EMTECH

Emerging Technologies Institute

Nombre: Jesus Alejandro Colin Vilchis

PROYECTO 1 EMTECH

ÍNDICE

INTRODUCCION

El presente trabajo se hace con el objetivo de 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.

Se considera que 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 solicita que realice un análisis de la rotación de productos identificando los siguientes elementos:

- Productos más vendidos y productos rezagados a partir del análisis de las categorías con menores ventas y categorías con menores búsquedas.
- Productos por reseña en el servicio a partir del análisis de categorías con mayores ventas y categorías con mayores búsquedas.
- 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.

La información necesaria está contenida en el archivo lifestore-file.py, con registros de las compras, búsquedas y productos manejados por la tienda.

El trabajo fue realizado en Google Colab con la intención de poder replicar el código sin la necesidad de la replicación de un ambiente virtual, además de la comodidad de poder comentar el código

Original file is located at

https://colab.research.google.com/drive/1ScVAzYfeh4Trmk4JG42ubkhtOfYncVun

PROYECTO 1

JESUS ALEJANDRO COLIN VILCHIS

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

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.

Importamos las bases de datos a utilizar durante el proyecto

"""

```
from lifestore_file import lifestore_products, lifestore_sales, lifestore_searches
import uuid
import random
import datetime

"""Login: En este apartado se crean cuentas con diferentes diveles de acceso"""

dbLogin={}
numEmpleados = random.randint(0,10)
typeUsers = {
        'Admin':{'Edith':True},
        'Reader':{'Edith':False}
         }

for i in range(numEmpleados):
```

```
dbLogin[i]={'id':uuid.uuid1(),'name':f'User{i}','license':random.choice(list(typeUsers.keys())),'creadedon':datetime.datetime.now(tz=
None),'password':str(uuid.uuid1())[:8]}


dbLogin # se muestra la tabla como Id, usuario, permisos y contraseña


"""Aquí se valida que el nombre y la contraseña sean corrector"""


valid=False


def validador(user,password):
  valid = [users['id'] for users in dbLogin.values() if user in users['name'] and password in users['password']]
  if valid == []:
    print('No se encontro registro')
    return False
  else:
    print(f'Usuario identificado con el id: {valid[0]}')
    return True



while valid == False:
  Usuario = input("Escriba su usuario: ")
  Contraseña = input("Escriba su contraseña: ")
  valid = validador(Usuario,Contraseña)


"""Creación de la tabla de analisis y producto más vendido"""


def ordenar(diccionario,orden):
    arreglo=[[k,v] for k,v in diccionario.items()]
    # Calculamos la longitud del arreglo para tener un código más limpio
    longitud = len(arreglo)
    # Recorremos todo el arreglo
    for i in range(longitud):
        # Dentro del ciclo, volvemos a recorrerlo. Pero ahora hasta el penúltimo elemento
        for indice_actual in range(longitud - 1):
            indice_siguiente_elemento = indice_actual + 1
```

```
            # Si el actual es mayor que el siguiente, ...
            # Nota: para un orden inverso, cambia `>` por `<`
            if orden == 'min':
              if arreglo[indice_actual][1] > arreglo[indice_siguiente_elemento][1]:
                # ... intercambiamos los elementos
                arreglo[indice_siguiente_elemento], arreglo[indice_actual] = arreglo[indice_actual], arreglo[indice_siguiente_elemento]
            elif orden == 'max':
              if arreglo[indice_actual][1] < arreglo[indice_siguiente_elemento][1]:
                # ... intercambiamos los elementos
                arreglo[indice_siguiente_elemento], arreglo[indice_actual] = arreglo[indice_actual], arreglo[indice_siguiente_elemento]
    #print(arreglo)
    return arreglo


def recuentos(miLista,tipo,numberofTop,logica,promedio):
    '''
    Inicializamos las variables para ventas pero la funcion es universal
    miLista, se refiere a la lista por analizar
    indice, se refiere a la posición en el listado o columna de la tabla
    tipo, se refiere a la palabra a concatenar
    numberofTop, se refiere al numero de busquedas que se considerarán
    logica, se refiere a que si se hara el analisis ascendente o descendente
    '''
    table={}
    indice=1
    listTopSale=[]
    print('Analisis para '+tipo+'...')
    print('ordenado por '+logica+'imo')

    if promedio:
      for sale in miLista: #inicialmente era para ventas pero aplica con cualquier listado
        try:
          table[sale[indice]]['total']+=1
          table[sale[indice]]['score']+=sale[2]
        except:
          table[sale[indice]]={'total':1,'score':sale[2]}
```

```
  else:
    for sale in miLista: #inicialmente era para ventas pero aplica con cualquier listado
      try:
        table[sale[indice]]+=1
      except:
        table[sale[indice]]=1
  if promedio:
    table = {k:v['score']/v['total'] for k,v in table.items()}
    print(table)
  print(table)
  tabla = ordenar(table,logica)

  # convertimos en diccionario para manipular los datos mejor
  mylifestore_products={i[0]:{'name':i[1], 'price':i[2], 'category':i[3], 'stock':i[4]} for i in lifestore_products}
  k=1
  print('\n')
  for i,product in enumerate(tabla[:numberofTop]):
    print('El producto en la prosición: {} con {} {} es \t {}'.format(i+1,product[1],tipo,mylifestore_products[product[0]]['name']))
  print('\n')

recuentos(lifestore_sales,'ventas',5,'max',False)
recuentos(lifestore_searches,'busquedas',10,'max',False)
recuentos(lifestore_sales,'ventas',5,'min',False)
recuentos(lifestore_searches,'busquedas',10,'min',False)

# analisis como sumatoria total que aplicaria si todos tuvieran el mismo numero de aparisiones
recuentos(lifestore_sales,'reseñas',5,'max',True)
recuentos(lifestore_sales,'reseñas',5,'min',True)

"""Analisis de fechas, creamos el diccionario y lo ordenamos """

mylifestore_sales={}
for i in lifestore_sales:
  try:
    mylifestore_sales[datetime.datetime.strptime(i[3],'%d/%m/%Y')]+=1
```

```
  except:
    mylifestore_sales[datetime.datetime.strptime(i[3],'%d/%m/%Y')]=1
tabla=ordenar(mylifestore_sales,'max')

"""Mostramos los 5 dias con más ventas"""

for i,product in enumerate(tabla[:5]):
  print('El producto en la posición: {} con {} compras es \t {}'.format(i+1,product[1],product[0]))

import plotly.express as px
import pandas as pd

"""Visualización de las ventas a lo largo del tiempo"""

tabla2=pd.DataFrame(tabla)
fig=px.area(tabla2,x=0,y=1,line_shape="spline")
fig.show()

"""Calculo de ventas totales y promedio mensual, anual y el total de la tabla. """

eachWeek={}
eachWeekday={}
eachMonth={}
eachYear={}

for k,v in mylifestore_sales.items():
  try:
    eachWeek[k.strftime("%V")]+=1
    eachWeekday[k.weekday()]+=1
    eachMonth[k.month]+=1
    eachYear[k.year]+=1
  except:
    eachWeek[k.strftime("%V")]=1
    eachWeekday[k.weekday()]=1
    eachMonth[k.month]=1
```

```
  eachYear[k.year]=1


"""En el siguiente diccionario podemos ver que la semana 5,14 y 7 son en las cuales se ha vendido mayor cantidad.


*Nota: Los diccionarios no tienen orden por lo que se imprimen en desorden, no vi restricción en ello pero decidi mostrarlo al final para mayor claridad*

"""


for k in eachWeek.keys():

  print(f'La semana {k} hubo un total de {eachWeek[k]} ventas con un promedio productos/día de \t de {eachWeek[k]/7}')


eachWeek


"""Podemos ver que los dias jueves son los dias de mayor ventas"""


eachWeekday #0: lunes 1: Martes 2: Miercoles 3: Jueves 4: Viernes 5: Sabado 6: Domingo


"""Ademas de que se vende más en el mes de marzo"""


for k in eachMonth.keys():

  print(f'En el mes {k} hubo un total de {eachMonth[k]} ventas con un promedio productos/día de \t de {eachMonth[k]/30}')


"""Y el año de más ventas fue el 2020"""


for k in eachYear.keys():

  print(f'En el año {k} hubo un total de {eachYear[k]} ventas con un promedio productos/día de \t de {eachYear[k]/365}')
```

SOLUCIÓN

LOGIN

```
Escriba su usuario: Jesus
Escriba su contraseña: 12354
No se encontro registro
Escriba su usuario: User1
Escriba su contraseña: 0a9c68f2
Usuario identificado con el id: 0a9c678a-8828-11ec-b445-0242ac1c0002
```

Variables

Las variables más importantes son

dbLogin : Es el registro de usuarios existentes para logiar

lifestore_sales: un diccionario para análisis de las ventas y reseñas

lifestore_searches: un diccionario para análisis de las búsquedas

tabla: Es la lista de ventas ordenadas

eachWeek: Es el recuento de repeticiones por semana

eachWeekday: Es el recuento de repeticiones por día de la semana

eachMonth: Es el recuento de repeticiones por mes

eachYear: Es el recuento de repeticiones por año

Validaciones

Se hace la validación del login para poder acceder al análisis

Código

https://colab.research.google.com/drive/1ScVAzYfeh4Trmk4JG42ubkhtOfYncVun

Formato de resultado

Ejemplo:

```
Analisis para ventas...
ordenado por maximo
{1: 2, 2: 13, 3: 42, 4: 13, 5: 20, 6: 3, 7: 7, 8: 4, 10: 1, 11: 3, 12: 9, 13: 1, 17: 1, 18: 5, 21: 2, 22: 1, 25: 2, 28: 1, 29: 14, 31: 6, 33: 2, 40: 1, 42: 18, 44: 6, 45: 1, 46: 1, 47: 11


El producto en la prosición: 1 con 50 ventas es         SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm
El producto en la prosición: 2 con 42 ventas es         Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth
El producto en la prosición: 3 con 20 ventas es         Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake)
El producto en la prosición: 4 con 18 ventas es         Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD
El producto en la prosición: 5 con 15 ventas es         SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm


Analisis para busquedas...
ordenado por maximo
{1: 10, 2: 24, 3: 55, 4: 41, 5: 30, 6: 10, 7: 31, 8: 20, 9: 1, 10: 1, 11: 5, 12: 15, 13: 2, 15: 4, 17: 3, 18: 11, 21: 15, 22: 5, 25: 10, 26: 5, 27: 1, 28: 5, 29: 60, 31: 10, 35: 1, 39: 3,


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


Analisis para ventas...
ordenado por minimo
{1: 2, 2: 13, 3: 42, 4: 13, 5: 20, 6: 3, 7: 7, 8: 4, 10: 1, 11: 3, 12: 9, 13: 1, 17: 1, 18: 5, 21: 2, 22: 1, 25: 2, 28: 1, 29: 14, 31: 6, 33: 2, 40: 1, 42: 18, 44: 6, 45: 1, 46: 1, 47: 11


El producto en la prosición: 1 con 1 ventas es   MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0
El producto en la prosición: 2 con 1 ventas es   Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0
El producto en la prosición: 3 con 1 ventas es   Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0
El producto en la prosición: 4 con 1 ventas es   Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC, 4GB 128-bit GDDR5, PCI Express x16 3.0
```

Conclusiones

Contratar mas personas para los días miércoles y adquirir más productos para las ventas del mes de marzo.