PROYECTO 1 DE EMTECH. DATA SCIENCE

MACEDO SERRANO ERNESTO

INGENIERO GEÓLOGO, ESPECIALIZADO EN CIENCIAS DE DATOS

ANÁLISIS ESTRATÉGICO PARA LA EMPRESA LIFESTORE SOBRE SUS VENTAS EN EL AÑO 2020

FECHA DE ENTREGA: 6 DE SEPTIEMBRE DE 2020

Índice

1. Introducción

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.

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

En las siguientes páginas se abordará el procedimiento utilizado para la realización de este análisis y se plantearán diversas soluciones basadas en los resultados obtenidos.

2. Definición de código

A continuación, se hacen breves comentarios para comprender la lógica detrás de la programación d este código. Son un total de 1123 líneas por lo que es imposible comentar cada paso, sin embargo, se dejará claro cada proceso.

Primero se buscó importar las listas otorgadas por LifeStore con la información a analizar. Se utilizó la función import y para no saturar demasiado el archivo principal, se vaciaron en un archivo distinto llamado listas.py:

main.py

```
#IMPORTACIÓN DE LISTAS:

from listas import lifestore_products
from listas import lifestore_sales
from listas import lifestore_searches

```

El inicio de sesión se planteó de la siguiente forma: primero se da la bienvenida al programa, después se solicita el tipo de usuario que quiere ingresar. Si bien se dan las opciones, el código garantiza que sólo al seleccionar la opción de Gerente se podrá continuar al siguiente paso; de lo contrario se saldrá del sistema pues no se tienen las credenciales necesarias.

Posteriormente se crea un ciclo de 3 iteraciones máximo para que se ingrese la contraseña (ls4ever) que sólo un Gerente puede tener. Si se exceden los intentos, el programa se cerrará nuevamente para prevenir fraudes o robo de información.

Si se ingresa la contraseña en los intentos estipulados, el último paso para iniciar sesión es colocando el nombre del usuario. El cual se garantiza sea un Gerente, puesto que la contraseña sólo la tienen ellos.

main.py

```
#INICIO DE SESIÓN:

print("Bienvenido al portal de información de LifeStore.")
usuario = input(
    '\nFavor de ingresar su puesto en la empresa: \na)Gerente\nb)Vendedor\nc)Otro\n'
)

if usuario != 'a':
    print(
        '\nLo siento, sólo los usuarios con calidad de administrador pueden acceder a la base de
        datos. \nFavor de ponerse en contacto con su supervisor.'
    )

else:
    i = 0
    decis = 0
    for i in range(0, 3):
        password = input(
            '\nIngrese su contraseña de administrador. Máximo 3 intentos\n')
        if password == 'ls4ever':
            print(
                '\nContrasena correcta, ahora puede acceder a la base de datos.'
            )
            admin = input("Ingrese su nombre, por favor\n")
            print("\nHola gerente", admin)
            decis = 1
            break

        else:
            i += 1
            print("Contraseña incorrecta, intento número", i)
```

Posteriormente se presenta el Menú principal del programa, al que se regresará cada vez que se concluya una tarea o que ocurra algún error cuando un usuario ingrese una opción inválida. Este se programó con un ciclo while que se activa o desactiva con una instrucción binaria:

```
38  #MENÚ PRINCIPAL:
39
40      while decis == 1:
41          print("\nMenú principal, ¿qué desea consultar?")
42          opcion = input(
43              "\na)Menú de ventas \nb)Menú de búsquedas \nc)Menú de reseñas y devoluciones \nd)Nada,
                deseo salir\n"
44          )
45
```

El menú principal indenta a todas las opciones principales: Menú de ventas, de búsquedas, de reseñas y devoluciones y la salida del programa cuando sea voluntaria:

```
174
175             if opcion == "d":
176                 print("\nVuelva pronto gerente", admin,
177                       "fue un placer atenderle.\n")
178                 decis = 0
179
180             elif opcion != 'a' and opcion != 'b' and opcion != 'c':
181                 print(
182                     '\nNo ha seleccionado alguna opción válida, intente de nuevo')
183                 decis = 1
```

Antes de ingresar a alguna opción mediante la instrucción del usuario, se construyen listas "pivote" que serán utilizadas en diversas opciones del menú. Se programan en esta parte puesto que, si se programan dentro de alguna de las tres opciones propuestas, sus elementos (listas y algunas variables) no podrían utilizarse en las otras dos. Algunos ejemplos:

```
46              ventas = []
47              for producto in lifestore_products:
48                  contador = 0
49                  for venta in lifestore_sales:
50                      if producto[0] == venta[1]:
51                          contador += 1
52                  ventas_formato = [
53                      producto[0], producto[1], producto[3], producto[4], contador,
54                      producto[2]
55                  ]
56                  ventas.append(ventas_formato)
57
58              busquedas = []
59              for producto in ventas:
60                  contador = 0
61                  for busca in lifestore_searches:
62                      if producto[0] == busca[1]:
63                          contador += 1
64                  busca_formato = [
65                      producto[0], producto[1], producto[2], producto[3],
66                      producto[4], contador
67                  ]
68                  busquedas.append(busca_formato)
69
70              categorias = []
71              for producto in busquedas:
72                  if producto[2] not in categorias:
73                      categorias.append(producto[2])
74
```

Una vez programadas las listas que se consideran necesarias mediante distintas herramientas (ciclos for, while y condicionales), se procede a desarrollar cada una de las tres grandes opciones que ofrece el menú general.

MENÚ DE VENTAS:

Se presenta un submenú con cuatro opciones principales. Antes de ingresar a cada una se asegura el código de que, en caso de ingresar una opción errónea, se regrese al menú principal y pueda volver a elegirse una opción correcta o, en su defecto, salir del programa:

```
184  #INGRESO FORMAL A A
185          elif opcion == 'a':
186
187              print('\nEste es el menú de ventas, elija una opcion:')
188              op_ventas = input(
189                  '\na1)Productos más vendidos globales \na2)Productos menos vendidos globales\na3)
                     Ventas por categoría\na4)Ingresos y ventas\n'
190              )
191
192              if op_ventas != 'a1' and op_ventas != 'a2' and op_ventas != 'a3' and op_ventas != 'a4' and
                 op_ventas != 'a5':
193                  print(
194                      '\nNo ha seleccionado alguna opción válida, volverá al menú principal'
195                  )
196                  decis = 1
197
```

En general, una vez que se hayan creado las listas que se mencionaron previamente, sólo se utilizaron en diversas ocasiones dependiendo de los elementos sobre los que se deseaba iterar y adaptando las condiciones a solicitudes particulares. Por ejemplo, ventas fue una lista que se usó en diversas tareas y para armarla fue necesario iterar en ciclo doble sobre las listas primarias: lifestore_products y lifestore_sales. Se utilizaba la coincidencia del id de cada producto para contar las veces que uno se repetía en la sección de ventas y así añadir ese dato en una nueva lista: ventas.

La estructura de las demás listas como busquedas, resenas, devueltos y otras más, se crearon de la misma manera.

```
45
46              ventas = []
47              for producto in lifestore_products:
48                  contador = 0
49                  for venta in lifestore_sales:
50                      if producto[0] == venta[1]:
51                          contador += 1
52                  ventas_formato = [
53                      producto[0], producto[1], producto[3], producto[4], contador,
54                      producto[2]
55                  ]
56                  ventas.append(ventas_formato)
57
```

Para las opciones de ordenamiento, ya fuera ascendente o descendente, se utilizó el siguiente principio básico: se creó una lista vacío y luego se accedió a un ciclo while que funcionaría mientras la lista iterada (en este caso ventas) tuviera elementos. Se utilizaba un pivote que cambiaba cada que era sustituido por un nuevo valor más, o menos grande (dependiendo del orden deseado). Finalmente la lista iterada se vaciaba y el resultado eran sus elementos en la lista definida al inicio.

Para imprimir las listas finales, usualmente se utilizaba un ciclo for para sólo imprimir los elementos deseados y para dar un mejor formato de salida. También se solía definir listas nuevas con sólo los elementos solicitados por las instrucciones. Hubo algunos inconvenientes

para realizar esto al pie de la letra puesto que no siempre se completaban las cantidades solicitadas.

main.py

```
198                 elif op_ventas == 'a1':
199
200                     ventas_ordenadas_1 = []
201                     while ventas:
202                         maximo = ventas[0][4]
203                         valor_actual = ventas[0]
204                         for venta in ventas:
205                             if venta[4] > maximo:
206                                 maximo = venta[4]
207                                 valor_actual = venta
208                         ventas_ordenadas_1.append(valor_actual)
209                         ventas.remove(valor_actual)
210
211                     print(
212                         '\nFormato de presentación de datos: id, Producto, Stock, Ventas\n'
213                     )
214
215                     mayores_ventas = []
216                     i = 1
217                     for venta in ventas_ordenadas_1:
218                         if venta[4] == 0:
219                             continue
220                         else:
221                             mayores_ventas.append(venta)
222                             print(i, '.-', venta[0], ',', venta[1], ',', venta[3],
223                                   ',', venta[4], '\n')
224                             i += 1
225
226                     print(
227                         'Estos son los', i - 1,
228                         'productos con mayores ventas globales.\nLos demás no registraron ninguna.\n'
229                     )
```

Para las secciones que requerían analizar los datos por categorías, se crearon, como las otras listas mencionadas anteriormente, antes de ingresar a alguna de las opciones principales del menú, de modo que pudieran utilizarse en todas ellas. Primero se creó una lista genérica de categorías donde se almacenaron los 8 tipos distintos. Luego se programó cada una de las listas por categoría, a las que se les asignaron los valores de interés como id, nombre del producto, número de ventas, entre otros. Un ejemplo a continuación:

```
69
70                     categorias = []
71                     for producto in busquedas:
72                         if producto[2] not in categorias:
73                             categorias.append(producto[2])
74
75                     procesadores = []
76                     for producto in busquedas:
77                         if producto[2] == categorias[0]:
78                             procesadores_formato = [
79                                 producto[0], producto[1], producto[4], producto[5]
80                             ]
81                             procesadores.append(procesadores_formato)
82
```

Ahora sí, para las secciones que analizaron categorías, simplemente se creó un submenú nuevo desde el cual el usuario puede elegir la categoría a visualizar. Posteriormente el

programa ordena cada una de las categorías dependiendo el parámetro solicitado (en este caso cantidad de ventas) y las imprime:

```
261     elif op_ventas == 'a3':
262         choice = input(
263             "\n¿Qué categoría quieres consultar?\n1)Procesadores \n2)Tarjetas de video \n3)
                Tarjetas madre \n4)Discos duros \n5)USB \n6)Pantallas \n7)Bocinas \n8)Audífonos"
264         )
265
266         if choice == '1':
267             procesadores_ordenadas = []
268             while procesadores:
269                 minimo = procesadores[0][2]
270                 valor_actual = procesadores[0]
271                 for proceso in procesadores:
272                     if proceso[2] < minimo:
273                         minimo = proceso[2]
274                         valor_actual = proceso
275                 procesadores_ordenadas.append(valor_actual)
276                 procesadores.remove(valor_actual)
277
278             print(
279                 '\nVentas ascendentes de procesadores.\nFormato de presentación de datos: id,
                    Producto, Ventas\n'
280             )
281             for proceso in procesadores_ordenadas:
282                 print(proceso[0], ',', proceso[1], ',', proceso[2],
283                       '\n')
284
```

Para la opción de ingresos y ventas del menú de ventas, se requirieron procesos adicionales: paralelamente a la creación de listas para cada categoría, se programaron listas para cada uno de los meses del año, en las que se registraron datos de interés como id, nombre, precio, número de ventas mensuales, entre otros. Esto se hizo con un mismo barrido doble entre dos listas que tuvieran los datos deseados: ventas y lifestore_sales. Las iteraciones en la segunda lista fueron un poco distintas pues se agregó un índice extra que permitiera ubicar los dos dígitos que indican el mes de cada venta. Una vez localizados se usó nuevamente un contador que permitió generar una lista para cada mes con el dato deseado: venta de cada producto al mes.

Finalmente, se calcularon los ingresos y las ventas totales para cada mes y se utilizaron más adelante, cuando el usuario accede a cada una de esas opciones y las imprime. Es necesario aclarar que no se contaron aquellos productos que fueron devueltos pues no formaban parte de los ingresos y de las ventas al cierre de caja.

```
414
415     elif op_ventas == 'a4':
416         choice = input(
417             "\nElija una opcion:\n1)Enero \n2)Febrero \n3)Marzo \n4)Abril \n5)Mayo \n6)Junio
                \n7)Julio \n8)Agosto\n9)Septiembre\n10)Octubre\n11)Noviembre\n12)Diciembre\n13)
                Ingresos y ventas mensuales y totales\n"
418         )
419
420         enero = []
421         for producto in ventas:
422             i = 0
423             for venta in lifestore_sales:
424                 if producto[0] == venta[1] and venta[3][
425                         3:5] == '01' and venta[4] == 0:
426                     i += 1
427             if i > 0:
428                 enero_formato = [
429                     producto[0], producto[1], i, producto[5],
430                     i * producto[5]
431                 ]
432                 enero.append(enero_formato)
433
434         enero_ventas = 0
435         enero_ingresos = 0
436         for venta in enero:
437             enero_ventas += venta[2]
438             enero_ingresos += venta[4]
439
```

En la imagen anterior también se nota un nuevo submenú que permite acceder a cada mes y a una opción que desglosa todos los cálculos para conocer los mejores meses, los ingresos totales y las ventas e ingresos por promedio mensual. Así se calcularon, a partir de las listas que ya se tenían construidas:

```
722                 if choice == '13':
723
724                     total_ingresos = 0
725                     total_ventas = 0
726                     for ingreso in meses:
727                         total_ingresos += ingreso[2]
728                         total_ventas += ingreso[1]
729
730                     promedio_ingresos = total_ingresos / 12
731                     promedio_ventas = total_ventas / 12
732                     promedio_ingresos2 = total_ingresos / 8
733                     promedio_ventas2 = total_ventas / 8
734
746                 meses_ordenados = []
747                 while meses:
748                     maximo = meses[0][2]
749                     valor_actual = meses[0]
750                     for mes in meses:
751                         if mes[2] > maximo:
752                             maximo = mes[2]
753                             valor_actual = mes
754                     meses_ordenados.append(valor_actual)
755                     meses.remove(valor_actual)
756
757                 print(
758                     '\nMeses ordenados por mayores ingresos.\nFormato de datos: Mes, #Ventas
                        mensuales, Ingresos al mes\n'
759                 )
760                 for mes in meses_ordenados:
761                     print(mes, '\n')
```

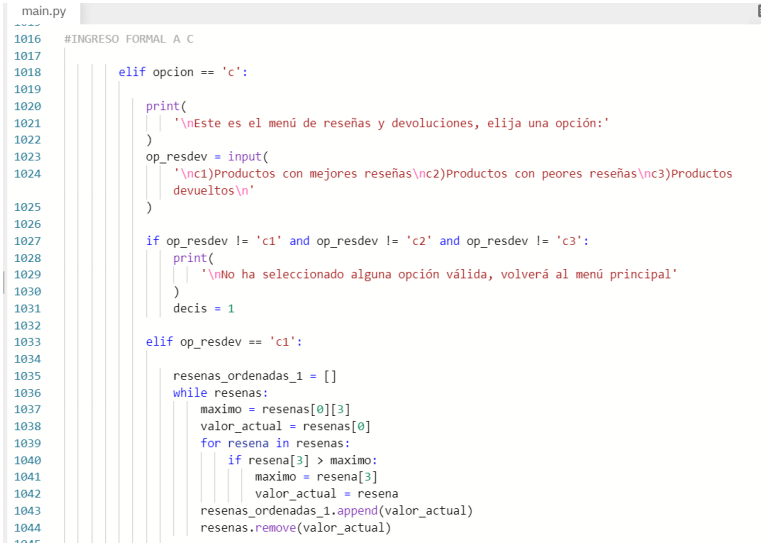
MENÚ DE BÚSQUEDAS

Para el menú de búsquedas se propone un submenú que funciona de manera similar a las opciones del menú anterior, sólo que utiliza la lista busquedas como pivote para ordenarlas a voluntad. También ordena en b3 por categorías a las búsquedas de cada producto. Se barrió la lista lifestore_searches con un contador y acumuló las veces que aparecía cada producto. No se detallan los procedimientos con imágenes, pero como se ha mencionado, se utilizaron las mismas estructuras que en el menú anterior. Ejemplo:

```
773 #INGRESO FORMAL A B
774
775         elif opcion == 'b':
776
777             print('\nEste es el menú de búsquedas, elija una opción:')
778             op_busca = input(
779                 '\nb1)Productos más buscados globales\nb2)Productos menos buscados globales\nb3)
                    Búsquedas por categorías\n'
780             )
781
782             if op_busca != 'b1' and op_busca != 'b2' and op_busca != 'b3':
783                 print(
784                     '\nNo ha seleccionado alguna opción válida, volverá al menú principal'
785                 )
786                 decis = 1
787
788             elif op_busca == 'b1':
789
790                 busquedas_ordenadas_1 = []
791                 while busquedas:
792                     maximo = busquedas[0][5]
793                     valor_actual = busquedas[0]
794                     for busqueda in busquedas:
795                         if busqueda[5] > maximo:
796                             maximo = busqueda[5]
797                             valor_actual = busqueda
798                     busquedas_ordenadas_1.append(valor_actual)
799                     busquedas.remove(valor_actual)
800
```

MENÚ DE RESEÑAS Y DEVOLUCIONES

Para este menú se utilizaron las mismas estructuras que en los pasados, sólo que se debieron hacer adecuaciones puesto que lo solicitado lo ameritaba. Por ejemplo, se adicionaron condicionales que permitieran hacer sumas y divisiones dentro de los ciclos para obtener el promedio de reseña (del 1-5) y para evitar divisiones sobre 0 en aquellos casos donde no existían reseñas. Una vez calculada la nueva lista pivote "resenas", se iteró nuevamente en ciclo while para ordenarla ascendente y descendentemente a partir del valor de la reseña primedio. También se agregó la cantidad de reseñas y se consideraron aquellaos que no habían recibido ninguna por no haber sido comprados:



Para la lista pivote de devueltos se consideraron otras pequeñas adecuaciones pues además de incluir el valor de la reseña promedio general de los productos que fueron devueltos, se contó cuántas veces habían sido devueltos y qué promedio recibieron en esas ocasiones. Así se programaron "resenas" y devueltos":

main.py

```
            resenas.append(resena_formato)
        else:
            resena_formato = [
                producto[0],
                producto[1],
                contador,
                suma / contador,
            ]
            resenas.append(resena_formato)

    devueltos = []
    for producto in resenas:
        i = 0
        suma = 0
        for devolucion in lifestore_sales:
            if producto[0] == devolucion[1] and devolucion[4] == 1:
                i += 1
                suma += devolucion[2]
            else:
                continue
        if i > 0:
            devueltos_formato = [
                producto[0], producto[1], producto[3], i, suma / i
            ]
            devueltos.append(devueltos_formato)

```

3. Solución al problema

La siguiente tabla muestra los principales productos vendidos en el año y su remanente en stock.

Formato de presentación de datos: id, Producto, Stock, Ventas

1 .- 54 , SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm , 300 , 50
2 .- 3 , Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth , 987 , 42
3 .- 5 , Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) , 130 , 20
4 .- 42 , Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD , 0 , 18
5 .- 57 , SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm , 15 , 15
6 .- 29 , Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD , 10 , 14
7 .- 2 , Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth , 182 , 13
8 .- 4 , Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire , 295 , 13
9 .- 47 , SSD XPG SX8200 Pro, 256GB, PCI Express, M.2 , 8 , 11
10 .- 12 , Tarjeta de Video ASUS NVIDIA GeForce GTX 1660 SUPER EVO OC, 6GB 192-bit GDDR6, PCI Express x16 3.0 , 0 , 9
11 .- 48 , SSD Kingston A2000 NVMe, 1TB, PCI Express 3.0, M2 , 50 , 9
12 .- 7 , Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) , 114 , 7

13 .- 31 , Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD , 120 , 6
14 .- 44 , Tarjeta Madre MSI ATX B450 TOMAHAWK MAX, S-AM4, AMD B450, 64GB DDR4 para AMD , 0 , 6
15 .- 18 , Tarjeta de Video Gigabyte NVIDIA GeForce GT 1030, 2GB 64-bit GDDR5, PCI Express x16 3.0 , 5 , 5
16 .- 8 , Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake) , 8 , 4
17 .- 6 , Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake) , 54 , 3
18 .- 11 , Tarjeta de Video ASUS AMD Radeon RX 570, 4GB 256-bit GDDR5, PCI Express 3.0 , 2 , 3
19 .- 49 , Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm , 3 , 3
20 .- 51 , SSD Kingston UV500, 480GB, SATA III, mSATA , 0 , 3
21 .- 1 , Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache , 16 , 2
22 .- 21 , Tarjeta de Video MSI AMD Mech Radeon RX 5500 XT MECH Gaming OC, 8GB 128-bit GDDR6, PCI Express 4.0 , 0 , 2
23 .- 25 , Tarjeta de Video Sapphire AMD Pulse Radeon RX 5500 XT Gaming, 8GB 128-bit GDDR6, PCI Express 4.0 , 10 , 2
24 .- 33 , Tarjeta Madre ASUS ATX PRIME Z390-A, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel , 43 , 2
25 .- 52 , SSD Western Digital WD Blue 3D NAND, 2TB, M.2 , 13 , 2
26 .- 74 , Logitech Bocinas para Computadora con Subwoofer G560, Bluetooth, Inalámbrico, 2.1, 120W RMS, USB, negro , 1 , 2
27 .- 85 , Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul , 39 , 2
28 .- 10 , MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0 , 13 , 1
29 .- 13 , Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0 , 1 , 1
30 .- 17 , Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0 , 1 , 1
31 .- 22 , Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC, 4GB 128-bit GDDR5, PCI Express x16 3.0 , 0 , 1
32 .- 28 , Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti, 6GB 192-bit GDDR6, PCI Express x16 3.0 , 3 , 1
33 .- 40 , Tarjeta Madre Gigabyte XL-ATX TRX40 Designare, S-sTRX4, AMD TRX40, 256GB DDR4 para AMD , 1 , 1
34 .- 45 , Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel , 25 , 1
35 .- 46 , Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel , 49 , 1
36 .- 50 , SSD Crucial MX500, 1TB, SATA III, M.2 , 4 , 1
37 .- 60 , Kit Memoria RAM Corsair Dominator Platinum DDR4, 3200MHz, 16GB (2x 8GB), Non-ECC, CL16, XMP , 10 , 1
38 .- 66 , TCL Smart TV LED 55S425 54.6, 4K Ultra HD, Widescreen, Negro , 188 , 1
39 .- 67 , TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro , 411 , 1
40 .- 84 , Logitech Audífonos Gamer G332, Alámbrico, 2 Metros, 3.5mm, Negro/Rojo , 83 , 1
41 .- 89 , Cougar Audífonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro. , 4 , 1

42 .- 94 , HyperX Audífonos Gamer Cloud Flight para PC/PS4/PS4 Pro, Inalámbrico, USB, 3.5mm, Negro , 12 , 1

La siguiente tabla muestra los productos menos vendidos y su remanente en stock:

Formato de presentación de datos: id, Producto, Stock, Ventas

1 .- 9 , Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generación - Coffee Lake) , 35 , 0
2 .- 14 , Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3, PCI Express 2.0 , 36 , 0
3 .- 15 , Tarjeta de Video EVGA NVIDIA GeForce GTX 1660 Ti SC Ultra Gaming, 6GB 192-bit GDDR6, PCI 3.0 , 15 , 0
4 .- 16 , Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming, 6GB 192-bit GDDR6, PCI Express 3.0 , 10 , 0
5 .- 19 , Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile, 4GB 128-bit GDDR5, PCI Express 3.0 x16 , 8 , 0
6 .- 20 , Tarjeta de Video Gigabyte NVIDIA GeForce RTX 2060 SUPER WINDFORCE OC, 8 GB 256 bit GDDR6, PCI Express x16 3.0 , 10 , 0
7 .- 23 , Tarjeta de Video MSI Radeon X1550, 128MB 64 bit GDDR2, PCI Express x16 , 10 , 0
8 .- 24 , Tarjeta de Video PNY NVIDIA GeForce RTX 2080, 8GB 256-bit GDDR6, PCI Express 3.0 , 2 , 0
9 .- 26 , Tarjeta de Video VisionTek AMD Radeon HD 5450, 1GB DDR3, PCI Express x16 2.1 , 180 , 0
10 .- 27 , Tarjeta de Video VisionTek AMD Radeon HD5450, 2GB GDDR3, PCI Express x16 , 43 , 0
11 .- 30 , Tarjeta Madre AORUS ATX Z390 ELITE, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel , 50 , 0
12 .- 32 , Tarjeta Madre ASRock Z390 Phantom Gaming 4, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel , 10 , 0
13 .- 34 , Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING WI-FI, S-AM4, AMD B550, HDMI, max. 128GB DDR4 para AMD , 2 , 0
14 .- 35 , Tarjeta Madre Gigabyte micro ATX Z390 M GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel , 30 , 0

15 .- 36 , Tarjeta Madre Gigabyte micro ATX Z490M GAMING X (rev. 1.0), Intel Z490, HDMI, 128GB DDR4 para Intel , 10 , 0
16 .- 37 , Tarjeta Madre ASRock ATX Z490 STEEL LEGEND, S-1200, Intel Z490, HDMI, 128GB DDR4 para Intel , 60 , 0
17 .- 38 , Tarjeta Madre Gigabyte Micro ATX H310M DS2 2.0, S-1151, Intel H310, 32GB DDR4 para Intel , 15 , 0
18 .- 39 , ASUS T. Madre uATX M4A88T-M, S-AM3, DDR3 para Phenom II/Athlon II/Sempron 100 , 98 , 0
19 .- 41 , Tarjeta Madre ASUS micro ATX Prime H370M-Plus/CSM, S-1151, Intel H370, HDMI, 64GB DDR4 para Intel , 286 , 0
20 .- 43 , Tarjeta Madre ASUS ATX ROG STRIX Z390-E GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel , 5 , 0
21 .- 53 , SSD Addlink Technology S70, 512GB, PCI Express 3.0, M.2 , 1 , 0
22 .- 55 , SSD para Servidor Supermicro SSD-DM128-SMCMVN1, 128GB, SATA III, mSATA, 6Gbit/s , 10 , 0
23 .- 56 , SSD para Servidor Lenovo Thinksystem S4500, 480GB, SATA III, 3.5'', 7mm , 3 , 0
24 .- 58 , SSD para Servidor Lenovo Thinksystem S4510, 480GB, SATA III, 2.5'', 7mm , 16 , 0
25 .- 59 , SSD Samsung 860 EVO, 1TB, SATA III, M.2 , 10 , 0
26 .- 61 , Kit Memoria RAM Corsair Vengeance LPX DDR4, 2400MHz, 32GB, Non-ECC, CL16 , 5 , 0
27 .- 62 , Makena Smart TV LED 32S2 32'', HD, Widescreen, Gris , 6 , 0
28 .- 63 , Seiki TV LED SC-39HS950N 38.5, HD, Widescreen, Negro , 146 , 0
29 .- 64 , Samsung TV LED LH43QMREBGCXGO 43, 4K Ultra HD, Widescreen, Negro , 71 , 0
30 .- 65 , Samsung Smart TV LED UN70RU7100FXZX 70, 4K Ultra HD, Widescreen, Negro , 7 , 0
31 .- 68 , Makena Smart TV LED 40S2 40'', Full HD, Widescreen, Negro , 239 , 0
32 .- 69 , Hisense Smart TV LED 40H5500F 39.5, Full HD, Widescreen, Negro , 94 , 0
33 .- 70 , Samsung Smart TV LED 43, Full HD, Widescreen, Negro , 10 , 0
34 .- 71 , Samsung Smart TV LED UN32J4290AF 32, HD, Widescreen, Negro , 3 , 0

35 .- 72 , Hisense Smart TV LED 50H8F 49.5, 4K Ultra HD, Widescreen, Negro , 11 , 0
36 .- 73 , Samsung Smart TV LED UN55TU7000FXZX 55, 4K Ultra HD, Widescreen, Negro/Gris , 4 , 0
37 .- 75 , Lenovo Barra de Sonido, Alámbrico, 2.5W, USB, Negro , 11 , 0
38 .- 76 , Acteck Bocina con Subwoofer AXF-290, Bluetooth, Inalámbrico, 2.1, 18W RMS, 180W PMPO, USB, Negro , 18 , 0
39 .- 77 , Verbatim Bocina Portátil Mini, Bluetooth, Inalámbrico, 3W RMS, USB, Blanco , 1 , 0
40 .- 78 , Ghia Bocina Portátil BX300, Bluetooth, Inalámbrico, 40W RMS, USB, Rojo - Resistente al Agua , 2 , 0
41 .- 79 , Naceb Bocina Portátil NA-0301, Bluetooth, Inalámbrico, USB 2.0, Rojo , 31 , 0
42 .- 80 , Ghia Bocina Portátil BX800, Bluetooth, Inalámbrico, 2.1 Canales, 31W, USB, Negro , 15 , 0
43 .- 81 , Ghia Bocina Portátil BX900, Bluetooth, Inalámbrico, 2.1 Canales, 34W, USB, Negro - Resistente al Agua , 20 , 0
44 .- 82 , Ghia Bocina Portátil BX400, Bluetooth, Inalámbrico, 8W RMS, USB, Negro , 31 , 0
45 .- 83 , Ghia Bocina Portátil BX500, Bluetooth, Inalámbrico, 10W RMS, USB, Gris , 16 , 0
46 .- 86 , ASUS Audífonos Gamer ROG Theta 7.1, Alámbrico, USB C, Negro , 20 , 0
47 .- 87 , Acer Audífonos Gamer Galea 300, Alámbrico, 3.5mm, Negro , 8 , 0
48 .- 88 , Audífonos Gamer Balam Rush Orphix RGB 7.1, Alámbrico, USB, Negro , 15 , 0
49 .- 90 , Energy Sistem Audífonos con Micrófono Headphones 1, Bluetooh, Inalámbrico, Negro/Grafito , 1 , 0
50 .- 91 , Genius GHP-400S Audífonos, Alámbrico, 1.5 Metros, Rosa , 16 , 0
51 .- 92 , Getttech Audífonos con Micrófono Sonority, Alámbrico, 1.2 Metros, 3.5mm, Negro/Rosa , 232 , 0
52 .- 93 , Ginga Audífonos con Micrófono GI18ADJ01BT-RO, Bluetooth, Alámbrico/Inalámbrico, 3.5mm, Rojo , 139 , 0
53 .- 95 , Iogear Audífonos Gamer GHG601, Alámbrico, 1.2 Metros, 3.5mm, Negro , 2 , 0

54 .- 96 , Klip Xtreme Audífonos Blast, Bluetooth, Inalámbrico, Negro/Verde , 2 , 0
Estos son los 54 productos con menores ventas globales:
no presentaron ninguna venta.

El primer punto a analizar para resolver el problema de la acumulación de stock, es el indicado en esta información. Si hay productos que se encuentran entre los más vendidos durante el año y su stock es bajo, se debe solicitar la adquisición de más inmediatamente. Tal es el caso de la Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD cuyas ventas se encuentran en 4to lugar y su stock está vacío.

Por el contrario, para aquellos productos con ventas muy bajas y con un stock muy alto, deben solicitarse su devolución a la empresa central de ser posible. Si no lo es, se debe rotar el inventario y exhibirlo mejor. Asimismo, se pueden implementar estrategias de venta como promociones para liberar ese capital detenido y poder adquirir más productos que se están vendiendo en mayor proporción. Ejemplo de ello es la pantalla Makena Smart TV LED 40S2 40", Full HD, Widescreen, Negro cuyo stock es el más alto entre los menos vendidos y acumula cero ventas en el año.

A continuación, se presenta una tabla de las ventas por categorías:

Ventas ascendentes de procesadores.

Formato de presentación de datos: id, Producto, Ventas

9 , Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generación - Coffee Lake) , 0
1 , Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache , 2
6 , Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake) , 3
8 , Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake) , 4
7 , Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) , 7
2 , Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth , 13
4 , Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire , 13

5 , Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) , 20
3 , Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth , 42

Para cada uno de los datos presentes en estas tablas por categorías, es necesario hacer rotaciones de productos como se propuso previamente. Como usualmente el inventario se encuentra exhibido por categorías, esto puede ser de utilidad para exhibir o crear promociones para aquellos productos que se venden poco y que tienen mucho stock (por categorías. Asimismo, se obvia la sugerencia de no adquirir más inventario en estos productos cuyas ventas son muy bajas.

A continuación se presenta una tabla con los productos devueltos en el año:

Formato de presentación de datos: id, Producto, Reseña promedio (de 1 a 5), #Devoluciones, Reseña Promedio de devoluciones:

1 - [2, 'Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth', 4.230769230769231, 1, 3.0]
2 - [17, 'Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0', 1.0, 1, 1.0]
3 - [29, 'Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD', 4.142857142857143, 1, 1.0]
4 - [31, 'Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD', 1.8333333333333333, 3, 1.0]
5 - [45, 'Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel', 1.0, 1, 1.0]
6 - [46, 'Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel', 2.0, 1, 2.0]
7 - [54, "SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm", 4.72, 1, 2.0]
Estos son los 7 productos devueltos con sus respectivas reseñas promedio (globales y de devoluciones).

Es claro que hay una relación entre las malas reseñas y los productos que se devolvieron. Es preciso poner atención al cliente y saber qué hay de malo en estos productos. Visualizar la posibilidad de sustituirlos por mejores opciones.

Finalmente se presentan los datos de ventas e ingresos:

Tener cuidado con el sesgo de aquellos 4 meses en que no se vendió nada. Se hace un cálculo del promedio mensual sin considerarlos.

Ingresos anuales totales: $ 737916
Ventas totales anuales (devoluciones descontadas): 274
Promedio mensual de ingresos: $ 61493.0
Promedio mensual de ventas: 22.833333333333332
Si se consideran sólo los meses donde hubo ventas registradas(8), entonces los mismos promedios quedan así: $ 92239.5 , 34.25 ventas al mes
Meses ordenados por mayores ingresos.
Formato de datos: Mes, #Ventas mensuales, Ingresos al mes

Se analizan los mejores meses de ventas. E importante considerar los factores externos. Si no se pueden revertir, hay que enfocarse en estos meses para sacar más inventario

['abril', 74, 191066]
['marzo', 49, 162931]
['enero', 52, 117738]
['febrero', 40, 107270]
['mayo', 34, 91936]
['junio', 11, 36949]
['julio', 11, 26949]
['agosto', 3, 3077]
['septiembre', 0, 0]
['octubre', 0, 0]

['noviembr', 0, 0]
['diciembre', 0, 0]

4. Conclusión

- Rotación de productos.
- Promociones en productos con exceso de stock
- Mejorar estrategias de marketing para aquellos productos que son menos buscados
-

Enlaces:

El enlace para el repositorio Github es el siguiente. Se encuentra un zip con los dos archivos .py utilizados. Asimismo, se ha nombrado como PROYECTO-01-MACEDO-ERNESTO.py al archivo principal y listas.py al archivo complementario desde el que se cargaron las listas de la empresa LifeStore.

https://github.com/Ernesto06/EMTECH