UNITAR INTERNATIONAL UNIVERSITY

MASTER OF INFORMATION TECHNOLOGY (ONLINE)

ITIM5113– Big Data Analytics Programming

Final Project

NAME : NORAZLINA BINTI OSMAN

MATRIC NO : MC221019559

LECTURER NAME : DR. HADI NAGHAVIPOUR

In this assignment, I use Python, Pandas, and Matplotlib to analyse and visualize about the bike sales. There is 113037 of original data (csv file), the data contain bike sales broken down by month, year, country, state, profit, revenue, unit cost and etc. I'm using jupyter notebook to do this analysis.

Part 1: Read and explore data

1. Importing the libraries and dataset
   First, we will import the necessary library then we use panda to read the csv file and read a data frame from it. There is 113037 rows of data and 18 columns

In [1]:
```
# Import pandas
import pandas as pd

# Import matplotlib
import matplotlib.pyplot as plt

# Import numpy
import numpy as np

# Use pandas to read in sales_data_na.csv
sales_data_na = pd.read_csv('sales_data_na.csv')

# Print the shape
print(sales_data_na.shape)

# print(sales_data_na)
sales_data_na.head()
```

```
(113037, 18)
```

Out[1]:

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quanti
0	11/26/2013	26	November	2013	19.0	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks	NaN	
1	11/26/2015	26	NaN	2015	19.0	NaN	M	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	
2	3/23/2014	23	March	2014	NaN	Adults (35-64)	M	Australia	NaN	Accessories	Bike Racks	Hitch Rack - 4-Bike	
3	3/23/2016	23	March	2016	49.0	Adults (35-64)	M	Australia	New South Wales	Accessories	Bike Racks	Hitch Rack - 4-Bike	
4	5/15/2014	15	May	2014	47.0	Adults (35-64)	F	Australia	New South Wales	Accessories	Bike Racks	Hitch Rack - 4-Bike	

Part 2: Exploring the data

1. In this part we will look into the data types of each column more details. The data contain object, int64 and float64 data type.

In [2]:
```
# Print .dtypes
print(sales_data_na.dtypes)

# Exclude data of type object
sales_data_na.describe(exclude=['object'])

# print(sales_data_na)
sales_data_na.head(2)
```

```
Date                object
Day                  int64
Month               object
Year                 int64
Customer_Age       float64
Age_Group           object
Customer_Gender     object
Country             object
State               object
Product_Category    object
Sub_Category        object
Product             object
Order_Quantity       int64
Unit_Cost          float64
Unit_Price         float64
Profit             float64
Cost               float64
Revenue            float64
dtype: object
```

From the output below Revenue value is not correctly calculated.

Out[2]:

Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
19.0	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks	NaN	8	45.0	120.0	590.0	NaN	950.0
19.0	NaN	M	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	8	45.0	NaN	590.0	360.0	950.0

I create a new column named '*New_Revenue*' to store correct value.

In [3]:
```
# Revenue value is not correctly calculated

# Create a new column name 'New_Revenue' with correct value
New_Revenue= sales_data_na['Order_Quantity'] * sales_data_na['Unit_Price']
sales_data_na['New_Revenue'] = New_Revenue
sales_data_na.head(2)
```

Since the value is not correctly calculated, I drop the existing Revenue column and renamed the new column.

In [4]:
```
# Since the Revenue column is not correctly calculated so dropped the Revenue column

sales_data_na = sales_data_na.drop('Revenue',axis=1)
```

In [5]:
```
# Renamed the new column as Revenue

sales_data_na = sales_data_na.rename(columns = {'New_Revenue':'Revenue'})
sales_data_na.head(2)
```

Now the value is correct.

Out[5]:

Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
19.0	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks	NaN	8	45.0	120.0	590.0	NaN	960.0
19.0	NaN	M	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	8	45.0	NaN	590.0	360.0	NaN

Since the Revenue value changed, this affected the Profit column value also. Same as revenue, I create a new column name New_Profit

In [6]:
```
# Since the Revenue value changed, this affected the Profit column value
# Same step as Revenue

# Create new column for Profit
New_Profit= sales_data_na['Revenue'] - sales_data_na['Cost']

# Insert the new column beside the existing Profit column
sales_data_na.insert(loc = 15,column = 'New_Profit', value = New_Profit)
sales_data_na.head(3)
```

Out[6]:

ıp	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit_Cost	Unit_Price	New_Profit	Profit	Cost	Revenue
5)	M	Canada	British Columbia	Accessories	Bike Racks	NaN	8	45.0	120.0	NaN	590.0	NaN	960.0
ıN	M	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	8	45.0	NaN	NaN	590.0	360.0	NaN
5- 4)	M	Australia	NaN	Accessories	Bike Racks	Hitch Rack - 4-Bike	23	45.0	120.0	1725.0	1366.0	1035.0	2760.0

Then I remove the existing Profit column and rename the new column

In [7]:
```
# Since the Profit column is not correctly calculated so dropped the profit column

sales_data_na = sales_data_na.drop('Profit',axis=1)
```

In [8]:
```
# Renamed the column as Profit

sales_data_na = sales_data_na.rename(columns = {'New_Profit':'Profit'})
sales_data_na.head(3)
```

Out[8]:

e	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
0	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks	NaN	8	45.0	120.0	NaN	NaN	960.0
0	NaN	M	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	8	45.0	NaN	NaN	360.0	NaN
N	Adults (35- 64)	M	Australia	NaN	Accessories	Bike Racks	Hitch Rack - 4-Bike	23	45.0	120.0	1725.0	1035.0	2760.0

Part 3: Data Cleaning

In this part I will do the data cleaning before visualize the data.

1. Missing value / empty value
   Now, we will check for overall column if there is missing values or not using isnull().sum() function

```
In [9]: # Checking for missing values using isnull()
        sales_data_na.isnull()

        # Print the total missing values in each column
        print(sales_data_na.isnull().sum())
```

As output below there is empty value in few variables.

```
Date                0
Day                 0
Month               1
Year                0
Customer_Age        1
Age_Group           1
Customer_Gender     0
Country             0
State               1
Product_Category    0
Sub_Category        0
Product             1
Order_Quantity      0
Unit_Cost           3
Unit_Price          1
Profit              2
Cost                1
Revenue             1
dtype: int64
```

I drop the missing value using dropna() function

```
In [10]: # Drop the missing value row
         remove_nan = sales_data_na.dropna()

         sales_data_na = remove_nan.copy()

         # Print again to check the missing values
         print(sales_data_na.isnull().sum())
```

Now missing value has been removed from the dataset.

```
Date                0
Day                 0
Month               0
Year                0
Customer_Age        0
Age_Group           0
Customer_Gender     0
Country             0
State               0
Product_Category    0
Sub_Category        0
Product             0
Order_Quantity      0
Unit_Cost           0
Unit_Price          0
Profit              0
Cost                0
Revenue             0
dtype: int64
```

The size of the data after drop the missing value reduced to (113031, 18)

```
In [11]: # Print the shape after drop the missing values
         print(sales_data_na.shape)

         (113031, 18)
```

2. Inconsistent column names
   Check for any inconsistent column name. I print out the column list in the dataset. The column name all are in order.

```
In [12]: # Create a variable name 'data_sales_columns'
         data_sales_columns = list(sales_data_na.columns)

         # Print all the columns name in the data
         print(data_sales_columns)

         ['Date', 'Day', 'Month', 'Year', 'Customer_Age', 'Age_Group', 'Customer_Gender', 'Country', 'State', 'Product_Category', 'Sub_C
         ategory', 'Product', 'Order_Quantity', 'Unit_Cost', 'Unit_Price', 'Profit', 'Cost', 'Revenue']
```

I do some modification, I replaced the ‘M’ and ‘F’ in the gender column to ‘Male’ and ‘Female’ and removed the columns ‘Day’ and ‘Date’, I will be using columns 'Month' and 'Year'.

```
In [13]: # Replaced the ‘M’ and ‘F’ in the gender column to ‘Male’ and ‘Female’
         Gender_replace = sales_data_na['Customer_Gender'].replace('M','Male', inplace = True)
         Gender_replace = sales_data_na['Customer_Gender'].replace('F','Female', inplace = True)

         print(sales_data_na['Customer_Gender'])


         3           Male
         4         Female
         5         Female
         6         Female
         7         Female
                    ...
         113032      Male
         113033      Male
         113034      Male
         113035    Female
         113036    Female
         Name: Customer_Gender, Length: 113031, dtype: object
```

From 18 column now our data become 16 columns

```
In [14]: #removed the columns ‘Day’ and ‘Date’
         sales_data_na.drop('Day', axis=1, inplace = True)
         sales_data_na.drop('Date', axis=1, inplace = True)

         sales_data_na.head()

         # Print the shape
         print(sales_data_na.shape)

         (113031, 16)
```

3. Duplicated row

   We will check if there is any duplicated row in the data

In [15]:
```
#Finding duplicate rows
duplicate_row = sales_data_na[sales_data_na.duplicated(keep = 'first')]

print("Duplicate Rows :")
duplicate_row
```

There are 6224 duplicated rows

```
Duplicate Rows :
```

Out[15]:

	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit
309	September	2015	33.0	Young Adults (25-34)	Male	Canada	British Columbia	Accessories	Bike Racks	Hitch Rack - 4-Bike	2	
1021	December	2013	22.0	Youth (<25)	Male	Australia	New South Wales	Accessories	Bike Stands	All-Purpose Bike Stand	9	
1091	September	2015	42.0	Adults (35-64)	Female	Australia	Victoria	Accessories	Bottles and Cages	Mountain Bottle Cage	5	
1093	October	2013	42.0	Adults (35-64)	Female	Australia	Victoria	Accessories	Bottles and Cages	Mountain Bottle Cage	2	
1095	October	2015	42.0	Adults (35-64)	Female	Australia	Victoria	Accessories	Bottles and Cages	Mountain Bottle Cage	1	
...	...	...	...	...	...	...	...	...	...	...	...	
112028	June	2016	32.0	Young Adults (25-34)	Male	Germany	Hamburg	Bikes	Touring Bikes	Touring-3000 Yellow, 50	1	
112047	October	2013	22.0	Youth (<25)	Male	United Kingdom	England	Bikes	Touring Bikes	Touring-2000 Blue, 46	1	
112167	April	2014	28.0	Young Adults (25-34)	Male	United Kingdom	England	Clothing	Vests	Classic Vest, S	19	
112168	April	2016	28.0	Young Adults (25-34)	Male	United Kingdom	England	Clothing	Vests	Classic Vest, S	17	
112969	March	2014	30.0	Young Adults (25-34)	Male	Australia	Queensland	Clothing	Vests	Classic Vest, M	11	

6224 rows × 16 columns

Now we will remove the duplicated row using drop function. Total 6224 rows had been deleted from the record, left 106807 rows

In [16]:
```
# Remove duplicated row
sales_data_na.drop_duplicates(inplace = True)

# Print the shape
print(sales_data_na.shape)
```

```
(106807, 16)
```

4. Untidy

In this section I will tidy up the dataset. As can see from the below summary statistics the value all are in positive value

```
In [17]: # Print the summary statistics on the variables
sales_data_na.describe()
```

Out[17]:

	Year	Customer_Age	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
count	106807.000000	106807.000000	106807.00000	106807.000000	106807.000000	106807.000000	106807.000000	106807.000000
mean	2014.421620	35.998268	12.26061	240.389731	407.805593	358.503469	444.099778	802.603247
std	1.265451	11.107585	10.19217	525.342733	880.939417	640.422843	871.102148	1466.689422
min	2011.000000	17.000000	1.00000	1.000000	2.000000	1.000000	1.000000	2.000000
25%	2013.000000	28.000000	3.00000	2.000000	5.000000	36.000000	27.000000	70.000000
50%	2014.000000	35.000000	11.00000	9.000000	24.000000	126.000000	99.000000	231.000000
75%	2016.000000	43.000000	20.00000	38.000000	55.000000	407.000000	380.000000	810.000000
max	2016.000000	87.000000	1200.00000	2171.000000	3578.000000	83688.000000	42978.000000	84000.000000

Then I make all the data type from float64 to int64 using .astype function.

```
In [18]: # change float to int
sales_data_na['Customer_Age'] = sales_data_na['Customer_Age'].astype('Int64')
sales_data_na = sales_data_na.astype({"Customer_Age":'int64', "Unit_Cost":'int64',"Unit_Price":'int64',
                                      "Profit":'int64',"Cost":'int64',"Revenue":'int64'})

# Print .dtypes
print(sales_data_na.dtypes)
```

```
Month               object
Year                 int64
Customer_Age         int64
Age_Group           object
Customer_Gender     object
Country             object
State               object
Product_Category    object
Sub_Category        object
Product             object
Order_Quantity       int64
Unit_Cost            int64
Unit_Price           int64
Profit               int64
Cost                 int64
Revenue              int64
dtype: object
```

I random check on 'Month' column

```
In [19]: # Check Month column
sales_data_na['Month'].value_counts()
```

As from output below, there are spelling mistake

```
Out[19]: June        10499
December    10494
May         10397
April        9587
March        9141
January      8768
February     8579
October      8344
November     8249
August       7814
September    7797
July         7131
Jone            3
Aogust          2
Marsh           1
Joly            1
Name: Month, dtype: int64
```

This could be the human error. There is total 7 rows of data with the spelling mistake of month.

In [20]:
```
# Select the wrong spelling Month variable

wrong_month  = ['Jone', 'Aogust', 'Joly','Marsh']

sales_data_na[sales_data_na.Month.isin(wrong_month)]
```

Out[20]:

	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit_Cost
108	Aogust	2013	42	Adults (35-64)	Male	United States	Oregon	Accessories	Bike Racks	Hitch Rack - 4-Bike	17	45
148	Marsh	2014	33	Young Adults (25-34)	Male	Australia	Queensland	Accessories	Bike Racks	Hitch Rack - 4-Bike	8	45
176	Jone	2014	55	Adults (35-64)	Male	United States	Washington	Accessories	Bike Racks	Hitch Rack - 4-Bike	9	45
177	Jone	2016	55	Adults (35-64)	Male	United States	Washington	Accessories	Bike Racks	Hitch Rack - 4-Bike	6	45
178	Jone	2014	55	Adults (35-64)	Male	United States	Oregon	Accessories	Bike Racks	Hitch Rack - 4-Bike	15	45
37838	Joly	2014	38	Adults (35-64)	Male	Canada	British Columbia	Accessories	Helmets	Sport-100 Helmet, Red	23	13
37851	Aogust	2015	31	Young Adults (25-34)	Male	Australia	Victoria	Accessories	Helmets	Sport-100 Helmet, Red	3	13

Since I will be using this column for data visualization later, I replace the spelling mistake to correct one.

In [21]:
```
# Replace the wrong name with new value
sales_data_na['Month'] = sales_data_na['Month'].replace(['Aogust', 'Marsh','Jone','Joly'], ['August', 'March','June','July'])
```

5. Outliers

   I'm using IQR, Interquartile Range method. It measures the statistical dispersion of the data values as a measure of overall distribution. IQR is equivalent to the difference between the first quartile (Q1) and the third quartile (Q3) respectively. Identifying Outliers with Interquartile Range (IQR). The lines of code below calculate and print the interquartile range for each of the variables in the dataset.

In [22]:
```
#IQR score

Q1 = sales_data_na.quantile(0.25)
Q3 = sales_data_na.quantile(0.75)

IQR = Q3 - Q1

# Print IQR for all column

print(IQR)

outliers = sales_data_na[((sales_data_na <(Q1-1.5*IQR)) | (sales_data_na >(Q3+1.5*IQR)))]

# If the value is not an outlier, it will display as NaN (not a number):
outliers.head(10)
```

```
Year                3.0
Customer_Age       15.0
Order_Quantity     17.0
Unit_Cost          36.0
Unit_Price         50.0
Profit            371.0
Cost              353.0
Revenue           740.0
dtype: float64
```

As we can see from below output, there is outliers in Profit, Cost and Revenue

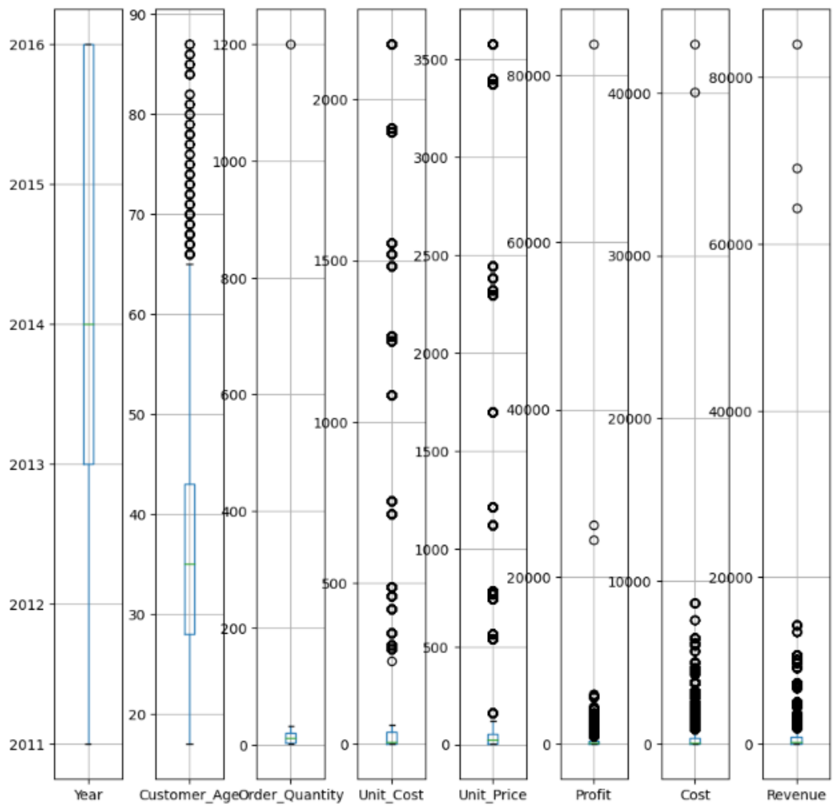
Out[22]:

r_Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1500.0	NaN	2400.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1650.0	990.0	2640.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1575.0	945.0	2520.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

I will be using Boxplots to detect and visualize the outliers present in the dataset.

In [23]:
```
# Plot a box plot to visualize the outliers

# 8 subplots in one row
fig, ax = plt.subplots(1, 8, figsize=(10, 10))

# draw boxplots - for one column in each subplot
sales_data_na.boxplot('Year', ax=ax[0])
sales_data_na.boxplot('Customer_Age', ax=ax[1])
sales_data_na.boxplot('Order_Quantity', ax=ax[2])
sales_data_na.boxplot('Unit_Cost', ax=ax[3])
sales_data_na.boxplot('Unit_Price', ax=ax[4])
sales_data_na.boxplot('Profit', ax=ax[5])
sales_data_na.boxplot('Cost', ax=ax[6])
sales_data_na.boxplot('Revenue', ax=ax[7])

plt.subplots_adjust(wspace=0.5)

plt.show()
```

Boxplot consists of Q1, Q2, Q3, lower limit and upper limit. Any data point that lies below the lower bound and above the upper bound is considered as an Outlier. Below boxplot for each column.



I will check for Order_Quantity variable only. First find the IQR then the upper limit and lower limit.

In [24]:
```
#IQR score for Order_Quantity
Q1 = sales_data_na['Order_Quantity'].quantile(0.25)
Q3 = sales_data_na['Order_Quantity'].quantile(0.75)

IQR = Q3 - Q1

upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)
lower_limit,upper_limit

# find the outliers in Order_Quantity column
sales_data_na.loc[(sales_data_na['Order_Quantity'] > upper_limit) |(sales_data_na['Order_Quantity'] < lower_limit)]
```

The output below is the outlier for Quantity variable, there is 1 row. As from the output, the cost value is incorrect, it should be 31200 instead of 312, I can confirm that this is an error and thus remove this from the dataset.

Out[24]:

ge	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category	Product	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
47	Adults (35-64)	Male	United States	California	Clothing	Shorts	Women's Mountain Shorts, S	1200	26	70	83688	312	84000

.

In [25]:
```
# trimming - delete the outliers data for Order_Quantity

new_sales_data_na = sales_data_na.loc[(sales_data_na['Order_Quantity'] < upper_limit) & (sales_data_na['Order_Quantity'] > lower_
print('before removing the Unit Price outliers:', len(sales_data_na))
print('after removing the Unit Price outliers:', len(new_sales_data_na))
print('outliers:', len(sales_data_na) - len(new_sales_data_na))
```

```
before removing the Unit Price outliers: 106807
after removing the Unit Price outliers: 106806
outliers: 1
```

I will be not removing the outliers for other column; it will affect the overall analysis and relation between variable later because there is a lot of outliers found in the dataset.

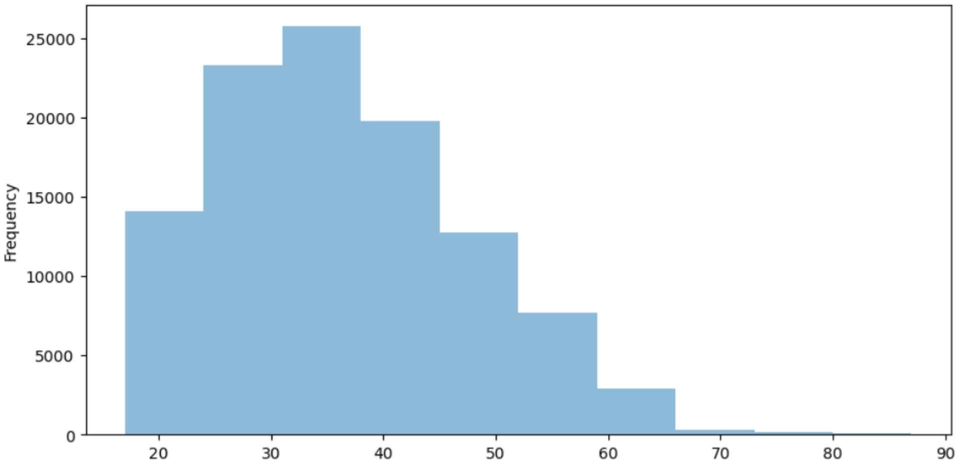
Part 4: Data Visualization

1. How old is the customer age?

In [26]:
```
# customer age histogram
new_sales_data_na['Customer_Age'].plot(kind='hist', figsize=(10,5),alpha = 0.5, xlabel ='Number of customer')
```

The company has more 20 to 40 years old customers

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x24f7d914fd0>



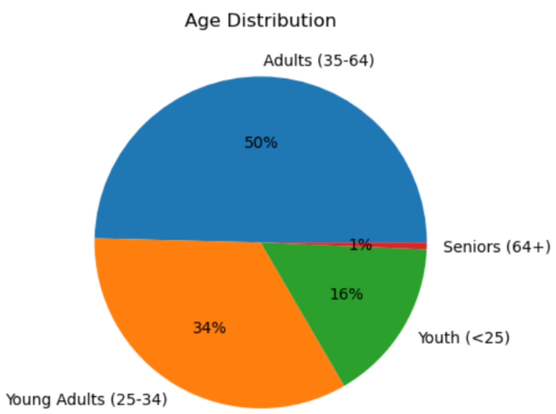
2. What is the age distribution of the customers?

In [27]:
```
# Select the Age_Group variable using counts() function
new_sales_data_na['Age_Group'].value_counts()
```

Out[27]:
```
Adults (35-64)          52982
Young Adults (25-34)    36077
Youth (<25)             17030
Seniors (64+)             717
Name: Age_Group, dtype: int64
```

It was determined that the highest number of customers were Adults between the ages of 35 to 64. I will plot pie chart to visualize this

In [28]:
```
import matplotlib.pyplot as plt
import numpy as np

new_sales_data_na['Age_Group'].value_counts().plot(kind = "pie",autopct='%.0f%%',title = 'Age Distribution', ylabel='')

plt.show()
```



3. How many sales per year?

```
In [29]: #sales per year
         new_sales_data_na["Year"].value_counts().sort_index()
```

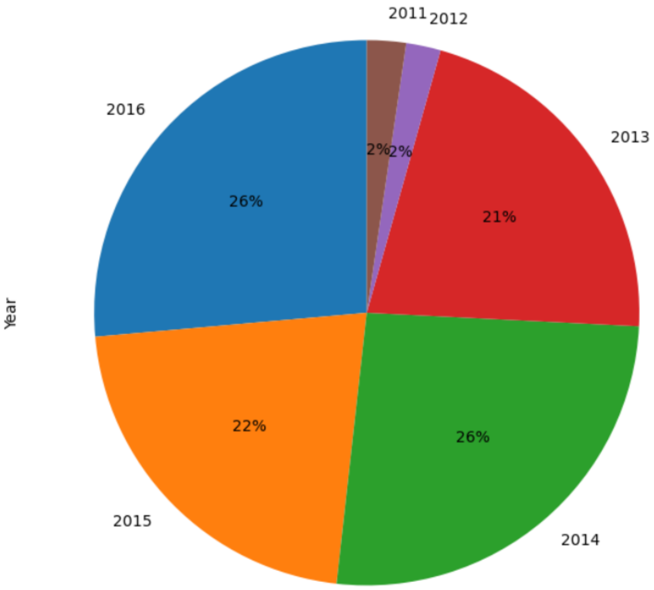
2016 has the highest sales followed by 2014

```
Out[29]: 2011     2473
         2012     2191
         2013    22883
         2014    27745
         2015    23314
         2016    28200
         Name: Year, dtype: int64
```

I will plot pie chart to visualize this

```
In [30]: #pie chart sales per year
         sales_year = new_sales_data_na['Year'].value_counts().sort_index(ascending = False)
         sales_year.plot(kind='pie', figsize = (8,8), startangle = 90, autopct='%.0f%%')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x24f03040f28>



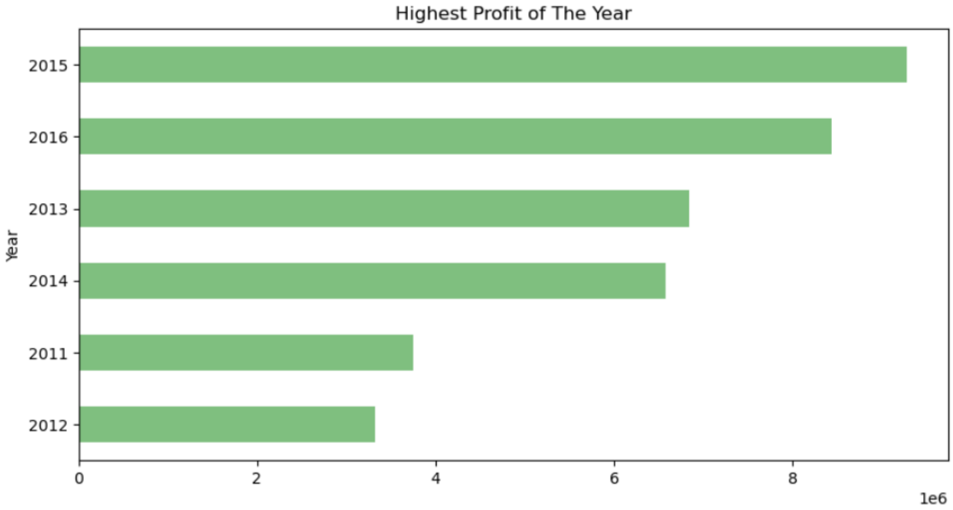
4. Which year was the most profitable?
   I grouped the data by year and summed up the profits. I sorted the results in from highest to lowest.

In [31]:
```
print(new_sales_data_na.groupby('Year').sum()['Profit'].sort_values(ascending = False))
```

```
Year
2015    9277542
2016    8442164
2013    6835476
2014    6577261
2011    3752951
2012    3321598
Name: Profit, dtype: int64
```

In [32]:
```
# Import matplotlib and pandas
import matplotlib.pyplot as plt
import pandas as pd

# Create horizontal bar chart
new_sales_data_na.groupby('Year').sum()['Profit'].sort_values(ascending = True).plot(kind = 'barh', color= 'green',alpha = 0.5, f

# Show the plot
plt.show()
```

The results show that the highest profitable year is the year 2015

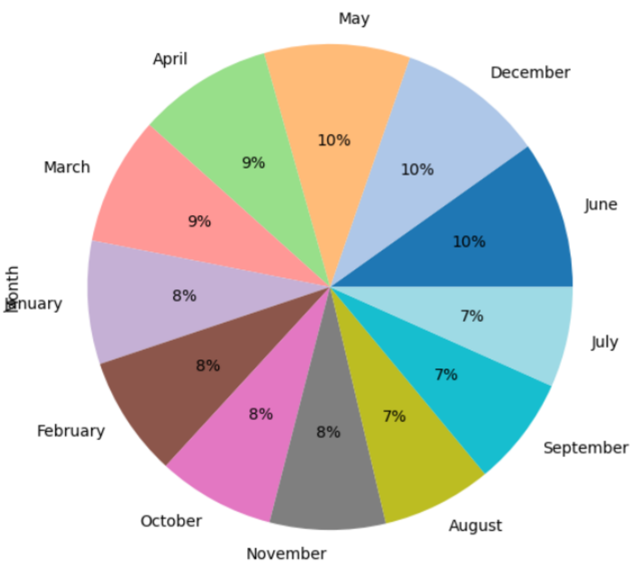


5. What are sales per month?

```
In [33]: #sales per month bar plot
new_sales_data_na["Month"].value_counts().sort_values(ascending = False).plot(kind='pie', figsize = (7,7),autopct='%.0f%%',cmap='
```

From the bar chart below, it shows that May, June and December have the highest sales

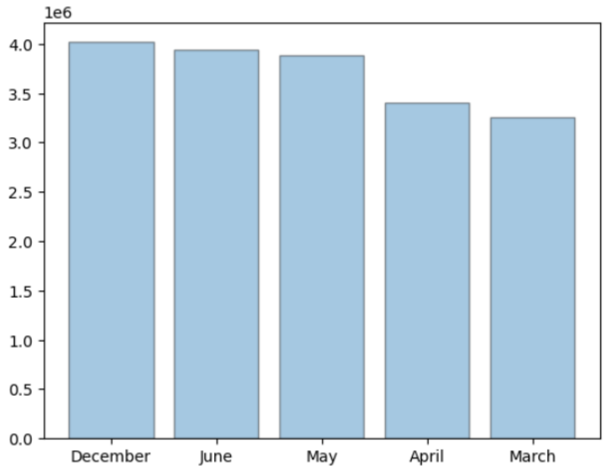
```
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x24f053260b8>
```



6. The most profitable month.

```
In [34]: # Import matplotlib and pandas
import matplotlib.pyplot as plt
import pandas as pd

# Create histogram
Profitable_month = new_sales_data_na.groupby('Month').sum()['Profit'].sort_values(ascending = False).head(5)
plt.bar(Profitable_month.index,Profitable_month, alpha = 0.4, edgecolor='black')

# Show the plot
plt.show()
```

The Chart shows that the highest profitable month is December, followed by June. A factor responsible for this is that during end of year people tend to buy things.

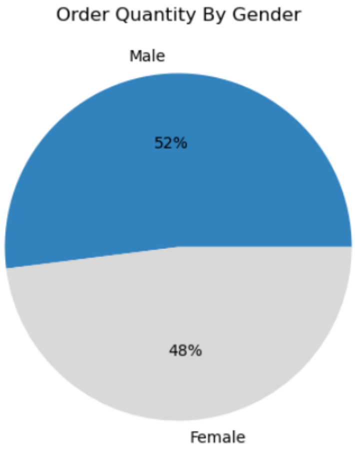


7. Which Gender has the most orders?

```
In [35]: new_sales_data_na.groupby('Customer_Gender').sum()['Order_Quantity'].sort_values(ascending = False)

Out[35]: Customer_Gender
Male      680136
Female    628183
Name: Order_Quantity, dtype: int64
```

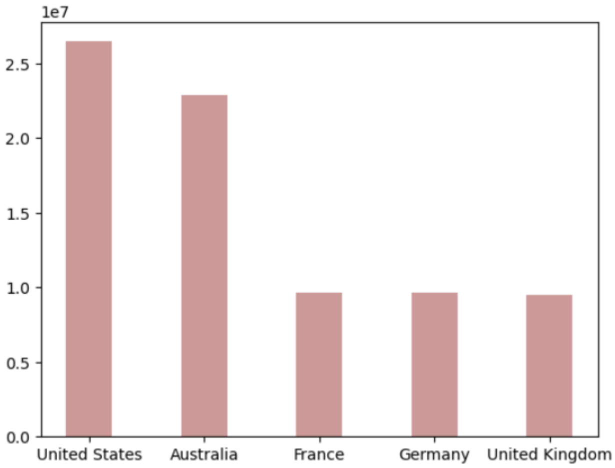
```
In [36]: # Import matplotlib and pandas
import matplotlib.pyplot as plt
import pandas as pd

# Create pie chart
new_sales_data_na.groupby('Customer_Gender').sum()['Order_Quantity'].sort_values(ascending = False).plot(kind = 'pie',autopct='%.
       figsize = (10,5), title = 'Order Quantity By Gender', ylabel='')

# show plot
plt.show()
```

Chart shows that about 52% of orders were made by Men



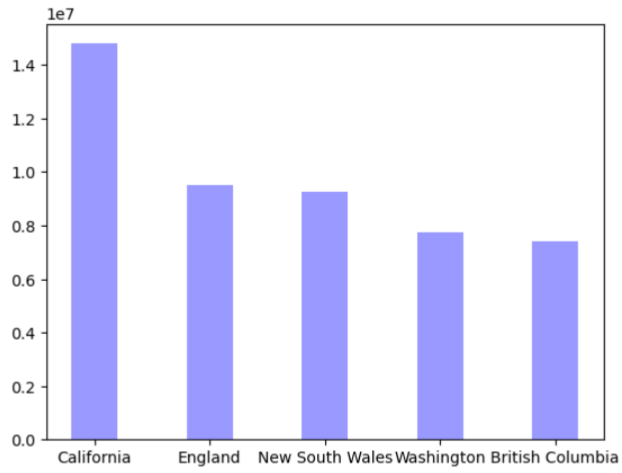
8. Which country/state generates the highest revenue?

```
In [37]: # Import matplotlib and pandas
import matplotlib.pyplot as plt
import pandas as pd

# Create histogram
HighestcountryRevenue = new_sales_data_na.groupby('Country').sum()['Revenue'].sort_values(ascending = False).head(5)
plt.bar(HighestcountryRevenue.index,HighestcountryRevenue, alpha = 0.4, color='maroon',width = 0.4)

# Show the plot
plt.show()
```



Bar Chart shows that the highest revenue generating country is the United States followed by Australia.

```
In [38]: # Import matplotlib and pandas
import matplotlib.pyplot as plt
import pandas as pd

# Create histogram
HigheststateRevenue = new_sales_data_na.groupby('State').sum()['Revenue'].sort_values(ascending = False).head(5)
plt.bar(HigheststateRevenue.index,HigheststateRevenue, alpha = 0.4, color='blue',width = 0.4)

# Show the plot
plt.show()
```



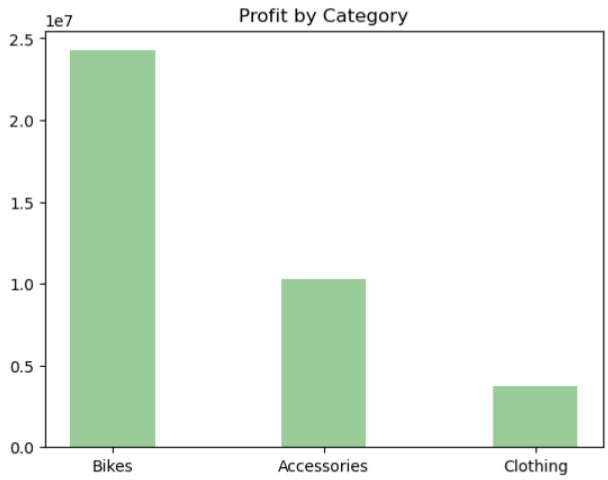
The highest revenue generating state is California.

9. Which category/subcategory generates the most profit?

```
In [39]: productCategory = new_sales_data_na.groupby('Product_Category').sum()['Profit'].sort_values(ascending = False)
         print(productCategory)
```

```
Product_Category
Bikes          24247113
Accessories    10252378
Clothing        3707501
Name: Profit, dtype: int64
```

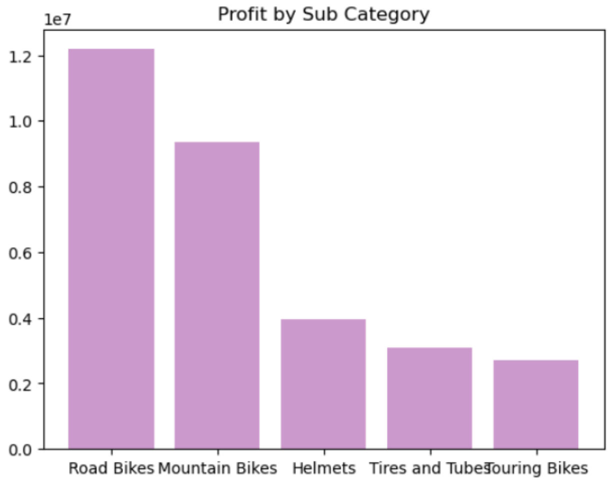
```
In [40]: # Import matplotlib and pandas
         import matplotlib.pyplot as plt
         import pandas as pd

         # Create histogram
         Highestcategory = new_sales_data_na.groupby('Product_Category').sum()['Profit'].sort_values(ascending = False).head(5)
         plt.bar(Highestcategory.index,Highestcategory, alpha = 0.4, color='green',width = 0.4)
         plt.title('Profit by Category')

         # Show the plot
         plt.show()
```



The results show that the most profitable category is bikes.

10. The most profitable subcategory.

```
In [41]: # Import matplotlib and pandas
import matplotlib.pyplot as plt
import pandas as pd

# Create histogram
Highestsubcategory = new_sales_data_na.groupby('Sub_Category').sum()['Profit'].sort_values(ascending = False).head(5)
plt.bar(Highestsubcategory.index,Highestsubcategory, alpha = 0.4, color='purple')
plt.title('Profit by Sub Category')

# Show the plot
plt.show()
```



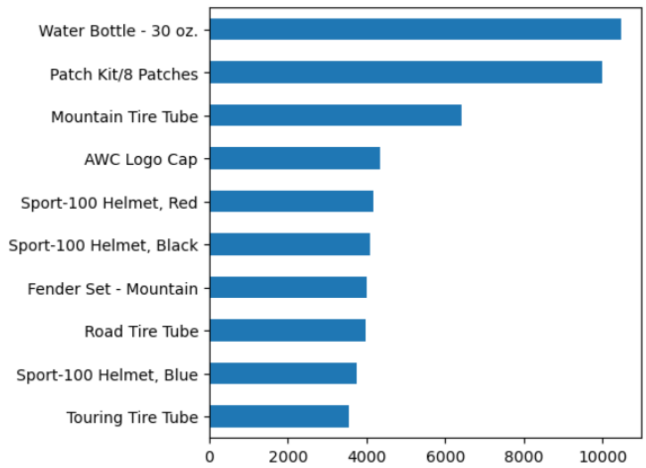
From the chart shows that the most profitable subcategory is Road bikes.

11. What is the top 10 best seller products

```
In [42]: new_sales_data_na["Product"].value_counts().head(10).sort_values(ascending = True).plot(kind="barh", figsize = (5,5))
```

The bar chart below shows the top 10 best seller products.

```
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x24f09528da0>
```

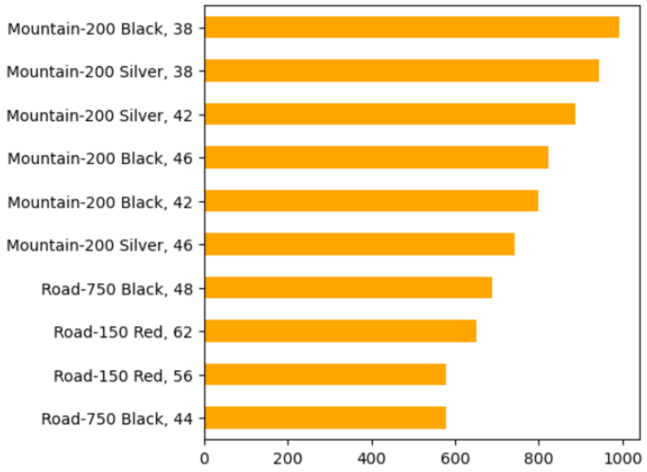


12. Best seller bikes

```
In [43]: top_bikes = new_sales_data_na.loc[new_sales_data_na["Product_Category"] == "Bikes", "Product"].value_counts().head(10).sort_value
         top_bikes
         top_bikes.plot(kind="barh", figsize=(5,5), color='orange')
```

Below show the best seller bikes sold

```
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x24f095ba128>
```



13. Correlation Table to find out the relationship

```
In [44]: cor = new_sales_data_na.corr().style.background_gradient(cmap='coolwarm')
         cor
```

The correlation heat map shows that there is no relationship between the Customer’s age and revenue. Instead, the relationship exists between the unit cost, price, profit and revenue.

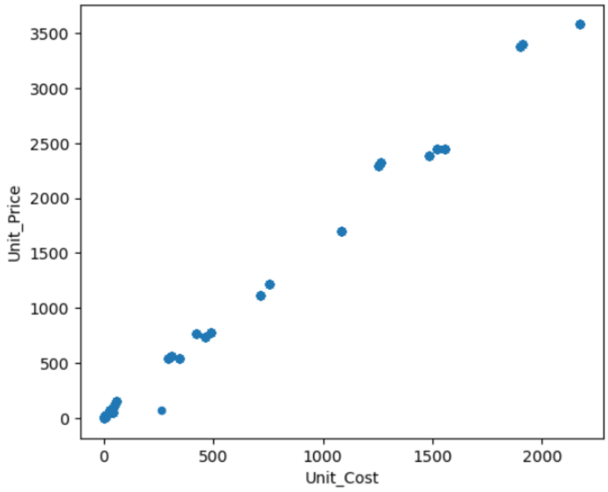
Out[44]:

	Year	Customer_Age	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
Year	1.000000	0.037111	0.111266	-0.207043	-0.203404	-0.184933	-0.202788	-0.197513
Customer_Age	0.037111	1.000000	0.023493	-0.015986	-0.014978	-0.003165	-0.011832	-0.008423
Order_Quantity	0.111266	0.023493	1.000000	-0.499868	-0.499928	-0.249350	-0.322639	-0.295992
Unit_Cost	-0.207043	-0.015986	-0.499868	1.000000	0.997916	0.783774	0.826813	0.817408
Unit_Price	-0.203404	-0.014978	-0.499928	0.997916	1.000000	0.790965	0.823451	0.818305
Profit	-0.184933	-0.003165	-0.249350	0.783774	0.790965	1.000000	0.959865	0.985592
Cost	-0.202788	-0.011832	-0.322639	0.826813	0.823451	0.959865	1.000000	0.993473
Revenue	-0.197513	-0.008423	-0.295992	0.817408	0.818305	0.985592	0.993473	1.000000

14. Relationship between Unit Cost and Unit Price
I will be using scatterplot to visualize this relationship

```
In [45]: # Relationship between Unit Cost and Unit Price
         new_sales_data_na.plot(x = "Unit_Cost", y = "Unit_Price", figsize = (6,5), kind = "scatter")
```

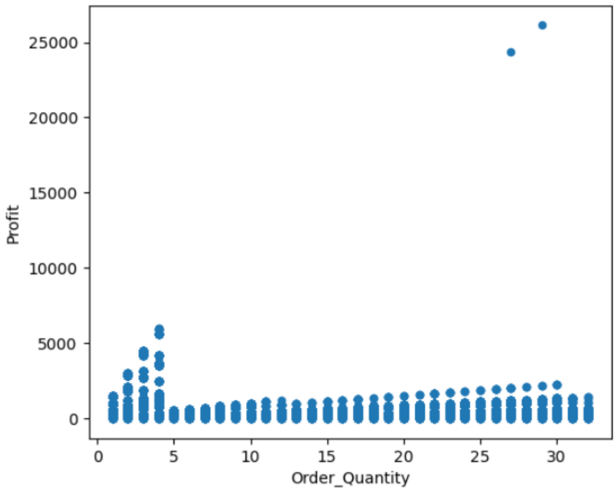
```
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x24f097d3438>
```



15. Relationship between order quantity and profit

```
In [46]: # Relationship between order quantity and profit
         new_sales_data_na.plot(x = "Order_Quantity", y = "Profit", kind = "scatter", figsize = (6,5))
```

```
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x24f09510240>
```

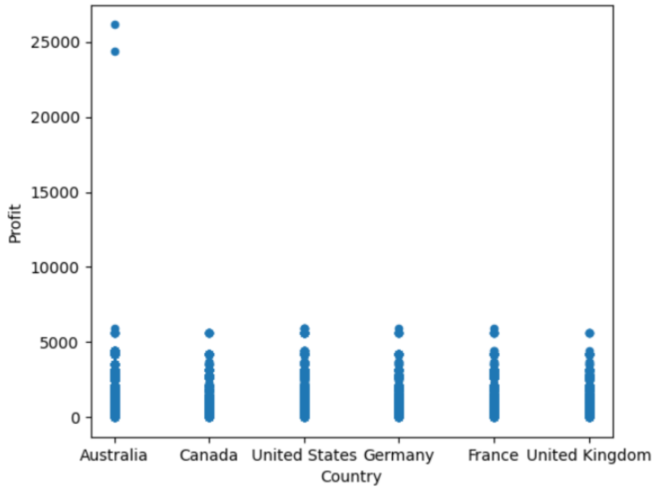


There is a positive linear relation between the points.

16. Relationship between Profit and Country

In [47]:
```
# Relationship between Profit and Country
new_sales_data_na.plot(x = "Country", y = "Profit", kind = "scatter", figsize = (6,5))
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x24f094a1588>

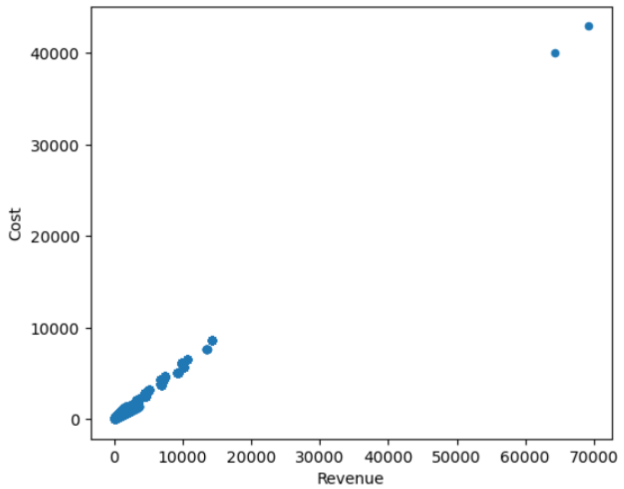


There is a positive linear relation between the points

17. Relationship between Revenue and Cost

In [48]:
```
# Relationship between Revenue and Cost
new_sales_data_na.plot(x = "Revenue", y = "Cost", kind = "scatter", figsize = (6,5))
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x24f09431940>

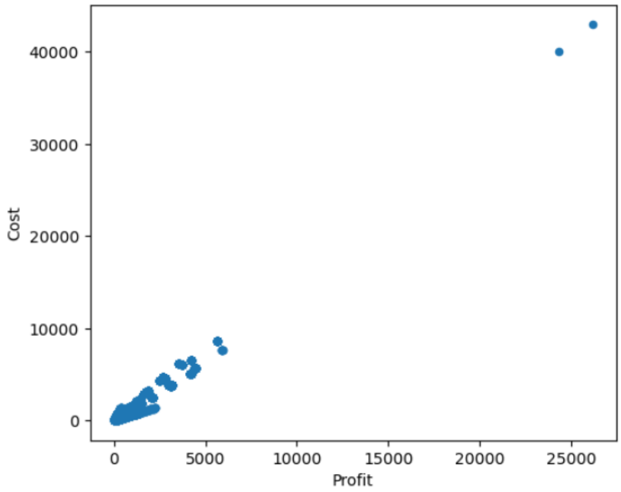


There is a positive linear relation between the points

18. Relationship between Profit and Cost

```
In [49]: # Relationship between Profit and Cost
new_sales_data_na.plot(x = "Profit", y = "Cost", kind = "scatter", figsize = (6,5))
```

```
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x24f06cd6400>
```

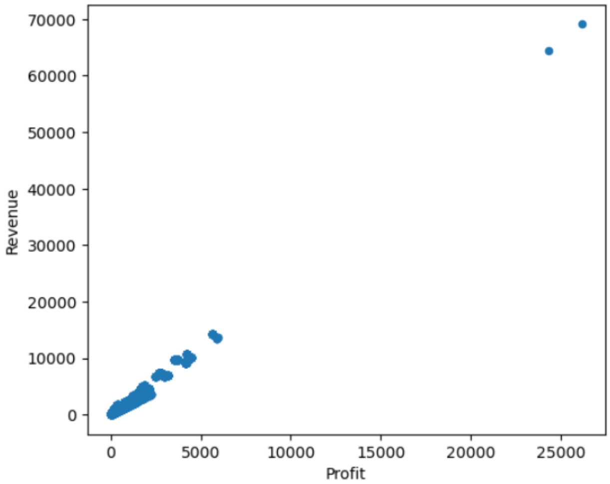


There is a positive linear relation between the points

19. Relationship between Profit and Revenue

```
In [50]: # Relationship between Profit and Revenue
new_sales_data_na.plot(x = "Profit", y = "Revenue", kind = "scatter", figsize = (6,5))
```

```
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x24f05336eb8>
```



There is a positive linear relation between the points