“CIENCIA DE DATOS”

PROYECTO:” FUNDAMENTOS DE PROGRAMACIÓN CON PYTHON”

LÍDERES FUNED

LISSET FLORES URBANO

Índice

Introducción

Este programa es un ejemplo de la clasificación de datos básica solicitada en varios sectores donde se puede obtener un análisis rápido y preciso de datos.

Los fundamentos de programación nos dan una noción y habilidad para poder formular una solución de procesamiento de datos e integrarla como solución a nuestras necesidades.

El procesamiento de datos por medio de programación es una revolución de modernización a la toma de decisiones de acuerdo a un análisis de información.

La empresa de Life Store tiene que clasificar la información de sus productos en categorías de ingresos, ventas y búsqueda para que por medio de los resultados pueda tomar decisiones que incrementaran su rentabilidad.

DESARROLLO

La empresa de LifeStore requiere discreción y confidencialidad al manejo de los datos, por lo cual debe tener una administración correcta de las personas que acceden a la información, la credencial para acceder será limitada según se requiera en la empresa y podrán ser modificadas en sospecha de mal uso, el usuario y contraseña son los siguientes:

Usuario: Administrador Contraseña: LifeStoreConfidential

LifeStore es una empresa que maneja mucha información, pero requiere seccionar la misma para una mejor gestión. De acuerdo al análisis de los datos registrados la empresa solicita saber los siguientes puntos:

A. Productos más vendidos y productos rezagados

Localizar los productos con los que se obtienen mejores ganancias es

- 5 Productos mayores ventas
- 5 menores ventas

Productos **más** vendidos			Productos **menos** vendidos		
No. ventas	ID	Nombre	No. ventas	ID	Nombre
236	54	SSD Kingston A400, 120GB, SATA III, 2.5", 7mm	1	17	Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0
202	3	Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth	1	45	Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel
94	5	Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake	2	46	Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel
82	42	Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD	3	89	Cougar Audífonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro
73	57	SSD Adata Ultimate SU800, 256GB, SATA III, 2.5", 7mm	4	10	MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0

- 10 productos mayores búsquedas
- 10 menores búsquedas

Productos **más** buscados			Productos **menos** buscados		
No. ventas	ID	Nombre	No. ventas	ID	Nombre
263	54	SSD Kingston A400, 120GB, SATA III, 2.5", 7mm	1	9	Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart
107	57	SSD Adata Ultimate SU800, 256GB, SATA III, 2.5", 7mm	1	10	MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0
60	29	Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450	1	27	ITarjeta de Video VisionTek AMD Radeon HD5450, 2GB GDDR3, PCI Express
55	3	Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache	1	35	Tarjeta Madre Gigabyte micro ATX Z390 M GAMING, S-1151, Intel
41	4	Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60	1	45	Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32
35	85	Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm	1	59	SSD Samsung 860 EVO, 1TB, SATA III, M.2
32	67	TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro	1	70	Samsung Smart TV LED 43, Full HD, Widescreen, Negro
31	7	Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee La (9na. Generación Coffee Lake)	1	80	Ghia Bocina Portátil BX800, Bluetooth, Inalámbrico, 2.1 Canales
30	47	SSD XPG SX8200 Pro, 256GB, PCI Express, M.2	1	93	Ginga Audífonos con Micrófono GI18ADJ01BT-RO, Bluetooth, Alámb
30	5	Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake. Generación - Coffee Lake)	2	13	Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB

B. Productos por reseña de servicio
    - 5 productos mejores reseñas
    - 5 productos con las peores reseñas

Peores reseñas		Mejores reseñas	
Promedio de reseña	ID	Promedio de reseña	ID
4	10	4.72	54
4	13	4.8	3
1	17	4.7	5
5	22	4.5	42
5	28	4.8	57

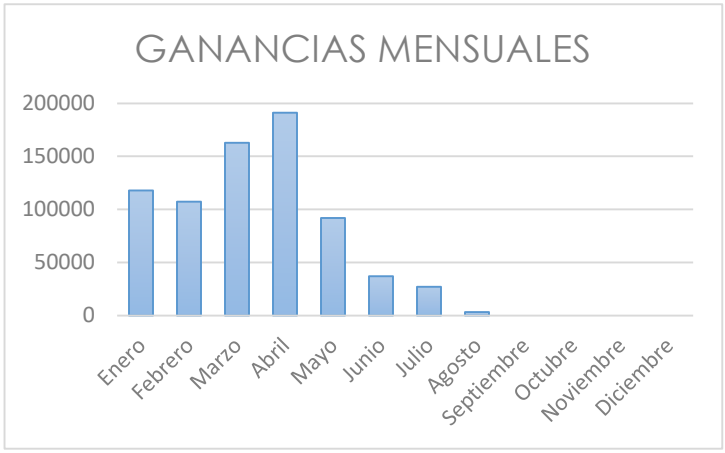
C. Total de ingresos

En los ingresos anuales se realizó el cálculo del total de las ventas que tuvieron éxito, eso quiere decir que el producto no tuvo retorno, y el monto total fue de: $737916

D. Ingresos mensuales

El ingreso mensual permite saber en qué temporada mensual se tuvieron mayores ingresos para la compañía, y fueron los siguientes:

Enero: $117738
Febrero: $107270
Marzo: $162931
Abril: $191066
Mayo: $91936
Junio: $36949
Julio: $26949
Agosto: $3077
Septiembre: $0
Octubre: $0
Noviembre: $0
Diciembre: $0



E. Ventas

El control de información de las ventas es fundamental para poder saber cuáles son los artículos que son la preferencia de los usuarios y en que temporada es donde la demanda incrementa para poder tener los artículos con más demanda disponibles y los que no presentan demanda considerar descartarlos y/ o no invertir en ese artículo debido a su baja demanda en el mercado.

- Ventas promedio mensuales:

Las ventas mensuales según el periodo de registro fueron los siguientes:

Enero: 52 ventas

Febrero: 40 ventas

Marzo: 49 ventas

Abril: 74 ventas

Mayo: 34 ventas

Junio: 11 ventas

Julio: 11 ventas

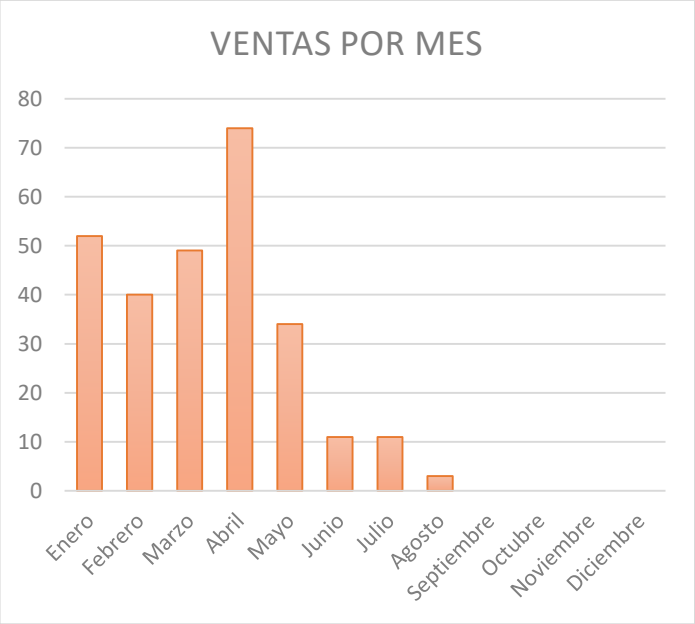
Agosto: 3 ventas

Septiembre: 0 ventas

Octubre: 0 ventas

Noviembre: 0 ventas

Diciembre: 0 ventas



- Total anual

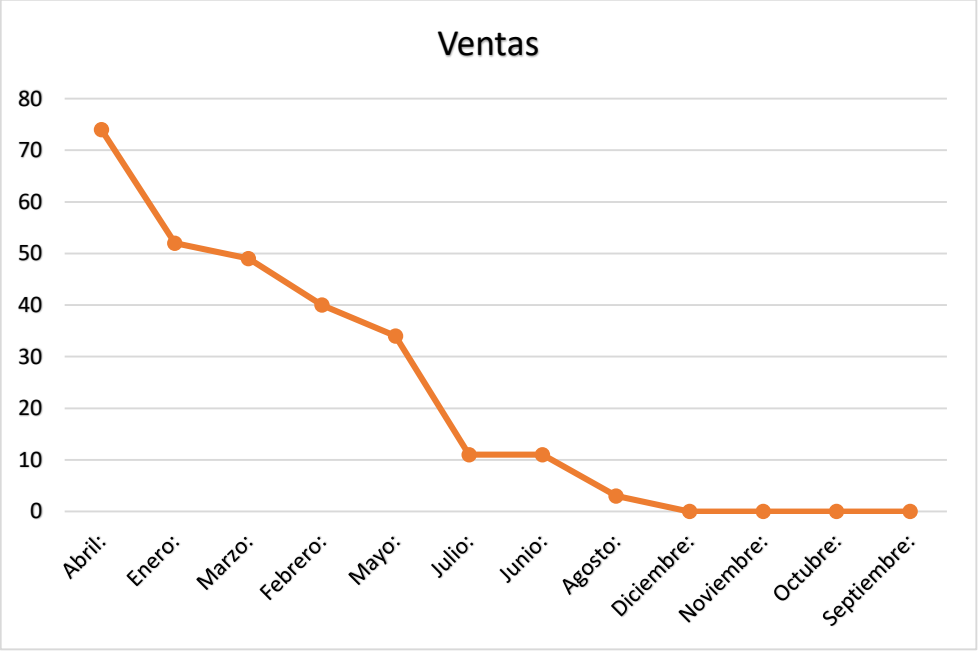
Las ventas anuales fueron efectuadas en dos categorías, las ventas efectivas y las ventas registradas, donde las ventas registradas con las cuales se están considerando aún si estas fueron regresadas. Y son las siguientes:

Ventas totales registradas: 283

Ventas efectivas totales: 274

- Meses con más ventas :
  La empresa LifeStore también solicito saber los meses donde se registraron más ventas para saber cuál fue su temporada más alta y si coincidió con el mes que registro más ingresos y el resultado fue el siguiente:

  Abril: 74 ventas
  Enero: 52 ventas
  Marzo: 49 ventas
  Febrero:40 ventas
  Mayo: 34 ventas
  Julio:11 ventas
  Junio:11 ventas
  Agosto: 3 ventas
  Diciembre: 0 ventas
  Noviembre:0 ventas
  Octubre: 0 ventas
  Septiembre: 0 ventas



g. Definición del código:

```
# Añadimos la función para importar otro programa
from lifestore_file import lifestore_products, lifestore_sales,
lifestore_searches

if __name__ == "__main__" : #Se define la condicion para empezar el
programa

    USUARIO_PERMITIDO = 'Administrador' #Definimos credenciales de inicio
    CONTRASENA = 'LifeStoreConfidential'


    INTENTOS = 3 #Se crea el número de veces que el usuario se puede
equivocar para permanecer en el bucle while


    while True: #Inicio de bucle

        if INTENTOS == 0 : #El contador se inicia en cero

            Salida()
        username = input ( "Ingrese su nombre de usuario: \ n>" ) #Se
solicitan las credenciales de usuario
        contraseña = input ( "Ingrese la contraseña: \ n>" )

        if username == USUARIO_PERMITIDO: #Se hace la comparación si el
usuario existe dentro de nuestra variable

            si contraseña == CONTRASENA:

                break #Solo si los datos fueron los admitidos aqui
termina el ciclo
        else :

            INTENTOS = INTENTOS - 1

            print (f '\ n !! Usuario / Contraseña incorrecto (s),
{INTENTOS} restantes !! \ n' )


    #Comenzamos el codigo de las listas
    print (f "\ n \ n \ n¡Bienvenido! {USUARIO_PERMITIDO}, \ nA
continuación se muestra el reporte administrativo de retutina" )

    prod_ventas = []
    cantidad_de_productos = len (lifestore_products) #Se obtiene la
cantidad de datos existentes en el arreglo
    cantidad_productos = str (cantidad_de_productos) #Se convierte el
dato en cadena para que pueda imprimirse con texto
    cantidad_de_busquedas = len (lifestore_searches)
```

```
    cantidad_busquedas = str (cantidad_de_busquedas)



    print ( "" )
     print ( "" )
     print ( "La cantidad de productos es:" + cantidad_productos) # se
imprime texto con resultado de variable
    print ( "" )
     print ( "La cantida de busquedas registrada al momento son:" +
cantidad_busquedas)
     print ( "" )

# ------------- Filtro de ventas reales -------

ventas = []   # se declara arreglo
for sales in lifestore_sales: #iterable con tabla de donde debe salir la
informacion
    if sales [ 4 ] == 1 : # indice de donde se tomara la información y la
condición donde este no tomara en cuenta
        continue  #da la instrucción de continuar si la condición es
diferente
    ventas.append (sales) #Arreglo con las ventas reales

print ( "Total de productos salidos / ventas registradas es:" )
 print (len (lifestore_sales))
 print ( "" )
 print ( "Total de ventas reales:" )
 print (len (ventas))
 print ( "" )
 print ( "" )
 print ( "----------- VENTAS POR MES ----------------" )
 # ----- Ventas por mes --- -------
meses = [
    '/ 01 /' , '/ 02 /' , '/ 03 /' , '/ 04 /' , '/ 05 /' , '/ 06 /' ,
     '/ 07 /' , '/ 08 /' , '/ 09 / ' , ' / 10 / ' , ' / 11 / ' , ' / 12 /
'
    ]   #definicion de arreglo de manera manual

ventas_por_mes = []
por mes en meses:
    lista_vacia = []
    ventas_por_mes.append (lista_vacia) # da la instrucción de añadir
datos recolectados

para venta en ventas:

    id_venta = venta [ 0 ] #Se definen los datos que se utilizan en la
lista
    fecha = venta [ 3 ]

    contador_de_mes = 0  #Se utilizara un contador y en este caso lo
iniciamos en cero
```

```
    por mes en meses:
         si mes en fecha:
            ventas_por_mes [contador_de_mes] .append (id_venta)
            continuar
        contador_de_mes = contador_de_mes + 1



contador_de_mes = 0
para venta_mensual en ventas_por_mes:

    print (f 'En el mes {meses [contador_de_mes]} se registraron: {len
(venta_mensual)} ventas' )

    contador_de_mes = contador_de_mes + 1

# --------- Ganancias mensuales --------
gancias_mensuales = []
para venta_mensual en ventas_por_mes:
    ganancia_del_mes = 0
    para id_venta en venta_mensual:
        indice_de_venta = id_venta - 1
        info_de_venta = lifestore_sales [indice_de_venta]

        id_prod = info_de_venta [ 1 ]
        indice_de_prod = id_prod - 1
        info_del_prod = lifestore_products [indice_de_prod]
        precio_de_prod = info_del_prod [ 2 ]
        ganancia_del_mes = ganancia_del_mes + precio_de_prod
    gancias_mensuales.append (ganancia_del_mes)


print ( "" )
 print ( "" )
 print ( "------------- INGRESOS MENSUALES SON -----------------" )
contador_de_mes = 0
n = 0  #ya que las ganancias por mes se muestran todas en una lista este
contador para realizar el cambio segun el mes
para ingresos_mes en el rango ( 12 ):
     print (f 'Las ganancias en el mes {meses [contador_de_mes ]} fueron:
$ {gancias_mensuales [n]} ' )
    contador_de_mes = contador_de_mes + 1
    n = n + 1

print ( "---------------- GANANCIA ANUAL --------------" )
Ingreso_anual = suma (gancias_mensuales)
print ( "" )
 print ( "" )
 print (f "El ingreso anual es: $ {Ingreso_anual}" )
 print ( "" )
```

```
print ( "------------ MESES CON MAYORES VENTAS ------------" ) #checar el
ordenamiento con sort

meses_mayores = []
para mes, venta_mensual en enumerate (ventas_por_mes):
    cant_ventas_mensuales = len (venta_mensual)
    sublista = [cant_ventas_mensuales, mes]
    meses_mayores.append (sublista)



meses_mayores.sort (reverse = True)

imprimir (meses_mayores)

imprimir ( "--------------------------------" )

prod robust = [] # reseñas de productos
para prod in lifestore_products: # de la LISTA DE PRODUCTOS sacamos los
índices que utilizaremos
    id_prod = prod [ 0 ] # el id que le corresponde al producto
    sublista = [id_prod, 0 , 0 ] # se crea de la lista con el id y dos
elementos de la misma
    vacios prod_reviews.append (sublista) #score
# en lugar de lifestroe_sales poner ventas [0]
for venta in lifestore_sales: # ahora se trabaja con la LISTA DE VENTAS
    id_prod = venta [ 1 ]# este índice también le corresponde al id del
producto
    review = venta [ 2 ] # este indice le corresponde a la calificación

    indice = id_prod - 1
    prod× [indice] [ 1 ] + = review #asignamos el valor a la lista de
score en index 1 reseña
    prod larger [indice] [ 2 ] + = 1  #Asignamos el valor a la lista
score index 2 // cantidad de ventas


para indice, lista en enumerate (prod×): #enumerate nos permite tener
control del iterable como del id siguiente
   suma = lista [ 1 ] ## reseña
   cant = lista [ 2 ] ## cantidad de ventas
   if cant> 0 :
       becerro_prom = suma / cant
       prod robust [indice] [ 1 ] = calf_prom # promedio de calificación
de reseña


# ********************** Para obtener los mas vendidos:
****************** *******
mejores_review = []
for lista in prod 0000-:
     #
```

```
    sublista = [lista [ 2 ], lista [ 0 ], lista [ 1 ]] # la misma lista
de review pero cambiando orden de iterables
    mejores_review.append (sublista)      #quedando como cantidad de
ventas / id de producto / promedio de reseña

# ------------------- PRODUCTOS CON RESEÑAS -------------------------- -
-----
review_existente = []
para cuentan_con_review en mejores_review:
     if cuentan_con_review [ 2 ] == 0 :
         continue
    review_existente.append (cuentan_con_review) #Arreglo con productos
que tienen reseña



review_existente.sort () #Se ordenan
print ( "Peores notas" )
 for top_menos_vendidos in review_existente [: 5 ]:
     print (top_menos_vendidos)



print ( "Mejores reseñas" )
 print ( "cantida de ventas, id producto, promedio de reseña" )
review_existente.sort (reverse = True)
para top_mas_vendidos en review_existente [: 5 ]:
     print (top_mas_vendidos)

print ( "" )
 print ( "----------- PRODUCTOS CON MEJORES Y PEORES VENTAS ------------
" )


prod_numeros_ventas = [] #productos por numero de ventas
for prod in lifestore_products: # de la LISTA DE PRODUCTOS sacamos los
índices que utilizaremos
    id_prod = prod [ 0 ] # el id que le corresponde al producto
    id_name = prod [ 1 ] #nombre del prodcuto
    sublista = [id_prod, 0 , id_name] # se selecciona la posición de los
elementos en la nueva lista
    prod_numeros_ventas.append (sublista) #score

for venta in lifestore_sales: # ahora se trabaja con la LISTA DE VENTAS
    id_prod = venta [ 1 ] # este indice tambien le corresponde al id del
producto
    review = venta [ 2 ] # este indice le corresponde a la calificación

    indice = id_prod - 1
    prod_numeros_ventas [indice] [ 1 ] + = review #asignamos el valor a
la lista de score en index 1 reseña
```

```
# Para obtener los productos top de mejorees y peores ventas:
ventas_product = []
for lista in prod_numeros_ventas:
     #
    sublista = [lista [ 1 ], lista [ 0 ], lista [ 2 ]] # la misma lista
de review pero cambiando orden de iterables
    ventas_product.append (sublista)      #quedando como cantidad de
ventas / id de producto / promedio de reseña




tuvieron_ventas = [] # arreglo para que filtre lo que tuvieron ventas
para cuentan_con_venta in ventas_product:
     if cuentan_con_venta [ 0 ] == 0 :
         continue
    tuvieron_ventas.append (cuentan_con_venta) #Arreglo con productos que
tienen reseña



imprimir ( "" )
tuvieron_ventas.sort ()
print ( "PRODUCTOS MENOS VENDIDOS" )
 print ( "cantida de ventas, id producto, nombre" )
 para top_menos_vendidos en tuvieron_ventas [: 5 ]:
     print (top_menos_vendidos)

imprimir ( "" )

imprimir ( "PRODUCTOS MÁS VENDIDOS" )
 imprimir ( "cantida de ventas, id producto, nombre" )
tuvieron_ventas.sort (reverse = True)
para top_mas_vendidos en tuvieron_ventas [: 5 ]:
     print (top_mas_vendidos)


# ---------- SECCION BUSQUEDAS DE LOS PRODUCTOS -------------
# ------------- PRODUCTOS CON MAYORES Y MENORES BUSQUEDAS - -------------
-----

prod_busquedas = []
para prod en lifestore_products:
    id_prod = prod [ 0 ]
    id_name = prod [ 1 ]
    sublista = [id_prod, 0 , 0 , id_name]
    prod_busquedas.append (sublista)
# en lugar de lifestroe_sales poner ventas [0]
for venta in lifestore_searches:
    id_prod = venta [ 1 ]
```

```
    contar = venta [ 1 ]

    indice = id_prod - 1
    prod_busquedas [indice] [ 1 ] + = contar #indice para el conteo
    prod_busquedas [indice] [ 2 ] + = 1   #posiblemente esta linea no me
sirva pero la añadi por que sin ella no me respetaba la fusion de las
listas


buscados_a = []
para lista en prod_busquedas:
    sublista = [lista [ 2 ], lista [ 0 ], lista [ 3 ]]   #numero de
busquedas, id de producto, nombre de producto
    buscados_a.append (sublista)


tuvieron_busquedas = [] # arreglo para que filtre lo que tuvieron ventas
for cuentan_con_busqueda in buscados_a:
     if cuentan_con_busqueda [ 0 ] == 0 :
         continue
    tuvieron_busquedas.append (cuentan_con_busqueda) #Arreglo con
productos que tienen reseña



imprimir ( "" )
tuvieron_busquedas.sort ()
print ( "PRODUCTOS MENOS BUSCADOS" )
 print ( "cantida de ventas, id producto, nombre" )
 para top_menos_buscados en tuvieron_busquedas [: 10 ]:
     print (top_menos_buscados)


imprimir ( "" )
 imprimir ( "PRODUCTOS MAS BUSCADOS" )
 imprimir ( "cantida de ventas, id producto, nombre" )
tuvieron_busquedas.sort (reverse = True)
para top_mas_buscados en tuvieron_busquedas [: 10 ]:
     imprimir (top_mas_buscados)
```

h. Solución al problema

Para un mejor manejo de la información LifeStore debería filtrar su información de manera diaria para poder monitorear el momento en línea donde sus ventas crecen o disminuyen, podría ser por medio de una base de datos conectada online y también con el dato de las búsquedas podrán determinar por medio de más parámetros las necesidades de los usuarios.

i.Conclusión

El curso de introducción a Python nos da una perspectiva de manejo de los datos para poder obtener resultados de acuerdo a una necesidad y criterios de la misma para obtener información más clara y precisa para según las necesidades. El manejo de información por medio de ciencia de datos reduce tiempo de inversión para el análisis de los datos más importantes para los clientes y aumentar su rentabilidad detectando problemas de una manera más eficiente y precisa.

j. Anexos

Link de GitHub: https://github.com/LissetUrbano/Proyecto_Emtech