Sales Products Analysis

August 2, 2023

The sales product data between 2019-2020. Dataset source: Kaggle, username: KNIGHT-BEARR, dataset name: Sales Product Data

Questions:

- What was the best month for sales? How much was earned that month?
- What City had the highest number of sales?
- What time should we display adverstisement to maximize likelihood of customer's buying product?
- What products are most often sold together?
- What product sold the most? Why do you think it sold the most?

Key finding:

- Total number of sales by month, hour
- Total number of ordered quantity by city
- Total number of ordered quantity by products

[1]:
```
import pandas as pd
```

Merge the 12 months of sales data into a single CSV file

[2]:
```
# read single .csv file
df = pd.read_csv("D:/jupyter_directory/Sales_Data/Sales_January_2019.csv")
df.head()
```

[2]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	141234	iPhone	1	700	01/22/19 21:25	944 Walnut St, Boston, MA 02215
1	141235	Lightning Charging Cable	1	14.95	01/28/19 14:15	185 Maple St, Portland, OR 97035
2	141236	Wired Headphones	2	11.99	01/17/19 13:33	538 Adams St, San Francisco, CA 94016
3	141237	27in FHD Monitor	1	149.99	01/05/19 20:33	738 10th St, Los Angeles, CA 90001
4	141238	Wired Headphones	1	11.99		

```
4  01/25/19 11:59          387 10th St, Austin, TX 73301
```

[3]:
```
# create file list
import glob
file_list = glob.glob('D:/jupyter_directory/Sales_Data/*.csv')

# Read all CSV files and store them in a list comprehension
dataframes = [pd.read_csv(file) for file in file_list]

# Concatenate the list of DataFrames into a single DataFrame
df = pd.concat(dataframes, ignore_index=True)
df
```

[3]:

	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
…	…	…	…	…	…	…
186845	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
186846	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
186847	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
186848	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
186849	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

[186850 rows x 6 columns]

[4]:
```
df.to_csv("all_sales_data.csv", index=False)
```

[5]:
```
# Read in updated dataframe
df = pd.read_csv("all_sales_data.csv")
df.tail()
```

[5]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
186845	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
186846	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
186847	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
186848	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
186849	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

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

Clean data

[6]:
```
# cheack missing values
df.isna().sum()
```

[6]:
```
Order ID            545
Product             545
Quantity Ordered    545
Price Each          545
Order Date          545
Purchase Address    545
dtype: int64
```

[7]:
```
# remove rows of missing values
df = df.dropna(how = 'all')
```

[8]:
```
# filter out text data that not related
print(df['Quantity Ordered'].unique())
print(df['Price Each'].unique())
```

```
['2' '1' '3' '5' 'Quantity Ordered' '4' '7' '6' '8' '9']
['11.95' '99.99' '600' '11.99' '1700' '14.95' '389.99' '3.84' '150' '2.99'
 '700' '300' '149.99' '109.99' '600.0' '999.99' '400' '379.99'
 'Price Each' '700.0' '1700.0' '150.0' '300.0' '400.0']
```

[9]:
```
#Filter out text data that not related
```

```
df[df['Quantity Ordered'] == 'Quantity Ordered']
```

[9]:
```
        Order ID  Product  Quantity Ordered  Price Each  Order Date  \
519     Order ID  Product  Quantity Ordered  Price Each  Order Date
1149    Order ID  Product  Quantity Ordered  Price Each  Order Date
1155    Order ID  Product  Quantity Ordered  Price Each  Order Date
2878    Order ID  Product  Quantity Ordered  Price Each  Order Date
2893    Order ID  Product  Quantity Ordered  Price Each  Order Date
…          …      …                …          …          …
185164  Order ID  Product  Quantity Ordered  Price Each  Order Date
185551  Order ID  Product  Quantity Ordered  Price Each  Order Date
186563  Order ID  Product  Quantity Ordered  Price Each  Order Date
186632  Order ID  Product  Quantity Ordered  Price Each  Order Date
186738  Order ID  Product  Quantity Ordered  Price Each  Order Date

        Purchase Address
519     Purchase Address
1149    Purchase Address
1155    Purchase Address
2878    Purchase Address
2893    Purchase Address
…                  …
185164  Purchase Address
185551  Purchase Address
186563  Purchase Address
186632  Purchase Address
186738  Purchase Address

[355 rows x 6 columns]
```

[10]:
```
# remove rows
df = df[df['Quantity Ordered'] != 'Quantity Ordered']
df['Quantity Ordered'].unique()
```

[10]:
```
array(['2', '1', '3', '5', '4', '7', '6', '8', '9'], dtype=object)
```

Change data type

[11]:
```
# check data type
df.dtypes
```

[11]:
```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
```

```
dtype: object
```

[12]:
```
# Change data type
df['Quantity Ordered'] = df['Quantity Ordered'].astype(int)
df['Price Each'] = df['Price Each'].astype(float)
df['Order Date'] = pd.to_datetime(df['Order Date'])

# check data type
df.dtypes
```

[12]:
```
Order ID                    object
Product                     object
Quantity Ordered             int32
Price Each                 float64
Order Date          datetime64[ns]
Purchase Address            object
dtype: object
```

Add month column

[13]:
```
df['month'] = df['Order Date'].dt.month
df.head()
```

[13]:
```
  Order ID                     Product  Quantity Ordered  Price Each  \
0   176558        USB-C Charging Cable                 2       11.95
2   176559  Bose SoundSport Headphones                 1       99.99
3   176560                Google Phone                 1      600.00
4   176560            Wired Headphones                 1       11.99
5   176561            Wired Headphones                 1       11.99

           Order Date                      Purchase Address  month
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4
```

Create total sales column

[14]:
```
df['total_sales'] = df['Quantity Ordered'] * df['Price Each']
df.head()
```

[14]:
```
  Order ID                     Product  Quantity Ordered  Price Each  \
0   176558        USB-C Charging Cable                 2       11.95
2   176559  Bose SoundSport Headphones                 1       99.99
3   176560                Google Phone                 1      600.00
```

```
4   176560            Wired Headphones                 1       11.99
5   176561            Wired Headphones                 1       11.99

           Order Date                      Purchase Address  month  \
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   total_sales
0        23.90
2        99.99
3       600.00
4        11.99
5        11.99
```

[15]:
```
best_month_sales = df.groupby('month')['total_sales'].sum().
↪sort_values(ascending = False).reset_index()
best_month_sales
```

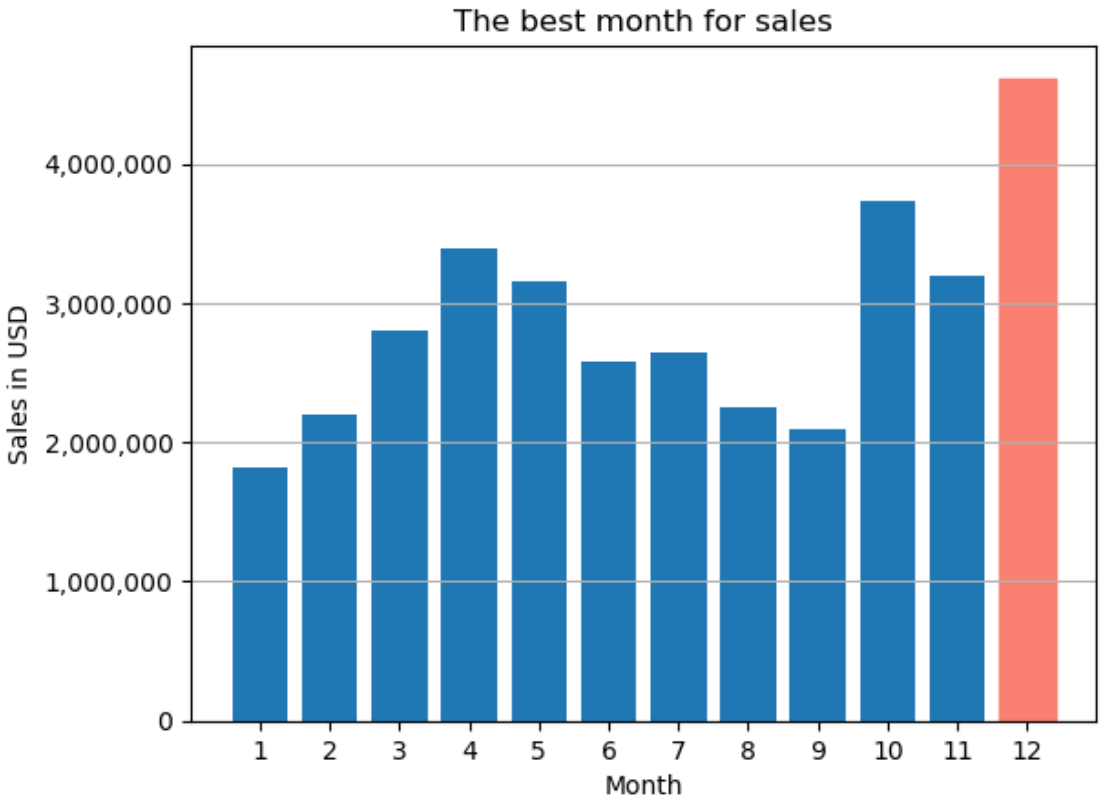
[15]:
```
    month  total_sales
0      12   4613443.34
1      10   3736726.88
2       4   3390670.24
3      11   3199603.20
4       5   3152606.75
5       3   2807100.38
6       7   2647775.76
7       6   2577802.26
8       8   2244467.88
9       2   2202022.42
10      9   2097560.13
11      1   1822256.73
```

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

Answer: December, 4613443.34

[16]:
```
# create bar chart
import matplotlib.pyplot as plt

bars = plt.bar(best_month_sales['month'], best_month_sales['total_sales'])
plt.xticks(best_month_sales['month'])  # show a name of each bar
plt.ylabel("Sales in USD")
plt.xlabel("Month")
```

```
plt.title('The best month for sales')
bars[0].set_color('salmon')

plt.grid(axis='y')

plt.gca().yaxis.set_major_formatter('{:,.0f}'.format) # y axis number format

plt.show()
```



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

[17]:
```
# preview
df.head()
```

[17]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	

```
4   176560            Wired Headphones                 1       11.99
5   176561            Wired Headphones                 1       11.99

           Order Date                       Purchase Address  month  \
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   total_sales
0        23.90
2        99.99
3       600.00
4        11.99
5        11.99
```

Create city column

[18]:
```
# find city(Regular expression)
import re
pattern = r',\s*([^,]+),'      #  \s*  ->  Matches zero or more white space
                               #  [^,]+ -> match one or more characters except a␣
 ↪comma
                               #   ()  ->  capture

df['city'] = df['Purchase Address'].str.extract(pattern)
df.head()
```

[18]:
```
  Order ID                    Product  Quantity Ordered  Price Each  \
0   176558        USB-C Charging Cable                 2       11.95
2   176559  Bose SoundSport Headphones                 1       99.99
3   176560                Google Phone                 1      600.00
4   176560            Wired Headphones                 1       11.99
5   176561            Wired Headphones                 1       11.99

           Order Date                       Purchase Address  month  \
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   total_sales         city
0        23.90       Dallas
2        99.99       Boston
```

```
3       600.00  Los Angeles
4        11.99  Los Angeles
5        11.99  Los Angeles
```

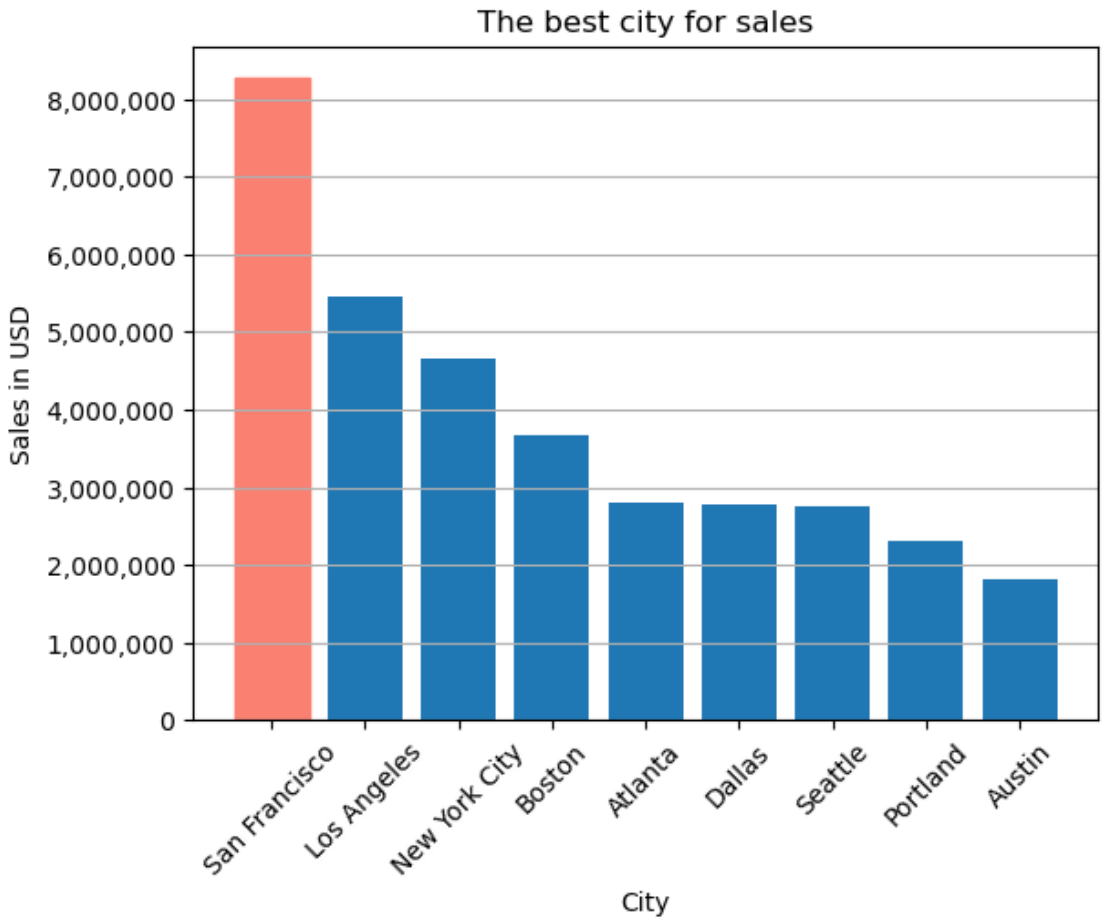
[19]:
```
# Sum 'total_sales' group by 'city'
best_city_sales = df.groupby('city')['total_sales'].sum().
 ↪sort_values(ascending=False).reset_index()
best_city_sales
```

[19]:
```
            city  total_sales
0  San Francisco   8262203.91
1    Los Angeles   5452570.80
2  New York City   4664317.43
3         Boston   3661642.01
4        Atlanta   2795498.58
5         Dallas   2767975.40
6        Seattle   2747755.48
7       Portland   2320490.61
8         Austin   1819581.75
```

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

Answer: San Francisco 8262203.91

[20]:
```
# create bar chart
bars = plt.bar(best_city_sales['city'], best_city_sales['total_sales'])

plt.xticks(rotation = 45)
plt.ylabel("Sales in USD")
plt.xlabel("City")
plt.title('The best city for sales')
bars[0].set_color('salmon')
#plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.grid(axis='y')
plt.gca().yaxis.set_major_formatter('{:,.0f}'.format)  # y axis number format

plt.show()
```



Question 3: What time should we display advertisements likelihood of customer's buying product?

[23]:
```
df.head()
```

[23]:
```
   Order ID                     Product  Quantity Ordered  Price Each  \
0    176558        USB-C Charging Cable                 2       11.95
2    176559  Bose SoundSport Headphones                 1       99.99
3    176560                Google Phone                 1      600.00
4    176560            Wired Headphones                 1       11.99
5    176561            Wired Headphones                 1       11.99

            Order Date                      Purchase Address  month  \
0  2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2  2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
```

```
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   total_sales         city
0        23.90       Dallas
2        99.99       Boston
3       600.00  Los Angeles
4        11.99  Los Angeles
5        11.99  Los Angeles
```

Add 'time' column

[24]:
```
# Add hour and minute column
df['Hour'] = df['Order Date'].dt.hour
df.head()
```

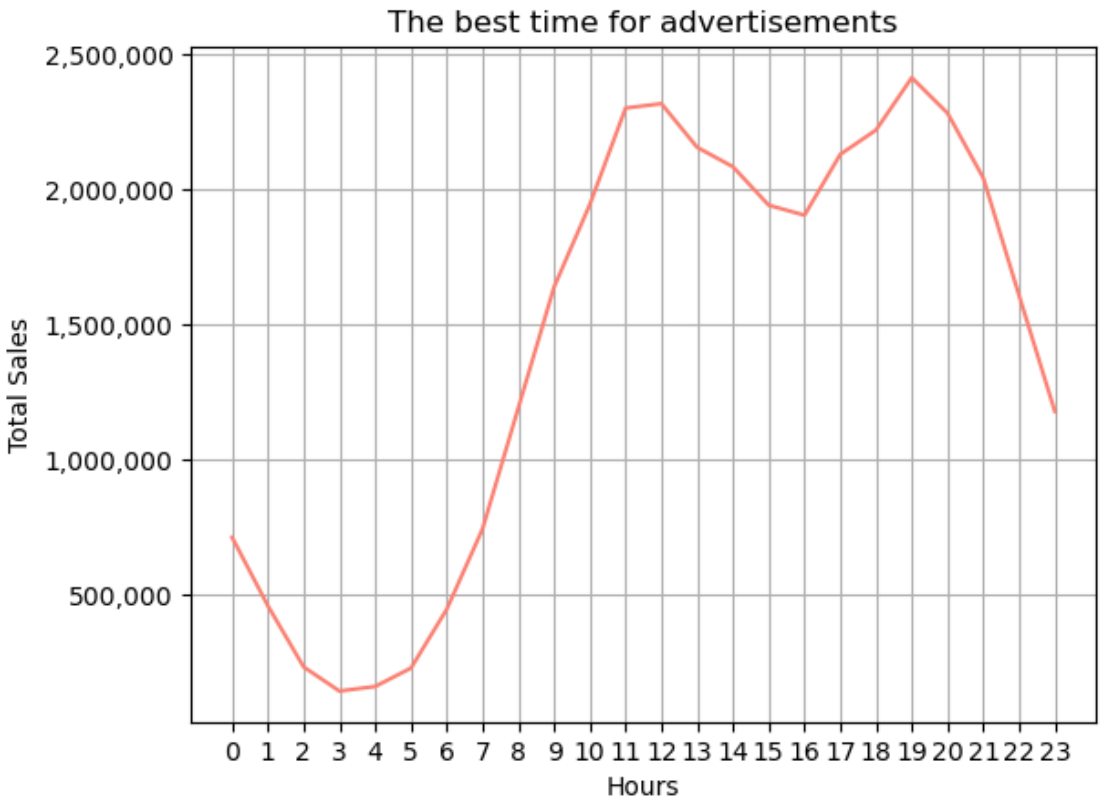
[24]:
```
  Order ID                    Product  Quantity Ordered  Price Each  \
0   176558        USB-C Charging Cable                 2       11.95
2   176559  Bose SoundSport Headphones                 1       99.99
3   176560                Google Phone                 1      600.00
4   176560            Wired Headphones                 1       11.99
5   176561            Wired Headphones                 1       11.99

           Order Date                      Purchase Address  month  \
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   total_sales         city  Hour
0        23.90       Dallas     8
2        99.99       Boston    22
3       600.00  Los Angeles    14
4        11.99  Los Angeles    14
5        11.99  Los Angeles     9
```

[25]:
```
# Sum 'total_sales' group by 'Hour'
hour_sales = df.groupby('Hour')['total_sales'].sum().reset_index().
 ↪sort_values(by = 'Hour')
hour_sales
```

[25]:
```
    Hour  total_sales
0      0    713721.27
1      1    460866.88
2      2    234851.44
```

```
3      3    145757.89
4      4    162661.01
5      5    230679.82
6      6    448113.00
7      7    744854.12
8      8   1192348.97
9      9   1639030.58
10    10   1944286.77
11    11   2300610.24
12    12   2316821.34
13    13   2155389.80
14    14   2083672.73
15    15   1941549.60
16    16   1904601.31
17    17   2129361.61
18    18   2219348.30
19    19   2412938.54
20    20   2281716.24
21    21   2042000.86
22    22   1607549.21
23    23   1179304.44
```

[26]:
```
# create chart
plot = plt.plot(hour_sales['Hour'],hour_sales['total_sales'], color = 'salmon')
plt.xticks(hour_sales['Hour'])
plt.ylabel("Total Sales")
plt.xlabel("Hours")
plt.title('The best time for advertisements')

plt.gca().yaxis.set_major_formatter('{:,.0f}'.format)  # y axis number format
plt.grid(True)


plt.show()
```



Question 3: What time should we display advertisements likelihood of customer's buying product?

Answer: 11.00-12.00 and 18.00-20.00

Question 4: What products are most often sold together?

[27]:
```
df.head()
```

[27]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

```
           Order Date                      Purchase Address  month  \
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   total_sales         city  Hour
0        23.90       Dallas     8
2        99.99       Boston    22
3       600.00  Los Angeles    14
4        11.99  Los Angeles    14
5        11.99  Los Angeles     9
```

[28]:
```
# find duplicate id
df_dup = df[df['Order ID'].duplicated(keep=False)] # False -> keep all␣
 ↪duplicate rows
df_dup.head()
```

[28]:
```
   Order ID                    Product  Quantity Ordered  Price Each  \
3    176560                Google Phone                 1      600.00
4    176560            Wired Headphones                 1       11.99
18   176574                Google Phone                 1      600.00
19   176574        USB-C Charging Cable                 1       11.95
30   176585  Bose SoundSport Headphones                 1       99.99

            Order Date                      Purchase Address  month  \
3  2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4  2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
18 2019-04-03 19:42:00     20 Hill St, Los Angeles, CA 90001      4
19 2019-04-03 19:42:00     20 Hill St, Los Angeles, CA 90001      4
30 2019-04-07 11:31:00     823 Highland St, Boston, MA 02215      4

    total_sales         city  Hour
3        600.00  Los Angeles    14
4         11.99  Los Angeles    14
18       600.00  Los Angeles    19
19        11.95  Los Angeles    19
30        99.99       Boston    11
```

[29]:
```
# add group_product column
df_dup['group'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.
 ↪join(x))
# .transform() method applies this lambda function to each group of 'Product'

df_dup.head(10)
```

```
C:\Users\AVS_KTB\AppData\Local\Temp\ipykernel_6124\1989000491.py:2:
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_dup['group'] = df.groupby('Order ID')['Product'].transform(lambda x:
','.join(x))
```

[29]:

	Order ID	Product	Quantity Ordered	Price Each	\
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
18	176574	Google Phone	1	600.00	
19	176574	USB-C Charging Cable	1	11.95	
30	176585	Bose SoundSport Headphones	1	99.99	
31	176585	Bose SoundSport Headphones	1	99.99	
32	176586	AAA Batteries (4-pack)	2	2.99	
33	176586	Google Phone	1	600.00	
119	176672	Lightning Charging Cable	1	14.95	
120	176672	USB-C Charging Cable	1	11.95	

	Order Date	Purchase Address	month	\
3	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	
4	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	
18	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	
19	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	4	
30	2019-04-07 11:31:00	823 Highland St, Boston, MA 02215	4	
31	2019-04-07 11:31:00	823 Highland St, Boston, MA 02215	4	
32	2019-04-10 17:00:00	365 Center St, San Francisco, CA 94016	4	
33	2019-04-10 17:00:00	365 Center St, San Francisco, CA 94016	4	
119	2019-04-12 11:07:00	778 Maple St, New York City, NY 10001	4	
120	2019-04-12 11:07:00	778 Maple St, New York City, NY 10001	4	

	total_sales	city	Hour	\
3	600.00	Los Angeles	14	
4	11.99	Los Angeles	14	
18	600.00	Los Angeles	19	
19	11.95	Los Angeles	19	
30	99.99	Boston	11	
31	99.99	Boston	11	
32	5.98	San Francisco	17	
33	600.00	San Francisco	17	
119	14.95	New York City	11	
120	11.95	New York City	11	

	group

```
3                       Google Phone,Wired Headphones
4                       Google Phone,Wired Headphones
18                   Google Phone,USB-C Charging Cable
19                   Google Phone,USB-C Charging Cable
30   Bose SoundSport Headphones,Bose SoundSport Hea…
31   Bose SoundSport Headphones,Bose SoundSport Hea…
32                AAA Batteries (4-pack),Google Phone
33                AAA Batteries (4-pack),Google Phone
119     Lightning Charging Cable,USB-C Charging Cable
120     Lightning Charging Cable,USB-C Charging Cable
```

[30]:
```
# select 2 columns & remove duplicate
df_dup = df_dup[['Order ID','group']].drop_duplicates()
df_dup.head(10)
```

[30]:
```
    Order ID                                              group
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
129   176681           Apple Airpods Headphones,ThinkPad Laptop
138   176689  Bose SoundSport Headphones,AAA Batteries (4-pack)
189   176739                 34in Ultrawide Monitor,Google Phone
225   176774      Lightning Charging Cable,USB-C Charging Cable
233   176781                     iPhone,Lightning Charging Cable
```

[31]:
```
from itertools import combinations   # generates all possible combinations of a␣
 ↪list of items
from collections import Counter      # count a combinations

count = Counter()                    # store the counts of combinations.

for row in df_dup['group']:
    row_list = row.split(',')        # separated string (row) into a list of␣
 ↪individual products (row_list).
                                     # ex. 'AAA Batteries (4-pack),Google Phone'␣
 ↪-> ['AAA Batteries (4-pack)', 'Google Phone']
    count.update(Counter(combinations(row_list, 2)))
    # combinations(row_list, 3) -> generates all combinations of 3 products␣
 ↪without repetition.(creates tuples)
    # Counter(combinations(row_list, 3)) -> count the occurrences & converts␣
 ↪the combinations into a dictionary (key, values)
    # count.update(...) -> The update method of the Counter class is used to␣
 ↪update the count object
```

```
count.most_common(10)  # method of the Counter class sorts the combinations(10 combinations)
```

[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)]
```

[32]:
```
# print only key & values
for key, values in count.most_common(10):
    print(key, values)
```

```
('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
```

Question 4: What products are most often sold together?

Answer: iPhone & Lightning Charging Cable

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

[33]:
```
df.head()
```

[33]:

	Order ID	Product	Quantity Ordered	Price Each	\
0	176558	USB-C Charging Cable	2	11.95	
2	176559	Bose SoundSport Headphones	1	99.99	
3	176560	Google Phone	1	600.00	
4	176560	Wired Headphones	1	11.99	
5	176561	Wired Headphones	1	11.99	

```
           Order Date                       Purchase Address  month  \
0 2019-04-19 08:46:00          917 1st St, Dallas, TX 75001      4
2 2019-04-07 22:30:00     682 Chestnut St, Boston, MA 02215      4
3 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
4 2019-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001      4
5 2019-04-30 09:27:00     333 8th St, Los Angeles, CA 90001      4

   total_sales         city  Hour
0        23.90       Dallas     8
2        99.99       Boston    22
3       600.00  Los Angeles    14
4        11.99  Los Angeles    14
5        11.99  Los Angeles     9
```

[21]:
```
quantity_ordered = df.groupby('Product')['Quantity Ordered'].sum().reset_index()
quantity_ordered
```

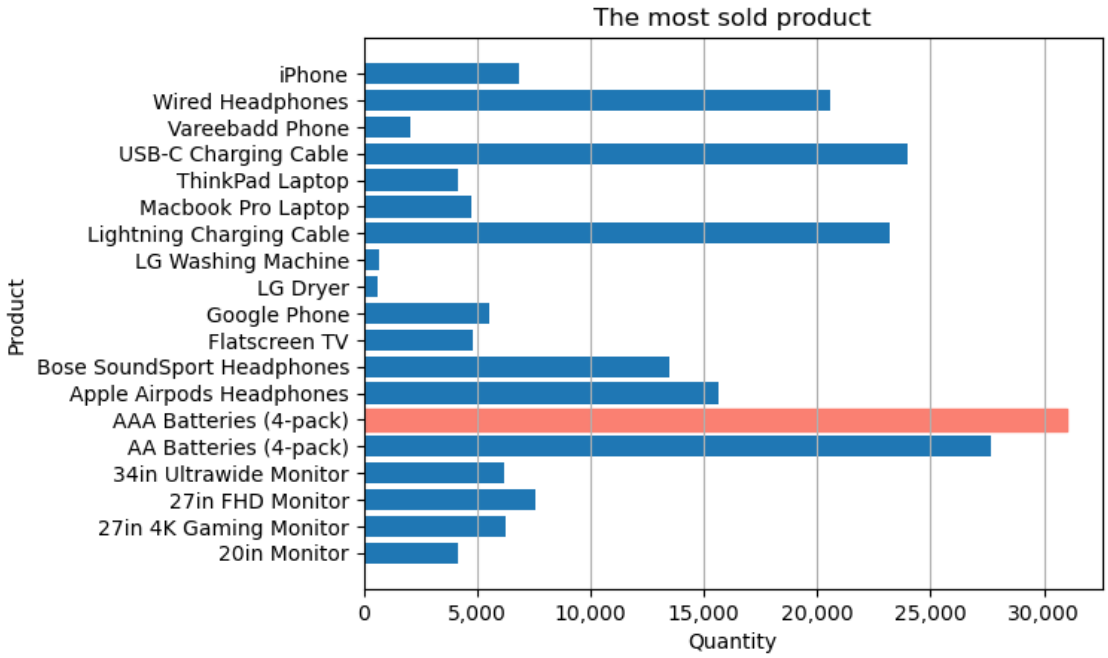
[21]:
```
                       Product  Quantity Ordered
0                  20in Monitor              4129
1        27in 4K Gaming Monitor              6244
2              27in FHD Monitor              7550
3        34in Ultrawide Monitor              6199
4         AA Batteries (4-pack)             27635
5        AAA Batteries (4-pack)             31017
6      Apple Airpods Headphones             15661
7    Bose SoundSport Headphones             13457
8                 Flatscreen TV              4819
9                  Google Phone              5532
10                     LG Dryer               646
11           LG Washing Machine               666
12    Lightning Charging Cable             23217
13           Macbook Pro Laptop              4728
14              ThinkPad Laptop              4130
15         USB-C Charging Cable             23975
16              Vareebadd Phone              2068
17             Wired Headphones             20557
18                       iPhone              6849
```

[22]:
```
# overview
df.groupby('Product')['Quantity Ordered'].sum().sort_values(ascending = False).
  ↪reset_index().head()
```

[22]:
```
                     Product  Quantity Ordered
0    AAA Batteries (4-pack)             31017
1     AA Batteries (4-pack)             27635
2      USB-C Charging Cable             23975
```

```
3  Lightning Charging Cable              23217
4          Wired Headphones              20557
```

[23]:
```
# create chart
products = plt.barh(quantity_ordered['Product'],quantity_ordered['Quantity␣
 ↪Ordered'])

plt.xlabel("Quantity")
plt.ylabel("Product")
plt.title('The most sold product')
products[5].set_color('salmon')

plt.gca().xaxis.set_major_formatter('{:,.0f}'.format)  # y axis number format

plt.grid(axis='x')

plt.show()
```



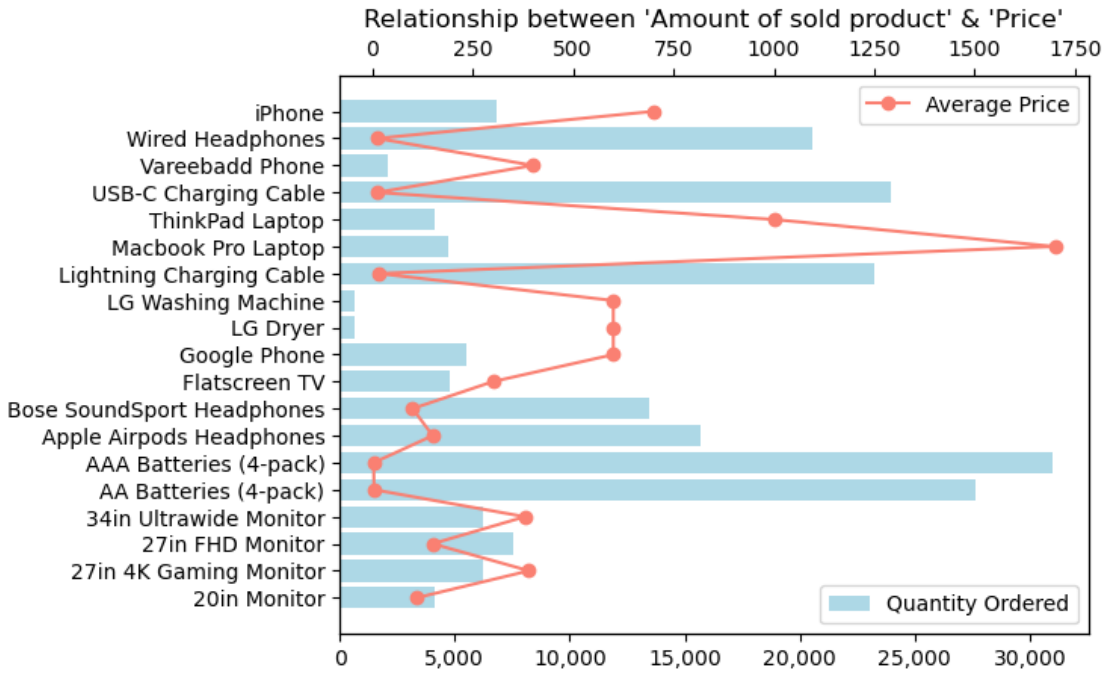
[37]:
```
prices = df.groupby('Product')['Price Each'].mean().reset_index()
prices
```

[37]:
```
                      Product  Price Each
0                  20in Monitor      109.99
1       27in 4K Gaming Monitor      389.99
2             27in FHD Monitor      149.99
```

```
3       34in Ultrawide Monitor      379.99
4        AA Batteries (4-pack)        3.84
5       AAA Batteries (4-pack)        2.99
6     Apple Airpods Headphones      150.00
7   Bose SoundSport Headphones       99.99
8                Flatscreen TV      300.00
9                 Google Phone      600.00
10                    LG Dryer      600.00
11          LG Washing Machine      600.00
12    Lightning Charging Cable       14.95
13          Macbook Pro Laptop     1700.00
14             ThinkPad Laptop      999.99
15        USB-C Charging Cable       11.95
16             Vareebadd Phone      400.00
17           Wired Headphones       11.99
18                     iPhone      700.00
```

[38]:
```python
# final version version
# Group the DataFrame by 'Product' and calculate the sum of 'Quantity Ordered' for each product
quantity_ordered = df.groupby('Product')['Quantity Ordered'].sum().reset_index()

# Group the DataFrame by 'Product' and calculate the mean of 'Price Each' for each product
prices = df.groupby('Product')['Price Each'].mean().reset_index()

# Create the figure and the first set of axes (for quantity ordered)
fig, ax1 = plt.subplots()

# Create the horizontal bar chart for quantity ordered
products = ax1.barh(quantity_ordered['Product'], quantity_ordered['Quantity Ordered'], color='lightblue', label='Quantity Ordered')

# Set labels for the first y-axis and a title for the chart
ax1.set_title("Relationship between 'Amount of sold product' & 'Price'")

# Format the first y-axis (quantity ordered) x-axis numbers to have commas for thousands separator
ax1.xaxis.set_major_formatter('{:,.0f}'.format)



# Create the second set of axes (for average prices) using twinx()
ax2 = ax1.twiny()  # use the same x-axis for both chart

# Create the line plot for average prices on the second axes
```

```
ax2.plot(prices['Price Each'], prices['Product'], marker='o', color='salmon',␣
 ↪label='Average Price')

# Show the legend for both sets of data
ax1.legend(loc='lower right')
ax2.legend(loc='upper right')

# Remove grid lines
ax1.grid(False)
ax2.grid(False)

# Display the chart
#plt.tight_layout()
plt.show()
```



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

Answer: AA Batteries (4-pack) are the cheapest product with a short usage life cycle, making them a frequent choice for people to buy.

Conclusions

1. Advertising in December, October, and April, as the highest order generation revenues as fol
   - December $4,613,443
   - October $3,736,726
   - April $3,390,670

2.The top three cities with the highest total sales are as follows:
   - San Francisco $8,262,203
   - Los Angeles $5,452,570
   - New York City $4,664,317

3.The best times for advertising were between 11:00-12:00 and 18:00-19:00.

4.The frequently co-purchased products are as follows:
   - iPhone and Lightning Charging Cable 1,005 orders
   - Google Phone and USB-C Charging Cable 987 orders
   - iPhone and Wired Headphones 447 orders

5.The top 5 best-selling products are as follows:
   - AAA Batteries (4-pack) 31,017 units
   - AA Batteries (4-pack) 27,635 units
   - USB-C Charging Cable 23,975 units
   - Lightning Charging Cable 23,217 units
   - Wired Headphones 20,557 units

note: Products with lower prices are more likely to be sold in higher quantities