University of Colima
Faculty of Mechanical and Electrical Engineering
Intelligent Computer Engineering

Analysis and Compare Dengue Cases in Mexico Until February 28, 2024
Data analysis and visualization

Ceballos Contreras Jesús Eduardo 20176724
Larios Bravo Cristian Armando 20188165
López Aguilar Alba Beth-birai 20186885
Olmos Romero Nydia Naomi 20184561

6°D

Place: Mexico, Colima, Coquimatlan.
Date: 12/04/2024.

Product Sales Analysis

1. Loading the dataset into a dataframe

In [ ]:
```
# import the libraries
import sklearn.datasets as ds
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:
```
# load the data
df = pd.read_csv('sales_data.csv')
df
```

Out[ ]:

	Order Date	Order ID	Product	Product_ean	catégorie	Purchase Address	Quantity Ordered	P E
0	2019-01-22 21:25:00	141234	iPhone	5.638009e+12	Vêtements	944 Walnut St, Boston, MA 02215	1	70(
1	2019-01-28 14:15:00	141235	Lightning Charging Cable	5.563320e+12	Alimentation	185 Maple St, Portland, OR 97035	1	1
2	2019-01-17 13:33:00	141236	Wired Headphones	2.113973e+12	Vêtements	538 Adams St, San Francisco, CA 94016	2	1
3	2019-01-05 20:33:00	141237	27in FHD Monitor	3.069157e+12	Sports	738 10th St, Los Angeles, CA 90001	1	14
4	2019-01-25 11:59:00	141238	Wired Headphones	9.692681e+12	Électronique	387 10th St, Austin, TX 73301	1	1
...	...	...	...	...	...	...	...	
185945	2019-12-11 20:58:00	319666	Lightning Charging Cable	6.545974e+12	Électronique	14 Madison St, San Francisco, CA 94016	1	1
185946	2019-12-01 12:01:00	319667	AA Batteries (4-pack)	5.352480e+12	Électronique	549 Willow St, Los Angeles, CA 90001	2	
185947	2019-12-09 06:43:00	319668	Vareebadd Phone	2.674213e+12	Alimentation	273 Wilson St, Seattle, WA 98101	1	40(
185948	2019-12-03 10:39:00	319669	Wired Headphones	5.216304e+12	Alimentation	778 River St, Dallas, TX 75001	1	1

	Order Date	Order ID	Product	Product_ean	catégorie	Purchase Address	Quantity Ordered	P E
185949	2019-12-21 21:45:00	319670	Bose SoundSport Headphones	8.081038e+12	Électronique	747 Chestnut St, Los Angeles, CA 90001	1	9

185950 rows × 11 columns

2. Perform an initial scan of the data.

In [ ]:
```
# Review the first rows of the DataFrame to understand the structure of the data.
df.head()
```

Out[ ]:

	Order Date	Order ID	Product	Product_ean	catégorie	Purchase Address	Quantity Ordered	Price Each
0	2019-01-22 21:25:00	141234	iPhone	5.638009e+12	Vêtements	944 Walnut St, Boston, MA 02215	1	700.00
1	2019-01-28 14:15:00	141235	Lightning Charging Cable	5.563320e+12	Alimentation	185 Maple St, Portland, OR 97035	1	14.95
2	2019-01-17 13:33:00	141236	Wired Headphones	2.113973e+12	Vêtements	538 Adams St, San Francisco, CA 94016	2	11.99
3	2019-01-05 20:33:00	141237	27in FHD Monitor	3.069157e+12	Sports	738 10th St, Los Angeles, CA 90001	1	149.99
4	2019-01-25 11:59:00	141238	Wired Headphones	9.692681e+12	Électronique	387 10th St, Austin, TX 73301	1	11.99

In [ ]:
```
# Check and handle missing data (NaN values) if necessary.
# df.isnull().sum()
print(f"Data before dropping NaN values: \n{df.isna().sum()}")
```

```
Data before dropping NaN values:
Order Date          0
Order ID            0
Product             0
Product_ean         0
catégorie           0
Purchase Address    0
Quantity Ordered    0
Price Each          0
Cost price          0
turnover            0
margin              0
dtype: int64
```

In [ ]:
```
# Obtain basic statistical information about numerical variables.
df.describe()
```

Out[ ]:

	Order ID	Product_ean	Quantity Ordered	Price Each	Cost price	tur
count	185950.000000	1.859500e+05	185950.000000	185950.000000	185950.000000	185950.0
mean	230417.569379	5.509211e+12	1.124383	184.399735	69.668583	185.4
std	51512.737110	2.598403e+12	0.442793	332.731330	109.424191	332.9
min	141234.000000	1.000083e+12	1.000000	2.990000	1.495000	2.9
25%	185831.250000	3.254280e+12	1.000000	11.950000	5.975000	11.9
50%	230367.500000	5.511235e+12	1.000000	14.950000	7.475000	14.9
75%	275035.750000	7.765195e+12	1.000000	150.000000	97.500000	150.0
max	319670.000000	9.999983e+12	9.000000	1700.000000	561.000000	3400.0

In [ ]:
```
# Identify the categorical variables in the dataset.
categorical = df.select_dtypes(include=['object'])
categorical
```

Out[ ]:

	Order Date	Product	catégorie	Purchase Address
0	2019-01-22 21:25:00	iPhone	Vêtements	944 Walnut St, Boston, MA 02215
1	2019-01-28 14:15:00	Lightning Charging Cable	Alimentation	185 Maple St, Portland, OR 97035
2	2019-01-17 13:33:00	Wired Headphones	Vêtements	538 Adams St, San Francisco, CA 94016
3	2019-01-05 20:33:00	27in FHD Monitor	Sports	738 10th St, Los Angeles, CA 90001
4	2019-01-25 11:59:00	Wired Headphones	Électronique	387 10th St, Austin, TX 73301
...	...	...	...	...
185945	2019-12-11 20:58:00	Lightning Charging Cable	Électronique	14 Madison St, San Francisco, CA 94016
185946	2019-12-01 12:01:00	AA Batteries (4-pack)	Électronique	549 Willow St, Los Angeles, CA 90001
185947	2019-12-09 06:43:00	Vareebadd Phone	Alimentation	273 Wilson St, Seattle, WA 98101
185948	2019-12-03 10:39:00	Wired Headphones	Alimentation	778 River St, Dallas, TX 75001
185949	2019-12-21 21:45:00	Bose SoundSport Headphones	Électronique	747 Chestnut St, Los Angeles, CA 90001

185950 rows × 4 columns

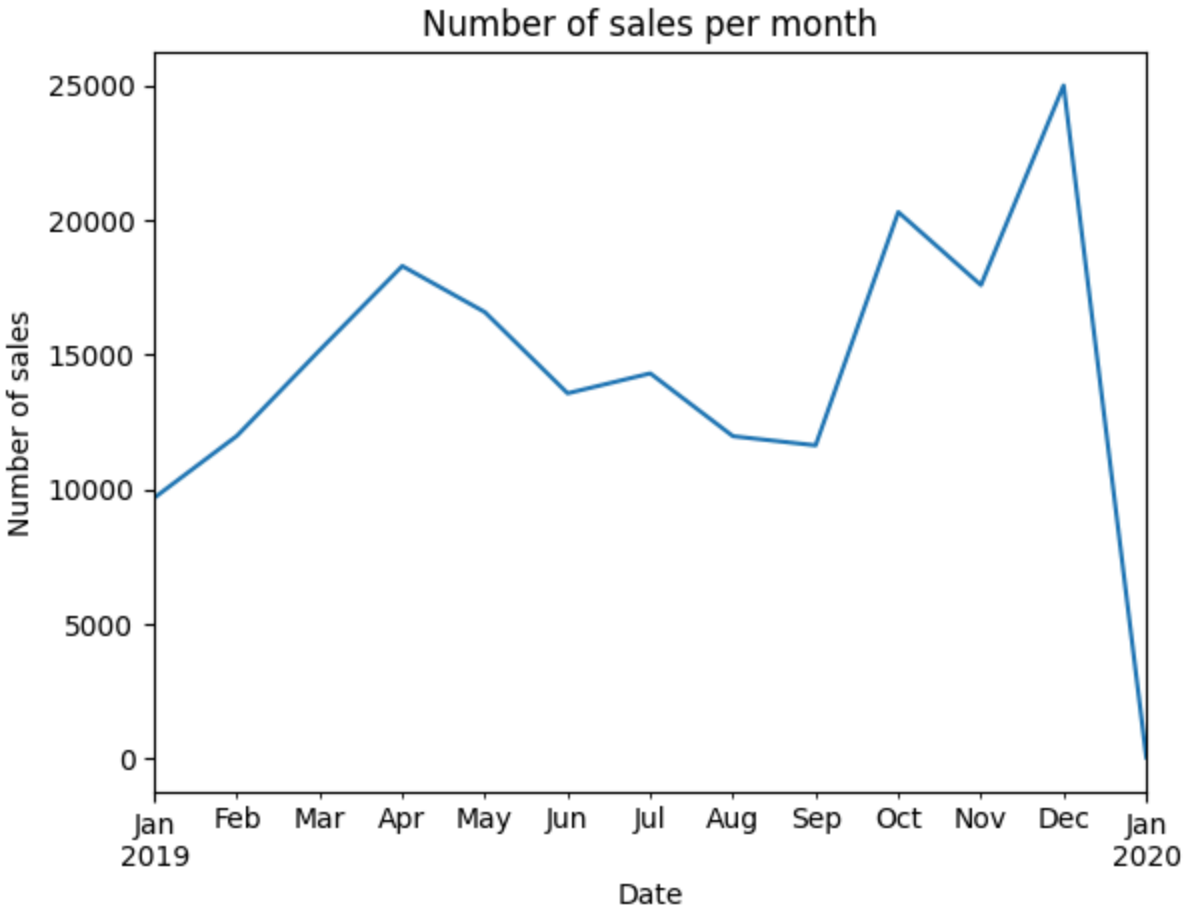
3. Visualización de Datos:

In [ ]:
```
# Date convertion
x = df["Order Date"]
x = x[:10]
# # x
# df["Order Date"] = x
df["Order Date"] = pd.to_datetime(df["Order Date"])
```

a). Crea una gráfica de línea que muestre la tendencia de ventas mensuales de todos los productos durante los últimos 12 meses.

In [ ]:
```
# a). Create a line graph that shows the monthly sales trend for all products over
#1. Get the number of sales per month
sales = df["Order Date"].value_counts().sort_index()
sales = sales.resample('ME').sum()
sales.plot(kind='line')
plt.title("Number of sales per month")
plt.ylabel("Number of sales")
```

```
plt.xlabel("Date")
plt.show()
plt.figure(figsize=(10, 6))
```



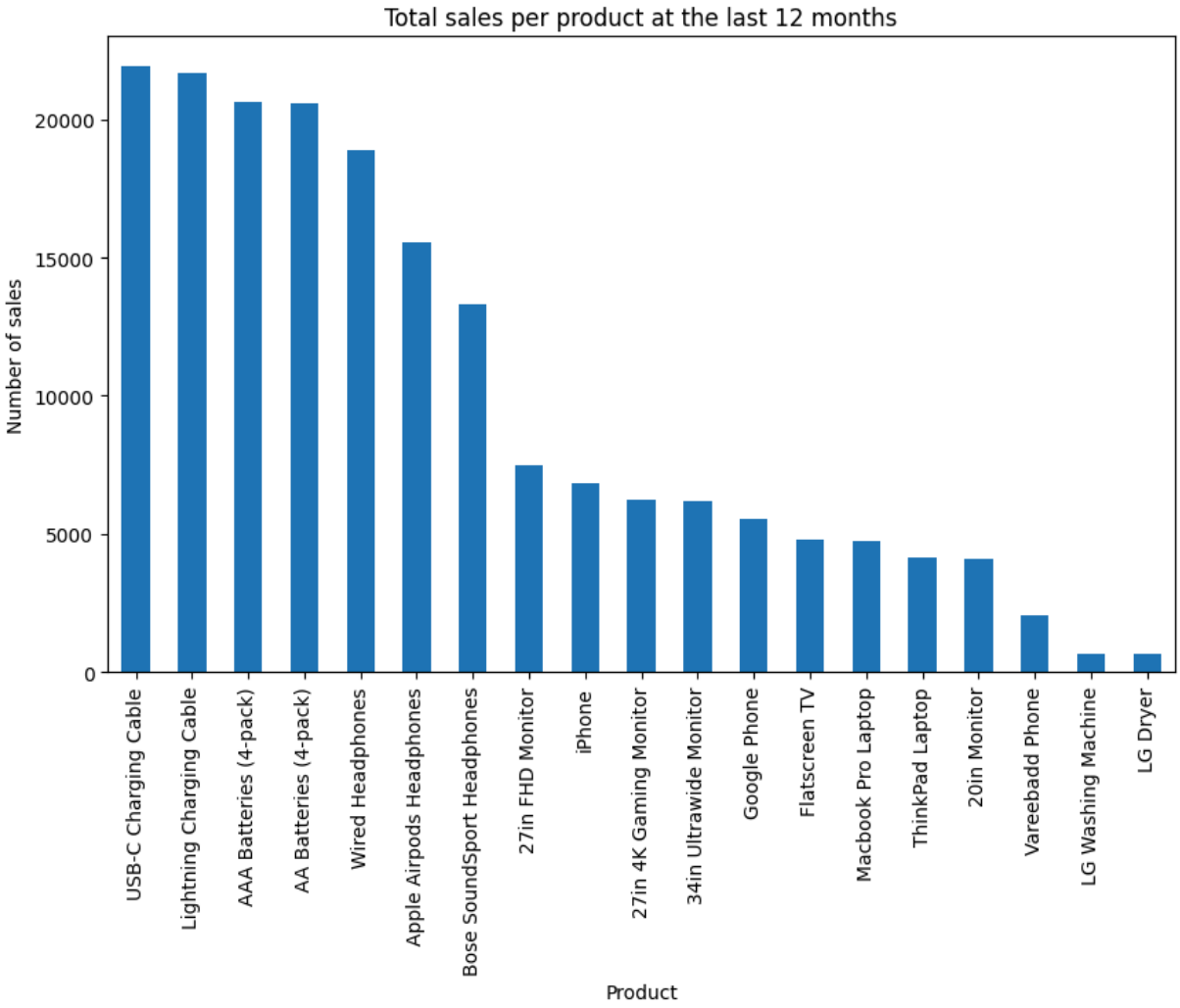
Out[ ]: <Figure size 1000x600 with 0 Axes>

<Figure size 1000x600 with 0 Axes>

b). Generate a bar chart showing the total sales of each product over the last 12 months.

In [ ]:
```
plt.figure(figsize=(10, 6))
productos = df["Product"].value_counts()
productos.plot(kind='bar')
plt.title("Total sales per product at the last 12 months")
plt.ylabel("Number of sales")
plt.xlabel("Product")
```

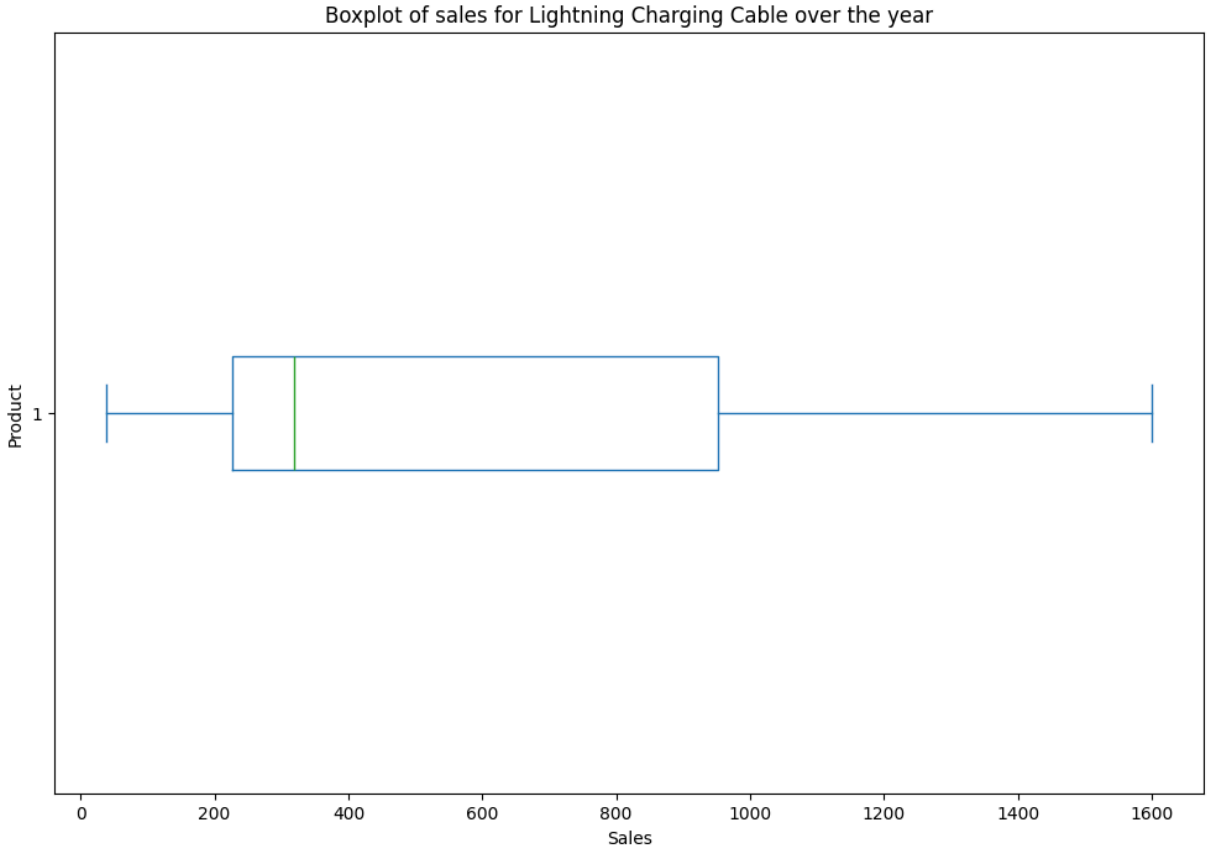
Out[ ]: Text(0.5, 0, 'Product')

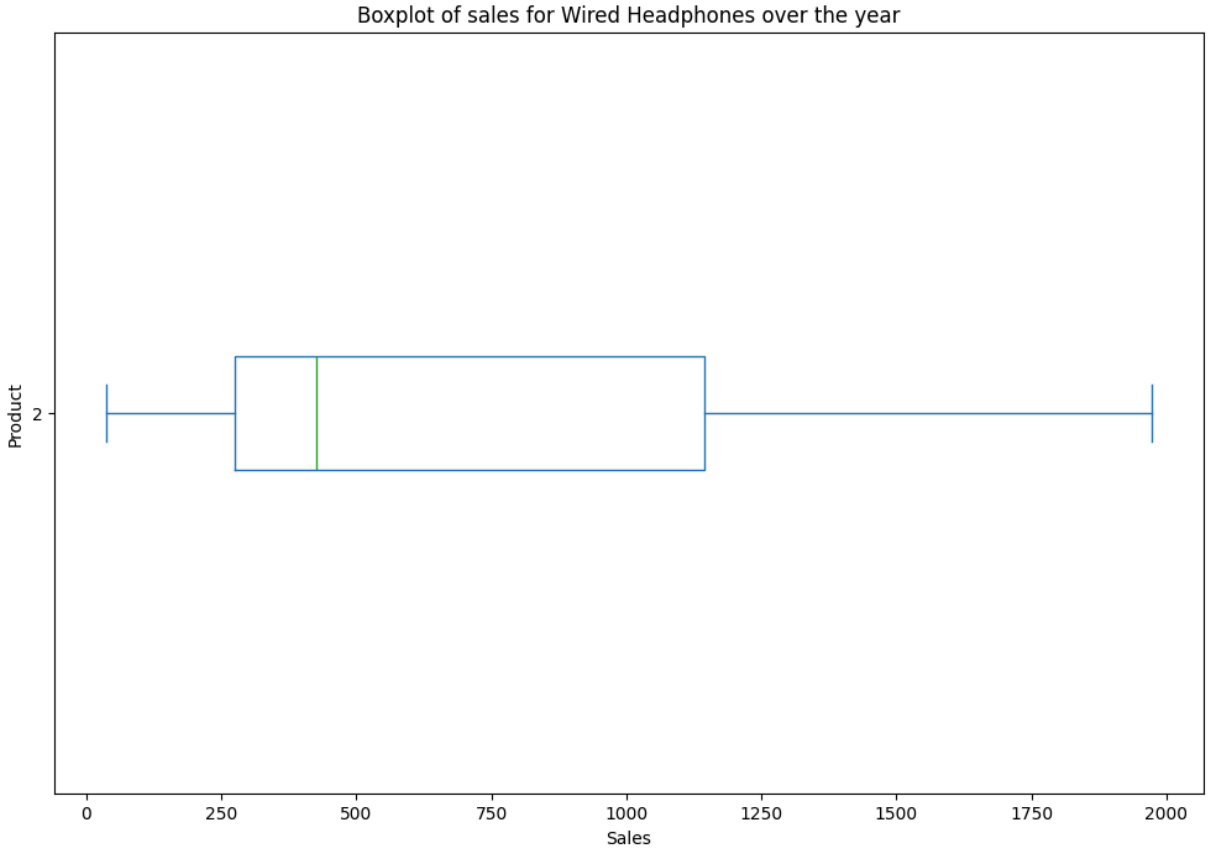


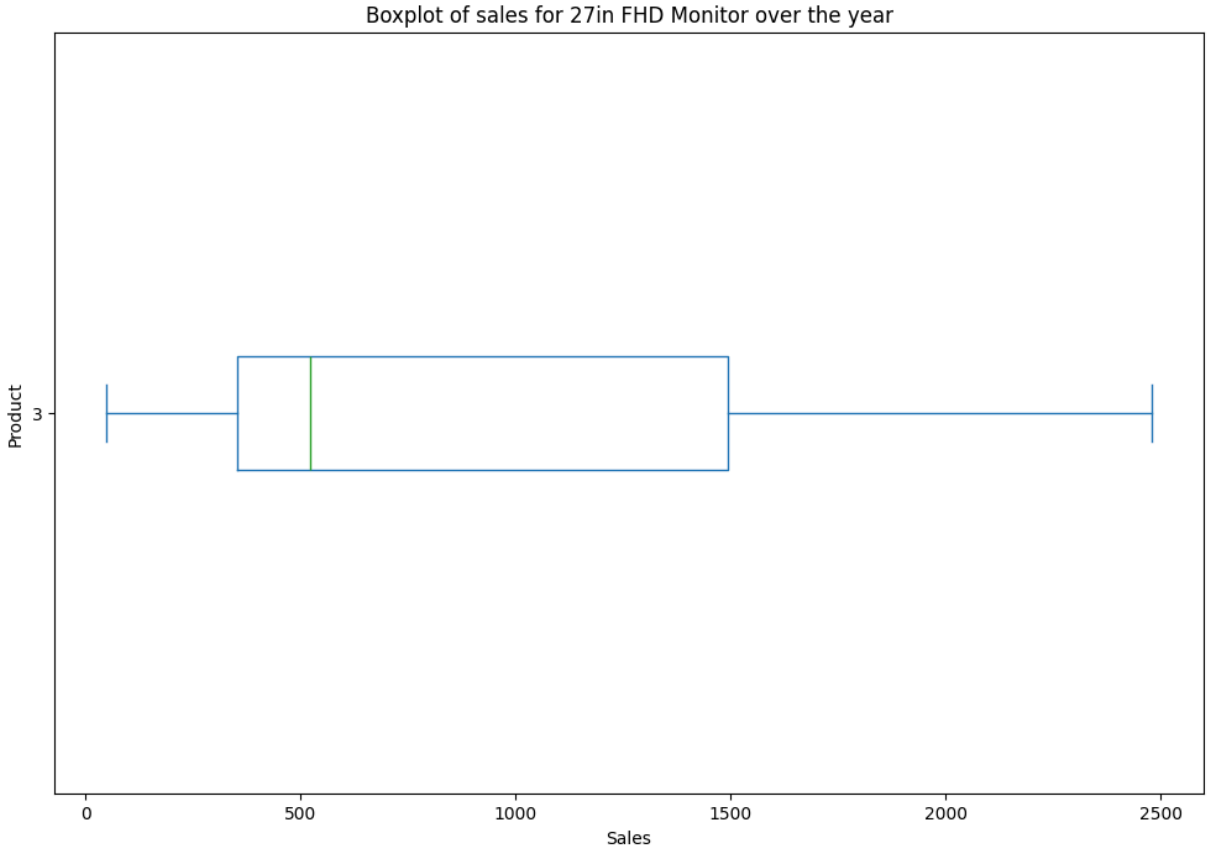
c). Create a boxplot to visualize the distribution of sales for each product during the year.

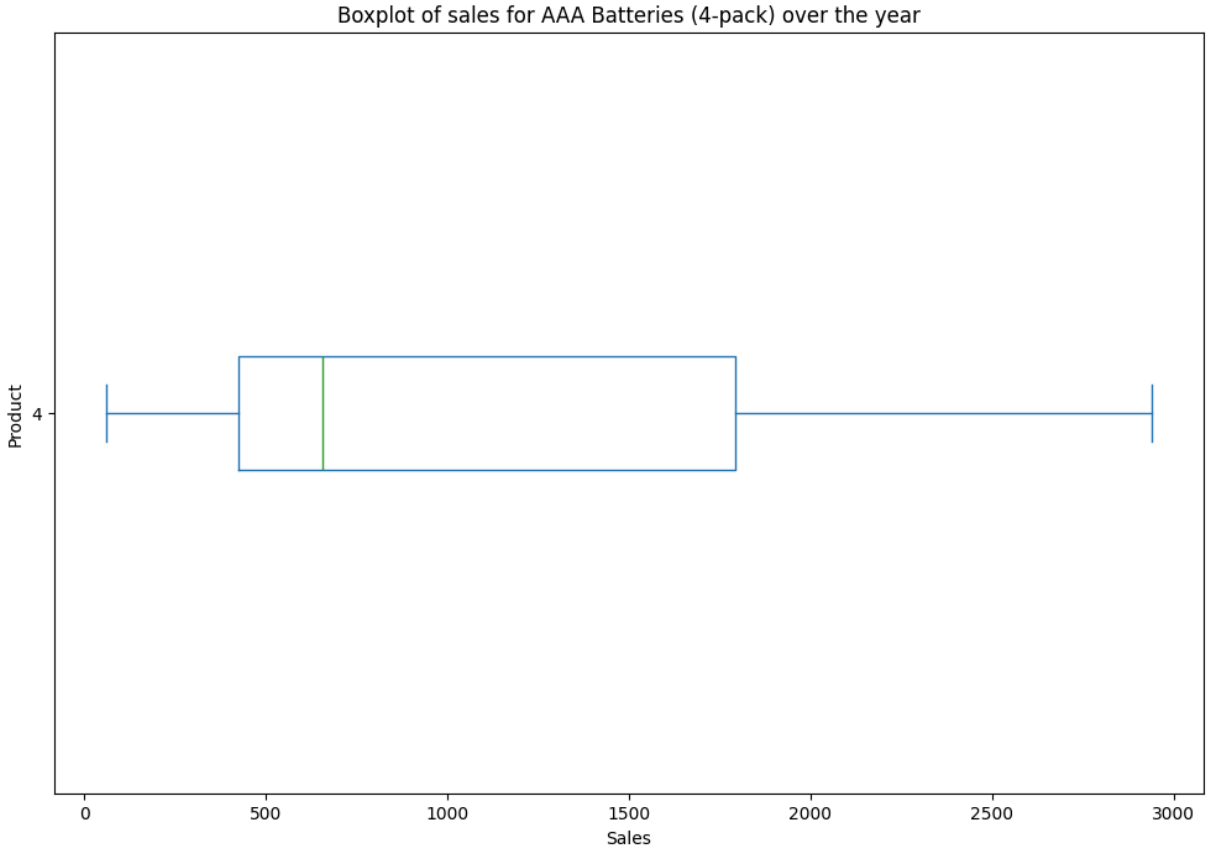
In [ ]:
```
# c). Create a boxplot to visualize the distribution of sales for each product duri
products = df["Product"].unique()
salesPerProductPerMonth = df.groupby(["Product", df["Order Date"].dt.month])["Quant
salesPerProductPerMonth = salesPerProductPerMonth.unstack(level=1)
salesPerProductPerMonth

for product in salesPerProductPerMonth:
    plt.figure(figsize=(12, 8))
    salesPerProductPerMonth[product].plot(kind='box', vert=False)
    plt.title(f"Boxplot of sales for {products[product]} over the year")
    plt.ylabel("Product")
    plt.xlabel("Sales")
    plt.show()
```

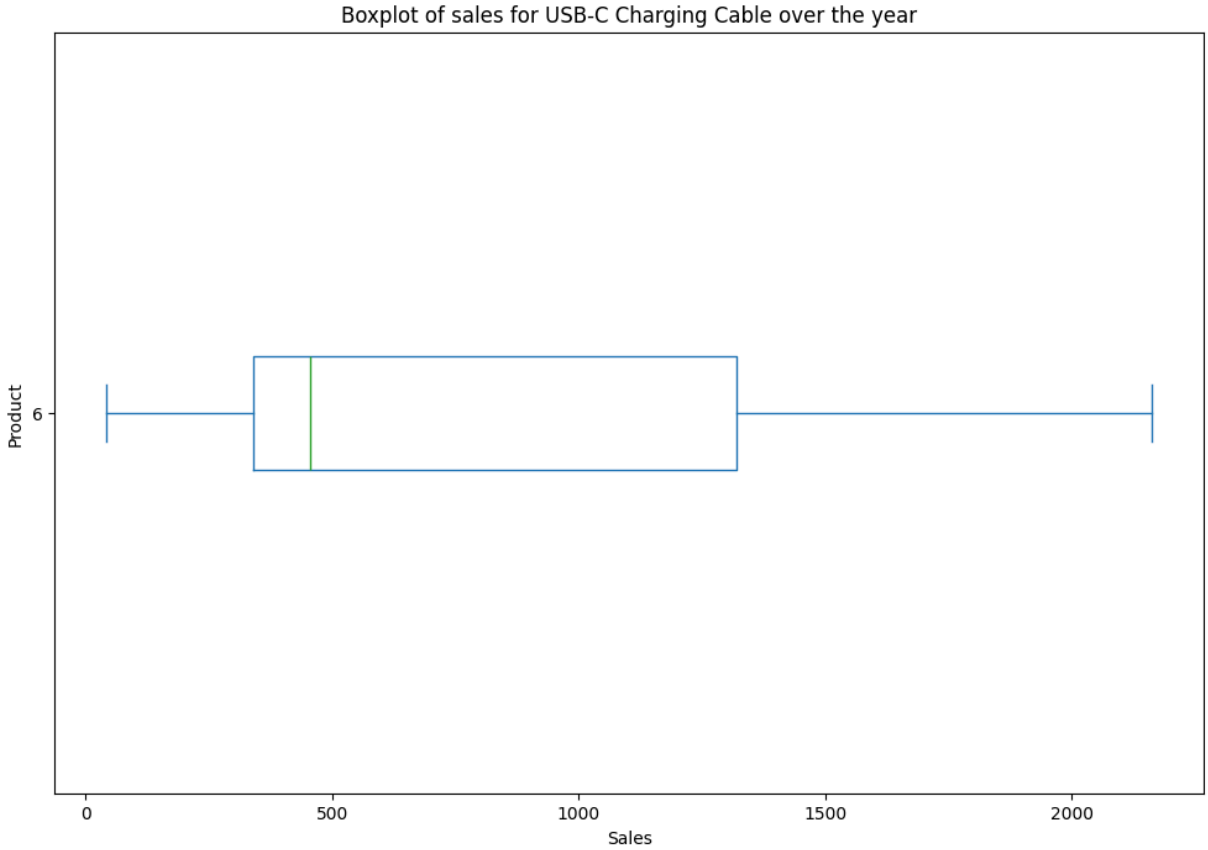

Boxplot of sales for Lightning Charging Cable over the year
Product
1
0
200
400
600
800
1000
1200
1400
1600
Sales

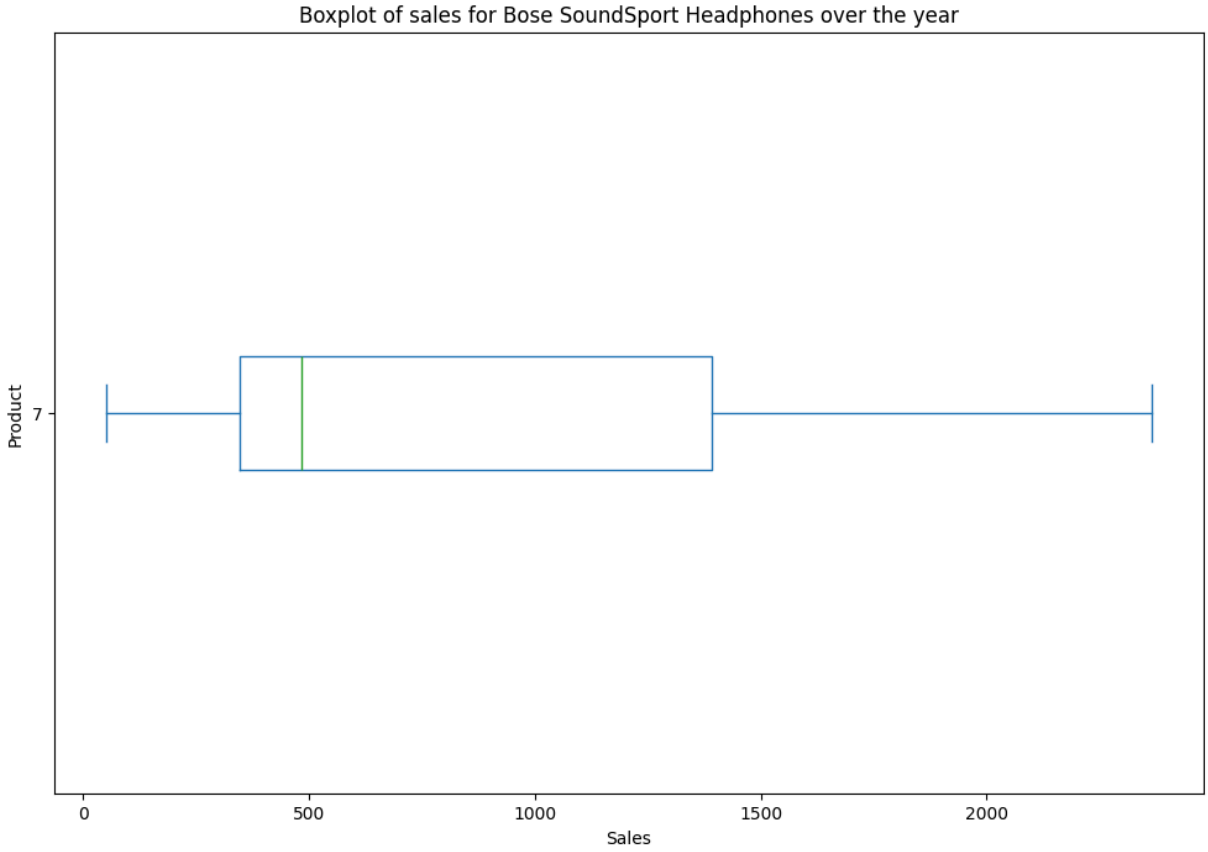

Boxplot of sales for Wired Headphones over the year
Product
2
0
250
500
750
1000
1250
1500
1750
2000
Sales

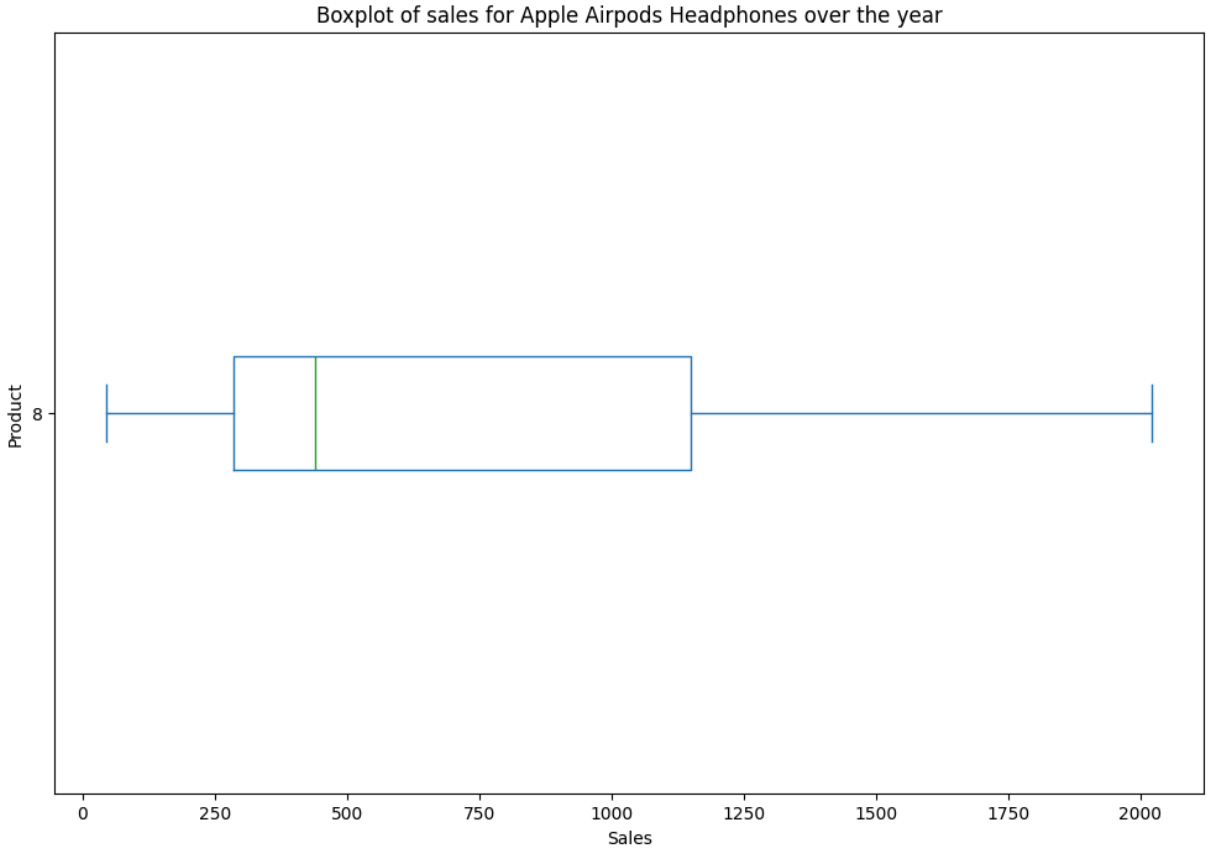

Boxplot of sales for 27in FHD Monitor over the year
Product
3
0
500
1000
1500
2000
2500
Sales

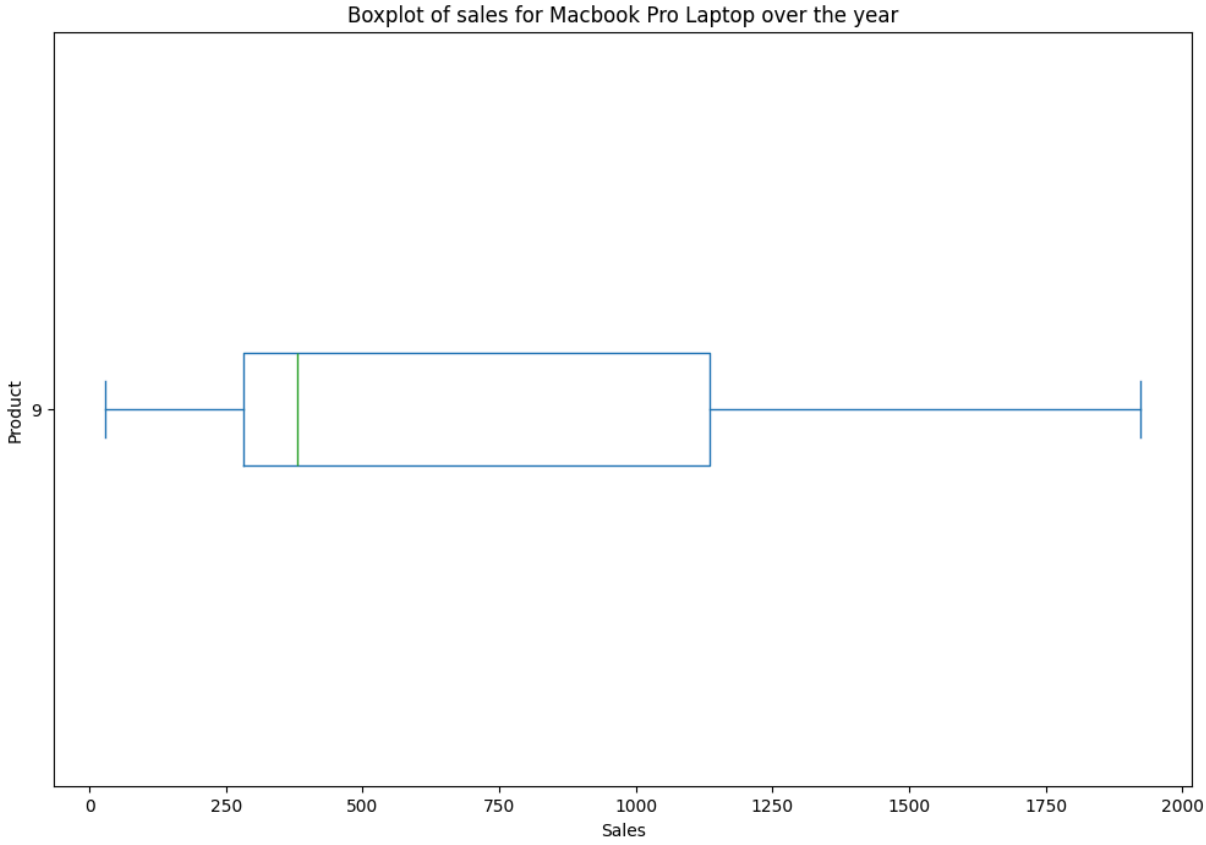

Boxplot of sales for AAA Batteries (4-pack) over the year
Product
4
0
500
1000
1500
2000
2500
3000
Sales

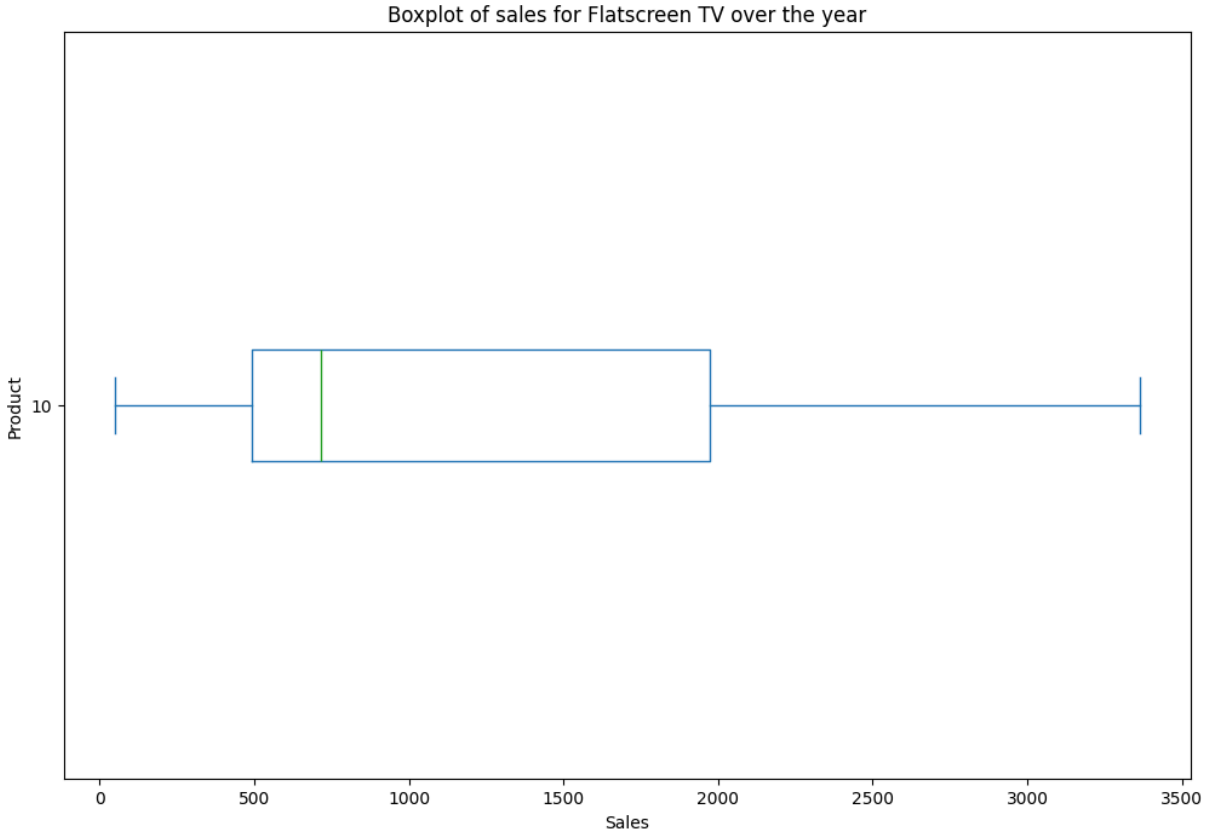

Boxplot of sales for 27in 4K Gaming Monitor over the year
Product
5
0
500
1000
1500
2000
2500
3000
Sales

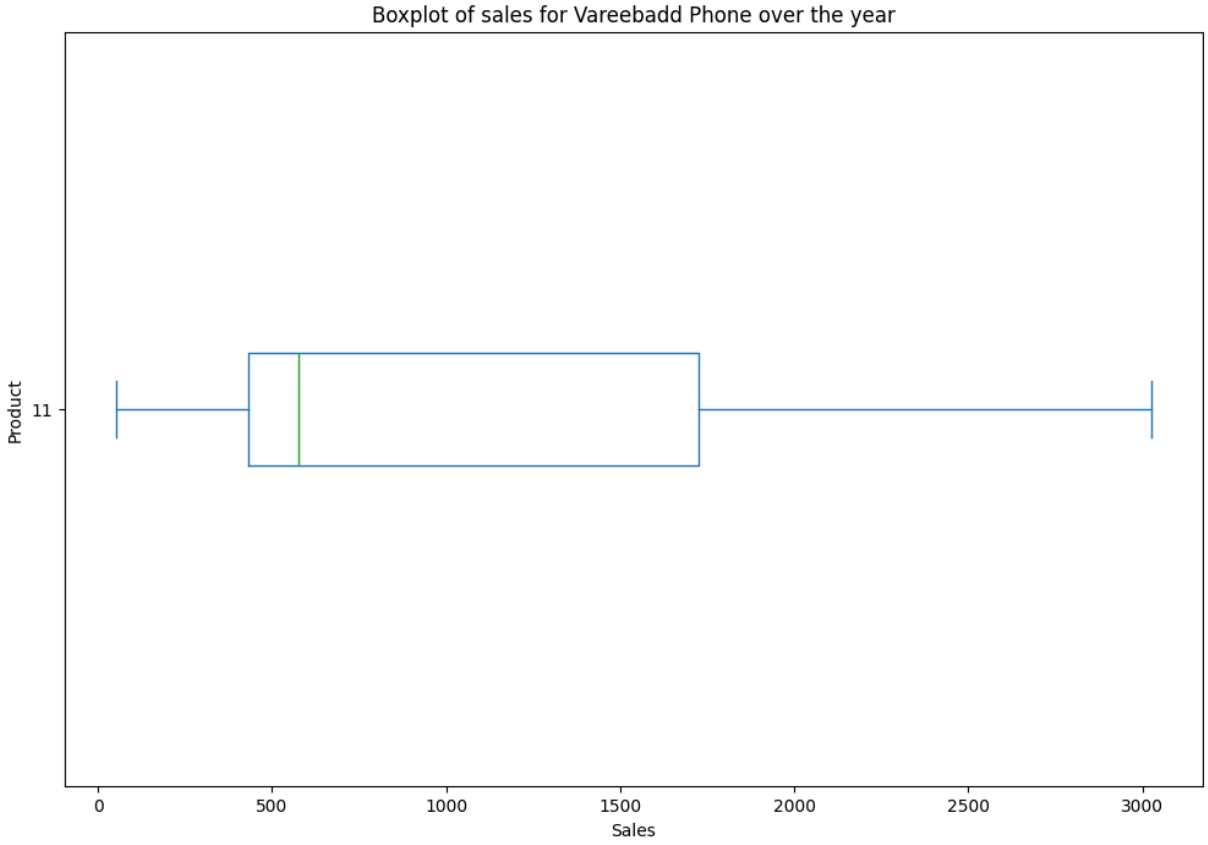

Boxplot of sales for USB-C Charging Cable over the year
Product
6
0
500
1000
1500
2000
Sales

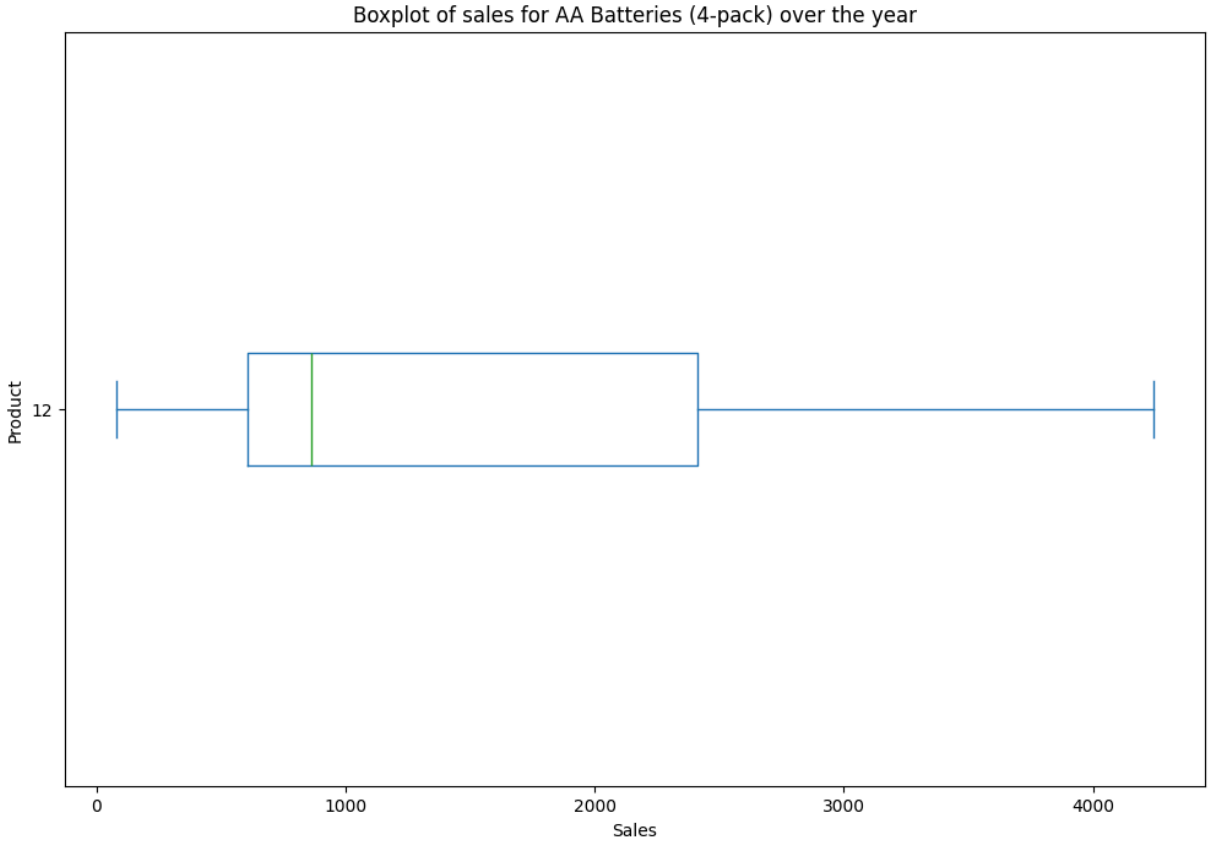

Boxplot of sales for Bose SoundSport Headphones over the year
Product
7
0
500
1000
1500
2000
Sales


Boxplot of sales for Apple Airpods Headphones over the year
Product
8
0
250
500
750
1000
1250
1500
1750
2000
Sales


Boxplot of sales for Macbook Pro Laptop over the year
Product
9
0
250
500
750
1000
1250
1500
1750
2000
Sales


Boxplot of sales for Flatscreen TV over the year
Product
10
0
500
1000
1500
2000
2500
3000
3500
Sales





d). Make a bubble graph that shows the relationship between the unit price of the products and their average monthly sales.

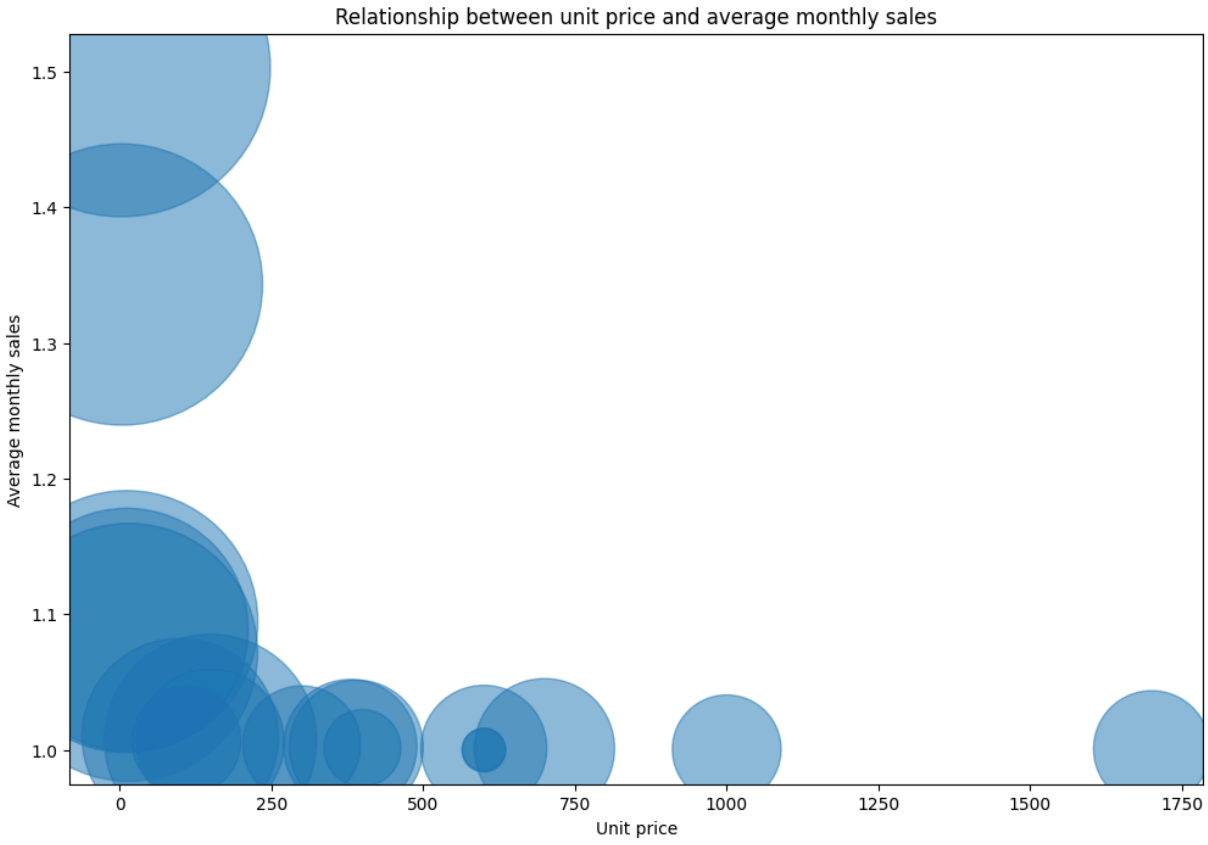
In [ ]:
```
# d). Make a bubble graph that shows the relationship between the unit price of the
# y axis: monthly sales
# x axis: unit price of each product
# bubble size: total sales

# Get the average monthly sales per product
averageMonthlySales = df.groupby("Product")["Quantity Ordered"].mean()
# averageMonthlySales

# Get the unit price of each product
unitPrice = df.groupby("Product")["Price Each"].mean()
# unitPrice

# Get the total sales per product
totalSales = df.groupby("Product")["Quantity Ordered"].sum()
# totalSales

plt.figure(figsize=(12, 8))
plt.scatter(unitPrice, averageMonthlySales, s=totalSales, alpha=0.5)
plt.title("Relationship between unit price and average monthly sales")
plt.ylabel("Average monthly sales")
plt.xlabel("Unit price")
plt.show()
```



4. Analysis and Conclusions: Based on the visualizations generated, provide a brief analysis of the data. What products had the highest sales?

Were there any months or periods when sales were noticeably high or low? Are there any trends or patterns in product sales?

What products had the highest sales?

Product | Quantity

- AAA Batteries (4-pack) | 31017
- AA Batteries (4-pack) | 27635
- USB-C Charging Cable | 23975
- Lightning Charging Cable | 23217
- Wired Headphones | 20557

Were there any months or periods when sales were noticeably high or low?

Yes, in the December-January period there was a peak in sales

Are there any trends or patterns in product sales?

Yes, people tend to buy more items when their unit price is lower.

In [ ]:
```
# What products had the highest sales?
totalSales.sort_values(ascending=False)[:5]
```

Out[ ]:
```
Product
AAA Batteries (4-pack)      31017
AA Batteries (4-pack)       27635
USB-C Charging Cable        23975
Lightning Charging Cable    23217
Wired Headphones            20557
Name: Quantity Ordered, dtype: int64
```