Análisis de ventas

Table of contents

0.0.1 Carga de librerías

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

0.0.2 Juntar todas las tablas disponibles en una sola

```
mainpath = r"C:\Users\Carlos\OneDrive\Formación\Python\Casos de análisis\Caso 1\SalesAnaly
# Almacenar el nombre de cada archivo según una dirección
files = [file for file in os.listdir(mainpath)]
```

```
mainpath = r"C:\Users\Carlos\OneDrive\Formación\Python\Casos de análisis\Caso 1\SalesAnaly
# Almacenar el nombre de cada archivo según una dirección
files = [file for file in os.listdir(mainpath)]

# Declaración de un Dataframe vacío
all_months_data = pd.DataFrame()

# Carga de todos los archivos en un dirección específica
for file in files:
    df = pd.read_csv(os.path.join(mainpath,file))
    df.insert(loc=len(df.columns),
              column='File_origin',
             value=file)
    all_months_data = pd.concat([all_months_data,df])

# Creación de un nuevo archivo csv con los datos combinados
all_months_data.to_csv("all_data.csv",index = False)
```

0.0.3 Importar el archivo generado

```
all_data = pd.read_csv("all_data.csv")
print(all_data.info())
print('------------------------------- \n')
print(all_data.isnull().values.any())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186850 entries, 0 to 186849
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          186305 non-null  object
 1   Product           186305 non-null  object
 2   Quantity Ordered  186305 non-null  object
 3   Price Each        186305 non-null  object
 4   Order Date        186305 non-null  object
 5   Purchase Address  186305 non-null  object
 6   File_origin       186850 non-null  object
dtypes: object(7)
memory usage: 10.0+ MB
None
```

```
--------------------------------

True
```

0.0.4 Lipieza del dataset

```
# Eliminar las filas vacías en todas sus variables
# Considerar que el método "drop.na" no modifica el dataset original
all_data = all_data.dropna(subset=all_data.columns.values.tolist()[:-1])
print(all_data.info())
print('-------------------------------- \n')
print(all_data.isnull().values.any())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 186305 entries, 0 to 186849
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   Order ID          186305 non-null  object
 1   Product           186305 non-null  object
 2   Quantity Ordered  186305 non-null  object
 3   Price Each        186305 non-null  object
 4   Order Date        186305 non-null  object
 5   Purchase Address  186305 non-null  object
 6   File_origin       186305 non-null  object
dtypes: object(7)
memory usage: 11.4+ MB
None
--------------------------------

False
```

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

Se detectó que existen campos con valores no numéricos

```
all_data["Quantity Ordered"].unique().tolist()
```

```
['2', '1', '3', '5', 'Quantity Ordered', '4', '7', '6', '8', '9']
```

```
# Los archivos tienen filas copiadas
# El método .loc no modifica el dataset original
all_data.loc[(all_data["Quantity Ordered"] == "Quantity Ordered")].head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	File_origin
519	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Sales_April_
1149	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Sales_April_
1155	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Sales_April_
2878	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Sales_April_
2893	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Sales_April_

```
all_data = all_data.loc[(all_data["Quantity Ordered"] != "Quantity Ordered")]
print(all_data.info())
print('------------------------------ \n')
print(all_data.isnull().values.any())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   Order ID          185950 non-null  object
 1   Product           185950 non-null  object
 2   Quantity Ordered  185950 non-null  object
 3   Price Each        185950 non-null  object
 4   Order Date        185950 non-null  object
 5   Purchase Address  185950 non-null  object
 6   File_origin       185950 non-null  object
```

```
dtypes: object(7)
memory usage: 11.3+ MB
None
--------------------------------

False
```

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

0.0.5 Pregunta 1: ¿Qué mes tuvo las ventas más altas? ¿Cuánto se ganó en dicho periodo?

0.0.5.1 Adición de nuevas columnas

```
all_data['Month'] = all_data["Order Date"].str[0:2]
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

0.0.5.2 Cambio del tipo de variables

```
#all_data["Month"] = all_data["Month"].astype('int32')
```

```
all_data.dtypes
```

```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
File_origin         object
Month               object
dtype: object
```

```
convert_dic = {"Order ID": "int32","Product": str,"Quantity Ordered": int,
               "Price Each": float,"Order Date": str,"Purchase Address": str,
               "File_origin": str}
all_data = all_data.astype(convert_dic)
print(all_data.dtypes)
```

```
Order ID              int32
Product              object
Quantity Ordered      int32
Price Each          float64
Order Date           object
Purchase Address     object
File_origin          object
Month                object
dtype: object
```

0.0.5.3 Adición de la columna de ventas

```
all_data["Sales"] = all_data["Quantity Ordered"] * all_data["Price Each"]
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

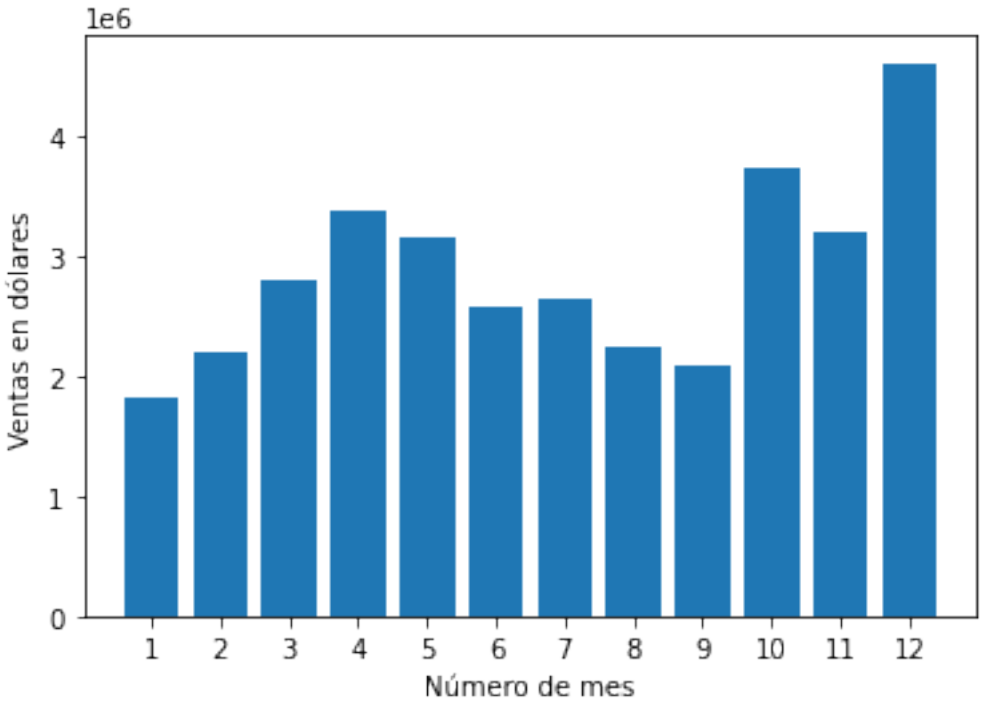
0.0.5.4 Respuesta a la pregunta

```
# Diciembre fue el mes con las mejores ventas
results = all_data.groupby("Month").sum()[["Quantity Ordered","Sales"]]
results
```

Month	Quantity Ordered	Sales
01	10903	1822256.73
02	13449	2202022.42
03	17005	2807100.38
04	20558	3390670.24
05	18667	3152606.75
06	15253	2577802.26
07	16072	2647775.76
08	13448	2244467.88
09	13109	2097560.13
10	22703	3736726.88
11	19798	3199603.20
12	28114	4613443.34

```
months = range(1,13)
plt.bar(months,results["Sales"])
plt.xticks(months)
plt.ylabel("Ventas en dólares")
plt.xlabel("Número de mes")
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



0.0.6 ¿Qué ciudad tuvo las ventas más altas? ¿Cuánto se ganó en dicho periodo?

0.0.6.1 Separar columnas

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

```
# split column and add new columns to df
all_data2 = all_data.copy()
all_data2[['Address', 'City', 'Code']] = all_data2['Purchase Address'].str.split(',', expa
all_data2.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Ad
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

```
results2 = all_data2.groupby("City").sum()[["Quantity Ordered","Sales"]]
results2
```

City	Quantity Ordered	Sales
Atlanta	16602	2795498.58
Austin	11153	1819581.75
Boston	22528	3661642.01
Dallas	16730	2767975.40
Los Angeles	33289	5452570.80
New York City	27932	4664317.43
Portland	14053	2320490.61
San Francisco	50239	8262203.91
Seattle	16553	2747755.48

Es posible aislar solamente la columna de interés

```
all_data['City'] = all_data['Purchase Address'].apply(lambda x: x.split(',')[1])
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Ad
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

Considerar que los nombre de las ciudades pueden estar duplicados si no se extrae el estado

```
results3 = all_data.groupby("City").sum()[["Quantity Ordered","Sales"]]
results3
```

City	Quantity Ordered	Sales
Atlanta	16602	2795498.58
Austin	11153	1819581.75
Boston	22528	3661642.01
Dallas	16730	2767975.40
Los Angeles	33289	5452570.80
New York City	27932	4664317.43
Portland	14053	2320490.61
San Francisco	50239	8262203.91
Seattle	16553	2747755.48

El método apply funcionará bien en tanto no se traten excesivas cantidades de datos

```
def get_city(address):
    return address.split(',')[1]

def get_state(address):
    return address.split(',')[2].split(' ')[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x:f"{get_city(x)} ({get_state
all_data.head()
```

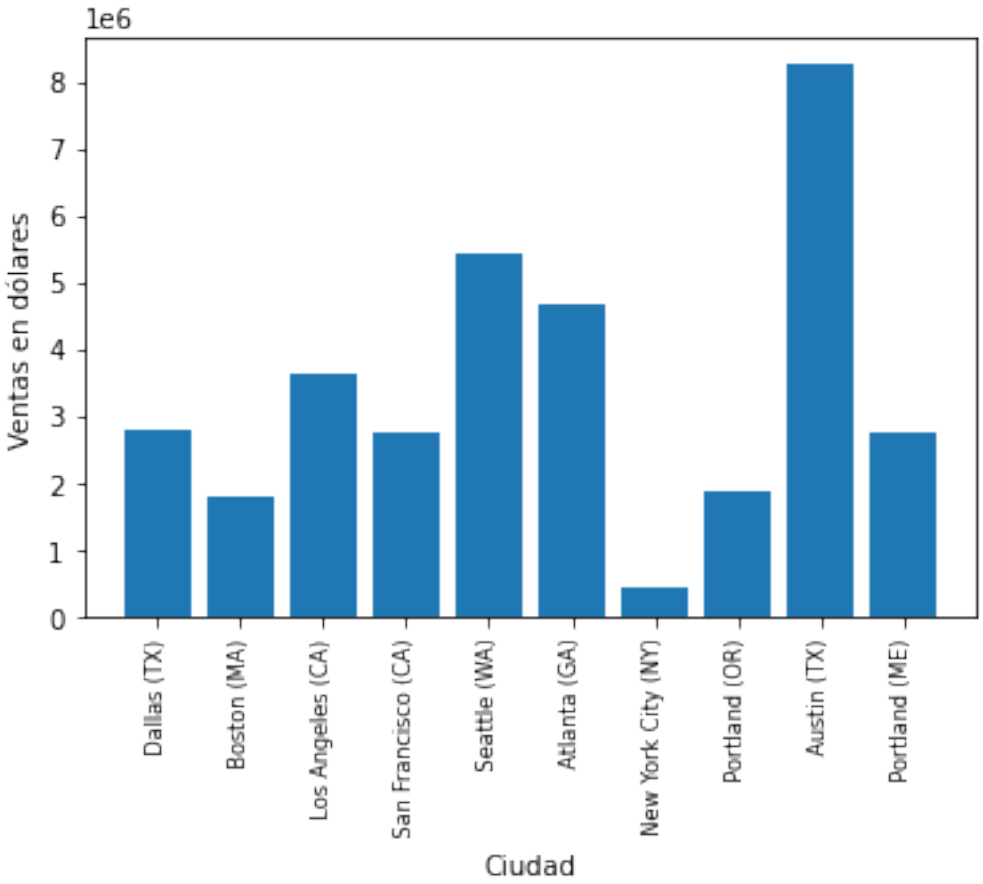
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

```
results4 = all_data.groupby("City").sum()[["Quantity Ordered","Sales"]]
results4
```

City	Quantity Ordered	Sales
Atlanta (GA)	16602	2795498.58
Austin (TX)	11153	1819581.75
Boston (MA)	22528	3661642.01

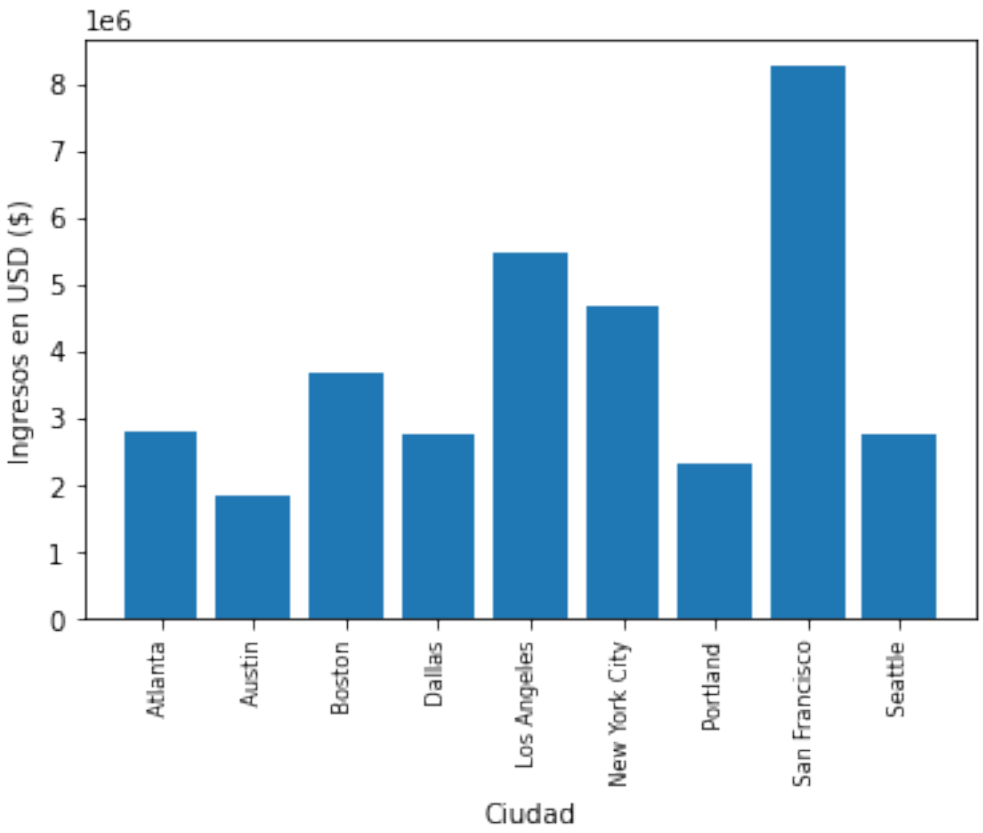
City	Quantity Ordered	Sales
Dallas (TX)	16730	2767975.40
Los Angeles (CA)	33289	5452570.80
New York City (NY)	27932	4664317.43
Portland (ME)	2750	449758.27
Portland (OR)	11303	1870732.34
San Francisco (CA)	50239	8262203.91
Seattle (WA)	16553	2747755.48

```
# Este gráfico está errado por el orden de los ejes y las cantidades
cities = all_data["City"].unique()
plt.bar(cities,results4["Sales"])
plt.xticks(cities, rotation = "vertical",size =8)
plt.ylabel("Ventas en dólares")
plt.xlabel("Ciudad")
plt.show()
```



Como científicos de datos debemos preguntarnos por qué San Francisco tinen las ventas más altas... Pueden formularse varias hipótesis que viene de una segmentación de mercado: nivel de ingresos, ubicación, edad, etc.

```
keys = [city for city, df in all_data2.groupby(['City'])]
plt.bar(keys,all_data2.groupby(['City']).sum()['Sales'])
plt.ylabel('Ingresos en USD ($)')
plt.xlabel('Ciudad')
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



0.0.7 ¿Cuál es el mejor momento para desplegar avisos publicitarios de manera que se maximice la probabilidad de que un cliente pueda comprar un producto?

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase A
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St,
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnu
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St,

```
all_data["Order Date"] = pd.to_datetime(all_data["Order Date"])
```

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purcha
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Ch
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spr
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spr
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th

```
all_data["Hour"] = all_data["Order Date"].dt.hour
all_data["Minute"] = all_data["Order Date"].dt.minute
all_data.head()
```

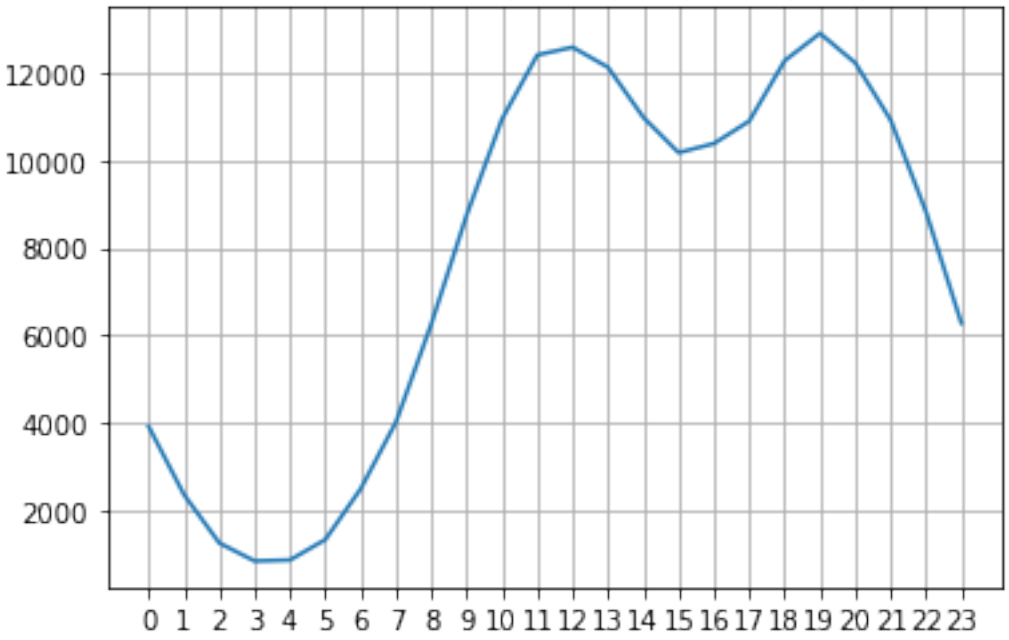
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purcha
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Ch
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spr
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spr
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th

```
# Add hour column
all_data['Hour'] = pd.to_datetime(all_data['Order Date']).dt.hour
all_data['Minute'] = pd.to_datetime(all_data['Order Date']).dt.minute
all_data['Count'] = 1
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purcha
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Ch
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spr
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spr
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th

```
keys = [pair for pair, df in all_data.groupby(['Hour'])]

plt.plot(keys, all_data.groupby(['Hour']).count()['Count'])
```

```
plt.xticks(keys)
plt.grid()
plt.show()
```



0.0.8 ¿Qué productos se venden juntos usualmente?

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purcha
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Ch
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spr
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spr
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th

```
df = all_data[all_data["Order ID"].duplicated(keep = False)]

df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
```

```
df = df[['Order ID','Grouped']].drop_duplicates()

df.head()
```

```
C:\Users\Carlos\AppData\Local\Temp\ipykernel_1800\1224629595.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
```

	Order ID	Grouped
3	176560	Google Phone,Wired Headphones
18	176574	Google Phone,USB-C Charging Cable
30	176585	Bose SoundSport Headphones,Bose SoundSport Hea...
32	176586	AAA Batteries (4-pack),Google Phone
119	176672	Lightning Charging Cable,USB-C Charging Cable

```
from itertools import combinations
from collections import Counter

count = Counter()

for row in df['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list,2)))

for key,value in count.most_common(10):
    print(key,value)
```

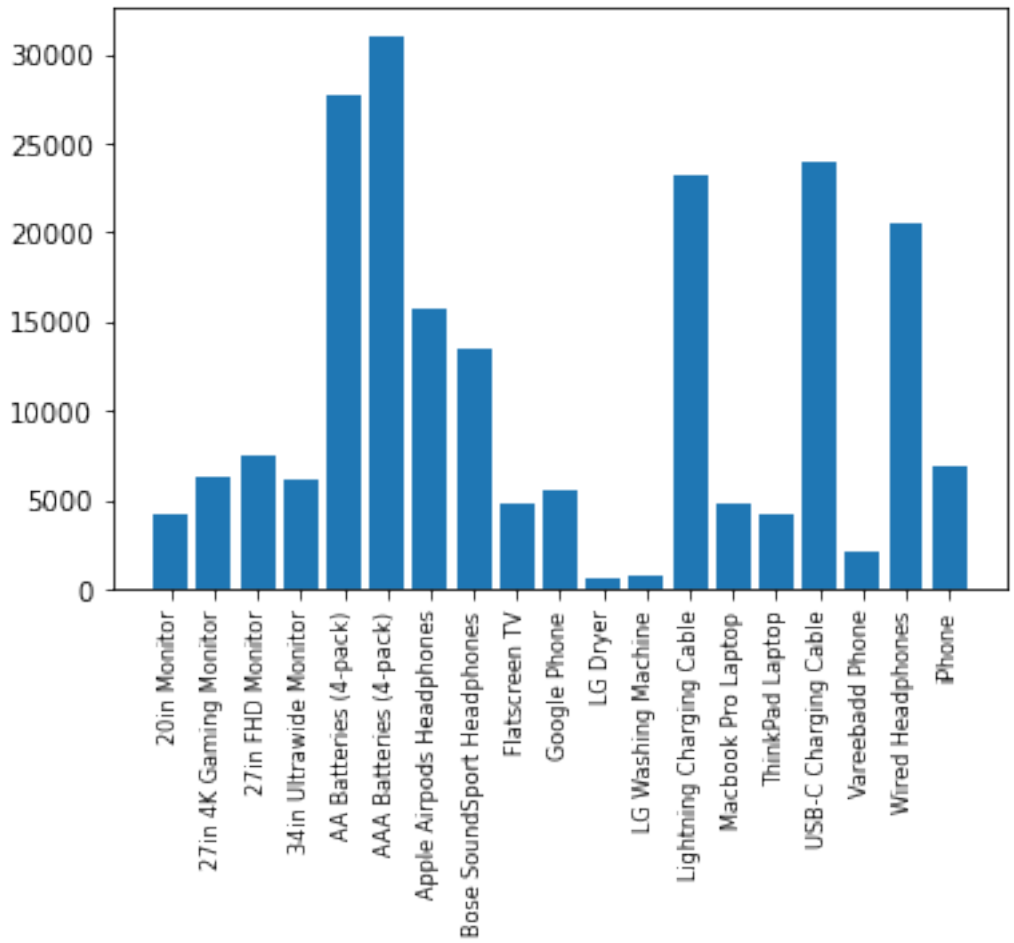
```
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
```

0.0.9 ¿Qué producto vendió más? ¿Por qué fue así?

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purcha
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Ch
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spr
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spr
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th

```
product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']

keys = [pair for pair, df in product_group]
plt.bar(keys, quantity_ordered)
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



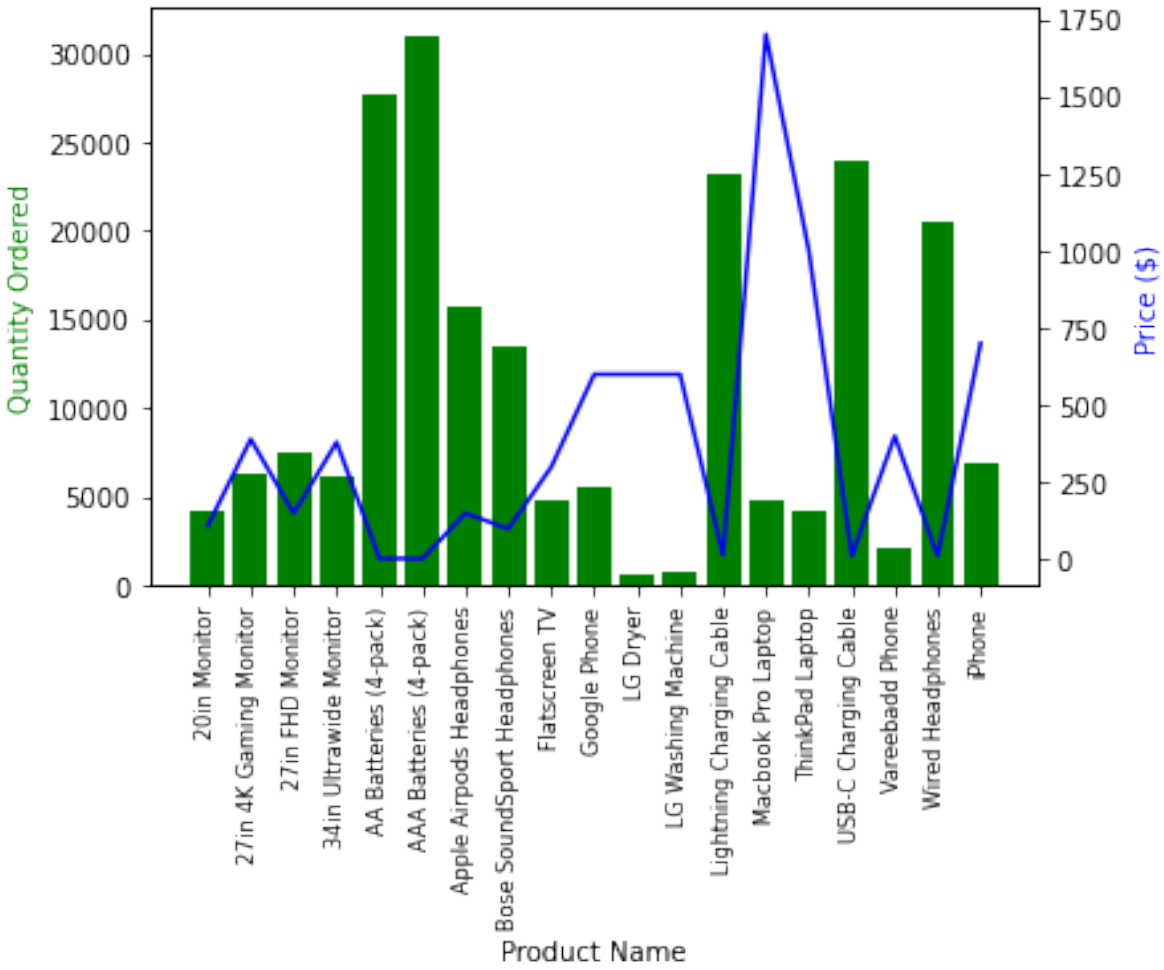

```
prices = all_data.groupby('Product').mean()['Price Each']
print(prices)
```

```
Product
20in Monitor                  109.99
27in 4K Gaming Monitor        389.99
27in FHD Monitor              149.99
34in Ultrawide Monitor        379.99
AA Batteries (4-pack)           3.84
AAA Batteries (4-pack)          2.99
Apple Airpods Headphones      150.00
Bose SoundSport Headphones     99.99
Flatscreen TV                 300.00
Google Phone                  600.00
```

```
LG Dryer                        600.00
LG Washing Machine              600.00
Lightning Charging Cable         14.95
Macbook Pro Laptop             1700.00
ThinkPad Laptop                 999.99
USB-C Charging Cable             11.95
Vareebadd Phone                 400.00
Wired Headphones                 11.99
iPhone                          700.00
Name: Price Each, dtype: float64
```

```
# Referenced: https://stackoverflow.com/questions/14762181/adding-a-y-axis-label-to-second

prices = all_data.groupby('Product').mean()['Price Each']

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(keys, quantity_ordered, color='g')
ax2.plot(keys, prices, color='b')

ax1.set_xlabel('Product Name')
ax1.set_ylabel('Quantity Ordered', color='g')
ax2.set_ylabel('Price ($)', color='b')
ax1.set_xticklabels(keys, rotation='vertical', size=8)

fig.show()
```

```
C:\Users\Carlos\AppData\Local\Temp\ipykernel_1800\136096346.py:14: UserWarning: FixedFormatte
  ax1.set_xticklabels(keys, rotation='vertical', size=8)
C:\Users\Carlos\AppData\Local\Temp\ipykernel_1800\136096346.py:16: UserWarning: Matplotlib is
  fig.show()
```


Quantity Ordered
Price ($)
Product Name
0
5000
10000
15000
20000
25000
30000
0
250
500
750
1000
1250
1500
1750
20in Monitor
27in 4K Gaming Monitor
27in FHD Monitor
34in Ultrawide Monitor
AA Batteries (4-pack)
AAA Batteries (4-pack)
Apple Airpods Headphones
Bose SoundSport Headphones
Flatscreen TV
Google Phone
LG Dryer
LG Washing Machine
Lightning Charging Cable
Macbook Pro Laptop
ThinkPad Laptop
USB-C Charging Cable
Vareebadd Phone
Wired Headphones
iPhone