PROYECTO 1

INTRODUCCIÓN A PYTHON

EMTECH
Emerging Technologies Institute

LifeStore

Emanuel Flores Martínez

Grupo 4

04/Septiembre/2020

ÍNDICE

INTRODUCCIÓN

OBJETIVO

En el transcurso de estas semanas, se ha aprendido los fundamentos del lenguaje de programación Python. Ahora es tiempo de poner en práctica lo aprendido, para esto, se analizarán los datos de la tienda **LifeStore**, la cuál es una tienda virtual encargada de vender artículos electrónicos. En los últimos meses ha notado una disminución en la venta de sus productos, por lo que, a través del análisis hecho, se logrará encontrar una respuesta a este cambio en las ventas, sugerencias para incrementar los ingresos, y estrategias para el inventario rezagado.

RESÚMEN DE LA SOLUCIÓN EMPLEADA

El sistema empleado se basa en un sistema de inventario, donde un **usuario administrador**, podrá ver todos los productos más vendidos, menos vendidos y rezagados. De igual manera, podrá ver los productos con mejores reseñas, los productos más buscados y un resumen de las ventas mensuales y las ventas anuales, esto permitirá ver como se relacionan los datos y que como afectan en las ventas de cada producto.

En el caso de ser un **usuario normal**, se podrá ver todos los productos vendidos por la tienda, así como su precio, su categoría y su stock.

Código: https://github.com/e-muf/EMTECH-Proyecto-1

DESCRIPCIÓN DEL CÓDIGO PARTE I

ESTRUCTURA DEL CÓDIGO

El código al ser muy grande se puede explicar por diferentes secciones, de igual manera, al haber muchas tareas que realizan la misma función, se intentará explicar cada caso de estudio y mostrar una parte del código que realiza dicha función, posteriormente las líneas que realizan lo mismo no serán explicadas nuevamente. Por otro lado, el código está dividido en tres archivos:

- **Lifestore_file.py:** En este archivo se almacenan los datos de la tienda, se encontrarán tres listas, una con los productos que están en el inventario de la tienda, en otra se almacenan las ventas hechas, y en la última están las búsquedas de los productos.
- **users.py:** Este su un archivo que contiene los usuarios que pueden acceder al sistema, es una lista de listas, donde cada lista contiene el usuario, la contraseña y el rol correspondiente al usuario. El rol se identifica con 1 en el caso de ser administrador y 0 en caso de ser usuario normal.

```
allowed_users = [
    ['Emanuel', 'asdfqwer', 1],
    ['Samantha', 'qwerasdf', 0],
    ['Javier', '1234asdf', 1]
]
```

DESCRIPCIÓN DEL CÓDIGO PARTE II

〉 **PROYECTO_01_FLORES_EMANUEL.py:** En este archivo se encuentra toda la lógica de la programación, en la primera parte se hace las importaciones de los dos archivos anteriores, para obtener los datos que estos almacenan, se podría ver como la consulta a una base de datos.

```
from users import allowed_users
from lifestore_file import lifestore_products, lifestore_sales, lifestore_searches
import os
```

Posterior a esto, se realiza el proceso de validación, lo primero que hacemos es mostrarle un mensaje al usuario dándole la bienvenida, y pidiéndole que ingrese los datos. Se define una variable **attemps,** que hace referencia a los intentos para ingresar un usuario y contraseña correctos, y una variable llamada **user_validated,** que será un valor booleano que en caso de que un usuario ingrese correctamente iniciará una nueva sesión.

```
attemps = 0
user_validated = False
```

Para continuar se hará un ciclo que valide el número de intentos y que el usuario ingrese correctamente, dentro del ciclo se pedirá al usuario que ingrese sus datos para la sesión.

DESCRIPCIÓN DEL CÓDIGO PARTE III

```
# El usuario tendrá tres intentos para identificarse
while not user_validated and attemps < 3:

    # El usuario ingresa el nombre y la constraseña, para poder ingresar
    name_input = input('Nombre de usuario: ')
    password_input = input('Contraseña: ')
```

Las variables **name_input** y **passwd_input** almacenan el nombre y la contraseña ingresado por el usuario, ahora se compara con la lista de los usuarios permitidos del archivo users.py, se almacena un valor booleano que nos dirá si un usuario es administrador.

```
for user, password, role in allowed_users:
        if user == name_input and password == password_input:
            print('[+] Usuario conectado correctamente.\n')
            user_validated = True
            is_admin = role == 1
```

Sí el usuario no ingresó un nombre y contraseña validos se les muestra un mensaje de error, pidiéndole otra vez los datos, y se suma el número de intentos en uno, cuando llegue a tres sale del ciclo.

```
if not user_validated:
        print('[-] Usuario incorrecto, verifique el nombre y la contraseña para ingresar.\n')
        attemps += 1
```

DESCRIPCIÓN DEL CÓDIGO PARTE IV

Sección de Análisis de Datos

La sección de análisis se base en generar varias listas, con información importante relacionada a los datos que se tienen. En primer lugar se hace el análisis de datos por ventas, para esto se crear una lista llena de valores **None,** pero es del tamaño de todos los productos que ofrece lifestore, en esta lista se almacenará el id del producto, un promedio del score obtenido de cada producto, las veces que fue devuelto, la cantidad vendida, y las veces que fue buscado, en cado de que el producto se haya vendido, en otro caso solo se guardará el di del producto y las veces que fue buscado.

```
sales_analysis = [None] * len(lifestore_products)

# Se creará una lista que almacene el mes y el en que fue vendido cada producto
date_sales = []

for product_sale in lifestore_sales:
    product_position = product_sale[1] - 1

    month, year = int(product_sale[3][3:5]), int(product_sale[3][6:])
    date_sales.append([product_sale[1], month, year])

    #
 Sí no está en la lista, agrego los datos que me interesan, y agrego la cantidad d
e veces que se vendió
    if sales_analysis[product_position] == None:
        sales_analysis[product_position] = product_sale[1:3] + product_sale[4:]
        sales_analysis[product_position].append(1)

    #
 En caso de que ya este en la lista, se revisa si tuvo alguna devolución, en caso
 de que sí
    #
 agrego uno a la cantidad de veces regresado, además se suma el score que le diero
n al
    # producto y se agrega uno a la cantidad de veces que se vendió
    else:
        sales_analysis[product_position][-2] += product_sale[-1] # Refunds
        sales_analysis[product_position][1] += product_sale[2]   # Score
        sales_analysis[product_position][-1] += 1                # Quantity Sold
```

DESCRIPCIÓN DEL CÓDIGO PARTE V

Además, se extrae el mes y el año y se agrega a una lista llamada **date_sales,** que será usada para hacer el análisis de las ventas mensuales y anuales.

En el siguiente ciclo se crea una nueva lista, llamada **selled_products,** en esta lista únicamente se guardaran los productos que si fueron vendidos, además de que se hace el promedio de la puntuación de cada producto. En este ciclo es donde se agregan los productos que no fueron vendidos a la lista donde se están estudiando todos los productos, y además se agrega un nuevo elemento en donde se llevará el conteo de las búsquedas de cada producto.

```
selled_products = []

for i, product in enumerate(sales_analysis):
    if product != None:
        product[1] /= (product[-1] + product[-2]) #
 Mean Score (Score / (quantity_sold + refunds))
        selled_products.append(product)
    else:
        sales_analysis[i] = [i + 1, 0]

    # Este cero que se agrega al final será el contador de las busquedas
    sales_analysis[i].append(0)

def take_last_second_thrid(elem):
    return (elem[-2], elem[-3])

selled_products.sort(key = take_last_second_thrid, reverse = True)
```

Además se ve una función que regresará solamente los elementos antepenúltimo y penúltimo de una lista dada, se hace uso de esta función para ordenar de mayor a menor los productos vendidos.

DESCRIPCIÓN DEL CÓDIGO PARTE VI

Posteriormente, mediante la lista de los productos que fueron buscados, se hará haciendo el conteo, en la lista donde se está haciendo todo el análisis, eso se hará con un ciclo for.

```
for search in lifestore_searches:
    position = search[1] - 1
    sales_analysis[position][-1] += 1
```

Para ordenar las listas se necesitan diferentes posiciones, por lo que se tienen diferentes funciones que hacen eso, pero ya se vio en un caso anterior, lo que está en el código son las funciones que regresan las posiciones.

```
def take_second(elem):
    return elem[1]

def take_third(elem):
    return elem[2]

def take_four(elem):
    return elem[3]

def take_last(elem):
    return elem[-1]
```

DESCRIPCIÓN DEL CÓDIGO PARTE VII

S se estudiarán por fechas los datos separados en pasos anteriores, para este caso se guardarán en una lista adicional llamada **date_total_sales** y **year_total_sales**. Empezaremos ordenando la lista donde se almacena el mes y el año, primero por el mes, para que quede ordenado de menor a mayor mes, y posteriormente por año para que los años queden de menor a mayor. Una vez ordenadas se definen unas variables las cuales almacenan el mes y el año actual sobre el que se está iterando, y otras variables que llevaran la suma de ingresos mensuales y los productos vendidos mensualmente.

```
date_sales.sort(key = take_second)
date_sales.sort(key = take_third)

date_total_sales = []
actual_month = date_sales[0][1]
actual_year = date_sales[0][2]
total_amount = 0
product_count = 0
```

Ahora se realizará un ciclo sobre las fechas almacenadas, si el mes actual es igual al mes que estamos analizando se suma el precio del producto, sobre el que se itera, además se suma uno al producto, en caso contrario significa que estamos cambiando de mes, por lo que se reinician los contadores del total de ingreso mensual y el de productos mensual, y los datos generados anteriormente se agregan a la lista **date_total_sales**.

DESCRIPCIÓN DEL CÓDIGO PARTE VIII

```
for id_product, month, year in date_sales:
    if actual_month == month:
        product_count += 1
        total_amount += lifestore_products[id_product - 1][2]
    else:
        date_total_sales.append([
actual_month, actual_year, product_count, total_amount])
        actual_year = year
        actual_month = month
        total_amount = 0
        product_count = 0
```

Para hacer el análisis por año se hace un proceso similar al anterior, solo que ahora se sumaran los ingresos de cada mes por año, cuando se cambia de año, la variable que hace la suma se regresa a cero.

```
year_total_sales = []
year_amount = 0
actual_year = date_sales[0][2]
for date_sale in date_total_sales:
    if actual_year == date_sale[1]:
        year_amount += date_sale[-1]
    else:
        year_total_sales.append([actual_year, year_amount])
        actual_year = date_sale[1]
        year_amount = 0
year_total_sales.append([actual_year, year_amount])
```

Para finalizar, decidí utilizar un análisis de categorías lo creí conveniente por que así podemos saber que productos son los que menos se están vendiendo, para esto, se creará una nueva lista llamada **category_analysis**.

DESCRIPCIÓN DEL CÓDIGO PARTE IX

Se almacenará en una lista llamada **categories,** todas las categorías que tenga la tienda nos ayudarán para saber a que categoría hacemos referencia, después agregaremos en la lista **category_analysis**, el nombre de la categoría, el total de ventas que tiene cada categoría y el número diferente de productos por categoría. Además, se agregará una relación entre estos dos últimos datos que nos será útil para saber cual es el producto que genera menos ingresos.

```
category_analysis = []
categories = []

for product in sorted(sales_analysis, key = take_second_last, reverse=True):
    id_product = product[0] - 1
    if lifestore_products[id_product][3] not in categories:
        categories.append(lifestore_products[id_product][3])
        category_analysis.append([lifestore_products[id_product][3], lifestore_products[id_product][2] * product[-2], 1])
    else:
        category_position = categories.index(lifestore_products[id_product][3])
        category_analysis[category_position][1] += lifestore_products[id_product][2] * product[-2]
        category_analysis[category_position][2] += 1

for category_sale in category_analysis:
    category_sale.append( category_sale[1] / (category_sale[2] * 100) )

category_analysis.sort(key=take_four, reverse=True)
```

Por último, se crea la interfaz de usuario para el administrador y para el usuario normal, este se creará por medio de una ciclo while, que se ejecutará mientras el usuario este identificado con la variable **user_validated** y sabrá en que ciclo while entrar sabiendo si es administrador o no. El usuario verá un menú, que será el menú principal donde podrá seleccionar mediante un número la opción que él quiera consultar.

DESCRIPCIÓN DEL CÓDIGO PARTE X

```
while user_validated and is_admin:
    os.system('clear')
    print('**** LifeStore Panel (Administrador) [{}] ****\n'.format(name_input))
    option = input('''Seleccione el número con la opción deseada:
1. Mostrar productos por ventas
2. Mostrar productos por cantidad de búsquedas
3. Mostrar productos por reseñas
4. Mostrar ingresos por fecha
5. Mostrar resumen por categorias
6. Cerrar Sesión
>> ''')
```

Como se puede ver en la imagen, podrá ver los productos ordenado por **cantidad de ventas**, los productos por **cantidad de búsquedas**, los productos por **reseñas**, los ingresos **por fecha (mes y año)** y para finalizar un **resumen por categorías**. Igual le dará la opción de **cerrar sesión**, que será como podrá salir del programa. Hay opciones, que tienen opciones dentro, pero funcionan de la misma manera del menú.

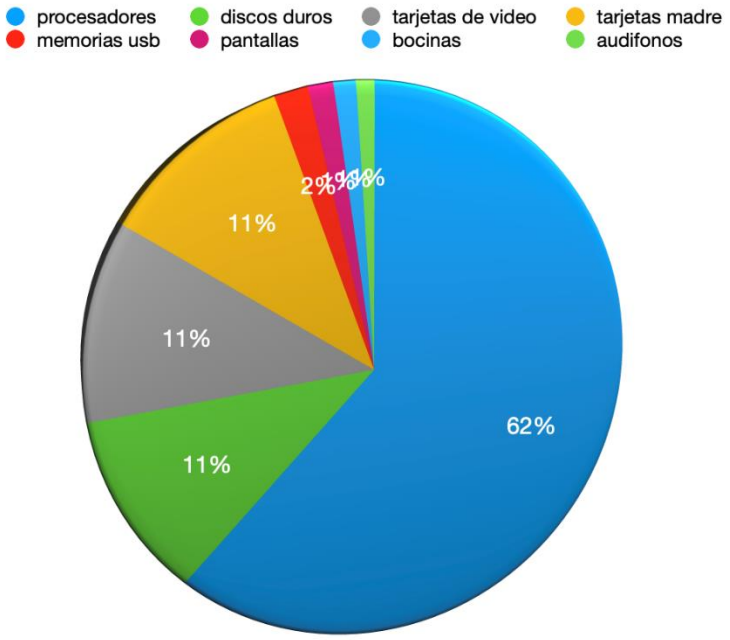
Para mostrar los productos simplemente se itera sobre las listas creadas. Dejando espacios para que se pueda en forma de tablas y sea más fácil de leer para el usuario. Por último, se le pedirá al usuario que presione Enter para volver al menú principal donde podrá seleccionar otra opción.

```
elif sale_filter == '3':
            print('PRODUCTO', ' ' * 110, '| STOCK \t| PRECIO')
            print('-' * 145)
            for product in sales_analysis:
                if product[-2] == 0:
                    id_product = lifestore_products[product[0] - 1]
                    print(id_product[1], ' ' * (119 - len(id_product[1])) + '|', id_product[-1], '\t\t|', id_product[-3])
        else:
            continue

        input('\nPresiona ENTER para regresar al menú principal.\n')
```

SOLUCIÓN AL PROBLEMA PARTE I

Con la información analizada anteriormente, podemos ver primero cuál es la categoría de productos que más se vendo y que menos se vende para eso, se muestra la siguiente gráfica.



Se observa que el producto que genera más ganancias son los **procesadores**, que son el 62% del ingreso de la tienda, mientras que los productos con menores ingresos son los **audífonos**, **bocinas** y **pantallas**, con apenas 1% de ingresos.

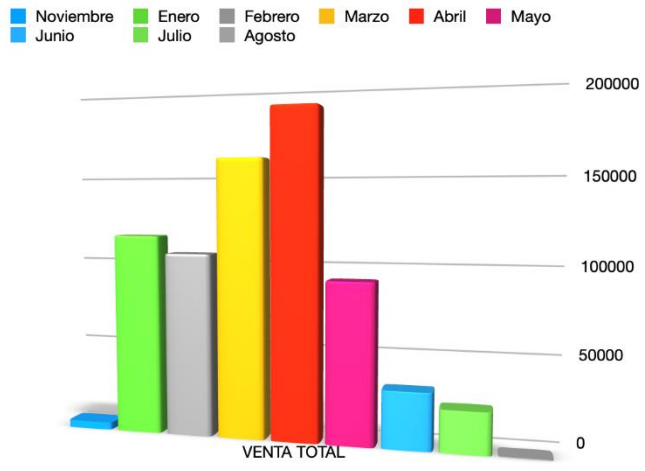
Esto es importante considerar esto, porque al ver el listado de productos vendidos, la mayoría son procesadores, discos duros y tarjetas de video, mientras que los productos rezagados son las pantallas, memorias/usb, bocinas, y audífonos. Entonces como recomendación a la empresa es que se concentre en vender esos artículos ya que al revisar el stock hay algunos productos que son muy vendidos, pero no tienen nada en stock como es el caso de la tarjeta madre ASRock Micro ATX B450M-PLUS. De igual manera al ser los productos más vendidos son los productos que tienen reseñas, donde los mejores valorados son los procesadores, y los menos valorados las tarjetas madre.

SOLUCIÓN AL PROBLEMA PARTE II

Por lo que se tendría que revisar con el proveedor de estas la calidad de las tarjetas madre recibidas, o cambiar de proveedor por uno que ofrezca mejores y poder aumentar o mantener las ventas de este sin que estas sean devueltas.

Por otro lado, los productos más buscados tienen una correlación con los productos más vendidos, puede ser porque es lo que más les interesa a la gente o porque los productos la tienda no hace suficiente promoción de los productos como los audífonos o las memorias, una sugerencia para aumentar la venta de estos podría ser que hicieran ofertas con estos productos, o hacerlos más visibles dentro de la página.

Para finalizar, se muestra la siguiente gráfica:



En esta se observa, que las ventas aumentaron en enero, marzo y abril, y disminuyeron demasiado de marzo a agosto, esto podría ser porque son meses donde no salen novedades en los productos, por lo que sería recomendable tener gran inventario durante los primeros meses del año, y de mayo en adelante vender y promocionar los productos que se quedaron durante esos meses, es importante analizar durante algunos años para saber con certeza a que se debe esto.

CONCLUSIONES

Del análisis de datos

LifeStore es una empresa que apenas está creciendo, es importante que se preocupe por los productos que más vende que son los procesadores, discos duros, tarjetas madre y tarjetas de video, ya que la mayoría de comentarios negativos son de tarjetas madre, por lo que tiene que cuidar mejor la calidad de estos, además balancear un poco más su inventario, ya que tiene mucho inventario de algunos productos y muy poco de otros que se venden muy bien. Además al ver los precios la gente muchas veces se va por lo más barato por lo que es conveniente tener productos que puedan comprar los clientes, y que sean de un proveedor confiable.

Del proyecto

El proyecto en general fue entretenido, yo sabía programar en Python, y se em dificulto volver un poco a la programación estructurada, ya que hay muchas cosas que se repetían y podían hacerse más eficientes por medio de funciones. Por otro lado no pude olvidar lo aprendido y separe las cosas de acuerdo a la función como fue el caso de la sección de login, la sección de análisis de datos, y la sección de la interfaz de usuario.

Aprendí mucho con este proyecto, y es importante observar como interactúan los datos entre sí para obtener buenos resultados.