



PROYECTO 1

GitHub

Desarrollado por:

Diana Yesenia Campos Tenorio

Proyecto:

Gerencia de Ventas de LifeStore

Curso:

Fundamentos de Programación con Python
Emerging Technologies Institute

Fecha de entrega:

13/02/2022

Contenido

1	Objetivo	1
2	Introducción	1
3	Solución al problema	1
	3.1 Productos más vendidos y productos rezagados	1
	3.2 Productos por reseña en el servicio	2
	3.3 Total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año	2
4	Definición del código	3
	4.1 Login de usuario	3
	4.2 Productos más vendidos y productos rezagados	5
	4.3 Productos por reseña en el servicio	7
	4.4 Total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año	9
5	Conclusiones	11

1 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.

2 Introducción

- LifeStore es una tienda virtual que maneja una amplia gama de artículos, recientemente, la Gerencia de ventas, se percató que la empresa tiene una importante acumulación de inventario. Asimismo, se ha identificado una reducción en las búsquedas de un grupo importante de productos, lo que ha redundado en una disminución sustancial de sus ventas del último trimestre.

3 Solución al problema

3.1 Productos más vendidos y productos rezagados

Para dar solución a este problema lo primero que se realizó fue declarar diferentes variables en donde se iba a guardar la lista de productos vendidos, lista de productos buscados y un contador para cada uno de ellos. Posteriormente se van creando diferentes listas en donde se guardará información respecto a la cantidad de veces vendidas de cada producto y otra lista en donde se guarde la cantidad de veces que se buscan cada producto, para ambas listas se utilizan dos ciclos *for*, uno anidado en otro, en donde el primer ciclo recorre cada espacio de la lista y el segundo va integrando las características de cada uno. Para finalizar, se agregan funciones las cuales ordenan las listas de productos vendidos y productos buscados según lo deseado, lo cual para nuestro caso serían los siguientes:

- Los productos mas vendido
- Los productos menos vendidos
- Los productos mas buscados
- Los productos menos buscados

3.2 Productos por reseña en el servicio

Durante el desarrollo de la funcionalidad de búsqueda por reseñas se necesito definir las variables que funcionan como contadores y listas vacías donde se ira integrando la información respecto a las reseñas. Para obtener los productos mejor y peor calificados se tomaron en cuenta los productos vendidos de la lista de Sales donde se tomaron las calificaciones que tenían asignadas y se genero un diccionario con los mejores calificados. Si bien se pudo sacar estos datos como en el problema anterior se decidio usar diccionarios para obtener estos resultados.

3.3 Total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año

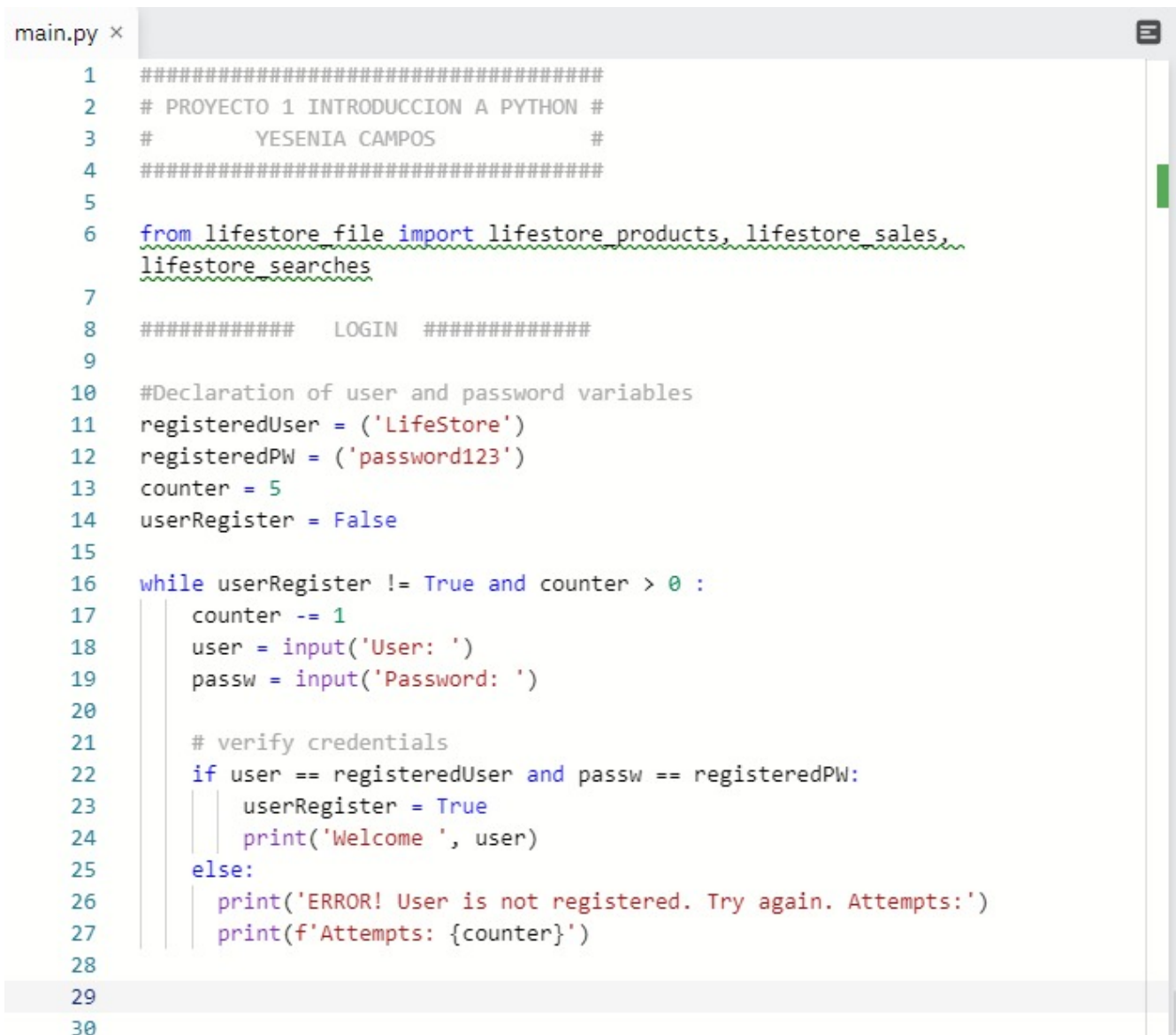
De productos a retirar del mercado así como sugerencia de cómo reducir la acumulación de inventario considerando los datos de ingresos y ventas mensuales.

En la solución del presente apartado también se utilizaron diccionarios para obtener los totales, primero obtuvimos una lista con los productos vendidos, después los ingresos por cada producto y de estos realizamos una suma. Posteriormente se pudo obtener los ingresos por mes y por año generando una lista de ventas ayudándonos de las fechas. Al final solo se obtuvo un total por año y el total por mes.

4 Definición del código

4.1 Login de usuario

En primer lugar, en la figura 4.1 se muestra el código implementado para realizar el login del usuario.



```
main.py x
1 #####
2 # PROYECTO 1 INTRODUCCION A PYTHON #
3 #      YESENIA CAMPOS      #
4 #####
5
6 from lifestore file import lifestore_products, lifestore_sales,
   lifestore_searches
7
8 ##### LOGIN #####
9
10 #Declaration of user and password variables
11 registeredUser = ('LifeStore')
12 registeredPW = ('password123')
13 counter = 5
14 userRegister = False
15
16 while userRegister != True and counter > 0 :
17     counter -= 1
18     user = input('User: ')
19     passw = input('Password: ')
20
21     # verify credentials
22     if user == registeredUser and passw == registeredPW:
23         userRegister = True
24         print('Welcome ', user)
25     else:
26         print('ERROR! User is not registered. Try again. Attempts:')
27         print(f'Attempts: {counter}')
28
29
30
```

Figura 4.1: Código de Login

Una vez corriendo el código, la terminal se abre y te solicita que se ingrese un usuario y una contraseña para acceder a las siguientes opciones del programa, esto se puede observar en la figura 4.2 en donde se

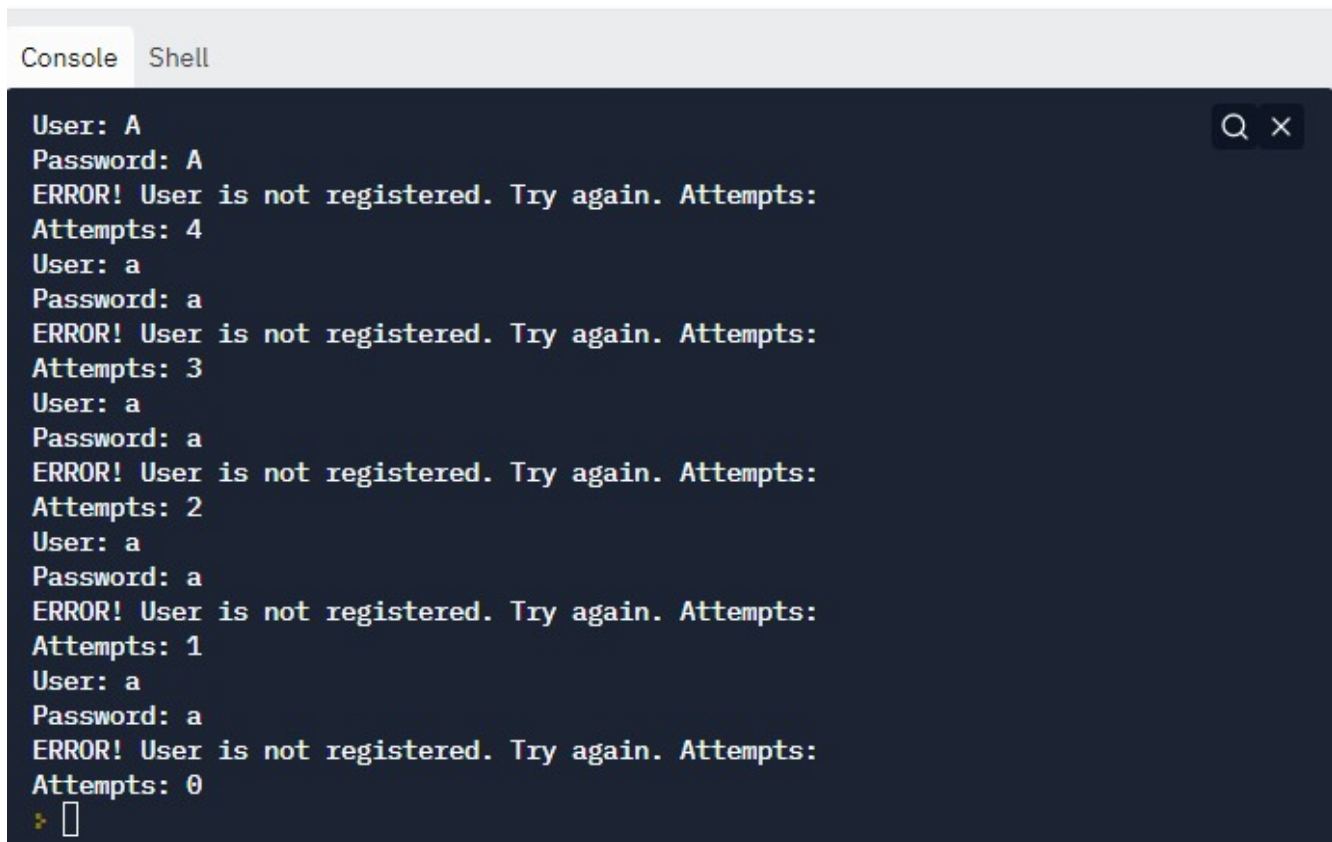
ingresan las credenciales correctas.



```
Console Shell
User: LifeStore
Password: password123
Welcome LifeStore
> 
```

Figura 4.2: Funcionamiento del login

Por otro lado, en caso de ingresar de forma incorrecta el usuario y/o contraseña se muestra un mensaje de error y un contador donde se observa la cantidad de veces que se intento ingresar al programa pero fallo, tal como se muestra en la figura 4.3.



```
Console Shell
User: A
Password: A
ERROR! User is not registered. Try again. Attempts:
Attempts: 4
User: a
Password: a
ERROR! User is not registered. Try again. Attempts:
Attempts: 3
User: a
Password: a
ERROR! User is not registered. Try again. Attempts:
Attempts: 2
User: a
Password: a
ERROR! User is not registered. Try again. Attempts:
Attempts: 1
User: a
Password: a
ERROR! User is not registered. Try again. Attempts:
Attempts: 0
> 
```

Figura 4.3: Intentos de ingreso al programa

4.2 Productos más vendidos y productos rezagados

Para la sección de productos mas vendidos y productos rezagados, el código desarrollado se muestra a continuación en la figura 4.4.

```
main.py ×
30
31
32 ##### Top sellers and lagging products #####
33
34 #declaration of the variable products sold
35 productSold = [sale[1] for sale in lifestore_sales]
36 productSearches = [search[1] for search in lifestore_searches]
37 counterSold = []
38 counterSearch = []
39 #create list with the number of times sold
40 for sales in lifestore_products:
41     listProducts = []
42     counterSold.append(listProducts)
43     for i in range(1):
44         listProducts.append(sales[0]) #ID product
45         listProducts.append(sales[1]) #Name product
46         listProducts.append(sales[-2]) #Category product
47         listProducts.append(productSold.count(sales[0])) #Times Sold
48
49 #Function that sorts the sellers
50 def greaterSales(counterSold):
51     counterSold.sort(key = lambda x: x[3])
52     return counterSold
53
54 counterSold = greaterSales(counterSold)
55
56 #print 5 most and least sold products
57 print('#### The most sales products: ####')
58 for j in [-1,-2,-3,-4,-5]:
59     print(f'\t Id: {counterSold[j][0]} Name: {counterSold[j][1]} Sales: {counterSold[j][3]}')
60
61 print('#### The least sales products: ####')
62 for k in range(0,5):
63     print(f'\t Id: {counterSold[k][0]} Name: {counterSold[k][1]} Sales: {counterSold[k][3]}')
64
65 #create list with the number of times search
66 for search in lifestore_products:
67     listProducts2 = []
68     counterSearch.append(listProducts2)
69     for i in range(1):
70         listProducts2.append(search[0]) #ID product
71         listProducts2.append(search[1]) #Name product
72         listProducts2.append(search[-2]) #Category product
73         listProducts2.append(productSearches.count(search[0])) #Times Search
74
75 counterSearch = greaterSales(counterSearch)
76 #print 10 most and least searched products
77 print('#### The most searched products: ####')
78 for j in [-1,-2,-3,-4,-5,-6,-7,-8,-9,-10]:
79     print(f'\t Id: {counterSearch[j][0]} Name: {counterSearch[j][1]} Searched: {counterSearch[j][3]}')
80
81 print('#### The least searched products: ####')
82 for k in range(0,10):
83     print(f'\t Id: {counterSearch[k][0]} Name: {counterSearch[k][1]} Searched: {counterSearch[k][3]}')
```

Figura 4.4: Código de venta de productos

4. DEFINICIÓN DEL CÓDIGO

Como se puede observar en la figura 4.5, el programa despliegan diferentes listas en donde se seleccionan los productos con ciertas características tales como:

- Los productos mas vendido
- Los productos con menos vendidos
- Los productos mas buscados
- Los productos menos buscados

```
Console Shell

#### The most sales products: ####
Id: 54 Name: SSD Kingston A800, 120GB, SATA III, 2.5'', 7mm Sales: 50
Id: 3 Name: Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth Sales: 42
Id: 5 Name: Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) Sales: 20
Id: 42 Name: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Sales: 18
Id: 57 Name: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm Sales: 15
#### The least sales products: ####
Id: 9 Name: Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generación - Coffee Lake) Sales: 0
Id: 14 Name: Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3, PCI Express 2.0 Sales: 0
Id: 15 Name: Tarjeta de Video EVGA NVIDIA GeForce GTX 1660 Ti SC Ultra Gaming, 6GB 192-bit GDDR6, PCI 3.0 Sales: 0
Id: 16 Name: Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming, 6GB 192-bit GDDR6, PCI Express 3.0 Sales: 0
Id: 19 Name: Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile, 4GB 128-bit GDDR5, PCI Express 3.0 x16 Sales: 0
#### The most searched products: ####
Id: 54 Name: SSD Kingston A800, 120GB, SATA III, 2.5'', 7mm Searched: 263
Id: 57 Name: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm Searched: 187
Id: 29 Name: Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Searched: 60
Id: 3 Name: Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth Searched: 55
Id: 4 Name: Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire Searched: 41
Id: 85 Name: Logitech Audifonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul Searched: 35
Id: 67 Name: TV Monitor LED 24TL520S-PJ 24, HD, Widescreen, HDMI, Negro Searched: 32
Id: 7 Name: Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) Searched: 31
Id: 47 Name: SSD XPG SX8200 Pro, 256GB, PCI Express, M.2 Searched: 30
Id: 5 Name: Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) Searched: 30
#### The least searched products: ####
Id: 14 Name: Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3, PCI Express 2.0 Searched: 0
Id: 16 Name: Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming, 6GB 192-bit GDDR6, PCI Express 3.0 Searched: 0
Id: 19 Name: Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile, 4GB 128-bit GDDR5, PCI Express 3.0 x16 Searched: 0
Id: 20 Name: Tarjeta de Video Gigabyte NVIDIA GeForce RTX 2060 SUPER WINDFORCE OC, 8 GB 256 bit GDDR6, PCI Express x16 3.0 Searched: 0
Id: 23 Name: Tarjeta de Video MSI Radeon X1550, 128MB 64 bit GDDR2, PCI Express x16 Searched: 0
Id: 24 Name: Tarjeta de Video PHX NVIDIA GeForce RTX 2080, 8GB 256-bit GDDR6, PCI Express 3.0 Searched: 0
Id: 30 Name: Tarjeta Madre ASUS ATX Z390 ELITE, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel Searched: 0
Id: 32 Name: Tarjeta Madre ASRock Z390 Phantom Gaming 4, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel Searched: 0
Id: 33 Name: Tarjeta Madre ASUS ATX PRIME Z390-A, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel Searched: 0
Id: 34 Name: Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING WI-FI, S-AM4, AMD B550, HDMI, max. 128GB DDR4 para AMD Searched: 0
```

Figura 4.5: Consola de venta de productos

4.3 Productos por reseña en el servicio

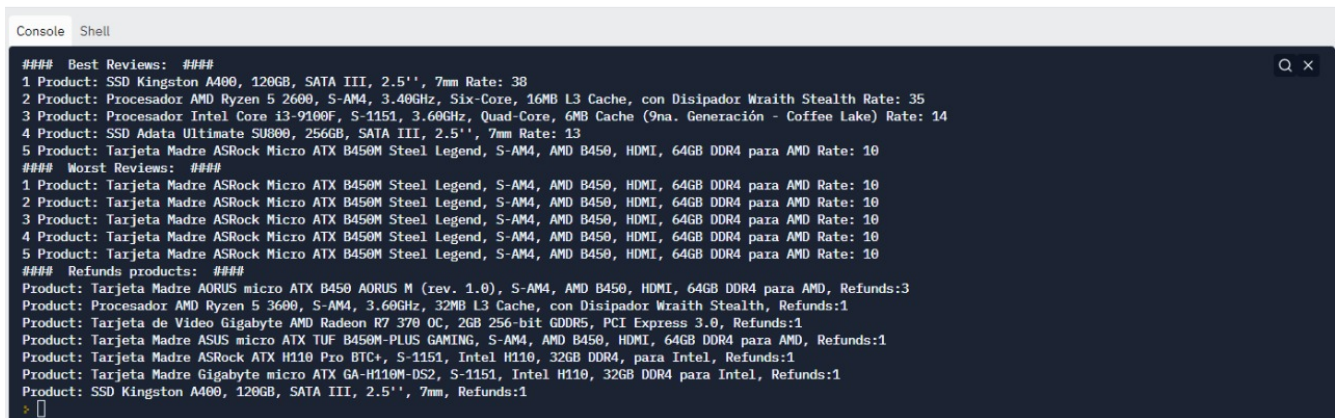
Para la sección de productos reseña en el servicio, el código desarrollado se muestra a continuación en la figura 4.6.

```
86 ##### Products per service review #####
87
88 #Generating list of products
89 bestRated = {}
90 for idSale,idProduct,rate,date,refund in lifestore_sales:
91     if rate not in bestRated.keys():
92         bestRated[rate] = [idProduct]
93     else: bestRated[rate].append(idProduct)
94
95 #To each scoreRate, sort idProducts considering rates given and refund
96 topBestRated = {}
97 for rate, producstList in bestRated.items():
98     topBestRated[rate] = []
99     for idProduct in range(len(lifestore_products)):
100         if idProduct in producstList:
101             topBestRated[rate].append((idProduct,producstList.count(idProduct)))
102         else: continue
103
104 #Sorting using lambda function
105 for rate,producstList in topBestRated.items():
106     topBestRated[rate] = sorted(producstList, key=lambda x: x[1], reverse=True)
107
108 #Verifying which products were sold and then returned to the store
109 refunds = []
110 totalRefunds = {}
111
112 for idSale,idProduct,rate,date,refund in lifestore_sales:
113     if refund == 1: refunds.append(lifestore_products[idProduct-1][1])
114 for product in refunds:
115     if product not in totalRefunds.keys(): totalRefunds[product] = refunds.count(product)
116     else: continue
117
118 print("#### Best Reviews: ####")
119 count = 1
120 for idProduct,numRates in topBestRated[5]:
121     print(count,"Product:",lifestore_products[idProduct-1][1], "Rate:",numRates)
122     count+=1
123     if count > 5: break
124
125 print("#### Worst Reviews: ####")
126 countW = 1
127 for rates,lists in list(topBestRated.items())[:-1]:
128     for id_product,num_rates in lists:
129         print(countW,"Product:",lifestore_products[idProduct-1][1], "Rate:",numRates)
130         countW+=1
131         if countW > 1: break
132
133 #Getting list of products returned
134 productsRefunded = sorted(totalRefunds.items(), key=lambda x: x[1], reverse=True)
135
136 print("#### Refunds products: ####")
137 for productNames,numRefunds in productsRefunded:
138     print(f'Product: {productNames}, Refunds:{numRefunds}')
139
140
```

Figura 4.6: Código de búsqueda por reseñas

4. DEFINICIÓN DEL CÓDIGO

Como se puede observar en la figura 4.7, el programa despliega diferentes lista en donde se seleccionan los productos con mejores y peores reseñas.



```
Console  Shell

#### Best Reviews: ####
1 Product: SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm Rate: 38
2 Product: Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth Rate: 35
3 Product: Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) Rate: 14
4 Product: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm Rate: 13
5 Product: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Rate: 10

#### Worst Reviews: ####
1 Product: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Rate: 10
2 Product: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Rate: 10
3 Product: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Rate: 10
4 Product: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Rate: 10
5 Product: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD Rate: 10

#### Refunds products: ####
Product: Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD, Refunds:3
Product: Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth, Refunds:1
Product: Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0, Refunds:1
Product: Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD, Refunds:1
Product: Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel, Refunds:1
Product: Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel, Refunds:1
Product: SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm, Refunds:1
>
```

Figura 4.7: Resultado de búsqueda por reseñas

4.4 Total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año

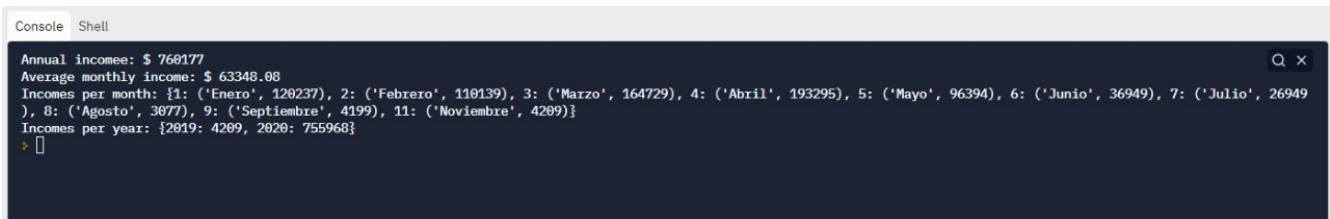
Para la sección de total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año, el código desarrollado se muestra a continuación en la figura 4.8.

```
main.py x
154
155 ### Total income and average monthly sales,annual total and months with more sales per year ####
156
157 #Getting just id of products sold
158 import datetime # To mmodify dates
159 totalsales = []
160
161 for index in range(len(lifstore_sales)):
162     totalsales.append(lifstore_sales[index][1])
163
164 #Getting incomes per each product
165 incomes = []
166
167 for product in totalsales:
168     incomes.append(lifstore_products[product-1][2])
169
170
171 #Total income
172 totalIncome = sum(incomes)
173
174
175 #Incomes per months and year, generatin list of sales by date
176 salesDate = []
177 for id_sales, idProduct, rate, date, refund in lifstore_sales:
178     newDate = datetime.datetime.strptime(date, '%d/%m/%Y').date()
179     salesDate.append((idProduct, newDate))
180
181
182 #Generating lists of months per year
183 month2019 = []
184 month2020 = []
185 for idProduct, date in salesDate:
186     if date.year == 2020:
187         month2020.append(date.month)
188     elif date.year == 2019:
189         month2019.append(date.month)
190
191 #Generatin a dict of dicts, to each year, show the quantity of sales by months registered
192 monthlySales = {}
193 monthlySales[2019] = {}
194 monthlySales[2020] = {}
195 for month in range(1, 13):
196     if month in month2019:
197         monthlySales[2019][month] = month2019.count(month)
198     elif month in month2020:
199         monthlySales[2020][month] = month2020.count(month)
200
201 monthNames = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
202               "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"]
203
204 #Total incomes per year
205 incomesPerYear = {}
206 for year in monthlySales.keys():
207     yearIncome = 0
208     for months in monthlySales[year].keys():
209         for dates, price in zip(salesDate, incomes):
210             if dates[1].year == year:
211                 if dates[1].month == months:
212                     yearIncome += price
213     incomesPerYear[year] = yearIncome
214
215 #Total incomes per month
216 incomesPerMont = {}
217 for months in list(monthlySales[2020].keys())+list(monthlySales[2019].keys()):
218     countIncome = 0
219     for dates, price in zip(salesDate, incomes):
220         if dates[1].month == months:
221             countIncome += price
222     else:
223         continue
224     incomesPerMont[months] = (monthNames[months-1], countIncome)
225
226 # Sort months by the income generated
227 topMonthIncomes = sorted(list(incomesPerMont.items()),key=lambda x: x[1][1], reverse=True)
228 print('Annual income: $', totalIncome)
229 print('Average monthly income: $', '%.2f' % (totalIncome/12))
230 print('Incomes per month:',incomesPerMont)
231 print('Incomes per year:',incomesPerYear)
```

Figura 4.8: Código de ingresos y ventas promedio

4. DEFINICIÓN DEL CÓDIGO

Como se puede observar en la figura 4.9, el programa despliegan los ingresos y ventas promedios por año y por cada mes.



```
Console Shell
Annual income: $ 768177
Average monthly income: $ 63348.08
Incomes per month: {1: ('Enero', 128237), 2: ('Febrero', 118139), 3: ('Marzo', 164729), 4: ('Abril', 193295), 5: ('Mayo', 96394), 6: ('Junio', 36949), 7: ('Julio', 26949), 8: ('Agosto', 3877), 9: ('Septiembre', 4199), 11: ('Noviembre', 4269)}
Incomes per year: {2019: 4269, 2020: 755968}
```

Figura 4.9: Resultado ingresos y ventas promedio

5 Conclusiones

En lo personal considero que la aplicación de métodos que permitan manejar información es muy relevante para cualquier persona que desea tomar alguna decisión respecto a compra o venta de productos ya que nos da una mejor perspectiva de aquellos factores claves a considerar tales como tendencias de venta, reseñas o precios con los cuales tomar la decisión mas acertada a nuestras necesidades.

Durante el desarrollo del proyecto me percate que la ciencia de datos es muy útil y que favorecen a todas las personas día con día. En la actualidad, es común escuchar que previo a la compra de algún producto se tomen en consideración las reseñas, así como tener filtros para definir rangos sobre precios, marcas, tipo de productos, entre otros.