Trabajo grupal o individual

Identificación del trabajo

a. **Módulo:** 4
b. **Asignatura:** Procesamiento de Datos
c. **RA:** Examina los datos mediante análisis exploratorios, aplicando las librerías base específicas, a partir de casos prácticos.

d. **Docente Online:** Gonzalo Esteban Cárdenas Rubio
e. **Fecha de entrega**:

Identificación del estudiante

Nombre y apellido	**Carrera**
Endert Guerrero	Data Science

Introducción

El Análisis Exploratorio de Datos (AED) es una etapa fundamental en el proceso de análisis de datos que tiene como objetivo explorar y comprender los datos antes de realizar un análisis más profundo o aplicar técnicas avanzadas. A través del AED, podemos obtener una visión general de la estructura y características de los datos, identificar patrones, tendencias y valores atípicos, y tomar decisiones informadas sobre cómo proceder en el análisis. Se realizan diversas tareas durante el AED, como la inspección de datos, el cálculo de estadísticas descriptivas, la visualización de gráficos y la limpieza de datos. Además, el AED es una etapa iterativa que nos permite identificar nuevas preguntas y áreas de interés a medida que exploramos y obtenemos información. Con el uso de librerías como NumPy, Matplotlib y Pandas en Python, podemos llevar a cabo un análisis exploratorio exhaustivo que nos ayudará a descubrir información relevante y tomar decisiones fundamentadas en nuestros datos.

Desarrollo

DataFrame

Importamos el DataFrame de la actividad del M3 y solucionamos sus errores.

```
Ventas_Clientes_2021 =
pd.Series([100.0,200.0,"110,0",90.0,150.0,80.0,70.0,100.0,100.00,110.0,90.0,"1
00,00"])

Ventas_Clientes_2021 = Ventas_Clientes_2021.replace({',': '.', '"': ''},
regex=True).astype(float)


Clientes_2021 = pd.Series
(["JUan","camila","DanIel","Luis","Juan","Ana","Camila","Ana","LuiS","Juan","A
na","Juan"])

Clientes_2021 = Clientes_2021.str.title()
```

Ítem 1

Creamos y mostramos en pantalla el DataFrame

```
df = pd.DataFrame({"Clientes":Clientes_2021,"Ventas":Ventas_Clientes_2021})

print(df)
```

[redacted]

Agrupamos por clientes y organizamos de mayor a menor

Ahora crearemos un nuevo DataFrame ("df2") donde agruparemos a los clientes y sumaremos sus compras.

```
df2 = df.groupby("Clientes").sum()

print(df2)
```

```
Luis        190.0
```

También podemos disponer dicho DataFrame de forma descendente para que, al mostrarlo en pantalla, podamos ver cuánto fue el total de las compras realizadas por el cliente organizado jerárquicamente según dicho total.

```
jerarquia = df2.sort_values(by="Ventas", ascending = 0)

print(jerarquia)
```

Clientes	Ventas
Juan	460.0
Ana	270.0
Camila	270.0
Luis	190.0
Daniel	110.0

Generamos un gráfico de barras.

Usaremos la función plot de matplotlib agregando el valor "bar" para obtener el gráfico deseado, haciéndolo de esta forma nos ahorramos el código para separar el índice de sus valores para generar los ejes, también así nos genera automáticamente el "label" de "Clientes". agregaremos el color en este caso "darkblue" en la misma función "df2.plot", especificaremos el "ylabel" y "title" para así tener un gráfico más completo.

en este ejemplo la separación y ancho de las barras se genera automáticamente, pero esto valores también los podemos modificar agregando "width" en la función "df2.plot" o usando la función "plt.bar()".

En el ejemplo de estudio no existe una leyenda como se puede ver en este en la parte superior derecha, está la podemos modificar usando la función "plt.legend", para este caso debemos acompañar la función con ".set_visible(False)".

```
df2.plot(kind = "bar", color = "darkblue")

plt.ylabel("Ventas en Millones")

plt.title("Ventas del año 2021 por cliente")


plt.show()
```

Ítem 2

Análisis de los datos

Ahora realizaremos un análisis y mostraremos los resultados para así sacar conclusiones más fácilmente.

Primero mostraremos la media, mínimo, máximo y desviación estándar usando las funciones pertinentes que nos ofrece Pandas para realizar esta operación.

```
med_gral = df["Ventas"].mean()

min_gral = df["Ventas"].min()

max_gral = df["Ventas"].max()

des_est = df["Ventas"].std()


print("Media de ventas durante 2021 en millones:", med_gral)

print("Menor venta durante 2021 en millones:", min_gral)

print("Mayor venta durante 2021 en millones:", max_gral)

print("Desviación estándar de las ventas durante 2021:", des_est)
```

```
Media de ventas durante 2021 en millones: 108.33333333333333
Menor venta durante 2021 en millones: 70.0
Mayor venta durante 2021 en millones: 200.0
Desviación estándar de las ventas durante 2021: 34.859023439441266
```

A partir de este análisis general podemos llegar a las siguientes conclusiones:

- La media de las ventas durante 2021 fue de 108.33 millones.

- La venta individual más pequeña durante el mismo ciclo fue de 70.0 millones.

- La venta individual más grande del 2021 fue de 200.00 millones

- La desviación estándar nos indica cuán dispersos están los datos en torno a la media, como podemos ver en el caso presentan una dispersión de 34.85 millones, esto significa que las ventas individuales se desvían en promedio aproximadamente 34.85 millones de la media de las ventas.

Ítem 3

Para hallar datos atípicos debemos hacer un estudio más completo de los datos por lo comenzare por hacer un estudio individual por clientes.

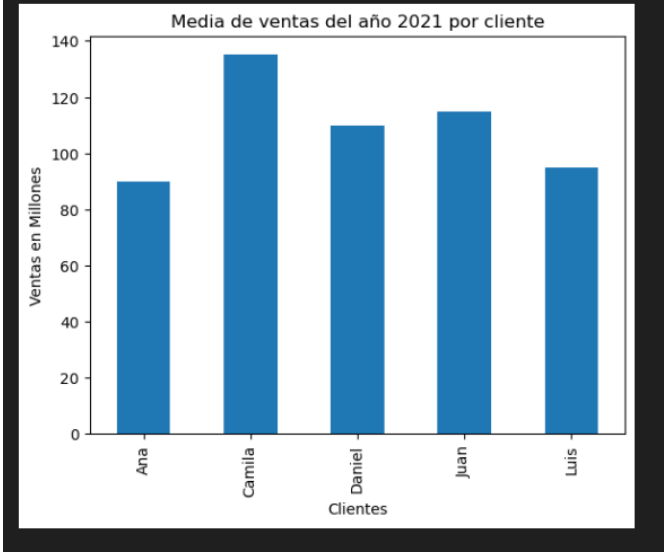
Estudio individual

También podemos generar la media, mínimo y máximo de las compras para cada cliente individualmente haciendo uso de "groupby" nuevamente.

Media

```
med_ind = df.groupby("Clientes")["Ventas"].mean()

print("Media de compras por cliente:")

print(med_ind)


med_ind.plot(kind="bar")


plt.ylabel("Ventas en Millones")

plt.title("Media de ventas del año 2021 por cliente")


plt.show()
```

```
Media de compras por cliente:
Clientes
Ana       90.0
Camila   135.0
Daniel   110.0
Juan     115.0
Luis      95.0
Name: Ventas, dtype: float64
```

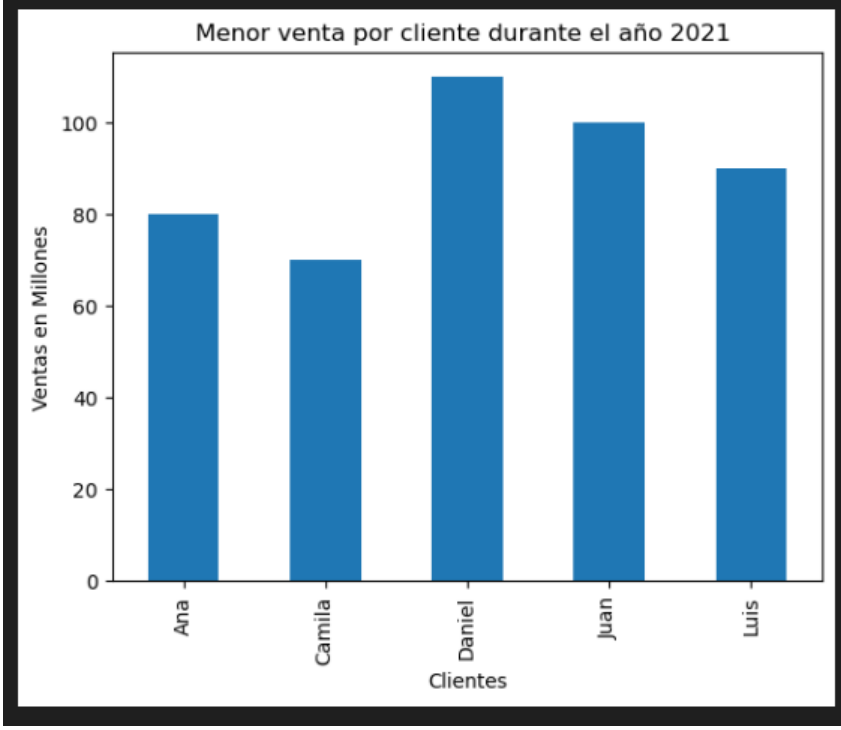


La media individual nos muestra que el cliente con mejor media fue Camila, mientras que el peor fue Ana, estos datos los podríamos comparar con la cantidad de compras que realizo cada cliente durante el ciclo para obtener un análisis más profundo.

Mínimo

```
min_ind = df.groupby("Clientes")["Ventas"].min()

print("Menor compra por cliente:")

print(min_ind)


min_ind.plot(kind="bar")

plt.ylabel("Ventas en Millones")

plt.title("Menor venta por cliente durante el año 2021 ")


plt.show()
```

```
Menor compra por cliente:
Clientes
Ana        80.0
Camila     70.0
Daniel    110.0
Juan      100.0
Luis       90.0
Name: Ventas, dtype: float64
```

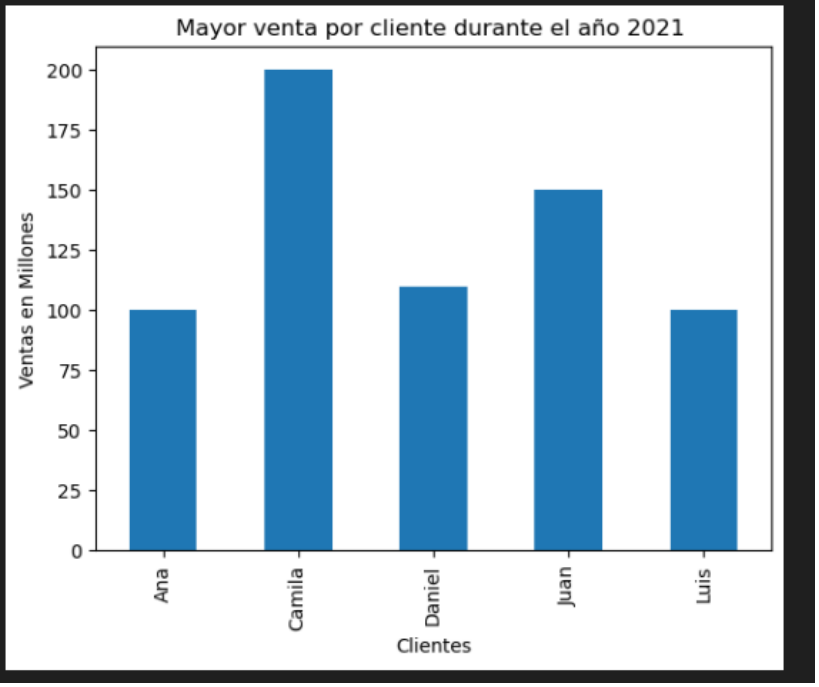


Ahora podemos ver que la menor compra individual la realizo Camila dato de bastante valor, mientras que Daniel entre las compras que hizo el menor valor fue la más alta en esta escala, pero también estos datos los podemos correlacionar con la cantidad de comprar realizadas por los clientes.

Máximo

```
max_ind = df.groupby("Clientes")["Ventas"].max()

print("Mayor compra por cliente:")

print(max_ind)


max_ind.plot(kind="bar")

plt.ylabel("Ventas en Millones")

plt.title("Mayor venta por cliente durante el año 2021 ")


plt.show()
```

```
Mayor compra por cliente:
Clientes
Ana       100.0
Camila    200.0
Daniel    110.0
Juan      150.0
Luis      100.0
Name: Ventas, dtype: float64
```



Parecido al estudio anterior podemos ver la mayor compra individual realizada por cliente, Camila realizo la compra individual de mayor valor.

Correlaciones de los datos obtenidos

Ahora mostrare los datos en un mismo DataFrame para poder estudiarlo y llegar a mejores conclusiones, además de agregar un dato muy importante, Cantidad de transacciones realizada por el cliente.

```
cant_compras = df["Clientes"].value_counts()

corr_cant_compras = pd.DataFrame({"Media": med_ind, "Menor compra": min_ind,
"Mayor compra": max_ind, "Cantidad de compras": cant_compras})

print(corr_cant_compras)
```

```
        Media  Menor compra  Mayor compra  Cantidad de compras
Ana      90.0          80.0         100.0                    3
Camila  135.0          70.0         200.0                    2
Daniel  110.0         110.0         110.0                    1
Juan    115.0         100.0         150.0                    4
Luis     95.0          90.0         100.0                    2
```

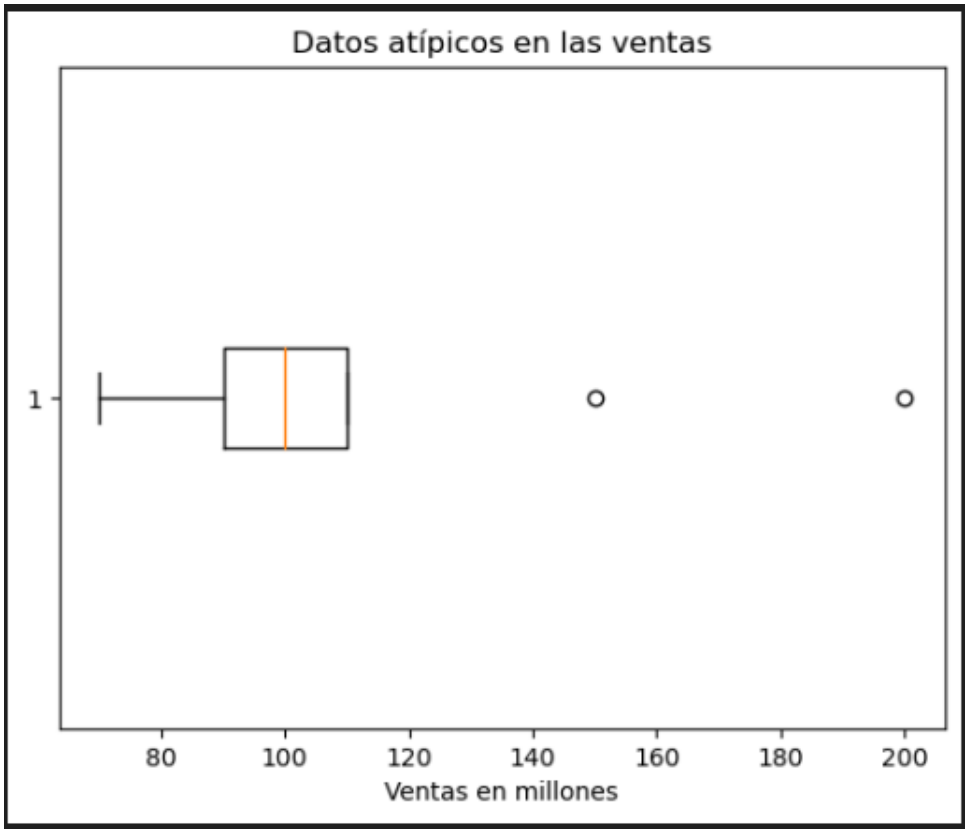
Como dije anteriormente Camila tiene la compra de mayor y menor valor, este dato nos puede ayudar a entender cuál puede ser el dato atípico.

Datos Atípicos

Ahora a través de un boxplot podemos ver si existe un outlier y si uno o este es Camila.

Boxplot

```
plt.boxplot(df["Ventas"], vert=False)

plt.title("Datos atípicos en las ventas")

plt.xlabel("Ventas en millones")

plt.show()


print(df)
```



A través del Boxplot podemos ver que tenemos dos Outliers que a diferencia de lo que pensaba la Venta realizada a Camila por 70 millones no es uno de ellos, pero si la venta por 200 millones, también podemos ver que la venta realizada a Juan por 150 millones es otro outlier, por lo que estos datos los podríamos descartar si así quisiéramos.

Encontrar Outliers calculando los Quintiles

También podemos encontrar los Outliers calculando los quintiles de forma manual siguiendo la formula:

IQR = Q3 - Q1

Esta fórmula nos calcula el rango intercuartílico, que es igual al tercer cuartil menos el primer cuartil, los cuales hallamos haciendo uso de la función "np.quantile", por lo que no debemos restarlos para hallar dicho rango ya que NumPy lo hace automáticamente a traves de esa función, lo que si debemos hacer ahora es calcular los limites inferior (LI) y superior (LS) según la siguiente formula.

LI = Q1 - 1.5 * IQR (o lo que es lo mismo) Q3 - Q1

LS = Q3 + 1.5 * IQR

Ya habiendo calculado esto podemos mostrar los resultados por pantalla luego de ingresar los resultados de estos quintiles en una nueva variable que llamare Outliers.

```
Q1 = np.quantile(df["Ventas"], 0.25)
```

```
Q3 = np.quantile(df["Ventas"], 0.75)

LI = Q1 - 1.5 * (Q3 - Q1)
LS = Q3 + 1.5 * (Q3 - Q1)

outliers = df[(df["Ventas"] < LI) | (df["Ventas"] > LS)]

print("Valores atípicos:")
print(outliers)
```

```
Valores atípicos:
  Clientes  Ventas
1   Camila   200.0
4     Juan   150.0
```

Análisis correcto con DataFrame sin Outliers

Ahora crearemos un nuevo DataFrame sin los outliers el cual también podemos mostrar gráficamente, esto lo haremos usando un código similar de la variable "outliers" a diferencia de que invertiremos los operadores de comparación y agregaremos el operado &, haciendo esto nos mostrara los datos dentro del rango deseado.

```
fin = df[(df["Ventas"] >= LI) & (df["Ventas"] <= LS)]
print(fin)
```

Análisis

```
media = fin["Ventas"].mean()
minimo = fin["Ventas"].min()
maximo = fin["Ventas"].max()
desviacion_estandar = fin["Ventas"].std()

print("Media de ventas durante 2021 en millones:", media)
print("Menor venta durante 2021 en millones:", minimo)
print("Mayor venta durante 2021 en millones:", maximo)
print("Desviación estándar de las ventas durante 2021:", desviacion_estandar)
```

Ahora según este nuevo análisis de sin Outliers nos dice que:

- La media de ventas fue de 95.0 millones.
- La menor venta sigue siendo 70 millones.
- La mayor venta fue 110 millones.
- Y la desviación estándar fue de 12.70 millones.

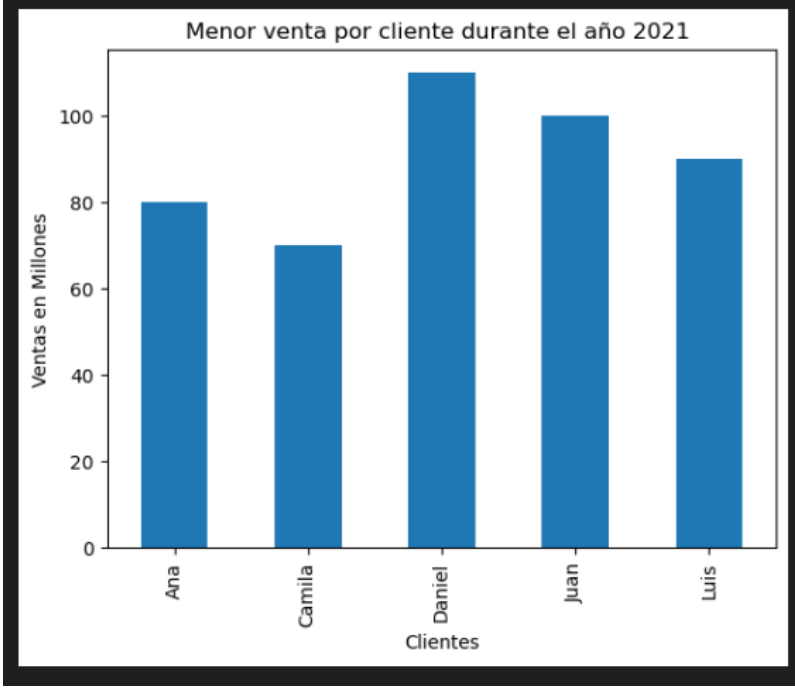
Media individual

```
med_ind2 = fin.groupby("Clientes")["Ventas"].mean()
print("Media de compras por cliente:")
print(med_ind2)

med_ind2.plot(kind="bar")


plt.ylabel("Ventas en Millones")
plt.title("Media de ventas del año 2021 por cliente")

plt.show()
```

Mínimo Individual

```
min_ind2 = fin.groupby("Clientes")["Ventas"].min()
print("Menor compra por cliente:")
print(min_ind2)

min_ind2.plot(kind="bar")


plt.ylabel("Ventas en Millones")
plt.title("Menor venta por cliente durante el año 2021 ")
```

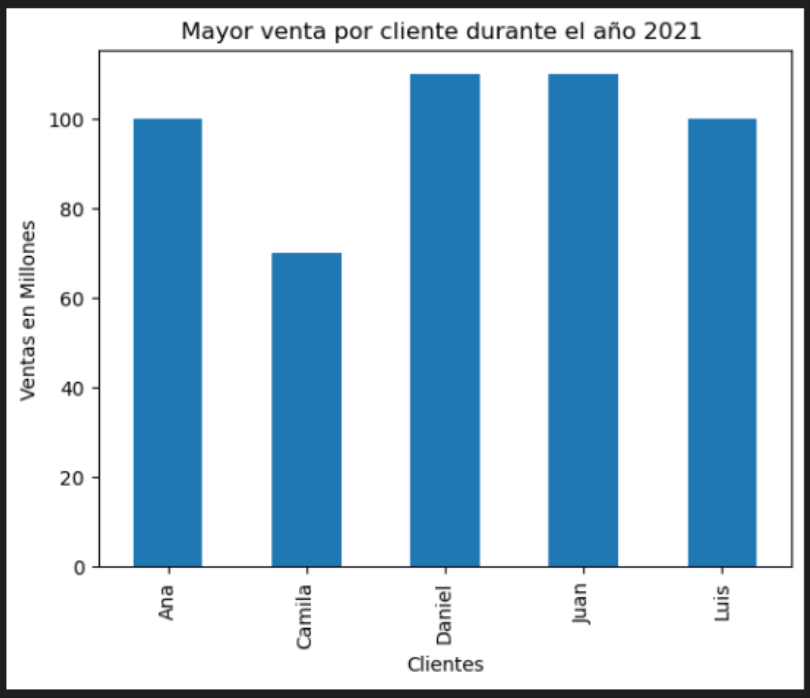
```
Menor compra por cliente:
Clientes
Ana        80.0
Camila     70.0
Daniel    110.0
Juan      100.0
Luis       90.0
Name: Ventas, dtype: float64

Text(0.5, 1.0, 'Menor venta por cliente durante el año 2021 ')
```



Máximo Individual

```
max_ind2 = fin.groupby("Clientes")["Ventas"].max()
print("Mayor compra por cliente:")
print(max_ind2)


max_ind2.plot(kind="bar")


plt.ylabel("Ventas en Millones")
plt.title("Mayor venta por cliente durante el año 2021 ")


plt.show()
```

```
Mayor compra por cliente:
Clientes
Ana       100.0
Camila     70.0
Daniel    110.0
Juan      110.0
Luis      100.0
Name: Ventas, dtype: float64
```



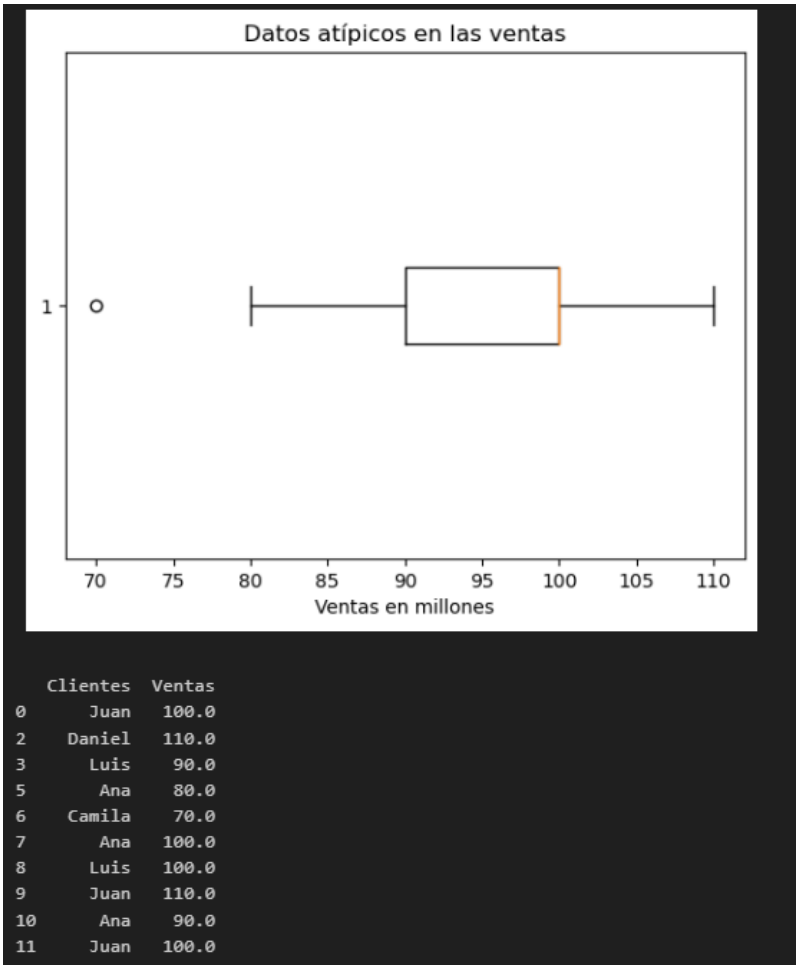
Correlación

```
corr_cant_compras2 = pd.DataFrame({"Media": med_ind2, "Menor compra":
min_ind2, "Mayor compra": max_ind2, "Cantidad de compras": cant_compras})

print(corr_cant_compras2)
```

Boxplot

Ahora haremos un boxplot para revisar nuevamente que no hallan outliers.

```
plt.boxplot(fin["Ventas"], vert=False)

plt.title("Datos atípicos en las ventas")

plt.xlabel("Ventas en millones")

plt.show()

print(fin)
```



	Clientes	Ventas
0	Juan	100.0
2	Daniel	110.0
3	Luis	90.0
5	Ana	80.0
6	Camila	70.0
7	Ana	100.0
8	Luis	100.0
9	Juan	110.0
10	Ana	90.0
11	Juan	100.0

Como pensaba al antes de ejecutar el primer Boxplot la venta por 70 millones también es un outlier, por lo que este también debemos descartarlo.

Análisis Final

Encontramos y descartamos el nuevo outlier

```
Q1_2 = np.quantile(fin["Ventas"], 0.25)
Q3_2 = np.quantile(fin["Ventas"], 0.75)


LI_2 = Q1_2 - 1.5 * (Q3_2 - Q1_2)
LS_2 = Q3_2 + 1.5 * (Q3_2 - Q1_2)


outliers2 = fin[(fin["Ventas"] < LI_2) | (fin["Ventas"] > LS_2)]


print("Valores atípicos:")
print(outliers2)
```

```
Valores atípicos:
  Clientes  Ventas
6   Camila    70.0
```

Mostramos en pantalla el DataFrame definitivo.

```
fin2 = fin[(fin["Ventas"] >= LI_2) & (fin["Ventas"] <= LS_2)]
print(fin2)
```

```
   Clientes  Ventas
0      Juan   100.0
2    Daniel   110.0
3      Luis    90.0
5       Ana    80.0
7       Ana   100.0
8      Luis   100.0
9      Juan   110.0
10      Ana    90.0
11     Juan   100.0
```

Boxplot sin datos atípicos

```
plt.boxplot(fin2["Ventas"], vert=False)
plt.title("Datos atípicos en las ventas")
plt.xlabel("Ventas en millones")
plt.show()
```


Conclusión del análisis

```
med_ind3 = fin2.groupby("Clientes")["Ventas"].mean()
print("Media de compras por cliente:")
print(med_ind3)


med_ind3.plot(kind="bar")
plt.ylabel("Ventas en Millones")
plt.title("Media de ventas del año 2021 por cliente")
```

```
plt.show()


min_ind3 = fin2.groupby("Clientes")["Ventas"].min()
print("Menor compra por cliente:")
print(min_ind3)


min_ind3.plot(kind="bar")
plt.ylabel("Ventas en Millones")
plt.title("Menor venta por cliente durante el año 2021")
plt.show()


max_ind3 = fin2.groupby("Clientes")["Ventas"].max()
print("Mayor compra por cliente:")
print(max_ind3)

max_ind3.plot(kind="bar")
plt.ylabel("Ventas en Millones")
plt.title("Mayor venta por cliente durante el año 2021 ")
plt.show()


corr_cant_compras3 = pd.DataFrame({"Media": med_ind3, "Menor compra":
min_ind3, "Mayor compra": max_ind3, "Cantidad de compras": cant_compras})
print(corr_cant_compras3)


print("")


media3 = fin2["Ventas"].mean()
minimo3 = fin2["Ventas"].min()
maximo3 = fin2["Ventas"].max()
desviacion_estandar3 = fin2["Ventas"].std()
```

```
print("Media de ventas durante 2021 en millones:", media3)
print("Menor venta durante 2021 en millones:", minimo3)
print("Mayor venta durante 2021 en millones:", maximo3)
```


```
             Media  Menor compra  Mayor compra  Cantidad de compras
Ana      90.000000          80.0         100.0                    3
Camila         NaN           NaN           NaN                    2
Daniel  110.000000         110.0         110.0                    1
Juan    103.333333         100.0         110.0                    4
Luis     95.000000          90.0         100.0                    2

Media de ventas durante 2021 en millones: 97.77777777777777
Menor venta durante 2021 en millones: 80.0
Mayor venta durante 2021 en millones: 110.0
Desviación estándar de las ventas durante 2021: 9.718253158075502
```

Como conclusión Camila y sus compras son un dato atípico, mientras que la compra de 150 millones realizada por Juan también lo es, tenemos que la media de ventas fue de 97.7 millones, la menor venta fue de 80 millones, la mayor de 110 millones y la desviación estándar fue de 9.7 millones.

Conclusión

En este análisis exploratorio de datos, hemos utilizado Python y diversas librerías, como NumPy, Matplotlib y Pandas, para examinar y comprender los datos relacionados con las ventas de clientes durante el año 2021.

Después de importar el DataFrame y solucionar los errores, obtuvimos un panorama general de los datos, donde identificamos los clientes y sus respectivas ventas. Agrupamos los datos por cliente, lo que nos permitió identificar el total de compras realizadas por cada cliente.

Luego, procedimos a visualizar la distribución de las ventas mediante un gráfico de barras, que nos mostró una representación clara de las ventas de cada cliente.

En el siguiente análisis, calculamos estadísticas descriptivas, como la media, el mínimo, el máximo y la desviación estándar de las ventas durante el año 2021. Además, realizamos un estudio individual para cada cliente, calculando sus respectivas medias, mínimos y máximos de compras.

En el proceso de detección de valores atípicos, utilizamos un boxplot y cálculos de quintiles para identificar aquellos datos que estaban fuera del rango esperado. Observamos dos clientes con valores atípicos en sus compras: Camila con sus dos compras (70 y 200 millones) y Juan con una compra de 150 millones.

Para eliminar los outliers y realizar un análisis más preciso, creamos un nuevo DataFrame sin estos datos atípicos. Así, pudimos calcular nuevas estadísticas descriptivas y realizar un análisis más detallado, centrándonos en los clientes sin valores extremos.

En conclusión, el análisis exploratorio de datos nos ha permitido comprender mejor las ventas de los clientes durante el año 2021. Hemos identificado patrones, tendencias y valores atípicos en las compras. Gracias a este análisis, hemos obtenido valiosa información para la toma de decisiones.

Bibliografia

- Zach. (2022, 15 marzo). *Pandas: How to create bar plot from GroupBy*. Statology. Recuperado 3 de agosto de 2023, de https://www.statology.org/pandas-groupby-bar-plot/#:~:text=Pandas%3A%20How%20to%20Create%20Bar%20Plot%20from%20GroupBy,df.groupby(['group_var'])%20['values_var'].sum()%20%23create%20bar%20plot%20by%20group%20df_groups.plot(kind%3D'bar')

- Hunter, J., Dale, D., Firing, E., Droettboom, M., & Matplotlib development team. (s. f.). *Matplotlib.pyplot.bar — matplotlib 3.7.2 documentation*. Recuperado 3 de agosto de 2023, de https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html

- Hunter, J., Dale, D., Firing, E., Droettboom, M., & Matplotlib development team. (s. f.). *Legend guide — matplotlib 3.7.2 documentation*. Recuperado 3 de agosto de 2023, de https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html

- Hunter, J., Dale, D., Firing, E., Droettboom, M., & Matplotlib development team. (s. f.). *matplotlib.pyplot.boxplot — matplotlib 3.7.2 documentation*. Recuperado 3 de agosto de 2023, de https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.boxplot.html

- Sotaquirá, M. (2021). *¿Como hacer el análisis exploratorio de datos?* [PDF].

- Bytepeaker. (2021, 29 agosto). *Detección y tratamiento de valores atípicos | Cómo manejar valores atípicos*. Datapeaker. Recuperado 3 de agosto de 2023, de https://datapeaker.com/big-data/deteccion-y-tratamiento-de-valores-atipicos-como-manejar-valores-atipicos/#:~:text=4.1%20Detecci%C3%B3n%20de%20valores%20at%C3%ADpicos%20mediante%20Boxplot%3A%20El,vert%3DFalse)%20plt.title%20(%22Detecting%20outliers%20using%20Boxplot%22)%20plt.xlabel%20('Sample')
- Rodríguez, D. (2022, 11 agosto). *Gráficos Boxplot con Matplotlib en Python*. Analytics Lane. Recuperado 3 de agosto de 2023, de https://www.analyticslane.com/2022/08/11/graficos-boxplot-con-matplotlib-en-python/
- S/N. (2021, 9 diciembre). *Diagrama de caja y bigotes (Boxplot)*. Probabilidad y Estadística. Recuperado 3 de agosto de 2023, de https://www.probabilidadyestadistica.net/diagrama-de-caja-y-bigotes-boxplot/#:~:text=Para%20hacer%20un%20diagrama%20de%20caja%20y%20bigotes,LI%20o%20mayores%20que%20LS.%20. . .%20M%C3%A1s%20elementos
- NumPy Developers. (2022, 18 diciembre). *numpy.quantile — NumPY v1.25 Manual*. NumPy. Recuperado 1 de agosto de 2023, de https://numpy.org/doc/stable/reference/generated/numpy.quantile.html