Import Necessary Libraries

In [4]:

```
import pandas as pd
import os
```

Task 1 #: Merging 12 months of sales data into a single CSV file

In [5]:

```
df = pd.read_csv("./SalesAnalysis/Sales_Data/Sales_April_2019.csv")
df.head()
```

Out[5]:

	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	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

In [ ]:

```
# Read all files of directory in Python:
#https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory
```

In [6]:

```
df = pd.read_csv("./SalesAnalysis/Sales_Data/Sales_April_2019.csv")
files = [file for file in os.listdir('./SalesAnalysis/Sales_Data')]

for file in files:
   print(file)
```

```
Sales_April_2019.csv
Sales_August_2019.csv
Sales_December_2019.csv
Sales_February_2019.csv
Sales_January_2019.csv
Sales_July_2019.csv
Sales_June_2019.csv
Sales_March_2019.csv
Sales_May_2019.csv
Sales_November_2019.csv
Sales_October_2019.csv
Sales_September_2019.csv
```

In [12]:

```
#Concatenate files a single CSV:
```

In [7]:

```
df = pd.read_csv("./SalesAnalysis/Sales_Data/Sales_April_2019.csv")

files = [file for file in os.listdir('./SalesAnalysis/Sales_Data')]

all_months_data = pd.DataFrame()

for file in files:
  df = pd.read_csv("./SalesAnalysis/Sales_Data/"+file)
all_months_data = pd.concat([all_months_data,df])

all_months_data.head()
```

Out[7]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301

In [ ]:

```
#all months data in one CSV file
```

In [8]:

```
all_months_data = pd.DataFrame()

for file in files:
  df = pd.read_csv("./SalesAnalysis/Sales_Data/"+file)
  all_months_data = pd.concat([all_months_data,df])

all_months_data.to_csv("all_data.csv", index = False)
```

In [9]:

```
all_data = pd.read_csv("all_data.csv")
all_data.head()
```

Out[9]:

	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	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

Argument data with additional columns

Task 2: Add Month Column

In [10]:

```
all_data['Month']= 3
all_data.head()
```

Out[10]:

	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	3
1	NaN	NaN	NaN	NaN	NaN	NaN	3
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	3
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	3
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	3

Convert this into a string

In [11]:

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

Out[11]:

	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	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04

Clean up the data :-) Drop rows on NaN (https://stackoverflow.com/questions/43424199/display-rows-with-one-or-more-nan-values-in-pandas-dataframe)

In [14]:

```
nan_df = all_data[all_data.isna().any(axis=1)]
nan_df.head()
```

Out[14]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
356	NaN	NaN	NaN	NaN	NaN	NaN	NaN
735	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1433	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1553	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [15]:

```
all_data = all_data.dropna(how='all')
```

In [16]:

```
all_data.head()
```

Out[16]:

	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
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04

In [17]:

```
all_data['Month']= all_data['Order Date'].str[0:2]
all_data['Month']= all_data['Month'].astype('int32')
all_data.head()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-7f1c9af7181b> in <module>
      1 all_data['Month']= all_data['Order Date'].str[0:2]
----> 2 all_data['Month']= all_data['Month'].astype('int32')
      3 all_data.head()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in astype(self, dtype, copy, errors)
   5544         else:
   5545             # else, only a single dtype is given
-> 5546             new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors,)
   5547             return self._constructor(new_data).__finalize__(self, method="astype")
   5548

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in astype(self, dtype, copy,
errors)
    593         self, dtype, copy: bool = False, errors: str = "raise"
    594     ) -> "BlockManager":
--> 595         return self.apply("astype", dtype=dtype, copy=copy, errors=errors)
    596
    597     def convert(

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in apply(self, f, align_keys
, **kwargs)
    404                 applied = b.apply(f, **kwargs)
    405             else:
--> 406                 applied = getattr(b, f)(**kwargs)
    407             result_blocks = _extend_blocks(applied, result_blocks)
    408

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\blocks.py in astype(self, dtype, copy, e
rrors)
    593             vals1d = values.ravel()
    594             try:
--> 595                 values = astype_nansafe(vals1d, dtype, copy=True)
    596             except (ValueError, TypeError):
    597                 # e.g. astype_nansafe can fail on object-dtype of strings

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\dtypes\cast.py in astype_nansafe(arr, dtype, copy,
skipna)
    970         # work around NumPy brokenness, #1987
    971         if np.issubdtype(dtype.type, np.integer):
--> 972             return lib.astype_intsafe(arr.ravel(), dtype).reshape(arr.shape)
    973
    974         # if we have a datetime/timedelta array of objects

pandas\_libs\lib.pyx in pandas._libs.lib.astype_intsafe()

ValueError: invalid literal for int() with base 10: 'Or'
```

Find'Or* and delete it

In [18]:

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

Out[18]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
519	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
1149	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
1155	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
2878	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
2893	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or

it is correct! Reset it to be all_data

In [19]:

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

Task 2: Add Month Column

In [20]:

```
all_data['Month']= all_data['Order Date'].str[0:2]
all_data['Month']= all_data['Month'].astype('int32')
all_data.head()
```

Out[20]:

	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	4
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

Task 3_ add a sales column

In [21]:

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in na_arithmetic_op(left, right,
op, is_cmp)
    142     try:
--> 143         result = expressions.evaluate(op, left, right)
    144     except TypeError:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\computation\expressions.py in evaluate(op, a, b, u
se_numexpr)
    232         if use_numexpr:
--> 233             return _evaluate(op, op_str, a, b)  # type: ignore
    234     return _evaluate_standard(op, op_str, a, b)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\computation\expressions.py in _evaluate_numexpr(op
, op_str, a, b)
    118     if result is None:
--> 119         result = _evaluate_standard(op, op_str, a, b)
    120

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\computation\expressions.py in _evaluate_standard(o
p, op_str, a, b)
     67     with np.errstate(all="ignore"):
---> 68         return op(a, b)
     69

TypeError: can't multiply sequence by non-int of type 'str'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
<ipython-input-21-f37fc00a967a> in <module>
----> 1 all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']
      2 all_data.head()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\common.py in new_method(self, other)
     63         other = item_from_zerodim(other)
     64
---> 65         return method(self, other)
     66
     67     return new_method

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\__init__.py in wrapper(left, right)
    341         lvalues = extract_array(left, extract_numpy=True)
    342         rvalues = extract_array(right, extract_numpy=True)
--> 343         result = arithmetic_op(lvalues, rvalues, op)
    344
    345         return left._construct_result(result, name=res_name)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in arithmetic_op(left, right, op)
    188     else:
    189         with np.errstate(all="ignore"):
--> 190             res_values = na_arithmetic_op(lvalues, rvalues, op)
    191
    192     return res_values

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in na_arithmetic_op(left, right,
op, is_cmp)
    148             #  will handle complex numbers incorrectly, see GH#32047
    149             raise
--> 150         result = masked_arith_op(left, right, op)
    151
    152     if is_cmp and (is_scalar(result) or result is NotImplemented):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops\array_ops.py in masked_arith_op(x, y, op)
     90         if mask.any():
     91             with np.errstate(all="ignore"):
---> 92                 result[mask] = op(xrav[mask], yrav[mask])
     93
     94     else:

TypeError: can't multiply sequence by non-int of type 'str'
```

Convert columns to the correct type

In [22]:

```
all_data.info()
```

```
<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   Month             185950 non-null  int32 
dtypes: int32(1), object(6)
memory usage: 10.6+ MB
```

In [23]:

```
# Make int
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
# Make float
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
all_data.head()
```

Out[23]:

	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	4
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

In [51]:

```
all_data.info()
```

```
<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  int64  
 3   Price Each        185950 non-null  float64
 4   Order Date        185950 non-null  object 
 5   Purchase Address  185950 non-null  object 
 6   Month             185950 non-null  int32  
dtypes: float64(1), int32(1), int64(1), object(4)
memory usage: 10.6+ MB
```

The calculation is working, Well done!...==> take a look at "Sales"

In [24]:

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

Out[24]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99

Question 1: What was the best month for sales? How much was earned that month?

In [25]:

```
# here! Can you see we are grouping by month (12 months)
all_data.groupby('Month').sum()
```

Out[25]:

Month	Quantity Ordered	Price Each	Sales
1	10903	1.811768e+06	1.822257e+06
2	13449	2.188885e+06	2.202022e+06
3	17005	2.791208e+06	2.807100e+06
4	20558	3.367671e+06	3.390670e+06
5	18667	3.135125e+06	3.152607e+06
6	15253	2.562026e+06	2.577802e+06
7	16072	2.632540e+06	2.647776e+06
8	13448	2.230345e+06	2.244468e+06
9	13109	2.084992e+06	2.097560e+06
10	22703	3.715555e+06	3.736727e+06
11	19798	3.180601e+06	3.199603e+06
12	28114	4.588415e+06	4.613443e+06

In [26]:

```
# Month 12 was the best month for sales 4.613443e+06
all_data.groupby('Month').sum()['Sales']
```

Out[26]:

```
Month
1     1.822257e+06
2     2.202022e+06
3     2.807100e+06
4     3.390670e+06
5     3.152607e+06
6     2.577802e+06
7     2.647776e+06
8     2.244468e+06
9     2.097560e+06
10    3.736727e+06
11    3.199603e+06
12    4.613443e+06
Name: Sales, dtype: float64
```

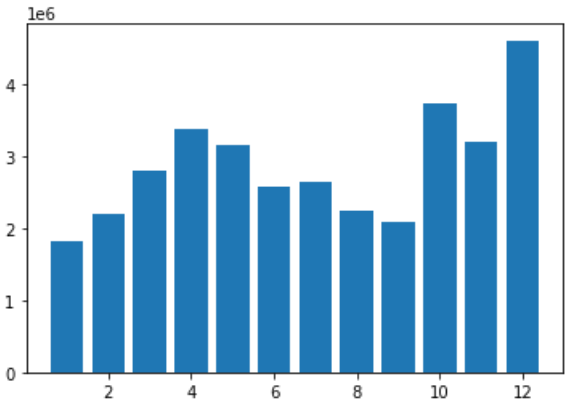
Visualization with Matplotlib

In [27]:

```
#all_data group within results variable
results = all_data.groupby('Month').sum()
```

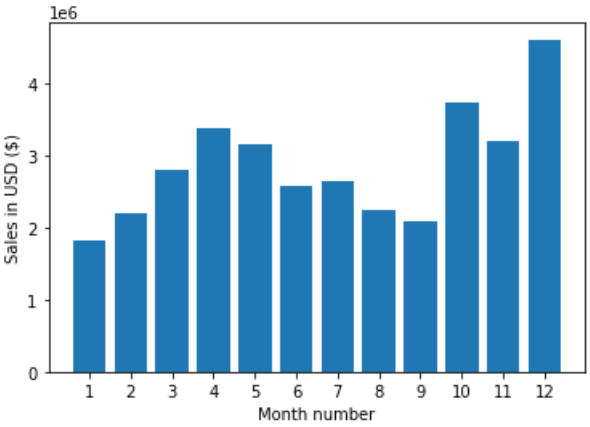
In [28]:

```
import matplotlib.pyplot as plt
months = range(1,13)

plt.bar(months, results['Sales'])
plt.show()
```



In [61]:

```
import matplotlib.pyplot as plt
months = range(1,13)

plt.bar(months, results['Sales'])
plt.xticks(months)
plt.ylabel('Sales in USD ($)')
plt.xlabel('Month number')
plt.show()
```



Question 2: What city had the highest number of sales?

Task 4 : Add a city column

In [29]:

```
all_data.head()
```

Out[29]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99

In [30]:

```
#let's use apply()
#How can we extract the city of "Purchase Order" between two commas for example:, Dallas,

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

all_data.head()
```

Out[30]:

	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
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles

Another kind of syntax would take in some sort of address value

In [31]:

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

all_data['City'] = all_data['Purchase Address'].apply(lambda x: get_city(x))

all_data.head()
```

Out[31]:

	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
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles

If we want to know the Zip code of the city, we could add others lines into the syntax

**Full syntax

In [34]:

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

#ZIP CODE

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

# First case: we concatenate the city with zip code
#all_data['Column'] = all_data['Purchase Address'].apply(lambda x: get_city(x) + ' ( '+ get_state(x) +')

# Second case: we concatenate the city with zip code
all_data['City'] = all_data['Purchase Address'].apply(lambda x:f"{get_city(x)} ( {get_state(x) })")


all_data.head()
```

Out[34]:

	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)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)

Visualization with Matplotlib

In [35]:

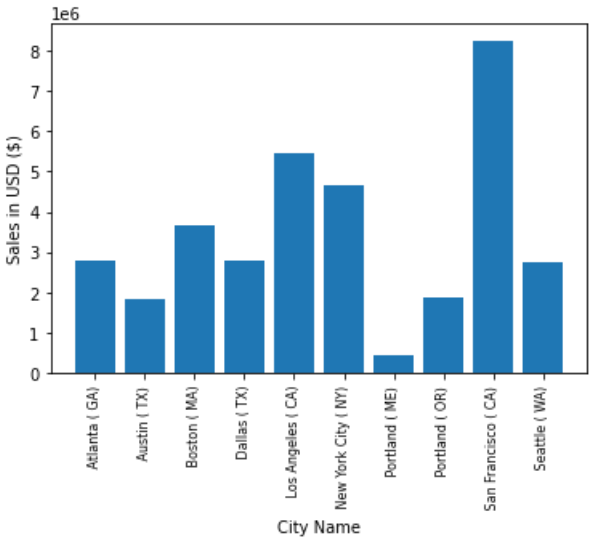
```
results = all_data.groupby('City').sum()
results
```

Out[35]:

City	Quantity Ordered	Price Each	Month	Sales
Atlanta (GA)	16602	2.779908e+06	104794	2.795499e+06
Austin (TX)	11153	1.809874e+06	69829	1.819582e+06
Boston (MA)	22528	3.637410e+06	141112	3.661642e+06
Dallas (TX)	16730	2.752628e+06	104620	2.767975e+06
Los Angeles (CA)	33289	5.421435e+06	208325	5.452571e+06
New York City (NY)	27932	4.635371e+06	175741	4.664317e+06
Portland (ME)	2750	4.471893e+05	17144	4.497583e+05
Portland (OR)	11303	1.860558e+06	70621	1.870732e+06
San Francisco (CA)	50239	8.211462e+06	315520	8.262204e+06
Seattle (WA)	16553	2.733296e+06	104941	2.747755e+06

In [39]:

```
import matplotlib.pyplot as plt

cities = [city for city,df in all_data.groupby('City')]

plt.bar(cities, results['Sales'])

#We have rotated the names of cities and formatted with a Size 8

plt.xticks(cities, rotation = 'vertical', size=8)
plt.ylabel('Sales in USD ($)')
plt.xlabel('City Name')
plt.show()
```



Question 3: What time should we diplay advertisements to maximize likelihood of custumer's buying product?

In [41]:

```
all_data.head()
#Check out the column "Order Date" we can extract the time
```

Out[41]:

	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)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	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 [ ]:

In [43]:

```
# We Converted the Order Date (object) to Order Date (datetime64)
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
```

In [45]:

```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 9 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  int64         
 3   Price Each        185950 non-null  float64       
 4   Order Date        185950 non-null  datetime64[ns]
 5   Purchase Address  185950 non-null  object        
 6   Month             185950 non-null  int32         
 7   Sales             185950 non-null  float64       
 8   City              185950 non-null  object        
dtypes: datetime64[ns](1), float64(2), int32(1), int64(1), object(4)
memory usage: 13.5+ MB
```

Well done !! Check out the columns hour and minute

In [47]:

```
all_data['Hour'] = all_data['Order Date'].dt.hour
```

```
all_data['Minute'] = all_data['Order Date'].dt.minute
all_data.head()
```

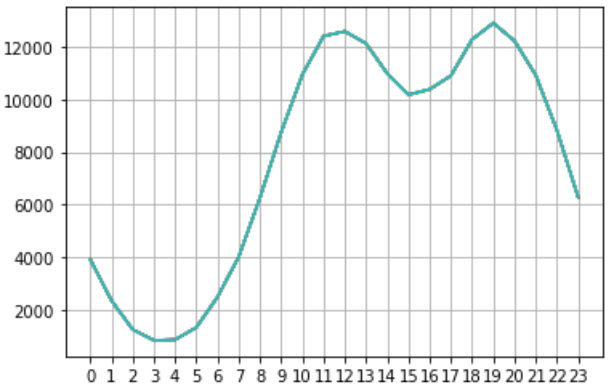
Out[47]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
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	46
2	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	30
3	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	38
4	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	38
5	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	27

Visualization

In [53]:

```
hours = [hour for hour,df in all_data.groupby('Hour')]

plt.plot(hours, all_data.groupby(['Hour']).count())

all_data.groupby(['Hour']).count()
plt.xticks(hours)
plt.xlabel('Hour')
plt.ylabel('Number of Orders')
plt.grid()

plt.show()

# My recommnendation is around 12pm (11) or 7pm (19)
```



Question 4: What products are most often sold together?

In [59]:

```
all_data.head()
# check out the ID 176560 is in two Product
```

Out[59]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
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	46
2	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	30
3	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	38
4	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	38
5	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	27

In [63]:

```
#This syntax can shows you the products are duplicated in Order ID

df = all_data[all_data['Order ID'].duplicated(keep=False)]
df.head(20)
```

Out[63]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
3	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	38
4	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	38
18	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	42
19	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	42
30	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	31
31	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	31
32	176586	AAA Batteries (4-pack)	2	2.99	2019-04-10 17:00:00	365 Center St, San Francisco, CA 94016	4	5.98	San Francisco (CA)	17	0
33	176586	Google Phone	1	600.00	2019-04-10 17:00:00	365 Center St, San Francisco, CA 94016	4	600.00	San Francisco (CA)	17	0
119	176672	Lightning Charging Cable	1	14.95	2019-04-12 11:07:00	778 Maple St, New York City, NY 10001	4	14.95	New York City (NY)	11	7
120	176672	USB-C Charging Cable	1	11.95	2019-04-12 11:07:00	778 Maple St, New York City, NY 10001	4	11.95	New York City (NY)	11	7
129	176681	Apple Airpods Headphones	1	150.00	2019-04-20 10:39:00	331 Cherry St, Seattle, WA 98101	4	150.00	Seattle (WA)	10	39
130	176681	ThinkPad Laptop	1	999.99	2019-04-20 10:39:00	331 Cherry St, Seattle, WA 98101	4	999.99	Seattle (WA)	10	39
138	176689	Bose SoundSport Headphones	1	99.99	2019-04-24 17:15:00	659 Lincoln St, New York City, NY 10001	4	99.99	New York City (NY)	17	15
139	176689	AAA Batteries (4-pack)	2	2.99	2019-04-24 17:15:00	659 Lincoln St, New York City, NY 10001	4	5.98	New York City (NY)	17	15
189	176739	34in Ultrawide Monitor	1	379.99	2019-04-05 17:38:00	730 6th St, Austin, TX 73301	4	379.99	Austin (TX)	17	38
190	176739	Google Phone	1	600.00	2019-04-05 17:38:00	730 6th St, Austin, TX 73301	4	600.00	Austin (TX)	17	38
225	176774	Lightning Charging Cable	1	14.95	2019-04-25 15:06:00	372 Church St, Los Angeles, CA 90001	4	14.95	Los Angeles (CA)	15	6
226	176774	USB-C Charging Cable	1	11.95	2019-04-25 15:06:00	372 Church St, Los Angeles, CA 90001	4	11.95	Los Angeles (CA)	15	6
233	176781	iPhone	1	700.00	2019-04-03 07:37:00	976 Hickory St, Dallas, TX 75001	4	700.00	Dallas (TX)	7	37
234	176781	Lightning Charging Cable	1	14.95	2019-04-03 07:37:00	976 Hickory St, Dallas, TX 75001	4	14.95	Dallas (TX)	7	37

In [64]:

```
# The idea es put the products togheter in one ID Order (group them together)

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

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

df.head()

# In this syntax we have groups togheter of products, but are duplications of groups :-(
```

```
<ipython-input-64-618a97b89bfd>:5: 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/indexing.html#returning-a-view-versus-a-copy
  df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
```

Out[64]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute	Grouped
3	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	38	Google Phone,Wired Headphones
4	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	38	Google Phone,Wired Headphones
18	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	42	Google Phone,USB-C Charging Cable
19	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	42	Google Phone,USB-C Charging Cable
30	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	31	Bose SoundSport Headphones,Bose SoundSport Hea...

In [65]:

```
# Cool, this had worked, Well done!!
# However The next syntax we need to count the products :-)


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(100)
```

```
<ipython-input-65-255388792c19>: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/indexing.html#returning-a-view-versus-a-copy
  df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x))
```

Out[65]:

	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
...	...	...
2662	179108	Lightning Charging Cable,AAA Batteries (4-pack)
2683	179128	iPhone,Apple Airpods Headphones
2718	179162	Google Phone,USB-C Charging Cable
2783	179226	34in Ultrawide Monitor,Macbook Pro Laptop
2829	179270	iPhone,Lightning Charging Cable

100 rows × 2 columns

In [70]:

```
#Referenced: https://stackoverflow.com/questions/52195887/counting-unique-pairs-of-numbers-into-a-python
```

```
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)))

print(count)


#  you can see all of products, but the syntax doesn't work correctly :-(
```

```
Counter({('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, ('Lightning Charging
Cable', 'Apple Airpods Headphones'): 81, ('Vareebadd Phone', 'Bose SoundSport Headphones'): 80, ('USB-C
Charging Cable', 'Bose SoundSport Headphones'): 77, ('Apple Airpods Headphones', 'Wired Headphones'): 69,
('Lightning Charging Cable', 'USB-C Charging Cable'): 58, ('Lightning Charging Cable', 'AA Batteries (4-
pack)'): 55, ('Lightning Charging Cable', 'Lightning Charging Cable'): 54, ('Bose SoundSport
Headphones', 'Wired Headphones'): 53, ('AA Batteries (4-pack)', 'Lightning Charging Cable'): 51, ('AAA B
atteries (4-pack)', 'USB-C Charging Cable'): 50, ('Apple Airpods Headphones', 'AAA Batteries (4-pack)'):
48, ('AA Batteries (4-pack)', 'AAA Batteries (4-pack)'): 48, ('USB-C Charging Cable', 'USB-C Charging Ca
ble'): 48, ('AAA Batteries (4-pack)', 'AAA Batteries (4-pack)'): 48, ('USB-C Charging Cable', 'AAA
Batteries (4-pack)'): 45, ('Wired Headphones', 'USB-C Charging Cable'): 45, ('AA Batteries (4-pack)', 'W
ired Headphones'): 44, ('AAA Batteries (4-pack)', 'Lightning Charging Cable'): 44, ('AAA Batteries (4-pa
ck)', 'Wired Headphones'): 43, ('Wired Headphones', 'AAA Batteries (4-pack)'): 43, ('USB-C Charging
Cable', 'Lightning Charging Cable'): 42, ('AA Batteries (4-pack)', 'Apple Airpods Headphones'): 41, ('AA
A Batteries (4-pack)', 'AA Batteries (4-pack)'): 39, ('Wired Headphones', 'AA Batteries (4-pack)'): 39,
('Lightning Charging Cable', 'Bose SoundSport Headphones'): 39, ('USB-C Charging Cable', 'AA Batteries
(4-pack)'): 38, ('Bose SoundSport Headphones', 'AAA Batteries (4-pack)'): 37, ('AA Batteries (4-pack)',
'USB-C Charging Cable'): 37, ('Wired Headphones', 'Lightning Charging Cable'): 37, ('Lightning Charging
Cable', 'AAA Batteries (4-pack)'): 36, ('Apple Airpods Headphones', 'Lightning Charging Cable'): 35, ('W
ired Headphones', 'Wired Headphones'): 35, ('AA Batteries (4-pack)', 'AA Batteries (4-pack)'): 35, ('USB
-C Charging Cable', 'Apple Airpods Headphones'): 34, ('Bose SoundSport Headphones', 'Lightning Charging
Cable'): 33, ('AAA Batteries (4-pack)', 'Apple Airpods Headphones'): 33, ('Apple Airpods Headphones',
'Bose SoundSport Headphones'): 32, ('Wired Headphones', 'Apple Airpods Headphones'): 31, ('USB-C
Charging Cable', '27in FHD Monitor'): 31, ('Apple Airpods Headphones', 'USB-C Charging Cable'): 29,
('Apple Airpods Headphones', 'AA Batteries (4-pack)'): 29, ('AA Batteries (4-pack)', 'Bose SoundSport He
adphones'): 28, ('Bose SoundSport Headphones', 'Bose SoundSport Headphones'): 27, ('Bose SoundSport Headp
hones', 'AA Batteries (4-pack)'): 27, ('Bose SoundSport Headphones', 'USB-C Charging Cable'): 25,
('Apple Airpods Headphones', 'Apple Airpods Headphones'): 24, ('AAA Batteries (4-pack)', '27in FHD
Monitor'): 22, ('27in FHD Monitor', 'AAA Batteries (4-pack)'): 21, ('Wired Headphones', 'Bose SoundSport
Headphones'): 21, ('AAA Batteries (4-pack)', 'Bose SoundSport Headphones'): 20, ('34in Ultrawide
Monitor', 'AA Batteries (4-pack)'): 19, ('Lightning Charging Cable', '27in 4K Gaming Monitor'): 18, ('AA
Batteries (4-pack)', 'iPhone'): 18, ('27in FHD Monitor', 'Lightning Charging Cable'): 18, ('Lightning Ch
arging Cable', '27in FHD Monitor'): 18, ('34in Ultrawide Monitor', 'Lightning Charging Cable'): 18,
('Wired Headphones', '27in 4K Gaming Monitor'): 18, ('Bose SoundSport Headphones', 'Apple Airpods
Headphones'): 18, ('iPhone', 'AAA Batteries (4-pack)'): 17, ('Wired Headphones', '34in Ultrawide
Monitor'): 17, ('ThinkPad Laptop', 'AAA Batteries (4-pack)'): 16, ('Lightning Charging Cable', 'Google P
hone'): 16, ('27in 4K Gaming Monitor', 'Lightning Charging Cable'): 16, ('34in Ultrawide Monitor', 'USB-
C Charging Cable'): 15, ('27in FHD Monitor', 'AA Batteries (4-pack)'): 15, ('Wired Headphones',
'iPhone'): 15, ('AAA Batteries (4-pack)', '27in 4K Gaming Monitor'): 15, ('iPhone', 'USB-C Charging
Cable'): 15, ('20in Monitor', 'USB-C Charging Cable'): 15, ('Lightning Charging Cable', '20in Monitor'):
15, ('27in 4K Gaming Monitor', 'AAA Batteries (4-pack)'): 15, ('Lightning Charging Cable', '34in
Ultrawide Monitor'): 15, ('Google Phone', 'AA Batteries (4-pack)'): 14, ('Apple Airpods Headphones', 'Go
ogle Phone'): 14, ('USB-C Charging Cable', 'iPhone'): 14, ('Bose SoundSport Headphones', '27in FHD
Monitor'): 14, ('AA Batteries (4-pack)', '27in 4K Gaming Monitor'): 14, ('AAA Batteries (4-pack)',
'iPhone'): 14, ('iPhone', 'AA Batteries (4-pack)'): 14, ('AA Batteries (4-pack)', 'Flatscreen TV'): 13,
('AA Batteries (4-pack)', '34in Ultrawide Monitor'): 13, ('AAA Batteries (4-pack)', '34in Ultrawide
Monitor'): 13, ('Apple Airpods Headphones', 'iPhone'): 13, ('Wired Headphones', 'Macbook Pro Laptop'): 1
3, ('Apple Airpods Headphones', '27in 4K Gaming Monitor'): 12, ('Apple Airpods Headphones', '27in FHD
Monitor'): 12, ('27in FHD Monitor', 'Bose SoundSport Headphones'): 12, ('27in FHD Monitor', 'USB-C
Charging Cable'): 12, ('Google Phone', 'Lightning Charging Cable'): 12, ('Apple Airpods Headphones', 'Ma
cbook Pro Laptop'): 12, ('27in 4K Gaming Monitor', 'USB-C Charging Cable'): 12, ('Macbook Pro Laptop', '
USB-C Charging Cable'): 12, ('Wired Headphones', '27in FHD Monitor'): 12, ('20in Monitor', 'Wired
Headphones'): 12, ('Lightning Charging Cable', 'Flatscreen TV'): 12, ('27in FHD Monitor', 'Apple Airpods
Headphones'): 12, ('USB-C Charging Cable', 'Google Phone'): 12, ('27in 4K Gaming Monitor', 'AA Batteries
(4-pack)'): 12, ('34in Ultrawide Monitor', 'AAA Batteries (4-pack)'): 12, ('AAA Batteries (4-pack)', 'Go
ogle Phone'): 11, ('AAA Batteries (4-pack)', 'Macbook Pro Laptop'): 11, ('USB-C Charging Cable', '27in 4
K Gaming Monitor'): 11, ('USB-C Charging Cable', 'ThinkPad Laptop'): 11, ('34in Ultrawide Monitor',
'Wired Headphones'): 11, ('20in Monitor', 'Lightning Charging Cable'): 11, ('AA Batteries (4-pack)', '27
in FHD Monitor'): 11, ('Bose SoundSport Headphones', '34in Ultrawide Monitor'): 11, ('ThinkPad Laptop',
```

in FHD Monitor'): 11, ('Bose SoundSport Headphones', '34in Ultrawide Monitor'): 11, ('ThinkPad Laptop', 'Lightning Charging Cable'): 11, ('Google Phone', 'AAA Batteries (4-pack)'): 11, ('USB-C Charging Cable', '34in Ultrawide Monitor'): 11, ('Macbook Pro Laptop', 'Lightning Charging Cable'): 11, ('AA Batteries (4-pack)', 'Google Phone'): 11, ('AAA Batteries (4-pack)', 'ThinkPad Laptop'): 11, ('Macbook Pro Laptop', 'Bose SoundSport Headphones'): 11, ('27in 4K Gaming Monitor', 'Wired Headphones'): 11, ('Flatscreen TV', 'AAA Batteries (4-pack)'): 11, ('Flatscreen TV', 'Lightning Charging Cable'): 10, ('Wired Headphones', 'ThinkPad Laptop'): 10, ('USB-C Charging Cable', '20in Monitor'): 10, ('27in 4K Gaming Monitor', 'Apple Airpods Headphones'): 10, ('USB-C Charging Cable', 'Flatscreen TV'): 10, ('27in FHD Monitor', 'Wired Headphones'): 10, ('AA Batteries (4-pack)', '20in Monitor'): 10, ('AAA Batteries (4-pack)', 'Flatscreen TV'): 10, ('Lightning Charging Cable', 'iPhone'): 10, ('Bose SoundSport Headphones', 'Flatscreen TV'): 10, ('Lightning Charging Cable', 'Macbook Pro Laptop'): 10, ('Bose SoundSport Headphones', '27in 4K Gaming Monitor'): 10, ('Apple Airpods Headphones', 'ThinkPad Laptop'): 9, ('Wired Headphones', 'Google Phone'): 9, ('27in 4K Gaming Monitor', 'Bose SoundSport Headphones'): 9, ('20in Monitor', 'Bose SoundSport Headphones'): 9, ('Macbook Pro Laptop', 'AA Batteries (4-pack)'): 9, ('ThinkPad Laptop', 'USB-C Charging Cable'): 9, ('ThinkPad Laptop', 'Bose SoundSport Headphones'): 9, ('Vareebadd Phone', 'AA Batteries (4-pack)'): 9, ('USB-C Charging Cable', 'Macbook Pro Laptop'): 9, ('27in FHD Monitor', '27in FHD Monitor'): 9, ('AA Batteries (4-pack)', 'ThinkPad Laptop'): 9, ('Lightning Charging Cable', 'ThinkPad Laptop'): 9, ('AA Batteries (4-pack)', 'Macbook Pro Laptop'): 8, ('Flatscreen TV', 'AA Batteries (4-pack)'): 8, ('Apple Airpods Headphones', 'Flatscreen TV'): 8, ('ThinkPad Laptop', 'AA Batteries (4-pack)'): 8, ('AAA Batteries (4-pack)', '20in Monitor'): 8, ('34in Ultrawide Monitor', 'Apple Airpods Headphones'): 8, ('Bose SoundSport Headphones', 'Google Phone'): 8, ('20in Monitor', 'Apple Airpods Headphones'): 7, ('Macbook Pro Laptop', 'Apple Airpods Headphones'): 7, ('Wired Headphones', 'Flatscreen TV'): 7, ('Wired Headphones', '20in Monitor'): 7, ('Macbook Pro Laptop', 'Wired Headphones'): 7, ('USB-C Charging Cable', 'Vareebadd Phone'): 7, ('Google Phone', '27in FHD Monitor'): 7, ('Macbook Pro Laptop', 'AAA Batteries (4-pack)'): 7, ('34in Ultrawide Monitor', 'iPhone'): 7, ('34in Ultrawide Monitor', '34in Ultrawide Monitor'): 7, ('Flatscreen TV', 'USB-C Charging Cable'): 7, ('Bose SoundSport Headphones', 'iPhone'): 7, ('ThinkPad Laptop', 'Apple Airpods Headphones'): 7, ('Google Phone', 'Apple Airpods Headphones'): 7, ('Macbook Pro Laptop', '27in 4K Gaming Monitor'): 7, ('iPhone', '27in 4K Gaming Monitor'): 6, ('Flatscreen TV', 'Flatscreen TV'): 6, ('Apple Airpods Headphones', '34in Ultrawide Monitor'): 6, ('iPhone', '34in Ultrawide Monitor'): 6, ('Vareebadd Phone', 'Apple Airpods Headphones'): 6, ('27in 4K Gaming Monitor', '34in Ultrawide Monitor'): 6, ('27in 4K Gaming Monitor', 'Macbook Pro Laptop'): 6, ('Bose SoundSport Headphones', '20in Monitor'): 6, ('iPhone', 'Flatscreen TV'): 6, ('Apple Airpods Headphones', '20in Monitor'): 6, ('Apple Airpods Headphones', 'Vareebadd Phone'): 6, ('Wired Headphones', 'Vareebadd Phone'): 6, ('34in Ultrawide Monitor', 'Bose SoundSport Headphones'): 6, ('Google Phone', 'iPhone'): 6, ('27in FHD Monitor', 'Macbook Pro Laptop'): 6, ('20in Monitor', 'AA Batteries (4-pack)'): 6, ('iPhone', 'Bose SoundSport Headphones'): 5, ('27in 4K Gaming Monitor', '27in 4K Gaming Monitor'): 5, ('Flatscreen TV', '34in Ultrawide Monitor'): 5, ('27in 4K Gaming Monitor', 'Google Phone'): 5, ('27in FHD Monitor', '34in Ultrawide Monitor'): 5, ('Flatscreen TV', 'Apple Airpods Headphones'): 5, ('34in Ultrawide Monitor', '27in FHD Monitor'): 5, ('Macbook Pro Laptop', '34in Ultrawide Monitor'): 4, ('iPhone', 'Vareebadd Phone'): 4, ('Bose SoundSport Headphones', 'ThinkPad Laptop'): 4, ('20in Monitor', 'Macbook Pro Laptop'): 4, ('Vareebadd Phone', '34in Ultrawide Monitor'): 4, ('Flatscreen TV', 'Wired Headphones'): 4, ('Flatscreen TV', '27in FHD Monitor'): 4, ('LG Dryer', 'AA Batteries (4-pack)'): 4, ('Flatscreen TV', 'Macbook Pro Laptop'): 4, ('27in FHD Monitor', '27in 4K Gaming Monitor'): 4, ('ThinkPad Laptop', 'Flatscreen TV'): 4, ('Flatscreen TV', 'iPhone'): 4, ('27in 4K Gaming Monitor', 'ThinkPad Laptop'): 4, ('Vareebadd Phone', 'Google Phone'): 4, ('Macbook Pro Laptop', 'Google Phone'): 4, ('27in 4K Gaming Monitor', '27in FHD Monitor'): 4, ('Lightning Charging Cable', 'LG Washing Machine'): 4, ('27in FHD Monitor', 'ThinkPad Laptop'): 4, ('ThinkPad Laptop', 'Wired Headphones'): 4, ('iPhone', 'ThinkPad Laptop'): 4, ('Bose SoundSport Headphones', 'Macbook Pro Laptop'): 4, ('AAA Batteries (4-pack)', 'Vareebadd Phone'): 4, ('LG Washing Machine', 'AAA Batteries (4-pack)'): 4, ('Macbook Pro Laptop', 'ThinkPad Laptop'): 3, ('ThinkPad Laptop', 'Google Phone'): 3, ('34in Ultrawide Monitor', 'Macbook Pro Laptop'): 3, ('Lightning Charging Cable', 'Vareebadd Phone'): 3, ('Google Phone', 'ThinkPad Laptop'): 3, ('20in Monitor', '20in Monitor'): 3, ('ThinkPad Laptop', 'iPhone'): 3, ('Vareebadd Phone', 'Flatscreen TV'): 3, ('34in Ultrawide Monitor', 'Flatscreen TV'): 3, ('Macbook Pro Laptop', 'Macbook Pro Laptop'): 3, ('34in Ultrawide Monitor', 'ThinkPad Laptop'): 3, ('Macbook Pro Laptop', 'iPhone'): 3, ('Vareebadd Phone', 'iPhone'): 3, ('Wired Headphones', 'LG Washing Machine'): 3, ('Google Phone', '34in Ultrawide Monitor'): 3, ('Macbook Pro Laptop', '27in FHD Monitor'): 3, ('Flatscreen TV', 'Bose SoundSport Headphones'): 3, ('AA Batteries (4-pack)', 'Vareebadd Phone'): 3, ('27in FHD Monitor', '20in Monitor'): 3, ('iPhone', 'Google Phone'): 3, ('27in 4K Gaming Monitor', 'iPhone'): 3, ('Google Phone', 'Google Phone'): 3, ('Flatscreen TV', 'Google Phone'): 3, ('Google Phone', 'Macbook Pro Laptop'): 3, ('27in 4K Gaming Monitor', 'Flatscreen TV'): 3, ('Apple Airpods Headphones', 'LG Dryer'): 3, ('20in Monitor', 'AAA Batteries (4-pack)'): 3, ('iPhone', 'Macbook Pro Laptop'): 3, ('34in Ultrawide Monitor', 'Google Phone'): 2, ('Macbook Pro Laptop', '20in Monitor'): 2, ('Lightning Charging Cable', 'LG Dryer'): 2, ('Flatscreen TV', '27in 4K Gaming Monitor'): 2, ('ThinkPad Laptop', 'Macbook Pro Laptop'): 2, ('Macbook Pro Laptop', 'LG Washing Machine'): 2, ('20in Monitor', '27in FHD Monitor'): 2, ('ThinkPad Laptop', 'ThinkPad Laptop'): 2, ('Bose SoundSport Headphones', 'Vareebadd Phone'): 2, ('Vareebadd Phone', 'ThinkPad Laptop'): 2, ('20in Monitor', 'ThinkPad Laptop'): 2, ('iPhone', 'iPhone'): 2, ('27in FHD Monitor', 'LG Dryer'): 2, ('Vareebadd Phone', '27in 4K Gaming Monitor'): 2, ('27in 4K Gaming Monitor', '20in Monitor'): 2, ('LG Washing Machine', 'Lightning Charging Cable'): 2, ('LG Washing Machine', 'Bose SoundSport Headphones'): 2, ('AA Batteries (4-pack)', 'LG Dryer'): 2, ('Vareebadd Phone', 'AAA Batteries (4-pack)'): 2, ('iPhone', '20in Monitor'): 2, ('20in Monitor', 'Google Phone'): 2, ('Flatscreen TV', 'ThinkPad Laptop'): 2, ('ThinkPad Laptop', '27in FHD Monitor'): 2, ('27in FHD Monitor', 'Flatscreen TV'): 2, ('Google Phone', '20in Monitor'): 2, ('27in 4K Gaming Monitor', 'Vareebadd Phone'): 1, ('27in FHD Monitor', 'iPhone'): 1, ('Vareebadd Phone', 'Lightning Charging Cable'): 1, ('Google Phone', 'Vareebadd Phone'): 1, ('20in Monitor', 'iPhone'): 1, ('LG Dryer', 'Vareebadd Phone'): 1, ('Macbook Pro Laptop', 'Flatscreen TV'): 1, ('ThinkPad Laptop', 'Vareebadd Phone'): 1, ('Google Phone', 'Flatscreen TV'): 1, ('LG Washing Machine', 'Google Phone'): 1, ('LG Washing Machine', 'Wired Headphones'): 1, ('LG Dryer', 'Flatscreen TV'): 1, ('27in FHD Monitor',

```
'LG Washing Machine'): 1, ('LG Dryer', '27in FHD Monitor'): 1, ('20in Monitor', '34in Ultrawide Monitor')
: 1, ('34in Ultrawide Monitor', '20in Monitor'): 1, ('34in Ultrawide Monitor', 'LG Washing Machine'): 1,
('Google Phone', '27in 4K Gaming Monitor'): 1, ('LG Washing Machine', 'iPhone'): 1, ('LG Dryer', 'Wired H
eadphones'): 1, ('27in FHD Monitor', 'Vareebadd Phone'): 1, ('LG Washing Machine', '27in 4K Gaming
Monitor'): 1, ('LG Washing Machine', 'Apple Airpods Headphones'): 1, ('27in 4K Gaming Monitor', 'LG
Dryer'): 1, ('20in Monitor', 'LG Washing Machine'): 1, ('LG Dryer', 'Google Phone'): 1, ('Vareebadd
Phone', '27in FHD Monitor'): 1, ('ThinkPad Laptop', '27in 4K Gaming Monitor'): 1, ('20in Monitor',
'Flatscreen TV'): 1, ('USB-C Charging Cable', 'LG Dryer'): 1, ('LG Washing Machine', '20in Monitor'): 1,
('Flatscreen TV', '20in Monitor'): 1, ('27in FHD Monitor', 'Google Phone'): 1, ('iPhone', '27in FHD
Monitor'): 1, ('LG Dryer', 'AAA Batteries (4-pack)'): 1, ('ThinkPad Laptop', '34in Ultrawide Monitor'):
1, ('iPhone', 'LG Washing Machine'): 1, ('AAA Batteries (4-pack)', 'LG Dryer'): 1, ('LG Dryer', '27in 4K
Gaming Monitor'): 1, ('LG Dryer', 'Lightning Charging Cable'): 1, ('ThinkPad Laptop', 'LG Dryer'): 1,
('LG Washing Machine', 'AA Batteries (4-pack)'): 1})
```

In [71]:

```
count = Counter()


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


# you can see the most common 10 Products
count.most_common(10)
```

Out[71]:

```
[(('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)]
```

10 products the most sold togheter

In [73]:

```
count = Counter()


for row in df['Grouped']:
    row_list = row.split(',')

# I changed the combination line (3) you can see better it
    count.update(Counter(combinations(row_list, 3)))

# 10 products the most sold togheter
for key, value in count.most_common(10):
    print( key, value)
```

```
('Google Phone', 'USB-C Charging Cable', 'Wired Headphones') 87
('iPhone', 'Lightning Charging Cable', 'Wired Headphones') 62
('iPhone', 'Lightning Charging Cable', 'Apple Airpods Headphones') 47
('Google Phone', 'USB-C Charging Cable', 'Bose SoundSport Headphones') 35
('Vareebadd Phone', 'USB-C Charging Cable', 'Wired Headphones') 33
('iPhone', 'Apple Airpods Headphones', 'Wired Headphones') 27
('Google Phone', 'Bose SoundSport Headphones', 'Wired Headphones') 24
('Vareebadd Phone', 'USB-C Charging Cable', 'Bose SoundSport Headphones') 16
('USB-C Charging Cable', 'Bose SoundSport Headphones', 'Wired Headphones') 5
('Vareebadd Phone', 'Bose SoundSport Headphones', 'Wired Headphones') 5
```

Question 5: What product sold the most? Why do you think it sold the most?

In [74]:

```
all_data.head()
```

Out[74]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
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	46
2	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	30
3	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	38
4	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	38
5	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	27

In [75]:

```
product_group = all_data.groupby('Product')

print(product_group)
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000023023F85C40>
```

In [76]:

```
product_group = all_data.groupby('Product')

product_group.sum()
```

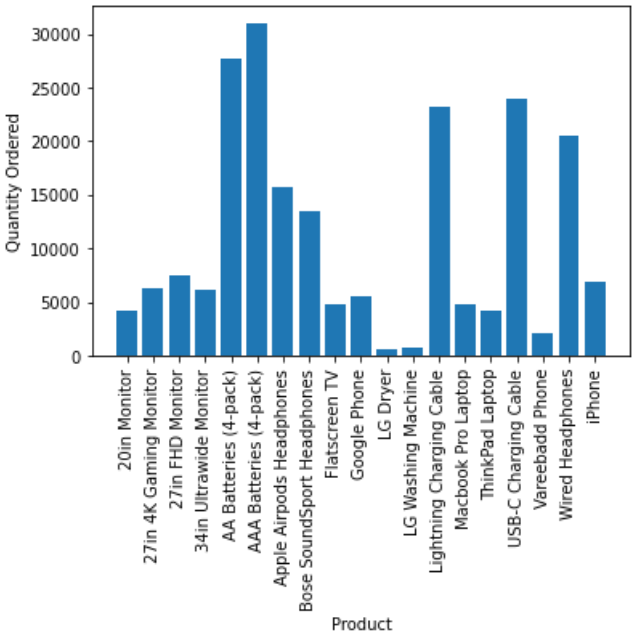
Out[76]:

Product	Quantity Ordered	Price Each	Month	Sales	Hour	Minute
20in Monitor	4129	451068.99	29336	454148.71	58764	122252
27in 4K Gaming Monitor	6244	2429637.70	44440	2435097.56	90916	184331
27in FHD Monitor	7550	1125974.93	52558	1132424.50	107540	219948
34in Ultrawide Monitor	6199	2348718.19	43304	2355558.01	89076	183480
AA Batteries (4-pack)	27635	79015.68	145558	106118.40	298342	609039
AAA Batteries (4-pack)	31017	61716.59	146370	92740.83	297332	612113
Apple Airpods Headphones	15661	2332350.00	109477	2349150.00	223304	455570
Bose SoundSport Headphones	13457	1332366.75	94113	1345565.43	192445	392603
Flatscreen TV	4819	1440000.00	34224	1445700.00	68815	142789
Google Phone	5532	3315000.00	38305	3319200.00	79479	162773
LG Dryer	646	387600.00	4383	387600.00	9326	19043
LG Washing Machine	666	399600.00	4523	399600.00	9785	19462
Lightning Charging Cable	23217	323787.10	153092	347094.15	312529	634442
Macbook Pro Laptop	4728	8030800.00	33548	8037600.00	68261	137574
ThinkPad Laptop	4130	4127958.72	28950	4129958.70	59746	121508
USB-C Charging Cable	23975	261740.85	154819	286501.25	314645	647586
Vareebadd Phone	2068	826000.00	14309	827200.00	29472	61835
Wired Headphones	20557	226395.18	133397	246478.43	271720	554023
iPhone	6849	4789400.00	47941	4794300.00	98657	201688

In [82]:

```
product_group = all_data.groupby('Product')

quantity_ordered = product_group.sum()['Quantity Ordered']

products = [product for product,df in product_group]

plt.bar(products, quantity_ordered)
```

```
plt.xticks(products, rotation = 'vertical', size=10)
plt.ylabel('Quantity Ordered')
plt.xlabel('Product')
plt.show()
```



In [83]:

```
all_data.head()
```

Out[83]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	Hour	Minute
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	46
2	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	30
3	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	38
4	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	38
5	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	27

In [84]:

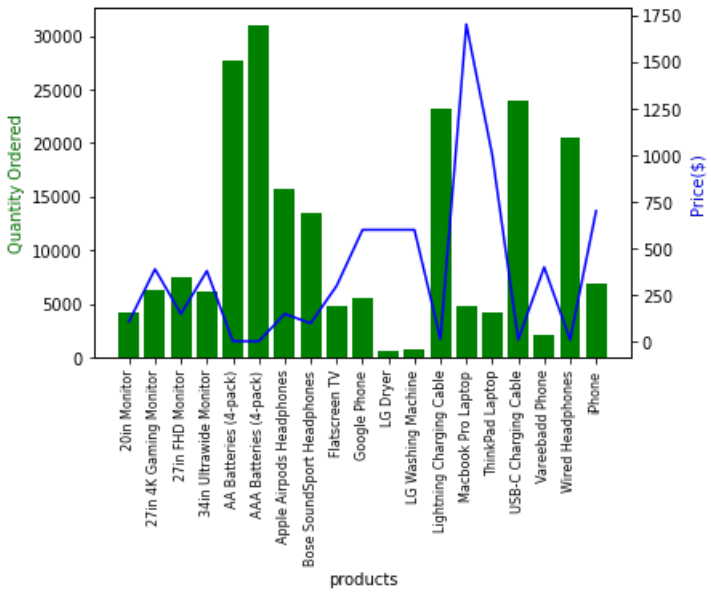
```
# Prices of each Products

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
```

In [88]:

```
# Referenced : https://stackoverflow.com/questions/14762181/adding-a-y-axis-label-to-secondary-y-axis-in
# The way is to interact with the axes object directly.
# Where X = Product, y2 = prices

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


fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.bar(products, quantity_ordered, color='g')
ax2.plot(products, prices, 'b-')

ax1.set_xlabel('products')
ax1.set_ylabel('Quantity Ordered', color='g')
ax2.set_ylabel('Price($)', color='b')

ax1.set_xticklabels(products, rotation = 'vertical', size=8)
plt.show()
```

```
<ipython-input-88-ef9df438dc82>:17: UserWarning: FixedFormatter should only be used together with
FixedLocator
  ax1.set_xticklabels(products, rotation = 'vertical', size=8)
```



In [ ]: