LifeStore Sales Management

EmTech

Python Final Project

Data Analysis Report

Author:
Alan Garcia

Supervisor:
Jaime Alonso

February 14, 2022
https://github.com/Aland135/LifeStore-Sales-Analysis

EMTECH
Emerging Technologies Institute

Abstract

LifeStore's Sales Management has provided a file that contains data about their sales. The company wishes to analyze their sales in order to find out how to improve their stock problems, as well as to know which of their products get sold the most. With Python a program has been made to analyze their data and obtain the answers to their questions. Based on the results obtained, advice is given in order to fix LifeStore's stock issues.

Contents

Chapter 1

Introduction

1.1 LifeStore

LifeStore is a company dedicated to sell technology items, such as hard disk drives, headphones, USB devices etc. The company has recently seen issues related to their sales. One of this problems is the accumulation of stock of many products, and another problem related to the previous one is the fact that many of their products are not being searched in their site. Due to these issues, the company has requested an analysis of their data that provides:

- Most and least searched products, as well as most and least searched products.
- Products by score.
- A suggestion to fix their issues.

These requests will be accomplished via a program that analyzes the data provided by the company and provides the results they require.

1.2 Data Analysis

EmTech has provided training on programming with the Python language to many students and professionals. Now EmTech requests their students to provide a solution to LifeStore with the skills they have taught. The author of this document hopes it will fulfill the demands of both EmTech and Lifestore. For this, a program has been made making use of loops, lists, dictionaries and many other programming tools.

Chapter 2

Code Breakdown

2.1 User login

For the user login, a username as well as a password are required in order to access the program. An arbitrary username and an arbitrary password have been chosen by the author of this program, but it can be edited if requested by Lifestore's Sales Management. The chosen username is "Lifestore" and the password is "store15". The code section responsible for the login process is shown in Code 1

```
userAccess = False
attempts = 0

#Welcome message
print("Welcome to the data analysis program!\nPlease log in with your username and password\n")

#loop for log in attempts
while userAccess == False and attempts < 3:
  user = input("Please write your username here: ")
  password = input("Please write your password here: ")
  attempts += 1
  if user == "Lifestore" and password == "store15":
    userAccess = True
  elif attempts == 3:
    print("You have reached the maximum number of attempts. The program will now close.")
    exit()
  elif user != "Lifestore":
    print("This user is not registered in the system. Please try again (attempt " + str(attempts) + "/3)")
  elif password != "store15":
    print("The password is incorrect. Please try again(attempt " + str(attempts) + "/3)")

print("\n\nWelcome, ", user, "\n\n")
```

Code 1: Login code

A while loop was used in which the condition was only met when both the username and the password were provided correctly. The user has only 3 attempts to get the answer correctly, this is to avoid any undesired infinite loops.

2.2 Most and least sold products

Code 2 shows the code used to obtain the 5 most sold products by LifeStore.

```
#Generate a list with the product id appearing as many times as it was sold without taking into account the refunds
sales_per_product = [sales[1] for sales in lifestore_sales if sales[4] != 1]

#Generate a list with the product id appearing as many times as it was searched
search_per_product =[search[1] for search in lifestore_searches]

#Generate a list with 4 values per item: product id, sales per product, searches per product and category
product_sales_searches = [[product[0], sales_per_product.count(product[0]), search_per_product.count(product[0]), product[3]] for product in lifestore_products]

#Defines a function necessary to sort product_sales_searches by sales
def function_sales(e):
  return e[1]

#Sort our list by sales, with the most sold products on the top
product_sales_searches.sort(reverse = True, key = function_sales)

#Displays the 5 most sold products
print("The most sold products were: ")
for q in range(5):
  print(f"ID number: {product_sales_searches[q][0]} // Name: {lifestore_products[product_sales_searches[q][0] - 1 ][1]} // Sales: {product_sales_searches[q][1]}.")
print("\n\n")
```

Code 2: Most sold products

A list of lists called "product_sales_searches" was made were each list contained 3 values: the product id, the sales of that product and the searches of that product. With this list, the task of organizing the values was made easy by using the sort function and to print each value a for loop was used. It is important to note that the refunds were not taken into account as sales, since no gain was obtained from them.

The least sold products were provided by category. The code used to display the 5 least sold products by category is shown in Code 3.

```
#Sort our list by sales, with the least sold products on the top
product_sales_searches.sort(reverse = False, key = function_sales)

#Displays the 5 least sold products by category
print("The least sold products by category were: \n")
i = 0
for item in categories:
  print("\n" + categories[i] + ": ")
  i += 1
  q1 = 0
  q2 = 0
  for q in product_sales_searches:
    if item == product_sales_searches[q1][3] and q2 < 5:
      print(f"ID number: {product_sales_searches[q1][0]} // Name: {lifestore_products[product_sales_searches[q1][0] - 1 ][1]} // Sales: {product_sales_searches[q1][1]}.")
      q2 += 1
    q1 += 1
print("\n\n")
```

Code 3: Least sold products by category

Two loops were used, to associate the id number with the category in the Lifestore_sales list and then to print the product by category and by number of sales.

2.3 Most and least searched products

The five most searched products in the Lifestore site are obtained by the code shown in Code 4.

```
#Sort our list by searches, with the most searched products on the top
product_sales_searches.sort(reverse = True, key = function_searches)

#Displays the 10 most searched products
print("The most searched products were: ")
for q in range(10):
  print(f"ID number: {product_sales_searches[q][0]} // Name: {lifestore_products[product_sales_searches[q][0]
  ↪  - 1 ][1]} // Searches: {product_sales_searches[q][2]}.")
print("\n\n")
```

Code 4: Most searched products

Again we make use of the "product_sales_searches" list. The list is sorted by searches and a for loop is used to print the values.

The least searched products by category are obtained by Code 5.

```
#Sort our list by searches, with the least searched products on the top
product_sales_searches.sort(reverse = False, key = function_searches)

#Displays the 10 least searched products by category
print("The least searched products by category were: \n")
i = 0
for item in categories:
  print("\n" + categories[i] + ": ")
  i += 1
  q1 = 0
  q2 = 0
  for q in product_sales_searches :
    if item == product_sales_searches[q1][3] and q2 < 10 :
      print(f"ID number: {product_sales_searches[q1][0]} // Name:
      ↪  {lifestore_products[product_sales_searches[q1][0] - 1 ][1]} // Searches:
      ↪  {product_sales_searches[q1][2]}.")
      q2 += 1
    q1 += 1
print("\n\n")
```

Code 5: Least searched products by category

The process is similar as the one used for the 5 least sold products by category. Only a few parameters needed to be changed in order to use the same loops to obtain this new result.

2.4 Best and worst scored products

Code 6 shows the solution found to obtain the best rated products.

```
#Make a dictionary to save the scores as lists for each product
id_reviews_count = {}

#Add values to each element in the dictionary and add score value lists for each element
for par in id_reviews_not_separated:
    id = par[0]
    review = par[1]
    if id not in id_reviews_count.keys():
        id_reviews_count[id] = []
    id_reviews_count[id].append(review)

#Create a list containing only the product ids of the products with a score
id_with_score = []
[id_with_score.append(idprdct) for idprdct in id_reviews_count if idprdct not in id_with_score]

#Create a list to save the average values of each product
averages = []

#Get the average score value of each product and put it into the averages list
for id_product in id_reviews_count.keys():
    lista_reviews = id_reviews_count[id_product]
    promedio = sum(lista_reviews) / len(lista_reviews)
    # Set the decimals to 2
    decimales = 2
    multiplicador = 10 ** decimales
    promedio = math.ceil(promedio * multiplicador) / multiplicador
    averages.append(promedio)

#Join the id_with_score list with the averages list
id_avgscore = list(zip(id_with_score,averages))
```

Code 6: Products by score

Once again a list is made which contains the important values for this section: the id of the product and the average score of that product. For this, a dictionary in which the keys are the product id numbers is made, and the score values are stored into each key. Then, an average is obtained for the scores of each key, and these values are saved into a list. Then, a list containing the id numbers of the scored products and the list containing the average scores are joined into a single list. This final list, it is quite easy to sort by score and then print the values with a simple for loop.

2.5 Income and sales

The same process of making a list that contains the important parameters is being used again here. The process is shown in Code 7.

```
# Creates a list that contains the sale and the date of each sale
id_date = [ [sale[0], sale[3]] for sale in lifestore_sales if sale[4] == 0 ]

# Creates a dictionary that contains the months and the sales per month
category_months = {}

#Create a list of months that had any sales in them
months_with_sales = []
```

```
9
10  for par in id_date:
11      # Obtains sale and month
12      id = par[0]
13      _, month, _ = par[1].split('/')
14      # Makes each month into a key, if it isn't one already
15      if month not in category_months.keys():
16          category_months[month] = []
17      #Include into months_with_sales every month that had any sales in it (no refunds)
18      if month not in months_with_sales:
19          months_with_sales.append(month)
20      #Appends each sale to the month it belongs to
21      category_months[month].append(id)
22
23  #Creates a list that contains the sum of money earned each month, without mentioning the month
24  money_per_month = []
25
26  #Creates a list that contains the amount of sales each month
27  sales_per_month = []
28
29  for key in category_months.keys():
30      list_month = category_months[key]
31      sum_sale = 0
32      for id_sale in list_month:
33          index = id_sale - 1
34          info_sale = lifestore_sales[index]
35          id_product = info_sale[1]
36          price = lifestore_products[id_product-1][2]
37          sum_sale += price
38      #Append the sum money earned every month
39      money_per_month.append(sum_sale)
40      #Append the number of sales of each month
41      sales_per_month.append(len(list_month))
42
43  #Join the months_with_sales, money_per_month and sales_per_month lists into a single list
44  month_money_sale = list(zip(months_with_sales,money_per_month, sales_per_month))
```

Code 7: Incomes and sales

A dictionary is made which contains each month that had sales in it as a key, and the sale id numbers that belong to each month are appended to each key. Three lists are made: a list that contains the months that had sales in them, a list that contains the total money obtained each month, and a list that contains the number of sales of each month. Then these lists are joined together and from this list of lists it is simple to sort by money and by sales and to display the result with a for loop.

Chapter 3

Results

3.1 Most sold and searched products

The 5 most sold products by LifeStore, as well as their number of sales, are shown in Table 3.1.

Table 3.1: Most sold products.

ID number: 54	Name: SSD Kingston A400	Sales: 49
ID number: 3	Name: Procesador AMD Ryzen 5 2600	Sales: 42
ID number: 5	Name: Procesador Intel Core i3-9100F	Sales: 20
ID number: 42	Name: Tarjeta Madre ASRock Micro ATX B450M Steel Legend	Sales: 18
ID number: 57	Name: SSD Adata Ultimate SU800	Sales: 15

The 10 most searched products in the LifeStore site are shown in Table 3.2

Table 3.2: Most searched products.

ID number: 54	Name: SSD Kingston A400	Searches: 263.
ID number: 57	Name: SSD Adata Ultimate SU800	Searches: 107.
ID number: 29	Name: Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING	Searches: 60.
ID number: 3	Name: Procesador AMD Ryzen 5 2600	Searches: 55.
ID number: 4	Name: Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8	Searches: 41.
ID number: 85	Name: Logitech Audífonos Gamer G635 7.1	Searches: 35.
ID number: 67	Name: TV Monitor LED 24TL520S-PU 24	Searches: 32.
ID number: 7	Name: Procesador Intel Core i7-9700K	Searches: 31.
ID number: 47	Name: SSD XPG SX8200 Pro	Searches: 30.
ID number: 5	Name: Procesador Intel Core i3-9100F	Searches: 30.

3.2 By category, least sold and searched products

The 5 least sold products by category are shown in Table 3.3.

Table 3.3: Least sold products by category.

audifonos		
ID number: 86	Name: ASUS Audífonos Gamer ROG Theta 7.1	Sales: 0.
ID number: 87	Name: Acer Audífonos Gamer Galea 300	Sales: 0.
ID number: 88	Name: Audífonos Gamer Balam Rush Orphix RGB 7.1	Sales: 0.
ID number: 90	Name: Energy Sistem Audífonos con Micrófono Headphones 1	Sales: 0.
ID number: 91	Name: Genius GHP-400S Audífonos	Sales: 0.
bocinas		
ID number: 75	Name: Lenovo Barra de Sonido	Sales: 0.
ID number: 76	Name: Acteck Bocina con Subwoofer AXF-290	Sales: 0.
ID number: 77	Name: Verbatim Bocina Portátil Mini	Sales: 0.
ID number: 78	Name: Ghia Bocina Portátil BX300	Sales: 0.
ID number: 79	Name: Naceb Bocina Portátil NA-0301	Sales: 0.
discos duros		
ID number: 53	Name: SSD Addlink Technology S70	Sales: 0.
ID number: 55	Name: SSD para Servidor Supermicro SSD-DM128-SMCMVN1	Sales: 0.
ID number: 56	Name: SSD para Servidor Lenovo Thinksystem S4500	Sales: 0.
ID number: 58	Name: SSD para Servidor Lenovo Thinksystem S4510	Sales: 0.
ID number: 59	Name: SSD Samsung 860 EVO	Sales: 0.
memorias usb		
ID number: 61	Name: Kit Memoria RAM Corsair Vengeance LPX DDR4	Sales: 0.
ID number: 60	Name: Kit Memoria RAM Corsair Dominator Platinum DDR4	Sales: 1.
pantallas		
ID number: 62	Name: Makena Smart TV LED 32S2 32”	Sales: 0.
ID number: 63	Name: Seiki TV LED SC-39HS950N 38.5	Sales: 0.
ID number: 64	Name: Samsung TV LED LH43QMREBGCXGO 43	Sales: 0.
ID number: 65	Name: Samsung Smart TV LED UN70RU7100FXZX 70	Sales: 0.
ID number: 68	Name: Makena Smart TV LED 40S2 40”	Sales: 0.
procesadores		
ID number: 9	Name: Procesador Intel Core i3-8100	Sales: 0.
ID number: 1	Name: Procesador AMD Ryzen 3 3300X S-AM4	Sales: 2.
ID number: 6	Name: Procesador Intel Core i9-9900K	Sales: 3.
ID number: 8	Name: Procesador Intel Core i5-9600K	Sales: 4.
ID number: 7	Name: Procesador Intel Core i7-9700K	Sales: 7.
tarjetas de video		
ID number: 14	Name: Tarjeta de Video EVGA NVIDIA GeForce GT 710	Sales: 0.
ID number: 15	Name: Tarjeta de Video EVGA NVIDIA GeForce GTX 1660 Ti SC Ultra Gaming	Sales: 0.
ID number: 16	Name: Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming	Sales: 0.
ID number: 17	Name: Tarjeta de Video Gigabyte AMD Radeon R7 370 OC	Sales: 0.
ID number: 19	Name: Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile	Sales: 0.
tarjetas madre		
ID number: 30	Name: Tarjeta Madre AORUS ATX Z390 ELITE	Sales: 0.
ID number: 32	Name: Tarjeta Madre ASRock Z390 Phantom Gaming 4	Sales: 0.
ID number: 34	Name: Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING WI-FI	Sales: 0.
ID number: 35	Name: Tarjeta Madre Gigabyte micro ATX Z390 M GAMING	Sales: 0.
ID number: 36	Name: Tarjeta Madre Gigabyte micro ATX Z490M GAMING X (rev. 1.0)	Sales: 0.

The 10 least searched products by category are shown in Table 3.4.

Table 3.4: Least searched products by category .

audifonos		
ID number: 86	Name: ASUS Audífonos Gamer ROG Theta 7.1	Searches: 0.

ID number: 87	Name: Acer Audífonos Gamer Galea 300	Searches: 0.
ID number: 88	Name: Audífonos Gamer Balam Rush Orphix RGB 7.1	Searches: 0.
ID number: 90	Name: Energy Sistem Audífonos con Micrófono Headphones 1	Searches: 0.
ID number: 92	Name: Getttech Audífonos con Micrófono Sonority	Searches: 0.
ID number: 96	Name: Klip Xtreme Audífonos Blast	Searches: 0.
ID number: 93	Name: Ginga Audífonos con Micrófono GI18ADJ01BT-RO	Searches: 1.
ID number: 91	Name: Genius GHP-400S Audífonos	Searches: 2.
ID number: 95	Name: Iogear Audífonos Gamer GHG601	Searches: 3.
ID number: 94	Name: HyperX Audífonos Gamer Cloud Flight para PC	Searches: 6.
bocinas		
ID number: 75	Name: Lenovo Barra de Sonido	Searches: 0.
ID number: 77	Name: Verbatim Bocina Portátil Mini	Searches: 0.
ID number: 78	Name: Ghia Bocina Portátil BX300	Searches: 0.
ID number: 79	Name: Naceb Bocina Portátil NA-0301	Searches: 0.
ID number: 81	Name: Ghia Bocina Portátil BX900	Searches: 0.
ID number: 82	Name: Ghia Bocina Portátil BX400	Searches: 0.
ID number: 83	Name: Ghia Bocina Portátil BX500	Searches: 0.
ID number: 80	Name: Ghia Bocina Portátil BX800	Searches: 1.
ID number: 76	Name: Acteck Bocina con Subwoofer AXF-290	Searches: 2.
ID number: 74	Name: Logitech Bocinas para Computadora con Subwoofer G560	Searches: 6.
discos duros		
ID number: 53	Name: SSD Addlink Technology S70	Searches: 0.
ID number: 55	Name: SSD para Servidor Supermicro SSD-DM128-SMCMVN1	Searches: 0.
ID number: 58	Name: SSD para Servidor Lenovo Thinksystem S4510	Searches: 0.
ID number: 59	Name: SSD Samsung 860 EVO	Searches: 1.
ID number: 56	Name: SSD para Servidor Lenovo Thinksystem S4500	Searches: 2.
ID number: 52	Name: SSD Western Digital WD Blue 3D NAND	Searches: 5.
ID number: 50	Name: SSD Crucial MX500	Searches: 7.
ID number: 49	Name: Kit SSD Kingston KC600	Searches: 10.
ID number: 51	Name: SSD Kingston UV500	Searches: 11.
ID number: 48	Name: SSD Kingston A2000 NVMe	Searches: 27.
memorias usb		
ID number: 61	Name: Kit Memoria RAM Corsair Vengeance LPX DDR4	Searches: 0.
ID number: 60	Name: Kit Memoria RAM Corsair Dominator Platinum DDR4	Searches: 0.
pantallas		
ID number: 62	Name: Makena Smart TV LED 32S2 32”	Searches: 0.
ID number: 64	Name: Samsung TV LED LH43QMREBGCXGO 43	Searches: 0.
ID number: 65	Name: Samsung Smart TV LED UN70RU7100FXZX 70	Searches: 0.
ID number: 68	Name: Makena Smart TV LED 40S2 40”	Searches: 0.
ID number: 69	Name: Hisense Smart TV LED 40H5500F 39.5	Searches: 0.
ID number: 71	Name: Samsung Smart TV LED UN32J4290AF 32	Searches: 0.
ID number: 72	Name: Hisense Smart TV LED 50H8F 49.5	Searches: 0.
ID number: 70	Name: Samsung Smart TV LED 43	Searches: 1.
ID number: 63	Name: Seiki TV LED SC-39HS950N 38.5	Searches: 4.
ID number: 73	Name: Samsung Smart TV LED UN55TU7000FXZX 55	Searches: 4.
procesadores		
ID number: 9	Name: Procesador Intel Core i3-8100	Searches: 1.
ID number: 1	Name: Procesador AMD Ryzen 3 3300X S-AM4	Searches: 10.
ID number: 6	Name: Procesador Intel Core i9-9900K	Searches: 10.
ID number: 8	Name: Procesador Intel Core i5-9600K	Searches: 20.
ID number: 2	Name: Procesador AMD Ryzen 5 3600	Searches: 24.
ID number: 5	Name: Procesador Intel Core i3-9100F	Searches: 30.
ID number: 7	Name: Procesador Intel Core i7-9700K	Searches: 31.
ID number: 4	Name: Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8	Searches: 41.

ID number: 3	Name: Procesador AMD Ryzen 5 2600	Searches: 55.
tarjetas de video		
ID number: 14	Name: Tarjeta de Video EVGA NVIDIA GeForce GT 710	Searches: 0.
ID number: 16	Name: Tarjeta de Video EVGA NVIDIA GeForce RTX 2060	Searches: 0.
ID number: 19	Name: Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650	Searches: 0.
ID number: 20	Name: Tarjeta de Video Gigabyte NVIDIA GeForce RTX 2060	Searches: 0.
ID number: 23	Name: Tarjeta de Video MSI Radeon X1550	Searches: 0.
ID number: 24	Name: Tarjeta de Video PNY NVIDIA GeForce RTX 2080	Searches: 0.
ID number: 27	Name: Tarjeta de Video VisionTek AMD Radeon HD5450	Searches: 1.
ID number: 10	Name: MSI GeForce 210	Searches: 1.
ID number: 13	Name: Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix	Searches: 2.
ID number: 17	Name: Tarjeta de Video Gigabyte AMD Radeon R7 370 OC	Searches: 3.
tarjetas madre		
ID number: 30	Name: Tarjeta Madre AORUS ATX Z390 ELITE	Searches: 0.
ID number: 32	Name: Tarjeta Madre ASRock Z390 Phantom Gaming 4	Searches: 0.
ID number: 34	Name: Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING WI-FI	Searches: 0.
ID number: 36	Name: Tarjeta Madre Gigabyte micro ATX Z490M GAMING X (rev. 1.0)	Searches: 0.
ID number: 37	Name: Tarjeta Madre ASRock ATX Z490 STEEL LEGEND	Searches: 0.
ID number: 38	Name: Tarjeta Madre Gigabyte Micro ATX H310M DS2 2.0	Searches: 0.
ID number: 41	Name: Tarjeta Madre ASUS micro ATX Prime H370M-Plus	Searches: 0.
ID number: 43	Name: Tarjeta Madre ASUS ATX ROG STRIX Z390-E GAMING	Searches: 0.
ID number: 33	Name: Tarjeta Madre ASUS ATX PRIME Z390-A	Searches: 0.
ID number: 35	Name: Tarjeta Madre Gigabyte micro ATX Z390 M GAMING	Searches: 1.

3.3 Best and worst rated products

The best rated products are shown in Table 3.5.

Table 3.5: Highest rated products .

ID number: 1	Name: Procesador AMD Ryzen 3 3300X S-AM4	Average score: 5.0.
ID number: 6	Name: Procesador Intel Core i9-9900K	Average score: 5.0.
ID number: 7	Name: Procesador Intel Core i7-9700K	Average score: 5.0.
ID number: 8	Name: Procesador Intel Core i5-9600K	Average score: 5.0.
ID number: 11	Name: Tarjeta de Video ASUS AMD Radeon RX 570	Average score: 5.0.

The worst rated products are shown in Table 3.6.

Table 3.6: Lowest rated products .

ID number: 17	Name: Tarjeta de Video Gigabyte AMD Radeon R7 370 OC	Average score: 1.0.
ID number: 45	Name: Tarjeta Madre ASRock ATX H110 Pro BTC+	Average score: 1.0.
ID number: 31	Name: Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0)	Average score: 1.84.
ID number: 46	Name: Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2	Average score: 2.0.
ID number: 89	Name: Cougar Audífonos Gamer Phontum Essential	Average score: 3.0.

3.4 Income and sales

The total amount of money gained per month is shown in Figure 3.1.

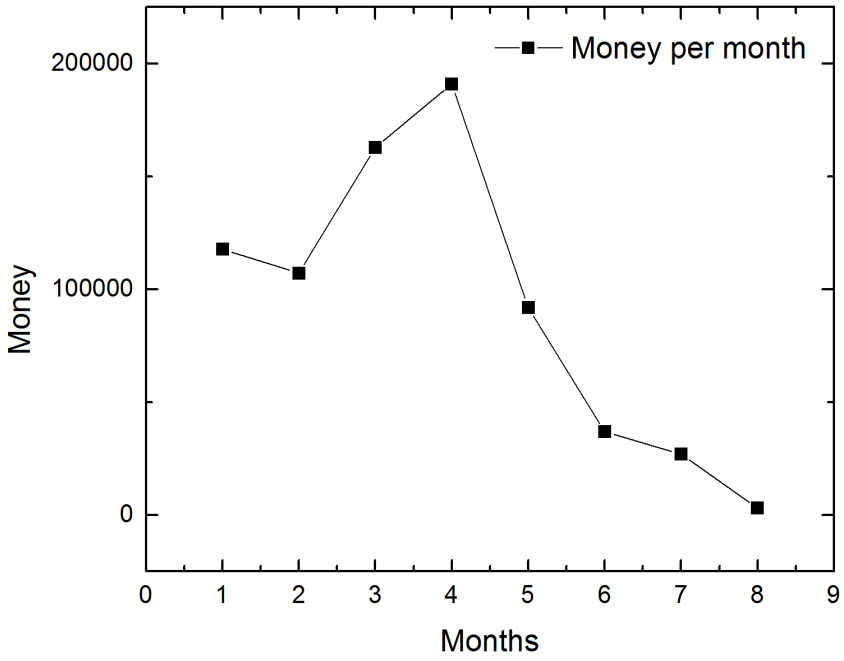

Figure 3.1: Sum of money gained per month.

As we can see there was a peak in April.
The total amount of money earned throughout all months was:
737916
And the average number of sales per month was:
34.25
The months with the most sales are shown in Table 3.7

Table 3.7: Months with the most sales.

April	74
January	52
March	49
February	40
May	34

The months with the most income are shown in Table 3.8

Table 3.8: Months with the most income.

April	191066
March	162931
January	117738
February	107270
May	91936

Chapter 4

Proposed solution

As we can see from the results, there are categories which get no sales at all, and are also barely searched on the LifeStore site. The categories with at least 5 articles with no sales at all are audifonos, bocinas, discos duros, pantallas, tarjetas de video and tarjetas madre, and the categories with the least amount of searches by their products are audifonos, bocinas, memorias usb, pantallas, tarjetas de video and tarjetas madre. Several approaches can be taken to fix the stock issues. The approach suggested by the author of this work is to have the company focus on processors and hard drives, since that is where most of its income comes from. But whole categories such as pantallas and audifonos, as well as tarjetas de video should have their stock drastically reduced if not completely removed from the market. The company should instead focus on their strengths such as the processors.

Chapter 5

Conclusion

A Python program was written to fulfill the requests of the LifeStore company, which has stock issues. EmTech has asigned this project to their students, not before training them in the Python programming language and teaching them about the programming concepts necessary to accomplish this task. Therefore, a program was made to analyze the sales and stock data provided by LifeStore's Sales Management, and to obtain certain values such as the 5 most sold products and the 5 worst rated products. From the values obtained it can be easy to make a plan for the future of the company.

Appendix A

Python Code

```
from lifestore_file import lifestore_products, lifestore_sales, lifestore_searches
import math

"""
User login section
User: Lifestore
Password: store15
"""

#--------------------------LOGIN-------------------------------#

userAccess = False
attempts = 0

#Welcome message
print("Welcome to the data analysis program!\nPlease log in with your username and password\n")

#loop for log in attempts
while userAccess == False and attempts < 3:
  user = input("Please write your username here: ")
  password = input("Please write your password here: ")
  attempts += 1
  if user == "Lifestore" and password == "store15":
    userAccess = True
  elif attempts == 3:
    print("You have reached the maximum number of attempts. The program will now close.")
    exit()
  elif user != "Lifestore":
    print("This user is not registered in the system. Please try again (attempt " + str(attempts) + "/3)")
  elif password != "store15":
    print("The password is incorrect. Please try again(attempt " + str(attempts) + "/3)")

print("\n\nWelcome, ", user, "\n\n")

#--------------------------MOST SOLD PRODUCTS-------------------------------#

#Generate a list with the product id appearing as many times as it was sold without taking into account the
↪  refunds
sales_per_product = [sales[1] for sales in lifestore_sales if sales[4] != 1]

```

```python
#Generate a list with the product id appearing as many times as it was searched
search_per_product =[search[1] for search in lifestore_searches]

#Generate a list with the product category appearing as many times as the products that belong to it
products_per_category =[producto[3] for producto in lifestore_products]

#Generate a list containing each category once
categories = []
[categories.append(ctgry) for ctgry in products_per_category if ctgry not in categories]

#Sort the categories in alphabetical order
categories.sort()

#Generate a list with 4 values per item: product id, sales per product, searches per product and category
product_sales_searches = [[product[0], sales_per_product.count(product[0]),
↪  search_per_product.count(product[0]), product[3]] for product in lifestore_products]

#Defines a function necessary to sort product_sales_searches by sales
def function_sales(e):
  return e[1]

#Defines a function necessary to sort product_sales_searches by searches
def function_searches(e):
  return e[2]

#Sort our list by sales, with the most sold products on the top
product_sales_searches.sort(reverse = True, key = function_sales)

#Displays the 5 most sold products
print("The most sold products were: ")
for q in range(5):
  print(f"ID number: {product_sales_searches[q][0]} // Name: {lifestore_products[product_sales_searches[q][0]
  ↪  - 1 ][1]} // Sales: {product_sales_searches[q][1]}.")
print("\n\n")

#----------------------------LEAST SOLD PRODUCTS---------------------------------#

#Sort our list by sales, with the least sold products on the top
product_sales_searches.sort(reverse = False, key = function_sales)

#Displays the 5 least sold products by category
print("The least sold products by category were: \n")
i = 0
for item in categories:
  print("\n" + categories[i] + ": ")
  i += 1
  q1 = 0
  q2 = 0
  for q in product_sales_searches:
    if item == product_sales_searches[q1][3] and q2 < 5:
      print(f"ID number: {product_sales_searches[q1][0]} // Name:
      ↪  {lifestore_products[product_sales_searches[q1][0] - 1 ][1]} // Sales:
      ↪  {product_sales_searches[q1][1]}.")
      q2 += 1
    q1 += 1
print("\n\n")
```

```
92
93  #----------------------------MOST SEARCHED PRODUCTS---------------------------------#
94
95  #Sort our list by searches, with the most searched products on the top
96  product_sales_searches.sort(reverse = True, key = function_searches)
97
98  #Displays the 10 most searched products
99  print("The most searched products were: ")
100 for q in range(10):
101   print(f"ID number: {product_sales_searches[q][0]} // Name: {lifestore_products[product_sales_searches[q][0]
      ↪  - 1 ][1]} // Searches: {product_sales_searches[q][2]}.")
102 print("\n\n")
103
104 #Sort our list by searches, with the least searched products on the top
105 product_sales_searches.sort(reverse = False, key = function_searches)
106
107 #----------------------------LEAST SEARCHED PRODUCTS---------------------------------#
108
109 #Displays the 10 least searched products by category
110 print("The least searched products by category were: \n")
111 i = 0
112 for item in categories:
113   print("\n" + categories[i] + ": ")
114   i += 1
115   q1 = 0
116   q2 = 0
117   for q in product_sales_searches :
118     if item == product_sales_searches[q1][3] and q2 < 10 :
119       print(f"ID number: {product_sales_searches[q1][0]} // Name:
        ↪  {lifestore_products[product_sales_searches[q1][0] - 1 ][1]} // Searches:
        ↪  {product_sales_searches[q1][2]}.")
120       q2 += 1
121     q1 += 1
122 print("\n\n")
123
124 #----------------------------BEST SCORED PRODUCTS---------------------------------#
125
126 #Make a list containing product id and score from lifestore_sales
127 id_reviews_not_separated = [[sale[1], sale[2]] for sale in lifestore_sales]
128
129 #Make a dictionary to save the scores as lists for each product
130 id_reviews_count = {}
131
132 #Add values to each element in the dictionary and add score value lists for each element
133 for par in id_reviews_not_separated:
134     id = par[0]
135     review = par[1]
136     if id not in id_reviews_count.keys():
137         id_reviews_count[id] = []
138     id_reviews_count[id].append(review)
139
140 #Create a list containing only the product ids of the products with a score
141 id_with_score = []
142 [id_with_score.append(idprdct) for idprdct in id_reviews_count if idprdct not in id_with_score]
143
144 #Create a list to save the average values of each product
```

```
averages = []

#Get the average score value of each product and put it into the averages list
for id_product in id_reviews_count.keys():
    lista_reviews = id_reviews_count[id_product]
    promedio = sum(lista_reviews) / len(lista_reviews)
    # Set the decimals to 2
    decimales = 2
    multiplicador = 10 ** decimales
    promedio = math.ceil(promedio * multiplicador) / multiplicador
    averages.append(promedio)

#Join the id_with_score list with the averages list
id_avgscore = list(zip(id_with_score,averages))

#Defines a function necessary to sort id_avgscore by the average score
def function_avgscore(e):
  return e[1]

#Sort our list by averege score, with the highest scored products on the top
id_avgscore.sort(reverse = True, key = function_avgscore)

#Displays the 5 highest rated products
print("The highest rated products were: ")
for q in range(5):
  print(f"ID number: {id_avgscore[q][0]} // Name: {lifestore_products[id_avgscore[q][0] - 1 ][1]} // Average
  ↪  score: {id_avgscore[q][1]}.")
print("\n\n")

#----------------------------WORST SCORED PRODUCTS---------------------------------#

#Sort our list by averege score, with the lowest scored products on the top
id_avgscore.sort(reverse = False, key = function_avgscore)

#Displays the 5 lowest rated products
print("The lowest rated products were: ")
for q in range(5):
  print(f"ID number: {id_avgscore[q][0]} // Name: {lifestore_products[id_avgscore[q][0] - 1 ][1]} // Average
  ↪  score: {id_avgscore[q][1]}.")
print("\n\n")

#----------------------------TOTAL MONEY BY MONTH---------------------------------#

# Creates a list that contains the sale and the date of each sale
id_date = [ [sale[0], sale[3]] for sale in lifestore_sales if sale[4] == 0 ]

# Creates a dictionary that contains the months and the sales per month
category_months = {}

#Create a list of months that had any sales in them
months_with_sales = []

for par in id_date:
    # Obtains sale and month
    id = par[0]
    _, month, _ = par[1].split('/')
```

```
    # Makes each month into a key, if it isn't one already
    if month not in category_months.keys():
        category_months[month] = []
    #Include into months_with_sales every month that had any sales in it (no refunds)
    if month not in months_with_sales:
        months_with_sales.append(month)
    #Appends each sale to the month it belongs to
    category_months[month].append(id)

#Creates a list that contains the sum of money earned each month, without mentioning the month
money_per_month = []

#Creates a list that contains the amount of sales each month
sales_per_month = []

for key in category_months.keys():
    list_month = category_months[key]
    sum_sale = 0
    for id_sale in list_month:
        index = id_sale - 1
        info_sale = lifestore_sales[index]
        id_product = info_sale[1]
        price = lifestore_products[id_product-1][2]
        sum_sale += price
    #Append the sum money earned every month
    money_per_month.append(sum_sale)
    #Append the number of sales of each month
    sales_per_month.append(len(list_month))

#Join the months_with_sales, money_per_month and sales_per_month lists into a single list
month_money_sale = list(zip(months_with_sales,money_per_month, sales_per_month))

#Defines a function necessary to sort month_money_sale by month
def function_month1(e):
  return e[0]

#Defines a function necessary to sort month_money_sale by ammount of money
def function_money1(e):
  return e[1]

#Defines a function necessary to sort month_money_sale by number of sales
def function_sale1(e):
  return e[2]

#Sort our list by month, with the first on the top
month_money_sale.sort(reverse = False, key = function_month1)

#Prints the sum of money obtained by each month
m = 0
for months in month_money_sale:
  print(f"The sum of money obtained in the month {month_money_sale[m][0]} was: {month_money_sale[m][1]}")
  m += 1
print("\n\n")

#---------------------------AVERAGE SALES PER MONTH--------------------------------#

```

```
#Average number of sales per month
avg_salePerMonth = sum(sales_per_month) / len(sales_per_month)

print(f"The average number of sales per month was: {avg_salePerMonth}\n\n")

#----------------------------TOTAL MONEY THROUGHOUT YEAR---------------------------------#

#Total money obtained throughout the year
total_money = sum(money_per_month)

print(f"The total sum of money throughout the year was: {total_money}\n\n")

#----------------------------MONTHS WITH MOST SALES---------------------------------#

#Sort our list by number of sales, with the biggest ammount of sales on the top
month_money_sale.sort(reverse = True, key = function_sale1)

#Prints the 5 months with the most sales
print("The months with the most ammount of sales were :")
for m in range(5):
  print(f"Month :{month_money_sale[m][0]} // Number of sales: {month_money_sale[m][2]}")
  m
print("\n\n")

#----------------------------MONTHS WITH MOST MONEY GAINED---------------------------------#

#Sort our list by money, with the biggest ammount of money on the top
month_money_sale.sort(reverse = True, key = function_money1)

#Prints the 5 months with the most sales
print("The months with the most ammount of money earned were :")
for m in range(5):
  print(f"Month :{month_money_sale[m][0]} // Ammount of money: {month_money_sale[m][1]}")
  m
print("\n\n")
```