Trabajo práctico CaC - Parte 2

Grupo 097

- Rodolfo Albornoz

Comisión 24003

Importamos las librerias correspondientes y usamos la URL del google sheets con los archivos para utilizarlos en el collab

In [500]:

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

# Para redondear los flotantes a 2 decimales
pd.options.display.float_format = "{:.2f}".format

ID_planilla = "1IkytFzvzkWp9d5rSR6K8F-eWK4ddjasX3JKzRGzoGUw"

URL = f'https://docs.google.com/spreadsheets/d/{ID_planilla}/gviz/tq?tqx=out:csv&she
```

Ventas

Levantamos el dataframe de ventas

In [501]:

```
df_ventas = pd.read_csv(URL + "ventas_Paraguay", thousands = '.')
```

Exploración de los datos

Empezamos explorando los datos

In [502]:

```
df_ventas.shape
```

Out[502]:

```
(45, 13)
```

In [503]:

```
df_ventas.dtypes
```

Out[503]:

```
distributor    float64
Rubber          object
Brass           object
Vinyl           object
Granite         object
Stone           object
Brick           object
Aluminum        object
Glass           object
Plexiglass      object
Steel           object
Wood            object
Plastic         object
dtype: object
```

Observaciones:

- Tenemos que convertir los valores de la columna distributor de float a int (Que son valores enteros)
- Tenemos que convertir los object a float (Ya que representan precios con decimales)

Exploramos las primeras y últimas filas del dataframe

In [504]:
```
df_ventas.head()
```

Out[504]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	A
0	29.00	$54.510.203,61	$45.268.636,86	$51.579.748,25	$21.780.180,58	$26.576.776,52	$0,00	$55.8
1	161.00	NaN	$25.837.100,49	$36.603.264,50	$21.883.374,92	$1.473.437,08	$0,00	$46.2
2	175.00	$21.780.180,58	$78.927.599,01	$25.837.100,49	$33.102.840,61	$51.579.748,25	$0,00	$21.8
3	234.00	$79.358.855,35	$90.185.311,22	$45.268.636,86	$54.510.203,61	$59.358.855,35	$0,00	$79.3
4	241.00	$11.758.005,07	$21.780.180,58	$57.187.306,41	$9.945.371,16	$32.067.534,68	$0,00	$53.1

In [505]:
```
df_ventas.tail()
```

Out[505]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone
40	1666.00	$3.221.765,13	$10.536.882,23	$35.563.624,11	$46.039.695,30	$22.300.838,58
41	1679.00	$36.999.867,09	$6.662.942,92	$76.443.998,86	$45.268.636,86	$18.050.385,1
42	1710.00	$55.872.547,77	$18.421.797,13	$12.466.621,84	$59.862.809,01	$21.883.374,9
43	1723.00	$65.978.582,24	$36.603.264,50	$57.009.912,25	$16.931.446,04	$41.572.788,8
44	NaN	$1.680.453.941,34	$1.840.659.381,00	$1.912.589.120,61	$1.845.294.550,32	$1.651.519.395,6

Observaciones:

- Tenemos que sacar los $ y . de todas las columnas con valores de tipo precio, y cambiar los "," por "." (Para convertir las columnas a float)
- Tenemos que sacar la última fila de los datos (Que en la hoja de sheets representa el total)
- La columna de "Brick" tiene muchos valores de 0, y la de "Steel" también tiene varios valores en 0. Mas tarde vamos a investigar y apoyarnos con las demás hojas de cálculo para descubrir por qué podría ser esto.

Revisemos los valores null

In [506]:
```
nulos_por_filas_ventas = df_ventas[df_ventas.isnull().any(axis=1)]
nulos_por_filas_ventas
```

Out[506]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone
1	161.00	NaN	$25.837.100,49	$36.603.264,50	$21.883.374,92	$1.473.437,08
44	NaN	$1.680.453.941,34	$1.840.659.381,00	$1.912.589.120,61	$1.845.294.550,32	$1.651.519.395,6

In [507]:

```
nulos_por_columnas_ventas = df_ventas.isnull().any()
nulos_por_columnas_ventas
```

Out[507]:

```
distributor     True
Rubber          True
Brass          False
Vinyl          False
Granite        False
Stone          False
Brick          False
Aluminum       False
Glass          False
Plexiglass     False
Steel          False
Wood           False
Plastic        False
dtype: bool
```

La fila 44 es la última, la de totales, y dijimos que vamos a eliminarla. Respecto a la primera que tiene valor nulo en la columna "Rubber", no vamos a sacar la fila porque podria darnos información importante, pero podemos llenar su valor nulo con cualquier valor (Por ejemplo 0)

Por último, solo por curiosidad, veamos si hay valores duplicados. Si no los hay no nos preocupamos

In [508]:

```
df_ventas.duplicated().sum()
```

Out[508]:

```
0
```

Limpieza de datos

1. Eliminar símbolos de $ y . y reemplazar , por .

In [509]:

```
columnas_ventas = list(df_ventas.columns[1:13])
```

In [510]:

```
for col in columnas_ventas:
  df_ventas[col] = df_ventas[col].str.replace('$','')
  df_ventas[col] = df_ventas[col].str.replace('.', '')
  df_ventas[col] = df_ventas[col].str.replace(',', '.')

df_ventas.head()
```

Out[510]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminum	
0	29.00	54510203.61	45268636.86	51579748.25	21780180.58	26576776.52	0.00	55872547.77	18
1	161.00	NaN	25837100.49	36603264.50	21883374.92	1473437.08	0.00	46239695.30	33
2	175.00	21780180.58	78927599.01	25837100.49	33102840.61	51579748.25	0.00	21883374.92	78
3	234.00	79358855.35	90185311.22	45268636.86	54510203.61	59358855.35	0.00	79358855.35	32
4	241.00	11758005.07	21780180.58	57187306.41	9945371.16	32067534.68	0.00	53172624.14	57

2. Eliminar última fila

In [511]:

```
df_ventas.drop(df_ventas.index[-1], inplace = True)
```

3. Llenar valores nulos

In [512]:

```
df_ventas['Rubber'].fillna(0, inplace = True)
```

4. Convertir filas a otros tipos de datos

In [513]:

```
# Para la columna distributor solo casteo a int para eliminar decimales
df_ventas['distributor'] = df_ventas['distributor'].astype(int)

for col in columnas_ventas:
  df_ventas[col] = df_ventas[col].astype(float)

df_ventas.dtypes
```

Out[513]:

```
distributor      int64
Rubber         float64
Brass          float64
Vinyl          float64
Granite        float64
Stone          float64
Brick          float64
Aluminum       float64
Glass          float64
Plexiglass     float64
Steel          float64
Wood           float64
Plastic        float64
dtype: object
```

Dataframe de ventas final

In [514]:

```
df_ventas
```

Out[514]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Alumi
0	29	54510203.61	45268636.86	51579748.25	21780180.58	26576776.52	0.00	558725
1	161	0.00	25837100.49	36603264.50	21883374.92	1473437.08	0.00	462396
2	175	21780180.58	78927599.01	25837100.49	33102840.61	51579748.25	0.00	218833
3	234	79358855.35	90185311.22	45268636.86	54510203.61	59358855.35	0.00	793588
4	241	11758005.07	21780180.58	57187306.41	9945371.16	32067534.68	0.00	531726
5	308	90185311.22	43512922.94	11758005.07	76443998.86	36999867.09	23012.00	516734
6	325	78927599.01	57187306.41	41899590.44	18050385.12	20344007.84	0.00	355636
7	364	5662736.92	59862809.01	46239695.30	90185311.22	36603264.50	0.00	818129
8	378	57009912.25	53172624.14	36999867.09	59199680.31	78927599.01	12548.00	424113
9	379	32438788.20	54510203.61	1346769.34	18421797.13	10536882.23	0.00	217801
10	504	59199680.31	59199680.31	9945371.16	55872547.77	43512922.94	11452.00	265767
11	565	61236075.66	32067534.68	21883374.92	53172624.14	59199680.31	0.00	331028
12	583	26576776.52	22300838.58	1473437.08	41572788.80	65978582.24	0.00	418995
13	619	59862809.01	55020982.80	78927599.01	55020982.80	54510203.61	0.00	284217
14	707	33102840.61	21883374.92	43512922.94	177874267.01	32438788.20	0.00	180503
15	715	1346769.34	16086016.33	65978582.24	12466621.84	11758005.07	0.00	366032
16	723	45268636.86	79358855.35	59199680.31	26576776.52	33102840.61	2669.00	452686
17	808	20344007.84	1346769.34	22300838.58	35563624.11	27045333.20	0.00	764439
18	815	77874267.01	51579748.25	32067534.68	11758005.07	57009912.25	0.00	550209
19	818	42411396.23	57009912.25	90185311.22	57187306.41	77874267.01	0.00	545102
20	846	10536882.23	27179878.86	79358855.35	27009912.25	57187306.41	0.00	901853
21	860	46239695.30	1473437.08	41572788.80	61236075.66	5662736.92	0.00	599453
22	920	51579748.25	46239695.30	53172624.14	27179878.86	25837100.49	0.00	435129
23	1017	41572788.80	35563624.11	27179878.86	79358855.35	1346769.34	13455.00	271798
24	1055	16086016.33	11812951.32	18050385.12	51579748.25	21780180.58	0.00	213440
25	1062	35563624.11	42411396.23	81812951.32	65978582.24	90185311.22	0.00	415727
26	1093	22300838.58	61236075.66	61236075.66	36059867.09	61236075.66	0.00	117580
27	1104	21883374.92	18050385.12	59862809.01	25837100.49	9945371.16	34521.00	320675
28	1169	53172624.14	33102840.61	177874267.01	81812951.32	42411396.23	0.00	67063
29	1183	41899599.44	55872547.77	26576776.52	41899590.44	55072547.77	0.00	591996
30	1235	32067534.68	65978582.24	16086016.33	36603264.50	53172624.14	0.00	659785
31	1302	11812951.32	11758005.07	18421797.13	43512922.94	16086016.33	0.00	213467
32	1384	57187306.41	177874267.01	54510203.61	32067534.68	3221765.13	0.00	789275
33	1418	1473437.08	76443998.86	42411396.23	22300838.58	46239695.30	0.00	1571873
34	1463	55020982.80	9945371.16	20344007.84	41927599.01	41899599.44	0.00	369998
35	1489	18050385.12	41899590.44	33102840.61	20344007.84	45268636.86	0.00	1778742
36	1526	27179878.86	36999867.09	55872547.77	1346769.34	35563624.11	0.00	258371

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Alumi
37	1553	25837100.49	26576776.52	21780180.58	6662942.92	51862809.01	12543.00	3608601
38	1560	43512922.94	20344007.84	55020982.80	42411396.23	11812951.32	0.00	5986280
39	1599	9945371.16	41572788.80	6662942.92	1473437.08	55020982.80	0.00	6123607
40	1666	3221765.13	10536882.23	35563624.11	46039695.30	22300838.58	0.00	5157974
41	1679	36999867.09	6662942.92	76443998.86	45268636.86	18050385.12	0.00	2230083
42	1710	55872547.77	18421797.13	12466621.84	59862809.01	21883374.92	0.00	5700991
43	1723	65978582.24	36603264.50	57009912.25	16931446.04	41572788.80	42458.00	1246662

Importaciones

Levantamos el dataframe de importaciones

In [515]:
```
df_importaciones = pd.read_csv(URL + "importaciones_Paraguay",  thousands = '.')
```

Exploración de los datos

Empezamos explorando los datos

In [516]:
```
df_importaciones.shape
```

Out[516]:
```
(46, 16)
```

In [517]:
```
df_importaciones.dtypes
```

Out[517]:
```
distributor      int64
Rubber          object
Brass           object
Vinyl           object
Granite         object
Stone           object
Brick           object
Aluminum        object
Glass           object
Plexiglass      object
Steel           object
Wood            object
Plastic         object
Unnamed: 13    float64
Unnamed: 14     object
Columnas        object
dtype: object
```

Observaciones:

- Tenemos que convertir los object a float (Ya que representan precios con decimales)

Exploramos las primeras y últimas filas del dataframe

In [518]:
```
df_importaciones.head()
```

Out[518]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Bric
0	1017	$22.431.099,00	$36.031.577,00	$31.118.167,00	$21.322.223,00	$35.382.848,00	$35.280.292,
1	1055	$27.566.922,00	$21.996.538,00	$39.412.316,00	$25.681.987,00	$41.861.783,00	$22.408.742,
2	1062	$37.577.095,00	$41.457.655,00	$31.467.967,00	$37.577.926,00	$35.845.106,00	$42.953.168,
3	1093	$36.012.730,00	$41.667.692,00	$22.837.073,00	$29.288.200,00	$39.553.494,00	$33.513.588,
4	1104	$43.416.417,00	$36.290.780,00	$23.679.738,00	$21.183.706,00	$25.210.622,00	$30.864.041,

In [519]:

```python
df_importaciones.tail()
```

Out[519]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Br
41	815	$25.138.331,00	$38.634.366,00	$35.724.628,00	$21.355.595,00	$42.958.842,00	$27.048.824
42	818	$41.761.417,00	$33.709.306,00	$27.497.587,00	$28.311.310,00	$21.683.294,00	$26.276.785
43	846	$20.880.116,00	$37.111.825,00	$34.422.161,00	$40.759.561,00	$26.050.968,00	$21.080.855
44	860	$34.050.845,00	$30.151.961,00	$26.956.027,00	$45.003.866,00	$23.879.569,00	$27.287.921
45	920	$35.580.430,00	$39.978.595,00	$32.046.855,00	$30.147.636,00	$36.212.747,00	$33.365.715

Observaciones:

- Tenemos que sacar los $ y . de todos los precios y reemplazar los "," por "." (Para convertir las columnas a float)
- Tenemos que sacar las últimas 3 columnas que tienen todos valores nulos
- Una vez convertidas las columnas de tipo object a float, se deben convertir a entero, ya que los precios son flotantes con parte decimal 0

Revisemos los valores null

In [520]:

```python
nulos_por_filas_importaciones = df_importaciones[df_importaciones.isnull().any(axis=
nulos_por_filas_importaciones
```

Out[520]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	
0	1017	$22.431.099,00	$36.031.577,00	$31.118.167,00	$21.322.223,00	$35.382.848,00	$35.280.2
1	1055	$27.566.922,00	$21.996.538,00	$39.412.316,00	$25.681.987,00	$41.861.783,00	$22.408.7
2	1062	$37.577.095,00	$41.457.655,00	$31.467.967,00	$37.577.926,00	$35.845.106,00	$42.953.1
3	1093	$36.012.730,00	$41.667.692,00	$22.837.073,00	$29.288.200,00	$39.553.494,00	$33.513.5
4	1104	$43.416.417,00	$36.290.780,00	$23.679.738,00	$21.183.706,00	$25.210.622,00	$30.864.0
5	1169	$43.909.858,00	$27.128.183,00	$36.391.549,00	$21.542.467,00	$34.971.118,00	$24.495.0
6	1169	$28.550.182,00	$36.499.732,00	$41.512.708,00	$22.262.065,00	$30.270.432,00	$34.506.2
7	1183	$21.771.886,00	$43.109.659,00	$23.247.953,00	$22.866.072,00	$31.459.461,00	$30.652.6
8	1235	$36.667.511,00	$25.366.105,00	$24.902.331,00	$36.605.735,00	$44.673.772,00	$31.541.3
9	1302	$39.958.024,00	$40.169.347,00	$35.399.555,00	$25.461.219,00	$37.801.759,00	$30.154.2
10	1384	$23.722.590,00	$24.623.967,00	$41.133.545,00	$35.578.933,00	$45.015.411,00	$139.362.6
11	1418	$25.138.331,00	$38.634.366,00	$35.724.628,00	$21.355.595,00	$42.958.842,00	$27.048.8
12	1463	$37.817.347,00	$23.272.550,00	$36.228.786,00	$40.908.298,00	$44.287.587,00	$20.840.1
13	1489	$38.180.046,00	$28.960.197,00	$38.400.970,00	$23.646.886,00	$39.383.770,00	$122.710.0
14	1526	$21.870.115,00	$29.728.962,00	$29.018.088,00	$21.585.132,00	$44.760.537,00	$37.955.1
15	1553	$28.415.088,00	$38.304.440,00	$32.154.709,00	$23.047.802,00	$23.121.310,00	$39.288.0
16	1560	$23.507.965,00	$41.845.467,00	$42.055.834,00	$35.453.957,00	$27.278.926,00	$28.907.4
17	1599	$33.377.107,00	$29.404.830,00	$37.302.875,00	$38.573.860,00	$27.524.149,00	$29.226.1
18	1666	$40.875.271,00	$22.374.529,00	$20.935.764,00	$31.145.028,00	$25.311.142,00	$42.786.0
19	1679	$38.180.046,00	$28.960.197,00	$38.400.970,00	$23.646.886,00	$39.383.770,00	$122.710.0
20	1710	$30.129.379,00	$36.728.723,00	$32.550.897,00	$20.871.609,00	$44.756.568,00	$1.643.078.8
21	1723	$31.142.709,00	$28.352.801,00	$28.274.723,00	$43.010.066,00	$44.667.972,00	$40.622.1
22	161	$22.957.310,00	$24.387.893,00	$41.579.620,00	$26.804.859,00	$37.698.018,00	$25.843.7
23	175	$28.135.181,00	$23.695.417,00	$32.743.214,00	$23.511.170,00	$29.707.140,00	$38.298.7
24	234	$34.544.298,00	$27.882.865,00	$22.664.200,00	$21.057.646,00	$22.595.798,00	$39.745.9
25	241	$28.318.181,00	$23.132.460,00	$29.998.057,00	$42.240.878,00	$25.721.016,00	$20.894.7
26	29	$37.794.819,00	$26.812.043,00	$25.270.289,00	$28.139.664,00	$26.173.115,00	$43.607.6
27	308	$35.622.161,00	$22.845.538,00	$31.984.270,00	$32.821.654,00	$31.742.905,00	$36.952.9
28	325	$42.018.216,00	$21.936.967,00	$36.106.133,00	$29.906.692,00	$25.729.289,00	$36.214.5
29	325	$34.349.322,00	$20.737.841,00	$26.603.366,00	$32.141.549,00	$31.818.580,00	$124.353.9
30	364	$44.229.593,00	$42.500.506,00	$33.478.392,00	$37.522.478,00	$32.650.463,00	$29.599.6
31	378	$41.444.583,00	$23.735.516,00	$29.708.711,00	$45.198.396,00	$26.911.979,00	$27.704.9
32	379	$41.387.139,00	$32.973.505,00	$38.520.126,00	$37.505.722,00	$32.781.557,00	$159.624.5
33	504	$35.858.240,00	$22.072.249,00	$21.646.802,00	$39.650.737,00	$40.985.840,00	$25.023.1
34	565	$37.489.497,00	$22.875.902,00	$34.374.638,00	$23.295.485,00	$33.803.575,00	$925.835.5
35	583	$20.607.246,00	$23.005.062,00	$27.506.653,00	$26.062.955,00	$27.696.836,00	$37.419.9
36	619	$32.853.600,00	$43.936.493,00	$31.944.768,00	$23.457.213,00	$30.668.319,00	$940.975.5

	distributor	Rubber	Brass	Vinyl	Granite	Stone	
37	707	$29.585.885,00	$42.849.169,00	$30.521.280,00	$21.745.658,00	$32.580.700,00	$45.204.9
38	715	$43.576.523,00	$30.181.558,00	$32.773.382,00	$36.669.396,00	$45.128.426,00	$34.340.3
39	723	$42.160.913,00	$41.409.478,00	$37.110.518,00	$25.210.663,00	$28.016.394,00	$43.238.0
40	808	$27.799.295,00	$44.568.676,00	$26.843.228,00	$41.036.010,00	$38.237.752,00	$29.463.7
41	815	$25.138.331,00	$38.634.366,00	$35.724.628,00	$21.355.595,00	$42.958.842,00	$27.048.8
42	818	$41.761.417,00	$33.709.306,00	$27.497.587,00	$28.311.310,00	$21.683.294,00	$26.276.7
43	846	$20.880.116,00	$37.111.825,00	$34.422.161,00	$40.759.561,00	$26.050.968,00	$21.080.8
44	860	$34.050.845,00	$30.151.961,00	$26.956.027,00	$45.003.866,00	$23.879.569,00	$27.287.9
45	920	$35.580.430,00	$39.978.595,00	$32.046.855,00	$30.147.636,00	$36.212.747,00	$33.365.7

In [521]:
```
nulos_por_columnas_importaciones = df_importaciones.isnull().any()
nulos_por_columnas_importaciones
```

Out[521]:
```
distributor    False
Rubber         False
Brass          False
Vinyl          False
Granite        False
Stone          False
Brick          False
Aluminum       False
Glass          False
Plexiglass     False
Steel          False
Wood           False
Plastic        False
Unnamed: 13     True
Unnamed: 14     True
Columnas        True
dtype: bool
```

Confirmamos que todos los valores nulos están en las últimas 3 columnas

Por último, solo por curiosidad, veamos si hay valores duplicados. Si no los hay no nos preocupamos

In [522]:
```
df_importaciones.duplicated().sum()
```

Out[522]:
```
0
```

Limpieza de datos

1. Eliminar símbolos de $ y . y reemplazar , por .

In [523]:
```
columnas_importaciones = list(df_importaciones.columns[1:13])
```

In [524]:
```
for col in columnas_importaciones:
  df_importaciones[col] = df_importaciones[col].astype(str).str.replace('$','')
  df_importaciones[col] = df_importaciones[col].astype(str).str.replace('.', '')
  df_importaciones[col] = df_importaciones[col].astype(str).str.replace(',', '.')
```

```
df_importaciones.head()
```

Out[524]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminu
0	1017	22431099.00	36031577.00	31118167.00	21322223.00	35382848.00	35280292.00	32362235.
1	1055	27566922.00	21996538.00	39412316.00	25681987.00	41861783.00	22408742.00	40690302.
2	1062	37577095.00	41457655.00	31467967.00	37577926.00	35845106.00	42953168.00	33817289.
3	1093	36012730.00	41667692.00	22837073.00	29288200.00	39553494.00	33513588.00	36827718.
4	1104	43416417.00	36290780.00	23679738.00	21183706.00	25210622.00	30864041.00	41173207.

2. Eliminar últimas 3 columnas

In [525]:

```
df_importaciones.drop(columns = df_importaciones.columns[-3:], inplace = True)
```

3. Convertir filas a otros tipos de datos

In [526]:

```
for col in columnas_importaciones:
  # Convierto primero a float ya que si no, no me deja pasar a int, porque son strin
  df_importaciones[col] = df_importaciones[col].astype(float).astype(int)

df_importaciones.dtypes
```

Out[526]:

```
distributor    int64
Rubber         int64
Brass          int64
Vinyl          int64
Granite        int64
Stone          int64
Brick          int64
Aluminum       int64
Glass          int64
Plexiglass     int64
Steel          int64
Wood           int64
Plastic        int64
dtype: object
```

Dataframe de importaciones final

In [527]:

```
df_importaciones
```

Out[527]:

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminum	Glass
0	1017	22431099	36031577	31118167	21322223	35382848	35280292	32362235	36836190
1	1055	27566922	21996538	39412316	25681987	41861783	22408742	40690302	37958885
2	1062	37577095	41457655	31467967	37577926	35845106	42953168	33817289	41602183
3	1093	36012730	41667692	22837073	29288200	39553494	33513588	36827718	29669764
4	1104	43416417	36290780	23679738	21183706	25210622	30864041	41173207	36719169
5	1169	43909858	27128183	36391549	21542467	34971118	24495086	37708256	26877104
6	1169	28550182	36499732	41512708	22262065	30270432	34506212	29217327	36706402
7	1183	21771886	43109659	23247953	22866072	31459461	30652665	42058181	41910985
8	1235	36667511	25366105	24902331	36605735	44673772	31541390	36511291	23484274
9	1302	39958024	40169347	35399555	25461219	37801759	30154265	42855837	31635653
10	1384	23722590	24623967	41133545	35578933	45015411	139362635	30191296	44104922
11	1418	25138331	38634366	35724628	21355595	42958842	27048824	32985225	33904151
12	1463	37817347	23272550	36228786	40908298	44287587	20840169	36762686	36281329
13	1489	38180046	28960197	38400970	23646886	39383770	122710022	39875739	29282534
14	1526	21870115	29728962	29018088	21585132	44760537	37955177	28714842	31884090
15	1553	28415088	38304440	32154709	23047802	23121310	39288035	22141394	29376999
16	1560	23507965	41845467	42055834	35453957	27278926	28907442	34052900	29536682
17	1599	33377107	29404830	37302875	38573860	27524149	29226121	42965132	31745917
18	1666	40875271	22374529	20935764	31145028	25311142	42786006	25934382	26856470
19	1679	38180046	28960197	38400970	23646886	39383770	122710022	39875739	29282534
20	1710	30129379	36728723	32550897	20871609	44756568	1643078851	34134175	21672939
21	1723	31142709	28352801	28274723	43010066	44667972	40622140	31505354	25344685
22	161	22957310	24387893	41579620	26804859	37698018	25843761	30924443	27838729
23	175	28135181	23695417	32743214	23511170	29707140	38298798	29378520	34091594
24	234	34544298	27882865	22664200	21057646	22595798	39745929	31984425	33608261
25	241	28318181	23132460	29998057	42240878	25721016	20894738	40671768	44403938
26	29	37794819	26812043	25270289	28139664	26173115	43607672	43718631	27250266
27	308	35622161	22845538	31984270	32821654	31742905	36952951	32453436	33975565
28	325	42018216	21936967	36106133	29906692	25729289	36214594	44207777	28603420
29	325	34349322	20737841	26603366	32141549	31818580	124353986	35732931	40238444
30	364	44229593	42500506	33478392	37522478	32650463	29599618	34120715	44676489
31	378	41444583	23735516	29708711	45198396	26911979	27704919	32220459	43944535
32	379	41387139	32973505	38520126	37505722	32781557	159624598	23844317	40207992
33	504	35858240	22072249	21646802	39650737	40985840	25023137	43022733	29624708
34	565	37489497	22875902	34374638	23295485	33803575	925835590	42627612	38176198
35	583	20607246	23005062	27506653	26062955	27696836	37419941	26762715	29284702
36	619	32853600	43936493	31944768	23457213	30668319	940975527	21908767	41517336

	distributor	Rubber	Brass	Vinyl	Granite	Stone	Brick	Aluminum	Glass
37	707	29585885	42849169	30521280	21745658	32580700	45204963	27528292	36931432
38	715	43576523	30181558	32773382	36669396	45128426	34340349	34807596	25353918
39	723	42160913	41409478	37110518	25210663	28016394	43238079	42829565	25437130
40	808	27799295	44568676	26843228	41036010	38237752	29463712	27749526	29618358
41	815	25138331	38634366	35724628	21355595	42958842	27048824	32985225	33904151
42	818	41761417	33709306	27497587	28311310	21683294	26276785	32530973	21500385
43	846	20880116	37111825	34422161	40759561	26050968	21080855	21443932	22394142
44	860	34050845	30151961	26956027	45003866	23879569	27287921	44733537	21113491
45	920	35580430	39978595	32046855	30147636	36212747	33365715	35482192	26604405

Análisis de los datos

Obtengamos algunos valores de interés para el dataframe de ventas e importaciones

In [528]:

```
medidas_ventas = {
    'Media': df_ventas.mean(),
    'Mediana': df_ventas.median(),
    'Maximo': df_ventas.max(),
    'Minimo': df_ventas.min(),
    'Desviacion Estandar': df_ventas.std(),
    'Suma de ventas': df_ventas.sum()
}

df_medidas_ventas = pd.DataFrame(medidas_ventas)
df_medidas_ventas
```

Out[528]:

	Media	Mediana	Maximo	Minimo	Desviacion Estandar	Suma de ventas
distributor	933.34	890.00	1723.00	29.00	500.12	41067.00
Rubber	37360242.65	36281745.60	90185311.22	0.00	23575592.67	1643850676.79
Brass	41833167.75	39286327.95	177874267.01	1346769.34	30637106.50	1840659380.95
Vinyl	43467934.56	41736189.62	177874267.01	1346769.34	30812148.98	1912589120.56
Granite	41938512.51	39088026.65	177874267.01	1346769.34	30316025.99	1845294550.27
Stone	37534531.72	36801565.80	90185311.22	1346769.34	22046995.00	1651519395.63
Brick	3469.50	0.00	42458.00	0.00	9185.90	152658.00
Aluminum	49178012.97	42962159.58	177874267.01	6706393.21	33115375.06	2163832570.85
Glass	49392633.70	42962159.58	177874267.01	1586769.34	37333892.57	2173275882.75
Plexiglass	590319106.11	136731561.59	5441399590.44	1346769.34	957632863.33	25974040668.70
Steel	1328.59	192.10	11121.98	0.00	2458.28	58457.97
Wood	43876478.65	41735189.62	177874267.01	1346769.34	30330485.02	1930565060.44
Plastic	46530413.05	32585187.64	325185311.22	20982.80	62847437.58	2047338174.32

In [529]:
```
medidas_importaciones = {
    'Media': df_importaciones.mean(),
    'Mediana': df_importaciones.median(),
    'Maximo': df_importaciones.max(),
    'Minimo': df_importaciones.min(),
    'Desviacion Estandar': df_importaciones.std(),
    'Suma de importaciones': df_importaciones.sum()
}

df_medidas_importaciones = pd.DataFrame(medidas_importaciones)
df_medidas_importaciones
```

Out[529]:

	Media	Mediana	Maximo	Minimo	Desviacion Estandar	Suma de importaciones
distributor	925.24	890.00	1723	29	498.40	42561
Rubber	33225236.07	34446810.00	44229593	20607246	7267904.19	1528360859
Brass	31783336.70	29940461.50	44568676	20737841	7770710.72	1462033488
Vinyl	31960348.93	32100782.00	42055834	20935764	5788841.17	1470176051
Granite	30047227.07	28225487.00	45198396	20871609	7848823.72	1382172445
Stone	33845945.67	32716010.00	45128426	21683294	7334839.60	1556913501
Brick	117615388.15	34423280.50	1643078851	20840169	295829071.51	5410307855
Aluminum	34478925.96	34086807.50	44733537	21443932	6458598.84	1586030594
Glass	32455944.57	31690785.00	44676489	21113491	6715936.36	1492973450
Plexiglass	453398023.54	35446610.00	9440858902	21395275	1870101786.45	20856309083
Steel	298366739.78	32093718.00	12242337842	21114989	1800191367.67	13724870030
Wood	31783618.13	32084491.00	44049215	20797348	7753252.00	1462046434
Plastic	33008902.98	33213202.50	44234252	20684792	7450260.53	1518409537

De acá sacamos varios puntos:

- Los productos Brick y Steel son los que menores ventas dieron
- Hay ventas con valor 0 en rubber, brick y steel
- El valor de las importaciones para brick, steel y plexiglass son muy altos, contrastando con estos valores para las ventas

Modelado de datos

Vamos a crear un dataframe con la suma de las ventas y importaciones para cada producto, y el ratio entre las ventas y las importaciones

In [530]:
```
# Aprovecho los dataframes con las medidas para ventas e importaciones y tomo la
# suma para cada producto de cada dataframe
df_suma_ventas = pd.DataFrame(df_medidas_ventas['Suma de ventas'])
df_suma_importaciones = pd.DataFrame(df_medidas_importaciones['Suma de importaciones
```

In [531]:
```
# Hago el merge
df_totales = pd.merge(df_suma_ventas, df_suma_importaciones, left_index = True, righ
df_totales
```

Out[531]:

	Suma de ventas	Suma de importaciones
distributor	41067.00	42561
Rubber	1643850676.79	1528360859
Brass	1840659380.95	1462033488
Vinyl	1912589120.56	1470176051
Granite	1845294550.27	1382172445
Stone	1651519395.63	1556913501
Brick	152658.00	5410307855
Aluminum	2163832570.85	1586030594
Glass	2173275882.75	1492973450
Plexiglass	25974040668.70	20856309083
Steel	58457.97	13724870030
Wood	1930565060.44	1462046434
Plastic	2047338174.32	1518409537

In [532]:

```
# Para convertir a millones
df_totales['Suma de ventas'] = df_totales['Suma de ventas'] / 1000000
df_totales['Suma de importaciones'] = df_totales['Suma de importaciones']  / 1000000

df_totales['Ratio ventas/importaciones'] = (df_totales['Suma de ventas'] / df_totale

# Elimino la fila que tiene los distributor y no nos interesa
df_totales = df_totales.drop(index = 'distributor')

df_totales.rename(columns = {
    'Suma de ventas': 'Suma de ventas (En millones)',
    'Suma de importaciones': 'Suma de importaciones (En millones)'
}, inplace = True)

df_totales
```

Out[532]:

	Suma de ventas (En millones)	Suma de importaciones (En millones)	Ratio ventas/importaciones
Rubber	1643.85	1528.36	1.08
Brass	1840.66	1462.03	1.26
Vinyl	1912.59	1470.18	1.30
Granite	1845.29	1382.17	1.34
Stone	1651.52	1556.91	1.06
Brick	0.15	5410.31	0.00
Aluminum	2163.83	1586.03	1.36
Glass	2173.28	1492.97	1.46
Plexiglass	25974.04	20856.31	1.25
Steel	0.06	13724.87	0.00
Wood	1930.57	1462.05	1.32
Plastic	2047.34	1518.41	1.35

Con esto confirmamos lo que hicimos en el análisis, que las ventas eran mucho menores que las importaciones para los productos "Brick" y "Steel"

Visualización

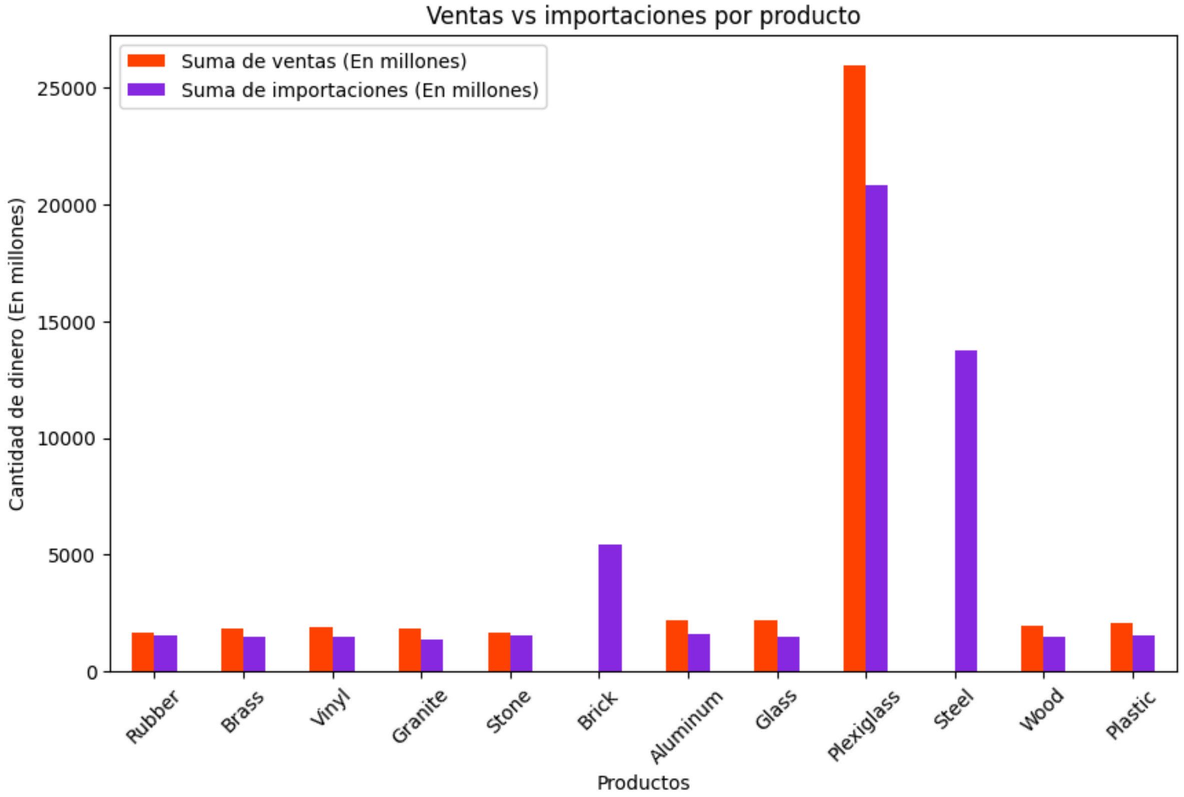
Ahora veamos gráficamente la pequeña cantidad de ventas que hay para los productos "Brick" y "Steel", comparando las ventas e importaciones de forma grafica

In [533]:

```
ax = df_totales[['Suma de ventas (En millones)', 'Suma de importaciones (En millones

ax.set_title('Ventas vs importaciones por producto')
ax.set_xlabel('Productos')
ax.set_ylabel('Cantidad de dinero (En millones)')
plt.xticks(rotation = 0)
plt.xticks(rotation = 45)

plt.show()
```

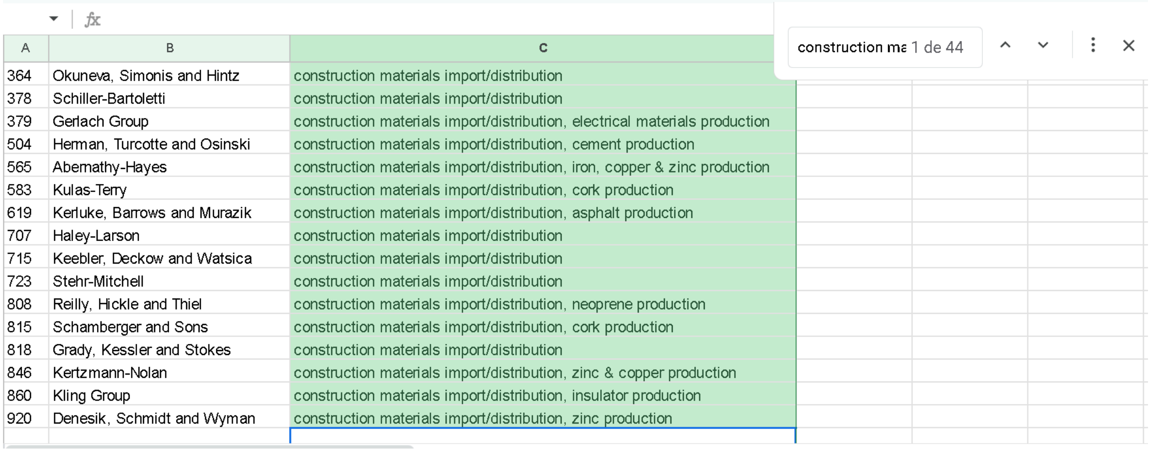


Y confirmamos, graficamente, que los productos "Brick" y "Steel" no aportan de forma positiva a las ventas

Conclusiones finales

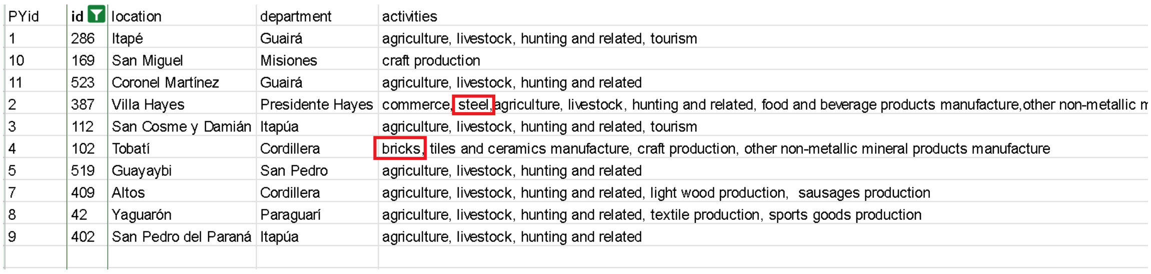
Ya sabemos que el problema fue en las ventas de los productos "Brick" y "Steel", viendolo al comparar el precio en los datos de ventas e importaciones

Cuando miramos la hoja "distributors_profiles", luego de eliminar la columna "Years in the construction market" (Que no sirve para nada) y filtrar por id no nulos, vemos que casi todas las actividades de los distribuidores están relacionadas con la importacion/distribucion de materiales de construcción, y algunos otros producen cierto material en particular



A	B	C
364	Okuneva, Simonis and Hintz	construction materials import/distribution
378	Schiller-Bartoletti	construction materials import/distribution
379	Gerlach Group	construction materials import/distribution, electrical materials production
504	Herman, Turcotte and Osinski	construction materials import/distribution, cement production
565	Abernathy-Hayes	construction materials import/distribution, iron, copper & zinc production
583	Kulas-Terry	construction materials import/distribution, cork production
619	Kerluke, Barrows and Murazik	construction materials import/distribution, asphalt production
707	Haley-Larson	construction materials import/distribution
715	Keebler, Deckow and Watsica	construction materials import/distribution
723	Stehr-Mitchell	construction materials import/distribution
808	Reilly, Hickle and Thiel	construction materials import/distribution, neoprene production
815	Schamberger and Sons	construction materials import/distribution, cork production
818	Grady, Kessler and Stokes	construction materials import/distribution
846	Kertzmann-Nolan	construction materials import/distribution, zinc & copper production
860	Kling Group	construction materials import/distribution, insulator production
920	Denesik, Schmidt and Wyman	construction materials import/distribution, zinc production

Cuando miramos la hoja "location_profiles", luego de filtrar por ids no nulos ni 0 (Mostrando las ciudades en donde existe un distribuidor de la compañia) vemos que hay muy pocas ciudades cuyas actividades están relacionadas con los productos "Brick" y "Steel"



PYid	id	location	department	activities
1	286	Itapé	Guairá	agriculture, livestock, hunting and related, tourism
10	169	San Miguel	Misiones	craft production
11	523	Coronel Martínez	Guairá	agriculture, livestock, hunting and related
2	387	Villa Hayes	Presidente Hayes	commerce, steel, agriculture, livestock, hunting and related, food and beverage products manufacture,other non-metallic m
3	112	San Cosme y Damián	Itapúa	agriculture, livestock, hunting and related, tourism
4	102	Tobatí	Cordillera	bricks, tiles and ceramics manufacture, craft production, other non-metallic mineral products manufacture
5	519	Guayaybi	San Pedro	agriculture, livestock, hunting and related
7	409	Altos	Cordillera	agriculture, livestock, hunting and related, light wood production, sausages production
8	42	Yaguarón	Paraguarí	agriculture, livestock, hunting and related, textile production, sports goods production
9	402	San Pedro del Paraná	Itapúa	agriculture, livestock, hunting and related

Por lo tanto, podemos concluir que el fallo de la estrategia fue en querer vender los productos "Brick" y "Steel" en lugares donde no hay actividades donde se necesite el uso de estos productos, y por lo tanto es imposible venderlos