

# Analís is de rotación de productos

Propuesta de mejora



Presentado por

**DIANA AVILES**

ING. TECNOLOGÍAS DE CÓMPUTO Y  
TELECOMUNICACIONES

Revisado por

JAIME SAÚL ALONSO SÁNCHEZ

# Índice

• <u>Situación actual</u>	1
• <u>Propuesta</u>	2
○ <u>FASE 1</u>	3
○ <u>FASE 2</u>	4
○ <u>FASE 3</u>	5
○ <u>FASE 4</u>	6
○ <u>FASE 5</u>	11
• <u>Conclusión</u>	12
• <u>ANEXOS</u>	13

# Sitación actual

 **LifeStore** es una tienda virtual que maneja una amplia gama de artículos en su mayoría relacionados a la tecnología.

La Gerencia de ventas se ha percatado que la empresa tiene una importante acumulación de inventario. Asimismo, se identificó una reducción en las búsquedas de un grupo importante de productos, lo que ha redundado en una disminución sustancial de ventas del último trimestre.

Derivado de la situación la Gerencia de Ventas solicita un análisis de la rotación de productos identificando los siguientes requisitos.

- 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.

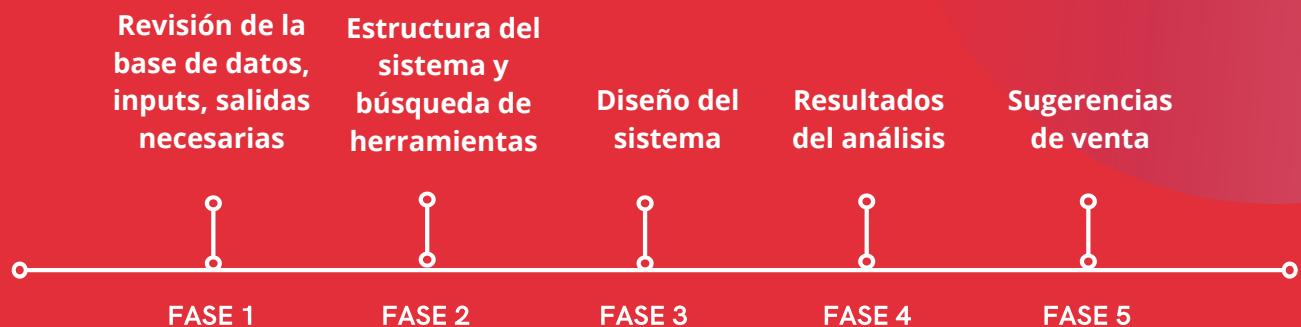


## Experiencia

- ✓ Egresada de la Universidad Iberoamericana en Ingeniería en Tecnologías de Cómputo y Telecomunicaciones.
- ✓ Certificada por EMTECH Institute en análisis de Datos con Python y en Project Management.
- ✓ **1er lugar** en Hackathon Hot Autonomous Pursuit por Accenture, dando la mejor solución en conectar distintas capacidades de AWS integradas a un vehículo autónomo con el análisis de datos.
- ✓ **3er lugar** en Innovation Challenge por Oracle, dando desarrollo a una solución para los Adultos mayores con la ayuda de IoT, Inteligencia Artificial, Blockchain y Big data.



# Propuesta



El análisis de rotación de productos señala el total de veces que el inventario del almacén requiere ser abastecido con nuevas existencias. Para realizar un análisis detallado es importante conocer los productos con más demanda, lo más buscados por los usuarios, así como los que menos se venden. Con esta información se puede obtener el total de ingresos anuales, el número de ventas, el mes con más ventas, etc.

Al recopilar toda la información se puede plantear una estrategia para mejorar las ventas.

La propuesta consiste diseñar un programa usando python, para analizar las entradas. Con este sistema es más fácil visualizar las ventas de **LifeStore** y poder generar un reporte de ventas para futuras decisiones.

Tomamos en cuenta la seguridad, por lo que solo personas autorizadas tendrán el acceso a dicho sistema.

# Fase 1

Antes de diseñar el sistema, es importante conocer la información necesaria para obtener los mejores resultados.

Nuestra **entrada** es la Base de datos de LifeStore y se llevó a cabo el siguiente análisis:

Para la **salida**, se busca realizar un reporte que indique:

- **Productos más vendidos y productos rezagados**
  - Top 5 de productos más vendidos
  - Top 10 de productos más buscados
  - Top 10 de productos menos buscados
  - Categorías menos vendidas
- **Productos por reseña en el servicio**
  - Top 5 productos con mejores reseñas
  - Top 5 productos con peores reseñas
  - Listado detallado de las mejores reseñas por producto
  - Listado detallado de las peores reseñas por producto
- **Total de ingresos anual, número de ventas y top 3 de meses con mayores ventas**
  - Número de ventas totales
  - Número de ventas totales sin reembolso
  - Total de Ingresos en el 2020 sin reembolso
  - Top 3 de Meses con más ventas en el año



## Salida



## Fase 2

En sistema será hecho en **Python** contiene librerías que nos ayudan en el análisis de datos. También nos apoyaremos en la herramienta **Replit**, ya que es un IDE, editor, compilador, intérprete en línea, simple pero muy poderoso; es muy útil para las pruebas, permitiendo a futuro crear una aplicación para mantener en tiempo real la información actualizada

El proyecto estará en un repositorio en github para llevar un control de versiones. Incorporo los links de acceso para la prueba del sistema.

Para ejecutarlo en replit seguir es entrar a la pestaña Shell y ejecutar el siguiente comando

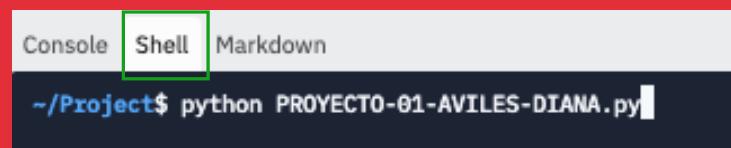
Se utilizó una programación modular para tener un mejor orden. Consta de tres archivos:

- **functions.py**
  - Archivo donde se desarrollan todas las funciones del sistema.
- **lifestore\_file.py :**
  - Base de datos de Lifestore donde obtendremos toda la información necesaria.
- **PROYECTO-01-AVILES-DIANA.py**
  - Es el archivo principal para ejecutar, donde se encuentra el esqueleto del sistema. Aquí solo se imprimen los resultados de las funciones



<https://github.com/avilesdiana/LifeStore>

<https://replit.com/join/ualagqspws-avilesdiana>

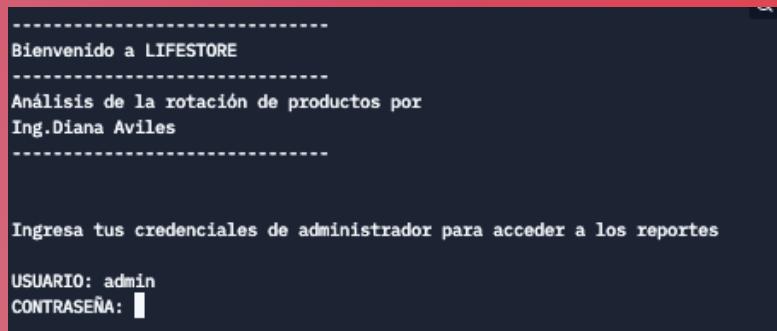


A screenshot of the Replit interface showing a terminal window. The tabs at the top are "Console", "Shell" (which is highlighted with a green border), and "Markdown". The terminal output shows the command `~/Project$ python PROYECTO-01-AVILES-DIANA.py`.

El código detallado se encuentra en los **Anexos** del presente documento, donde se explica cada función y el procedimiento.

# Fase 3

## Acceso



Al ingresar de manera correcta tus credenciales te abrirá el menu que se muestra en la imagen, cada opción te mostrará información explicada en la [FASE 1.](#)

Puedes regresar al menu las veces que tu quieras presionando **ENTER**, también siempre y cuando este la opción en el menú.

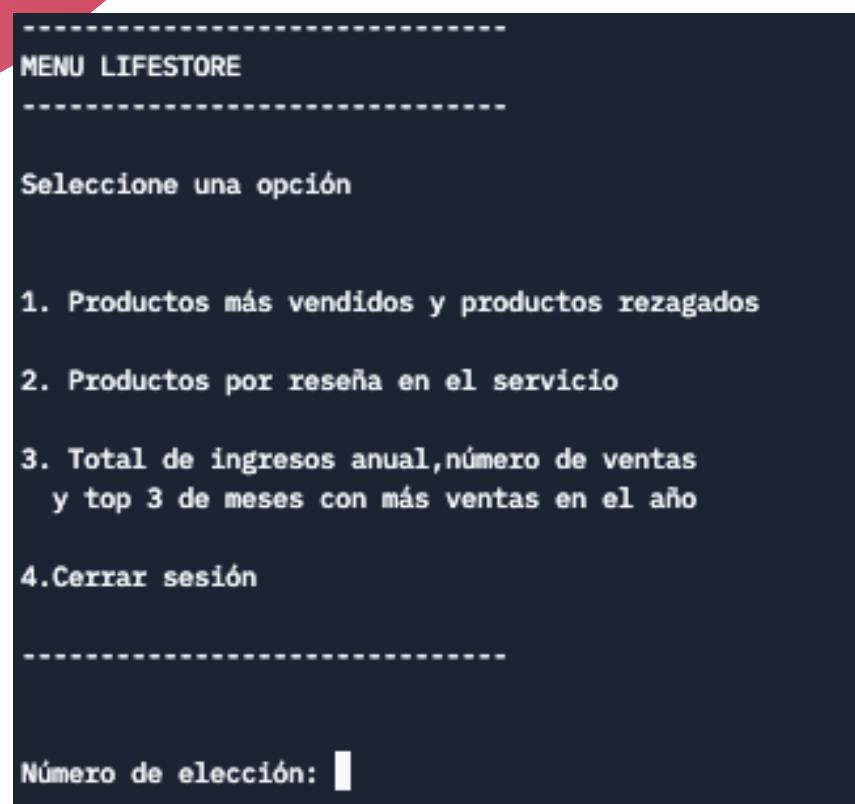
Para salir del menú solo basta con darle a la opción **Cerrar sesión**.

Para ingresar es necesario que el Gerente te de acceso autorizado, para fines de prueba usaremos las siguientes:

**Usuario: admin**  
**Contraseña: 1234**

\*Asegurate de ingresar tal cuál esta escrito arriba, a las tres veces de ingresar datos incorrectos, te bloqueará y seguirás las instrucciones del programa. Por seguridad al momento de ingresar la contraseña no aparecerá en la pantalla

## Menú



# Fase 4

## Resultados del Análisis



### 1. Productos más vendidos y productos rezagados

Solo se tomaron en cuenta los productos que no tuvieron reembolso

#### Top 5 más vendidos

##### id - Componente

[3, 'Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth']

[5, 'Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake)']

[42, 'Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD']

[54, "SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm"]

[57, "SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm"]

1



Procesador AMD Ryzen 5 Id: 3

2



Procesador Intel Corei3 Id: 5

3



Tarjeta ASRock Id: 42

4



SSD Kingston A400 Id: 54

5



SSD Adata Ultimate SU800 Id: 59

#### Top 10 más buscado

##### id - Componente

[3, 'Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth']

[4, 'Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire']

[5, 'Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake)']

[7, 'Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake)']

[29, 'Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD']

[47, 'SSD XPG SX8200 Pro, 256GB, PCI Express, M.2']

[54, "SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm"]

[57, "SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm"]

[67, 'TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro']

[85, 'Logitech Audifonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul']

1



2

3

4



6

7



9



10

# Fase 4

## Resultados del Análisis



### 1. Productos más vendidos y productos rezagados

Solo se tomaron en cuenta los productos que no tuvieron reembolso

-----  
Categorías menos vendidas  
-----  
  
Categorías existentes:  
['procesadores', 'tarjetas de video', 'tarjetas madre', 'discos duros', 'memorias usb', 'pantallas', 'bocinas', 'audifonos']  
  
-----  
  
Vendidos - Categoría  
  
(1, 'memorias usb')  
(26, 'tarjetas de video')  
(49, 'tarjetas madre')  
(94, 'discos duros')  
(104, 'procesadores')

-----  
Top 10 menos buscado  
-----  
  
id - Componente  
  
[36, 'Tarjeta Madre Gigabyte micro ATX Z490M GAMING X (rev. 1.0), Intel Z490, HDMI, 128GB DDR4 para Intel']  
  
[37, 'Tarjeta Madre ASRock ATX Z490 STEEL LEGEND, S-1200, Intel Z490, HDMI, 12 8GB DDR4 para Intel']  
  
[38, 'Tarjeta Madre Gigabyte Micro ATX H310M DS2 2.0, S-1151, Intel H310, 32GB DDR4 para Intel\xa0']  
  
[41, 'Tarjeta Madre ASUS micro ATX Prime H370M-Plus/CSM, S-1151, Intel H370, H DMI, 64GB DDR4 para Intel']  
  
[43, 'Tarjeta Madre ASUS ATX ROG STRIX Z390-E GAMING, S-1151, Intel Z390, HDMI , 64GB DDR4 para Intel']  
  
[53, 'SSD Addlink Technology S70, 512GB, PCI Express 3.0, M.2']  
  
[55, 'SSD para Servidor Supermicro SSD-DM128-SMCNVN1, 128GB, SATA III, mSATA, 6Gbit/s']  
  
[58, "SSD para Servidor Lenovo Thinksystem S4510, 480GB, SATA III, 2.5'', 7mm"]  
  
[60, 'Kit Memoria RAM Corsair Dominator Platinum DDR4, 3200MHz, 16GB (2x 8GB), Non-ECC, CL16, XMP']  
  
[61, 'Kit Memoria RAM Corsair Vengeance LPX DDR4, 2400MHz, 32GB, Non-ECC, CL16']



# Fase 4

## Resultados del Análisis



### 2. Productos por reseña en el servicio

Se tomo en cuenta a todos los productos con score >= 1

#### Productos con mejores reseñas (5-3)

id - Componente

- (1, 'Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache')
- (2, 'Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth')
- (3, 'Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth')
- (4, 'Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire')
- (5, 'Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na Generación - Coffee Lake)')

1



2



3



4



5



#### Productos con peores reseñas (1-3)

id - Componente

- (17, 'Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0')
- (29, 'Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD')
- (31, 'Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD')
- (45, 'Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel')
- (46, 'Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel')

1



2



3



5



4



8

# Fase 4

## Resultados del Análisis



### 2. Productos por reseña en el servicio

Para ver más a detalle la lista completa de todos los productos con su score de manera ordenada, se imprime lo siguiente:

```
-----  
Listado detallado de las mejores reseñas  
-----  
  
id_producto - Score  
  
(1, 5)  
(2, 5)  
(3, 5)  
(4, 5)  
(5, 5)  
(6, 5)  
(7, 5)  
(8, 5)  
(11, 5)
```

Se ordeno a los productos por score donde **5** es una excelente reseña y **1** es una mala reseña.



```
-----  
Listado detallado de las peores reseñas  
-----  
  
id_producto - Score  
  
(17, 1)  
(29, 1)  
(31, 1)  
(45, 1)  
(46, 2)  
(54, 2)  
(2, 3)  
(3, 3)  
(4, 3)  
(31, 3)  
(47, 3)  
(48, 3)  
(89, 3)  
(2, 4)
```

# Fase 4

## Resultados del Análisis

### 3. Total de ingresos anual, número de ventas y top 3 de meses con más ventas en el año



Se realizo la suma de todas las ventas en el año sin contar los reembolsos

Se contaron todos los productos vendedor por mes y se saco el Top 3 de los meses que más venden en el año

En este caso es ABRIL, ENERO Y MARZO

Número de Ventas Totales

283 Productos

Número de Ventas Totales sin reembolsos

274 Productos

Total de ingresos en el 2020 sin reembolso

\$737916 Pesos

Top 3 de meses con más ventas en el año

Mes - Número de ventas

(4, 74)

(1, 52)

(3, 49)

## Fase 5

La pandemia de la COVID-19 disparó en 2020 la compra de portátiles y de material informático para hacer posible el teletrabajo y la enseñanza online. Y también supuso un revulsivo para todo lo que tiene que ver con el ocio digital en el hogar. Y especialmente para el PC gaming.

Como podemos observar en los resultados del análisis los productos más vendidos son los **procesadores y discos duros**, esenciales para armar tu propia pc gamer, tambien son los más buscados y estan en el top de mejores reseñas.

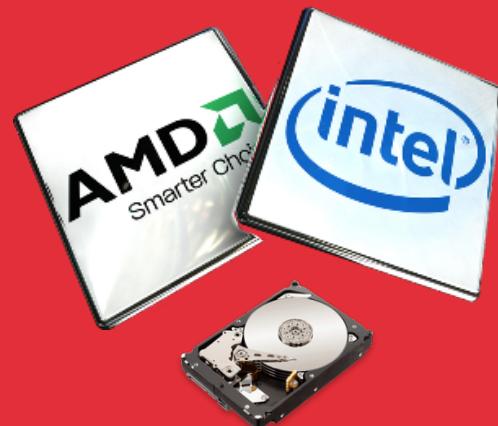
Por otro lado se puede observar que las **pantallas, bocinas y audifonos** son los menos comprados.

Los meses con más ventas son **Enero, Marzo y Abril** que se puede asociar con las vacaciones.

La estrategia para mejorar las ventas es tener en estos meses más vendido más stock de procesadores y discos duros, ya que es cuando tienen más demanda.

En cuanto a reseñas se tiene que mantener la buena calidad, el top es de sobre la categoría de **tarjetas madre**. Al no mantener la calidad se pueden perder ventas, ya que esta categoría esta en el top 5 de mejores ventas.

Para reducir la acumulación de inventario no es necesario la actualización rutinaria de las categorías **pantallas, bocinas y audífonos**.



Actualizar el inventario constantemente para los procesadores y discos duros, ya que tienen mayor demanda en

**LifeStore**



Las pantallas, audífonos y bocinas son las categorías menos vendidas, por lo que no es necesario un restock constante en **LifeStore** y se puede evitar la acumulación de inventario



## Conclusión

"De vez en cuando, una nueva tecnología, un antiguo problema y una gran idea se convierten en una innovación".

Dean Kamen. Creador del Segway y el iBOT.

Hoy en día se tiene que aprovechar la tecnología para mejorar tu negocio y no solo en ventas, te puedo ayudar a organizarte mejor y saber realmente lo que pasa en tu tienda.

Actualmente existen herramientas que nos pueden ayudar a analizar nuestros datos para sacar provecho.

Gracias a este sistema se puede observar de mejor manera el total de tus ventas, los productos más vendidos y tus ganancias , sin la necesidad de realizar el cálculo porqué el sistema lo hará por ti.

La pandemia afecto a muchos negocios por la falta de actualización tecnológica, puede parecer costoso pero es una gran inversión a futuro. No hay que tener miedo a las nuevas tecnologías.

Las soluciones que ofrezco se adaptan al cliente. En caso de aprobar esta herramienta, se pueden implementar dashboard para saber en tiempo real en cualquier lugar lo que sucede en tu negocio.

Espero con su contestación.

**Diana Aviles**

Ing. Tecnologías de Cómputo y  
Telecomunicaciones

[AVILESDIANA@OUTLOOK.COM](mailto:AVILESDIANA@OUTLOOK.COM)

## ANEXOS

### 1 PROYECTO-01-AVILES-DIANA.py

```
#Importa las funciones que desarrolle en mi archivo functions.py
import functions
```

```
#Librer a para mandar a llamar la informaci n de Lifestore
from lifestore_file import lifestore_products, lifestore_sales, lifestore_searches
#print(lifestore_searches[0])
```

```
#Inicializando el contador para la evaluaci n
```

```
validation_Count = 0
validation_Menu = 0
session_error = 3
```

```
#Login
```

```
user_Login = 'admin'
password_Login = '1234'
```

```
#Estructura del programa
```

```
#Mientras el contador == 0, repetira la acci n hasta que ingrese el valor correcto
while validation_Count == 0 and session_error != 0:
```

```
#Mensaje de bienvenida
```

```
functions.clearConsole()
```

```
functions.welcome_System()
```

```
user_Access = input('USUARIO: ')
```

```
if user_Access == user_Login:
```

```
#getpass es para que no aparezca la contrase a en la consola
password_Access = functions.getpass('CONTRASE A: ')
```

```
if password_Access == password_Login:
```

```
functions.clearConsole() #Limpiar consola
```

```
#Al ser diferente de 0 sale del ciclo while
```

```
validation_Count += 1
```

```
functions.log_In()
```

```
functions.clearConsole()
```

```
#validaci n del men , mientras el contador == 0 lo seguir repitiendo
while validation_Menu == 0:
    choiceMainMenu = int(functions.main_menu())
```

```
if choiceMainMenu == 1:
```

```
functions.clearConsole()
```

```
print('-----\nTop 5 m s vendidos\n-----')
```

```
top5 = functions.top5_mostSelledProducts()
print('\nid - Component\n')
for list in top5:
    print(list)
    print(" ")
```

```
print('-----\nTop 10 m s buscado\n-----')
```

```
top10_more = functions.top10_mostWantedProducts()
print('\nid - Componente\n')
for list in top10_more:
    print(list)
    print(" ")
```

```
print('-----\nTop 10 menos buscado\n-----')
```

```
top10_less = functions.top10_leastWantedProducts()
print('\nid - Componente\n')
for list in top10_less:
    print(list)
    print(" ")
```

```
print('-----\nCategor as menos vendidas\n-----')
```

```
print('\nCategor as existentes: \n')
categories, count_products = functions.existingCategories()
print(categories)
print('-----')
print('\nVendidos - Categor a\n')
for list in count_products:
    print(list)
```

```

print('\nid - Componente\n')
for list in top_Badreviews:
    print(list)
    print(" ")

print('\n-----\nListado detallado de las mejores
rese as\n-----')
print('\nid_producto - Score\n')
for list in listT:
    print(list)
    print(" ")

print('\n-----\nListado detallado de las peores
rese as\n-----')
print('\nid_producto - Score\n')
for list in listF:
    print(list)
    print(" ")
functions.enter()
functions.clearConsole()
continue
validation_Menu += 1

elif choiceMainMenu == 3:
    functions.clearConsole()
    sales_total, products_sale, freq_month = functions.sales()
    print('\n-----\nN mero de Ventas Totales\n-----')
    print(str(len(lifestore_sales)) + " Productos")
    print('\n-----\nN mero de Ventas Totales sin
reembolsos\n-----')
    print(str(sales_total) + " Productos")
    print('\n-----\nTotal de ingresos en el 2020 sin
reembolso\n-----')
    print("$" + str(products_sale) + " Pesos")
    print('\n-----\nTop 3 de meses con m s ventas en el
a o\n-----')
    print('\nMes - N mero de ventas\n')
    for list in freq_month:
        print(list)
        print(" ")
    functions.enter()
    functions.clearConsole()
    continue
    validation_Menu += 1

elif choiceMainMenu == 4:
    #funci n para cerrar sesi n
    functions.log_Out()
    validation_Menu += 1

else:
    #Ingreso una valor err neo en el men
    functions.clearConsole()
    functions.wrong_Value('men ')
    functions.enter()
    functions.clearConsole()

#else contrase a
else:
    #Ingreso una contrase a err nea
    functions.clearConsole()
    functions.wrong_Value('contrase a')
    functions.enter()
    functions.clearConsole()
    session_error -=1
    print("\n-----\nPASASTE LAS 3 VECES ,\nContacta a tu
gerente para revisar tus credenciales\
-----")

```

```

    n-----\n")

#else usuario
else:
    #Ingreso un Usuario err neo
    functions.clearConsole()
    functions.wrong_Value('usuario')
    functions.enter()
    functions.clearConsole()
    session_error ==1
    print("\n-----\nPASASTE LAS 3 VECES ,\nContacta a tu
          gerente para revisar tus credenciales\n
-----\n")

```

## 2 functions.py

```

from datetime import date, time, datetime #para usar las fechas
import os #Clase para limpiar pantalla
import time #Clase para pausar tiempo en la consola
import getpass #Clase para que la contraseña no aparezca en consola
import numpy as np
from lifestore_file import lifestore_products, lifestore_sales, lifestore_searches


#Función para limpiar pantalla
def clearConsole():
    command = 'clear'
    if os.name in ('nt', 'dos'): #Si el programa corre en Windows, use cls
        command = 'cls'
    os.system(command)

#Función para mostrar mensaje: Presionar Enter y continuar
def enter():
    input('\n[Presiona ENTER para continuar] ')

#Función para mostrar mensaje Bienvenida
def welcome_System():
    print('\n-----\nBienvenido a LIFESTORE\n-----')
    print('Análisis de la rotación de productos por \nIng. Diana Aviles\n-----\n')
    print('\nIngresa tus credenciales de administrador para acceder a los reportes\n')

#Función para mostrar el menú principal
def main_menu():
    print('\n-----\nMENU LIFESTORE\n-----')
    print('\nSeleccione una opción\n')
    print('\n1. Productos más vendidos y productos rezagados')
    print('\n2. Productos por reseña en el servicio')
    print('\n3. Total de ingresos anual, promedio de ventas\n y top 3 de meses con más
          ventas en el año')
    print('\n4.Cerrar sesión\n')
    print('-----\n')
    choiceOfMainMenu = input('\nNúmero de elección: ')
    return choiceOfMainMenu #Regresa la elección para su validación en el programa
                           #principal

#Función para mostrar el cerrado de sesión
def log_Out():
    clearConsole()
    print('\n-----\nCerrando sesión de LIFESTORE ....')
    print('-----\n')
    time.sleep(1)
    clearConsole()
    print('\n-----\nCerrando sesión de LIFESTORE ...')
    print('-----\n')

```

```

time.sleep(1)
clearConsole()
print('-----\nCerrando sesi n de LIFESTORE .')
print('-----\n')
time.sleep(1)
clearConsole()
print('-----\nCerrando sesi n de LIFESTORE .')
print('-----\n')
time.sleep(1)
clearConsole()

#Funci n para mostrar que ingreso un valor err neo
def wrong_Value(optionText):
    print('-----')
    print('Opci n INCORRECTA')
    print('Ingrese un valor de '+ optionText+' lido')
    print('-----\n')

#Funci n para mostrar el inicio de sesi n
def log_In():
    clearConsole()
    print('-----\nIniciando sesi n de LIFESTORE .')
    print('-----\n')
    time.sleep(1)
    clearConsole()
    print('-----\nIniciando sesi n de LIFESTORE .')
    print('-----\n')
    time.sleep(1)
    clearConsole()
    print('-----\nIniciando sesi n de LIFESTORE .')
    print('-----\n')
    time.sleep(1)
    clearConsole()
    print('-----\nIniciando sesi n de LIFESTORE ....')
    print('-----\n')
    time.sleep(1)
    clearConsole()
    print('-----\nIniciando sesi n de LIFESTORE ....')
    print('-----\n')
    time.sleep(1)
    clearConsole()

#Funci n para obtener el top de 5 productos m s vendidos
def top5_mostSelledProducts():

    validation_refund = []
    count_products = []
    order_list5 = []

    #Agregamos en una nueva lista los productos de ventas que no tuvieron reembolso == 0
    for sale0 in range(len(lifestore_sales)):
        if lifestore_sales[sale0][4] == 0:
            validation_refund.append(lifestore_sales[sale0])
            #Comprobamos que imprima los correctos
            #print(lifestore_sales[sale0])

    #Obtener la validation_refund[1] para guardarla en una lista donde vienen todos los
    #productos que se vendieron
    i = 1 #columna que queremos obtener
    column_products = [fila[i] for fila in validation_refund]

    for products in range(len(lifestore_products)+1):
        element = column_products.count(products)
        count_products.append(element)
        #Comprobamos que imprima los correctos
        #print(str(products) + " " + str(count_products[products]))

    #Convertir en un arreglo para ordenar la lista
    array_countProducts = np.array(count_products)
    #print(array_countProducts)

    #Comprobamos que se ordenen
    #commanded_frequency = np.sort(array_countProducts)[::-1]
    #print(commanded_frequency)

    #Ordenamos por id_producto
    idProduct_order = np.argsort(array_countProducts)[::-1]

```

```

#print(idProduct_order)

#Guardamos en un arreglo los primeros 5
array_idProduct = idProduct_order[0:5]

#Buscamos el id en el arreglo lifestore_products para imprimir el top 5
for column in range(len(lifestore_products)):
    for row in range(len(array_idProduct)):
        if lifestore_products[column][0] == array_idProduct[row]:
            order_list5.append(lifestore_products[column][0:2])

return order_list5

#Función para obtener el top de 10 productos más buscados
def top10_mostWantedProducts():

    count_products = []
    order_list10 = []

    #Obtener la validation_refaund[1] para guardarla en una lista donde vienen todos los
    #productos que se vendieron
    i = 1 #columna que queremos obtener
    column_products = [fila[i] for fila in lifestore_searches]

    for products in range(len(lifestore_products)+1):
        element = column_products.count(products)
        count_products.append(element)
    #Comprobamos que imprima los correctos
    #print(str(products) + " " + str(count_products[products]))

    #Convertir en un arreglo para ordenar la lista
    array_countProducts = np.array(count_products)
    #print(array_countProducts)

    #Comprobamos que se ordenen
    #commanded_frequency = np.sort(array_countProducts)[::-1]
    #print(commanded_frequency)

    #Ordenamos por id_producto
    idProduct_order = np.argsort(array_countProducts)[::-1]
    #print(idProduct_order)

    #Guardamos en un arreglo los primeros 5
    array_idProduct = idProduct_order[0:10]

    #Buscamos el id en el arreglo lifestore_products para imprimir el top 5
    for column in range(len(lifestore_products)):
        for row in range(len(array_idProduct)):
            if lifestore_products[column][0] == array_idProduct[row]:
                order_list10.append(lifestore_products[column][0:2])

    return order_list10

#Función para obtener el top de 10 productos menos buscados
def top10_leastWantedProducts():

    count_products = []
    order_list10 = []

    #Obtener la validation_refaund[1] para guardarla en una lista donde vienen todos los
    #productos que se vendieron
    i = 1 #columna que queremos obtener
    column_products = [fila[i] for fila in lifestore_searches]

    for products in range(len(lifestore_products)+1):
        element = column_products.count(products)
        count_products.append(element)
    #Comprobamos que imprima los correctos
    #print(str(products) + " " + str(count_products[products]))

```

```

#Convertir en un arreglo para ordenar la lista
array_countProducts = np.array(count_products)

#print(array_countProducts)

#Comprobamos que se ordenen
#commanded_frequency = np.sort(array_countProducts)[::-1]
#print(commanded_frequency)

#Ordenamos por id_producto
idProduct_order = np.argsort(array_countProducts)
#print(idProduct_order)

#Guardamos en un arreglo los primeros 10
array_idProduct = idProduct_order[0:11]

#Buscamos el id en el arreglo lifestore_products para imprimir el top 5
for column in range(len(lifestore_products)):
    for row in range(len(array_idProduct)):
        if lifestore_products[column][0] == array_idProduct[row]:
            order_list10.append(lifestore_products[column][0:2])

return order_list10

#Función para unir listas
def merge(list1, list2):
    merged_list = [(list1[i], list2[i]) for i in range(0, len(list1))]
    return merged_list

#Función para obtener el top de 5 productos más vendidos
def existingCategories():
    categories = []
    product_categories = []
    count_products = []
    validation_refund = []
    idProducto_SalesXProduct = []
    frecuency_categorie = []
    totalSalesByCategory = []

    #Guardamos en un arreglo las categorías existentes
    for product in range(len(lifestore_products)):
        if lifestore_products[product][3] not in categories:
            categories.append(lifestore_products[product][3])

    #Guardamos en una lista cada categoría de los productos en la lista de
    #lifestore_products
    for column in range(len(lifestore_products)):
        for row in range(len(categories)):
            if lifestore_products[column][3] == categories[row]:
                product_categories.append(lifestore_products[column][3])

    #Creamos una lista que imprima del 1 hasta la longitud de número de productos que hay
    # : 96
    for x in range(len(lifestore_products)+1):
        if x != 0:
            count_products.append(x)

    #Unir de listas de Id_producto y las categorías
    idProducto_category = merge(count_products, product_categories)

    #Agregamos en una nueva lista los productos de ventas que no tuvieron reembolso == 0
    for sale0 in range(len(lifestore_sales)):
        if lifestore_sales[sale0][4] == 0:
            validation_refund.append(lifestore_sales[sale0])

    #Creamos una lista para unir los id productos de lifestore_sales con la nueva lista
    #que contiene los id_producto y su
    #categoría
    for x in range(len(validation_refund)):
        for j in range(len(idProducto_category)):
            if lifestore_sales[x][1] == idProducto_category[j][0]:
                idProducto_SalesXProduct.append(idProducto_category[j][1])

```

```

#contamos las frecuencias de venta por cada categoria
for product in idProducto_SalesXProduct:
    frecuency_categorie.append(idProducto_SalesXProduct.count(product))

#Creamos una nueva lista para unir las frecuencias por categoria y su categoria
salesByCategory = merge(frecuency_categorie,idProducto_SalesXProduct)

#Ordenamos el numero de ventas por categoria de menor a mayor
leastSoldCategories = sorted(salesByCategory, key=lambda categorie : categorie[0])

for element in leastSoldCategories:
    if element not in totalSalesByCategory:
        totalSalesByCategory.append(element)

return categories, totalSalesByCategory

#Funcion para obtener el top de reviews
def top_Reviews(option):

    validation_refund = []
    resultantList = []
    product_top5 = []

    #Extraer la columna 0 y 1 de lifestore_products que corresponde al id_product y name
    #para guardarlos en una lista y comparar
    #despues de filtrar

    i = 0 #columna que queremos obtener
    column_idProduct = [fila[i] for fila in lifestore_products]

    i = 1 #columna que queremos obtener
    column_nameProduct = [fila[i] for fila in lifestore_products]

    productos_id_name = merge(column_idProduct ,column_nameProduct)

    #Agregamos en una nueva lista solamente los productos que tienen reseña, quitamos de
    #la lista todos los productos vendidos que
    #tengan score < 1

    for sale0 in range(len(lifestore_sales)):
        if lifestore_sales[sale0][2] >= 1:
            validation_refund.append(lifestore_sales[sale0])

    #Extraer la columna 1 y 2 de lifestore_sales que corresponde al id_product y score
    #para guardarlos en una lista y comparar
    #despues de filtrar

    i = 1 #columna que queremos obtener
    column_idProductSales = [fila[i] for fila in lifestore_sales]

    i = 2 #columna que queremos obtener
    column_scoreProduct = [fila[i] for fila in lifestore_sales]

    productos_id_score = merge(column_idProductSales, column_scoreProduct)

    #Ordenamos el Score
    ordenados = sorted(productos_id_score, key=lambda score : score[1], reverse = option)

    #Limpiamos los id_products y score repetidos.
    for element in ordenados:
        if element not in resultantList:
            resultantList.append(element)

    #Extraemos la columna con el Id_product de la nueva lista
    i = 0 #columna que queremos obtener
    column_idProductNew = [fila[i] for fila in resultantList]
    #Guardar solamente los primeros 5

```

```

top5 = column_idProductNew [0:5]

#Buscamos el id producto de la nueva lista, con la creada al principio con el filtrado
#de Lifestore_sales
for column in range(len(productos_id_name)):
    for row in range(len(top5)):
        if productos_id_name[column][0] == top5[row]:
            product_top5.append(productos_id_name[column][0:2])

return product_top5, resultantList

#Función para obtener el ventas totales, ventas totales sin reembolsos, total vendido,
#y el top 3 de meses con más ventas
def sales():

    validation_refund = []
    sales_total = []
    freq_month = []
    months_sales = []
    months_top3 = []
    column_dateSale = []
    column_month = []

    #Agregamos en una nueva lista los productos de ventas que no tuvieron reembolso == 0
    for sale0 in range(len(lifestore_sales)):
        if lifestore_sales[sale0][4] == 0:
            validation_refund.append(lifestore_sales[sale0])

    #Ventas totales registradas
    sales_Total_refund = len(validation_refund)

    #Convertimos la fecha a tipo datetime para extraerla
    for dateSale in range(len(validation_refund)):
        column_dateSale.append(datetime.strptime(validation_refund[dateSale][3], '%d/%m/%Y'))
        column_month.append(column_dateSale[dateSale].month)

    #Extraemos la el id producto de la nueva lista limpia
    i = 1 #columna que queremos obtener
    column_idProductSales = [fila[i] for fila in validation_refund]

    #Id_product Unir con el Mes de Venta
    idProduct_date = merge(column_idProductSales, column_month)

    #Extraemos la el id producto y el precio de lifestore_products
    i = 0 #columna que queremos obtener
    column_idProductP = [fila[i] for fila in lifestore_products]

    i = 2 #columna que queremos obtener
    column_priceP = [fila[i] for fila in lifestore_products]

    #Id_product Unir con el Precio del producto
    idProduct_Price = merge(column_idProductP, column_priceP)

    #Buscamos id_Product de sales con el Id_product de productos para hacer la suma de
    #precios y calcular la venta
    for column in range(len(idProduct_Price)):
        for row in range(len(idProduct_date)):
            if idProduct_Price[column][0] == idProduct_date[row][0]:
                sales_total.append(idProduct_Price[column][1])

    #Imprimir la suma total vendida con la unión
    salesTotal = sum(sales_total)

    #Ordenar por Mes empezando por Enero
    ordenados = sorted(column_month, key=lambda mes : mes)

    #Eliminar los meses repetidos
    for product in range(len(column_month)):
        if column_month[product] not in months_sales:

```

```

months_sales.append(column_month[product])

#Ocurrencia de los meses donde m s se compraron productos
for products in ordenados:
    element = ordenados.count(products)
    freq_month.append(element)

#Mes Un i n con frecuencia de venta
month_freq = merge(ordenados,freq_month)

#Ordenar por ocurrencia - (mes, ocurrencia)
ordenados_freq = sorted( month_freq, key=lambda freq : freq[1], reverse = True)

#Eliminar las ocurrencias con su mes repetidas
for product in range(len(ordenados_freq)):
    if ordenados_freq[product] not in months_top3:
        months_top3.append(ordenados_freq[product])

return sales_Total_refund, salesTotal, months_top3[0:3]

```