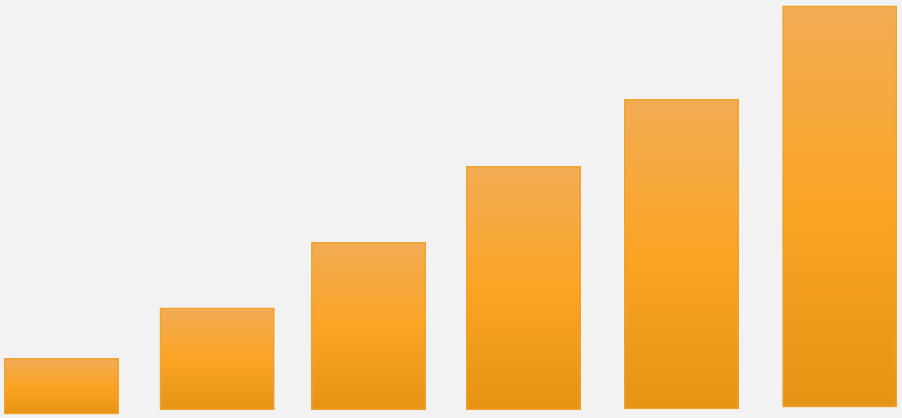
PROYECTO 1

INTRODUCCIÓN A PYTHON



EMTECH STUDENT

JESÚS ISMAEL

PINEDA HERNÁNDEZ

Enlace al repositorio:

DATA-SCIENCE-EMTECH/REPORTE_01_PINEDA_JESUS.ipynb at main · ispineda/DATA-SCIENCE-EMTECH (github.com)

ÍNDICE

INTRODUCCIÓN

Durante el siguiente reporte se practican los conceptos aprendidos durante el curso **Fundamentos de Programación con Python** para reforzar la creación y manejo de variables, textos, listas, índices, inputs, operadores lógicos, operadores relacionales, operadores de asignación, condiciones if, bucles while, bucles for y control de bucles.

Para dicho reporte se ha descrito un **caso práctico** donde se solicita crear un sistema con inicio de sesión acompañado de la generación de reportes de una tienda virtual llamada **LifeStore**. La información de la tienda se obtuvo mediante el siguiente enlace.

DESCRIPCIÓN DELCASO

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.

Derivado de la situación se solicita un análisis de la rotación de productos para identificar los siguientes elementos:

***Nota**: Se han agregado otros elementos para simplificar la visualización de datos en el sistema.*

1. Productos más vendidos y productos rezagados a partir del análisis de las categorías con menores ventas y categorías con menores búsquedas.

    - Enlistar los 50 productos con mayores ventas y los 100 productos con mayores búsquedas.
    - Enlistar los 50 productos con menores ventas y 100 productos con menores búsquedas.
    - *Listado por categoría de todos los productos ordenados por mayores ventas.*
    - *Listado por categoría de todos los productos ordenados por mayores búsquedas.*

2. Productos por reseña en el servicio a partir del análisis de categorías con mayores ventas y categorías con mayores búsquedas.

    - Enlistar 20 productos que cuentan con las mejores reseñas y 20 productos que cuentan con las peores reseñas considerando las devoluciones.
    - *Listado por categoría de todos los productos ordenados por mayores ventas.*
    - *Listado por categoría de todos los productos ordenados por mayores búsquedas.*

3. Estrategia de los productos a retirar del mercado, así estrategia para reducir el inventario acumulado considerando los ingresos y ventas mensuales.

    - Ventas promedio mensuales.
    - Ventas anuales.
    - Total de ingresos.
    - Meses con más ventas al año.

DEFINICIÓN DEL CÓDIGO

La realización del código se hizo con la herramienta *Google Colab* para detallar de mejor manera los puntos solicitados por la Gerencia de ventas, asimismo, se realizaron funciones (módulos) para el control general de la información y el flujo del sistema para mostrar los reportes; dicha solución se puede encontrar en el siguiente repositorio de **GitHub** o el **enlace que se proporciona en la portada.**

El código se dividió en 3 secciones principales y un apartado para correr el sistema:

Las secciones principales son:

1. Carga de datos proporcionados por la tienda.
2. Funciones para modificar matrices y columnas.
3. Seguimiento de consignas solicitadas.

El apartado para correr el sistema es:

4. Sistema

1. CARGA DE DATOS PROPORCIONADOS

Se cargan las listas lifestore_products, lifestore_sales y lifestore_searches.

```
"""
This is the LifeStore_SalesList data:
lifestore_products = [id_product, name, price, category, stock]
lifestore_searches = [id_search, id product]
lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for true or 0 to false)]
"""

lifestore_products = [···
]

lifestore_sales = [···
]

lifestore_searches = [···
]
```

2. FUNCIONES PARA MODIFICAR MATRICES Y COLUMNAS

Se crearon funciones para poder manipular de mejor manera los datos proporcionados y poder reutilizarlas en caso de extender el análisis.

FUNCIONES PARA MODIFICAR MATRICES E IMPRIMIRLAS

```python
# ================================================== #
# Crea matriz con nombres de columnas
# ================================================== #
def data(matrix, names = []):

  ''' Función para generar una matriz con columnas
  identificables mediante nombres '''

  # Genera matriz transpuesta
  transposte = list(map(list, zip(*matrix)))

  # Define dimensiones de las listas
  long_columns = len(transposte)
  long_names = len(names)

  # Asigna nombres a las columnas
  while long_names < long_columns:
    long_names += 1
    names.append('col_' + str(long_names))

  # Recorta vector de nombres en caso de necesitarse
  if long_names > long_columns and long_columns != 0:
    names = names[:long_columns]

  return [names, transposte] # Retorna nueva matriz
```

```python
# ================================================== #
# Muestra matriz en forma de tabla
# ================================================== #
def show(matrix, num_space = 12):

  ''' Muestra la matriz de forma ordenada '''

  header, columns = matrix.copy() # Asegura matriz original
  transposte_columns = list(map(list, zip(*columns))) # Transpone matriz

  # Imprime cabecera
  print('item | ', end = ' ')
  for name in header:
    print(f'{name} |', end = ' ')
  print('')

  # Imprime tabla ordenada
  for i, row in enumerate( transposte_columns ):
    print(f"{ i+1 }-", end = ' ')

    # Convierte en texto cualquier elementoen la matriz
    for value in row:
      value = str(value)
      if len(value) > num_space:
        value = value[:num_space]

      while len(value) < num_space:
        value = ' ' + value

      print(f'{ value }', end = ' ')
    print('', flush = True)
```

```python
# ================================================== #
# Muestra dimensiones de la tabla
# ================================================== #
def dim(matrix):

  ''' Obtiene las dimensiones de la matriz '''

  header, columns = matrix.copy()
  # Retorna las dimensiones de la matriz
  if not len(columns) == 0:
      return (len(columns[0]) , len(header))
  return (0, len(header))
```

```python
# ================================================== #
# Retorna número de filas especificadas de la tabla
# ================================================== #
def head( matrix, number ):

  ''' Obtiene el número de filas especificadas de la matriz'''

  number_rows = dim( matrix )[0] # Número de filas de la matriz

  # Ajusta número de filas solicitadas con las existentes
  if number_rows < number:
      number = number_rows

  matrix_copy = toMatrix( matrix ) # Genera transpuesta
  matrix_copy = matrix_copy[:number] # Reduce matriz

  return data( matrix_copy, names = matrix[0] ) # Retorna matriz
```

```python
# ================================================== #
# Convierte matriz con nombres en matriz simple
# ================================================== #
def toMatrix(matrix):

  ''' Retorna un matriz simple sin cabecera '''

  _, columns = matrix.copy() # Asegura matriz original
  return list(map(list, zip(*columns))) # Retorna la matriz transpuesta
```

FUNCIONES PARA MANIPULAR COLUMNAS (LISTAS)

```python
# ================================================== #
# Obtiene vector de la tabla
# ================================================== #
def column(matrix, name):

  ''' Retorna la columna solicitada de una matriz '''

  header, columns = matrix.copy()
  return columns[header.index(name)] # Retorna la columna
```

```python
# ================================================== #
# Encuentra valores únicos de un vector
# ================================================== #
def unique(vector):

  ''' Genera vector con valores únicos '''

  unique_values = [] # Vector que guarda nuevos valores únicos
  [unique_values.append(element) for element in vector if element not in unique_value
  return unique_values # Retorna vector
```

```python
# ================================================== #
# Separa elementos del vector y los convierte en tabla
# ================================================== #
def split_column(vector, separate = '/', names = []):

  ''' Separa la columna mediante el caracter deseado '''

  # Longitud de la cabecera
  long_header = len(names)

  # Almacena la nueva matriz
  matrix = []
  # Separa los elementos del vector
  for element in vector:
      if not long_header == 0:
          split_element = element.split(separate, long_header - 1)
      else:
          split_element = element.split(separate)

      matrix.append(split_element) # Agrega los elementos a la matriz

  return data(matrix, names) # Retorna matriz modificada
```

```python
# ================================================== #
# Rellena elementos None por valor especificado
# ================================================== #
def fillnone(matrix, name, new_value):

  ''' Rellena valores None de una columna '''

  # Obtiene cabecera y posición de la columna
  header, _= matrix.copy()
  position = header.index(name)

  # Protege matriz original
  matrix_copy = toMatrix(matrix.copy())

  # Reemplaza valor
  for i, row in enumerate(matrix_copy):
    value = row[position]
    if value == None:
      matrix_copy[i][position] = new_value

  return data(matrix_copy, header) # Retorna matriz
```

```python
# ================================================== #
# Resta dos vectores
# ================================================== #
def subt_vectors(vector1, vector2):

  ''' Resta dos vectores dados '''

  # Resta dos listas y guarda el resultado
  res_vector = list(map(lambda x, y: x - y, vector1, vector2))
  return res_vector
```

```python
# ================================================== #
# Multipla dos vectores
# ================================================== #
def multi_vectors(vector1, vector2):

  ''' Multiplica dos vectores dados '''

  # Multiplica dos listas y guarda el resultado
  multi_vector = list(map(lambda x, y: x * y, vector1, vector2))
  return multi_vector
```

FUNCIONES PARA REDIMENSIONAR MATRICES

```
# ================================================== #
# Agrega una columna nueva a la tabla especificada
# ================================================== #
def add_column(matrix, name, vector = []):

  ''' Agrega columna a matriz mediante un vector'''

  header, columns = matrix.copy() # Asegura la matriz original
  header.append(name) # Apila nombre de la columna

  # Obtiene longitudes de las columnas y el vector a incrustar
  long_column = len(columns[0])
  long_vector = len(vector)

  # Rellena espacios vacíos
  while long_column > long_vector:
      vector.append(None)
      long_vector = len(vector)

  # Recorta nombres excedentes
  if long_column < long_vector:
      vector = vector[:long_column]

  columns.append(vector)  # Apila columna nueva

  return [header, columns] # Retorna matriz modificada
```

```
# ================================================== #
# Elimina columna especificada de una tabla
# ================================================== #
def drop_column(matrix, name):

  ''' Elimina la columna especificada de la matriz '''

  header, columns = matrix.copy() # Asegura la matriz original

  if name in header:
    position = header.index(name) # Obtiene posición de la columna

    # Elimina la columna y nombre de cabecera
    header.pop(position)
    columns.pop(position)

  return [header, columns] # Genera nueva matriz
```

```
 # ================================================== #
 # Une dos tablas - Columnas
 # ================================================== #
 def expand( matrix1, matrix2 ):

   ''' Une columnas de dos matrices '''

   # Asegura las matrices originales
   header1, columns1 = matrix1.copy()
   header2, columns2 = matrix2.copy()

   # Retorna matriz extendida
   return [header1 + header2, columns1 + columns2]
```

```
# ================================================== #
# Elimina multiples columnas de una tabla
# ================================================== #
def drop(matrix, names = []):

  ''' Elimina las columnas especificadas de la matriz '''

  matrix_copy = matrix.copy() # Asegura matriz original

  #Elimina iterativamente las columnas especificadas
  for name in names:
    matrix_copy = drop_column(matrix, name)

  return matrix_copy # Retorna matriz modificada
```

FUNCIONES PARA MANIPULAR MATRICES

```python
# ================================================== #
# Filtra filas de una matriz con valores especificados
# ================================================== #
def filter_by( matrix, name, by=[] ):

  ''' Filtra filas para valores encontrado '''

  header, columns = matrix.copy() # Asegura matriz

  if name in header:
    # Obtiene la posición del nombre en cabecera
    position = header.index(name)

    # Crea vector con las posiciones donde existe el elemento
    list_founded = []
    for id, element in enumerate(columns[position]):
      if not element in by:
          list_founded.append(id)

    # Invierte el vector para no alterar posiciones
    list_founded = list_founded[::-1]

    # Elimina los elementos mediante las posiciones de las columnas
    for column in columns:
      for element in list_founded:
          column.pop(element)

  return [header, columns] # Retorna matriz modificada
```

```python
# ================================================== #
# Filtra filas de una matriz con un valor especificado
# ================================================== #
def filter_value(matrix, name, by, not_value = False):

  ''' Filtrar fila por valor encontrado '''

  matrix_copy = matrix.copy() # Asegura matriz original
  # Rescata cabecera y columnas de la matriz
  header, columns = matrix_copy

  if name in header:
    # Obtiene posición de columna
    position = header.index(name)

    # Enlista la posición donde se encuentra el valor
    found_id = []
    for id, element in enumerate(columns[position]):
      if not_value:
        if element != by:
          found_id.append(id)
      else:
        if element == by:
          found_id.append(id)

    # Convierte a matriz simple sin cabecera
    matrix_copy = toMatrix(matrix_copy)
    # Crea nueva matriz con valores
    matrix_new = []
    for id, element1 in enumerate(matrix_copy):
      for element2 in found_id:
          if element2 == id:
              matrix_new.append(element1)

    return data(matrix_new, names = header) # Retorna matriz

  return matrix_copy  # Retorna matriz original
```

```python
# ================================================== #
# Ordena por columna de manera ascendente o descendente
# ================================================== #
def sort_by( matrix, name, ascending = False ):

  ''' Ordena matriz a través de una columna '''

  header, columns = matrix.copy() # Asegura matriz original
  position = header.index(name) # Obtiene posición de columna
  long_header = len(header)

  # Genera vector único y ordenada de la columna seleccionada
  list_sort = sorted(unique(columns[position]) , reverse = not ascending)

  # Busca elementos de manera ordenada para acomodarlos
  sorted_list = []
  for value1 in list_sort:
    for i, value2 in enumerate(columns[position]):
        if value1 == value2:
            sorted_list.append(i)

  matrix_copy = matrix.copy()
  matrix_transposte = toMatrix(matrix_copy)

  sorted_matrix = []

  for item in sorted_list:
    sorted_matrix.append(matrix_transposte[item])

  return data(sorted_matrix, header[:long_header])
```

```python
# ================================================== #
# Cuenta el número de repeticiones de un valor en una tabla
# ================================================== #
def value_counts(matrix, name, name_output):

  ''' Cuenta valores repetidos de una columna '''

  header, columns = matrix.copy() # Asegura la matriz original
  position = header.index(name) # Obtiene la posición de la columna

  # Crea vector de valores únicos de la columna
  values_uniques = unique(columns[position])

  # Recorre y cuenta los valores
  counts_values = []
  for value1 in values_uniques:
    count = 0
    for value2 in columns[position]:
      if value1 == value2:
        count+=1
    counts_values.append(count) # Apila el conteo

  # Devuelve matriz modificada
  return [[name, name_output] ,[values_uniques, counts_values]]
```

```python
# ========================================================== #
# Crea intersección de tablas
# ========================================================== #
def merge(matrix1, matrix2, left_index, right_index):

  ''' Une dos matrices a través de las columnas especificadas (crea unión) '''
  header1, columns1 = matrix1.copy()
  header2, columns2 = matrix2.copy()

  position1 = header1.index(left_index)
  position2 = header2.index(right_index)

  values_uniques_1 = unique(columns1[position1])
  values_uniques_2 = unique(columns2[position2])

  # Obtiene valores repetidos en ambos vectores
  same_values = []
  [same_values.append(element) for element in values_uniques_1 if element in values_uniques_2]

  # Filtra matriz izquierda con los valores compartidos
  matrix1_copy = filter_by(matrix1, left_index, by = same_values)
  matrix2_copy = matrix2.copy()   # Asegura matriz 2

  # Obtiene cabeceras
  header1, _= matrix1_copy
  header2, _= matrix2_copy

  # Obtiene posición de las cabeceras
  position1 = header1.index(left_index)
  position2 = header2.index(right_index)

  # Transpone matrices para manipularlas
  matrix1_copy = toMatrix(matrix1_copy)
  matrix2_copy = toMatrix(matrix2_copy)

  # Agrega elementos a los vectores de filas
  for row1 in matrix1_copy:
    for row2 in matrix2_copy:
      if row1[position1] == row2[position2]:
        row1 += row2

  new_data = data( matrix1_copy, header1 + header2 )
  return drop(new_data , names = [left_index]) # Elimina columnas
```

```python
# ========================================================== #
# Une elementos de tabla 1 con elementos de tabla 2
# ========================================================== #
def join(matrix1, matrix2, on):

  ''' Función para unir matrices '''

  # Obtiene cabeceras de las matrices
  header1, _= matrix1.copy()
  header2, _= matrix2.copy()

  # Encuentra la columna para realizar el join
  position1 = header1.index( on )
  position2 = header2.index( on )

  header2[position2] = on + '_right'

  # Transpone la información
  matrix1_aux = toMatrix( matrix1.copy() )
  matrix2_aux = toMatrix( matrix2.copy() )

  long_matrix1_aux = len(matrix1_aux)
  # Une matrices a través de columna
  matrix_new = []
  for item, row1 in enumerate(matrix1_aux):
    matrix_new.insert(item, row1)

    for row2 in matrix2_aux:
      if row1[position1] == row2[position2]:
        matrix_new.insert(item, row1 + row2)
        break # Si encuentra detiene

  # Verifica que no haya rezagos de datos
  matrix_new = matrix_new[:long_matrix1_aux]

  # Obtiene la longitud máxima de las listas
  long_elements = 0
  for row1 in matrix_new:
    if len(row1) > long_elements:
      long_elements = len(row1)

  # Establece valores None en unión sin coincidencia
  for row1 in matrix_new:
    while len(row1) < long_elements:
      row1.append(None)

  # Elimina cabeceras de unión
  data_join = drop(data(matrix_new, header1 + header2),
                   [header2[position2]])
  return data_join # Retorna matriz unida
```

```python
# ========================================================== #
# Agrupa elementos de una tabla generando n número de tablas
# ========================================================== #
def group_by(matrix, name):

  ''' Función para agrupar por los datos de una columna '''

  unique_values = unique( column( matrix, name ) )

  group = [] # Guarda matrices apiladas
  for value in unique_values:
    group.append( filter_value( matrix, name, value ) )

  return (unique_values, group) # Retorna apilamiento de nombres y matrices
```

3. SEGUIMIENTO DE LA CONSIGNA E INSTRUCCIONES

Para conseguir los reportes solicitados se unió la información proporcionada a través del identificador del producto *id_product* y se hizo el conteo de ventas, devoluciones, búsquedas y reseñas promediadas en base a las ventas. La tabla resultante se almaceno en *products_sales.*

```
products = data(lifestore_products, names = ['id_product', 'name', 'price', 'category', 'stock'])
searches = data(lifestore_searches, names = ['id_search','id_product'])
sales = data(lifestore_sales, names = ['id_sale','id_product','score','date','refund'])

# Cuenta total de ventas por producto
count_sales = value_counts(sales, 'id_product', 'count_sales')

# Cuenta total de devoluciones por producto
count_refund = value_counts( filter_value(sales, 'refund', 1) , 'id_product', 'count_refunds')

# Cuenta total de búsquedas por producto
count_searches = value_counts(searches, 'id_product', 'count_searches')

# Une en una tabla los productos con el recuento de ventas, rembolsos y búsquedas
products_sales = join( products, count_sales, 'id_product' )
products_sales = join( products_sales, count_refund, 'id_product' )
products_sales = join( products_sales, count_searches, 'id_product' )

# Elimina los valores con None
products_sales = fillnone( products_sales, 'count_sales', 0 )
products_sales = fillnone( products_sales, 'count_refunds', 0 )
products_sales = fillnone( products_sales, 'count_searches', 0 )

# Agrega columna de ventas absolutas ( Las ventas que se efectuaron sin rembolso )
products_sales = add_column( products_sales,
                      'absolute_sales',
                      subt_vectors(column(products_sales,'count_sales'), column(products_sales,'count_refunds')))

# Agrega columna del total de ganancias de las ventas absolutas
products_sales = add_column(products_sales,
                      'total_profit',
                      multi_vectors(column(products_sales,'price'), column(products_sales,'absolute_sales')))


## ================ Crea la media del score de cada producto vendido ===========
# Agrupa las ventas del mismo producto
products, group_products_sales = group_by(sales, 'id_product')

matrix_scores = [] # Almacena nueva matriz con promedios de los scores por producto

# Desempaqueta productos y matrices por productos
for product, table in zip(products, group_products_sales):

  summary_scores = [] # Almacena el resumen de los scores por producto
  scores = column( table , 'score' )  # Obtiene los scores

  # Promedia score de total de ventas del mismo producto
  sum_scores = sum(scores)
  count_sales = len(scores)
  mean = round(sum_scores/count_sales,2)

   # Apila información para crear una nueva tabla
   summary_scores.append(product)
   summary_scores.append(mean)

   # Apila filas de la nueva tabla
   matrix_scores.append(summary_scores)

# Crea tabla de productos promediando los scores
products_mean_scores = data(matrix_scores, names = ['id_product','mean_score'])

# Crea tabla con los productos y las ventas
products_sales = join(products_sales, products_mean_scores, on= 'id_product')
products_sales = fillnone(products_sales, 'mean_score', 0) # Rellena espacios None

# Muestra tabla creada
show(products_sales,10)

# Obtiene el total de ganancias de todas las ventas
total = sum(column( products_sales, 'total_profit')) # Suma todos los valores de la columna total_profit
print(f'Total de ganancias: ${total}')
```

CONSIGNA 1

Productos más vendidos y productos rezagados a partir del análisis de las categorías con menores ventas y categorías con menores búsquedas.

- Enlistar los 50 productos con mayores ventas y los 100 productos con mayores búsquedas.

```
# 50 productos más vendidos
most_selled_products = head( sort_by( products_sales, 'absolute_sales' ), 50 ) # Ordena tabla con 50 filas (productos)
most_selled_products = drop( most_selled_products, ['mean_score','count_searches','price','total_profit'] )

# 100 productos más buscados
most_wanted_products = head( sort_by( products_sales, 'count_searches' ), 100 ) # Ordena tabla con 100 filas (productos)
most_wanted_products = drop( most_wanted_products, ['stock','mean_score','count_sales', 'count_refunds','total_profit','price'] )
```

- Enlistar 50 productos con menores ventas y 100 productos con menores búsquedas.

```
# 50 productos menos vendidos
lower_sales_products = head( sort_by( products_sales, 'absolute_sales', ascending = True), 50 ) # Ordena tabla con 50 filas (productos)
lower_sales_products = drop(lower_sales_products, ['mean_score','count_searches','price','total_profit'])

# 100 productos menos buscados
less_searches_products = head( sort_by( products_sales, 'count_searches', ascending = True), 100 ) # Ordena tabla con 100 filas (productos)
less_searches_products = drop( less_searches_products, ['stock','mean_score','count_sales', 'count_refunds','total_profit','price'])
#show( less_searches_products ) # Muestra tabla
```

- *Listado por categoría de todos los productos ordenados por mayores ventas.*

```
# Productos más vendidos por categorías
most_selled_products_sales = drop( sort_by( products_sales, 'absolute_sales' ), names=['count_searches', 'mean_score'])
categorys_most_selled, group_categorys_selled = group_by( most_selled_products_sales, name = 'category' )

# Obtiene tabla del total de ventas por categoría y las ordena
table_most_selled_categorys = []
for category, table in zip(categorys_most_selled, group_categorys_selled):
  absolute_sales = sum(column(table, 'absolute_sales'))
  total_profit = sum(column(table, 'total_profit'))
  table_most_selled_categorys.append([category, absolute_sales, total_profit])

# Crea y ordena la tabla por el número de ventas
most_selled_categorys = data(table_most_selled_categorys, names=['category','absolute_sales','total_profit'])
most_selled_categorys = sort_by(most_selled_categorys,'absolute_sales')

# Obtiene orden de impresión de las tablas
categorys_sorted_sales = column(most_selled_categorys, 'category')
```

- *Listado por categoría de todos los productos ordenados por mayores búsquedas.*

```
# Productos más buscados por categoría
most_wanted_products_sales = drop( sort_by( products_sales, 'count_searches' ), names = ['count_sales','count_refunds','total_profit','mean_score'])
categorys_most_searched, group_categorys_searched = group_by( most_wanted_products_sales, name = 'category' )

# Obtiene tabla del total de búsquedas por categoría y las ordena
table_most_searched_categorys = []
for category, table in zip(categorys_most_searched, group_categorys_searched):
  count_searches = sum( column(table, name='count_searches') )
  absolute_sales = sum( column(table, name='absolute_sales') )
  table_most_searched_categorys.append([category,count_searches, absolute_sales])

# Crea y ordena la tabla por el número total de búsquedas
most_searched_categorys = data(table_most_searched_categorys, names=['category','count_searches','absolute_sales'])
most_searched_categorys = sort_by(most_searched_categorys, name='count_searches')

# Obtiene orden de impresión de las tablas
categorys_sorted_searches = column(most_searched_categorys, 'category')
```

CONSIGNA 2

Productos por reseña en el servicio a partir del análisis de categorías con mayores ventas y categorías con mayores búsquedas.

- Enlistar 20 productos que cuentan con las mejores reseñas y 20 productos que cuentan con las peores reseñas considerando las devoluciones.

```
# Filtra productos que tienen puntuación en sus ventas
products_with_score = filter_value( products_sales, 'mean_score', 0, not_value = True )

# Reseñas altas
higher_scores = head(drop(sort_by(products_with_score, 'mean_score'),
                          names=['absolute_sales','count_searches','count_refunds', 'total_profit']), 20)
# Reseñas bajas
lower_scores = head(drop(sort_by(products_with_score, 'mean_score', ascending = True),
                         names=['absolute_sales','count_searches','count_refunds','total_profit']), 20)
```

- *Listado por categoría de todos los productos ordenados por mayores ventas.*

```
# Productos con mayores ventas
all_higher_scores_sales = drop(sort_by( products_with_score, 'absolute_sales' ), names=['count_searches'])
categorys_higher_scores_sales, group_categorys_higher_scores_sales = group_by(all_higher_scores_sales, name='category')
```

- *Listado por categoría de todos los productos ordenados por mayores búsquedas.*

```
# Productos con mayores búsquedas
all_higher_scores_searches = drop(sort_by( products_with_score, 'count_searches'),names=['count_sales','total_profit'])
categorys_higher_scores_searches, group_categorys_higher_scores_searches = group_by(all_higher_scores_searches, name='category')
```

CONSIGNA 3

Estrategia de los productos a retirar del mercado, así estrategia para reducir el inventario acumulado considerando los ingresos y ventas mensuales.

- Ventas promedio mensuales.

```
# Vuelve a cargar datos para evitar errores
sales = data(lifestore_sales, names = ['id_sale','id_product','score','date','refund'])
products = data(lifestore_products, names = ['id_product', 'name', 'price', 'category', 'stock'])

# Separa la columna 'date'
date_split = split_column( column(sales,  'date'), '/', names = ['day', 'month','year'] )

# Une tabla de ventas con los días, meses y años correspondientes
expanded_sales = add_column(sales, 'day', column( date_split, 'day') )
expanded_sales = add_column(expanded_sales, 'month', column(date_split,'month'))
expanded_sales = add_column(expanded_sales, 'year', column(date_split,'year'))

# Ordena tabla de ventas extendida por meses
expanded_sales = sort_by(expanded_sales, name = 'month', ascending = True)
```

```
# Secciona tabla de ventas por años
years, group_sales_years = group_by(expanded_sales, name = 'year')

# Ordena años de mayor a menor
years_sorted = sorted(years, reverse = True)

cointainer_summary_month = []
summary_years_months = []
data_summary_month = []

for year_sorted in years_sorted:
  for year, table_year in zip(years, group_sales_years):
    if year == year_sorted:

      # Secciona por meses
      months, group_sales_months = group_by( table_year, name = 'month' )

      for month, table_month in zip(months, group_sales_months):
        #print(f'{year}/{month}')

        # Obtiene conteo de productos vendidos por mes
        count_values_summary_month = value_counts(table_month, name= 'id_product', name_output='count_sales')
        summary_month = merge(count_values_summary_month, products, 'id_product', 'id_product')

        # Filtra productos con devoluciones y los cuenta por mes
        count_refunds_month = value_counts( filter_by( table_month, name='refund', by = [1]),
                                           name = 'id_product', name_output='count_refunds')
         # Une conteo de productos con devoluciones con tabla de mes
         summary_month = join(summary_month, count_refunds_month, on = 'id_product')

         # Agrega count_refunds para las tablas que no cuentan con devoluciones
         header, _= summary_month
         if not 'count_refunds' in header:
          summary_month = add_column(summary_month, name = 'count_refunds', vector=[])

         # Rellena con 0's
         summary_month = fillnone(summary_month, 'count_refunds', 0)

         count_total_sales = sum( column(summary_month, 'count_sales'))
         count_total_refunds = sum( column(summary_month, 'count_refunds'))

         # Obtiene total de ventas efectivas por mes
         absolute_total_sales = subt_vectors(column(summary_month,'count_sales'), column(summary_month,'count_refunds'))
         summary_month = add_column(summary_month, name ='absolute_total_sales', vector = absolute_total_sales)
         total_sales = sum( absolute_total_sales )

         total_profit = multi_vectors(column(summary_month, 'absolute_total_sales'), column(summary_month, 'price'))
         summary_month = add_column(summary_month, name ='total_profit', vector = total_profit)
         count_total_profit = sum(total_profit)

         #show(summary_month)

         cointainer_summary_month.append(summary_month) # Apila resumen por mes

         data_summary_month.append([year,month])
         summary_years_months.append([year, month, count_total_sales, count_total_refunds, total_sales, count_total_profit])

summary_years = data(summary_years_months,
                     names = ['year','month','count_total_sales','count_total_refunds','total_sales','total_profit'])
#show(summary_years)
```

- Ventas anuales.
- Total de ingresos.
- Meses con más ventas al año.

```
#Ordena años
years = sorted( unique( column( summary_years, 'year') ), reverse = True )


container_years = [] # Contiene tablas por años
total_years = []     # Almacena total por año

all_sales = 0 # Almacena total

for year in years:
  table_filter_year = filter_value( summary_years, 'year', year )

  container_years.append(table_filter_year)
  total = sum( column( table_filter_year, 'total_profit' ) )
  total_years.append(total)

  all_sales += total
```

SISTEMA

MÉTODOS PARA INICIAR SESIÓN Y REGISTRARSE

```
users = ['admi']      # Guarda los nombres de usuario
passwords = ['super'] # Guarda las contraseñas

def login():

  ''' Función para acceder '''

  user = input('Introduzca su nombre de usuario: ')

  if user in users: # Busca nombre de usuario
    position = users.index(user) # Obtiene posición para comprobar contraseña
    password = input('Introduzca su contraseña: ')

    if password == passwords[position]: # Verifica la contraseña
      print('¡Bienvenid@!')
      return True

  print('No se encontro coincidencia') # Imprime mensaje
  return False
```

```
def singup():

  ''' Función para registrar '''

  while True:

    # Pide ingresar nuevo nombre y contraseña
    user = input('Introduzca su nuevo nombre de usuario: ')
    if not user in users:
      users.append(user)

      password = input('Introduzca su nueva contraseña: ')
      passwords.append(password)

      print('Se ha registrado con éxito... ')

      break
    else:
      print('El usuario ya existe, prueba con otro')
```

MÉTODOS PARA MOSTRAR MENÚ DE OPCIONES PARA CADA CONSIGNA

```
def menu_main():
  print(f'''
                ┌──────────────────────┐
                │         Menú         │
                │──────────────────────│
                │ [1] - Registrarse    │
                │ [2] - Acceder        │
                │ [otro] - Salir       │
                └──────────────────────┘

  ''')
  return input('¿Qué desea hacer?: ')
```

```
def menu_reports():
  print(f'''
        ┌──────────────────────────────────────────┐
        │                Consignas                 │
        │──────────────────────────────────────────│
        │ [1] Productos más vendidos y productos   │
        │     rezagados por categoría              │
        │ [2] Productos con mejores y peores       │
        │     reseñas considerando las devoluciones│
        │     y categorías                         │
        │ [3] Resumen de ventas por periodos       │
        │ [otro] Salir                             │
        └──────────────────────────────────────────┘
  ''')
  return input('¿Qué desea hacer?: ')
```

```
def reports_1():
  print(f'''
        ┌─────────────────────────────────────────────┐
        | Productos más vendidos y productos rezagados |
        |               por categoría                  |
        |---------------------------------------------|
        | [1] Ver listado con 50 productos con         |
        |     con mayores ventas y 100 con mayores     |
        |     búsquedas                                |
        | [2] Ver listado con 50 productos con         |
        |     con menores ventas y 100 con menores     |
        |     búsquedas                                |
        | [3] Ver listado por categoría de todos los   |
        |     productos ordenados por mayores ventas   |
        | [4] Ver listado por categoría de todos los   |
        |     productos ordenados por mayores búsquedas |
        | [otro] Salir                                 |
        └─────────────────────────────────────────────┘
  ''')
  return input('¿Qué desea hacer?: ')
```

```
def reports_2():
  print(f'''
        ┌─────────────────────────────────────────────┐
        |  Productos con mejores y peores reseñas      |
        |    considerando devoluciones y categorías    |
        |---------------------------------------------|
        | [1] Ver listado con 20 productos con las     |
        |     mejores reseñas y peores reseñas         |
        | [2] Ver listado por categoría de todos los   |
        |     productos ordenados por mayores ventas   |
        | [3] Ver listado por categoría de todos los   |
        |     productos ordenados por mayores búsquedas |
        | [otro] Salir                                 |
        └─────────────────────────────────────────────┘
  ''')
  return input('¿Qué desea hacer?: ')
```

```
def reports_3():
  print(f'''
        ┌─────────────────────────────────────────────┐
        |       Resumen de ventas por periodos         |
        |---------------------------------------------|
        | [1] Ventas promedio mensuales                |
        | [2] Ventas anuales                           |
        | [3] Total de ingresos                        |
        | [4] Meses con más ventas al año              |
        | [otro] Salir                                 |
        └─────────────────────────────────────────────┘
  ''')
  return input('¿Qué desea hacer?: ')
```

CONTROL DE FLUJO DEL SISTEMA

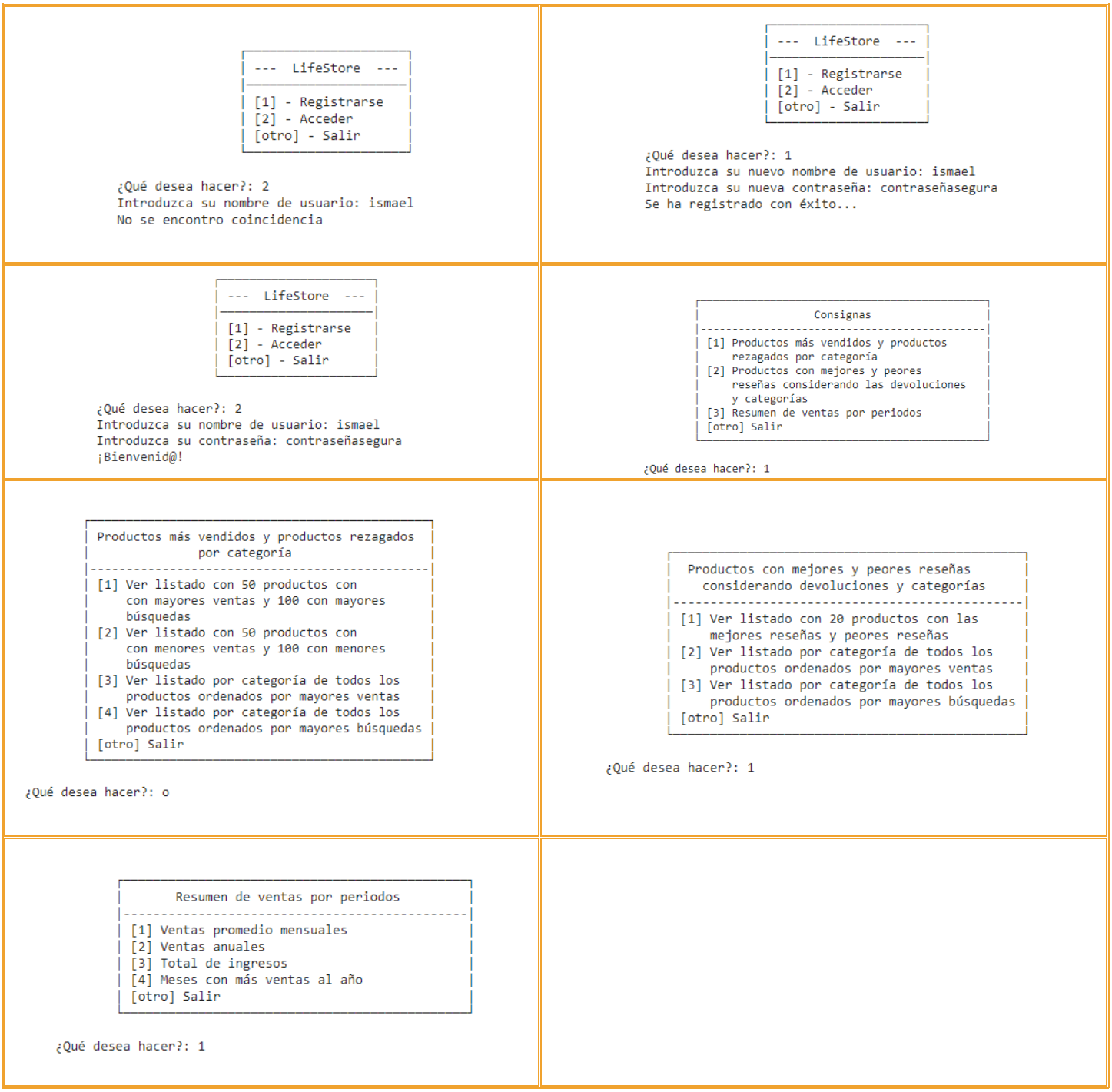
```
while True: # Ciclo menú principal

  choose_menu_main = menu_main()  # Muestra menú principal

  if choose_menu_main == '1':
    singup() # Registro de usuarios

  elif choose_menu_main == '2':
    if login(): # Acceso de usuarios

      while True: # Ciclo menú consignas
        choose_menu_reports = menu_reports() # Muestra menú con consignas
        if choose_menu_reports == '1':

          while True: # Ciclo menú consigna 1
            choose_reports_1 = reports_1() # Muestra menú para la consigna 1

            if choose_reports_1 == '1':
              print('┌──────────────────────────────────┐')
              print('|| 50 Productos con mayores ventas ||')
              print('└──────────────────────────────────┘')
              show(most_selled_products)
              print('┌──────────────────────────────────────┐')
              print('|| 100 Productos con mayores búsquedas ||')
              print('└──────────────────────────────────────┘')
              show(most_wanted_products)

            elif choose_reports_1 == '2':
              print('┌──────────────────────────────────┐')
              print('|| 50 Productos con menores ventas ||')
              print('└──────────────────────────────────┘')
              show(lower_sales_products)
              print('┌──────────────────────────────────┐')
```

```
        print('|| 100 Productos con menores búsquedas ||')
        print('└──────────────────────────────────────┘')
        show(less_searches_products)

      elif choose_reports_1 == '3':
        print('┌──────────────────────────────────────────────────────────────────────┐')
        print('|| Listado por categoría de todos los productos ordenados por mayores ventas ||')
        print('└──────────────────────────────────────────────────────────────────────┘')
        show( most_selled_categorys )

        for category_sorted in categorys_sorted_sales:
          for category, table in zip(categorys_most_selled, group_categorys_selled):
            if category_sorted == category:
              print(f'>> Categoría: {category} <<')
              show(table)

      elif choose_reports_1 == '4':
        print('┌──────────────────────────────────────────────────────────────────────┐')
        print('|| Listado por categoría de todos los productos ordenados por mayores búsquedas ||')
        print('└──────────────────────────────────────────────────────────────────────┘')
        show( most_searched_categorys )

        for category_sorted in categorys_sorted_searches:
          for category, table in zip(categorys_most_searched, group_categorys_searched):
            if category_sorted == category:
              print(f'>> Categoría: {category} <<')
              show(table)

      else:
        break

    elif choose_menu_reports == '2':

      while True: # Ciclo menú consigna 2
        choose_reports_2 = reports_2() # Muestra menú para la consigna 2

        if choose_reports_2 == '1':
          print('┌──────────────────────────────────────────────────┐')
          print('|| Ver listado con 20 productos con las mejores reseñas ||')
          print('└──────────────────────────────────────────────────┘')
          show(higher_scores)
          print('┌──────────────────────────────────────────────────┐')
          print('|| Ver listado con 20 productos con las peores reseñas ||')
          print('└──────────────────────────────────────────────────┘')
          show(lower_scores)

        elif choose_reports_2 == '2':
          print('┌──────────────────────────────────────────────────────────────────────┐')
          print('|| Listado por categoría de todos los productos ordenados por mayores ventas ||')
          print('└──────────────────────────────────────────────────────────────────────┘')
          show( most_selled_categorys )

          for category_sorted in categorys_sorted_sales:
            for category, table in zip(categorys_higher_scores_sales, group_categorys_higher_scores_sales):
              if category_sorted == category:
                print(f'>> Categoría: {category} <<')
                show(sort_by(table, 'mean_score'))

        elif choose_reports_2 == '3':
          print('┌──────────────────────────────────────────────────────────────────────┐')
          print('|| Listado por categoría de todos los productos ordenados por mayores búsquedas ||')
          print('└──────────────────────────────────────────────────────────────────────┘')
          show( most_searched_categorys )

          for category_sorted in categorys_sorted_searches:
            for category, table in zip(categorys_higher_scores_searches, group_categorys_higher_scores_searches):
              if category_sorted == category:
                print(f'>> Categoría: {category} <<')
                show(sort_by(table, 'mean_score'))

        else:
          break

    elif choose_menu_reports == '3':
```

```
        while True: # Ciclo menú consigna 3
          choose_reports_3 = reports_3() # Muestra menú para la consigna 3

          if choose_reports_3 == '1':
            print('┌──────────────────────────────┐')
            print('|| Ventas promedio mensuales ||')
            print('└──────────────────────────────┘')
            show( summary_years )
            for summary_month, table_month in zip(data_summary_month, cointainer_summary_month):
              print( summary_month )
              show( table_month )

          elif choose_reports_3 == '2':
            print('┌──────────────────┐')
            print('|| Ventas anuales ||')
            print('└──────────────────┘')
            for year, table_year in zip(years_sorted, container_years):
              print(f'>> Año: {year} <<')
              show(table_year)
              print(f"Total de ventas: {sum(column(table_year, 'count_total_sales'))}")
              print(f"Total de devoluciones: {sum(column(table_year, 'count_total_refunds'))}")
              print(f"Total de ganancias por año: ${sum(column(table_year, 'total_profit'))}")

          elif choose_reports_3 == '3':
            print('┌──────────────────┐')
            print('|| Total de ingresos ||')
            print('└──────────────────┘')
            print(f'El total de ingresos registrados fueron: ${all_sales}')

          elif choose_reports_3 == '4':
            print('┌──────────────────────────────┐')
            print('|| Meses con más ventas al año ||')
            print('└──────────────────────────────┘')
            years, category_years_months = group_by( sort_by(summary_years, 'total_sales'), 'year' )
            for year, table in zip(years, category_years_months):
              print(f'>> Año: {year} <<')
              show( table )
          else:
            break
        else:
          break
else:
  print('¡Vuelva pronto!')
  break
```

SOLUCIÓN AL PROBLEMA

En esta solución se muestra inicialmente las capturas de la corrida del sistema (se recomienda se corra directamente el código para ver todos los datos de salida) y posteriormente se realiza el análisis solicitado en la consigna 3.



|| 50 Productos con mayores ventas ||

item	id_product	name	category	stock	count_sale	count_refu	absolute_s
1-	54	SSD Kingst	discos dur	300	50	1	49
2-	3	Procesador	procesador	987	42	0	42
3-	5	Procesador	procesador	130	20	0	20
4-	42	Tarjeta Ma	tarjetas m	0	18	0	18
5-	57	SSD Adata	discos dur	15	15	0	15
6-	4	Procesador	procesador	295	13	0	13
7-	29	Tarjeta Ma	tarjetas m	10	14	1	13
8-	2	Procesador	procesador	182	13	1	12
9-	47	SSD XPG SX	discos dur	8	11	0	11
10-	12	Tarjeta de	tarjetas d	0	9	0	9
11-	48	SSD Kingst	discos dur	50	9	0	9
12-	7	Procesador	procesador	114	7	0	7
13-	44	Tarjeta Ma	tarjetas m	0	6	0	6
14-	18	Tarjeta de	tarjetas d	5	5	0	5

|| 100 Productos con mayores búsquedas ||

item	id_product	name	category	count_sear	absolute_s
1-	54	SSD Kingst	discos dur	263	49
2-	57	SSD Adata	discos dur	107	15
3-	29	Tarjeta Ma	tarjetas m	60	13
4-	3	Procesador	procesador	55	42
5-	4	Procesador	procesador	41	13
6-	85	Logitech A	audifonos	35	2
7-	67	TV Monitor	pantallas	32	1
8-	7	Procesador	procesador	31	7
9-	5	Procesador	procesador	30	20
10-	47	SSD XPG SX	discos dur	30	11
11-	48	SSD Kingst	discos dur	27	9
12-	44	Tarjeta Ma	tarjetas m	25	6
13-	2	Procesador	procesador	24	12
14-	42	Tarjeta Ma	tarjetas m	23	18
15-	8	Procesador	procesador	20	4
16-	12	Tarjeta de	tarjetas d	15	9
17-	21	Tarjeta de	tarjetas d	15	2
18-	66	TCL Smart	pantallas	15	1

|| 50 Productos con menores ventas ||

item	id_product	name	category	stock	count_sale	count_refu	absolute_s
1-	9	Procesador	procesador	35	0	0	0
2-	14	Tarjeta de	tarjetas d	36	0	0	0
3-	15	Tarjeta de	tarjetas d	15	0	0	0
4-	16	Tarjeta de	tarjetas d	10	0	0	0
5-	17	Tarjeta de	tarjetas d	1	1	1	0
6-	19	Tarjeta de	tarjetas d	8	0	0	0
7-	20	Tarjeta de	tarjetas d	10	0	0	0
8-	23	Tarjeta de	tarjetas d	10	0	0	0
9-	24	Tarjeta de	tarjetas d	2	0	0	0
10-	26	Tarjeta de	tarjetas d	180	0	0	0
11-	27	Tarjeta de	tarjetas d	43	0	0	0
12-	30	Tarjeta Ma	tarjetas m	50	0	0	0
13-	32	Tarjeta Ma	tarjetas m	10	0	0	0
14-	34	Tarjeta Ma	tarjetas m	2	0	0	0
15-	35	Tarjeta Ma	tarjetas m	30	0	0	0
16-	36	Tarjeta Ma	tarjetas m	10	0	0	0
17-	37	Tarjeta Ma	tarjetas m	60	0	0	0

|| 100 Productos con menores búsquedas ||

item	id_product	name	category	count_sear	absolute_s
1-	14	Tarjeta de	tarjetas d	0	0
2-	16	Tarjeta de	tarjetas d	0	0
3-	19	Tarjeta de	tarjetas d	0	0
4-	20	Tarjeta de	tarjetas d	0	0
5-	23	Tarjeta de	tarjetas d	0	0
6-	24	Tarjeta de	tarjetas d	0	0
7-	30	Tarjeta Ma	tarjetas m	0	0
8-	32	Tarjeta Ma	tarjetas m	0	0
9-	33	Tarjeta Ma	tarjetas m	0	2
10-	34	Tarjeta Ma	tarjetas m	0	0
11-	36	Tarjeta Ma	tarjetas m	0	0
12-	37	Tarjeta Ma	tarjetas m	0	0
13-	38	Tarjeta Ma	tarjetas m	0	0
14-	41	Tarjeta Ma	tarjetas m	0	0
15-	43	Tarjeta Ma	tarjetas m	0	0
16-	53	SSD Addlin	discos dur	0	0
17-	55	SSD para S	discos dur	0	0
18-	58	SSD para S	discos dur	0	0

|| Listado por categoría de todos los productos ordenados por mayores ventas ||

item	category	absolute_s	total_prof
1-	procesador	103	367517
2-	discos dur	93	93237
3-	tarjetas m	43	113727
4-	tarjetas d	25	132025
5-	audifonos	5	9135
6-	bocinas	2	8478
7-	pantallas	2	11278
8-	memorias u	1	2519

>> Categoría: procesadores <<

item	id_product	name	price	category	stock	count_sale	count_refu	absolute_s	total_prof
1-	3	Procesador	3089	procesador	987	42	0	42	129738
2-	5	Procesador	1779	procesador	130	20	0	20	35580
3-	4	Procesador	2209	procesador	295	13	0	13	28717
4-	2	Procesador	4209	procesador	182	13	1	12	50508
5-	7	Procesador	8559	procesador	114	7	0	7	59913
6-	8	Procesador	5399	procesador	8	4	0	4	21596
7-	6	Procesador	11809	procesador	54	3	0	3	35427
8-	1	Procesador	3019	procesador	16	2	0	2	6038
9-	9	Procesador	2549	procesador	35	0	0	0	0

|| Listado por categoría de todos los productos ordenados por mayores búsquedas ||

item	category	count_sear	absolute_s
1-	discos dur	463	93
2-	procesador	222	103
3-	tarjetas m	137	43
4-	tarjetas d	82	25
5-	audifonos	64	5
6-	pantallas	56	2
7-	bocinas	9	2
8-	memorias u	0	1

>> Categoría: discos duros <<

item	id_product	name	price	category	stock	count_sear	absolute_s
1-	54	SSD Kingst	259	discos dur	300	263	49
2-	57	SSD Adata	889	discos dur	15	107	15
3-	47	SSD XPG SX	1209	discos dur	8	30	11
4-	48	SSD Kingst	2559	discos dur	50	27	9
5-	51	SSD Kingst	2399	discos dur	0	11	3
6-	49	Kit SSD Ki	3139	discos dur	3	10	3
7-	50	SSD Crucia	2949	discos dur	4	7	1

|| Ver listado con 20 productos con las mejores reseñas ||

item	id_product	name	price	category	stock	count_sale	mean_score
1-	1	Procesador	3019	procesador	16	2	5.0
2-	6	Procesador	11809	procesador	54	3	5.0
3-	7	Procesador	8559	procesador	114	7	5.0
4-	8	Procesador	5399	procesador	8	4	5.0
5-	11	Tarjeta de	7399	tarjetas d	2	3	5.0
6-	21	Tarjeta de	5159	tarjetas d	0	2	5.0
7-	22	Tarjeta de	3429	tarjetas d	0	1	5.0
8-	25	Tarjeta de	5529	tarjetas d	10	2	5.0

|| Ver listado con 20 productos con las peores reseñas ||

item	id_product	name	price	category	stock	count_sale	mean_score
1-	17	Tarjeta de	4199	tarjetas d	1	1	1.0
2-	45	Tarjeta Ma	2869	tarjetas m	25	1	1.0
3-	31	Tarjeta Ma	2229	tarjetas m	120	6	1.83
4-	46	Tarjeta Ma	1539	tarjetas m	49	1	2.0
5-	89	Cougar Aud	859	audifonos	4	1	3.0
6-	10	MSI GeForc	889	tarjetas d	13	1	4.0
7-	13	Tarjeta de	3989	tarjetas d	1	1	4.0
8-	94	HyperX Aud	2869	audifonos	12	1	4.0
9-	29	Tarjeta Ma	2499	tarjetas m	10	14	4.14

|| Listado por categoría de todos los productos ordenados por mayores ventas ||

item	category	absolute_s	total_prof
1-	procesador	103	367517
2-	discos dur	93	93237
3-	tarjetas m	43	113727
4-	tarjetas d	25	132025
5-	audifonos	5	9135
6-	bocinas	2	8478
7-	pantallas	2	11278
8-	memorias u	1	2519

>> Categoría: procesadores <<

item	id_product	name	price	category	stock	count_sale	count_refu	absolute_s	total_prof	mean_score
1-	7	Procesador	8559	procesador	114	7	0	7	59913	5.0
2-	8	Procesador	5399	procesador	8	4	0	4	21596	5.0
3-	6	Procesador	11809	procesador	54	3	0	3	35427	5.0
4-	1	Procesador	3019	procesador	16	2	0	2	6038	5.0
5-	3	Procesador	3089	procesador	987	42	0	42	129738	4.81
6-	5	Procesador	1779	procesador	130	20	0	20	35580	4.7
7-	4	Procesador	2209	procesador	295	13	0	13	28717	4.46

|| Listado por categoría de todos los productos ordenados por mayores búsquedas ||

item	category	count_sear	absolute_s
1-	discos dur	463	93
2-	procesador	222	103
3-	tarjetas m	137	43
4-	tarjetas d	82	25
5-	audifonos	64	5
6-	pantallas	56	2
7-	bocinas	9	2
8-	memorias u	0	1

>> Categoría: discos duros <<

item	id_product	name	price	category	stock	count_refu	count_sear	absolute_s	mean_score
1-	49	Kit SSD Ki	3139	discos dur	3	0	10	3	5.0
2-	50	SSD Crucia	2949	discos dur	4	0	7	1	5.0
3-	52	SSD Wester	5659	discos dur	13	0	5	2	5.0
4-	57	SSD Adata	889	discos dur	15	0	107	15	4.87
5-	54	SSD Kingst	259	discos dur	300	1	263	49	4.72
6-	48	SSD Kingst	2559	discos dur	50	0	27	9	4.67
7-	51	SSD Kingst	2399	discos dur	0	0	11	3	4.67
8-	47	SSD XPG SX	1209	discos dur	8	0	30	11	4.55

>> Año: 2020 <<

item	year	month	count_total_sales	count_total_refunds	total_sales	total_profit
1-	2020	01	53	1	52	117738
2-	2020	02	41	1	40	107270
3-	2020	03	51	2	49	162931
4-	2020	04	75	1	74	191066
5-	2020	05	35	2	33	91677
6-	2020	06	11	0	11	36949
7-	2020	07	11	0	11	26949
8-	2020	08	3	0	3	3077
9-	2020	09	1	1	0	0

Total de ventas: 281
Total de devoluciones: 8
Total de ganancias por año: $737657

>> Año: 2019 <<

item	year	month	count_total_sales	count_total_refunds	total_sales	total_profit
1-	2019	11	1	1	0	0

Total de ventas: 1
Total de devoluciones: 1
Total de ganancias por año: $0

>> Año: 2002 <<

item	year	month	count_total_sales	count_total_refunds	total_sales	total_profit
1-	2002	05	1	0	1	259

Total de ventas: 1
Total de devoluciones: 0
Total de ganancias por año: $259

ANÁLISIS

|| Listado por categoría de todos los productos ordenados por mayores búsquedas ||

item	category	count_searches	absolute_sales
1-	discos duros	463	93
2-	procesadores	222	103
3-	tarjetas mad	137	43
4-	tarjetas de	82	25
5-	audifonos	64	5
6-	pantallas	56	2
7-	bocinas	9	2
8-	memorias usb	0	1

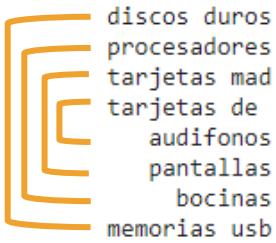
Categorías más buscadas

Categorías menos buscadas

Las categorías con mayores ventas también corresponden a las de categorías más buscadas, lo que permite asumir un coeficiente de correlación significante entre el número de búsquedas y el número de ventas.

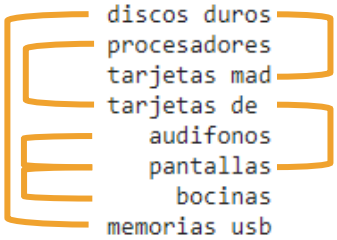
Las categorías menos compradas a lo largo de los años han sido: audífonos, pantallas, bocinas y memorias. Los productos de estas categorías podrían ofertarse de manera conjunta o emergente durante la búsqueda de las categorías con más ventas como lo son: discos duros, procesadores, tarjetas madre y tarjetas de video.

Un ejemplo de la agrupación de aparición o mención de búsquedas, serían los conjuntos:



Memorias usb → Discos duros
Bocinas → Procesadores
Pantallas → Tarjetas madre
Audífonos → Tarjetas de video

O bien a través de la semejanza de la categoría:



Memorias usb → Discos duros
Tarjetas de video → Procesadores
Tarjetas madre → Discos duros
Pantallas → Tarjetas de video
Bocinas → Audífonos
Pantallas → Audífonos

Una solución para definir las que categorías y específicamente los productos que podrían ofertarse en conjunto con la búsqueda de otros productos es conociendo el historial de ventas por mes a lo largo de los años y las categorías que se ven implicadas en dicho mes.

|| Meses con más ventas al año ||

>> Año: 2020 <<

item	year	month	count_total_sales	count_total_refunds	total_sales
1-	2020	04	75	1	74
2-	2020	01	53	1	52
3-	2020	03	51	2	49
4-	2020	02	41	1	40
5-	2020	05	35	2	33
6-	2020	06	11	0	11
7-	2020	07	11	0	11
8-	2020	08	3	0	3
9-	2020	09	1	1	0

>> Año: 2002 <<

item	year	month	count_total_sales	count_total_refunds	total_sales
1-	2002	05	1	0	1

>> Año: 2019 <<

item	year	month	count_total_sales	count_total_refunds	total_sales
1-	2019	11	1	1	0

|| Ventas promedio mensuales ||

item	year	month	count_total_sales	sorted_most_sales_category
1-	2020	01	53	['procesadores', 'discos duros', 'tarjetas madre', 'audifonos']
2-	2020	02	41	['discos duros', 'procesadores', 'tarjetas madre', 'tarjetas de video', 'bocinas']
3-	2020	03	51	['procesadores', 'discos duros', 'tarjetas madre', 'tarjetas de video']
4-	2020	04	75	['procesadores', 'discos duros', 'tarjetas madre', 'tarjetas de video', 'audifonos', 'pantallas']
5-	2020	05	35	['procesadores', 'discos duros', 'tarjetas madre', 'tarjetas de video', 'audifonos', 'pantallas']
6-	2020	06	11	['procesadores', 'tarjetas de video', 'discos duros', 'memorias usb']
7-	2020	07	11	['procesadores', 'discos duros', 'tarjetas madre']
8-	2020	08	3	['discos duros']
9-	2020	09	1	['tarjetas de video']
10-	2019	11	1	['procesadores']
11-	2002	05	1	['discos duros']

En base a las tablas anteriores se pueden ofertar las categorías con productos que no son buscados y comprados en ciertos meses, por ejemplo, en el mes 04 y 05 se puede ofertar “memorias usb “ al comprar algún producto de “discos duros”. También se puede promocionar las categorías con menos búsquedas en los meses que tienen pocas ventas por categorías, como es el caso del mes 08 y 09.

Otra forma de reducir el stock es considerar dejar de vender los productos cuyas evaluaciones y ventas sean bajas una vez que se haya terminado o sacado el stock, así mismo, cuyo número de devoluciones sea semejante o igual al número de ventas registradas; esto debido a que no representan un beneficio real a la tienda.

Productos cuyo Stock y precio son altos

Productos que han sido devueltos con baja reseña

item	id_prod	name	price	categor	stock	count_s	count_r	count_s	absolut	total_p	mean_sc
1-	9	Procesa	2549	procesa	35	0	0	1	0	0	0
2-	14	Tarjeta	1439	tarjeta	36	0	0	0	0	0	0
3-	15	Tarjeta	8439	tarjeta	15	0	0	4	0	0	0
4-	16	Tarjeta	9799	tarjeta	10	0	0	0	0	0	0
5-	17	Tarjeta	4199	tarjeta	1	1	1	3	0	0	1.0
6-	19	Tarjeta	4509	tarjeta	8	0	0	0	0	0	0
7-	20	Tarjeta	11509	tarjeta	10	0	0	0	0	0	0
8-	23	Tarjeta	909	tarjeta	10	0	0	0	0	0	0
9-	24	Tarjeta	30449	tarjeta	2	0	0	0	0	0	0
10-	26	Tarjeta	1249	tarjeta	180	0	0	5	0	0	0
11-	27	Tarjeta	2109	tarjeta	43	0	0	1	0	0	0
12-	30	Tarjeta	4029	tarjeta	50	0	0	0	0	0	0
13-	32	Tarjeta	4309	tarjeta	10	0	0	0	0	0	0
14-	34	Tarjeta	5289	tarjeta	2	0	0	0	0	0	0
15-	35	Tarjeta	3419	tarjeta	30	0	0	1	0	0	0
16-	36	Tarjeta	4159	tarjeta	10	0	0	0	0	0	0

37	Tarjeta	4289	tarjeta	60	0	0	0	0	0	0
38	Tarjeta	1369	tarjeta	15	0	0	0	0	0	0
39	ASUS T.	2169	tarjeta	98	0	0	3	0	0	0
41	Tarjeta	3329	tarjeta	286	0	0	0	0	0	0
43	Tarjeta	6369	tarjeta	5	0	0	0	0	0	0
45	Tarjeta	2869	tarjeta	25	1	1	1	0	0	1.0
46	Tarjeta	1539	tarjeta	49	1	1	4	0	0	2.0
53	SSD Add	2039	discos	1	0	0	0	0	0	0
55	SSD par	4399	discos	10	0	0	0	0	0	0
56	SSD par	3269	discos	3	0	0	2	0	0	0
58	SSD par	3679	discos	16	0	0	0	0	0	0
59	SSD Sam	5539	discos	10	0	0	1	0	0	0
61	Kit Mem	5209	memoria	5	0	0	0	0	0	0
62	Makena	2899	pantall	6	0	0	0	0	0	0
63	Seiki T	3369	pantall	146	0	0	4	0	0	0
64	Samsung	12029	pantall	71	0	0	0	0	0	0
65	Samsung	21079	pantall	7	0	0	0	0	0	0
68	Makena	4229	pantall	239	0	0	0	0	0	0
69	Hisense	5359	pantall	94	0	0	0	0	0	0
70	Samsung	7679	pantall	10	0	0	1	0	0	0
71	Samsung	4829	pantall	3	0	0	0	0	0	0
72	Hisense	9759	pantall	11	0	0	0	0	0	0
73	Samsung	10559	pantall	4	0	0	4	0	0	0
75	Lenovo	441	bocinas	11	0	0	0	0	0	0
76	Acteck	589	bocinas	18	0	0	2	0	0	0
77	Verbati	178	bocinas	1	0	0	0	0	0	0
78	Ghia Bo	769	bocinas	2	0	0	0	0	0	0
79	Naceb B	709	bocinas	31	0	0	0	0	0	0
80	Ghia Bo	1359	bocinas	15	0	0	1	0	0	0
81	Ghia Bo	1169	bocinas	20	0	0	0	0	0	0
82	Ghia Bo	549	bocinas	31	0	0	0	0	0	0
83	Ghia Bo	499	bocinas	16	0	0	0	0	0	0
86	ASUS Au	8359	audifon	20	0	0	0	0	0	0
87	Acer Au	1719	audifon	8	0	0	0	0	0	0
88	Audífon	909	audifon	15	0	0	0	0	0	0
90	Energy	539	audifon	1	0	0	0	0	0	0
91	Genius	137	audifon	16	0	0	2	0	0	0
92	Getttec	149	audifon	232	0	0	0	0	0	0
93	Ginga A	160	audifon	139	0	0	1	0	0	0
95	Iogear	999	audifon	2	0	0	3	0	0	0
96	Klip Xt	769	audifon	2	0	0	0	0	0	0

CONCLUSIÓN

La tienda cuenta con un gran stock de productos que no se han vendido, se sugiere promocionar el stock existente mediante la clasificación de categorías con mayores ventas según los meses donde se perciben mayores flujos. Ya que la estrategia de venta depende fuertemente de las búsquedas del producto se debe buscar enfatizar los productos que menos se venden de las categorías que menos se venden, también se considera descartar la cantidad de productos ofertados por categoría para conseguir eliminar los que tienen menores ventas.

La practica propicio la manipulación de datos para poder visualizar y cumplir con los requerimientos solicitados por la gerencia de ventas generando un sistema básico pero útil en la agrupación y visualización de datos.