Informe empresa A&A Ltda

Inmuebles disponibles para la venta

Cristian Andres Figueroa Castro

En el siguiente informe realizaremos un estudio y análisis de precios de vivienda y locales para la venda en diferentes regiones en Colombia, este informe será de información para la empresa A&A Ltda para analizar la posible realización de diferentes proyectos.

La información se obtuvo de el enlace: https://www.datos.gov.co/Hacienda-y-Cr-dito-P-blico/Inmuebles-Disponibles-Para-La-Venta/72gd-px77/data. Del cual descargamos el archivo CSV para analizar en nuestra herramienta Jupyter lab.

Al empezar el análisis tener que leer nuestro archivo CSV llamado Inmuebles_Disponibles_Para_La_Venta.csv con el siguiente código.

```
df = pd.read_csv('Inmuebles_Disponibles_Para_La_Venta.csv')
```

Para poder usar este código debemos como primer paso importar las librerías:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Las cuales nos ayudar a ejecutar diferentes gráficos y funciones matemáticas para realizar el análisis. Continuando con el análisis debemos visualizar que contiene el archivo leído **df** que hacer referencia a el data frame de los datos obtenidos de el archivo CVS por lo que debemos ejecutar el siguiente comando.

```
df.info()
```

Este comando nos entrega la siguiente información:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 448 entries, 0 to 447
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Codigo                 448 non-null    int64 
 1   Ciudad                 448 non-null    object
 2   Departamento           448 non-null    object
 3   Barrio                 59 non-null     object
 4   Direccion              448 non-null    object
 5   Area Terreno           448 non-null    int64 
 6   Area Construida        448 non-null    int64 
 7   Detalle Disponibilidad 448 non-null    object
 8   Estrato                448 non-null    object
 9   Precio                 448 non-null    int64 
 10  Tipo de Inmueble       448 non-null    object
 11  Datos Adicionales      77 non-null     object
dtypes: int64(4), object(8)
memory usage: 42.1+ KB
```

De la imagen anterior podemos observar que el archivo CSV cuanta con 448 filas y con 12 columnas de las cuales las columnas Barrio y Datos Adicionales tienes valores nulos, además en la derecha de la imagen vemos Dtype, esa columna nos dice el tipo de datos que tiene cada columna de la tabla por lo que vemos datos de tipo numero ‘int64’ y datos tipo ‘object’.

Como dijimos anterior mente algunas columnas cuentan con valores nulos por lo que ejecutamos el siguiente comando.

```
df.isnull().sum()
```

Dando lo siguiente:

```
Codigo                    0
Ciudad                    0
Departamento              0
Barrio                  389
Direccion                 0
Area Terreno              0
Area Construida           0
Detalle Disponibilidad    0
Estrato                   0
Precio                    0
Tipo de Inmueble          0
Datos Adicionales       371
dtype: int64
```

Ese comando nos hace el conteo de cuantos campos hay con valores nulos, es decir, celdas o espacios en las columnas donde no hay información. Al analizar que tipo de columna son, vemos que no son columnas de mayor importancia porque no todas las direcciones son necesarias con un barrio, o no todas las viviendas o locales cuantas con datos adicionales por lo que no es necesario eliminar los datos Nulos. Despues de revisar los datos nulos hay que revisar los datos duplicados por lo que también debemos analizar cada columna para hacer una correcta eliminación de datos. Observamos que al momento de mirar todas las columnas podemos tener datos duplicados en algunas columnas, pero hay una columna en específico que no puede duplicarse y es la columna código, ya que para cada propiedad debe tener su propio código por lo que eliminamos los duplicados de esa columna.

```
df = df.drop_duplicates(subset=['Codigo'])
```

Con este comando realizamos la eliminación de códigos duplicados en la columna Codigo y visualizamos de nuevo la nueva información con df.info().

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 437 entries, 0 to 447
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Codigo                  437 non-null    int64
 1   Ciudad                  437 non-null    object
 2   Departamento            437 non-null    object
 3   Barrio                  59 non-null     object
 4   Direccion               437 non-null    object
 5   Area Terreno            437 non-null    int64
 6   Area Construida         437 non-null    int64
 7   Detalle Disponibilidad  437 non-null    object
 8   Estrato                 437 non-null    object
 9   Precio                  437 non-null    int64
 10  Tipo de Inmueble        437 non-null    object
 11  Datos Adicionales       69 non-null     object
dtypes: int64(4), object(8)
memory usage: 44.4+ KB
```

Observamos que pasamos de 448 filas a 437 por lo que se eliminaron 11 datos duplicados que no generar mayores inconvenientes al momento de analizar el documento.

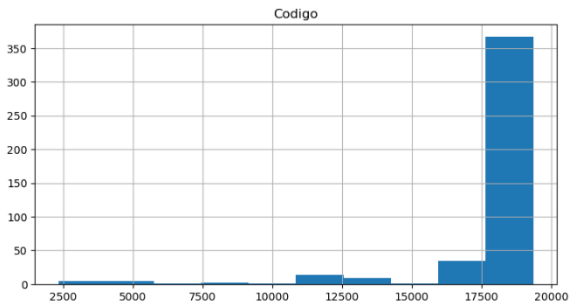
En este momento realizamos un análisis para descubrir que tipo de datos puede ser de importancia y nos pueden aportan gran información, por lo que generamos las siguientes preguntas:

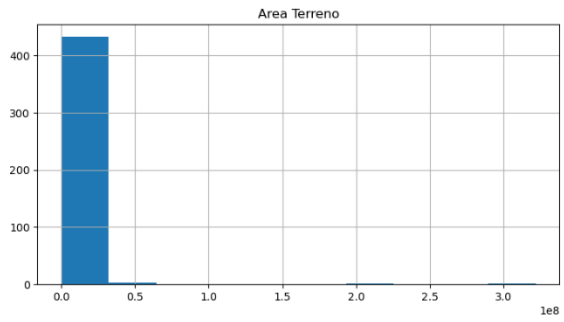
1. ¿En qué ciudades o departamento hay mayor cantidad de inmuebles en venta?
2. ¿Qué tipo de inmuebles son los que están a la venta?
3. ¿Qué rangos de precios maneja cada tipo estrato?
4. Identificar ¿Qué tipo de detalles de disponibilidad son los más disponibles?

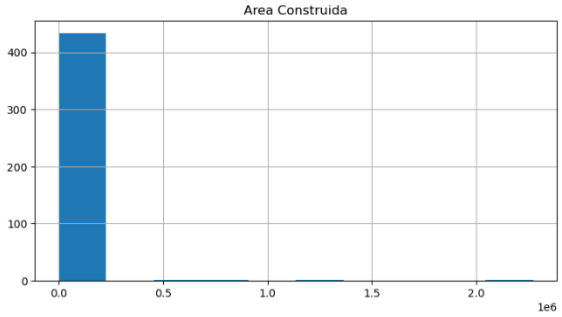
Después de identificar las preguntas continuamos a resolverlas y generar gráficos que nos ayuden a entender la información, como primer paso realizamos histogramas a las columnas con valores de numero entero y además generamos su media y desviación estándar con .describe() .

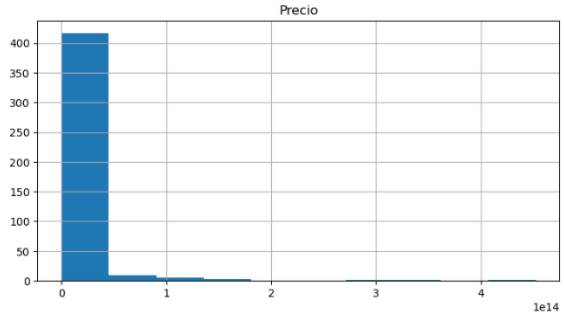
```
df.hist(bins=10,figsize=(20,10))
df.describe()
```

que nos generan los siguientes gráficos.









	Codigo	Area Terreno	Area Construida	Precio
count	437.000000	4.370000e+02	4.370000e+02	4.370000e+02
mean	17641.157895	1.660313e+06	1.184230e+04	9.166279e+12
std	2562.716889	1.881980e+07	1.301156e+05	3.792268e+13
min	2330.000000	0.000000e+00	0.000000e+00	0.000000e+00
25%	18123.000000	0.000000e+00	0.000000e+00	1.257250e+11
50%	18286.000000	0.000000e+00	0.000000e+00	1.652050e+11
75%	18461.000000	0.000000e+00	0.000000e+00	1.510604e+12
max	19353.000000	3.217197e+08	2.272400e+06	4.523379e+14

Como podemos observar en el histograma y en la tabla solo tenemos las columnas código, área Terreno, Area Construida y precio ya que son las únicas columnas que se puede representar con número y hacer los cálculos necesario.
Al revisar los resultados se puede interpretar que podemos omitir la columna Codigo ya que son números dados al momento del registro, además vemos que en la columna precio tenemos valores de 0, eso quiere decir que son datos que no están correctos, por lo que se decide eliminarlos.

```
df = df.loc[df['Precio'] != 0]
```

Con df.loc hacemos un filtrido de valores que son diferentes a 0 por lo que los iguales a 0 sin eliminados.

	Codigo	Area Terreno	Area Construida	Precio
count	436.000000	4.360000e+02	4.360000e+02	4.360000e+02
mean	17638.383028	1.664121e+06	1.186946e+04	9.187303e+12
std	2565.003511	1.884125e+07	1.302638e+05	3.796369e+13
min	2330.000000	0.000000e+00	0.000000e+00	6.599040e+10
25%	18122.500000	0.000000e+00	0.000000e+00	1.257250e+11
50%	18285.500000	0.000000e+00	0.000000e+00	1.652050e+11
75%	18460.250000	0.000000e+00	0.000000e+00	1.530501e+12
max	19353.000000	3.217197e+08	2.272400e+06	4.523379e+14

Volvemos a revisar los datos y ya se refleja el cambio realizado, con el tipo de números se puede observar que hay una gran variedad de datos ya que la desviación estándar es elevada, los datos que podemos considerar importantes son la media, valores máximos y mínimos.

Para continuar realizamos el calculo de las ciudades.

```
ciudades=df['Ciudad'].groupby(df['Ciudad']).count()
print(ciudades)
```

El anterior comando nos genera con conteo de ciudades, este comando es necesario para poder realizar graficas tipo pie, aun así, se puede visualizar la información.

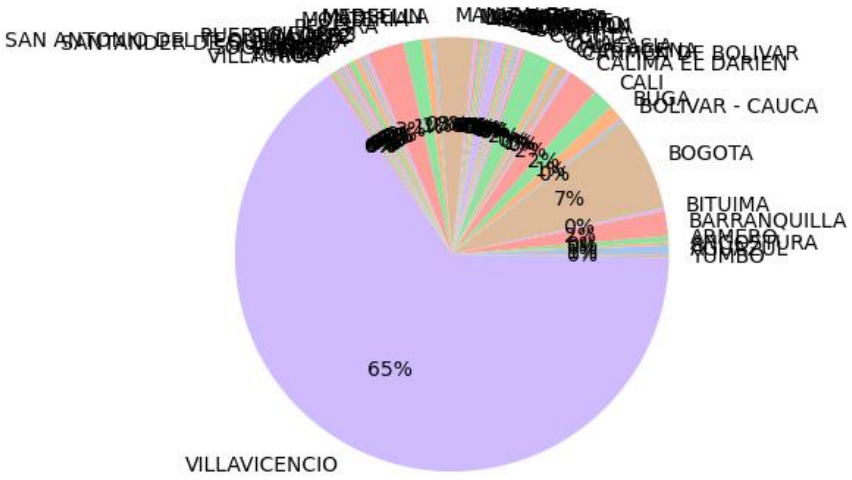
```
Ciudad
AGUAZUL                  3
ANGOSTURA                1
ARMERO                   2
BARRANQUILLA             8
BITUIMA                  1
BOGOTA                  31
BOLIVAR - CAUCA          1
BUGA                     5
CALI                     7
CALIMA EL DARIEN        10
CARMEN DE BOLIVAR        1
CARTAGENA                3
CAUCASIA                 1
CHIA                     2
CUCUTA                   9
CURITI                   1
DAGUA                    1
EL AGUILA                1
EL PLAYON                1
EL ROSAL                 1
ENVIGADO                 1
FUNZA                    1
GIRARDOT                 4
IBAGUE                   1
LA CALERA                1
LA DORADA                1
```

```
LA VIRGINIA                      1
LOS PATIOS                       1
MADRID                           1
MANIZALES                       13
MARSELLA                         1
MEDELLIN                         3
MONTERIA                         6
PEREIRA                         12
PIEDRAS                          1
PUERTO LOPEZ                     1
RICAURTE                         1
SAN ANTONIO DEL TEQUENDAMA       2
SANTANDER DE QUILICHAO           2
SOATA                            1
SOGAMOSO                         1
TARAZA                           1
TENJO                            1
TIBU                             1
TURBO                            1
VILLA RICA                       1
VILLAVICENCIO                  285
YUMBO                            1
```

Nos genera la anterior tabla donde no se puede visualizar de manera fácil la información por lo que procedemos a graficarla.

```
array_ciudades=(ciudades).index
colors = sns.color_palette('pastel')[0:6]
plt.pie(ciudades, labels = array_ciudades, colors = colors,
autopct='%.0f%%')
plt.show()
```

Es necesario incluir el comando ciudades.index ya que por la gran cantidad de ciudades es complicado escribir cada titulo para el grafico pie.

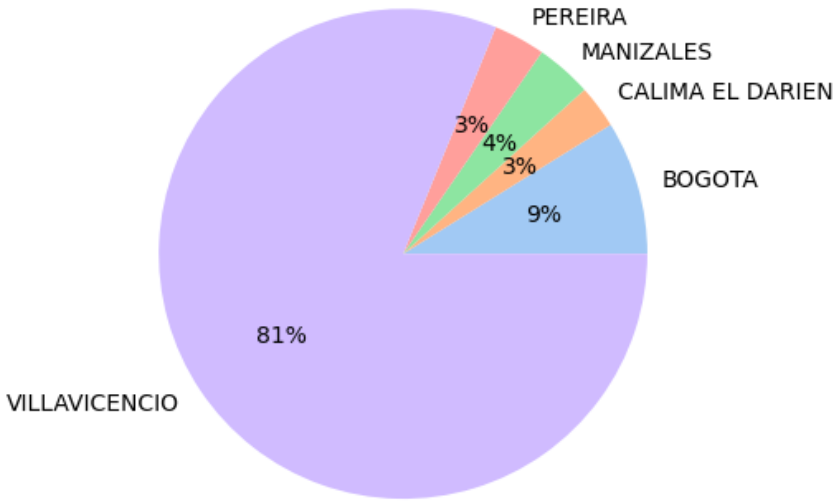


Al tener el grafico podemos observar que por la cantidad de ciudades registradas es difícil leer el grafico, aun asi se ve que hay gran concentración de datos en Villavicencio con un

65% y Bogotá con un 7%. Para realizar una corrección en el grafico obtenido se decide realizar un filtrado de ciudades que cuenten con mas de 9 inmuebles en venta, por lo que realizamos la siguiente lógica.

```
acu=0
array_ciudades2=[]
valores=[]
for i in ciudades:
        if i>9:
                array_ciudades2.append(ciudades.index[acu])
                valores.append(i)
acu=acu+1
```

Este código nos permite agrupar los nombres de las ciudades que cuenten con un conteo mayor a 9, las ciudades que cumplan se agrupan en la variable array_ciudades2 y los valores se acumulan en la variable Valores. Con esta modificación quedaría el siguiente gráfico.

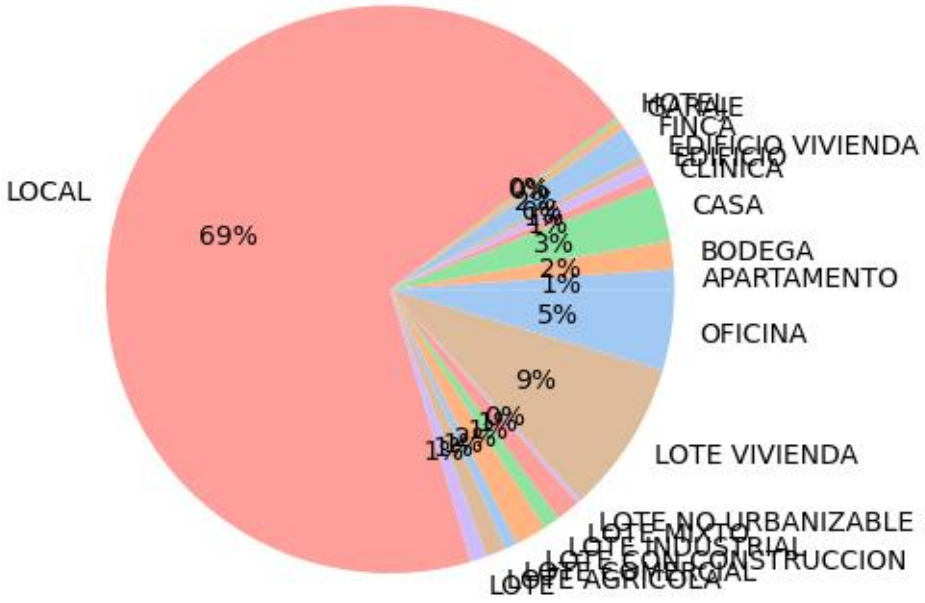


Podemos ver una información más clara y además las ciudades que cuentan con mas de 9 inmuebles en venta. Es importante resaltar la presencia de Villavicencio en la venta de inmuebles, esto se puede ver como una oportunidad para que la empresa se enfoque en esta ciudad.

Otro punto importante que debemos abordar es los tipos de inmuebles que hay a la venta por que realizamos un conteo y graficamos.

```
tipo_inmueble=df['Tipo de Inmueble'].groupby(df['Tipo de
Inmueble']).count()
array_tipo=tipo_inmueble.index
plt.pie(tipo_inmueble, labels = array_tipo, colors = colors,
autopct='%.0f%%')
plt.show()
```

Este código nos permite llevar el conteo de las filas que manejen algún tipo de inmueble y asi graficar en una grafica tipo pie que es la siguiente.



Vemos que hay una gran cantidad de inmuebles de tipo local con un 69% de todos los datos, le sigue lote vivienda y oficinas, con el 9% y 5% respectivamente.

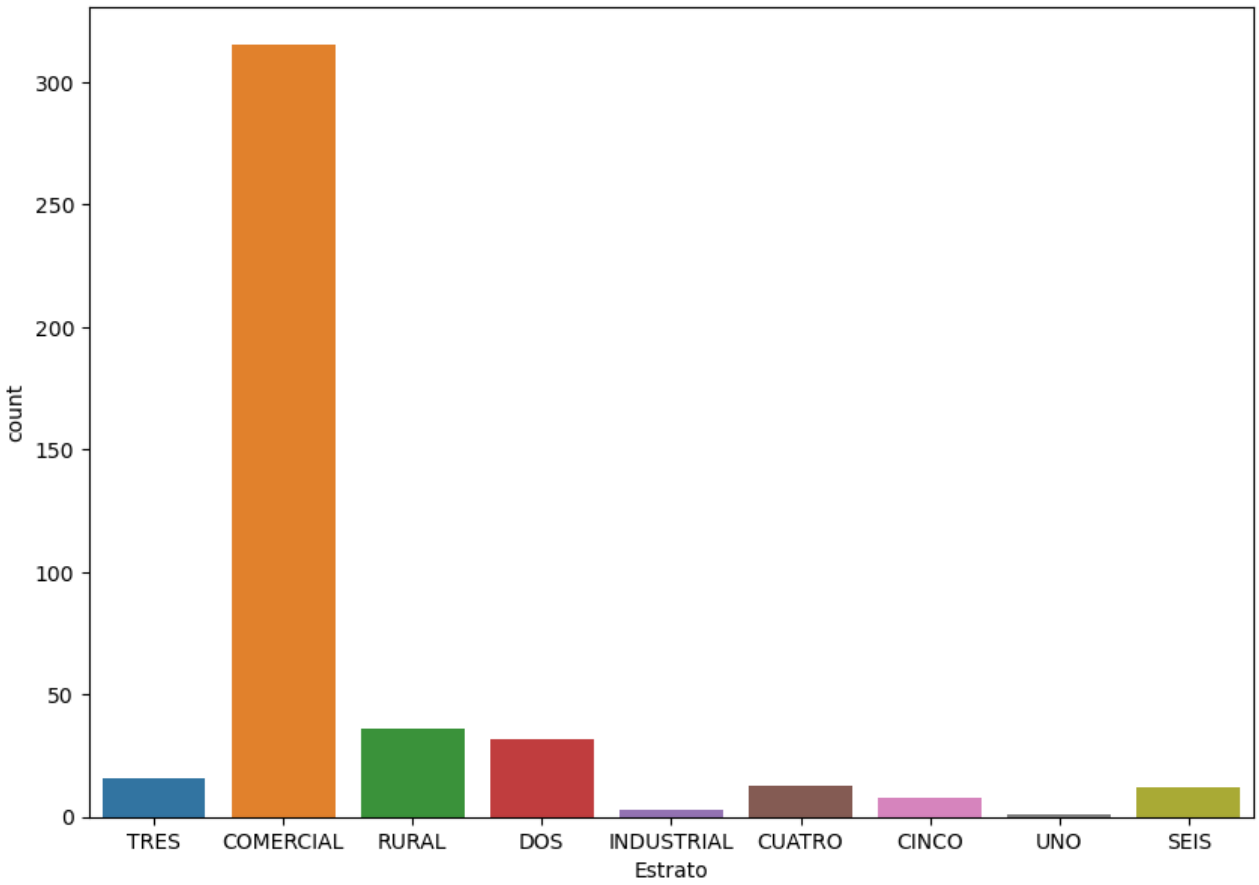
Continuando con el estudio, estudiamos la columna estratos, ya que queremos analizar cómo se comporta cada estrato dependiendo de los precios, y como primer paso realizamos lo siguiente:

```
estratos=df["Estrato"].groupby(df['Estrato']).count()
print(estratos)
```

Que nos imprime los siguiente.

```
Estrato
CINCO            8
COMERCIAL      315
CUATRO          13
DOS             32
INDUSTRIAL       3
RURAL           36
SEIS            12
TRES            16
UNO              1
```

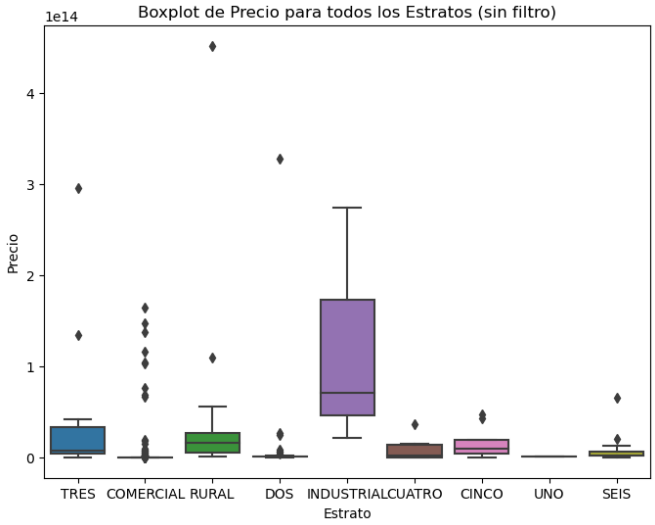
Para entender mejor esta tabla realizamos un gráfico de barras.



Es evidente la presencia en su mayoría de estratos tipo comercial, y uniéndolo con la información anterior que en su gran mayoría el tipo de el inmueble era Local, información de gran importancia para la toma de decisiones de la empresa.

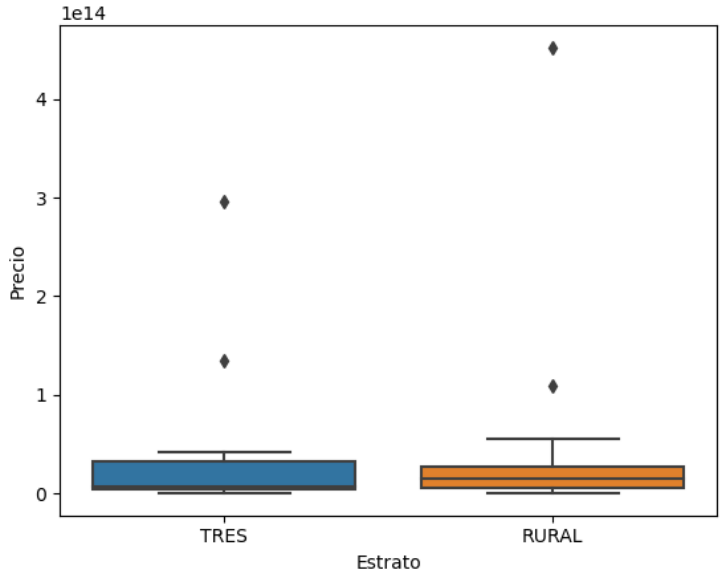
Posterior analizaremos cada estrato con respecto al precio y tenemos el siguiente gráfico.

```
calculoprecio = sns.boxplot(x=df["DetalleDisponibilidad"],y=df["Precio"])
```



Se puede visualizar una gran dispersión en los datos por lo que se dividirán en grupos.

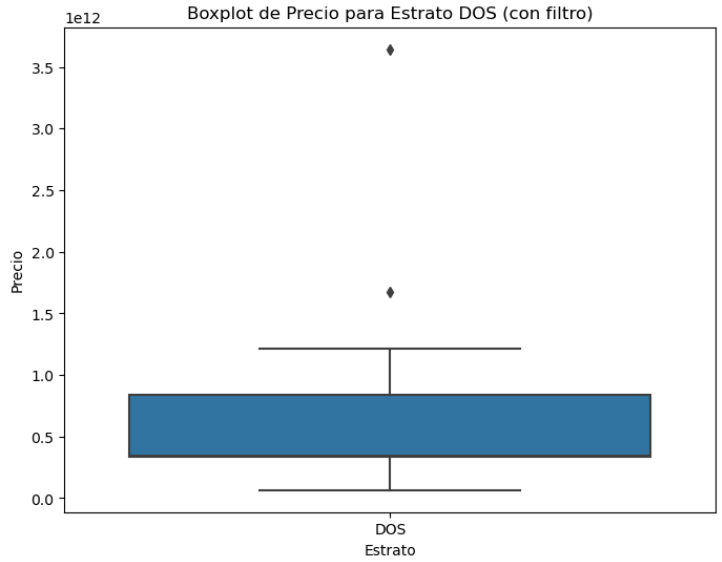
```
valores_filtrados = df[df['Estrato'].isin(['TRES','RURAL'])]
calculoprecio2 = sns.boxplot(x=valores_filtrados["Estrato"],
y=df["Precio"])
```



Ambos tienen un grupo similar con pequeñas dispersiones, pero se puede observar sus datos correctamente.

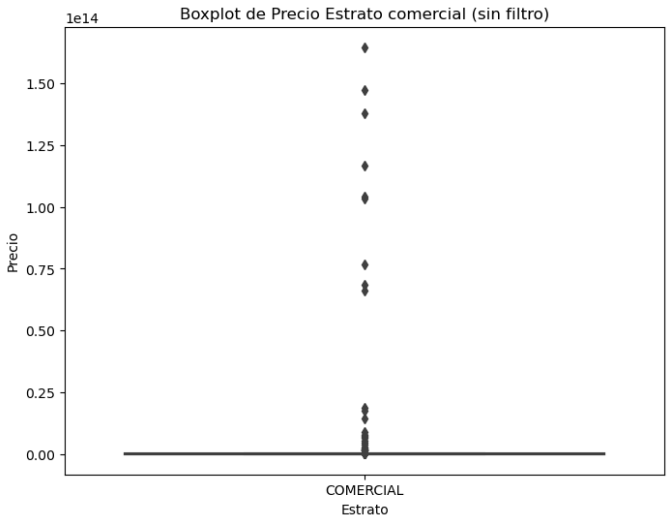
Ahora analizaremos el estrato DOS y como vimos en la grafica genera tiene una dispersión por que decidimos filtrarla.

```
valores_filtradosN = df[(df['Estrato'] == 'DOS')]
plt.figure(figsize=(8, 6))
sns.boxplot(x=valores_filtradosN["Estrato"], y=df[df['Precio'] <=
5000000000000]["Precio"])
plt.title('Boxplot de Precio para Estrato DOS (con filtro)')
plt.xlabel('Estrato')
plt.ylabel('Precio')
plt.show()
```



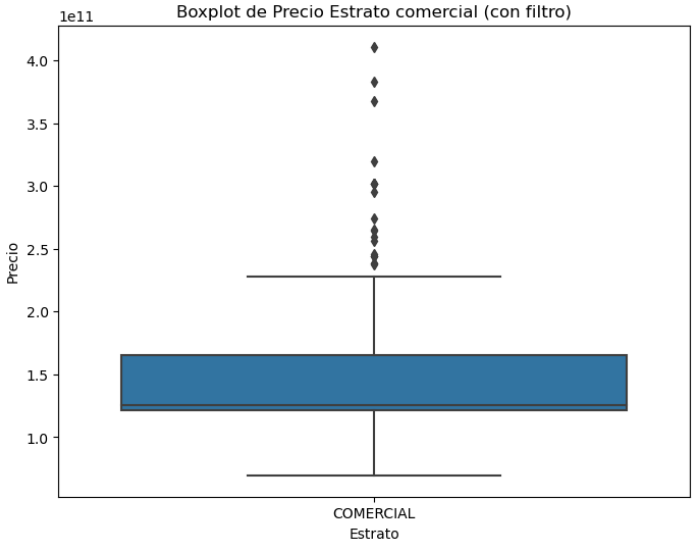
Es importante también visualizar el sector donde hay mas presencia de datos entonces tomamos el siguiente código.

```
plt.figure(figsize=(8, 6))
sns.boxplot(x=df[(df['Estrato'] == 'COMERCIAL')]["Estrato"],
y=df['Precio'])
plt.title('Boxplot de Precio Estrato comercial (sin filtro)')
plt.xlabel('Estrato')
plt.ylabel('Precio')
plt.show()
```



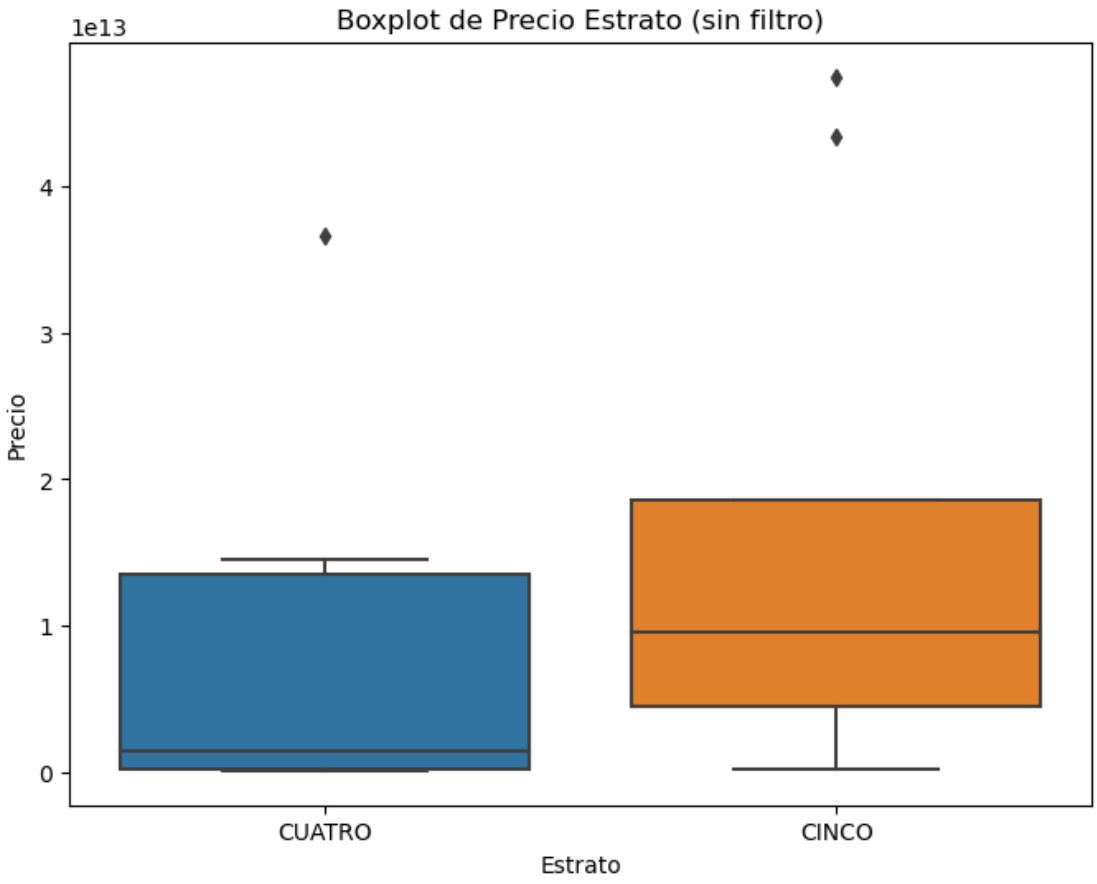
vemos que aquí también tenemos una dispersión de datos elevada, por lo que decidimos realizar un filtro, y visualizar mejor los datos ente 0 y 0.25.

```
sns.boxplot(x=df[(df['Estrato'] == 'COMERCIAL')]["Estrato"],
y=df[df['Precio']>= 500000000000]['Precio'])
plt.title('Boxplot de Precio Estrato comercial (con filtro)')
plt.xlabel('Estrato')
plt.ylabel('Precio')
plt.show()
```



aquí son más visualizables sus valores, aunque no hay que olvidar los de la primera grafica. Por ultimo visualizaremos los estratos CUATRO Y CINCO ya que se encuentran con muchas similitudes y se pueden visualizar correctamente.

```
plt.figure(figsize=(8, 6))
valores_filtrados = df[df['Estrato'].isin(['CUATRO','CINCO'])]
sns.boxplot(x=valores_filtrados["Estrato"], y=df["Precio"])
plt.title('Boxplot de Precio Estrato (sin filtro)')
plt.xlabel('Estrato')
plt.ylabel('Precio')
plt.show()
```



CONCLUCION:

- Hay una gran cantidad de datos en una ciudad en particular, así como establecimiento de tipo local con disponibilidad comercial, pero es necesaria una discusión ya que esto puede ser una gran oportunidad o por el contrario una oferta muy grande en este tipo de inmuebles y no sería una buena compra.