DATASET EXPLORATION WITH PANDAS

In [2]:
```
import pandas as pd
import os
```

merging sales data of 12 months into a single file

In [3]:
```
files=[file for file in os.listdir('C:/Users/lekshmi/Downloads/salesdata')]
all_months_data=pd.DataFrame()
for file in files:
    df=pd.read_csv("C:/Users/lekshmi/Downloads/salesdata/"+file)
    all_months_data=pd.concat([all_months_data,df])
all_months_data.to_csv("all_data.csv",index=False)
```

In [4]:
```
all_data=pd.read_csv("all_data.csv")
all_data.head()
```

Out[4]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	176559	Bose SoundSport Headphones	1	99.99	04-07-2019 22:30	682 Chestnut St, Boston, MA 02215
2	176560	Google Phone	1	600	04-12-2019 14:38	669 Spruce St, Los Angeles, CA 90001
3	176560	Wired Headphones	1	11.99	04-12-2019 14:38	669 Spruce St, Los Angeles, CA 90001
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

preprocssing of data

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Order ID            40
Product             40
Quantity Ordered    40
Price Each          40
Order Date          40
Purchase Address    40
dtype: int64
```

In [6]:
```
all_data=all_data.dropna(how='all')
```

In [ ]:
```

```

adding month column separately

In [7]:
```
all_data['Month']=all_data['Order Date'].str[0:2]
all_data.head()
```

Out[7]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	04
1	176559	Bose SoundSport Headphones	1	99.99	04-07-2019 22:30	682 Chestnut St, Boston, MA 02215	04
2	176560	Google Phone	1	600	04-12-2019 14:38	669 Spruce St, Los Angeles, CA 90001	04
3	176560	Wired Headphones	1	11.99	04-12-2019 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04

#as while conversion to int data type for column month, the error pops that "or" cannot be converted. hence we are eliminating those

In [8]:
```
all_data=all_data[all_data['Order Date'].str[0:2]!="Or"]
```

In [9]:
```
all_data['Month']=all_data['Month'].astype('int32')
```

month with highest sales

In [10]:
```
all_data['Quantity Ordered']=pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each']=pd.to_numeric(all_data['Price Each'])
```

In [11]:
```
all_data['Sales']=all_data['Quantity Ordered']*all_data['Price Each']
```

In [12]:
```
results=all_data.groupby('Month').sum()
results
```

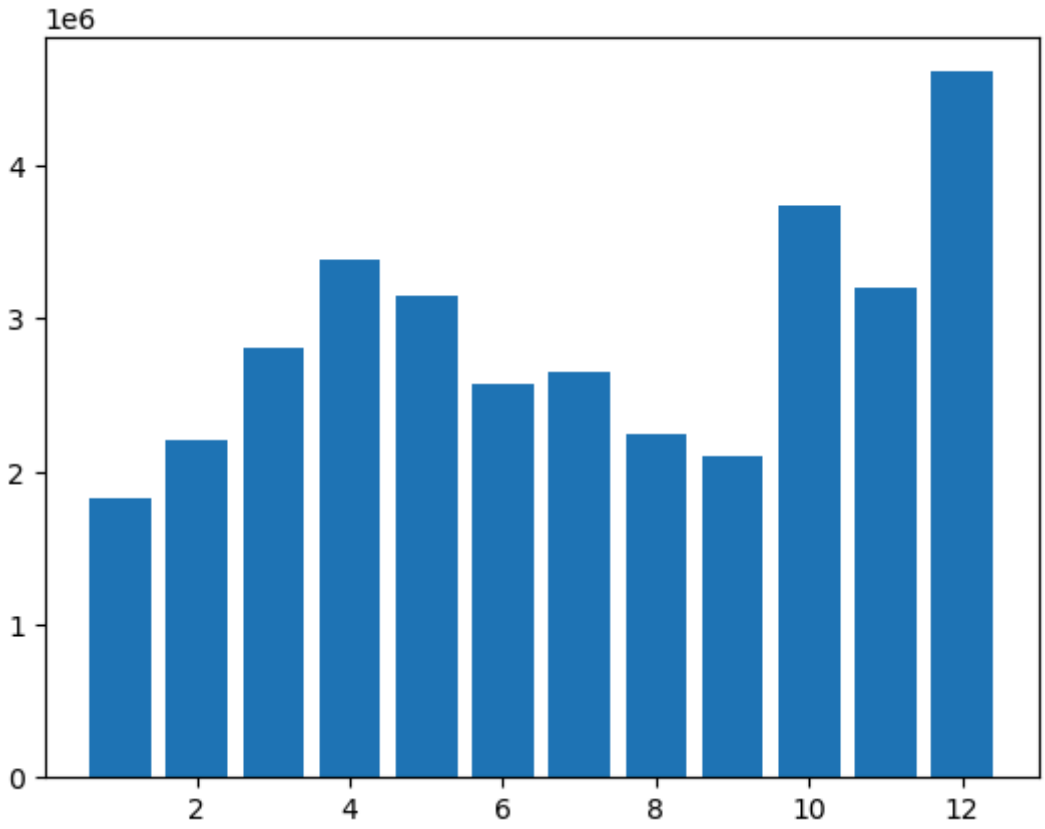
```
C:\Users\lekshmi\AppData\Local\Temp\ipykernel_18872\1850623623.py:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  results=all_data.groupby('Month').sum()
```

Out[12]:

Month	Quantity Ordered	Price Each	Sales
1	10903	1811768.38	1822256.73
2	13449	2188884.72	2202022.42
3	17005	2791207.83	2807100.38
4	20558	3367671.02	3390670.24
5	18667	3135125.13	3152606.75
6	15253	2562025.61	2577802.26
7	16072	2632539.56	2647775.76
8	13448	2230345.42	2244467.88
9	13109	2084992.09	2097560.13
10	22703	3715554.83	3736726.88
11	19798	3180600.68	3199603.20
12	28114	4588415.41	4613443.34

In [13]:
```
import matplotlib.pyplot as plt
months=range(1,13)
plt.bar(months,results['Sales'])
```

Out[13]: <BarContainer object of 12 artists>



city with most sales

for this initially we have to extract the city and state from purchase address

In [20]:
```
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: get_city(x) +
all_data.head()
```

Out[20]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
1	176559	Bose SoundSport Headphones	1	99.99	04-07-2019 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
2	176560	Google Phone	1	600.00	04-12-2019 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
3	176560	Wired Headphones	1	11.99	04-12-2019 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)

In [21]:
```
results=all_data.groupby('City').sum()
results
```

```
C:\Users\lekshmi\AppData\Local\Temp\ipykernel_18872\3338049859.py:1: Futur
eWarning: The default value of numeric_only in DataFrameGroupBy.sum is dep
recated. In a future version, numeric_only will default to False. Either s
pecify numeric_only or select only columns which should be valid for the f
unction.
  results=all_data.groupby('City').sum()
```

Out[21]:

City	Quantity Ordered	Price Each	Month	Sales
Atlanta (GA)	16602	2779908.20	104794	2795498.58
Austin (TX)	11153	1809873.61	69829	1819581.75
Boston (MA)	22528	3637409.77	141112	3661642.01
Dallas (TX)	16730	2752627.82	104620	2767975.40
Los Angeles (CA)	33289	5421435.23	208325	5452570.80
New York City (NY)	27932	4635370.83	175741	4664317.43
Portland (ME)	2750	447189.25	17144	449758.27
Portland (OR)	11303	1860558.22	70621	1870732.34
San Francisco (CA)	50239	8211461.74	315520	8262203.91
Seattle (WA)	16553	2733296.01	104941	2747755.48

suitable time for displaying advertisements to maximise likelihood of customer buying products

for this we will be using date time library

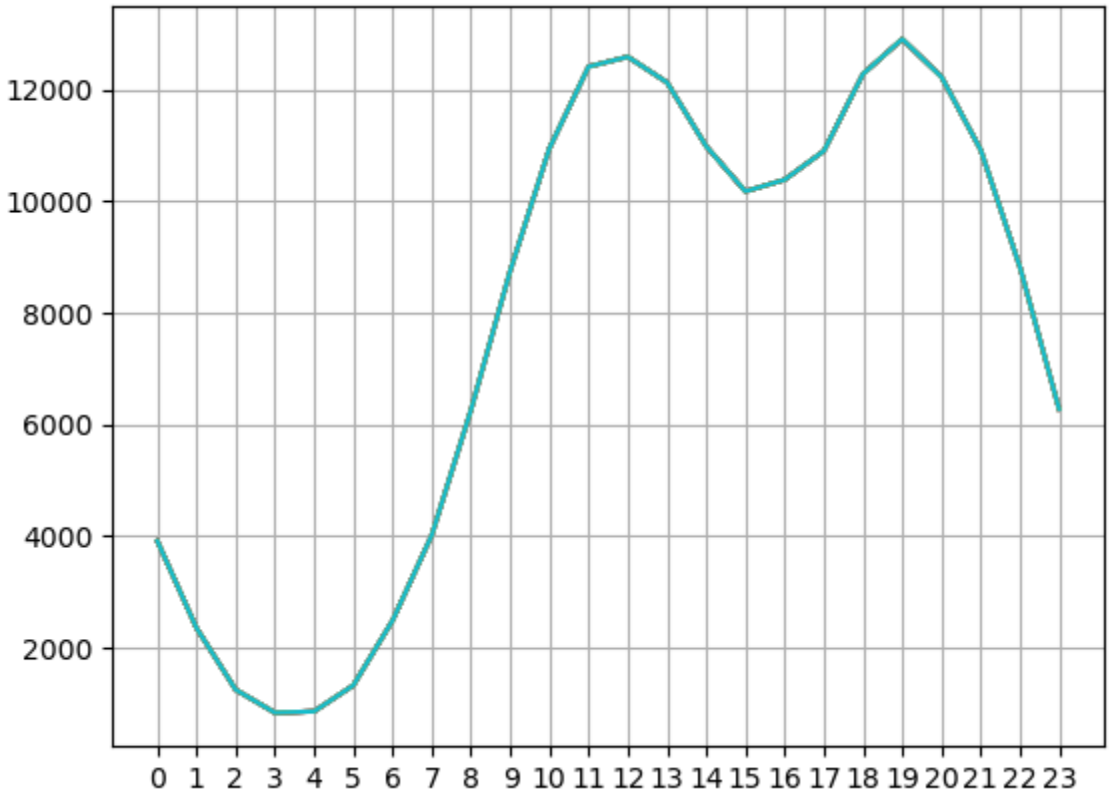
In [22]:
```
all_data['Order Date']=pd.to_datetime(all_data['Order Date'])
```

In [23]:
```
all_data['Hour']=all_data['Order Date'].dt.hour
all_data['Minute']=all_data['Order Date'].dt.minute
all_data.head()
```

Out[23]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)	8
1	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)	22
2	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)	14
3	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)	14
4	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)	9

In [27]:
```
hours= [hour for hour, df in all_data.groupby('Hour')]
plt.plot(hours,all_data.groupby(['Hour']).count())
plt.xticks(hours)
plt.grid()
plt.show()
```



identifying products which are sold together

In [29]:
```
df=all_data[all_data['Order ID'].duplicated(keep=False)]
df['Grouped']=df.groupby('Order ID')['Product'].transform(lambda x: ','.joi
df.head()
```

```
C:\Users\lekshmi\AppData\Local\Temp\ipykernel_18872\2805960672.py:2: Setti
ngWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  df['Grouped']=df.groupby('Order ID')['Product'].transform(lambda x: ','.
join(x))
```

Out[29]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour
2	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)	14
3	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)	14
17	176574	Google Phone	1	600.00	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)	19
18	176574	USB-C Charging Cable	1	11.95	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	11.95	Los Angeles (CA)	19
29	176585	Bose SoundSport Headphones	1	99.99	2019-04-07 11:31:00	823 Highland St, Boston, MA 02215	4	99.99	Boston (MA)	11

In [30]:
```
df=df[['Order ID','Grouped']].drop_duplicates()
df.head()
```

Out[30]:

	Order ID	Grouped
2	176560	Google Phone,Wired Headphones
17	176574	Google Phone,USB-C Charging Cable
29	176585	Bose SoundSport Headphones,Bose SoundSport Hea...
31	176586	AAA Batteries (4-pack),Google Phone
118	176672	Lightning Charging Cable,USB-C Charging Cable

counting the number of pairs

In [31]:
```
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)))
count.most_common(10)
```

Out[31]:
```
[(('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)]
```

identifying the most sold product and the reason for the same

In [34]:
```
product_group=all_data.groupby('Product')
quantity_ordered=product_group.sum()['Sales']
quantity_ordered
```

```
C:\Users\lekshmi\AppData\Local\Temp\ipykernel_18872\1518575911.py:2: Futur
eWarning: The default value of numeric_only in DataFrameGroupBy.sum is dep
recated. In a future version, numeric_only will default to False. Either s
pecify numeric_only or select only columns which should be valid for the f
unction.
  quantity_ordered=product_group.sum()['Sales']
```

Out[34]:
```
Product
20in Monitor                   454148.71
27in 4K Gaming Monitor        2435097.56
27in FHD Monitor              1132424.50
34in Ultrawide Monitor        2355558.01
AA Batteries (4-pack)          106118.40
AAA Batteries (4-pack)          92740.83
Apple Airpods Headphones      2349150.00
Bose SoundSport Headphones    1345565.43
Flatscreen TV                 1445700.00
Google Phone                  3319200.00
LG Dryer                       387600.00
LG Washing Machine             399600.00
Lightning Charging Cable       347094.15
Macbook Pro Laptop            8037600.00
ThinkPad Laptop               4129958.70
USB-C Charging Cable           286501.25
Vareebadd Phone                827200.00
Wired Headphones               246478.43
iPhone                        4794300.00
Name: Sales, dtype: float64
```

In [ ]: