtener una lista de la forma [id_product,name,category,numero_ventas]
lista_productos_vendidos=[]
for producto in lifestore_products:
    #Mientras el id_product este en la lista de vendidos
    if producto[0] in vendidos:
        lista_auxiliar=[]
        #Obtengo el id, nombre y categoria del producto
        lista_auxiliar.append(producto[0])
        lista_auxiliar.append(producto[1])
        lista_auxiliar.append(producto[-2])
        #Obtengo el numero de ventas
        lista_auxiliar.append(vendidos.count(producto[0]))
        lista_productos_vendidos.append(lista_auxiliar)

#Paso 3: Ordenar la lista con base al numero_ventas de forma descendente
lista_productos_vendidos=sorted(lista_productos_vendidos, key=lambda x:x[-1], reverse=True)
```

*Imagen 1 Código para obtener los productos con mayor venta*

- Se creó **vendidos** que contiene los id_product de los productos de la lista **lifestore_sales.**
- Se creó **lista_productos_vendidos** que guarda una lista de la forma:
  [ [id_product,name,category,num_ventas], [ ], [ ], … N ]
- Se iteró sobre **lifestore_products** para obtener con **lista_auxiliar** los campos del id, nombre, categoría y número de ventas.
- Para obtener el número de ventas se usó la función **count** sobre la lista **vendidos**, misma que se le pasa como parámetro el **id_product** actual de la iteración.
- Finalmente se ordenó la lista con **sorted** de forma **descendente** (mayor a menor) y solo se imprimen los primeros cinco elementos de la lista.

## → Productos con mayor búsqueda

```
#-----------------------------------------------------------------------------------------
""" Identificar los 10 productos con mayor busqueda """

#Paso 1: obtener los identificadores de los productos buscados de la lista lifestore_searches
busquedas=[]
for i in lifestore_searches:
    busquedas.append(i[1])

#Paso 2: Obtener una lista de la forma [id_product,name,category,numero_busquedas]
lista_productos_buscados=[]
for producto in lifestore_products:
    #Mientras el id_product este en la lista de busquedas
    if producto[0] in busquedas:
        lista_aux=[]
        #Obtengo el id, nombre y categoria del producto
        lista_aux.append(producto[0])
        lista_aux.append(producto[1])
        lista_aux.append(producto[-2])
        #Obtengo el numero de ventas
        lista_aux.append(busquedas.count(producto[0]))
        lista_productos_buscados.append(lista_aux)

#Paso 3: Ordenar la lista con base al numero_busquedas de forma descendente
lista_productos_buscados=sorted(lista_productos_buscados, key=lambda x:x[-1],reverse=True)
```

*Imagen 2 Código para obtener los productos con mayor búsqueda*

- Se creó **busquedas** que contiene los id_product de los productos de la lista **lifestore_searches**.
- Se creó **lista_productos_buscados** que guarda una lista de la forma:
  [ [id_product,name,category,num_busquedas], [ ], [ ], … N ]
- Se iteró sobre **lifestore_products** para obtener con **lista_aux** los campos del id, nombre, categoría y número de búsquedas.
- Para obtener el número de búsquedas se usa la función **count** sobre la lista **busquedas**, misma que se le pasa como parámetro el **id_product** actual de la iteración.
- Finalmente se ordena la lista con **sorted** de forma **descendente** (mayor a menor) y solo se imprimen los primeros diez elementos de la lista.

## → Productos con menor venta por categoría

```python
#-------------------------------------------------------------------------------------------------------#
"""Identificar los 5 productos con menor venta por categoria"""

#Paso 1: hacer una copia de lista_productos_vendidos
cp_lista_productos_vendidos=lista_productos_vendidos[:]
cp_lista_productos_vendidos=sorted(cp_lista_productos_vendidos, key=lambda x:x[-1], reverse=False)

#Paso 2: crear un diccionario donde las keys sean las categorias de los productos y los values sean una lista de listas de productos [[],[],...N]
productos_vendidos_categoria={}
for producto in lifestore_products:
    #Mientras la categoria no este como key  en el diccionario entonces se agrega
    if producto[3] not in productos_vendidos_categoria.keys():
        productos_vendidos_categoria[producto[3]]=[]

#Paso 3: hacer cruce entre el diccionario y la lista cp_lista_productos_vendidos con el elemento de la categoria
for i in cp_lista_productos_vendidos:
    for key in productos_vendidos_categoria.keys():
        #Mientras la categoria del producto sea igual a la key del diccionario
        if i[2]==key:
            productos_vendidos_categoria[key].append(i)
```

*Imagen 3 Código para obtener los productos con menor venta por categoría*

- Se creó **cp_lista_productos_vendidos** que es una copia de **lista_productos_vendidos** y se ordena de forma **ascendente**.
- Se creo **productos_vendidos_categoria** que guarda un diccionario de la forma:

  {

  'categoria': [ [producto1], [producto2], … n ],

  … N

  }
- Se iteró sobre **lifestore_products** y se verifico con un if que la categoría no estuviera dentro de las keys del diccionario para poder agregarla.
- Se hizo un cruce entre **cp_lista_productos_vendidos** y **productos_vendidos_categoria** con el campo **categoría**, si este es igual en ambos entonces se agregó el producto a la lista actual de la key.
- Finalmente se imprime el diccionario recorriendo las keys, en los values solo se imprimen los primeros cinco elementos de cada lista.

## → Productos con menor búsqueda por categoría

```python
#-------------------------------------------------------------------------------------------------------#
""" Identificar los 10 productos con menor busqueda por categoria """

#Paso 1: copiar la lista lista_productos_buscados
cp_lista_productos_buscados=lista_productos_buscados[:]
cp_lista_productos_buscados=sorted(cp_lista_productos_buscados, key=lambda x:x[-1],reverse=False)

#Paso 2: crear un diccionario donde las keys sean las categorias de los productos y los values sean una lista de listas de productos [[],[],...N]
productos_buscados_categoria={}
for producto in lifestore_products:
    #Mientras la categoria no este como key  en el diccionario entonces se agrega
    if producto[3] not in productos_buscados_categoria.keys():
        productos_buscados_categoria[producto[3]]=[]

#Paso 3: hacer cruce entre el diccionario y la lista cp_lista_productos_buscados con el elemento de la categoria
for i in cp_lista_productos_buscados:
    for key in productos_buscados_categoria.keys():
        #Mientras la categoria del producto sea igual a la key del diccionario
        if i[2]==key:
            productos_buscados_categoria[key].append(i)
```

*Imagen 4 Código para obtener los productos con menor búsqueda por categoría*

- Se creó **cp_lista_productos_buscados** que es una copia de **lista_productos_buscados** y se ordena de forma **ascendente.**
- Se creo **productos_buscados_categoria** que guarda un diccionario de la forma:
  {
    'categoria': [ [producto1], [producto2], … n ],
      … N
  }
- Se iteró sobre **lifestore_products** y se verifico con un if que la categoría no estuviera dentro de las keys del diccionario para poder agregarla.
- Se hizo un cruce entre **cp_lista_productos_buscados** y **productos_buscados_categoria** con el campo **categoría**, si este es igual en ambos entonces se agregó el producto a la lista actual de la key.
- Finalmente se imprime el diccionario recorriendo las keys, en los values solo se imprimen los primeros diez elementos de cada lista.

## → Productos con mejor y peor reseña

```
155 #-------------------------------------------------------------------------------------------------------#
156 """ Identificar 5 productos con mejor reseña """
157
158 #Paso 1: obtener id_product y score de cada venta
159 productos_scores=[[venta[1], venta[2]] for venta in lifestore_sales]
160
161 #Paso 2: crear un diccionario de tipo {'id_product':[score1,score2,...n],... N}
162 productos_scores2={}
163 for producto in productos_scores:
164     #Mientras id_product no se encuentre como key del diccionario, entonces se agrega
165     if producto[0] not in productos_scores2.keys():
166         #Se crea una lista por cada key
167         productos_scores2[producto[0]]=[]
168     #Se agrega la lista de listas de los scores
169     productos_scores2[producto[0]].append(producto[1])
```

*Imagen 5 Código para obtener los id_product que serán las keys de productos_scores2*

- Se creó **productos_scores** que es una lista de listas con los valores ['id_product','score'] de **lifestore_sales**.
- Se creó **productos_scores2** que es un diccionario de la forma:
  { 'id_product': [score1, score2, … score n], … N}
- Se hizo un cruce entre **productos_scores** y **productos_scores2** donde los **id_product** del primero son las llaves del segundo. Para evitar duplicidad en las llaves se verificó que no estuviera para poder agregarla.
- Después se agregó cada **score** al **id_product** que le perteneciera en la iteración actual.

```
171  #Paso 3: encontrar el promedio_scores por producto y guardarlo en el nuevo diccionario
172  productos_score_promedio={}
173  for key in productos_scores2.keys():
174      lista_scores=productos_scores2[key]
175      promedio_scores=sum(lista_scores)/len(lista_scores)
176      # Arreglo promedio a 2 decimales
177      decimales=2
178      multiplicador=10**decimales
179      #Se obtiene el promedio redondeado a 2 decimales
180      promedio=math.ceil(promedio_scores*multiplicador)/multiplicador
181
182      #Mientras id_product no este como key del nuevo diccionario, entonces se agrega
183      if key not in productos_score_promedio.keys():
184          productos_score_promedio[key]=0
185          productos_score_promedio[key]=promedio
186
187  #Paso 4: ordenar el diccionario ascendente por el valor del promedio_scores
188  productos_score_promedio=sorted(productos_score_promedio.items(),key=lambda x:x[1],reverse=True)
189
190  #Paso 5: convertir la lista de tuplas a una lista de listas
191  lista_productos_promScore=[]
192  for i in productos_score_promedio:
193      lista_productos_promScore.append(list(i))
```

*Imagen 6 Código para obtener los scores promedio de cada producto vendido*

- Se creó un segundo diccionario **productos_score_promedio** de la forma: {'id_product': score_promedio, … N}
- Se itera sobre **productos_scores2** para obtener el **id_product** que es la llave del segundo diccionario.
- Se obtiene **lista_scores** que representa todos los scores asociados a un producto.
- Se obtiene con la función **sum** la suma de la lista **lista_scores**, también con **len** se obtiene la longitud de dicha lista. Para obtener el promedio de score se realiza la operación de suma/longitud.
- Se hace un redondeo con **ceil** para obtener solo dos decimales.
- Se ordenó el diccionario con la función **sorted** y como esta regresa una lista de tuplas, después se hace un casteo de tipo de dato de tupla a lista con la función **list** y se guarda en **lista_productos_promScore**.

```
207  #-----------------------------------------------------------------------------------------------------------#
208  """ Identificar 5 productos con peor reseña """
209
210  #Paso 1: copiar lista ordenada alreves
211  cp_lista_productos_promScore=sorted(lista_productos_promScore, key=lambda x:x[1], reverse=False)
212
```

*Imagen 7 Código para obtener los promedios score de los productos con peor reseña*

- Para los productos con peor reseña, solo se hace una copia de **lista_productos_promScore** y se ordena de forma **ascendente**. Se imprimen solo los primeros cinco elementos.

## → Venta e ingresos por mes y anual

```
225  #-----------------------------------------------------------------------------------------------------------#
226  """ Total de ventas e ingresos por mes """
227
228  #Paso 1: obtener id_sale y fecha de cada  producto vendido, no incluir los productos devueltos
229  ventas_por_fecha=[[venta[0],venta[3]] for venta in lifestore_sales if venta[4]==0]
230
231  #Paso 2: crear diccionario donde las keys son el numero de mes de la fecha
232  ventas_por_mes={}
233  for elemento in ventas_por_fecha:
234      mes=elemento[1][3:5]
235      #Mientras el mes no sea una key del diccionario
236      if mes not in ventas_por_mes.keys():
237          ventas_por_mes[mes]=[]
238      ventas_por_mes[mes].append(elemento[0])
```

*Imagen 8 Código que obtiene el id_sale y la fecha de cada venta*

- Se define **ventas_por_fecha** que contiene el id_sale y la fecha de las ventas de **lifestore_sales**.
- Se creó un diccionario **ventas_por_mes** el cual tiene como keys los meses de las fechas. La variable **mes** extrae de la cadena fecha las letras de los índices 4 y 5. Sí se tiene '24/03/2020' la variable mes queda '03' y así para cada una.
- Se verifica que las keys del diccionario no este el mes para poder ser agregado.
- Se agrega a la key actual el valor del id_sale y así para cada iteración.

```
240  #Paso 3: crear otro diccionario de la forma {'mes':[total_ventas][total_ingresos], ... N}
241  venta_productos_por_mes={}
242  for key in ventas_por_mes.keys():
243      #Obtengo la lista de id_venta de la key actual
244      lista_mes=ventas_por_mes[key]
245      suma_venta=0
246      for id_venta in lista_mes:
247          indice=id_venta-1
248          info_venta=lifestore_sales[indice]
249          id_product=info_venta[1]
250          precio=lifestore_products[id_product-1][2]
251          suma_venta+=precio
252      venta_productos_por_mes[key]=[]
253      venta_productos_por_mes[key].append(len(lista_mes))
254      venta_productos_por_mes[key].append(suma_venta)
255
256  #Paso 4: obtener las ventas  e ingresos de todo el año
257  total_ventas=0
258  total_ingresos=0
259  for key in venta_productos_por_mes.keys():
260      total_ventas+=venta_productos_por_mes[key][0]
261      total_ingresos+=venta_productos_por_mes[key][1]
```

*Imagen 9 Código para obtener las ventas e ingresos por mes y del año*

- Se definió el diccionario **venta_productos_por_mes** que tiene la forma: {'mes':[ventas_totales,ingresos_totales], … N}
- Se itera por **ventas_por_mes** a través de sus keys y se obtiene **lista_mes** para cada key. Recuerde que dicha lista contiene los id_venta.

- Se creó **suma_venta** que guarda la suma de los ingresos de cada mes.
- Se iteró sobre **lista_mes** y con **índice** se obtiene la posición actual, dicho índice se le pasa como parámetro a **lifestore_sale** para obtener la información de esa venta y se guarda en **info_venta**.
- **id_product** guarda el identificador del producto de la venta especifica.
- **precio** obtiene el precio del producto de la posición que se le pasa, el [2] especifica el campo a obtener, en este caso price.
- Se realizó la suma entre las variables **suma_venta** y **precio**.
- Al final se agrega a **venta_productos_por_mes** la key actual, el tamaño de la lista **lista_mes** (para ello se usa la función **len**) y el total de ingresos que corresponde a la variable **suma_venta**. Esto se hace por cada iteración.

```
256  #Paso 4: obtener las ventas  e ingresos de todo el año
257  total_ventas=0
258  total_ingresos=0
259  for key in venta_productos_por_mes.keys():
260      total_ventas+=venta_productos_por_mes[key][0]
261      total_ingresos+=venta_productos_por_mes[key][1]
```

*Imagen 10 Código para obtener las ventas e ingresos anuales*

- Se iteró sobre el diccionario **venta_productos_por_mes** a través de sus keys.
- **total_ventas** guarda el total de ventas anual.
- **total_ingresos** guarda el total de ingresos anual.
- Se van sumando las respectivas variables.

## → Login de acceso

```
#-------------------------------------------------------------------------------------------
""" Login principal del programa """

#Credenciales de administrador
#usuario="SYS_admin"
#contrasena="SYS_admin%95"

#Variable para limitar los intentos de acceso al sistema
intentos=0
opcion=0
while True:
    print("-------------------------------------------")
    print("| B I E N V E N I D O   A L   S I S T E M A |")
    print("-------------------------------------------")
    usuario=input("Ingrese su usuario: ")
    contrasena=input("Ingrese su contraseña: ")
    intentos+=1

    #Credenciales validas, entonces se muestra un menu del reporte
    if usuario=="SYS_admin" and contrasena=="SYS_admin123":
```

*Imagen 11 Código que lee los datos que ingresa el usuario*

Se definió el usuario SYS_admin y la contraseña SYS_admin123 como credenciales validas para que el administrador ingrese al sistema. Se definió **intentos** para limitar un máximo de 3 veces al intentar ingresar al sistema con credenciales invalidas. La variable **opcion** permite elegir entre un menú principal una vez que se ingresó al sistema.

Se crea un ciclo y se da un mensaje de bienvenida, después se piden las credenciales para acceder al sistema, los cuales se guardan en las variables **usuario** y **contraseña**. El primer if verifica que las credenciales sean validas.

```
        while opcion!=4:
            print("\n")
            print("********* R E P O R T E *********")
            print("1: Productos más vendidos y productos rezagados")
            print("2: Productos por reseña en el servicio")
            print("3: Ventas e ingresos por mes y anual")
            print("4: Salir")
            opcion=int(input("Elija una opción: "))

            if opcion==1:···

            elif opcion==2:···

            elif opcion==3:···

        print("\n")
        print("Saliendo del sistema ... ")
        exit()
```

*Imagen 12 Código que muestra un menú una vez que se ingresa al sistema con credenciales validas*

El ciclo interno y externo se rompe cuando la variable **opcion** toma el valor de **4** se llama a la función **exit()**. Dicha variable guarda la opción que elige del menú el usuario. Si el usuario elige la opción **1** se muestra la información de top 5 productos con mayor venta, top 10 de productos con mayor búsqueda, top 5 de productos con menor venta por categoría y top 10 de productos con menor búsqueda por categoría.

Sí se elige la opción **2** se muestran el top 5 de productos con mejor reseña y los 5 productos con peor reseña. Mientras que la opción **3** muestra las ventas e ingresos totales por mes y anual.

```
399     elif usuario=="SYS_admin" and contrasena!="SYS_admin123":
400         print("----> Error en contraseña")
401         print("\n")
402 
403     elif usuario!="SYS_admin" and contrasena=="SYS_admin123":
404         print("----> Error en usuario")
405         print("\n")
406 
407     else:
408         print("----> Error en usuario y contraseña")
409         print("\n")
410 
411     if intentos==3:
412         print("Ya intentaste acceder 3 veces")
413         print("\n")
414         exit()
```

*Imagen 13 Código que valida las credenciales de acceso*

Una vez que se sale del ciclo interno del menú del sistema, se regresa a la pantalla donde se piden las credenciales de acceso. Se vuelve a validar usuario, contraseña y ambos, en cada caso se envía un mensaje de cual ha sido el error. Si el usuario llega a 3 intentos se le saca del sistema.

## | Resultados

Repositorio de github: https://github.com/Grissie/LifeStore

→ Top 5: productos más vendidos

01 | SSD Kingston A400 120GB

*Discos duros*

*50 ventas*

02 | Procesador AMD Ryzen 5 2600

*Categoría: Procesadores*

*42 ventas*

03 | Procesador Intel Core i3-9100F

*Procesadores*

*20 ventas*

04 | Tarjeta Madre ASRock Micro ATX B450M Steel Legend

*Tarjetas madre*

*18 ventas*

05 | SSD Adata Ultimate SU800 256GB

*Discos duros*

*15 ventas*

## → Top 10: productos más buscados

01 SSD Kingston A400 120GB
*Discos duros*
*263 búsquedas*

02 SSD Adata Ultimate SU800 256GB
*Discos duros*
*107 búsquedas*

03 Tarjeta Madre ASUS micro ATX TUF B450M-PLUS
*Tarjetas madre*
*60 búsquedas*

04 Procesador AMD Ryzen 5 2600 S-AM4
*Procesadores*
*55 búsquedas*

05 Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8
*Procesadores*
*41 búsquedas*

06 Logitech Audífonos Gamer G635 7.1
*Audífonos*
*35 búsquedas*

07 TV Monitor LED 24TL520S-PU 24
*Pantallas*
*32 búsquedas*

08 Procesador Intel Core i7-9700K
*Procesadores*
*31 búsquedas*

09 Procesador Intel Core i3-9100F
*Procesadores*
*30 búsquedas*

10 XPG SX8200 Pro 256GB
*Discos duros*
*30 búsquedas*

→ Top 5: productos con mejor reseña

01 | Procesador AMD Ryzen 3 3300X S-AM4

★★★★★

02 | Procesador Intel Core i9-9900K

★★★★★

03 | Procesador Intel Core i7-9700K

★★★★★

04 | Procesador Intel Core i5-9600K

★★★★★

05 | Tarjeta de Video ASUS AMD Radeon RX 570

★★★★★

→ Top 5: productos con peor reseña

01 Tarjeta de Video Gigabyte AMD Radeon R7 370 OC

★☆☆☆☆

02 Tarjeta Madre ASRock ATX H110 Pro BTC+

★☆☆☆☆

03 Tarjeta Madre AORUS micro ATX B450 AORUS M

★★☆☆☆

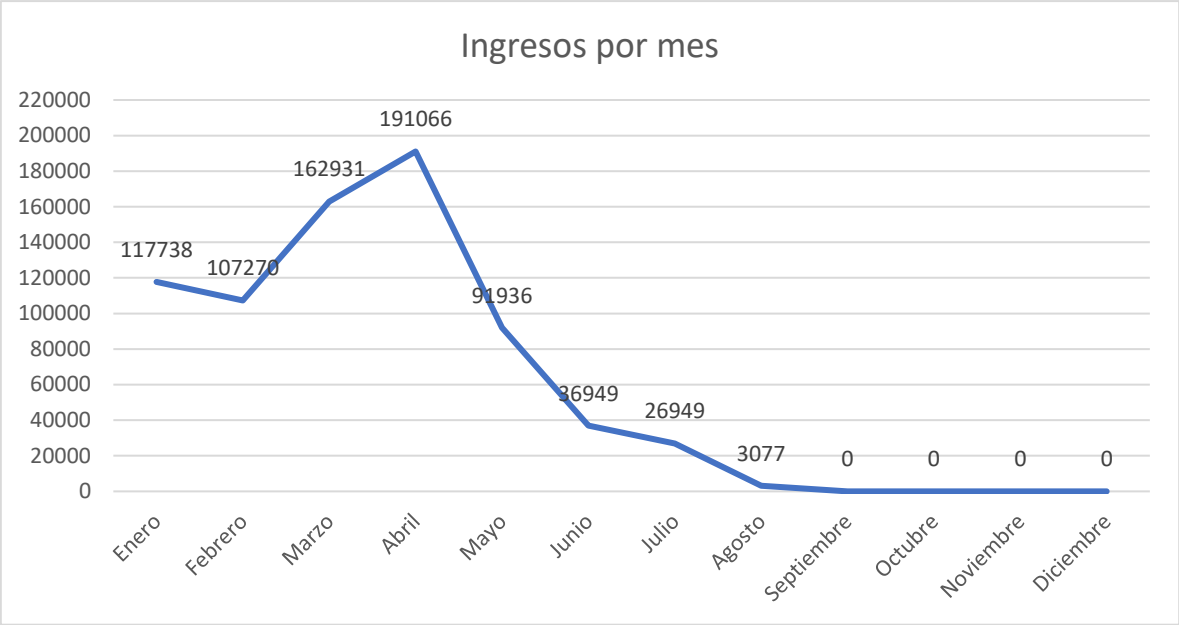
04 Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2,

★★☆☆☆

05 Cougar Audífonos Gamer Phontum Essential

★★★☆☆

## → Ventas e ingresos por mes




# | Análisis y conclusiones

Tras realizar el análisis se comprueba que el producto con más búsquedas y ventas es el disco duro **SSD Kingston A400 120GB** con 50 ventas y 263 búsquedas, ya que representa el 25 % de búsquedas de todo el año y el 18 % de ventas de todo el año.

Otro producto destacable fue el **Procesador AMD Ryzen 5 2600** con 42 ventas y 55 búsquedas. Que representan el 15 % y 5 % de ventas y búsquedas respectivamente al año. Se puede notar que las categorías de **discos duros** (94 productos vendidos) y **procesadores** (104 productos vendidos) en conjunto representan el 70 % de los productos que se vendieron en el año.

Aunque la categoría de **tarjetas de video** es la segunda categoría con más ventas, por encima de discos duros cabe destacar que hay 19 productos diferentes dentro de esta categoría, misma que solo tiene un producto con un máximo de 9 ventas ya que el resto está por debajo de las 5 ventas. Algo similar sucede con la categoría de **tarjetas madre** con 18 productos diferentes, de los cuales solo 2 productos tienen ventas mayores a 10.

Mientras que las categorías de **memorias usb** contiene solo 2 productos de los cuales ninguno se ha vendido. También la categoría de **bocinas** contiene 10 productos de los cuales solo 3 se están vendiendo con un máximo de ventas de 6.

También se identificó que los **productos con mejor reseña** pertenecen a la categoría de **procesadores** con un promedio de reseñas igual a 5, mientras que los productos con una reseña poco favorable pertenecen a la categoría de **tarjetas madre** y **tarjetas de video** con un promedio de reseñas igual a 1.

Los meses de enero, marzo y abril tuvieron 52, 49 y 74 ventas respectivamente posicionándolos como los mejores meses en cuestión de ventas. Para el caso de los ingresos esos mismos meses lideran la tabla con un total por mes de $ 117738, $ 162931 y $ 191066 respectivamente.

### Conclusiones

El proyecto me permitió poner en práctica los conceptos que se vieron en el curso, se logró trabajar con listas, tuplas y diccionarios para categorizar a los productos e identificar los productos con mayor venta. También se logró implementar ciclos iterativos, condicionales y expresiones aritméticas para identificar productos y totales de ventas e ingresos por mes.

Se logró hacer un análisis por producto, categoría y mes lo cual a su vez permite identificar que productos se tienen que seguir vendiendo y cuales es mejor retirar. Por ejemplo, los productos de las categorías de **memorias usb** y **bocinas** es mejor **retirarlas de la tienda**, ya que sus ventas y búsquedas no son rentables. Los productos que pertenecen a las categorías **tarjetas de video** y **tarjetas madre** tienen malas reseñas por parte de los clientes, mismas categorías que tienen un promedio de 18 productos cada una, de los cuales solo se venden 4, entonces se podrían retirar los productos de estas categorías con bajas ventas.

# | Pruebas de ejecución

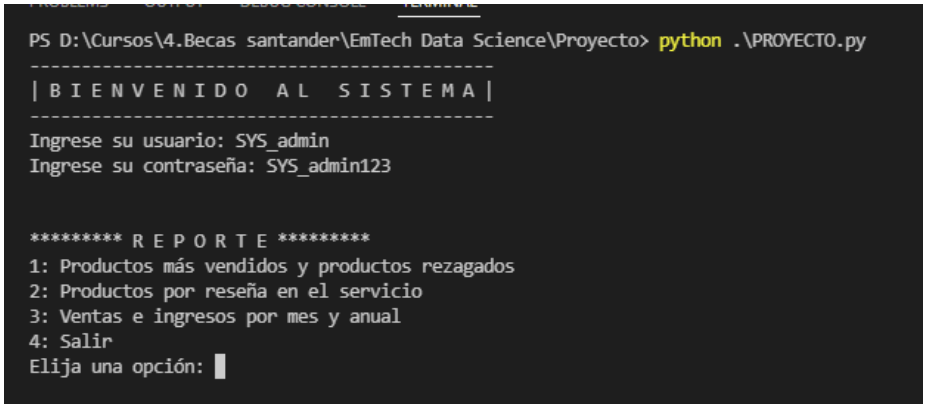



*Imagen 14 Login de acceso*


*Imagen 15 Ejemplo de como se muestran los 5 productos con mayor venta*

```
------------------------------------------------------------------------
|T O P  1 0 : P R O D U C T O S  C O N  M A Y O R  B U S Q U E D A|
------------------------------------------------------------------------
Id de producto:  54
  Nombre de producto:  SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm
  Categoría del producto:  discos duros
  Número de busquedas:  263


Id de producto:  57
  Nombre de producto:  SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm
  Categoría del producto:  discos duros
  Número de busquedas:  107


Id de producto:  29
  Nombre de producto:  Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD
  Categoría del producto:  tarjetas madre
  Número de busquedas:  60
```

*Imagen 16 Ejemplo de como se muestran los 10 productos más buscados*

```
------------------------------------------------------------------------------------------
|T O P  5 : P R O D U C T O S  C O N  M E N O R  V E N T A  P O R  C A T E G O R I A|
------------------------------------------------------------------------------------------
---------------------------------
Categoria:  procesadores
---------------------------------
   Id de producto:  1
   Nombre de producto:  Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache
   Categoría:  procesadores
   Número de ventas:  2


   Id de producto:  6
   Nombre de producto:  Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake)
   Categoría:  procesadores
   Número de ventas:  3


   Id de producto:  8
   Nombre de producto:  Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake)
   Categoría:  procesadores
   Número de ventas:  4
```

*Imagen 17 Ejemplo de como se muestran los 5 productos con menor venta por categoría*

```
------------------------------------------------------------
|T O P  5 : P R O D U C T O S  C O N  M E J O R  R E S E Ñ A|
------------------------------------------------------------
Id producto:  1
Nombre de producto:  Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache
Promedio de reseñas:  5.0
Id producto:  6
Nombre de producto:  Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake)
Promedio de reseñas:  5.0
Id producto:  7
Nombre de producto:  Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake)
Promedio de reseñas:  5.0
Id producto:  8
Nombre de producto:  Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake)
Promedio de reseñas:  5.0
Id producto:  11
Nombre de producto:  Tarjeta de Video ASUS AMD Radeon RX 570, 4GB 256-bit GDDR5, PCI Express 3.0
Promedio de reseñas:  5.0
```

*Imagen 18 Visualización de los 5 productos con mejor reseña*

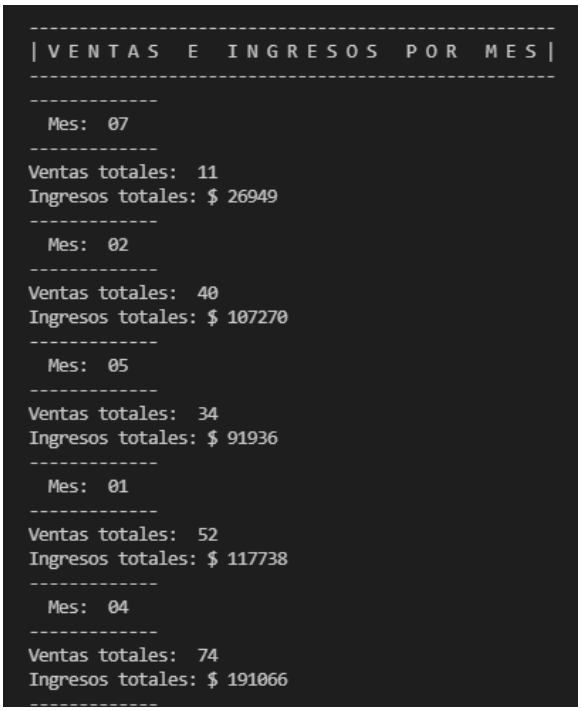



*Imagen 19 Visualización de ventas e ingresos por mes*