EMTECH
Proyecto 1

ALUMNO: Sandoval palma miguel angel

GRUPO: 3 Data Science

Contenido

Objetivo

Poner en práctica las bases de programación en Python para análisis y clasificación de datos mediante la creación de programas de entrada de usuario y validaciones, uso y definición de variables y listas, operadores lógicos y condicionales para la clasificación de información.

Consigna

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

1) Productos más vendidos y productos rezagados a partir del análisis de las categorías con menores ventas y categorías con menores búsquedas.

2) Productos por reseña en el servicio a partir del análisis de categorías con mayores ventas y categorías con mayores búsquedas.

3) Sugerir una estrategia de productos a retirar del mercado así como sugerencia de cómo reducir la acumulación de inventario considerando los datos de ingresos y ventas mensuales.

- Productos más vendidos y productos rezagados: Generar un listado de los 50 productos con mayores ventas y uno con los 100 productos con mayores búsquedas.
- Por categoría, generar un listado con los 50 productos con menores ventas y uno con los 100 productos con menores búsquedas.
- Productos por reseña en el servicio Mostrar dos listados de 20 productos cada una, un listado para productos con las mejores reseñas y otro para las peores, considerando los productos con devolución.
- Total, de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año

Main.py

```
import operations as op

from getpass import getpass

# Variables

usuario = 'Mick'

password = 'acceso12'

def login():

#funcion de login

    usuario_valido = input('Ingrese su usuario: ')

    password_valido = getpass('Ingrese su password: ')

    if (usuario_valido == usuario) & (password_valido == password):

        print('Hola: '+ usuario_valido)

        return True

    else:

        return False

def analysis(option):

#funcion switch en python

    switcher = {

        1: op.option_1,

        2: op.option_2,

        3: op.option_3

    }

    func = switcher.get(option, "Opcion no valida")

    return func()

logged = login()

finish = 'n'

while finish != ('y' or 'Y'):

    if not logged:

        print('Ingreso incorrecto')

        logged = login()

    else:

        print('''

        Opciones disponibles:
```

```
        1.- Top productos mas vendidos y los mas buscados
        2.- Productos por reseña en el servicio.
        3.- Total de ingresos y ventas promedio mensuales,total anual y meses con más ventas al año
    ''')
    analysis(int(input('Ingresa la opcion a realizar dentro de 1-4: ')))
    finish = input('Exit? (y/n): ')
```

Operations.py

```
from lifestore_file import lifestore_products, lifestore_sales, lifestore_searches
from quick_sort import quick_sort
import sys
numero_produtos = len(lifestore_products)
year_dictionary = {
    1: 'January',
    2: 'February',
    3: 'March',
    4: 'April',
    5: 'May',
    6: 'June',
    7: 'July',
    8: 'August',
    9: 'September',
    10: 'October',
    11: 'November',
    12: 'December'
}
def ventas_por_producto(reembolso=False):
    ventar_por_producto_lista = [[i + 1, 0] for i in range(numero_produtos)]
    for sale in lifestore_sales:
        if reembolso:
            if sale[4] == 0:
```

```
            ventar_por_producto_lista[sale[1] - 1][1] += 1
        else:
            ventar_por_producto_lista[sale[1] - 1][1] += 1
    return ventar_por_producto_lista
def calcular_busquedas_por_producto():
    busqueda_por_lista_productos = [[i + 1, 0] for i in range(numero_produtos)]
    for search in lifestore_searches:
        busqueda_por_lista_productos[search[1] - 1][1] += 1
    return busqueda_por_lista_productos
def ordenar_productos_por_ventas():
    ventar_por_producto_lista = ventas_por_producto()
    quick_sort(ventar_por_producto_lista, lambda x: x[1])
    return ventar_por_producto_lista
def organizar_productos_por_busquedas():
    busqueda_por_lista_productos = calcular_busquedas_por_producto()
    quick_sort(busqueda_por_lista_productos, lambda x: x[1])
    return busqueda_por_lista_productos
def calculadora_por_categorias(array):
    for element in array:
        product_id = element[0]
        categoria_producto = lifestore_products[product_id - 1][3]
        element.append(categoria_producto)
    por_categoria_lista = []
    categoria_anterior = array[0][2]
    acumulacion_categoria = 0
    productos_por_categoria = 0
    array.append([0, 0, 'dummy'])
    for product in array:
        category = product[2]
        if category != categoria_anterior:
            por_categoria_lista.append([categoria_anterior, acumulacion_categoria, productos_por_categoria])
            acumulacion_categoria = 0
            productos_por_categoria = 0
```

```
        categoria_anterior = category
        acumulacion_categoria += product[1]
        productos_por_categoria += 1
    del array[-1]
    return por_categoria_lista


def ordenar_categorias_por_ventas():
    ventas_por_producto = ventas_por_producto()
    ventas_por_lista_categoria = calculadora_por_categorias(ventas_por_producto)
    quick_sort(ventas_por_lista_categoria, lambda x: x[1])
    return ventas_por_lista_categoria
def ordenar_categorias_por_busquedas():
    busquedas_por_producto = calcular_busquedas_por_producto()
    busqueda_por_categoria_lista = calculadora_por_categorias(busquedas_por_producto)
    quick_sort(busqueda_por_categoria_lista, lambda x: x[1])
    return busqueda_por_categoria_lista
def mejor_y_peor_productos_reviews():
    puntuacion_promedio_por_producto = []
    suma_promedio_por_producto = [[i + 1, 0] for i in range(numero_produtos)]
    ventar_por_producto_lista = ventas_por_producto()
    for sale in lifestore_sales:
        product_id = sale[1]
        suma_promedio_por_producto[product_id - 1][1] += sale[2]
    for puntaje_acumulativo in suma_promedio_por_producto:
        product_id = puntaje_acumulativo[0]
        sales = ventar_por_producto_lista[product_id - 1][1]
        # Only count them if they have been sold
        if sales > 0:
            puntaje_promedio = puntaje_acumulativo[1] / sales
            puntuacion_promedio_por_producto.append([product_id, puntaje_promedio])
    # Sort the list from minimum to maximum
    quick_sort(puntuacion_promedio_por_producto, lambda x: x[1])
```

```
    return puntuacion_promedio_por_producto
def calculo_total_ingresos():
    # declare total_revenue equal to zero
    total_revenue = 0
    ventas_por_producto = ventas_por_producto(reembolso=True)
    for sales in ventas_por_producto:
        product_id = sales[0]
        total_revenue += lifestore_products[product_id - 1][2] * sales[1]
    return total_revenue
def calculo_mensual_ventas():
    # Create a list of lists with elements [month, sales_per_month]
    monthly_sales_list = [[month, 0] for month in range(1, 13)]
    total_concreted_sales = 0
    for sale in lifestore_sales:
        month = int(sale[3].split('/')[1])
        if sale[4] == 0:
            monthly_sales_list[month - 1][1] += 1
            total_concreted_sales += 1
    return monthly_sales_list, total_concreted_sales
def option_1():
    show_worst = 60;
    ventar_por_producto_lista = ordenar_productos_por_ventas()

    print('Los 10 productos mas vendidos son: ')
    for index in range(1, 11):
        print(
            f'{index}.- {ventar_por_producto_lista[-index][1]} ventas para'
            f' {lifestore_products[ventar_por_producto_lista[-index][0] - 1][1]}')
    print('\n')
    busqueda_por_lista_productos = organizar_productos_por_busquedas()
    print('El top 10 productos buscados: ')
    for index in range(1, 11):
        print(
```

```
            f'{index}.- {busqueda_por_lista_productos[-index][1]} busquedas para'
            f' {lifestore_products[busqueda_por_lista_productos[-index][0] - 1][1]}')
    print('\n')
    sales_per_categories_list = ordenar_categorias_por_ventas()
    sales_per_categories_list.reverse()
    print(f'ventas por categoria son:')
    for index in range(len(sales_per_categories_list)):
        print(
            f'{index + 1}.- {sales_per_categories_list[index][1]} para {sales_per_categories_list[index][0]} with '
            f'{sales_per_categories_list[index][2]} productos'
        )
    print('\n')

    searches_per_categories_list = ordenar_categorias_por_busquedas()
    searches_per_categories_list.reverse()
    print(f'Busquedas por categoria son:')
    for index in range(len(searches_per_categories_list)):
        print(
            f'{index + 1}.- {searches_per_categories_list[index][1]} para {searches_per_categories_list[index][0]} '
            f'con {searches_per_categories_list[index][2]} productos'
        )
def option_2():
    puntuacion_promedio_por_producto = mejor_y_peor_productos_reviews()
    print('El top 10 de los productos puntuados es: ')
    for index in range(1, 21):
        print(
            f'{index}.- {puntuacion_promedio_por_producto[-index][1]:.2f}/5 estrellas para'
            f' {lifestore_products[puntuacion_promedio_por_producto[-index][0] - 1][1]}')
    print('\n')
    print('los 10 peores rateados son: ')
    for index in range(10):
        print(
            f'{index + 1}.- {puntuacion_promedio_por_producto[index][1]:.2f}/5 estrellas para'
```

```
        f' {lifestore_products[puntuacion_promedio_por_producto[index][0] - 1][1]}')
def option_3():
    total_revenue = calculo_total_ingresos()
    monthly_sales_list, total_concreted_sales = calculo_mensual_ventas()
    print(f'los reembolsos mensuales son: {total_revenue}')
    print(f'Ingresos promedio: {total_revenue / 8:.2f}')
    print(f'las ventas del 2021: {total_concreted_sales} \n')
    print(f'Las ventas mensuales son:')
    for index, month in enumerate(monthly_sales_list):
        print(f'{index + 1} .- {month[1]}  ventas concretadas en {year_dictionary[month[0]]}')
```