EMTECH

Emerging Technologies Institute

Proyecto 1. Introducción a Python

"Lifestore Project"

Tutor: Alonso Sánchez Jaime Saúl

EmTech: Rojas Pimentel Julieta K.

---------- Índice ----------

-------- Introducción --------

La tienda virtual “LifeStore” presenta una gran acumulación de inventario y una reducción en las búsquedas de una parte importante de los productos que tienen en stock, esto ha redundado en una gran disminución de sus ventas en el último trimestre. Para encontrar una solución ante esta problemática se requiere un análisis de la rotación de productos, identificando los siguientes elementos:

1. Productos más vendidos y productos rezagados a partir de un análisis de las categorías con menores ventas y búsquedas.
2. Productos por reseña en el servicio a partir del análisis de categorías con mayores ventas y búsquedas.
3. Con base en los datos de ingresos y ventas mensuales, sugerir una estrategia de productos a retirar del mercado y una manera de reducir la acumulación de inventario.

Este trabajo tiene como objetivo solucionar la problemática a través de la resolución de las consignas anteriormente mencionadas y establecer una estrategia viable de acuerdo a los resultados obtenidos de los análisis.

---- Definición del código ----

1. Base de datos.

Para realizar los análisis pertinentes, se nos proporcionó una base de datos llamada lifestore-file.py obtenida del siguiente enlace perteneciente a la plataforma GitHub: https://github.com/emtechinstitute/proyecto1

Esta base de datos contiene información acerca de los productos ofrecidos por la tienda y se encuentra repartida en tres listas: lifestore_products, lifestore_sales y lifestore_searches.

```
1   """
2   This is the LifeStore_SalesList data:
3
4   lifestore_searches = [id_search, id product]
5   lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for
    true or 0 to false)]
6   lifestore_products = [id_product, name, price, category, stock]
7   """
8
9   lifestore_products = [⋯
107
108 lifestore_sales = [⋯
393
394 lifestore_searches = [⋯
```

2. Login.

Para comenzar crearemos un login para que sólo el administrador pueda acceder a la información obtenida del código:

```
1430  # Usuario = Administrador
1431  # Contraseña = L1f3St0r3
1432  # Generación de Login
1433  # Solicitamos nombre de usuario
1434  Usuario = input("Ingrese su nombre de usuario: ")
1435  # Solicitamos contraseña
1436  Contraseña = input("Ingrese la contraseña: ")
1437  # Mantenemos al usuario en un loop hasta obtener el usuario y la contraseña corrrecta
1438  while Usuario != "Administrador" or Contraseña != "L1f3St0r3":
1439      # Si se cumple el while se imprime lo siguiente:
1440      print("Usuario o contraseña incorrectos, intente nuevamente")
1441      # Y se vuelve pedir Usuario y Contraseña para volver a probar el bucle while
1442      Usuario =  input("Ingrese su nombre de usuario: ")
1443      Contraseña = input("Ingrese la contraseña: ")
```

Como puede observarse en este fragmento de código, para realizar el login ocupamos dos input, uno para el usuario y otro para la contraseña. Se utilizó el bucle while de manera que si el usuario introduce incorrectamente alguno de los input, nos mantendrá en el bucle hasta introducir correctamente el usuario y la contraseña.

3. *Top* 15 vendidos y *top* 20 buscados.

Generamos dos listados, el primero corresponde a los 15 productos más vendidos y el segundo a los 20 productos más buscados:

```
1445  print("")
1446  # 15 productos más vendidos
1447  # Creamos una lista vacía
1448  productos = []
1449  # Iteramos sobre lifestore_products
1450  for product in lifestore_products:
1451      name = product[1]
1452      lista = [product[0], name, 0, 0]
1453      # Por cada iteración agregamos una lista a productos que contiene el id del
      producto
1454      # Su nombre y dos ceros para usarlos como contadores
1455      productos.append(lista)
1456  print("Los 15 productos más vendidos.")
1457  print("")
```

Para obtener estos listados tenemos que comenzar creando una lista vacía donde almacenaremos los datos relevantes generados en las siguientes líneas de código. Debemos iterar sobre cada producto en la lista lifestore_products entonces generamos una lista interna, en este caso llamada “lista” donde guardamos el id del producto (producto[0] donde 0 corresponde al índice de id_producto), el nombre del producto (definido en la línea 1451), dos ceros que utilizaremos como contadores (que comenzarán en 0): uno para las ventas totales y otro para las búsquedas (que ocuparemos más adelante).

En la línea 1455 incluiremos nuestra lista interna en la lista de productos. Los print vacíos (“ ”) nos servirán como un espacio entre cada apartado.

```
1458  # Iteramos sobre lifestore_sales y vemos si venta[4] = devolución es falso para
      agregar a nuestra lista productos
1459  # A cada producto y a su correspondiente contador le vamos sumando un 1 cada vez que
      tengamos una venta sin devolución
1460  for venta in lifestore_sales:
1461      if venta[4] == 0:
1462          productos[venta[1]-1][2]+=1
1463  # Ordenamos nuestra lista productos de acuerdo al número de ventas sin devolución de
      manera decreciente
1464  ordena_por_ventas = sorted(productos, key=lambda venta: venta[2], reverse=True)
1465  # Identificamos al top15
1466  top_15_ventas = ordena_por_ventas[0:15]
1467  # Iteramos sobre top_15_ventas y enumeramos para tener una mejor presentación
1468  for i,j in enumerate(top_15_ventas):
1469      print(str(i+1) + ".\tEl producto " + str(j[0]) + " tiene " + str(j[2]) + "
      ventas.")
1470  print("")
```

Es necesario que para generar la lista de “Los 15 productos más vendidos” tomemos en cuenta las devoluciones, ya que no deben contabilizarse como ventas concretadas. El ciclo for de la línea 1460 nos permite contabilizar las ventas únicamente si no ha habido reembolso *i.e* if venta[4] == 0. Únicamente si se cumple esta condición se contabilizará como venta cada vez que el id del producto aparezca como se observa en la línea 1462. Posteriormente utilizamos la función sorted para ordenar los productos de acuerdo al número de ventas de manera descendente y comenzando con el más vendido, luego definimos el top_15_ventas con los 15 productos más vendidos, enumeramos e imprimimos. En este punto utilizamos str para convertir nuestros números a cadenas y éstas puedan visualizarse en el print, además se usó \t para añadir un espacio que separa la enumeración del texto.

Para el caso de la segunda lista que corresponde a los 20 productos más buscados, añadiremos un producto cada vez que aparezca el id del producto y ordenaremos con respecto al número de búsquedas de manera decreciente y comenzando con el producto más buscado. Definimos top20searches como los 20 productos más buscados, enumeramos e imprimimos de la misma manera que la lista anterior.

```
1472  # 20 productos más buscados
1473  # Iteramos sobre lifestore_searches para sumar un uno en el otro contador que
      agregamos a productos
1474  # Por cada vez que se busque un producto
1475  for search in lifestore_searches:
1476      productos[search[1]-1][3]+=1
1477  # Ordenamos productos de acuerdo al número de veces que se busca un producto
1478  ordenamiento1 = sorted(productos, key=lambda search: search[3], reverse=True)
1479  # Identificamos el top20
1480  top20searches = ordenamiento1[0:20]
1481  print("Los 20 productos más buscados")
1482  print("")
1483  # Iteramos sobre top20searches y enumeramos para tener una mejor presentación
1484  for i,j in enumerate(top20searches):
1485      print(str(i+1) + ".\tEl producto " + str(j[0]) + " tiene " + str(j[3]) + "
      búsquedas.")
1486  print("")
```

4. *Bottom* 5 en ventas por categoría y *bottom* 20 buscados.

En este punto debemos comenzar con la separación de los productos por categoría, para esto es necesario generar una lista vacía:

```
1488  # Separación de productos por categoría
1489  # Obtención de las categorías existentes
1490  # Creamos una lista vacia donde guardaremos las categorías
1491  categorias = []
1492  # Revisión de la lista entera de productos
1493  for producto in lifestore_products:
1494      # Leemos la categoría del producto
1495      categoria_del_producto = producto[3]
1496      # Revisamos si esa categoría aun no esta en nuestra lista de 'categorias'
1497      if categoria_del_producto not in categorias:
1498          # Si aun no está, la agregamos con append
1499          categorias.append(categoria_del_producto)
1500
1501  # Creamos el esqueleto de esta lista:
1502  categoria_productos = []
1503  for categoria in categorias:
1504      cat_prods = [categoria, []]
1505      categoria_productos.append(cat_prods)
```

Posteriormente utilizaremos un ciclo for para revisar la categoría de cada uno de los productos y obtener las categorías totales. En las siguientes líneas de código obtendremos las categorías totales revisando cada producto y añadiendo la categoría conforme vaya apareciendo sin repetirse. Luego crearemos una lista vacía para agregarle listas que contengan cada categoría que hay junto con una lista vacía que nos permitirá añadir otra lista con los datos de id del producto y las ventas totales por cada producto. Si la categoría del objeto que se itera (producto) en lifestore_products coincide con la categoría del objeto que se itera en categoria_productos, a su vez que el id del producto en lifestore_products coincide con el id del producto en productos, incluimos una lista con el id del producto y el número de ventas correspondientes a ese producto.

```
1507  # Vamos a llenar cada lista con los productos que pertenecen a esa categoria
1508  for lista in categoria_productos:
1509      # La categoria de la lista es el primer elemento
1510      categoria_de_lista = lista[0]
1511
1512      # Vamos a revisar todos los productos de lifestore
1513      for producto in lifestore_products:
1514          # Como pretendemos guardar el ID unicamente, obtenemos el ID del producto
1515          id_del_producto = producto[0]
1516          # Para saber la categoria del producto lo obtenemos asi:
1517          categoria_del_producto = producto[3]
1518          for product in productos:
1519          # Si coinciden ambas categorias, vamos a incluir el producto en la lista
1520              if categoria_del_producto == categoria_de_lista and product[0] ==
                  id_del_producto:
1521                  lista[1].append([id_del_producto, product[2]])
```

Ya con esta segunda lista creada, procederemos a llenarla con la categoría y con los productos pertenecientes a ésta donde la categoría de la lista corresponde al primer elemento con índice [0]. Mediante el id del producto y la categoría podemos llenar la lista categoria_de_productos.

```
1523  # Los 5 productos menos vendidos por categoría
1524  print("Los 5 productos menos vendidos por categoría.")
1525  print("")
1526  # Definimos una función que nos ayudará a organizar listas de listas
1527  def Sort(sub_li):
1528      l = len(sub_li)
1529      for i in range(0, l):
1530          for j in range(0, l-i-1):
1531              if (sub_li[j][1] > sub_li[j + 1][1]):
1532                  tempo = sub_li[j]
1533                  sub_li[j]= sub_li[j + 1]
1534                  sub_li[j + 1]= tempo
1535      return sub_li
```

Luego procedemos al print, en este caso ya que queremos ir ordenando por cada sublista de la lista utilizaremos la función Sort donde ordenaremos con base en la categoría y posteriormente por el número de ventas de manera ascendente y comenzando con el producto con peores ventas.

```
1536  # Iteramos sobre categoria_productos y ordenamos por numero de ventas
1537  for x in categoria_productos:
1538      Sort(x[1])
1539  # Iteramos sobre cateogria_productos ya ordenada
1540  for j in categoria_productos:
1541      # Fijamos un contador
1542      x=0
1543      print("Los productos menos vendidos de la categoría " + str(j[0]) + " son: ")
1544      while x < len(j[1][0:5]):
1545          print("El producto " + str(j[1][x][0]) + " tiene " + str(j[1][x][1]) + "
          ventas.")
1546          x+=1
1547      print("")
```

Finalmente usaremos un ciclo while para imprimir únicamente los 5 productos con peores ventas por categoría, ya que de lo contrario nos imprimirá todos los productos y sus respectivas ventas por categoría.

```
1548  # Ordenamos de manera creciente
1549  ordenamiento2 = sorted(productos, key=lambda search: search[2], reverse=False)
1550  # Asignamos bottom20searches
1551  bottom20searches = ordenamiento2[0:20]
1552  print("Los 20 productos menos buscados por categoría.")
1553  print("")
1554  # Iteramos sobre bottom20searches
1555  for j in bottom20searches:
1556      print("El producto " + str(j[0]) + " tiene " + str(j[2]) + " búsquedas.")
```

Para el caso de los 20 productos menos buscados ya sólo tendremos que ordenar mediante la función sorted, definir bottom20searches como los 20 productos menos buscados e imprimir. Esto porque en el código de ventas ya habíamos incluido un contador para el número de búsquedas por producto.

5. Lista *top* 20 y *bottom* 20 por reseñas.

Esta parte del código también comenzará con una lista vacía que iremos llenando mediante ciclos for. Primero generamos una sublista con el id del producto y dos contadores que nos permitirán llenarlos con el número de veces que se vendió el producto y obtener la suma de las reseñas (score).

```
1558  # Listas por score
1559  # Listas de listas donde cada una de éstas tiene el id del producto, un contador para
      registrar el número de veces que se vendió un producto y otro contador para obtener la
      suma de las reseñas (score).
1560  producto_reseña = []
1561  for producto in lifestore_products:
1562      id_producto = producto[0]
1563      sublista = [id_producto, 0, 0]
1564      producto_reseña.append(sublista)
1565  for venta in lifestore_sales:
1566      prod_vendido = venta[1]
1567      reseña = venta[2]
1568      producto_reseña[prod_vendido - 1][1]+= 1
1569      producto_reseña[prod_vendido - 1][2]+= reseña
```

Posteriormente iteramos sobre la lista producto_reseña para cambiar la suma del score por el promedio del score. Es muy importante tomar en cuenta el número de ventas, éstas deben ser mayor a 0 ya que para tener un score el producto debe tener al menos una venta, tomando en cuenta que no podemos dividir entre cero y por lo tanto, obtener un promedio. En la línea 1576 obtenemos el promedio de la suma total de las reseñas entre el número de ventas del producto.

```
1570  # Iteramos sobre producto_reseña para cambiar la suma de las reseñas (score) por el
      promedio de las reseñas
1571  # ¡Ojo! ventas_totales > 0
1572  for relacion in producto_reseña:
1573      sum_reseñas = relacion [2]
1574      ventas_totales = relacion[1]
1575      if ventas_totales > 0:
1576          relacion[2] = sum_reseñas/ventas_totales
1577  # Creamos una lista para agregar sólo los productos que fueron vendidos
1578  prod_reseña = []
1579  for prod in producto_reseña:
1580      if prod[1] > 0:
1581          prod_reseña.append(prod)
```

Para la lista de los 20 productos con peores reseñas crearemos otra lista vacía y agregaremos sólo los productos que tengan al menos una venta. Para poder utilizar la función sorted multiplicaremos el promedio de las reseñas por 10^4 para eliminar los decimales y transformarlo en entero mediante la función int, ya que sorted no funciona con flotantes. De esta manera podemos ordenar y posteriormente transformar los datos nuevamente a sus valores originales. Ordenamos dos veces,

la primera para el *top* 20 y la segunda para el *bottom* 20 de los productos con respecto a sus reseñas.

```
1583  # Creamos una lista donde multiplicaremos el promedio de las reseñas por 10^(4) para
      eliminar decimales, volver los flotantes enteros y así poder usar la función sorted
      que funciona sólo con enteros.
1584  reseñas_enteros = []
1585  for x in prod_reseña:
1586      x[2] = int(x[2]*10**(4))
1587      reseñas_enteros.append(x)
1588  # Ordenamos con la función sorted()
1589  ordenamiento4 = sorted(reseñas_enteros, key=lambda relacion: relacion[2],
      reverse=False)
1590  # Asignamos los últimos 20
1591  bottom20scorep = ordenamiento4[0:20]
1592  # Hacemos una lista vacía para agregar las reseñas dividas entre 10^(4) y así no
      afectar la calificación del promedio de las reseñas
1593  bottom20score = []
1594  for x in bottom20scorep:
1595      x[2] = x[2]/(10**4)
1596      bottom20score.append(x)

1597  # Ordenamos nuevamente pero ahora de manera decreciente para así obtener el top20
1598  ordenamiento5 = sorted(reseñas_enteros, key=lambda relacion: relacion[2], reverse=True)
1599  # Identificamos los primeros 20
1600  top20scorep = ordenamiento5[0:20]
1601  # Creamos una lista vacía para irle asignando las reseñas divididas entre 10^(4) y así
      no afectar la calificación del promedio de las reseñas
1602  top20score = []
1603  for x in top20scorep:
1604      x[2] = x[2]/(10**4)
1605      top20score.append(x)
1606  print("Los 20 productos con mejores reseñas son: ")
1607  print("")
1608  # Iteramos sobre top20score y enumeramos para tener una mejor presentación
1609  for i,j in enumerate(top20score):
1610      j[2] = round(j[2],2)
1611      print(str(i+1) + ".\tEl producto " + str(j[0]) + " tuvo " + str(j[1]) + " ventas,
      con un score de " + str(j[2]))
1612  print("")
1613  print("Los 20 productos con peores reseñas")
1614  print("")
```

Imprimimos ambas listas y redondeamos los decimales con la función round para que aparezcan sólo dos decimales.

6. Total de ingresos, promedio de ventas por mes, meses con mayores ventas y total de ventas anuales.

Creamos una lista vacía que iremos llenando con las ventas mediante iteraciones con ciclos for, tomando en cuenta como ventas concretadas a aquellas que no presentan devoluciones. En el primer ciclo for definimos los elementos del formato fecha tomando en cuenta todos los caracteres y mediante un operador condicional if agregamos a la lista vacía mes por mes sin repetirse, siempre y cuando al iterar en lifestore_sales, la venta de ese mes no tenga reembolso.

```
1620  # Ventas promedio mensuales y meses con más ventas al año.
1621  # Creamos una lista vacía para ir iterando sobre lifestore_sales y así ir agregando
      cada mes que hay en el registro de ventas y sólo se agregan las ventas que no tienen
      devoluciones.
1622  m3s3s = []
1623  for venta in lifestore_sales:
1624      devuelto = venta[4]
1625      fecha = venta[3]
1626      dia = fecha[:2]
1627      mes = fecha[3:5]
1628      año = fecha[-4:]
1629      if mes not in m3s3s and devuelto == False:
1630          m3s3s.append(mes)
1631  # Creamos una lista vacía para ir iterando sobre m3s3s y así por cada iteración iremos
      agregando una lista que contenga el mes, una lista vacía donde agregaremos más datos y
      un contador que empieza a correr desde 0.
1632  meses = []
1633  for mes in m3s3s:
1634      meses.append([mes,[],0])
```

Nuevamente creamos una lista vacía para que con el segundo ciclo for (línea 1633), se vayan agregando sublistas que contengan el mes, una lista vacía y un contador que corre desde cero.

```
1635  # Iteramos sobre meses para ir agregando en la lista vacía que esta en cada sublista
      de meses el id del producto y su precio sólo si id del producto concuerda con el id
      del producto de la venta y si el mes de la venta coincide con el mes que se itera en
      meses.
1636  for mes in meses:
1637      for venta in lifestore_sales:
1638          id_prod = venta[1]
1639          for prod in lifestore_products:
1640              if prod[0] == venta[1] and venta[3][3:5] == mes[0]:
1641                  mes[1].append([id_prod,prod[2]])
1642  # Hacemos una lista vacía en donde iremos agregando el id del producto y su precio
1643  precios_prods = []
1644  for prod in lifestore_products:
1645      precios_prods.append([prod[0],prod[2]])
1646  # Iteramos sobre meses y fijamos un contador que empieza a correr desde 0
1647  for x in meses:
1648      i = 0
```

Posteriormente iteramos sobre la lista meses y se agrega en el segundo elemento de cada sublista el id del producto y su precio. Luego iteramos sobre meses y fijamos un contador que empieza en cero. El ciclo while será de utilidad para crear un bucle que termine cuando i sea el número de ventas por mes, ordenamos la lista de meses de manera descendiente comenzando con el mes que tuvo un promedio mayor de ventas.

```
1646  # Iteramos sobre meses y fijamos un contador que empieza a correr desde 0
1647  for x in meses:
1648      i = 0
1649      # Usamos el ciclo while para que a nuestro contadaor de las sublistas de meses le
          vaya agregando el precio del artículo por cada artículo de manera correspondiente
          hasta que nuestro contador alcance el número de productos que se vendieron por mes
1650      while i < len(x[1]):
1651          x[2] += x[1][i][1]
1652          i += 1
1653  # Ordenamos los meses de manera decreciente
1654  meses_ordenados = sorted(meses, key=lambda mes:mes[2], reverse=True)
1655  # Fijamos un contador que corre a partir de 0
1656  ganancia_total = 0
1657  # Iteramos sobre meses_ordenados y por cada iteración mes vamos sumando las ganancias
      para obtener la gananacia total y después su promedio.
1658  for i in meses_ordenados:
1659      ganancia = i[2]
1660      ganancia_total += ganancia
1661  promedio_ganancia_total = ganancia_total/len(meses_ordenados)
1662  print("")
1663  print("El total de ingresos es: " + str(ganancia_total) + " MXN" )
```

Para obtener el ingreso total fijamos un contador que empezará a correr a partir de cero, donde iteraremos e iremos sumando las ganancias mensuales para obtener las totales y posteriormente obtener el promedio para cada mes.

```
1665  print("Los meses que están por encima del promedio de las ganacias totales son: ")
1666  print("")
1667  # Iteramos sobre meses_ordenados y enumeramos para tener una mejor presentación
1668  for i,j in enumerate(meses_ordenados):
1669      if j[2] > promedio_ganancia_total:
1670          print(str(i+1) + ".\tEn el mes " + j[0] + " se obtuvo una ganancia de " + str(j
              [2]))
1671  # Hacemos una lista vacía para agregar sólo los meses con ventas del año 2020
1672  mEsEs = []
1673  for venta in lifestore_sales:
1674      fecha = venta[3]
1675      dia = fecha[:2]
1676      mes = fecha[3:5]
1677      año = fecha[-4:]
1678      if mes not in mEsEs and año == "2020":
1679          mEsEs.append(mes)
1680  # print(len(mEsEs))
1681  # Imprime el número de meses que hubo venta en el año 2020 que es 9
```

Ahora imprimimos solamente los meses que están por encima del promedio mensual, los cuales consideraremos como aquellos con mayores ventas.
En la línea 1672 crearemos una lista vacía para iterar venta en lifestore_sales y en este caso se agregarán los meses sin repetirse, siempre y cuando correspondan al año 2020 ya que sólo hubo una venta en el año 2019 y ésta fue reembolsada.

```
1682  # Fijamos un contador para ir sumando 1 por cada venta sin reembolso o devolución y
      obtener el número de ventas totales.
1683  ventas_totales = 0
1684  for venta in lifestore_sales:
1685      num_venta = venta[0]
1686      reembolso = venta[4]
1687      if reembolso == False:
1688          ventas_totales += 1
1689  # Calculamos el promedio de ventas sin reembolso por cada mes que hubo ventas.
1690  promedio_ventas_por_mes = ventas_totales/len(mEsEs)
1691  print("\n")
1692  print("El promedio de ventas que hay por mes es: " + str(round(promedio_ventas_por_mes,
      1)))
```

Para obtener el promedio de ventas por mes, iniciaremos fijando un contador que corre en cero para ir sumando un 1 por cada venta no reembolsada. Así obtendremos el promedio de ventas por mes, dividiendo el número de ventas totales entre el número de meses que tuvieron ventas (9). Imprimimos.

```
1693  print("El número de ventas totales (anual) es: " + str(ventas_totales))
```

Finalmente este print nos mostrará el número de ventas totales en 2020.

Repositorio de GitHub con el código LifeStore Project:
https://github.com/JulietaKR0/Proyectos-EmTech

---- Solución al problema ----

Para los productos con menores búsquedas (o sin búsquedas) implementar una estrategia de marketing con el objetivo de brindar visibilidad a estos productos y realizar un estudio de mercado para poder liquidar los productos acumulados en stock. Con base en este estudio podemos observar que los productos con menores ventas coinciden en su mayoría con los productos con menores búsquedas, por lo que mediante las estrategias antes mencionadas podríamos liberar espacio siempre y cuando no se adquieran nuevos productos que entren en estos listados.
Si aún después de llevar a cabo un proyecto de marketing y estudios de mercado, se considera que deben excluirse ciertos productos del catálogo, podrían considerarse principalmente aquellos con peores reseñas y/o que presenten devoluciones.

--------- Conclusión ---------

De acuerdo a los resultados obtenidos, puede inferirse que lo que ocasiona la acumulación del stock es la falta de visibilidad de muchos productos ya que además no tienen búsquedas. Si la mayoría de los productos contara con búsquedas y no con ventas se podría concluir que el problema radica en el precio o en las reseñas del producto pero al no ser así, concluimos que el problema se solucionaría con una campaña de marketing.