Exercises

Bike store sales

Introduction to Data Analysis Welcome to the world of data analysis! In this course, we will embark on an exciting journey to explore, analyze, and derive meaningful insights from datasets. The process of data analysis can be broadly categorized into three key tasks, each playing a crucial role in unraveling the stories hidden within the data.

1. Load Dataset Loading the dataset is our starting point. Before we can dive into analysis, we need data. This step involves importing datasets into our analytical environment, whether it be a programming language like Python or a tool like Excel. Loading the dataset allows us to familiarize ourselves with the raw information we'll be working with—understanding the structure, format, and types of data.

2. Clean Dataset Cleaning the dataset is where we refine our raw data into a polished gem. Real-world data can be messy, containing errors, missing values, or inconsistencies. Cleaning involves handling these imperfections, ensuring that our dataset is accurate, complete, and ready for analysis. We'll address issues such as missing data, duplicates, outliers, and formatting discrepancies during this crucial stage.

3. Visualize Data Visualization breathes life into our data. Through visual representation, we can grasp patterns, trends, and relationships within the dataset more effectively than with raw numbers alone. Visualization tools, such as charts, graphs, and plots, will be our

artistic instruments to communicate insights to others and gain a deeper understanding of the data ourselves.

As we progress through these tasks, you'll not only develop practical skills in data manipulation and analysis but also cultivate a keen sense of curiosity and critical thinking. Remember, each dataset has its unique challenges and stories waiting to be discovered. Let's embark on this data analysis adventure together!

Exploring Global Bike Store Sales Data

Welcome to the world of data analysis! In this scenario, we have a comprehensive dataset containing information about bike sales from a global perspective. The dataset covers sales in various countries, including Australia, Canada, France, Germany, the UK, and the US, spanning the years 2011 to 2016.

Task 1: Familiarizing with the Data

Our first task is to load the dataset and get an overview of its structure. This involves importing necessary libraries and examining the raw data. As we delve into the dataset, we'll discover details about customer age, order quantities, sales revenue, and more.

Task 2: Exploring Customer and Order Statistics

Now, let's dive into the data analysis. We want to understand our customers better, starting with their ages and order quantities. We'll calculate the mean age of customers and visualize the distribution through density (KDE) and box plots. Additionally, we'll explore the mean order quantity and represent it with a histogram and box plot.

Task 3: Analyzing Sales Trends

Moving on, we aim to uncover trends in sales over the years. We'll count the number of sales per year and represent this information using a pie plot. Following that, we'll break down sales per month, presenting the results with a bar plot.

Task 4: Regional Sales Comparison

Our analysis extends to exploring sales on a global scale. We'll identify the country with the highest quantity of sales and visualize the sales distribution across different countries using a bar plot. Additionally, we'll compile a list of every product sold and showcase the top 10 best-selling products through a bar plot.

Task 5: Relationship Exploration

Time to investigate relationships within the dataset. We'll examine the correlation between unit cost and unit price through a scatter plot. Similarly, we'll explore relationships between order quantity and profit, presenting our findings in a scatter plot. Further, we'll delve into profit variations across countries and customer ages using grouped box plots.

Task 6: Temporal Analysis

Our analysis evolves to include a temporal dimension. We'll create a new column, 'Calculated_Date,' using day, month, and year information. Following that, we'll convert this column into a datetime object. Finally, we'll visualize the evolution of sales through the years using a line plot.

Task 7: Revenue Adjustment and Further Analysis

As a final touch, we'll introduce a revenue adjustment by increasing each sale's revenue by $50. This will impact our subsequent analysis. We'll explore specific scenarios, such as the number of orders in Canada or France, Bike Racks orders in Canada, and sales in each region of France. Additionally, we'll examine sales per category and sub-categories, emphasizing visualization through pie and bar plots.

By the end of this analysis, you will not only gain practical skills in data manipulation and visualization but also develop a deeper understanding of global bike sales trends. Let's embark on this data analysis journey together!

Load data and check if there are any Null

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import geopandas as gpd
```

```
data = pd.read_csv('/content/sales_data (3).csv')
data
```

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country
0	2013-11-26	26	November	2013	19	Youth (<25)	M	Canada
1	2015-11-26	26	November	2015	19	Youth (<25)	M	Canada
2	2014-03-23	23	March	2014	49	Adults (35-64)	M	Australia
3	2016-03-23	23	March	2016	49	Adults (35-64)	M	Australia
4	2014-05-15	15	May	2014	47	Adults (35-64)	F	Australia
...	...	...	...	...	...	...	...	...
113031	2016-04-12	12	April	2016	41	Adults (35-64)	M	United Kingdom
113032	2014-04-02	2	April	2014	18	Youth (<25)	M	Australia
113033	2016-04-02	2	April	2016	18	Youth (<25)	M	Australia
113034	2014-03-04	4	March	2014	37	Adults (35-64)	F	France
113035	2016-03-04	4	March	2016	37	Adults (35-64)	F	France

113036 rows × 18 columns

```
data.shape
```

```
(113036, 18)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113036 entries, 0 to 113035
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Date              113036 non-null  object
 1   Day               113036 non-null  int64
 2   Month             113036 non-null  object
 3   Year              113036 non-null  int64
 4   Customer_Age      113036 non-null  int64
 5   Age_Group         113036 non-null  object
```

```
     6   Customer_Gender   113036 non-null  object
     7   Country           113036 non-null  object
     8   State             113036 non-null  object
     9   Product_Category  113036 non-null  object
     10  Sub_Category      113036 non-null  object
     11  Product           113036 non-null  object
     12  Order_Quantity    113036 non-null  int64
     13  Unit_Cost         113036 non-null  int64
     14  Unit_Price        113036 non-null  int64
     15  Profit            113036 non-null  int64
     16  Cost              113036 non-null  int64
     17  Revenue           113036 non-null  int64
    dtypes: int64(9), object(9)
    memory usage: 15.5+ MB
```

```
Df = data.dropna()
```

```
Df.tail()
```

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	
113031	2016-04-12	12	April	2016	41	Adults (35-64)	M	United Kingdom	
113032	2014-04-02	2	April	2014	18	Youth (<25)	M	Australia	Q
113033	2016-04-02	2	April	2016	18	Youth (<25)	M	Australia	Q
113034	2014-03-04	4	March	2014	37	Adults (35-64)	F	France	
113035	2016-03-04	4	March	2016	37	Adults (35-64)	F	France	

What's the mean of `Customers_Age` ?

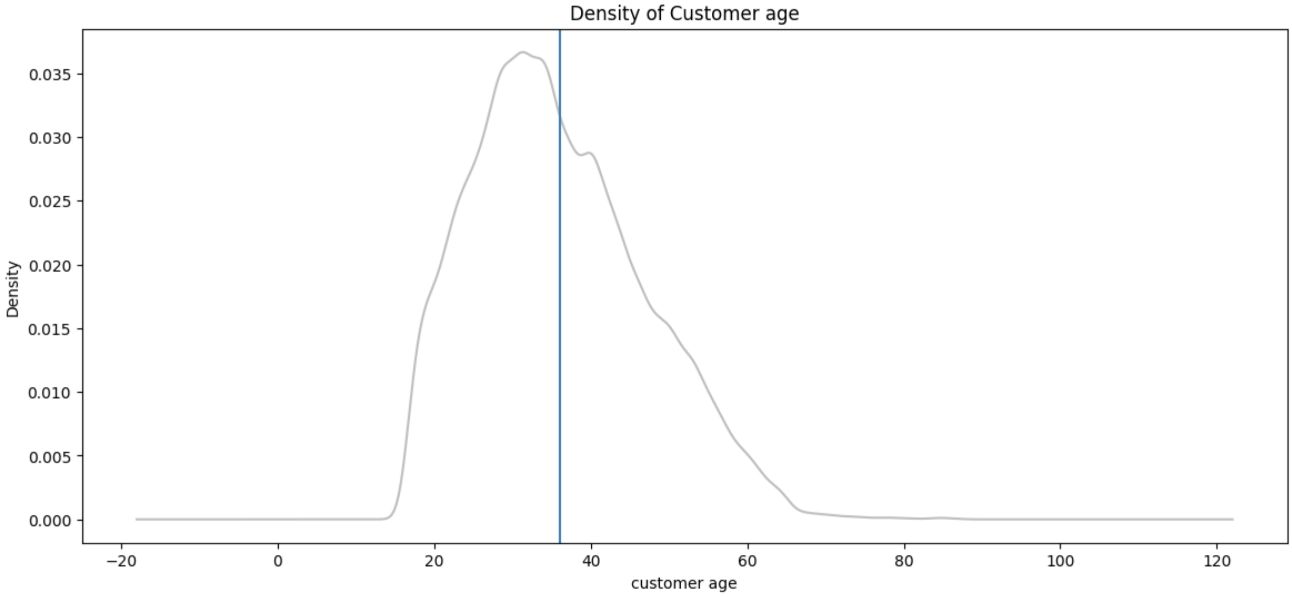
```
# your code goes here
M1 = Df['Customer_Age'].mean().round()
M1
```

```
    36.0
```

Go ahead and show a **density (KDE)** and a **box plot** with the `Customer_Age` data:

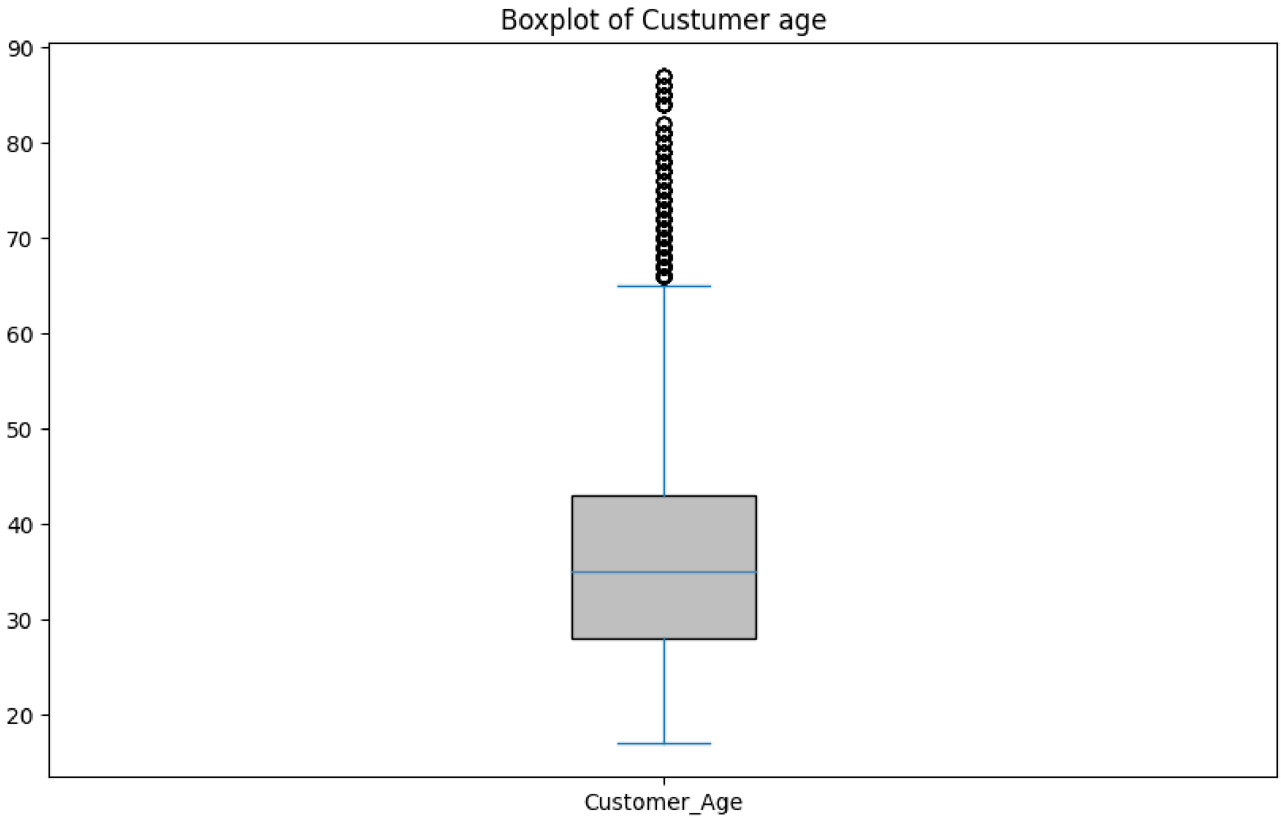
```
# your code goes here
a= Df['Customer_Age'].plot(kind= 'density', figsize= (14,6), color= '0.75')
a.axvline(M1).set_color('steelblue')
a.set_xlabel('customer age')
a.set_title('Density of Customer age')
```

```
Text(0.5, 1.0, 'Density of Customer age')
```



```
b = Df['Customer_Age'].plot(kind='box', figsize=(10,6), patch_artist=True, labels=['Custc
b.set_title('Boxplot of Custumer age')
```

```
Text(0.5, 1.0, 'Boxplot of Custumer age')
```



What's the mean of `Order_Quantity`?

```
# your code goes here
M2 = Df['Order_Quantity'].mean().round(2)
M2
```

```
     11.9
```

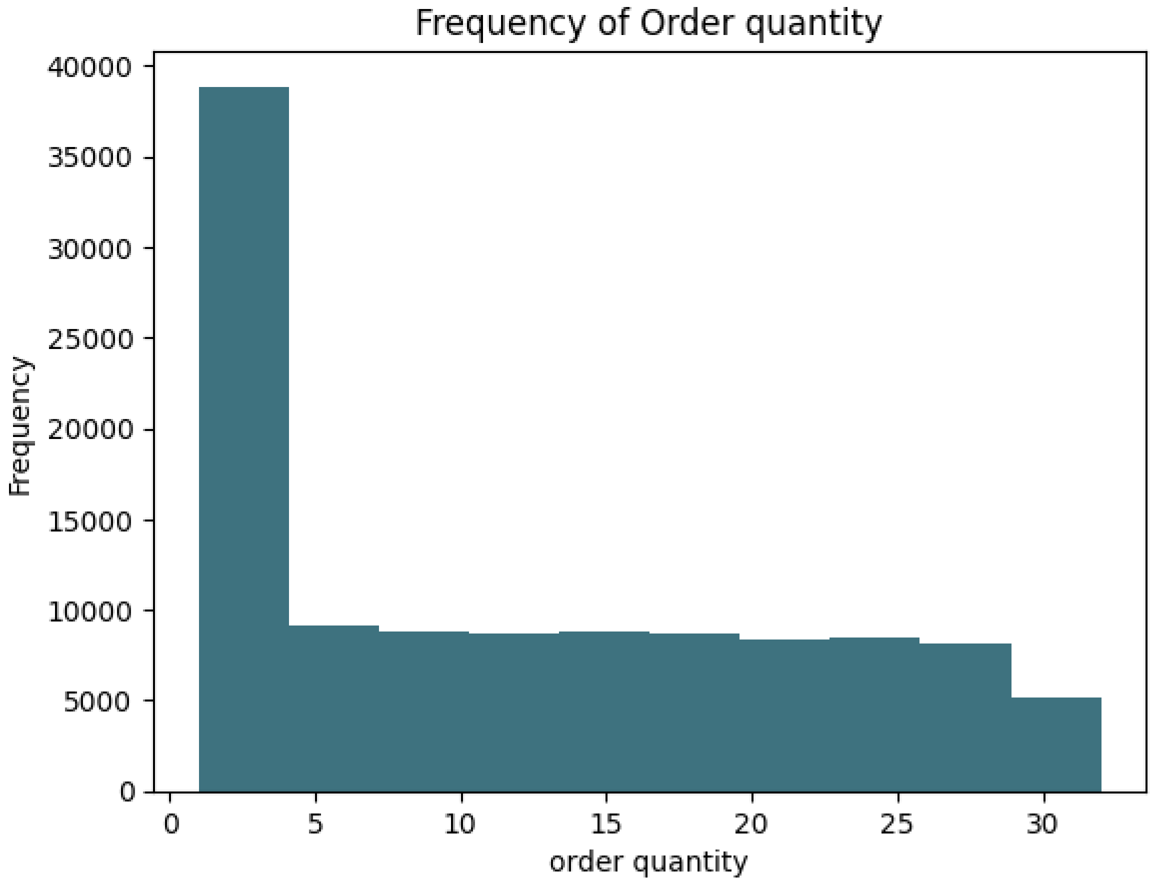
```
Df['Order_Quantity'].max().round(2)
```

```
     32
```

Go ahead and show a **histogram** and a **box plot** with the `Order_Quantity` data:

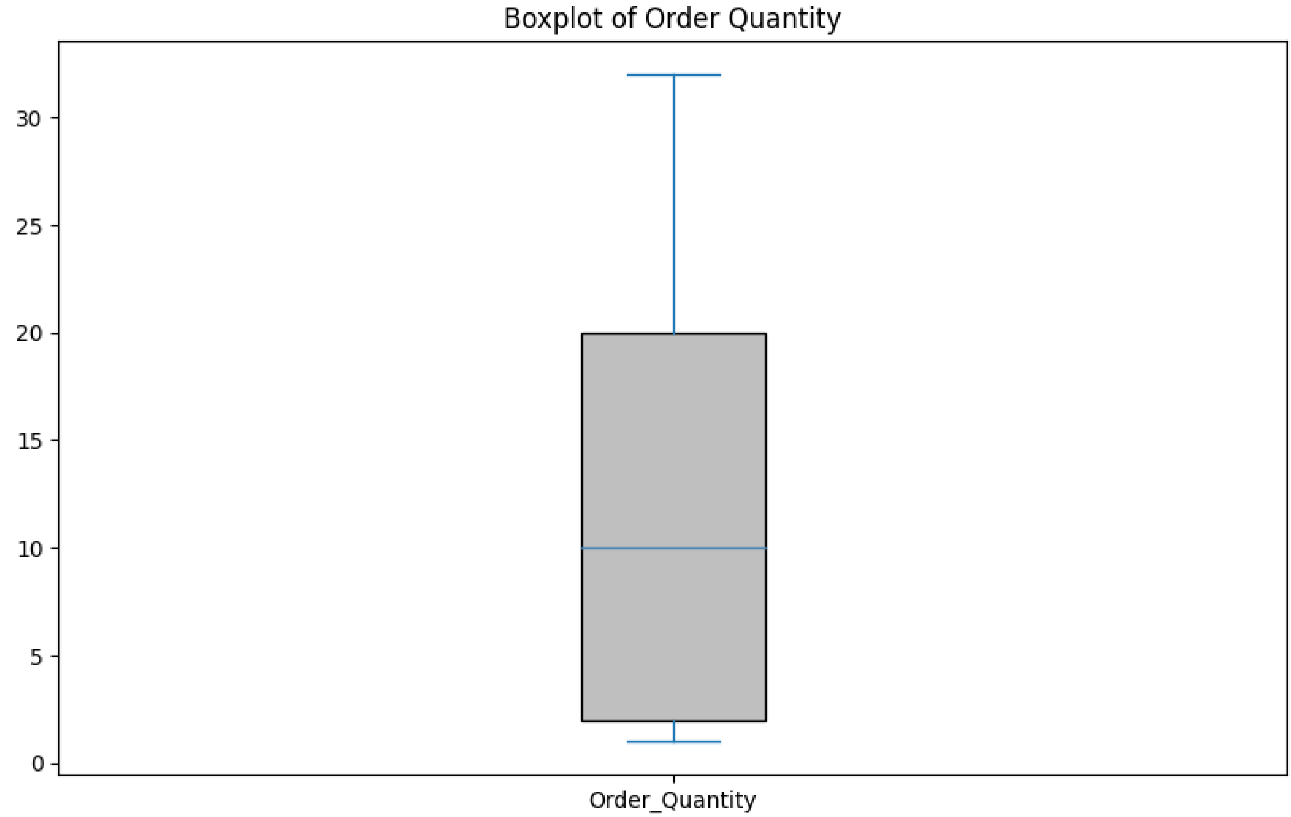
```
# your code goes here
c= Df['Order_Quantity'].plot(kind = 'hist', color= '#3E727F')
c.set_xlabel('order quantity')
c.set_title('Frequency of Order quantity')
```

```
     Text(0.5, 1.0, 'Frequency of Order quantity')
```



```
c= Df['Order_Quantity'].plot(kind='box', figsize=(10,6), patch_artist=True, labels=['Orde
c.set_title('Boxplot of Order Quantity')
```

```
Text(0.5, 1.0, 'Boxplot of Order Quantity')
```



How many sales per year do we have?

```
# your code goes here
sales_per_year = Df.groupby('Year')['Order_Quantity'].sum()
sales_per_year
```

```
Year
2011      5260
2012      5354
2013    294787
2014    379585
2015    289517
2016    370813
Name: Order_Quantity, dtype: int64
```

```
pd.crosstab(Df['Year'], Df['Order_Quantity'])
```

Order_Quantity	1	2	3	4	5	6	7	8	9	10	...	23	24	2
Year														
2011	1379	436	439	423	0	0	0	0	0	0	...	0	0	
2012	0	2677	0	0	0	0	0	0	0	0	...	0	0	
2013	6316	646	638	635	654	588	631	631	615	672	...	551	656	61
2014	5502	902	867	890	873	908	877	831	840	842	...	765	846	78
2015	4873	1477	1529	590	656	677	640	603	685	622	...	633	633	58
2016	4556	1512	1669	898	856	903	845	829	808	805	...	754	848	76

6 rows × 32 columns

```
Df.pivot_table(index='Year', values='Order_Quantity', aggfunc='sum')
```

	Order_Quantity
Year	
2011	5260
2012	5354
2013	294787
2014	379585
2015	289517
2016	370813

```
Total_salses = Df['Order_Quantity'].sum()
```

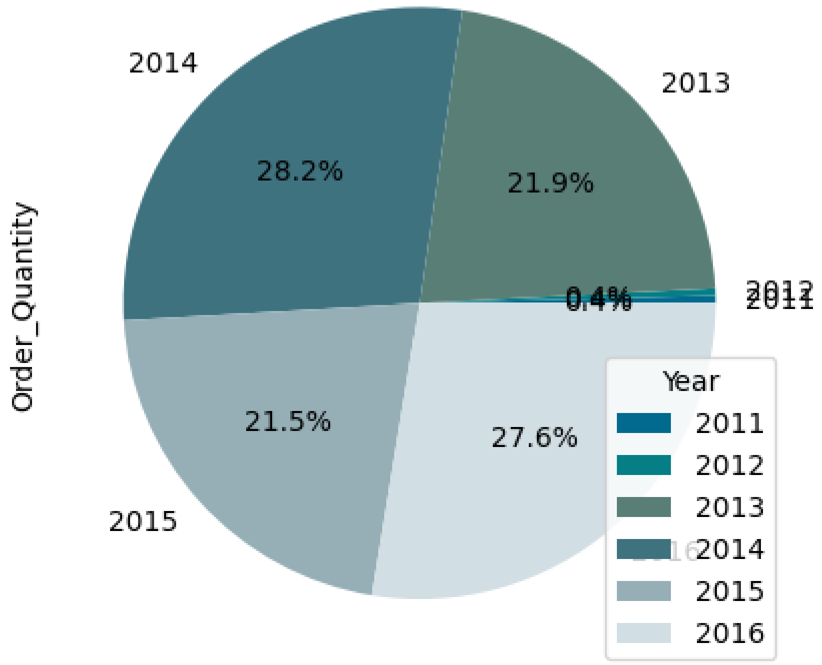
```
p= sales_per_year / Total_salses
percentage_of_sales_per_year = (p*100).round(2)
percentage_of_sales_per_year
```

```
Year
2011     0.39
2012     0.40
2013    21.91
2014    28.22
2015    21.52
2016    27.56
Name: Order_Quantity, dtype: float64
```

Go ahead and show a **pie plot** with the previous data:

```
# your code goes here
c = sales_per_year.plot(kind = 'pie', colors= ('#026A8F','#057E86','#597E76','#3E727F','#
c.legend(title="Year", loc="lower right")
```

```
<matplotlib.legend.Legend at 0x7c958b40ba00>
```



How many sales per month do we have?

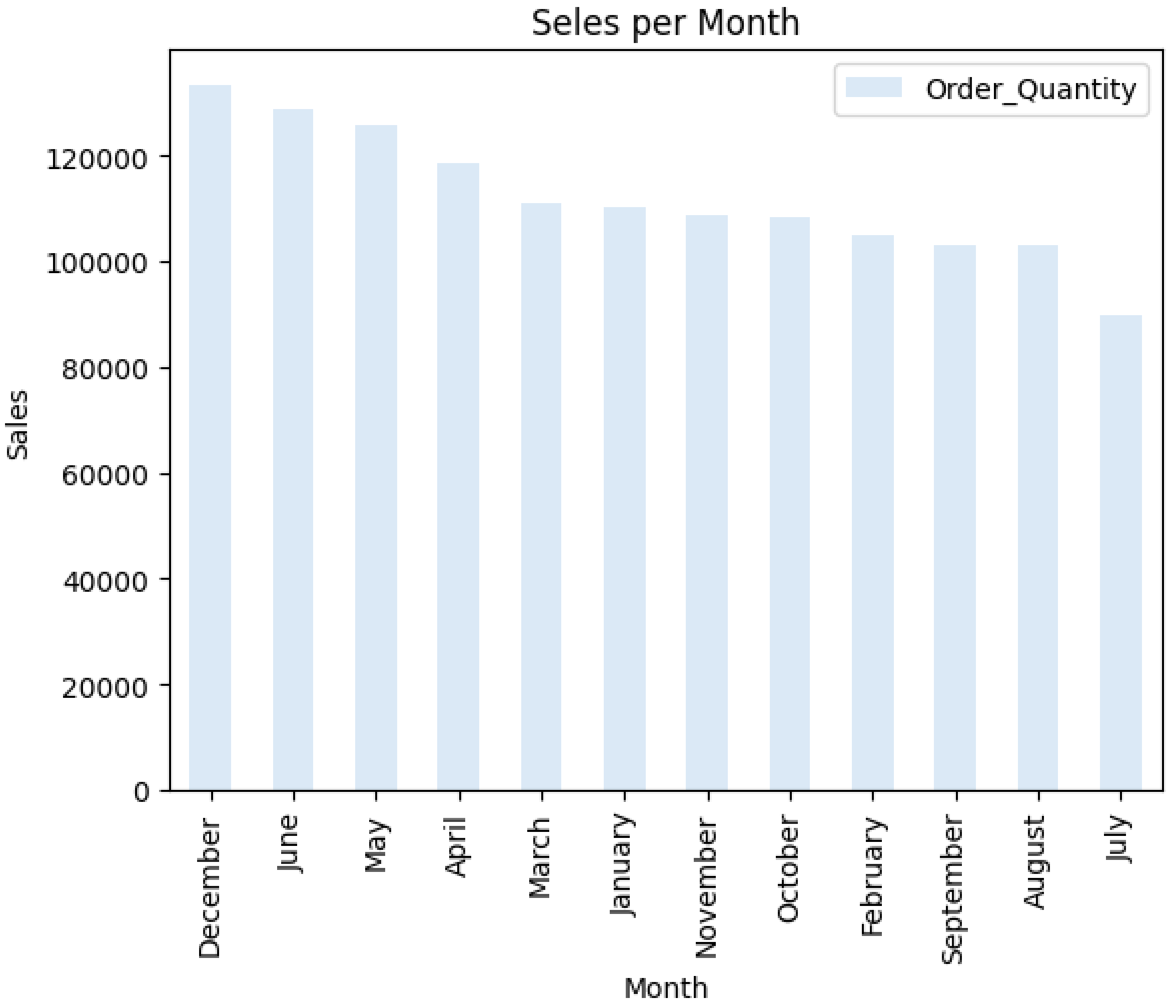
```
# your code goes here
sales_per_month = Df.pivot_table(index='Month', values='Order_Quantity', aggfunc='sum')
sort_sales = sales_per_month.sort_values(by='Order_Quantity', ascending=False)
sort_sales
```

	Order_Quantity
Month	
December	133312
June	128591
May	125715
April	118467
March	111085
January	110367
November	108637
October	108348
February	104717
September	103171
August	103119
July	89787

Go ahead and show a **bar plot** with the previous data:

```
# your code goes here
color = sns.color_palette('Blues')
e= sort_sales.plot(kind = 'bar', color= color)
e.set_xlabel('Month')
e.set_ylabel('Sales')
e.set_title('Seles per Month')
```

```
Text(0.5, 1.0, 'Seles per Month')
```



Which country has the most sales `quantity of sales`?

```
# your code goes here
sales_per_country = Df.pivot_table(index='Country', values='Order_Quantity', aggfunc='sum
sales_per_country
sort_country= sales_per_country.sort_values(by='Order_Quantity', ascending=False)
sort_country
```

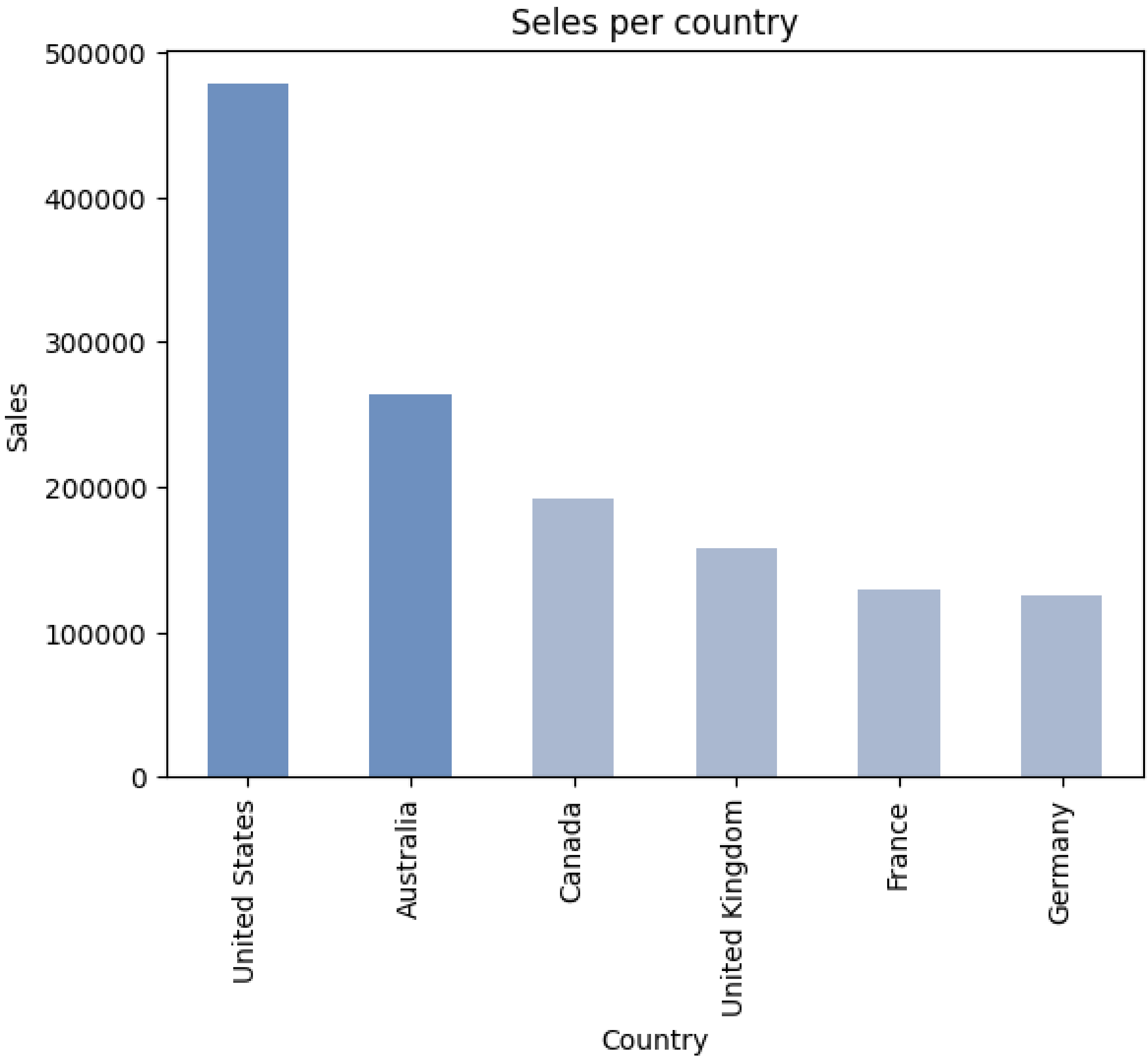
	Order_Quantity
Country	
United States	477539
Australia	263585
Canada	192259
United Kingdom	157218
France	128995
Germany	125720

Go ahead and show a **bar plot** of the sales per country:

```
# your code goes here

country_colors = {country: sns.color_palette('vlag')[i] for i, country in enumerate(Df['C
Df['Country_Color'] = Df['Country'].map(country_colors)
f = sort_country['Order_Quantity'].plot(kind='bar', color=Df['Country_Color'])
f.set_xlabel('Country')
f.set_ylabel('Sales')
f.set_title('Seles per country')
```

```
Text(0.5, 1.0, 'Seles per country')
```



Create a list of every product sold

```
# your code goes here
Product_sold = Df.pivot_table(index='Product', values='Order_Quantity', aggfunc='sum')
Product_sold
sort_sold= Product_sold.sort_values(by='Order_Quantity', ascending=False)
sort_sold
```

	Order_Quantity
Product	
Water Bottle - 30 oz.	164086
Patch Kit/8 Patches	157583
Mountain Tire Tube	102792
AWC Logo Cap	67316
Sport-100 Helmet, Red	63663
...	...
Mountain-100 Black, 42	73
Touring-3000 Blue, 50	70
Mountain-500 Silver, 48	52
Road-650 Red, 52	52
Mountain-500 Black, 52	40

130 rows × 1 columns

Create a **bar plot** showing the 10 most sold products (best sellers):

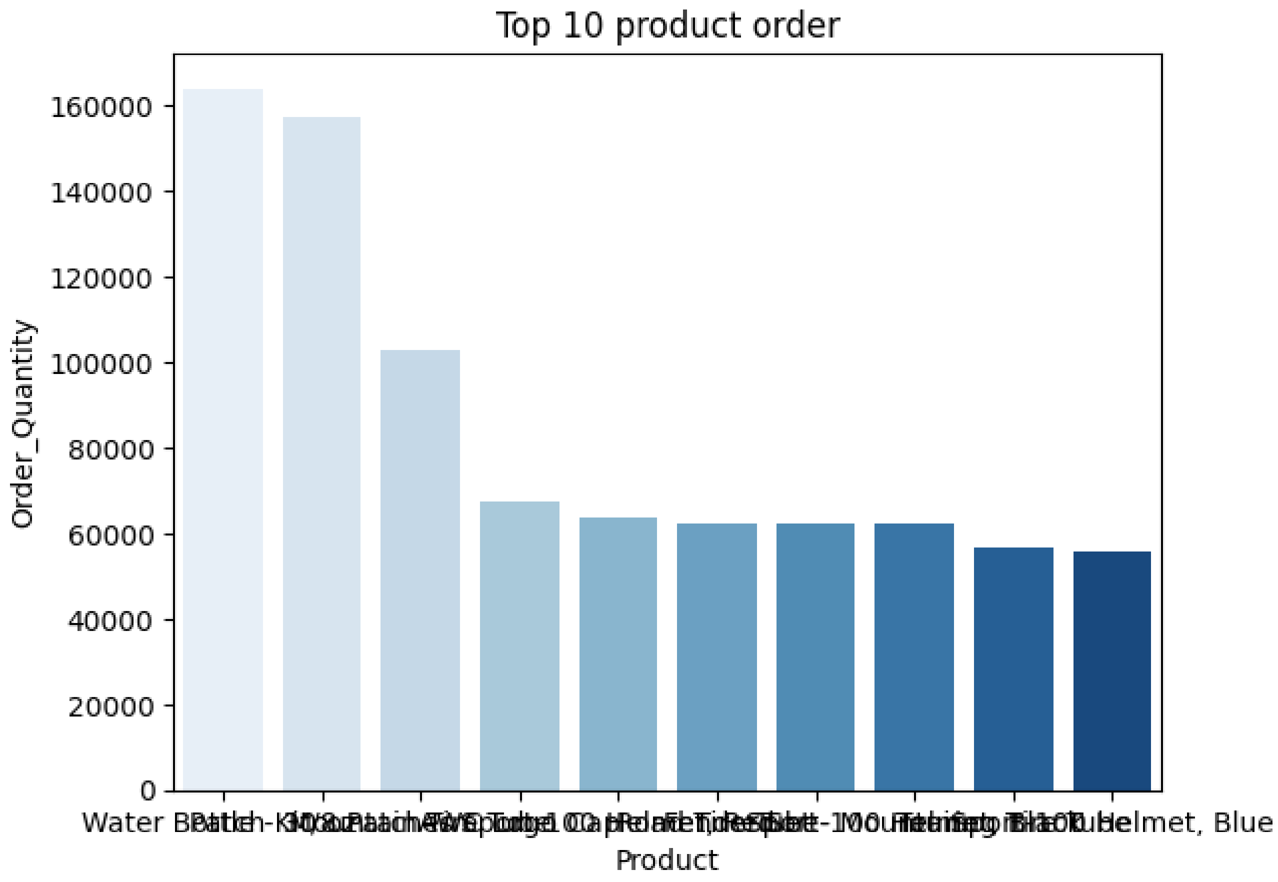
```
# your code goes here
top_10 = sort_sold.head(10)
top_10
```

	Order_Quantity
Product	
Water Bottle - 30 oz.	164086
Patch Kit/8 Patches	157583
Mountain Tire Tube	102792
AWC Logo Cap	67316
Sport-100 Helmet, Red	63663
Road Tire Tube	62296
Fender Set - Mountain	62118
Sport-100 Helmet, Black	62105
Touring Tire Tube	56802
Sport-100 Helmet, Blue	55895

```
sns.barplot(x=top_10.index, y='Order_Quantity', data = top_10, palette='Blues')

plt.xlabel('Product')
plt.ylabel('Order_Quantity')
plt.title('Top 10 product order')
```

```
    Text(0.5, 1.0, 'Top 10 product order')
```



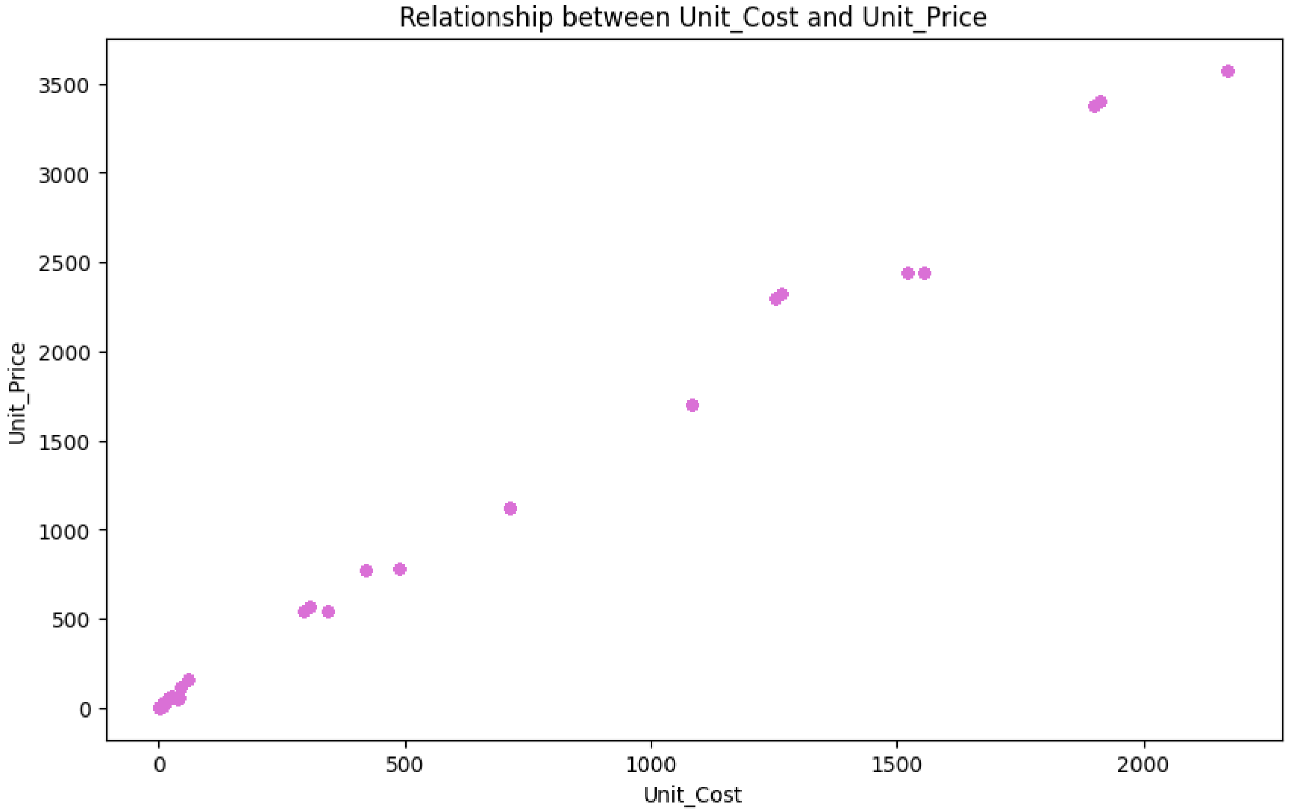
Can you see any relationship between `Unit_Cost` and `Unit_Price`?

Show a **scatter plot** between both columns.

```
Pofit_for_one= Df['Unit_Price']-Df['Unit_Cost']
```

```
j= Df.plot(kind= 'scatter', x='Unit_Cost', y='Unit_Price',figsize=(10, 6),color= 'orchid'
j.set_title('Relationship between Unit_Cost and Unit_Price')
j.set_xlabel('Unit_Cost')
j.set_ylabel('Unit_Price')
```

```
Text(0, 0.5, 'Unit_Price')
```

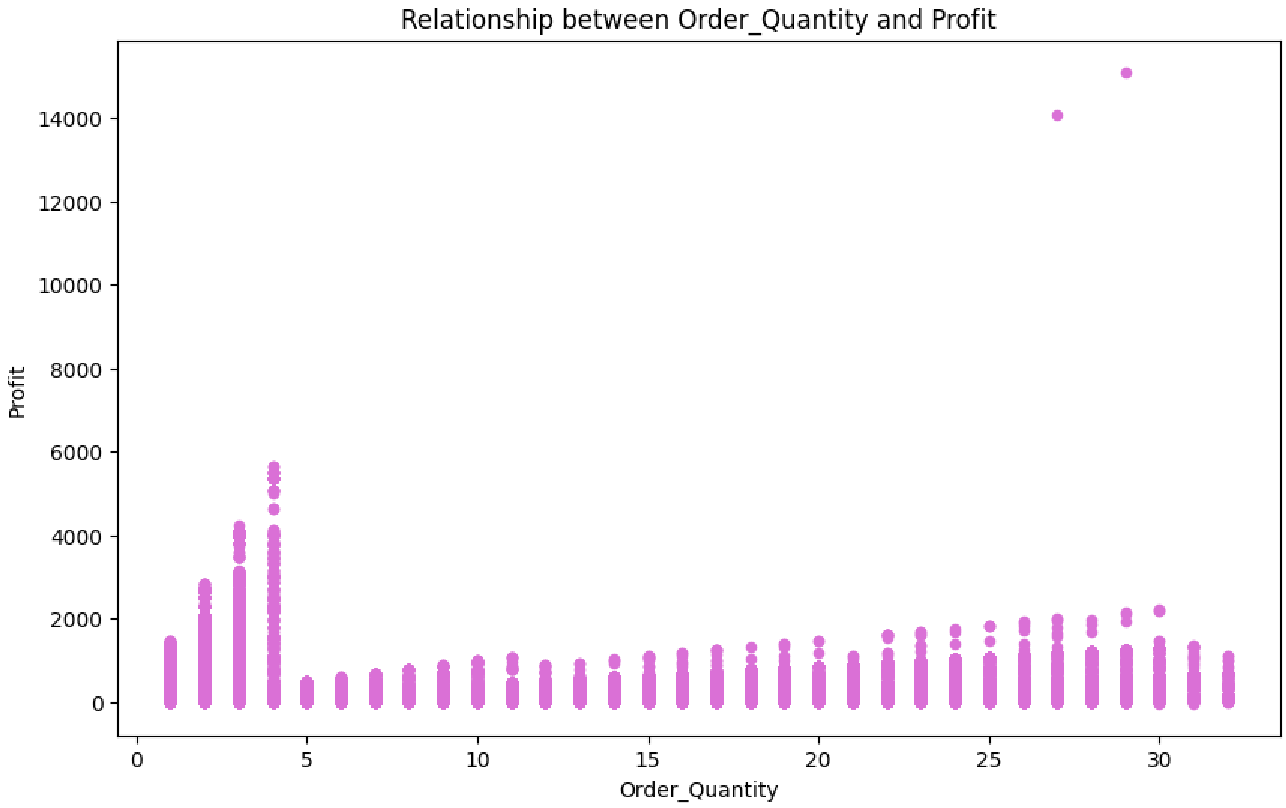


Can you see any relationship between `Order_Quantity` and `Profit`?

Show a **scatter plot** between both columns.

```
# your code goes here
# Profit = Unit_Price * Order_Quantity
k= Df.plot(kind= 'scatter', x= 'Order_Quantity', y= 'Profit', figsize= (10,6), color= 'or
k.set_title('Relationship between Order_Quantity and Profit')
k.set_xlabel('Order_Quantity')
k.set_ylabel('Profit')
```

```
Text(0, 0.5, 'Profit')
```



Relationship between Order_Quantity and Profit

Can you see any relationship between `Profit` per `Country`?

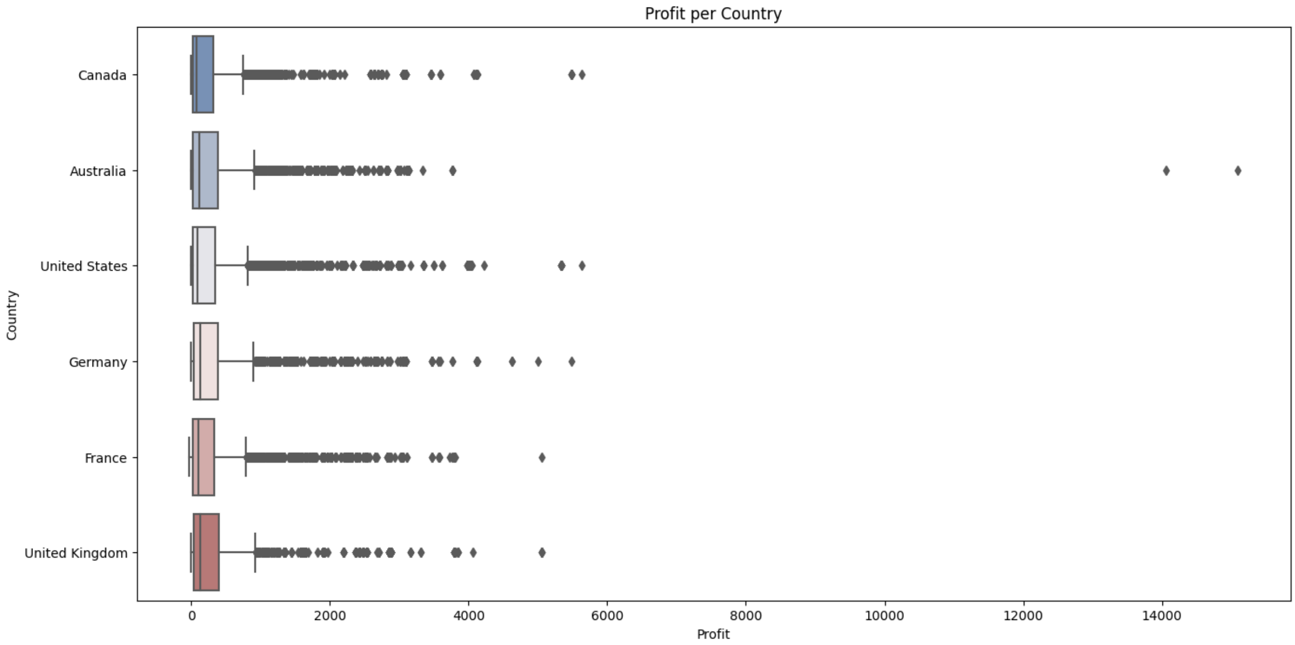
Show a grouped **box plot** per country with the profit values.

```
# your code goes here
profit_by_country = Df.groupby('Country')['Profit'].sum()
profit_by_country
```

```
Country
Australia          6776030
Canada             3717296
France             2880282
Germany            3359995
United Kingdom     4413853
United States     11073644
Name: Profit, dtype: int64
```

```
country_colors = {country: sns.color_palette('vlag')[i] for i, country in enumerate(Df['C
Df['Country_Color'] = Df['Country'].map(country_colors)

plt.figure(figsize=(16, 8))
sns.boxplot(x='Profit', y='Country', data=Df, palette=country_colors)
plt.title('Profit per Country')
plt.xlabel('Profit')
plt.ylabel('Country')
plt.show()
```



Can you see any relationship between the `Customer_Age` per `Country`?

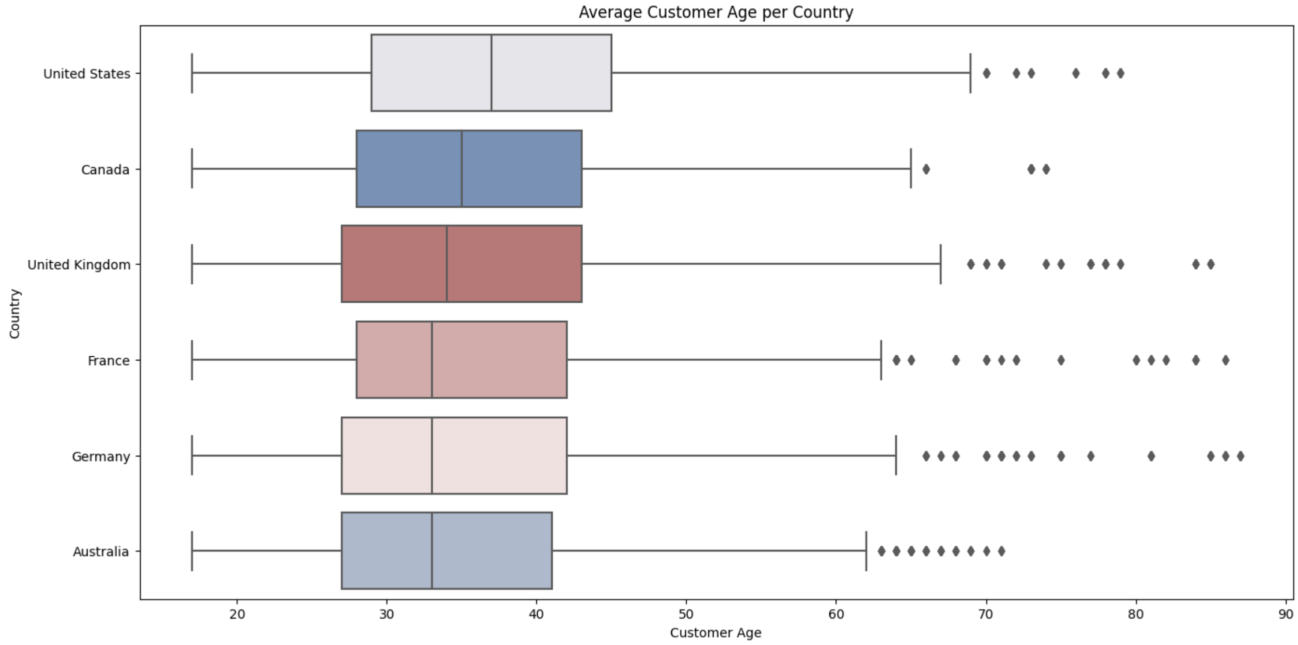
Show a grouped **box plot** per country with the customer age values.

```
# your code goes here
AVG_Age_per_Country = Df.pivot_table(index='Country', values='Customer_Age', aggfunc='mea
AVG_Age_per_Country
sort_AVG_Age_per_Country = AVG_Age_per_Country.sort_values(by='Customer_Age', ascending=F
sort_AVG_Age_per_Country
```

	Customer_Age
Country	
United States	37.391114
Canada	36.238962
United Kingdom	35.551836
France	35.116930
Germany	34.868084
Australia	34.383941

```
plt.figure(figsize=(16, 8))
sns.boxplot(x='Customer_Age', y='Country', data=Df, palette=country_colors, order=sort_AV
plt.title('Average Customer Age per Country')
plt.xlabel('Customer Age')
plt.ylabel('Country')
```

```
Text(0, 0.5, 'Country')
```



Add and calculate a new `Calculated_Date` column

Use `Day`, `Month`, `Year` to create a `Date` column (`YYYY-MM-DD`).

```
# your code goes here
Df['Day'] = Df['Day'].astype(str).str.zfill(2)
Df['Month'] = Df['Month'].astype(str)
Df['Year'] = Df['Year'].astype(str)

Df['Date'] = Df['Day'] + '-' + Df['Month'] + '-' + Df['Year']
```

Parse your `Calculated_Date` column into a datetime object

```
# your code goes here
Df['Calculated_Date'] = pd.to_datetime(Df['Date'], format='%d-%B-%Y')
```

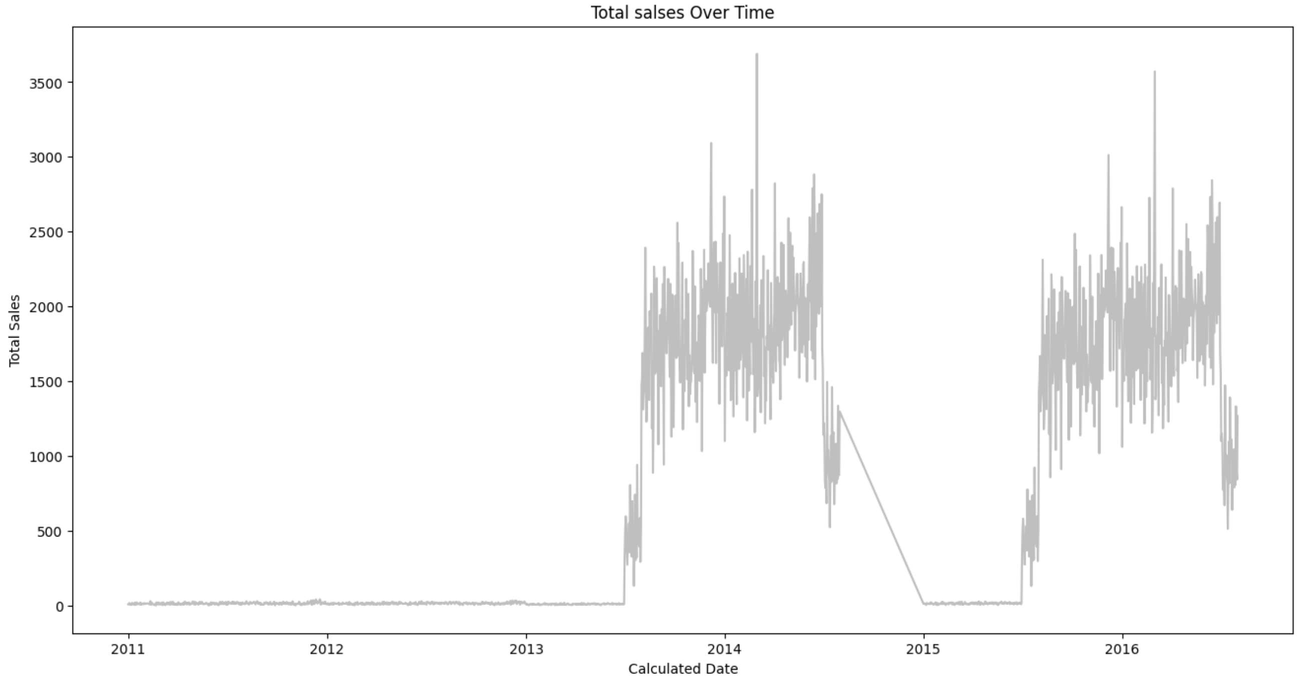
How did sales evolve through the years?

Show a **line plot** using `Calculated_Date` column as the x-axis and the count of sales as the y-axis.

```
# your code goes here
sales_per_date = Df.groupby('Calculated_Date')['Order_Quantity'].sum()


plt.figure(figsize=(16, 8))
sns.lineplot(x=sales_per_date.index, y=sales_per_date.values, color= '0.75')
plt.xlabel('Calculated Date')
plt.ylabel('Total Sales')
plt.title('Total salses Over Time')
```

```
Text(0.5, 1.0, 'Total salses Over Time')
```



Increase 50 U$S revenue to every sale

```
# your code goes here

Df['Increased_Revenue'] = Df['Revenue'] + 50

Df.head()
```

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country
0	26-November-2013	26	November	2013	19	Youth (<25)	M	Canada
1	26-November-2015	26	November	2015	19	Youth (<25)	M	Canada
2	23-March-2014	23	March	2014	49	Adults (35-64)	M	Australia
3	23-March-2016	23	March	2016	49	Adults (35-64)	M	Australia
4	15-May-2014	15	May	2014	47	Adults (35-64)	F	Australia

5 rows × 21 columns

How many orders were made in `Canada` or `France`?

```
# your code goes here# Assuming your DataFrame is named 'Df'
canada_france = Df[Df['Country'].isin(['Canada', 'France'])]
canada_france.groupby(['Order_Quantity', 'Country'])['Order_Quantity'].count()
```

```
Order_Quantity  Country
1               Canada     1646
                France     2408
2               Canada      763
                France      732
3               Canada      571
                           ...
30              France      221
31              Canada       71
                France       46
32              Canada       52
                France       24
Name: Order_Quantity, Length: 64, dtype: int64
```

```
canada_france.pivot_table(index='Country', values='Order_Quantity', aggfunc='count')
```

	Order_Quantity
Country	
Canada	14178
France	10998

How many `Bike Racks` orders were made from Canada?

```
bike_racks_canada_count = len(Df[(Df['Country'] == 'Canada') & (Df['Sub_Category'] == 'Bi
bike_racks_canada_count
```

```
     104
```

How many orders were made in each region (state) of France?

```
# your code goes here
orders_in_france_by_region = Df[Df['Country'] == 'France'].groupby('State')['Order_Quanti
orders_in_france_by_region
sort_orders_in_france_by_region= orders_in_france_by_region.sort_values( ascending =False
sort_orders_in_france_by_region
```

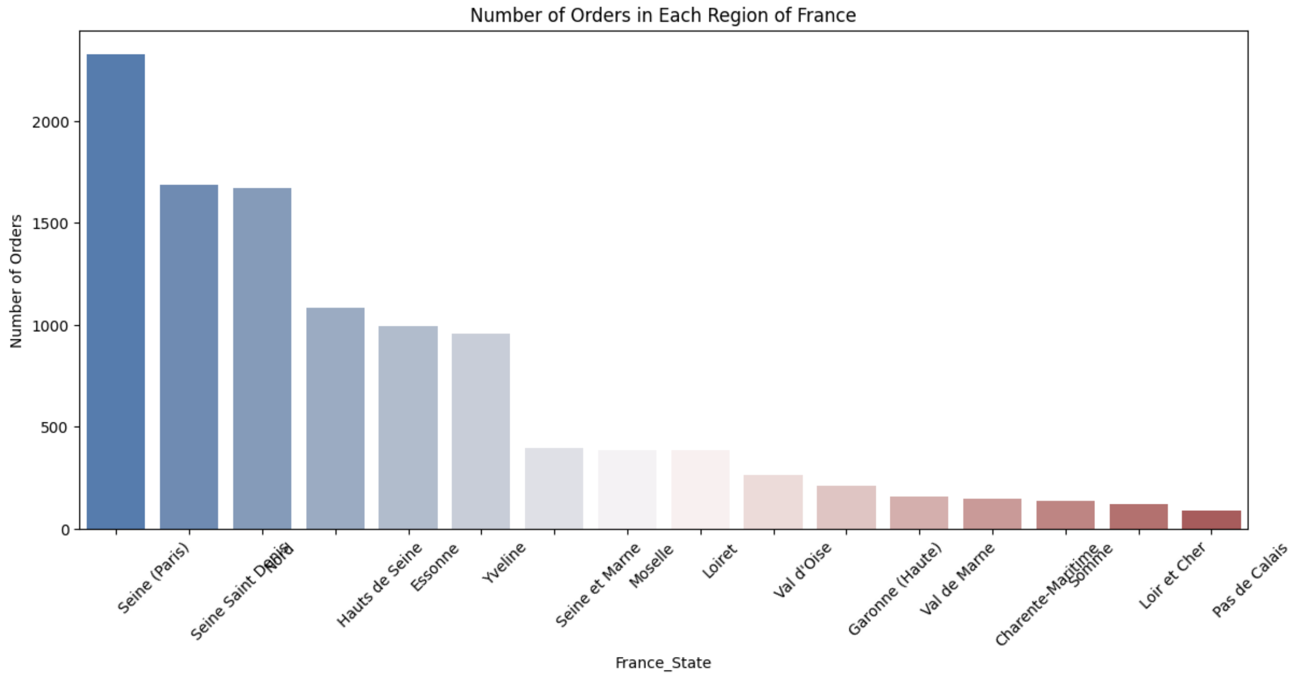
```
     State
     Seine (Paris)        2328
     Seine Saint Denis    1684
     Nord                 1670
     Hauts de Seine       1084
     Essonne               994
     Yveline               954
     Seine et Marne        394
     Moselle               386
     Loiret                382
     Val d'Oise            264
     Garonne (Haute)       208
     Val de Marne          158
     Charente-Maritime     148
     Somme                 134
     Loir et Cher          120
     Pas de Calais          90
     Name: Order_Quantity, dtype: int64
```

Go ahead and show a **bar plot** with the results:

```
plt.figure(figsize=(14, 6))
sns.barplot(x=sort_orders_in_france_by_region.index, y=sort_orders_in_france_by_region.va

plt.xlabel('France_State')
plt.ylabel('Number of Orders')
plt.title('Number of Orders in Each Region of France')
plt.xticks(rotation=45, ha='left')
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 [Text(0, 0, 'Seine (Paris)'),
  Text(1, 0, 'Seine Saint Denis'),
  Text(2, 0, 'Nord'),
  Text(3, 0, 'Hauts de Seine'),
  Text(4, 0, 'Essonne'),
  Text(5, 0, 'Yveline'),
  Text(6, 0, 'Seine et Marne'),
  Text(7, 0, 'Moselle'),
  Text(8, 0, 'Loiret'),
  Text(9, 0, "Val d'Oise"),
  Text(10, 0, 'Garonne (Haute)'),
  Text(11, 0, 'Val de Marne'),
  Text(12, 0, 'Charente-Maritime'),
  Text(13, 0, 'Somme'),
  Text(14, 0, 'Loir et Cher'),
  Text(15, 0, 'Pas de Calais')])
```



How many sales were made per category?

```
# your code goes here
sales_per_category= Df.pivot_table(index='Product_Category', values='Order_Quantity', agg
sales_per_category
```

	Order_Quantity
Product_Category	
Accessories	1054162
Bikes	36411
Clothing	254743

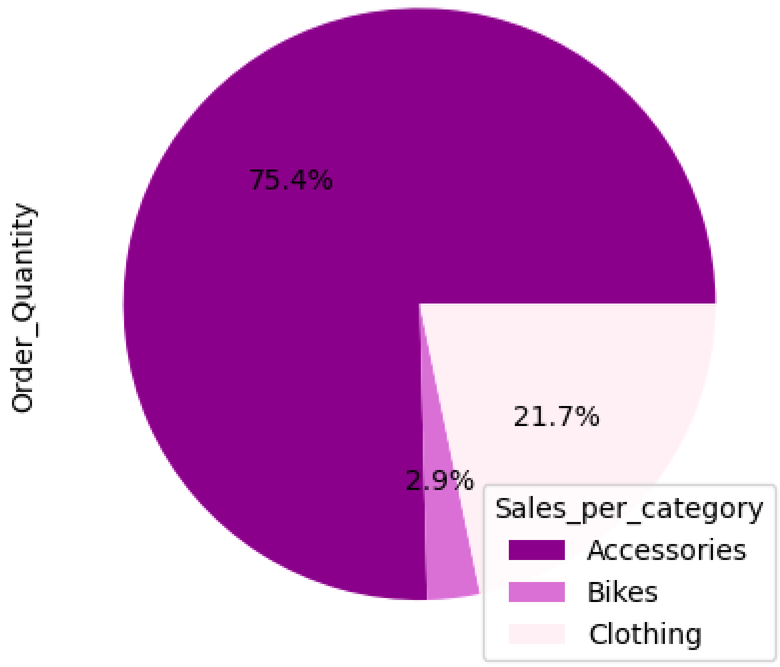
```
sales_per_category = Df.groupby('Product_Category')['Order_Quantity'].sum()
sales_per_category
```

```
Product_Category
Accessories    833461.0
Bikes           31763.0
Clothing       240217.0
Name: Order_Quantity, dtype: float64
```

Go ahead and show a **pie plot** with the results:

```
# your code goes here
colors = ['darkmagenta', 'orchid', 'lavenderblush']
n = sales_per_category.plot(kind='pie', colors=colors, autopct='%1.1f%%', labels=None)
plt.legend(labels=sales_per_category.index, title="Sales_per_category", loc="lower right"
```

```
<matplotlib.legend.Legend at 0x7fdbacbf29b0>
```



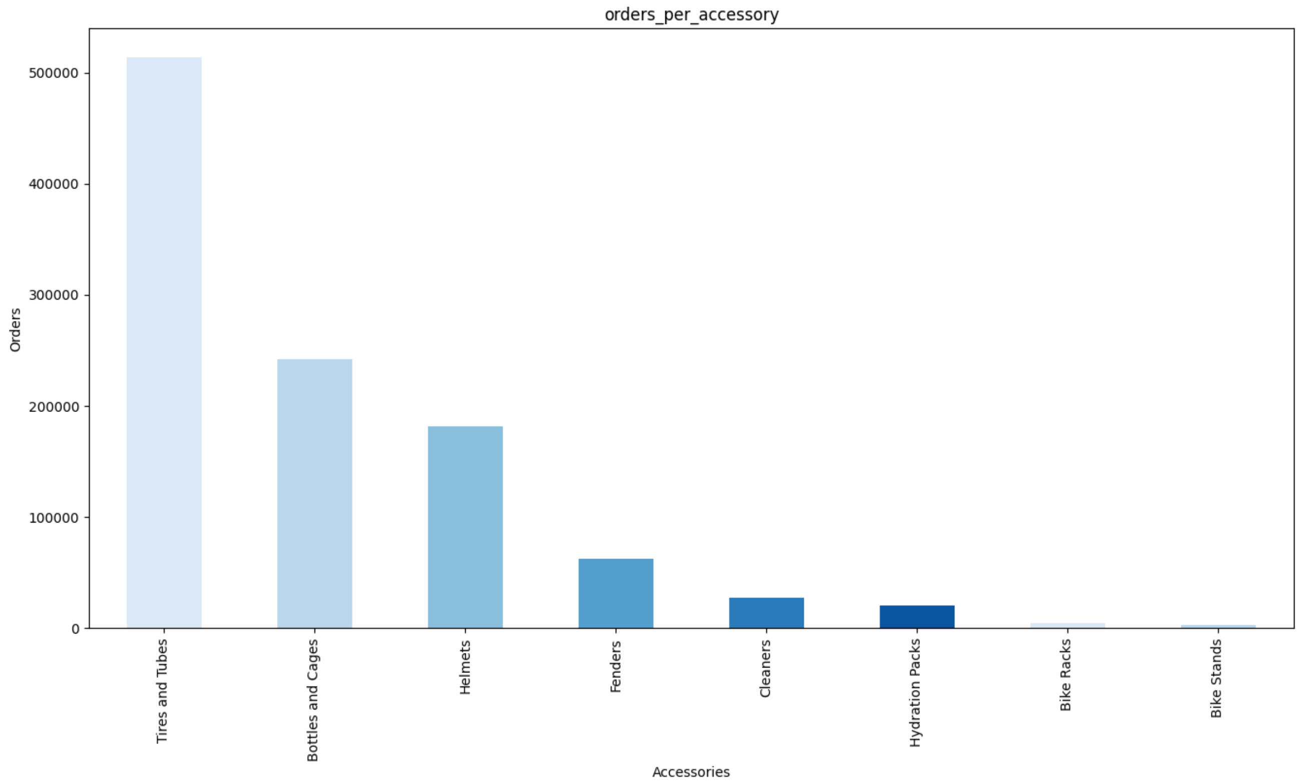
How many orders were made per accessory sub-categories?

```
# your code goes here
accessories_orders = Df[Df['Product_Category'] == 'Accessories']
orders_per_subcategory = accessories_orders.groupby('Sub_Category')['Order_Quantity'].sum
q = orders_per_subcategory.sort_values(ascending =False)
```

Go ahead and show a **bar plot** with the results:

```
# your code goes here
color = sns.color_palette('Blues')
p= q.plot(kind= 'bar', figsize=(16,8) , color = color)
p.set_xlabel('Accessories')
p.set_ylabel('Orders')
p.set_title('orders_per_accessory')
```

```
    Text(0.5, 1.0, 'orders_per_accessory')
```



How many orders were made per bike sub-categories?

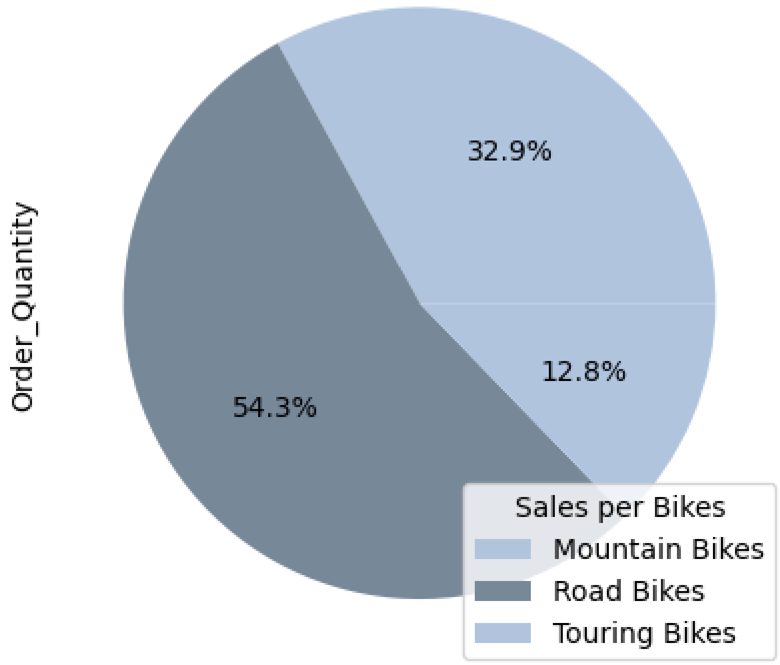
```
# your code goes here
Bike_orders = Df[Df['Product_Category'] == 'Bikes']
orders_per_Bike_subcategory = Bike_orders.groupby('Sub_Category')['Order_Quantity'].sum()
orders_per_Bike_subcategory
```

```
Sub_Category
Mountain Bikes    11992
Road Bikes        19771
Touring Bikes      4648
Name: Order_Quantity, dtype: int64
```

Go ahead and show a **pie plot** with the results:

```
# your code goes here
colors = ['lightsteelblue', 'lightslategray']
r = orders_per_Bike_subcategory.plot(kind='pie', colors=colors, autopct='%1.1f%%', labels
plt.legend(labels=orders_per_Bike_subcategory.index, title="Sales per Bikes", loc="lower
```

```
<matplotlib.legend.Legend at 0x7c95849536a0>
```



Which gender has the most amount of sales?

```
# your code goes here
Gender_sale= Df.pivot_table(index='Customer_Gender', values= 'Order_Quantity')
Gender_sale.sort_values(by='Order_Quantity', ascending=False)
```

	Order_Quantity
Customer_Gender	
M	11.997239
F	11.799814

How many sales with more than 500 in `Revenue` were made by men?

```
# your code goes here
sales_by_men = Df[(Df['Customer_Gender'] == 'M') & (Df['Revenue'] > 500)]
num_sales_by_men = len(sales_by_men)
num_sales_by_men

     19049
```

Get the top-5 sales with the highest revenue

```
# your code goes here
t = Df.pivot_table(index= 'Order_Quantity', values=('Revenue'))
t.sort values(by='Revenue' , ascending= False  ).head(5)
```