FUNDAMENTOS DE PROGRAMACIÓN CON PYTHON

PROYECTO FINAL

ALINA HELENA SÁNCHEZ GALLARDO

ANÁLISIS DE DATOS PARA LA TIENDA VIRTUAL LIFESTORE

https://github.com/AlinaGallardo/EmTech_Final_Project_1.git

EMTECH
Emerging Technologies Institute

Índice

Introducción

La cuarta revolución industrial ha causado un hito en la historia moderna de la sociedad, ya que ha marcado una convergencia de tecnologías y se anticipa que en pocos años se alcance un significativo cambio en el mundo como lo conocemos. La mayoría de los sistemas han comenzado a ser automatizados y ha existido una creciente demanda de profesionales en las áreas de tecnología, software y manejos de sistemas automatizados.

La revolución digital ha ido acelerándose con el paso del tiempo y lo seguirá haciendo al punto de que, en el futuro, la forma en la que convivimos, trabajamos y vivimos será totalmente distinta, esto sucede a una escala tal que, lo único que podemos hacer es adaptarnos y ser parte del cambio o quedar rezagados.

Debido a que la tecnología y los sistemas de interacción social online se han vuelto parte de la vida cotidiana de todas las personas en el mundo, se ha creado una nueva forma de riqueza, la cual ha sido considerada incluso más valiosa que el oro, los datos. Ya sean personales o los datos privados de compañías y bancos, la cantidad de datos que coexisten en la nube es infinita y crece exponencialmente a cada minuto, la única forma de analizar y depurar dicha información es con el uso de sistemas de software que permitan automatizar su análisis.

Gracias a esta situación nacen tres procesos importantes que poco a poco se abren espacio entre las carreras profesionales no solo de ingenieros y científicos, sino de economistas, administradores, actuarios, matemáticos, etc., la minería de datos, el análisis de *big data* y el análisis de datos han sido importantes aliados en el proceso de encontrar patrones, correlaciones y anomalías en conjuntos inmensos de datos.

Las bases de datos de empresas presentan un reto casi imposible de analizar sin el apoyo de un software apropiado, gracias al análisis de datos, las empresas pueden acceder a información concreta y especializada para tomar decisiones y estrategias de negocio que los lleven a alcanzar metas y objetivos pre estipulados.

Algunas de las otras aplicación del análisis de datos son: la optimización de procesos y eliminación de ineficiencias para ahorrar costes, la toma de decisiones reconociendo oportunidades y la mejora en la satisfacción de los clientes proporcionando datos objetivos acerca de las opiniones que dichos tengan sobre determinado producto.

Los lenguajes de programación mayormente utilizados en ciencia de datos son R, Python, SQL, Java, Scala, MATLAB y Julia, de entre los cuales, los lenguajes de programación con mayor uso son R y Python. Python encabeza la lista de preferencia debido a la sencillez de su sintaxis para el desarrollo de AI, de igual forma, es un lenguaje con un tiempo de desarrollo corto en comparación con Java, C++ o Ruby [Universidad de Alcalá, 2021].

El lenguaje de programación Python ofrece una codificación simple en comparación con otros idiomas y es capaz de ejecutar programas en la menor cantidad de líneas de código, puede procesar tareas largas en un lapso corto de tiempo y no hay limitación para el proceso de datos. De igual forma, Python es un programa de código abierto que se admite en múltiples plataformas y entornos como Windows, Linux y iOS.

Descripción del problema

A continuación se mostrará el análisis de un listado de datos proporcionados por la tienda online *Lifestore*. Dicho listado proporciona información sobre la cantidad de veces que se buscan determinados productos, la cantidad de veces que se han vendido por mes, así como la cantidad de devoluciones por clientes y reseñas marcadas del 0 al 5.

La petición de la tienda fue la depuración y análisis de dichos listados debido a una acumulación de inventario, reducción en la búsqueda de productos y disminución de las ventas en el último trimestre.

Login

Antes de proporcionar el análisis de datos a quien quiera acceder al programa se deben de pedir las credenciales para otorgar o negar su acceso. Para esto, se realizó el siguiente código:

```
usuarioAccedio = False
intentos = 0

mensaje_bienvenida = 'Bienvenid@ al sistema de datos de Lifestore\nA contnuación te
pediremos que ingreses tu usuario y contraseña'
print(mensaje_bienvenida)

# Recibo constantemente sus intentos
while not usuarioAccedio:
    # Primero ingresa Credenciales
    usuario = input('Usuario: ')
```

```
contras = input('Contraseña: ')
intentos += 1
# Reviso si el par coincide
if usuario == 'Alina' and contras == '981701ahsg':
    usuarioAccedio = True
    print(f'Hola {usuario}, bienvenido de nuevo al sistema')
else:
    # print('Tienes', 3 - intentos, 'intentos restantes')
    print(f'Tienes {3 - intentos} intentos restantes')
    if usuario == 'Alina':
        print(f'{usuario} Te equivocaste en la contraseña')
    else:
        print(f'El usuario: "{usuario}" no esta registrado')

if intentos == 3:
    exit()
```

Del cual el usuario y contraseña de acceso fueron:
Usuario: Alina
Contraseña: 981701ahsg

En la terminal se observa lo siguiente:

```
Bienvenid@ al sistema de datos de Lifestore
A contnuación te pediremos que ingreses tu usuario y contraseña
Usuario: Alina
Contraseña: 981701ahsg
Hola Alina, bienvenido de nuevo al sistema
```

De igual forma, el código está escrito de manera que si el usuario proporciona de manera errónea sus credenciales, el programa no otorgará los datos solicitados.

```
Bienvenid@ al sistema de datos de Lifestore
A contnuación te pediremos que ingreses tu usuario y contraseña
Usuario: Alina
Contraseña: fas
Tienes 2 intentos restantes
Alina Te equivocaste en la contraseña
Usuario: Alina
Contraseña: fasdfa
Tienes 1 intentos restantes
Alina Te equivocaste en la contraseña
Usuario: Alina
Contraseña: 3841
Tienes 0 intentos restantes
Alina Te equivocaste en la contraseña
```

De igual forma, si el usuario ingresado no es reconocido por el sistema se arrojará el mensaje “El usuario *** no está registrado”:

```
Bienvenid@ al sistema de datos de Lifestore
A contnuación te pediremos que ingreses tu usuario y contraseña
Usuario: fads
Contraseña: ag4
Tienes 2 intentos restantes
El usuario: "fads" no esta registrado
Usuario: jfa98
Contraseña: f9n48
Tienes 1 intentos restantes
El usuario: "jfa98" no esta registrado
Usuario: afs90
Contraseña: ak98
Tienes 0 intentos restantes
El usuario: "afs90" no esta registrado
```

Primera parte.

Productos más vendidos y productos rezagados.

Cuando las credenciales ingresadas son las correctas, entonces se accede al sistema y lo primero que solicita la tienda es:

¿Cuáles fueron los 5 productos con mayores ventas y los 5 productos con menores ventas?

Para esto lo principal fue saber la cantidad de veces que se vendía cada producto:

ID del producto	Cantidad de veces que se vendió
10	1
13	1
22	1
28	1

40	1
50	1
60	1
66	1
67	1
84	1
89	1
1	2
21	2
25	2
33	2
52	2
74	2
85	2
6	3
11	3
31	3
49	3
51	3
8	4
18	5
44	6
7	7
12	9
48	9
47	11
2	12
4	13
29	13
57	15
42	18
5	20
3	42
54	49

Posteriormente, para saber cuáles fueron los artículos con mayores y menores ventas, se le pidió a Python que ordenara las ventas de mayor a menor con el siguiente código:

```
#Ordenado por llave

ordenado_por_llave = {}

for llave in sorted(frecuencia):
```

```
    ordenado_por_llave[llave] = frecuencia[llave]


# Ordenado por valor
# primero convertimos el diccionario a una lista
ordenado_por_valor = [[key, value] for key, value in frecuencia.items()]
print("\t")
print("Lista del ID de los artículos vendidos junto con el número de veces que se vendieron es:
")
print("\t")
print(ordenado_por_valor)
ordenado_por_valor = [[value, key] for key, value in frecuencia.items()]
# Ordenamos de mayor a menor
ordenado_por_valor = sorted(ordenado_por_valor, reverse=True)
ordenado_por_valor = [[llave, valor] for valor, llave in ordenado_por_valor]
print("\t")
print("Orden de las ventas por artículo de mayor a menor en el orden [ID_producto,Cantidad
de venta]: ")
print(ordenado_por_valor)
```

Gracias al cual ahora sabemos que:

5 productos con mayores ventas	5 productos con menores ventas
SSD Kingston A400, 120GB, SATA III, 2.5	MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0
Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth	'Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0'
Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generacion - Coffee Lake)'	'Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC, 4GB 128-bit GDDR5, PCI Express x16 3.0
'Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD'	Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti, 6GB 192-bit GDDR6, PCI Express x16 3.0
SSD Adata Ultimate SU800, 256GB, SATA III	Tarjeta Madre Gigabyte XL-ATX TRX40 Designare, S-sTRX4, AMD TRX40, 256GB DDR4 para AMD'

Después de esto, la tienda nos pidió determinar:

¿Cuáles fueron los 10 productos con mayores búsquedas y los 10 productos con menores búsquedas?

El procedimiento fue muy similar, primero se determinaron las búsquedas que tuvieron los artículos por ID de producto:

ID del producto	Cantidad de búsquedas
9	1
10	1
27	1
35	1
45	1
59	1
70	1
80	1
93	1
13	2
56	2
76	2
91	2
17	3
39	3
95	3
15	4
46	4
63	4
73	4
11	5
22	5
26	5
28	5
52	5
74	6
96	6
50	7
89	7
1	10
6	10
25	10
31	10
40	10

49	10
84	10
18	11
51	11
12	15
21	15
66	15
8	20
42	23
2	24
44	25
48	27
5	30
47	30
7	31
67	32
85	35
4	41
3	55
29	60
57	107
54	263

10 productos con mayores busquedas	**10 productos con menores busquedas**
SSD Kingston A400, 120GB, SATA III, 2.5", 7mm	Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generacion - Coffee Lake)
SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'	MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0
Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD	Tarjeta de Video VisionTek AMD Radeon HD5450, 2GB GDDR3, PCI Express x16
Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth	'Tarjeta Madre Gigabyte micro ATX Z390 M GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel
Procesador AMD Ryzen 3 3200G con Graficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire	Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel
Logitech Audifonos Gamer G635 7.1, Alambrico, 1.5 Metros, 3.5mm, Negro/Azul	SSD Samsung 860 EVO, 1TB, SATA III, M.2

TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro	Samsung Smart TV LED 43, Full HD, Widescreen, Negro
Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generacion Coffee Lake)	Ghia Bocina Portatil BX800, Bluetooth, Inalambrico, 2.1 Canales, 31W, USB, Negro
SSD XPG SX8200 Pro, 256GB, PCI Express, M.2	Ginga Audifonos con Microfono GI18ADJ01BT-RO, Bluetooth, Alambrico/Inalambrico, 3.5mm, Rojo
Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generacion - Coffee Lake)	Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0

Segunda parte

Reseñas

La segunda parte del proyecto se basa en considerar las reseñas de los productos para saber cuáles fueron los mejores y peores calificados por los clientes con el objetivo de determinar la satisfacción del cliente conforme al producto.

Lo primero a hacer en este caso era depurar todos los productos que tuvieron reseñas, ya que hubo algunos que no recibieron calificación por parte de los clientes, para eso utilizamos el siguiente código:

```
prod_reviews = {}
for sale in lifestore_sales:
    # prod y review de venta
    prod_id = sale[1]
    review = sale[2]
    # categorización por id
    if prod_id not in prod_reviews.keys():
        prod_reviews[prod_id] = []
    prod_reviews[prod_id].append(review)
print("Product reviews")
print(prod_reviews)
```

```
A continuación se muestra el ID del producto junto con las reseñas que tuvieron:

[ID, Reiew]
1 [5, 5]
2 [5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 3]
3 [5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 4, 5, 5, 5, 5, 5]
4 [4, 4, 5, 4, 5, 5, 5, 4, 5, 3, 5, 4, 5]
5 [4, 4, 4, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 4, 5, 5, 5]
6 [5, 5, 5]
7 [5, 5, 5, 5, 5, 5, 5]
8 [5, 5, 5, 5]
10 [4]
11 [5, 5, 5]
12 [5, 4, 5, 5, 5, 5, 5, 4]
13 [4]
17 [1]
18 [5, 4, 5, 4, 4]
21 [5, 5]
22 [5]
25 [5, 5]
28 [5]
29 [4, 5, 4, 4, 5, 5, 5, 5, 5, 5, 5, 4, 1, 1]
31 [1, 1, 1, 4, 3, 1]
33 [5, 4]
40 [5]
42 [5, 5, 4, 4, 5, 5, 5, 4, 5, 5, 4, 4, 5, 5, 5, 4, 4, 4]
44 [5, 5, 5, 4, 4, 5]
45 [1]
46 [2]
47 [4, 5, 5, 4, 5, 4, 5, 5, 3, 5, 5]
48 [4, 3, 5, 5, 5, 5, 5, 5, 5]
49 [5, 5, 5]
50 [5]
51 [5, 4, 5]
52 [5, 5]
54 [4, 5, 5, 5, 5, 5, 4, 5, 5, 5, 4, 5, 5, 4, 4, 5, 4, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 2, 5, 5, 5, 4, 5, 5, 5, 4, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 4]
57 [5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 5, 4, 5, 5]
60 [5]
66 [5]
67 [5]
74 [4, 5]
84 [5]
85 [5, 5]
89 [3]
94 [4]
```

Posteriormente, se escribió código para enlsitar los productos seguido de las reseñas que tuvieron:

```
print("A continuación se muestra el ID del producto junto con las reseñas que tuvieron: ")
print("[ID, Reiew]")


id_rev_prom = {}
for id, reviews in prod_reviews.items():
    print(id, reviews)
    rev_prom = sum(reviews) / len(reviews)
    rev_prom = int(rev_prom*100)/100
    id_rev_prom[id] = [rev_prom, len(reviews)]
```

El resultado fue el siguiente:

Hasta aquí tenemos cuál fue la calificación que cada cliente le otorgó al producto, sin embargo, necesitamos saber el promedio de las reseñas así como la cantidad de veces que se opinó sobre dicho producto, por lo que se generó el siguiente código:

```
id_rev_prom = {}
for id, reviews in prod_reviews.items():
    print(id, reviews)
    rev_prom = sum(reviews) / len(reviews)
    rev_prom = int(rev_prom*100)/100
    id_rev_prom[id] = [rev_prom, len(reviews)]

print("ID, Promedio de Reviews")

dicc_en_list = []
for id, lista in id_rev_prom.items():
    print(id, rev_prom)
    rev_prom = lista[0]
    cant = lista[1]
    sub = [id, rev_prom, cant]
    dicc_en_list.append(sub)

def seg_elemnto(sub):
    return sub[1]

print("\n\t")
print("A continuación se muestra el ID de los productos, seguido del promedio de sus reseñas
y el número de reseñas que obtuvieron")
print("Ordenados de mejores a peores reseñas por producto")
print("\n\t")
print("[ID_producto, Promedio de reseñas, Número de reseñas]")
print("\n\t")

dicc_en_list = sorted(dicc_en_list, key=seg_elemnto, reverse=True)

for sublista in dicc_en_list:
```

```
print(sublista)
```

A lo cual, el resultado fue:

```
A continuación se muestra el ID de los productos, seguido del promedio de sus reseñas y el número de reseñas que obtuvieron
Ordenados de mejores a peores reseñas por producto


[ID_producto, Promedio de reseñas, Número de reseñas]


[1, 5.0, 2]
[6, 5.0, 3]
[7, 5.0, 7]
[8, 5.0, 4]
[11, 5.0, 3]
[21, 5.0, 2]
[22, 5.0, 1]
[25, 5.0, 2]
[28, 5.0, 1]
[40, 5.0, 1]
[49, 5.0, 3]
[50, 5.0, 1]
[52, 5.0, 2]
[60, 5.0, 1]
[66, 5.0, 1]
[67, 5.0, 1]
[84, 5.0, 1]
[85, 5.0, 2]
[57, 4.86, 15]
[3, 4.8, 42]
[12, 4.77, 9]
[54, 4.72, 50]
[5, 4.7, 20]
[44, 4.66, 6]
[48, 4.66, 9]
[51, 4.66, 3]
[42, 4.55, 18]
[47, 4.54, 11]
[33, 4.5, 2]
[74, 4.5, 2]
[4, 4.46, 13]
[18, 4.4, 5]
[2, 4.23, 13]
[29, 4.14, 14]
[10, 4.0, 1]
[13, 4.0, 1]
[94, 4.0, 1]
[89, 3.0, 1]
[46, 2.0, 1]
[31, 1.83, 6]
[17, 1.0, 1]
[45, 1.0, 1]
```

Lo cual nos deja con los siguientes datos:

5 productos con las mejores reseñas	**5 productos con las peores reseñas**
Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache	Cougar Audifonos Gamer Phontum Essential, Alambrico, 1.9 Metros, 3.5mm
Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generacion Coffee Lake)	Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel
Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generacion Coffee Lake)	Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD
Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiacion - Coffee Lake	Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0
Tarjeta de Video ASUS AMD Radeon RX 570, 4GB 256-bit GDDR5, PCI Express 3.0	Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel

Tercera parte.

Total de ingresos y ventas.

En la última parte del proyecto se solicita hacer un análisis sobre número de ventas totales, la cantidad de ingresos que se recibieron y las ventas mensuales.

Lo primero es saber cuáles fueron las ventas totales por artículo y para esto se debe saber cuántas veces se vendieron los productos y multiplicarlo por su precio para lo que se utilizó el siguiente código:

```
print("TODAS LAS VENTAS")
#Para el total de ingresos primero hay que considerar los productos que fueron vendidos sin devolución.
sale_no_refund=[ [sale[1]] for sale in lifestore_sales if sale[4]==0]
flat_list=[]
for item in sale_no_refund:
    flat_list += item
print(flat_list)
#ahora que tenemos los ID de productos que fueron vendidos, hay que saber los precios de cada uno
print("TODOS LOS PRECIOS")
price_all_products = [[price[2]] for price in lifestore_products]
flat_list=[]
for item in price_all_products:
    flat_list+=item
print(flat_list)


print("TOTAL DE GANANCIAS POR PRODUCTO")
for product in lifestore_products:
    total_price={}
    precio=lifestore_products[0]
print("Las ventas del producto con ID 1 dan una ganancia total de:")
```

```
print(precio[2]*2)
precio=lifestore_products[1]
print("Las ventas del producto con ID 2 dan una ganancia total de:")
print(precio[2]*12)
precio=lifestore_products[2]
print("Las ventas del producto con ID 3 dan una ganancia total de:")
print(precio[2]*42)
...
```

Lo que nos arrojo los siguientes resultados por artículo:

```
TOTAL DE GANANCIAS POR PRODUCTO
Las ventas del producto con ID 1 dan una ganancia total de:
6038
Las ventas del producto con ID 2 dan una ganancia total de:
50508
Las ventas del producto con ID 3 dan una ganancia total de:
129738
Las ventas del producto con ID 4 dan una ganancia total de:
28717
Las ventas del producto con ID 5 dan una ganancia total de:
35580
Las ventas del producto con ID 6 dan una ganancia total de:
35427
Las ventas del producto con ID 7 dan una ganancia total de:
59913
Las ventas del producto con ID 8 dan una ganancia total de:
21596
Las ventas del producto con ID 10 dan una ganancia total de:
889
Las ventas del producto con ID 11 dan una ganancia total de:
22197
```

Posteriormente, se calculó el total de ingreso considerando el ingreso individual por artículo vendido lo que arrojo un total de:

$ 723,854.00

Ahora queremos saber el monto total de ingreso por mes para saber cuáles fueron los meses en los que la tienda tuvo un mejor y peor desempeño de ventas para lo que se utilizó el siguiente código:

```
print("TOTAL DE INGRESO ECONÓMICO Y VENTAS POR MES:")
id_fecha = [ [sale[0], sale[3]] for sale in lifestore_sales if sale[4] == 0 ]
```

```
categorizacion_meses = {}

for par in id_fecha:
    id = par[0]
    _, mes, _ = par[1].split('/')
    if mes not in categorizacion_meses.keys():
        categorizacion_meses[mes] = []
    categorizacion_meses[mes].append(id)

print("A continuación se muestran los ingresos totales por mes así como las ventas que hubo durante dicho mes")
print("\n\t")

mes_info = {}
for mes, ids_venta in categorizacion_meses.items():
    lista_mes = ids_venta
    suma_venta = 0
    for id_venta in lista_mes:
        indice = id_venta - 1
        info_venta = lifestore_sales[indice]
        id_product = info_venta[1]
        info_prod = lifestore_products[id_product-1]
        precio = info_prod[2]
        suma_venta += precio
    print(f'Durante el mes {mes}, hubo una ganancia de ${suma_venta} y {len(lista_mes)} ventas totales')
    mes_info[mes] = [suma_venta, len(lista_mes)]
```

A lo que se obtuvieron los siguientes resultados:

```
A continuación se muestran los ingresos totales por mes así como las ventas que hubo durante dicho mes


Durante el mes 07, hubo una ganancia de $26949 y 11 ventas totales
Durante el mes 02, hubo una ganancia de $107270 y 40 ventas totales
Durante el mes 05, hubo una ganancia de $91936 y 34 ventas totales
Durante el mes 01, hubo una ganancia de $117738 y 52 ventas totales
Durante el mes 04, hubo una ganancia de $191066 y 74 ventas totales
Durante el mes 03, hubo una ganancia de $162931 y 49 ventas totales
Durante el mes 06, hubo una ganancia de $36949 y 11 ventas totales
Durante el mes 08, hubo una ganancia de $3077 y 3 ventas totales
```

De lo cual podemos resumir:

Mes	Ganancia ($)	Número de ventas
Enero	117,738	52
Febrero	107,270	40
Marzo	162,931	49
Abril	191,066	74
Mayo	91,936	34
Junio	36,949	11
Julio	26,949	11
Agosto	3,077	3

De lo que podemos concluir que el mes de mayores ventas y ganancias fue en Abril.

Ordenando de mayor a menor los meses con más ventas:

Mes	Número de ventas
Abril	75
Enero	52
Marzo	49
Febrero	40
Mayo	34
Junio	11
Julio	11
Agosto	3

Solución al problema

Antes de proponer soluciones a los problemas que enfrenta la tienda hay que sintetizar lo que se ha encontrado hasta el momento.

De enero a agosto hubo una notable reducción en las ventas y el resto de los meses de agosto a diciembre hubo devoluciones por parte de los clientes. De igual forma, es importante notar que en enero los ingresos fueron de $117,738.00 a comparación con los ingresos el mes de agosto que fueron de $3,077.00. La tienda también indicó que tiene un problema de acumulación de inventario.

De acuerdo al siguiente código:

```
print("suma de artículos en stock")

suma_stock=[ [stock[4]] for stock in lifestore_products ]

flat_list=[]

for item in suma_stock:

    flat_list+=item


suma_de_stocl=flat_list


print(flat_list)

print("\n\t")


suma_del_total_stock=sum(flat_list)

print(int(suma_del_total_stock))
```

Existen un total de 5341 productos en stock, considerando que se vendieron 274 artículos en total, se necesita una estrategia para vender todos los productos lo más pronto posible y recuperar la inversión.

Propuestas

- Preparar marketing apropiado dependiendo el tipo de consumidores que hagan compras en la página de la tienda. Solicitar un correo electrónico

posterior a la compra de un artículo para enviar publicidad sobre descuentos, promociones y artículos disponibles en almacén.

- Los artículos de la tienda son variados, sin embargo, todos pertenecen a la misma rama de consumidores, tecnología y videojuegos, por lo que las personas que compran en esta tienda online son en su mayoría entusiastas de la tecnología o jóvenes '*gamers'* por lo que convenios con marcas pueden ser recomendados, proponer la tienda Lifestore dentro de las opciones de compra de los sitios web de cada marca.
- Tener presencia en redes sociales como Discord, Reddit, Twitch o Steam pero también en YouTube (solicitar un espacio de máximo 5 segundos para un anuncio), Instagram y Twitter.

Referencia.

Universidad de Alcalá. 2021. Lenguajes de programación para Data Science. [https://www.master-data-scientist.com/lenguajes-programacion-data-science/]