```
import pandas as pd

#lectura de los datos
df = pd.read_csv('games.csv')
```

```
# 2. dimenciones del los datos Filas y columnas
print(df.shape)
```

```
(60952, 9)
```

```
#muestreo de los datos (primeros 5)
df.head()
```

	name	release_date	price	positive	negative	app_id	min_owners	max_owners	hltb_single
0	Train Bandit	Oct 12, 2017	0.99	53	5	655370	0	20000	NaN
1	Henosis™	Jul 23, 2020	5.99	3	0	1355720	0	20000	NaN
2	Two Weeks in Painland	Feb 3, 2020	0.00	50	8	1139950	0	20000	NaN
3	Wartune Reborn	Feb 26, 2021	0.00	87	49	1469160	50000	100000	NaN
4	TD Worlds	Jan 9, 2022	10.99	21	7	1659180	0	20000	NaN

Próximos pasos: Generar código con df | Ver gráficos recomendados | New interactive sheet

```
#  informacion sobre cada columna
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60952 entries, 0 to 60951
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          60952 non-null  object
 1   release_date  60952 non-null  object
 2   price         60952 non-null  float64
 3   positive      60952 non-null  int64
 4   negative      60952 non-null  int64
 5   app_id        60952 non-null  int64
 6   min_owners    60952 non-null  int64
 7   max_owners    60952 non-null  int64
 8   hltb_single   12972 non-null  float64
dtypes: float64(2), int64(5), object(2)
memory usage: 4.2+ MB
```

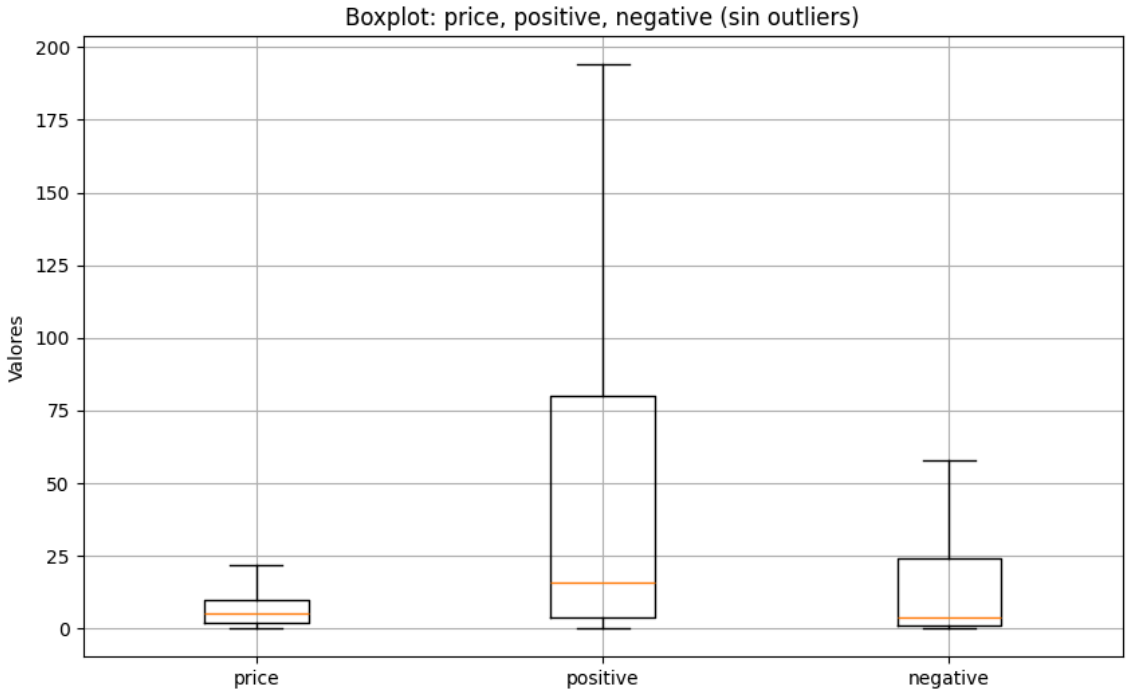
```
# 3. rangos de datos
print(df.describe())
```

```
              price      positive       negative        app_id    min_owners  \
count  60952.000000  6.095200e+04   60952.000000  6.095200e+04  6.095200e+04
mean       7.819159  1.045975e+03     193.455326  1.165637e+06  5.489319e+04
std        9.756732  1.498527e+04    4408.960253  5.986746e+05  6.753193e+05
min        0.000000  0.000000e+00       0.000000  5.700000e+02  0.000000e+00
25%        1.990000  4.000000e+00       1.000000  6.760850e+05  0.000000e+00
50%        4.990000  1.600000e+01       4.000000  1.095830e+06  0.000000e+00
75%        9.990000  8.000000e+01      24.000000  1.579002e+06  2.000000e+04
max      299.900000  1.477153e+06  895978.000000  2.690780e+06  1.000000e+08

         max_owners   hltb_single
count  6.095200e+04  12972.000000
mean   1.353160e+05      7.633595
std    1.451847e+06     11.943980
min    2.000000e+04      1.000000
25%    2.000000e+04      1.000000
50%    2.000000e+04      3.000000
75%    5.000000e+04      8.000000
max    2.000000e+08    100.000000
```

```
# 4.
"""
We can observe the following in each column:
The average price is $7.81, while the maximum price is $299.9.
Therefore, these values add noise to the measurements.

I used that as an example, but it applies to all columns since they all have
data that tends to add noise to the measurements.

"""
```

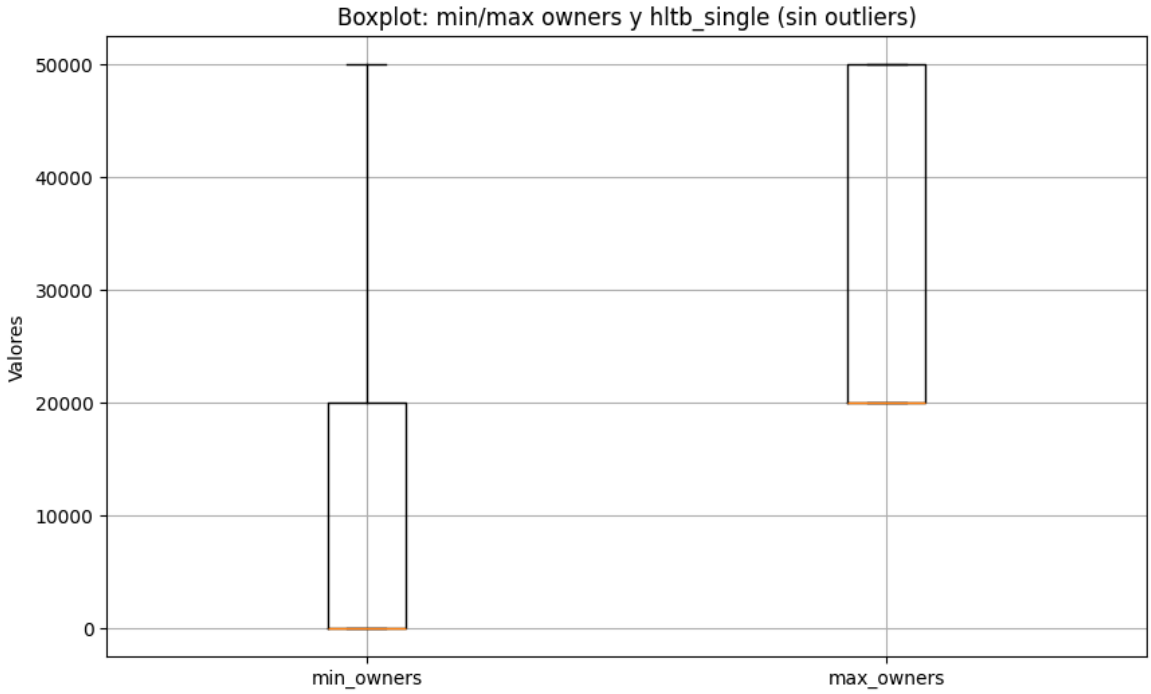
```
import matplotlib.pyplot as plt

cols_1 = ['price', 'positive', 'negative']

plt.figure(figsize=(10, 6))
plt.boxplot([df[col].dropna() for col in cols_1], labels=cols_1, showfliers=False)
plt.title("Boxplot: price, positive, negative (sin outliers)")
plt.ylabel("Valores")
plt.grid(True)
plt.show()
```

```
<ipython-input-10-d97ccf191029>:6: MatplotlibDeprecationWarning: The 'labels' parameter of boxplot() has been renamed 'tick_labels'
  plt.boxplot([df[col].dropna() for col in cols_1], labels=cols_1, showfliers=False)
```

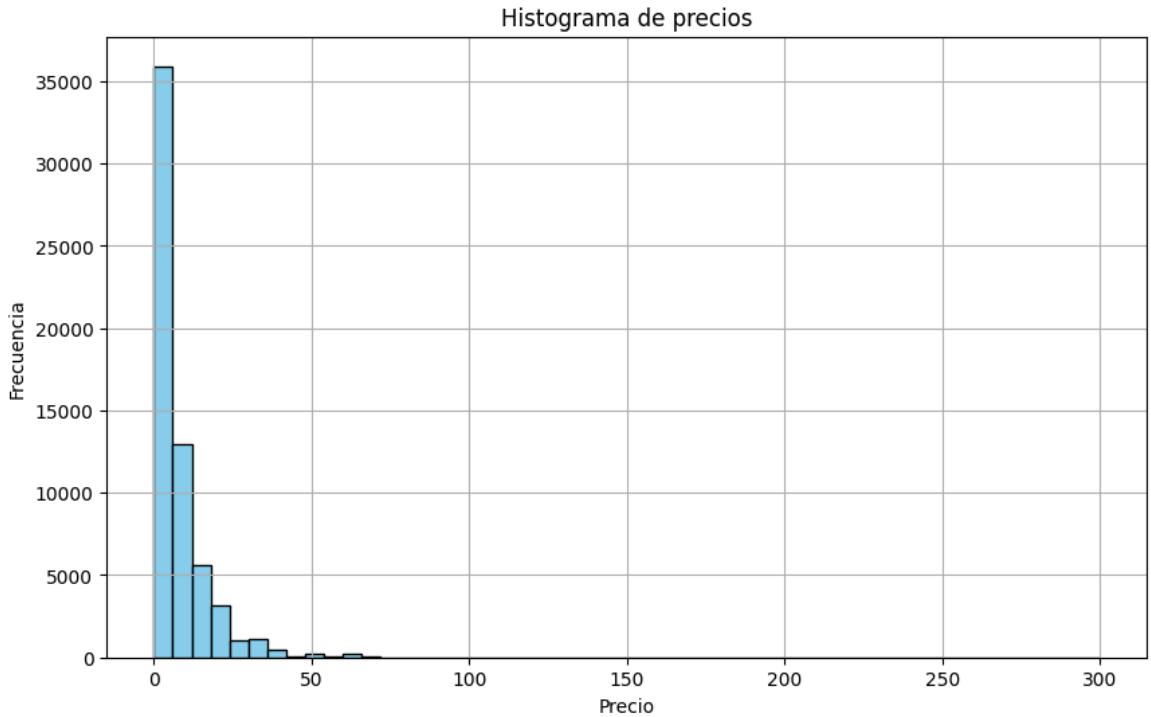



```
cols_2 = ['min_owners', 'max_owners']

plt.figure(figsize=(10, 6))
plt.boxplot([df[col].dropna() for col in cols_2], labels=cols_2, showfliers=False)
plt.title("Boxplot: min/max owners y hltb_single (sin outliers)")
plt.ylabel("Valores")
plt.grid(True)
plt.show()
```

```
<ipython-input-11-6cef3ac5579c>:4: MatplotlibDeprecationWarning: The 'labels' parameter of boxplot() has been renamed 'tick_labels'
  plt.boxplot([df[col].dropna() for col in cols_2], labels=cols_2, showfliers=False)
```

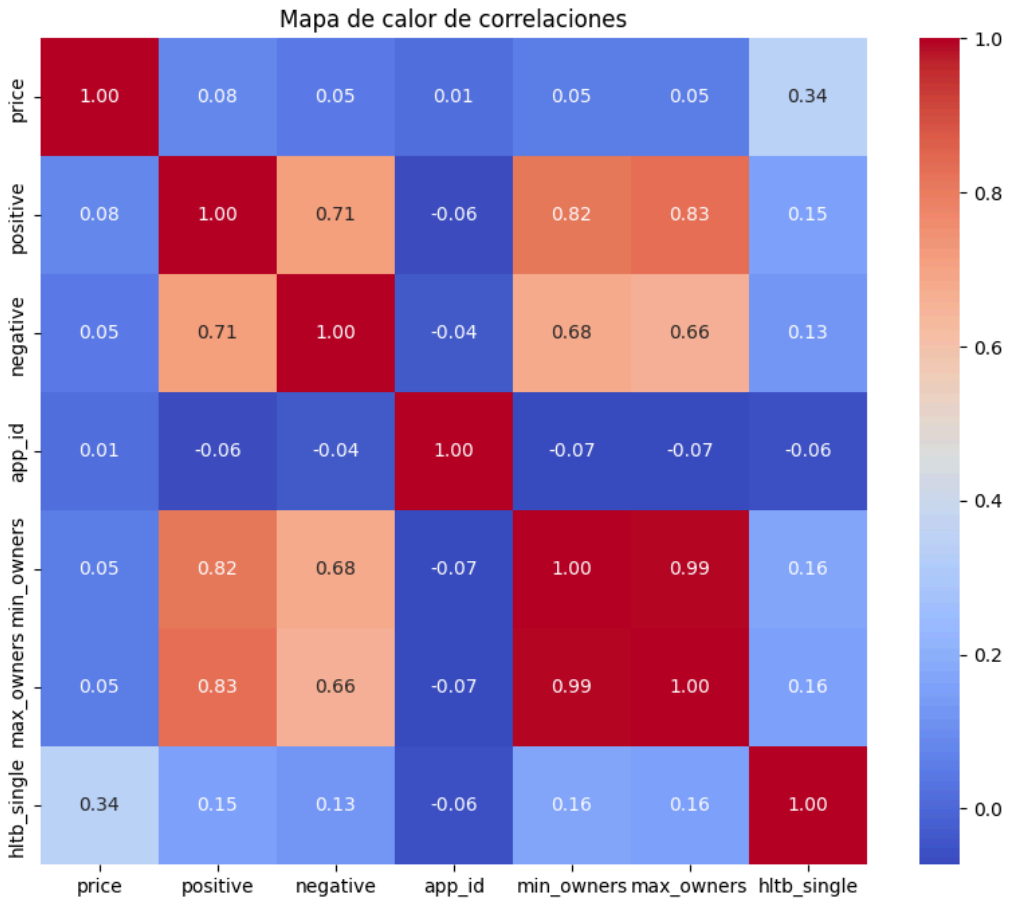



```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.hist(df['price'].dropna(), bins=50, color='skyblue', edgecolor='black')
plt.title("Histograma de precios")
plt.xlabel("Precio")
plt.ylabel("Frecuencia")
plt.grid(True)
plt.show()
```

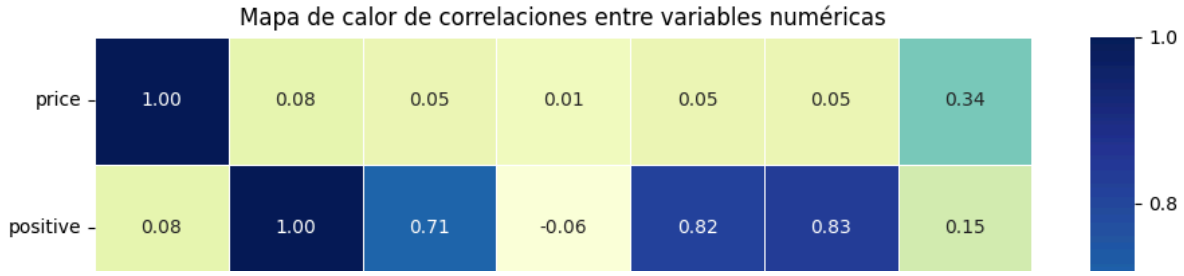



```
import seaborn as sns

# Calcular la matriz de correlación de las columnas numéricas
corr_matrix = df.corr(numeric_only=True)

# Crear el mapa de calor
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", square=True)
```

```
plt.title("Mapa de calor de correlaciones")
plt.show()
```




```
# Calcula la matriz de correlación solo con columnas numéricas
correlaciones = df.corr(numeric_only=True)

# Crear el heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlaciones, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=0.5)

plt.title("Mapa de calor de correlaciones entre variables numéricas")
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```


Mapa de calor de correlaciones entre variables numéricas
price
positive
1.00
0.08
0.05
0.01
0.05
0.05
0.34
0.08
1.00
0.71
-0.06
0.82
0.83
0.15
1.0
0.8