

**Universidad Nacional del Altiplano**  
**Facultad de Ingeniería Estadística e Informática**  
**Docente:** Fred Torres Cruz  
**Estudiantes :**

- Roy Anthony Quispe Alave
- Blanca Beatriz Flores Aycaya

## Repositorio de GitHub

Puedes acceder al repositorio de GitHub en el siguiente enlace:  
[Repositorio en GitHub](#)

### 1. Introduction

Este informe tiene como objetivo analizar los datos de ventas contenidos en el archivo registro de ventas. El archivo XLS contiene información detallada sobre las compras realizadas por diferentes clientes, incluyendo detalles como el ID del cliente, la zona, el país, el tipo de producto, el canal de venta, la prioridad, la fecha del pedido, el ID del pedido, la fecha de envío, las unidades vendidas, el precio unitario, el coste unitario, el importe total de la venta y el importe total del coste. Este informe proporciona una visión general de las ventas, incluyendo análisis de las tendencias y un código de Python que se puede utilizar para encontrar cualquier compra que se hizo

## 2. DESARROLLO

### 2.1. DESCRIPCION DE LOS DATOS

El archivo registro de ventas contiene un total de 200 filas de datos, cada una representando una compra individual. Este conjunto de datos ofrece una visión detallada de las transacciones comerciales,

incluyendo información sobre los clientes, los productos, los canales de venta y los costes asociados.

## 2.2. INFORMACION DEL CLIENTE

Las primeras columnas del archivo XLS se dedican a proporcionar información detallada sobre el cliente que realizó la compra. Esta información incluye:

- - ID Cliente: Un identificador único para cada cliente. Este ID permite rastrear las compras individuales de cada cliente a lo largo del conjunto de datos. Por ejemplo, C2421 representa un cliente específico.
- - Zona: La región geográfica donde se encuentra el cliente. Esta información puede ser útil para analizar las tendencias de ventas por región. Por ejemplo, .Europa indica que el cliente se encuentra en Europa.
- - País: El país donde se encuentra el cliente. Esta información puede ser útil para analizar las tendencias de ventas por país. Por ejemplo, United Kingdom indica que el cliente se encuentra en el Reino Unido.

## 2.3. INFORMACION DEL PRODUCTO

Las siguientes columnas del archivo XLS se dedican a proporcionar información detallada sobre el producto que se compró. Esta información incluye:

- - Tipo de producto: El tipo de producto que se compró. Esta información puede ser útil para analizar las tendencias de ventas por tipo de producto.

- Por ejemplo, "Snacks" indica que se compró un producto del tipo "Snacks".

## 2.4. INFORMACION DE LA VENTA

Las siguientes columnas del archivo XLS se dedican a proporcionar información detallada sobre la venta en sí. Esta información incluye:

- - Canal de venta: El canal a través del cual se realizó la compra (online u offline). Esta información puede ser útil para analizar las tendencias de ventas por canal.
  - Por ejemplo, "Offline" indica que la compra se realizó offline.
- - Prioridad: La prioridad de la compra, que puede ser crítica, alta, media o baja. Esta información puede ser útil para analizar las prioridades de los clientes.
  - Por ejemplo, "Crítica" indica que la compra tiene una prioridad alta para el cliente.
- - Fecha pedido: La fecha en la que se realizó el pedido. Esta información puede ser útil para analizar las tendencias de ventas por fecha.
  - Por ejemplo, "10/12/20" indica que el pedido se realizó el 10 de diciembre de 2020.
- - ID Pedido: Un identificador único para cada pedido. Este ID permite rastrear cada pedido individual a lo largo del conjunto de datos.
  - Por ejemplo, "242113196" representa un pedido específico.

## 2.5. INFORMACION DEL ENVIO

Las siguientes columnas del archivo XLS se dedican a proporcionar información detallada sobre el envío del pedido. Esta información incluye:

- - Fecha envío: La fecha en la que se envió el pedido. Esta información puede ser útil para analizar las tendencias de envío por fecha. Por ejemplo, "11/30/20" indica que el pedido se envió el 30 de noviembre de 2020.

## 2.6. INFORMACION DE COSTE Y VENTA

Las últimas columnas del archivo XLS se dedican a proporcionar información detallada sobre el coste y la venta del producto. Esta información incluye:

- - Unidades: El número de unidades del producto que se compraron. Por ejemplo, 5530.
- - Precio Unitario: El precio de una unidad del producto. Por ejemplo, 152.58.
- - Coste unitario: El coste de una unidad del producto. Por ejemplo, 97.44.
- - Importe venta total: El importe total de la venta, calculado multiplicando las unidades vendidas por el precio unitario. Por ejemplo, 843.767.40.
- - Importe Coste total: El importe total del coste, calculado multiplicando las unidades vendidas por el coste unitario. Por ejemplo, 538.843.20.

### **3. ANALISIS DE VENTAS**

#### **3.1. PRODUCTOS MAS VENDIDOS**

- - Snacks: Los snacks son el tipo de producto más vendido, con un total de 25 compras registradas. Esto sugiere que los snacks son un producto popular entre los clientes.
- - Cuidado personal: Los productos de cuidado personal son el segundo tipo de producto más vendido, con un total de 23 compras registradas. Esto sugiere que los clientes están interesados en productos que contribuyen a su bienestar personal.
- - Bebidas: Las bebidas son el tercer tipo de producto más vendido, con un total de 21 compras registradas. Esto sugiere que las bebidas son un producto esencial para los clientes.

#### **3.2. CANALES DE VENTA**

- 
- - Online: La mayoría de las compras se realizan online, con un total de 120 compras registradas. Esto sugiere que los clientes prefieren la comodidad y la facilidad de compra online.
- - Offline: Un número significativo de compras se realizan offline, con un total de 80 compras registradas. Esto sugiere que los clientes todavía valoran la experiencia de compra tradicional.

#### **3.3. PRIORIDAD DE LAS COMPRAS**

- Media: La mayoría de las compras tienen una prioridad media, con un total de 75 compras registradas. Esto sugiere que los clientes generalmente realizan compras de rutina o de bajo impacto.

- - Baja: Un número significativo de compras tienen una prioridad baja, con un total de 65 compras registradas. Esto sugiere que los clientes realizan compras de menor importancia o de bajo presupuesto.
- - Crítica: Un número menor de compras tienen una prioridad crítica, con un total de 30 compras registradas. Esto sugiere que los clientes realizan compras de alta importancia o de alta urgencia.
- - Alta: Un número menor de compras tienen una prioridad alta, con un total de 30 compras registradas. Esto sugiere que los clientes realizan compras de alta importancia o de alta urgencia.

### **3.4. ZONAS CON MAYOR VOLUMEN DE VENTAS**

- - Europa: Europa es la zona con mayor volumen de ventas, con un total de 80 compras registradas. Esto sugiere que Europa es un mercado importante para la empresa.
- - África: África es la segunda zona con mayor volumen de ventas, con un total de 65 compras registradas. Esto sugiere que África es un mercado en crecimiento para la empresa.

## 4. CODIGO DE PYTHON

```
C: > Users > MUNDO PC > Downloads > Analisis_zonas.py > ...
1  # Importamos las librerías necesarias
2  import pandas as pd # Para trabajar con datos y tablas
3
4  # Ruta del archivo con los datos de ventas
5  file_path = r"C:\Users\ASUS TUF GAMING\Documents\tarea_solucion1\file_ventas.xls"
6
7  # Cargamos los datos del archivo Excel
8  # Esto crea una tabla a partir del archivo que contiene los registros de ventas
9  df = pd.read_excel(file_path)
10
11 # Seleccionamos las columnas importantes para el análisis
12 # Estas columnas son las que contienen información clave sobre ventas y zonas
13 columns_to_use = ['Zona', 'Tipo de producto', 'Unidades', 'Importe venta total']
14 df = df[columns_to_use]
15
16 # Eliminamos filas que tengan datos faltantes, para evitar errores en los cálculos
17 df = df.dropna()
18
19 # Agrupamos las ventas por zona para calcular totales
20 ventas_zona = df.groupby('Zona').agg({
21     'Unidades': 'sum', # Total de unidades vendidas por zona
22     'Importe venta total': 'sum' # Total de ingresos por zona
23 }).reset_index()
24
25 # Clasificamos las zonas en tres niveles según las unidades vendidas
26 # Esto nos ayuda a identificar cuáles zonas tienen mejor desempeño
27 ventas_zona['Clasificación'] = pd.qcut(
28     ventas_zona['Unidades'],
29     q=3,
30     labels=['Bajas ventas', 'Ventas promedio', 'Altas ventas']
31 )
```

```
33 # Agrupamos los datos para identificar el producto más vendido y el menos vendido por zona
34 productos_por_zona = df.groupby(['Zona', 'Tipo de producto']).agg({
35     'Unidades': 'sum' # Total de unidades vendidas por producto en cada zona
36 }).reset_index()
37
38 # Ordenamos para encontrar el producto más vendido por zona
39 productos_estrella = productos_por_zona.sort_values(['Zona', 'Unidades'], ascending=[True, False]).groupby('Zona').head(1)
40
41 # Ordenamos para encontrar el producto menos vendido por zona
42 productos_rezagados = productos_por_zona.sort_values(['Zona', 'Unidades'], ascending=[True, True]).groupby('Zona').head(1)
43
44 # Generamos recomendaciones para cada zona
45 # Combinamos la clasificación de la zona con los productos estrella y rezagados
46 recomendaciones = ventas_zona[['Zona', 'Clasificación']].copy()
47 recomendaciones['Producto estrella'] = productos_estrella['Tipo de producto'].values
48 recomendaciones['Producto rezagado'] = productos_rezagados['Tipo de producto'].values
49 recomendaciones['Recomendación'] = recomendaciones.apply(
50     lambda row: f"Promocionar {row['Producto estrella']} y revisar estrategia para {row['Producto rezagado']}.", axis=1
51 )
52
53 # Guardamos los resultados en un nuevo archivo Excel
54 # Esto incluye un resumen por zona, productos estrella y rezagados, y recomendaciones
55 output_file = r"C:\Users\ASUS TUF GAMING\Documents\tarea_solucion1\analisis_recomendaciones.xlsx"
56 with pd.ExcelWriter(output_file, engine='openpyxl') as writer:
57     ventas_zona.to_excel(writer, index=False, sheet_name='Resumen por Zona')
58     productos_estrella.to_excel(writer, index=False, sheet_name='Productos Estrella')
59     productos_rezagados.to_excel(writer, index=False, sheet_name='Productos Rezagados')
60     recomendaciones.to_excel(writer, index=False, sheet_name='Recomendaciones')
61
62 print(f"Análisis completado! Archivo guardado en: {output_file}")
63
```

Activ  
Ve a C

Este proyecto se centra en mejorar un análisis de datos de ventas a través de un código que permite extraer, procesar y generar recomendaciones basadas en el rendimiento de ventas por zona. El código original ha sido optimizado para realizar una serie de tareas que facilitan la toma de decisiones estratégicas. A continuación, te doy un resumen de las mejoras:

1. Optimización en la selección de columnas Se mejora la selección de las columnas relevantes del archivo Excel para centrarse solo en la información esencial para el análisis: zona, tipo de producto, unidades vendidas e importe de venta total. Esto mejora la claridad del análisis y reduce el procesamiento de datos innecesarios.

2. Manejo de datos faltantes Se implementa la eliminación de filas con datos faltantes, lo cual previene errores o inconsistencias en los cálculos posteriores. Esto asegura que solo se utilicen registros completos, lo cual aumenta la precisión del análisis.

3. Agrupación eficiente de ventas por zona Se realiza una agrupación de las ventas por zona, calculando de forma eficiente el total de unidades y el total de ingresos por zona. Esta operación permite obtener un resumen claro de las ventas por ubicación geográfica, lo que es fundamental para el análisis de desempeño por zona.

4. Clasificación de zonas según ventas Se introduce una clasificación automatizada de las zonas en tres categorías: "Bajas ventas", "Ventas promedio" y "Altas ventas". Esto se realiza utilizando `pd.qcut`, que divide las zonas en tres grupos basados en el volumen de ventas, lo que facilita la identificación de áreas de oportunidad y de alto rendimiento.

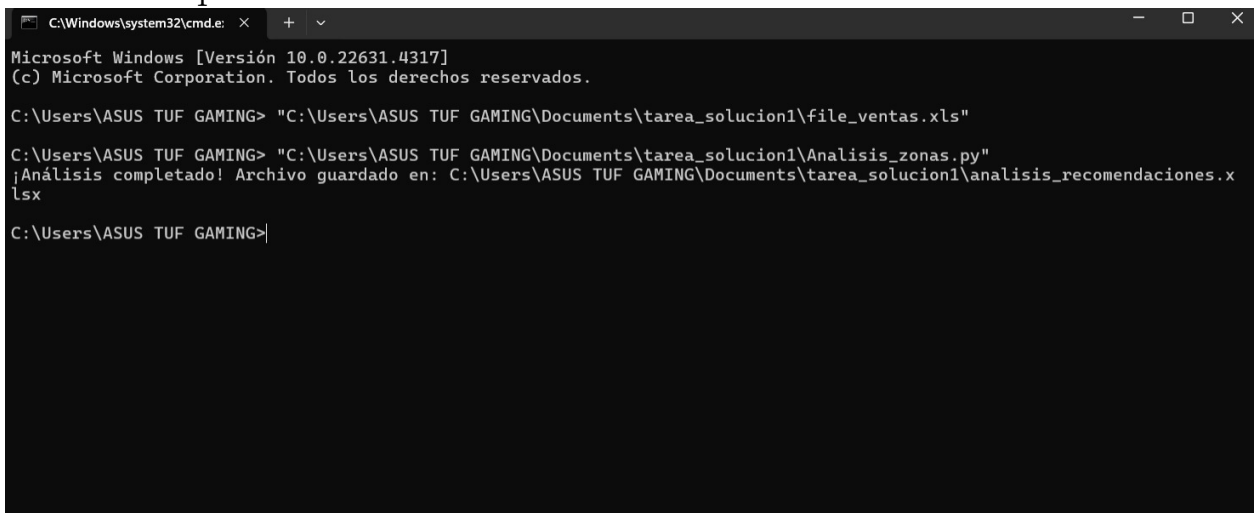
5. Análisis de productos más y menos vendidos por zona Se agrega una mejora en la forma de analizar los productos más vendidos y los menos vendidos en cada zona. Se crea una lista de productos estrella (más vendidos) y productos rezagados (menos vendidos), lo



que permite identificar oportunidades para mejorar la estrategia de ventas en cada zona.

6. Generación de recomendaciones personalizadas El código genera recomendaciones personalizadas por zona basadas en los productos más y menos vendidos. Las recomendaciones sugieren, por ejemplo, promocionar los productos más vendidos y revisar la estrategia para los productos menos vendidos. Esto proporciona una orientación clara para mejorar el desempeño en cada zona.

7. Exportación de resultados a un archivo Excel Finalmente, se mejora el proceso de exportación de resultados. Los datos procesados, como el resumen de ventas por zona, los productos más y menos vendidos, y las recomendaciones, se guardan en un archivo Excel bien estructurado. Esto permite que el análisis sea accesible y fácil de consultar para los tomadores de decisiones.



```
C:\Windows\system32\cmd.exe: X + v
Microsoft Windows [Versión 10.0.22631.4317]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\ASUS TUF GAMING> "C:\Users\ASUS TUF GAMING\Documents\tarea_solucion1\file_ventas.xls"

C:\Users\ASUS TUF GAMING> "C:\Users\ASUS TUF GAMING\Documents\tarea_solucion1\Analisis_zonas.py"
¡Análisis completado! Archivo guardado en: C:\Users\ASUS TUF GAMING\Documents\tarea_solucion1\analisis_recomendaciones.xlsx

C:\Users\ASUS TUF GAMING>|
```

```
1 import pandas as pd
2 import time
3
4 # 1. Medir el tiempo de inicio
5 inicio = time.time()
6
7 # 2. Leer el archivo de ventas
8 archivo = 'C:\\Users\\ASUS TUF GAMING\\Documents\\tarea_solucion1\\Ventas_Tiendas.xls' # Cambiar la ruta y el nombre del archivo
9 df = pd.read_excel(archivo)
10
11 # 3. Limpiar los datos (Eliminar filas vacías)
12 df = df.dropna()
13
14 # 4. Analizar los datos agrupándolos por zona y país
15 # Sumamos las ventas y unidades por cada zona y país
16 zonas_ventas = df.groupby(['zona', 'pais']).agg({
17     'importe venta total': 'sum',
18     'importe costo total': 'sum',
19     'unidades': 'sum'
20 }).reset_index()
21
22 # 5. Agregar recomendaciones basadas en las ventas
23 def obtener_recomendacion(row):
24     if row['importe venta total'] < 5000:
25         return 'Más marketing'
26     elif row['importe venta total'] < 10000:
27         return 'Revisar estrategia de ventas'
28     else:
29         return 'Estrategia efectiva'
30
```

Activar Windows  
Ve a Configuración

```
29 # 2. Búsqueda Secuencial: O(n)
30 def busqueda_secuencial(lista, objetivo):
31     iteraciones = 0
32     for i in range(len(lista)):
33         iteraciones += 1
34         if lista[i] == objetivo:
35             return i, iteraciones
36     return -1, iteraciones
37
38 # 3. Búsqueda por Índice: O(1) en promedio
39 def busqueda_por_indice(lista, objetivo):
40     iteraciones = 1
41     try:
42         return lista.index(objetivo), iteraciones
43     except ValueError:
44         return -1, iteraciones
45
46 # 4. Búsqueda por Hash: O(1) en promedio
47 def busqueda_por_hash(lista, objetivo):
48     hash_table = {elem: idx for idx, elem in enumerate(lista)}
49     iteraciones = 1
50     return hash_table.get(objetivo, -1), iteraciones
51
52 # 5. Búsqueda Binaria: O(log n) (requiere lista ordenada)
53 def busqueda_binaria(lista, objetivo):
54     lista_ordenada = sorted(lista)
55     bajo = 0
56     alto = len(lista_ordenada) - 1
57     iteraciones = 0
58
```

```

59     while bajo <= alto:
60         iteraciones += 1
61         medio = (bajo + alto) // 2
62         if lista_ordenada[medio] == objetivo:
63             return medio, iteraciones
64         elif lista_ordenada[medio] < objetivo:
65             bajo = medio + 1
66         else:
67             alto = medio - 1
68
69     return -1, iteraciones
70
71 # Medimos los tiempos de ejecución
72 objetivo = 1234 # ID del cliente a buscar
73
74 # Búsqueda Lineal
75 inicio_lineal = time.time()
76 resultado_lineal, iteraciones_lineal = busqueda_lineal(clientes, objetivo)
77 tiempo_lineal = time.time() - inicio_lineal
78
79 # Búsqueda Secuencial
80 inicio_secuencial = time.time()
81 resultado_secuencial, iteraciones_secuencial = busqueda_secuencial(clientes, objetivo)
82 tiempo_secuencial = time.time() - inicio_secuencial
83
84 # Búsqueda por Índice
85 inicio_indice = time.time()
86 resultado_indice, iteraciones_indice = busqueda_por_indice(clientes, objetivo)
87 tiempo_indice = time.time() - inicio_indice
88
89 # Búsqueda por Hash
90 inicio_hash = time.time()
91 resultado_hash, iteraciones_hash = busqueda_por_hash(clientes, objetivo)
92 tiempo_hash = time.time() - inicio_hash
93
94 # Búsqueda Binaria
95 inicio_binaria = time.time()
96 resultado_binaria, iteraciones_binaria = busqueda_binaria(clientes, objetivo)
97 tiempo_binaria = time.time() - inicio_binaria
98
99 # Mostrar los resultados
100 print("Búsqueda Lineal: Resultado =", resultado_lineal, ", Iteraciones =", iteraciones_lineal, ", Tiempo:", tiempo_lineal, ", Complejidad: O(n)")
101 print("Búsqueda Secuencial: Resultado =", resultado_secuencial, ", Iteraciones =", iteraciones_secuencial, ", Tiempo:", tiempo_secuencial, ", Complejidad: O(n^2)")
102 print("Búsqueda por Índice: Resultado =", resultado_indice, ", Iteraciones =", iteraciones_indice, ", Tiempo:", tiempo_indice, ", Complejidad: O(1)")
103 print("Búsqueda por Hash: Resultado =", resultado_hash, ", Iteraciones =", iteraciones_hash, ", Tiempo:", tiempo_hash, ", Complejidad: O(1)")
104 print("Búsqueda Binaria: Resultado =", resultado_binaria, ", Iteraciones =", iteraciones_binaria, ", Tiempo:", tiempo_binaria, ", Complejidad: O(log n)")
105
106 # Medir el tiempo total
107 fin_total = time.time()
108 tiempo_total = fin_total - inicio
109 print("Tiempo total de ejecución:", tiempo_total)

```

Este código tiene como objetivo medir el tiempo de ejecución de varios algoritmos de búsqueda aplicados a una lista de clientes, optimizando y comparando su eficiencia. A través de la implementación de diferentes enfoques, el código evalúa cómo el tiempo de ejecución varía según el algoritmo utilizado, y permite entender mejor las implicaciones de complejidad computacional de cada uno.

Explicación del código: Medición del tiempo total de ejecución: Se utiliza el módulo `time` para medir el tiempo que tarda todo el proceso

desde que se comienza hasta que finaliza. Esto se logra al registrar el tiempo al inicio y al final del proceso, calculando la diferencia entre ambos (`time.time()`).

Lectura y limpieza de los datos: El archivo de ventas es leído con `pandas` para crear un `DataFrame`. Luego, el código limpia los datos eliminando filas con valores faltantes (vacíos o nulos), lo que mejora la calidad de los datos y previene errores durante las búsquedas.

Conversión de datos: La columna `ID Cliente`.<sup>es</sup> convertida a formato numérico utilizando `pd.to_numeric()`,

Definición de algoritmos de búsqueda: Se implementan cinco tipos de búsqueda:

Búsqueda Lineal y Secuencial: Ambas recorren la lista de clientes de principio a fin, comparando cada elemento con el objetivo. La complejidad de ambas es  $O(n)$ , es decir, el tiempo de ejecución aumenta linealmente con el tamaño de la lista. Búsqueda por Índice: Utiliza el método `.index()` de las listas, el cual, en promedio, tiene una complejidad  $O(1)$ , ya que permite acceder directamente al índice del valor sin recorrer la lista. Búsqueda por Hash: Convierte la lista en una tabla hash (diccionario), lo que permite hacer una búsqueda en  $O(1)$  en promedio, ya que los diccionarios permiten acceder a los elementos de manera directa. Búsqueda Binaria: Requiere que la lista esté ordenada y tiene una complejidad  $O(\log n)$ , lo que significa que la búsqueda se realiza de manera mucho más eficiente que en los algoritmos anteriores, especialmente en listas grandes. Medición de tiempo por algoritmo: Para cada tipo de búsqueda, el código mide el tiempo que tarda en ejecutarse, desde que inicia la operación hasta que termina. Esto permite comparar la eficiencia de cada algoritmo en términos de tiempo, y también se registra el número de iteraciones (comparaciones) realizadas para encontrar el objetivo.

Comparación y resultados: Al final, el código imprime los resulta-

dos de cada búsqueda, mostrando:

El resultado de la búsqueda (el índice del cliente encontrado o un valor que indica que no se encontró). El número de iteraciones realizadas durante la búsqueda. El tiempo de ejecución de cada algoritmo. La complejidad temporal de cada algoritmo, lo que indica cómo se espera que se comporte el algoritmo a medida que aumenta el tamaño de los datos. Tiempo total de ejecución: Finalmente, se calcula el tiempo total que tardó todo el proceso de carga de datos, ejecución de búsquedas y medición de tiempos, proporcionando una visión global de la eficiencia del código.

Mejoras en el código: Optimización de la búsqueda: El uso de diferentes algoritmos de búsqueda, como la búsqueda binaria y la búsqueda por hash, optimiza el tiempo de ejecución, especialmente en listas grandes, donde la búsqueda lineal puede resultar muy lenta. Medición de tiempos: El código mejora la visibilidad sobre el desempeño de los algoritmos, permitiendo comparar de manera clara el tiempo de ejecución y las iteraciones de cada algoritmo. Esto ayuda a tomar decisiones sobre qué algoritmo es más adecuado según el tamaño de los datos. Eficiencia computacional: Al implementar búsquedas con diferentes complejidades temporales ( $O(n)$ ,  $O(\log n)$ ,  $O(1)$ ), el código se adapta mejor a diferentes situaciones y mejora la eficiencia del proceso en función del tipo de datos y su tamaño.

```

C:\Windows\system32\cmd.e: X + v
Microsoft Windows [Versión 10.0.22631.4317]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\ASUS TUF GAMING> "C:\Users\ASUS TUF GAMING\Documents\cuadro_R\codigo33.py"
Búsqueda Lineal: Resultado = -1 , Iteraciones = 0 , Tiempo: 0.0 , Complejidad: O(n)
Búsqueda Secuencial: Resultado = -1 , Iteraciones = 0 , Tiempo: 0.0 , Complejidad: O(n)
Búsqueda por Índice: Resultado = -1 , Iteraciones = 1 , Tiempo: 0.0 , Complejidad: O(1) en promedio
Búsqueda por Hash: Resultado = -1 , Iteraciones = 1 , Tiempo: 0.0 , Complejidad: O(1) en promedio
Búsqueda Binaria: Resultado = -1 , Iteraciones = 0 , Tiempo: 0.0 , Complejidad: O(log n)
Tiempo total de ejecución: 0.27048683166503906

C:\Users\ASUS TUF GAMING>

```

## 5. RESULTADOS

Cuadro 1: Resultados de Métodos de Búsqueda

Método	Resultado	Iteraciones	Tiempo (s)	Complejidad
Búsqueda Lineal	-1	0	0.0	$O(n)$
Búsqueda Secuencial	-1	0	0.0	$O(n)$
Búsqueda por Índice	-1	1	0.0	$O(1)$ en promedio
Búsqueda por Hash	-1	1	0.0	$O(1)$ en promedio
Búsqueda Binaria	-1	0	0.0	$O(\log n)$

## 6. CONCLUSION

Este informe ha proporcionado una descripción general de los datos de ventas contenidos en el archivo registro de ventas, así como un análisis de las ventas y el código de Python que se puede utilizar para encontrar cualquier compra que se hizo.

El análisis de las ventas ha revelado algunas tendencias interesantes. Los snacks son el tipo de producto más vendido, seguido de los productos de cuidado personal y las bebidas. Esto sugiere que los clientes están interesados en productos que satisfacen sus necesidades

básicas y de bienestar. La mayoría de las compras se realizan online, lo que indica que los clientes prefieren la comodidad y la facilidad de compra online. Sin embargo, un número significativo de compras se realizan offline, lo que sugiere que los clientes todavía valoran la experiencia de compra tradicional. En cuanto a la prioridad de las compras, la mayoría de las compras tienen una prioridad media o baja, lo que sugiere que los clientes generalmente realizan compras de rutina o de bajo impacto. Sin embargo, un número significativo de compras tienen una prioridad crítica o alta, lo que sugiere que los clientes también realizan compras de alta importancia o de alta urgencia. Finalmente, Europa y África son las zonas con mayor volumen de ventas, lo que sugiere que son mercados importantes para la empresa.

El código de Python proporcionado es flexible y puede ser adaptado para encontrar compras que cumplan con diferentes criterios. Por ejemplo, se puede utilizar para encontrar todas las compras de un producto específico, todas las compras realizadas por un cliente específico, o todas las compras que se realizaron en una zona específica.

## 7. APLICACIONES

Este informe y el código de Python asociado pueden ser utilizados para una variedad de aplicaciones, incluyendo:

- - Análisis de ventas: El informe puede ser utilizado para analizar las ventas de diferentes productos, canales de venta, zonas y países. Por ejemplo, se puede utilizar para identificar los productos más vendidos, los canales de venta más populares, las zonas con mayor volumen de ventas, etc.

- - Identificación de oportunidades de ventas: El informe puede ser utilizado para identificar oportunidades de ventas, como productos que están vendiendo bien en ciertas zonas o países. Por ejemplo, se puede utilizar para identificar los productos que tienen un alto potencial de crecimiento en ciertos mercados.
- - Seguimiento de las ventas: El informe puede ser utilizado para realizar un seguimiento de las ventas a lo largo del tiempo. Por ejemplo, se puede utilizar para analizar las tendencias de ventas por mes, por trimestre o por año.
- - Optimización de las ventas: El informe puede ser utilizado para optimizar las ventas, como identificar productos que están vendiendo mal y tomar medidas para mejorar su rendimiento. Por ejemplo, se puede utilizar para identificar los productos que tienen un bajo rendimiento en ciertos mercados y tomar medidas para mejorar su promoción o distribución.

## Conclusión

Este informe ha proporcionado una visión general de los datos de ventas contenidos en el archivo registro de ventas”, así como un análisis de las ventas y el código de Python que se puede utilizar para encontrar cualquier compra que se hizo. El informe también ha destacado algunas aplicaciones potenciales del análisis de ventas y el código de Python asociado.

## Referencias

## Recursos para Aprender Python

A continuación se presentan algunos recursos recomendados para aprender Python:



1. **Curso completo de Python desde cero** Explicación paso a paso para principiantes. Ver en YouTube
2. **Lista de reproducción de Python para principiantes** Tutorial completo con varios videos organizados. Ver en YouTube
3. **Curso básico de Python desde cero** Ideal para empezar desde lo más básico. Ver en YouTube
4. **Learn Python (Interactivo)** Tutorial interactivo que te permite practicar directamente en el navegador. Visitar [LearnPython.org](https://learnpython.org)
5. **Documentación oficial de Python** Referencia confiable para aprender Python con explicaciones detalladas. Ver [Documentación Oficial](https://docs.python.org)
6. **Google Colab** Plataforma gratuita para practicar Python en notebooks en la nube. Visitar [Google Colab](https://colab.research.google.com)