



# Reporte Final

Proyecto 01.

EmTech Institute

28/08/20

- Jorge Daniel Madrigal  
Hernandez.

Lufcy



## Indice

Introduccion	2
Definicion de código	3
Solucion del problema	10
Conclusion.	12

## Introducción.

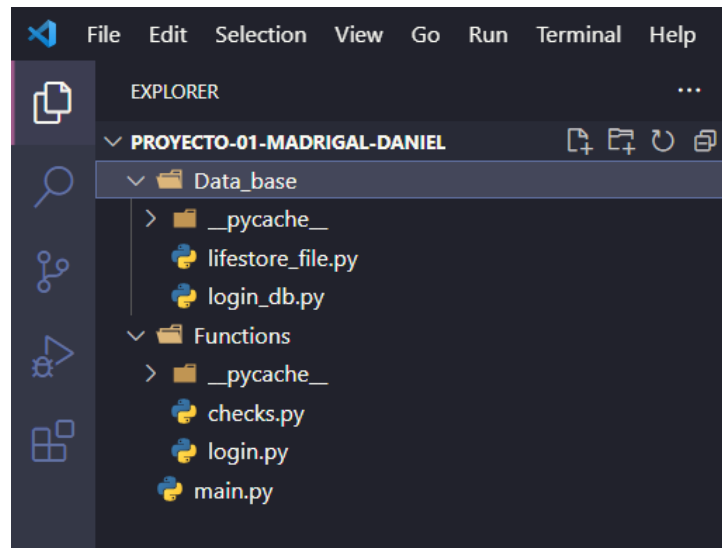
Programa en GitHub: <https://github.com/Lufcy1/PROYECTO-01--MADRIGAL-DANIEL>

Instrucciones que se tomaron en cuenta para la realización de este proyecto:

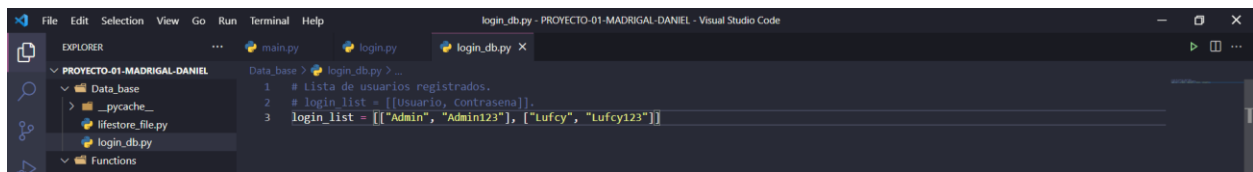
El análisis deberá considerar el desarrollo de un sistema de análisis, en el que mediante un login de usuario-administrador se muestre un reporte mensual que especifique los puntos señalados en la consigna, específicamente:

1. Productos más vendidos y productos rezagados:
  - Generar un listado de los 50 productos con mayores ventas y uno con los 100 productos con mayores búsquedas.
  - Por categoría, generar un listado con los 50 productos con menores ventas y uno con los 100 productos con menores búsquedas.
2. Productos por reseña en el servicio
  - Mostrar dos listados de 20 productos cada uno, un listado para productos con las mejores reseñas y otro para las peores, considerando los productos con devolución.
3. Total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año.

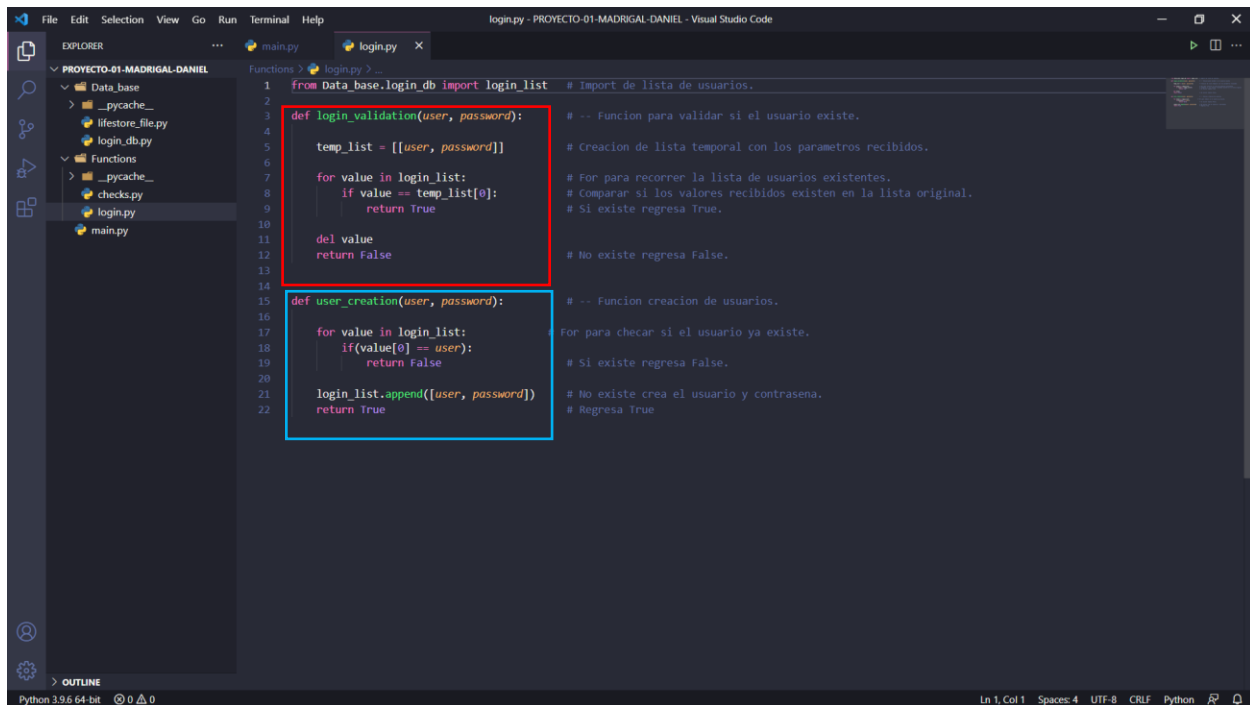
Definición de código.



Carpetas y archivos .py del programa, se separaron en carpetas los datos de las bases de datos, las funciones y el programa principal para un mejor ordenamiento de la información.



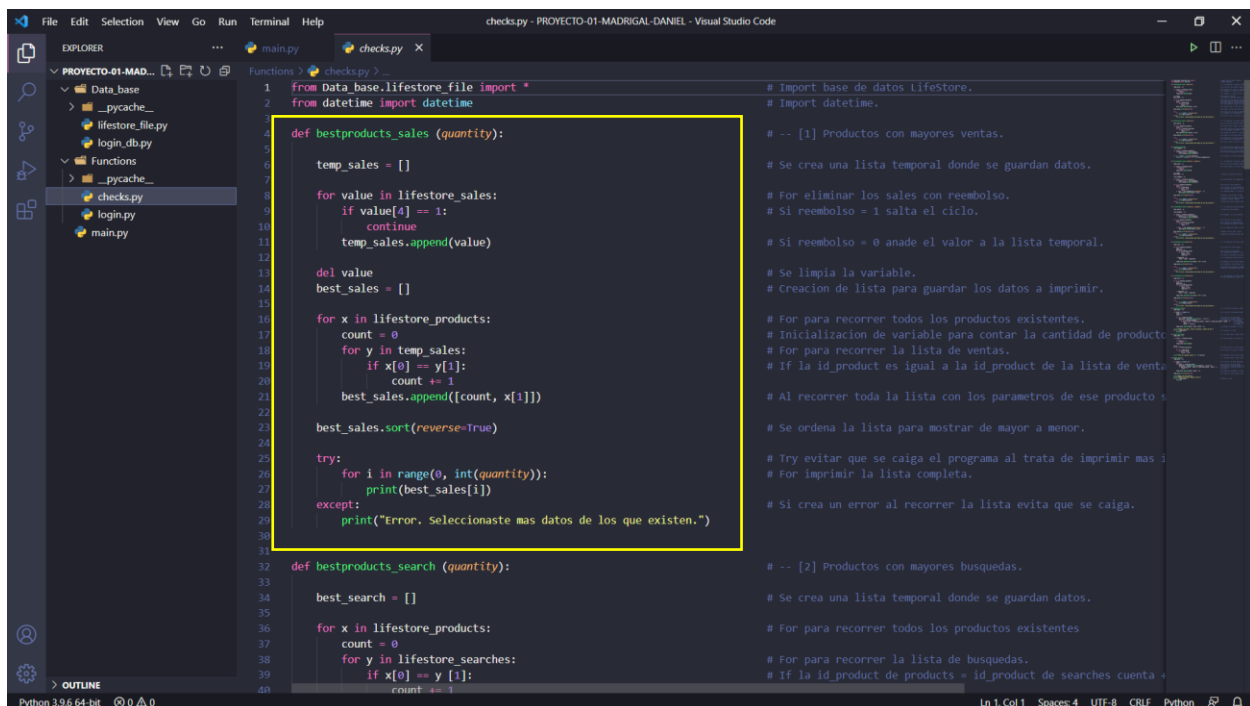
El archivo login\_db.py solo cuenta con una lista de sublistas con usuarios y contraseña predefinidos, aunque es posible añadir más en el registro de la aplicación.




```
1 from Data_base.login_db import login_list # Import de lista de usuarios.
2
3 def login_validation(user, password): # -- Funcion para validar si el usuario existe.
4     temp_list = [[user, password]] # Creacion de lista temporal con los parametros recibidos.
5
6     for value in login_list: # For para recorrer la lista de usuarios existentes.
7         if value == temp_list[0]: # Comparar si los valores recibidos existen en la lista original.
8             return True # Si existe regresa True.
9
10     del value # No existe regresa False.
11     return False
12
13 def user_creation(user, password): # -- Funcion creacion de usuarios.
14     for value in login_list: # For para checar si el usuario ya existe.
15         if(value[0] == user): # Si existe regresa False.
16             return False
17
18     login_list.append([user, password]) # No existe crea el usuario y contraseña.
19     return True # Regresa True
```

 Funcion para validar si el usuario y contraseña ingresado existen en la lista.

 Funcion para agregar nuevos usuarios y contraseñas a la lista.



```
1 from Data_base.lifestore_file import * # Import base de datos lifestore.
2 from datetime import datetime # Import datetime.
3
4 def bestproducts_sales(quantity): # -- [1] Productos con mayores ventas.
5     temp_sales = [] # Se crea una lista temporal donde se guardan datos.
6
7     for value in lifestore_sales: # For eliminar los sales con reembolso.
8         if value[4] == 1: # Si reembolso = 1 salta el ciclo.
9             continue
10         temp_sales.append(value) # Si reembolso = 0 anade el valor a la lista temporal.
11
12     del value # Se limpia la variable.
13     best_sales = [] # Creacion de lista para guardar los datos a imprimir.
14
15     for x in lifestore_products: # For para recorrer todos los productos existentes.
16         count = 0 # Inicializacion de variable para contar la cantidad de productos.
17         for y in temp_sales: # For para recorrer la lista de ventas.
18             if x[0] == y[1]: # If la id_product es igual a la id_product de la lista de ventas.
19                 count += 1
20             best_sales.append([count, x[1]]) # Al recorrer toda la lista con los parametros de ese producto
21
22     best_sales.sort(reverse=True) # Se ordena la lista para mostrar de mayor a menor.
23
24     try: # Try evitar que se caiga el programa al trata de imprimir mas de 10 productos.
25         for i in range(0, int(quantity)): # For imprimir la lista completa.
26             print(best_sales[i])
27     except: # Si crea un error al recorrer la lista evita que se caiga.
28         print("Error. Seleccionaste mas datos de los que existen.")
29
30 def bestproducts_search(quantity): # -- [2] Productos con mayores búsquedas.
31     best_search = [] # Se crea una lista temporal donde se guardan datos.
32
33     for x in lifestore_products: # For para recorrer todos los productos existentes
34         count = 0 # For para recorrer la lista de búsquedas.
35         for y in lifestore_searches: # If la id_product de products = id_product de searches cuenta + 1
36             if x[0] == y[1]:
37                 count += 1
```

 Se separo cada una de las opciones en funciones para mejorar su comprensión y hacer el programa mas editable en el futuro.

El uso y funcionamiento de cada una de estas funciones puede ser entendido leyendo los comentarios hechos en la misma programacion.

```
File Edit Selection View Go Run Terminal Help
check.py - PROYECTO-01-MADRIGAL-DANIEL - Visual Studio Code

main.py check.py
Functions > check.py > ...

31
32 def bestproducts_search (quantity): # -- [2] Productos con mayores búsquedas.
33     best_search = [] # Se crea una lista temporal donde se guardan datos.
34
35     for x in lifestore_products: # For para recorrer todos los productos existentes
36         count = 0 # For para recorrer la lista de búsquedas.
37         for y in lifestore_searches: # If la id_product de products = id_product de searches cuenta + 1.
38             if x[0] == y[1]:
39                 count += 1
40             best_search.append([count, x[1]])
41         # Al terminar de recorrer la lista búsquedas se agrega la cantidad de veces que se repitió y el nombre.
42     best_search.sort(reverse=True)
43     # Se ordena la lista para mostrar de mayor a menor.
44
45     try: # Try evitar que se caiga el programa al trata de imprimir mas índices de los que existen.
46         for i in range(0, int(quantity)): # For imprimir la lista completa.
47             print(best_search[i])
48     except: # Si crea un error al recorrer la lista evita que se caiga.
49         print("Error. Seleccionaste mas datos de los que existen.")
50
51
52 def category_select(): # -- Funcion mostrar categorias existentes.
53     list_category = [] # Se crea una lista temporal donde se guardan datos.
54
55     for value in lifestore_products: # For para recorrer la lista productos.
56         if value[3] not in list_category: # If el valor no existe en la lista categoria se agrega a esta misma lista.
57             list_category.append(value[3])
58
59     for i in range(0, len(list_category)): # For imprimir la lista de categorias existentes.
60         print("[ " + str(i+1) + " ] " + str(list_category[i]))
61
62
63 def worstproducts_sales (quantity, category): # -- [3] Productos con menores ventas por categoria.
64     temp_sales = [] # Se crea una lista temporal donde se guardan datos.
65
66     for value in lifestore_sales: # For para eliminar las ventas reembolsadas.
67         if value[4] == 1:
68             continue
69         temp_sales.append(value)
70
71     del value
72     best_sales = [] # Creacion de lista para imprimir.
73
74     list_category = []
75
76     for value in lifestore_products: # For para guardar las categorias existentes.
77         if value[3] not in list_category:
78             list_category.append(value[3])
79
80
81 def worstproducts_sales (quantity, category): # -- [3] Productos con menores ventas por categoria.
82     temp_sales = [] # Se crea una lista temporal donde se guardan datos.
83
84     for value in lifestore_sales: # For para eliminar las ventas reembolsadas.
85         if value[4] == 1:
86             continue
87         temp_sales.append(value)
88
89     del value
90     best_sales = [] # Creacion de lista para imprimir.
91
92     list_category = []
93
94     for value in lifestore_products: # For para guardar las categorias existentes.
95         if value[3] not in list_category:
96             list_category.append(value[3])
97
98     for x in lifestore_products: # For recorrer lista productos.
99         count = 0
100         for y in temp_sales: # for recorrer la lista temporal de ventas / Sin reembolsos.
101             if x[0] == y[1]: # If id_product de product = id_product de ventas, cuenta + 1.
102                 count += 1
103             if x[3] == list_category[int(category) - 1]: # If category_product = al numero de la categoria recibida por parametro se anade.
104                 best_sales.append([count, x[1]])
105
106     best_sales.sort(reverse=False) # Ordena la lista de menor a mayor.
107
108     try: # Try evitar que se caiga el programa al trata de imprimir mas índices de los que existen.
109         for i in range(0, int(quantity)): # Imprimir la lista a mostrar.
110             print(best_sales[i])
111     except: # Except evitar que se caiga el programa.
112         print("Error. Seleccionaste mas datos de los que existen.")
113
114
115 def worstproducts_search (quantity, category): # -- [4] Productos con menores búsquedas por categoria.
116     best_search = [] # Creacion de lista temporal.
117
118     list_category = []
119
120     for value in lifestore_products: # For Agregar a la lista categoria las categorias existentes en products.
121         if value[3] not in list_category:
122             list_category.append(value[3])
123
124     for x in lifestore_products: # For recorrer la lista products
125         count = 0
126         for y in lifestore_searches: # For recorrer la lista búsquedas.
```

```
File Edit Selection View Go Run Terminal Help
checkipy - PROYECTO-01-MADRIGAL-DANIEL - Visual Studio Code

main.py checkipy
Functions > checkipy > ...

98 def worstproducts_search(quantity, category):           # -- [4] Productos con menores búsquedas por categoría.
99                                                         # Creación de lista temporal.
100     best_search = []
101     list_category = []
102
103     for value in lifestore_products:                     # For Agregar a la lista categoría las categorías existentes en products.
104         if value[3] not in list_category:
105             list_category.append(value[3])
106
107     for x in lifestore_products:                         # For recorrer la lista products
108         count = 0
109         for y in lifestore_searches:                     # For recorrer la lista búsquedas.
110             if x[0] == y[1]:                             # If id_product de products = id_products de searches cuenta + 1.
111                 count += 1
112
113             if x[3] == list_category[int(category) - 1]: # If la categoría es igual a la categoría seleccionada se agrega.
114                 best_search.append([count, x[1]])
115
116     best_search.sort(reverse=False)                     # Ordenar lista de menor a mayor.
117
118     try:
119         for i in range(0, int(quantity)):                # Imprime la cantidad de datos seleccionado.
120             print(best_search[i])
121     except:
122         print("Error. Seleccionaste mas datos de los que existen.")
123
124
125 def bestproducts_score(quantity):                       # -- [5] Productos con mejores reseñas.
126     temp_score = []
127
128     for x in lifestore_products:                       # For recorrer la lista products.
129         count = 0
130         duplicate = 0
131
132         for y in lifestore_sales:                      # For recorrer la lista ventas.
133             if x[0] == y[1]:                             # If id_product de products = id product de sales se suma la la score de esta venta.
134                 count += y[2]
135                 duplicate += 1
136
137             if duplicate > 0:                           # If se repite mas de una vez se saca el promedio del score. (Para evitar dividir por cero.)
138                 count = count / duplicate
139
140     temp_score.append([format(count, ".2f"), x[1]])      # Se agrega el score con dos dígitos después del decimal y en nombre del producto.
141
142     temp_score.sort(reverse=True)                     # Se reordena la lista de mayor a menor.
143
144     try:
145         for i in range(0, int(quantity)):                # Try evitar que se caiga el programa por seleccionar un índice mayor al existente.
146             print(temp_score[i])                         # Imprimir la cantidad de datos seleccionado.
147     except:
148         print("Error. Seleccionaste mas datos de los que existen.")
149
150
151 def worstproducts_score(quantity):                     # -- [6] Productos con peores reseñas.
152                                                         # Se hace exactamente lo mismo con la anterior función solo se ordena de menor a mayor.
153     temp_score = []
154
155     for x in lifestore_products:                       # For recorrer la lista products.
156         count = 0
157         duplicate = 0
158
159         for y in lifestore_sales:                      # For recorrer la lista ventas.
160             if x[0] == y[1]:                             # If id_product de products = id product de sales se suma la la score de esta venta.
161                 count += y[2]
162                 duplicate += 1
163
164             if duplicate > 0:                           # If se repite mas de una vez se saca el promedio del score. (Para evitar dividir por cero.)
165                 count = count / duplicate
166
167     temp_score.append([format(count, ".2f"), x[1]])      # Se agrega el score con dos dígitos después del decimal y en nombre del producto.
168
169     temp_score.sort(reverse=False)                     # Se reordena la lista de mayor a menor.
170
171     try:
172         for i in range(0, int(quantity)):                # Try evitar que se caiga el programa por seleccionar un índice mayor al existente.
173             print(temp_score[i])                         # Imprimir la cantidad de datos seleccionado.
174     except:
175         print("Error. Seleccionaste mas datos de los que existen.")
176
177
178 Python 3.9.6 64-bit 0 0 0
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
```

```
File Edit Selection View Go Run Terminal Help
checkpy - PROYECTO-01-MADRIGAL-DANIEL - Visual Studio Code

main.py checkpy x
Functions > checkpy 2 ...

151 def worstproducts_score(quantity): # -- [6] Productos con peores reseñas.
152     # Se hace exactamente lo mismo con la anterior función solo se ordena de menor a mayor.
153     temp_score = []
154     for x in lifestore_products:
155         count = 0
156         duplicate = 0
157         for y in lifestore_sales:
158             if x[0] == y[1]:
159                 count += y[2]
160                 duplicate += 1
161         if duplicate > 0:
162             count = count / duplicate
163         temp_score.append([format(count, ".2f"), x[1]])
164     temp_score.sort(reverse=False)
165     try:
166         for i in range(0, int(quantity)):
167             print(temp_score[i])
168     except:
169         print("Error. Seleccionaste mas datos de los que existen.")
170
171 def bestmonth_sales(): # -- [7] Total de ingresos y ventas promedio mensuales.
172     temp_sales = []
173     for month in range(0,12): # For recorrer la cantidad de meses que existen.
174         total = 0
175         count = 0
176         for y in lifestore_sales: # For recorrer la lista ventas.
177             date_sale = datetime.strptime(y[3], "%d/%m/%Y") # convertir el string de fecha en datetime.
178             for x in lifestore_products: # For recorrer la lista productos.
179                 if x[0] == y[1] and date_sale.month > month and date_sale.month < month + 2: # If id product = id product en ventas y
180                     # la fecha es mayor al X mes pero menos a X mes + 1.
181                     count += 1 # se agrega las veces que se repite y el total de precios.
182                     total += int(x[2])
183             temp_sales.append([count, total, month + 1]) # Se agrega la cantidad, el total y el numero del mes en la lista.
184     print("Numero de ventas, total de ingresos, numero de mes")
185     for i in temp_sales: # For imprimir la lista.
186         print(i)
187
188 def total_earnings(): # -- [8] Total anual y meses con mas ventas. / 1
189     temp_sales = []
190
191     for value in lifestore_sales: # For para eliminar las ventas con reembolso.
192         if value[4] == 1:
193             continue
194         temp_sales.append(value)
195     price = 0
196     for x in lifestore_products: # For para recorrer la lista productos.
197         for y in temp_sales: # For recorrer la lista y si estan en la lista se suma el precio de este mismo.
198             if x[0] == y[1]:
199                 price += x[2]
200     print("Total de ingresos anual: $" + str(price)) # Se imprime el total con dos decimales.
201
202 def month_sales(): # -- [8] Total anual y meses con mas ventas. / 2
203     temp_sales = []
204     for month in range(0,12): # For recorrer los meses existentes.
205         count = 0
206         for value in lifestore_sales: # For recorrer la lista ventas.
207             date_sale = datetime.strptime(value[3], "%d/%m/%Y") # Convertir el string de fechas a datetime.
208             if date_sale.month > month and date_sale.month < month + 2: # If es mayor a X mes pero menor a este X mes + 1
209                 count += 1 # Se suma + 1.
210         temp_sales.append([count, month + 1]) # Se agrega las cantidad y el numero de mes.
211     temp_sales.sort(reverse=True) # Se ordena la lista de mayor a menor.
212     print("Meses con mas ventas")
213     print("Numero de ventas, Numero de mes")
214     for i in temp_sales: # Imprime la lista.
215         print(i)
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233

Python 3.9.6 64-bit 0 0.0
```



```
import os
import getpass
from time import sleep
import Functions.login
import Functions.checks

clear = lambda: os.system('cls')

print("----- Inicio de Sesión -----")

while True:
    choose = input("cuenta con un usuario existente? y/n: ")
    if choose == "y" or choose == "yes" or choose == "si":
        while True:
            username = input("Ingresa nombre de usuario: ")
            userpass = getpass.getpass("Ingresa contraseña de usuario: ")
            if Functions.login.login_validation(username, userpass):
                print("Ingreso Exitoso.")
                sleep(1.5)
                clear()
                print("----- Bienvenido a LifeStore Application -----")
                while True:
                    print("\n[1] Productos con mayores ventas.")
                    print("\n[2] Productos con mayores búsquedas.")
                    print("\n[3] Productos con menores ventas por categoría.")
                    print("\n[4] Productos con menores búsquedas por categoría.")
                    print("\n[5] Productos con mejores reseñas.")
                    print("\n[6] Productos con peores reseñas.")
                    print("\n[7] Total de ingresos y ventas promedio mensuales.")
                    print("\n[8] Total anual y meses con mas ventas.")
                    election = input("Selecciona una opción (Número): ")
                    if election == '1':
                        print("\n----- Productos con mayores ventas -----")
```

- Se importaron todas las librerías necesarias y las funciones hechas en otros archivos .py
- While True principal, para mantener siempre el programa en funcionamiento.
- Selección de si desea crear una cuenta o ingresar con una existente.
- Comprobación de si existe el nombre y contraseña en la base de datos, si regresa True entra la aplicación principal, si regresa False te regresa al inicio de ese While para ingresar de nuevo los datos.

```

26
27 while True:
28     print("\n[1] Productos con mayores ventas.")
29     print("\n[2] Productos con mayores búsquedas.")
30     print("\n[3] Productos con menores ventas por categoría.")
31     print("\n[4] Productos con menores búsquedas por categoría.")
32     print("\n[5] Productos con mejores reseñas.")
33     print("\n[6] Productos con peores reseñas.")
34     print("\n[7] Total de ingresos y ventas promedio mensuales.")
35     print("\n[8] Total anual y meses con mas ventas.")
36
37     eleccion = input("Selecciona una opción (Número): ")
38
39     if eleccion == '1':
40         print("\n----- Productos con mayores ventas -----")
41         Functions.checks.bestproducts_sales(input("Selecciona la cantidad de productos que desea mostrar: "))
42
43     elif eleccion == '2':
44         print("\n----- Productos con mayores búsquedas -----")
45         Functions.checks.bestproducts_search(input("Selecciona la cantidad de productos que desea mostrar: "))
46
47     elif eleccion == '3':
48         print("\n----- Productos con menores ventas por categoría -----")
49         Functions.checks.categories_select()
50         category = input("Selecciona la categoría (Número): ")
51         quantity = input("Selecciona la cantidad de productos que desea mostrar: ")
52         Functions.checks.worstproducts_sales(quantity, category)
53
54     elif eleccion == '4':
55         print("\n----- Productos con menores búsquedas por categoría -----")
56         Functions.checks.categories_select()
57         category = input("Selecciona la categoría (Número): ")
58         quantity = input("Selecciona la cantidad de productos que desea mostrar: ")
59         Functions.checks.worstproducts_search(quantity, category)
60
61     elif eleccion == '5':
62         print("\n----- Productos con mejores reseñas -----")
63         Functions.checks.bestproducts_score(input("Selecciona la cantidad de productos que desea mostrar: "))
64
65     elif eleccion == '6':

```

**E** Elección de opciones a elegir ingresando el número, después de su elección devuelve una lista o dato que se eligió.

**I** If e elif de opciones que se elijan que mandan a su respectiva función y devuelven una lista o dato impreso en pantalla.

```

72
73
74     elif eleccion == '8':
75         print("\n----- Total anual y meses con mas ventas. -----")
76         Functions.checks.total_earnings()
77         Functions.checks.month_sales()
78
79     else:
80         print("Error. Usuario o contraseña incorrecto, intenta nuevamente.\n")
81
82     elif choose == "n" or choose == "no":
83         while True:
84             reg_name = input("Ingresa nombre de usuario de tu preferencia: ")
85
86             while True:
87                 reg_pass = getpass.getpass("Ingresa contraseña de usuario de tu elección: ")
88                 check_pass = getpass.getpass("Confirma la contraseña: ")
89
90                 if reg_pass == check_pass:
91                     break
92             else:
93                 print("Error. Contraseñas no coinciden, intenta nuevamente.\n")
94
95             if Functions.login.user_creation(reg_name, reg_pass):
96                 print("Usuario creado exitosamente.")
97                 sleep(2)
98                 clear()
99                 break
100
101     else:
102         print("Error. Usuario ya existe, intenta de nuevo.\n")
103
104     else:
105         print("Error. Ingrese un valor correcto.\n")

```

**E** Elif para registro de usuario (Si al inicio se selecciono que no cuenta con un usuario).

Si las contraseñas no coinciden regresa al inicio de su While True, si coinciden rompe este ciclo.

Crea el usuario y la contraseña, si el usuario ya existe regresa un False (Regresa al While de registro de usuario) si el usuario no existe, lo crea y regresa al inicio del programa.

Solucion del problema.

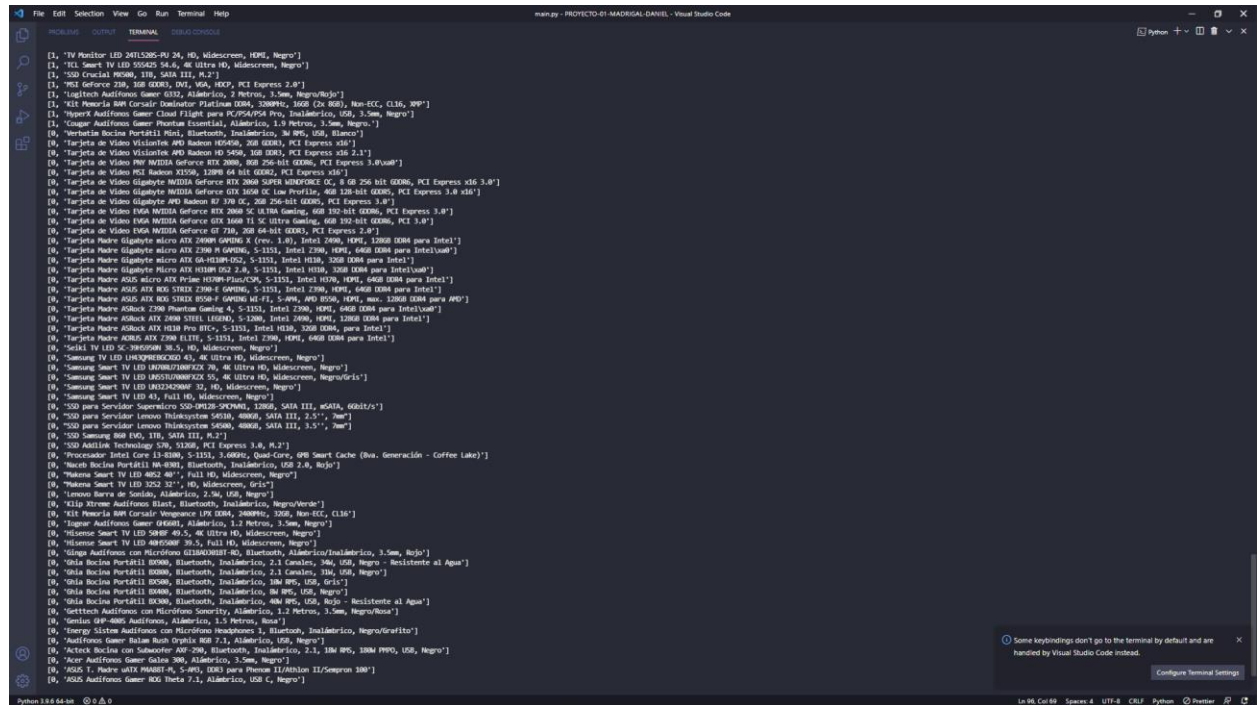
```
----- Productos con peores resenas -----
Selecciona la cantidad de productos que desea mostrar: 100
['0.00', 'ASUS Audifonos Gamer ROG Theta 7.1, Alámbrico, USB C, Negro']
['0.00', 'ASUS T. Madre uATX MAA88T-M, S-AM3, DDR3 para Phenom II/Athlon II/Sempron 100']
['0.00', 'Acer Audifonos Gamer Galea 300, Alámbrico, 3.5mm, Negro']
['0.00', 'Ateck Bocina con Subwoofer AXF-290, Bluetooth, Inalámbrico, 2.1, 18W RMS, 180W PMPO, USB, Negro']
['0.00', 'Audifonos Gamer Balam Rush Orphix RGB 7.1, Alámbrico, USB, Negro']
['0.00', 'Energy Sistem Audifonos con Micrófono Headphones 1, Bluetooth, Inalámbrico, Negro/Grafito']
['0.00', 'Genius GHP-400S Audifonos, Alámbrico, 1.5 Metros, Rosa']
['0.00', 'Getttech Audifonos con Micrófono Sonority, Alámbrico, 1.2 Metros, 3.5mm, Negro/Rosa']
['0.00', 'Ghia Bocina Portátil BX300, Bluetooth, Inalámbrico, 40W RMS, USB, Rojo - Resistente al Agua']
['0.00', 'Ghia Bocina Portátil BX400, Bluetooth, Inalámbrico, 8W RMS, USB, Negro']
['0.00', 'Ghia Bocina Portátil BX500, Bluetooth, Inalámbrico, 10W RMS, USB, Gris']
['0.00', 'Ghia Bocina Portátil BX800, Bluetooth, Inalámbrico, 2.1 Canales, 31W, USB, Negro']
['0.00', 'Ghia Bocina Portátil BX900, Bluetooth, Inalámbrico, 2.1 Canales, 34W, USB, Negro - Resistente al Agua']
['0.00', 'Ginga Audifonos con Micrófono GI18A0J018T-RO, Bluetooth, Alámbrico/Inalámbrico, 3.5mm, Rojo']
['0.00', 'Hisense Smart TV LED 40H5500F 39.5, Full HD, Widescreen, Negro']
['0.00', 'Hisense Smart TV LED 50H8F 49.5, 4K Ultra HD, Widescreen, Negro']
['0.00', 'Hogear Audifonos Gamer GH6601, Alámbrico, 1.2 Metros, 3.5mm, Negro']
['0.00', 'Kit Memoria RAM Corsair Vengeance LPX DDR4, 2400MHz, 32GB, Non-ECC, CL16']
['0.00', 'Klip Xtreme Audifonos Blast, Bluetooth, Inalámbrico, Negro/Verde']
['0.00', 'Lenovo Barra de Sonido, Alámbrico, 2.5W, USB, Negro']
['0.00', 'Makena Smart TV LED 32S2 32'', HD, Widescreen, Gris']
['0.00', 'Makena Smart TV LED 40S2 40'', Full HD, Widescreen, Negro']
['0.00', 'Naceb Bocina Portátil NA-0301, Bluetooth, Inalámbrico, USB 2.0, Rojo']
['0.00', 'Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generación - Coffee Lake)']
['0.00', 'SSD Addlink Technology S70, 512GB, PCI Express 3.0, M.2']
['0.00', 'SSD Samsung 860 EVO, 1TB, SATA III, M.2']
['0.00', 'SSD para Servidor Lenovo Thinksystem S4500, 480GB, SATA III, 3.5'', 7mm']
['0.00', 'SSD para Servidor Lenovo Thinksystem S4510, 480GB, SATA III, 2.5'', 7mm']
['0.00', 'SSD para Servidor Supermicro SSD-DM128-SMCMW1, 128GB, SATA III, mSATA, 6Gb/s']
['0.00', 'Samsung Smart TV LED 43, Full HD, Widescreen, Negro']
['0.00', 'Samsung Smart TV LED UN32J4290AF 32, HD, Widescreen, Negro']
['0.00', 'Samsung Smart TV LED UN55TU7000FXZX 55, 4K Ultra HD, Widescreen, Negro/Gris']
['0.00', 'Samsung Smart TV LED UN70RU7100FXZX 70, 4K Ultra HD, Widescreen, Negro']
['0.00', 'Samsung TV LED LH43QMREBGCXG0 43, 4K Ultra HD, Widescreen, Negro']
['0.00', 'Seiki TV LED SC-39H5950N 38.5, HD, Widescreen, Negro']
['0.00', 'Tarjeta Madre AORUS ATX Z390 ELITE, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel']
['0.00', 'Tarjeta Madre ASRock ATX Z490 STEEL LEGEND, S-1200, Intel Z490, HDMI, 128GB DDR4 para Intel']
['0.00', 'Tarjeta Madre ASRock Z390 Phantom Gaming 4, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel']
['0.00', 'Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING MI-FI, S-AM4, AMD B550, HDMI, max. 128GB DDR4 para AMD']
['0.00', 'Tarjeta Madre ASUS ATX ROG STRIX Z390-E GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel']
['0.00', 'Tarjeta Madre ASUS micro ATX Prime H370M-Plus/CSM, S-1151, Intel H370, HDMI, 64GB DDR4 para Intel']
['0.00', 'Tarjeta Madre Gigabyte Micro ATX H310M DS2 2.0, S-1151, Intel H310, 32GB DDR4 para Intel']
['0.00', 'Tarjeta Madre Gigabyte micro ATX Z390 M GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel']
['0.00', 'Tarjeta Madre Gigabyte micro ATX Z490M GAMING X (rev. 1.0), Intel Z490, HDMI, 128GB DDR4 para Intel']
['0.00', 'Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3, PCI Express 2.0']
['0.00', 'Tarjeta de Video EVGA NVIDIA GeForce GTX 1660 Ti SC Ultra Gaming, 6GB 192-bit GDDR6, PCI 3.0']
['0.00', 'Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming, 6GB 192-bit GDDR6, PCI Express 3.0']
['0.00', 'Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile, 4GB 128-bit GDDR5, PCI Express 3.0 x16']
['0.00', 'Tarjeta de Video Gigabyte NVIDIA GeForce RTX 2060 SUPER WINDFORCE OC, 8 GB 256 bit GDDR6, PCI Express x16 3.0']
['0.00', 'Tarjeta de Video MSI Radeon X1550, 128MB 64 bit GDDR2, PCI Express x16']
['0.00', 'Tarjeta de Video PNY NVIDIA GeForce RTX 2080, 8GB 256-bit GDDR6, PCI Express 3.0']
['0.00', 'Tarjeta de Video VisionTek AMD Radeon HD 5450, 1GB DDR3, PCI Express x16 2.1']
['0.00', 'Tarjeta de Video VisionTek AMD Radeon HD5450, 2GB GDDR3, PCI Express x16']
['0.00', 'Verbatim Bocina Portátil Mini, Bluetooth, Inalámbrico, 3W RMS, USB, Blanco']
['1.00', 'Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel']
['1.00', 'Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0']
['1.83', 'Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD']
['2.00', 'Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel']
['3.00', 'Cougar Audifonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro']
['4.00', 'HyperX Audifonos Gamer Cloud Flight para PC/PS4/PS4 Pro, Inalámbrico, USB, 3.5mm, Negro']
['4.00', 'MSI GeForce 210, 1GB GDDR3, DVI, VGA, HDCP, PCI Express 2.0']
['4.00', 'Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0']
['4.14', 'Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD']
```

Se recomienda retirar del mercado:

1. Tarjeta Madre ASRock ATXH110 PRO BTC+.
2. Tarjeta de Video Gigabyte AMD Radeon R7 370 OC.
3. Tarjeta Madre AORUS micro ATX B450 AORUS M.
4. Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2.

Ya que son los artículos que han sido comprados y han recibido una baja calificación.

Ademas existen 57 articulos que nunca han sido comprados.



```
[1] "TV Panelizer LED 24115206-P1 24", HD, Midscreen, HDMI, Negro"]
[2] "TV Smart TV LED 555425 54.6", 4K Ultra HD, Midscreen, Negro"]
[3] "SSD Crucial PH100 1TB", SATA III, M.2"]
[4] "MSI GeForce Z89, 16G GDDR5, DVI, VGA, HPC, PCI Express 2.0"]
[5] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[6] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[7] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[8] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[9] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[10] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[11] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[12] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[13] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[14] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[15] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[16] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[17] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[18] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[19] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[20] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[21] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[22] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[23] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[24] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[25] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[26] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[27] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[28] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[29] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[30] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[31] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[32] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[33] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[34] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[35] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[36] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[37] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[38] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[39] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[40] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[41] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[42] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[43] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[44] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[45] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[46] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[47] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[48] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[49] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[50] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[51] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[52] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[53] "Logitech AudioFocus Gamer G332, Alámbrico, 2 Pétros, 3.5mm, Negro/Blanco"]
[54] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
[55] "NVIDIA GeForce RTX 3080 Ti, 10GB GDDR6X, PCI Express 4.0"]
[56] "Gigabyte Aorus Gaming 7 Pro, 16GB DDR4, 2666MHz, 2x 8GB, 1.5 Petros, 3.5mm, Negro"]
[57] "Memoria RAM Corsair Dominator Platinum DDR4, 32GB, 2880MHz (2x 16G), Non-ECC, CL16, 10P"]
```

Y por lo visto la mayoría de estos artículos son un derivado de otros artículos, así que se recomienda bajar las variantes de un artículo y tener en inventario artículos mas generales sin crear demasiadas opciones, ya que esta acumulación de artículos no comprados se puede deber a la paradoja de elección.

“Según un estudio de la Universidad de Columbia elaborado por Ivengar y Lepper (2000) ha demostrado que la capacidad de gestión humana ante la toma de decisión sobre múltiples opciones es limitada. Es mejor tener opciones, pero hasta cierto punto. En sus investigaciones, expusieron a un grupo de personas a la decisión de comprar mermeladas gourmet y chocolate. El grupo de personas a las que se les hizo decidir entre 6 tipos de mermeladas tuvo un porcentaje mucho mayor de compra final del producto (12%) que aquellos que podían elegir entre 24 o hasta 30 opciones (2%).”

Artículo completo: <https://psicopico.com/la-paradoja-de-eleccion-muchas-opciones-pocas-decisiones/>.

```
----- Bienvenido a Lifestore Application -----

(1) Productos con mayores ventas.
(2) Productos con mayores ganancias.
(3) Productos con menores ventas por categoría.
(4) Productos con menores ganancias por categoría.
(5) Productos con mejores resúmenes.
(6) Productos con peores resúmenes.
(7) Total de ingresos y ventas promedio mensuales.
(8) Total anual y meses con mas ventas.
Seleccione una opción (Número): 8

----- Total anual y meses con mas ventas. -----
Total de ingresos anual: $739266
Meses con mas ventas
(Número de ventas, Nombre de mes)
(29, 4)
(19, 12)
(19, 3)
(41, 2)
(16, 1)
(11, 7)
(11, 6)
(1, 5)
(1, 11)
(1, 9)
(0, 12)
(0, 10)

(1) Productos con mayores ventas.
(2) Productos con mayores ganancias.
(3) Productos con menores ventas por categoría.
(4) Productos con menores ganancias por categoría.
(5) Productos con mejores resúmenes.
(6) Productos con peores resúmenes.
(7) Total de ingresos y ventas promedio mensuales.
(8) Total anual y meses con mas ventas.
Seleccione una opción (Número):
```

Para finalizar los meses de final de año (9,10,11,12) suelen ser los meses con menores compras por decir que estos en algunos de estos meses no cuentan con ninguna compra, así que no se recomienda tener mucho inventario en estos meses o crear una estrategia de marketing para mejorar el flujo de ventas en estos meses.

Conclusión.

Productos mayores ventas.







[illegible]

### Productos con menor calificación

[illegible]

Total de ingresos y promedio de ventas mensuales.



```
main.py - PROYECTO-01-MADRIGAL-DANIEL - Visual Studio Code

[1.00], "Tarjeta de Video PCI Radeon RX590, 12000 64 bit GDDR5, PCI Express x8"
[1.00], "Tarjeta de Video PCI NVIDIA GeForce RTX 2080, 8GB 256-bit GDDR6, PCI Express 3.0(x8)"
[1.00], "Tarjeta de Video Visioline APD Radeon HD 5450, 1GB DDR3, PCI Express x16 2.1"
[1.00], "Tarjeta de Video Visioline APD Radeon HD5450, 2GB GDDR3, PCI Express x16"
[1.00], "Verbatim Racine PortB11 Mini, Bluetooth, Inalámbrico, 34 MP, USB, Blanco"
[1.00], "Tarjeta Madre AIOck A10 H30 Pro 8TC, 5-1151, Intel H30, 1GB DDR3, para Intel"
[1.00], "Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0"
[1.00], "Tarjeta Madre ASUS Micro ATX B450 Aorus M (rev. 1.0), 5-AM, AMD B450, HDMI, 64GB DDR4 para AMD"
[1.00], "Tarjeta Madre Gigabyte micro ATX G41M3P-S2, 5-1151, Intel H30, 1GB DDR3 para Intel"
[1.00], "Casper Audifonos Gamer Phantom Essential, Alámbrico, 1.9 Petros, 3.5mm, Negro"
[1.00], "HyperX Audifonos Gamer Cloud Flight para PC/PS4/PS4 Pro, Inalámbrico, USB, 3.5mm, Negro"
[1.00], "PCI GeForce 730, 1GB GDDR3, DVI, VGA, HDMI, PCI Express 2.0"
[1.00], "Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix, 4GB 128-bit GDDR5, PCI Express 3.0"
[1.00], "Tarjeta Madre ASUS Micro ATX B450-PLUS Gaming, 5-AM, AMD B450, HDMI, 64GB DDR4 para AMD"
[1.00], "Procesador AMD Ryzen 5 3600, 5-AM, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth"
[1.00], "Tarjeta de Video Gigabyte NVIDIA GeForce GT 1030, 2GB 64-bit GDDR5, PCI Express x16 3.0"
[1.00], "Procesador AMD Ryzen 3 3200G con Gráfico Radeon Vega 8, 5-AM, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire"
[1.00], "Logitech Racine para Computadora con Subwoofer G560, Bluetooth, Inalámbrico, 2.1, 120W RMS, USB, negro"
[1.00], "Tarjeta Madre ASUS ATX B450, 5-1151, Intel 3900, HDMI, 64GB DDR4 para Intel/AMD"
[1.00], "SSD M.2 NVMe 500GB Pro, 256GB, PCI Express, 8.2"
[1.00], "Tarjeta Madre AIOck Micro ATX B450M Steel Legend, 5-AM, AMD B450, HDMI, 64GB DDR4 para AMD"
[1.00], "SSD Kingston A800, 1TB, PCI Express 3.0, M.2"
[1.00], "SSD Kingston UV500, 480GB, SATA III, eSATA"
[1.00], "Tarjeta Madre PCI ATX B450 TUF Gaming, 5-AM, AMD B450, 64GB DDR4 para AMD"
[1.00], "Procesador Intel Core i3-10100F, 5-1151, 3.60GHz, Quad-Core, 9MB Cache (No. Generación - Coffee Lake)"
[1.00], "SSD Kingston A800, 120GB, SATA III, 2.5", 7mm"
[1.00], "Tarjeta de Video Asus NVIDIA GeForce GTX 1660 SUPER D6 OC, 6GB 128-bit GDDR6, PCI Express x16 3.0"
[1.00], "Procesador AMD Ryzen 5 3600, 5-AM, 3.60GHz, Six-Core, 32MB L3 Cache, con Disipador Wraith Stealth"
[1.00], "SSD Adata Ultimate 500GB, 256GB, SATA III, 2.5", 7mm"
[1.00], "Kit Memoria RAM Corsair Dominator Platinum D8, 32GB, 3000MHz, 16GB (2x 8GB), Non-ECC, CL16, 30P"
[1.00], "Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm"

[1] Productos con mayores ventas.
[2] Productos con mayores ganancias.
[3] Productos con menores ventas por categoría.
[4] Productos con menores ganancias por categoría.
[5] Productos con mejores resúmenes.
[6] Productos con peores resúmenes.
[7] Total de ingresos y ventas promedio mensuales.
[8] Total anual y meses con mas ventas.
Selecciona una opción (Número): 7

----- Total de Ingresos y ventas promedio mensuales -----
(Número de ventas, total de ingresos, número de mes)
(1), 12032, 1)
(4), 11813, 2)
(1), 16729, 3)
(5), 15256, 4)
(6), 9639, 5)
(1), 1694, 6)
(1), 2698, 7)
(1), 3077, 8)
(1), 4599, 9)
(0, 0, 10)
(1), 5009, 11)
(0, 0, 12)

[1] Productos con mayores ventas.
[2] Productos con mayores ganancias.
[3] Productos con menores ventas por categoría.
[4] Productos con menores ganancias por categoría.
[5] Productos con mejores resúmenes.
[6] Productos con peores resúmenes.
[7] Total de ingresos y ventas promedio mensuales.
[8] Total anual y meses con mas ventas.
Selecciona una opción (Número): 8

----- Total anual y meses con mas ventas. -----
Total de ingresos anual: $75786
Meses con mas ventas
(Número de ventas, Número de mes)
(75, 4)
(13, 1)
(13, 3)
(41, 2)
(16, 1)
(11, 7)
(11, 6)
(1, 8)
(1, 11)
(1, 9)
(0, 12)
(0, 10)

[1] Productos con mayores ventas.
[2] Productos con mayores ganancias.
[3] Productos con menores ventas por categoría.
[4] Productos con menores ganancias por categoría.
[5] Productos con mejores resúmenes.
[6] Productos con peores resúmenes.
[7] Total de ingresos y ventas promedio mensuales.
[8] Total anual y meses con mas ventas.
Selecciona una opción (Número): 8
```

Total anual y meses con mas ventas.

```
main.py - PROYECTO-01-MADRIGAL-DANIEL - Visual Studio Code

[1.00], "Procesador AMD Ryzen 5 3600, 5-AM, 3.60GHz, Six-Core, 32MB L3 Cache, con Disipador Wraith Stealth"
[1.00], "SSD Adata Ultimate 500GB, 256GB, SATA III, 2.5", 7mm"
[1.00], "Kit Memoria RAM Corsair Dominator Platinum D8, 32GB, 3000MHz, 16GB (2x 8GB), Non-ECC, CL16, 30P"
[1.00], "Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm"

[1] Productos con mayores ventas.
[2] Productos con mayores ganancias.
[3] Productos con menores ventas por categoría.
[4] Productos con menores ganancias por categoría.
[5] Productos con mejores resúmenes.
[6] Productos con peores resúmenes.
[7] Total de ingresos y ventas promedio mensuales.
[8] Total anual y meses con mas ventas.
Selecciona una opción (Número): 7

----- Total de Ingresos y ventas promedio mensuales -----
(Número de ventas, total de ingresos, número de mes)
(1), 12032, 1)
(4), 11813, 2)
(1), 16729, 3)
(5), 15256, 4)
(6), 9639, 5)
(1), 1694, 6)
(1), 2698, 7)
(1), 3077, 8)
(1), 4599, 9)
(0, 0, 10)
(1), 5009, 11)
(0, 0, 12)

[1] Productos con mayores ventas.
[2] Productos con mayores ganancias.
[3] Productos con menores ventas por categoría.
[4] Productos con menores ganancias por categoría.
[5] Productos con mejores resúmenes.
[6] Productos con peores resúmenes.
[7] Total de ingresos y ventas promedio mensuales.
[8] Total anual y meses con mas ventas.
Selecciona una opción (Número): 8

----- Total anual y meses con mas ventas. -----
Total de ingresos anual: $75786
Meses con mas ventas
(Número de ventas, Número de mes)
(75, 4)
(13, 1)
(13, 3)
(41, 2)
(16, 1)
(11, 7)
(11, 6)
(1, 8)
(1, 11)
(1, 9)
(0, 12)
(0, 10)

[1] Productos con mayores ventas.
[2] Productos con mayores ganancias.
[3] Productos con menores ventas por categoría.
[4] Productos con menores ganancias por categoría.
[5] Productos con mejores resúmenes.
[6] Productos con peores resúmenes.
[7] Total de ingresos y ventas promedio mensuales.
[8] Total anual y meses con mas ventas.
Selecciona una opción (Número): 8
```

Para la realización de este proyecto no se contaba antes con conocimientos avanzados ni intermedios en el lenguaje de programación Python, pero se utilizó la lógica de programación aprendida en otros lenguajes y con la experiencia. Programar en Python es algo de lo mas divertido

que puede existir, ya que al ser un lenguaje de programación de alto nivel permite muchas variantes y hace todo mas intuitivo.