Exploatory Data Analysis (EDA) for Shoes Sales

Importing Libraries(Modules)

[2]:
```
import pandas as pd
```

[3]:
```
from matplotlib import pyplot
```

Reading CSV File(shoes_sales.csv)

[4]:
```
df = pd.read_csv("shoe_sales.csv")
df.head()
```

[4]:

	date	brand	sold_qty
0	9/1/2023	Nike	24.0
1	9/1/2023	Adidas	14.0
2	9/2/2023	Nike	21.0
3	9/2/2023	Adidas	12.0
4	9/3/2023	Nike	18.0

Finded Total No. of Rows and Columns

[5]:
```
df.shape
```

[5]: (60, 3)

Quick Statistical Summary

[6]:
```
df.describe()
```

[6]:

	sold_qty
count	58.000000
mean	27.482759
std	88.519844
min	7.000000
25%	12.250000
50%	16.000000
75%	19.750000
max	689.000000

A Rows whose sold_qty is below 12.25 (less then 25% percentile)

[7]:
```
df[df.sold_qty < 12.25]
```

[7]:

	date	brand	sold_qty
3	9/2/2023	Adidas	12.0
5	9/3/2023	Adidas	11.0
9	9/5/2023	Adidas	10.0
15	9/8/2023	Adidas	8.0
19	9/10/2023	Adidas	7.0
21	9/11/2023	Adidas	9.0
25	9/13/2023	Adidas	11.0
29	9/15/2023	Adidas	10.0
33	9/17/2023	Adidas	8.0
37	9/19/2023	Adidas	7.0
43	9/22/2023	Adidas	12.0
45	9/23/2023	Adidas	11.0
49	9/25/2023	Adidas	10.0
51	9/26/2023	Adidas	9.0
57	9/29/2023	Adidas	8.0

A Rows whose sold_qty is above 19.75 (greater then 75% percentile)

[8]:
```
df[df.sold_qty > 19.75]
```

[8]:

	date	brand	sold_qty
0	9/1/2023	Nike	24.0
2	9/2/2023	Nike	21.0
6	9/4/2023	Nike	22.0
8	9/5/2023	Nike	20.0
10	9/6/2023	Nike	23.0
16	9/9/2023	Nike	25.0
20	9/11/2023	Nike	23.0
23	9/12/2023	Adidas	689.0
26	9/14/2023	Nike	22.0
30	9/16/2023	Nike	21.0
38	9/20/2023	Nike	24.0
40	9/21/2023	Nike	24.0
44	9/23/2023	Nike	20.0
52	9/27/2023	Nike	22.0
54	9/28/2023	Nike	21.0

1. Analysis For Nike Shoes

[9]:
```
df_nike = df[df.brand == "Nike"]
df_nike.head()
```

[9]:

	date	brand	sold_qty
0	9/1/2023	Nike	24.0
2	9/2/2023	Nike	21.0

4	9/3/2023	Nike	18.0
6	9/4/2023	Nike	22.0
8	9/5/2023	Nike	20.0

Total No. of Rows and Column whose brand is Nike

[10]:
```
df_nike.shape
```

[10]:
```
(30, 3)
```

Quick Statistical Analysis for df_nike

[11]:
```
df_nike.describe()
```

[11]:

	sold_qty
count	28.000000
mean	19.642857
std	3.117624
min	14.000000
25%	17.000000
50%	19.500000
75%	22.000000
max	25.000000

Median of total Sales qty of Nike Shoes

[12]:
```
df_nike_median = round(df_nike.sold_qty.median())
df_nike_median
```

[12]:
```
20
```

Finding Rows who contain Null atleast in One Row

[13]:
```
df_nike.isnull()
```

[13]:

	date	brand	sold_qty
0	False	False	False
2	False	False	False
4	False	False	False
6	False	False	False
8	False	False	False
10	False	False	False
12	False	False	False
14	False	False	False
16	False	False	False
18	False	False	False
20	False	False	False
22	False	False	False
24	False	False	False
26	False	False	False
28	False	False	False
30	False	False	False
32	False	False	True
34	False	False	False
36	False	False	False
38	False	False	False
40	False	False	False
42	False	False	False
44	False	False	False
46	False	False	False
48	False	False	True
50	False	False	False
52	False	False	False
54	False	False	False
56	False	False	False
58	False	False	False

Finding Null Value of Sold_qty

[14]:
```
df_nike[df_nike.sold_qty.isnull()]
```

[14]:

	date	brand	sold_qty
32	9/17/2023	Nike	NaN
48	9/25/2023	Nike	NaN

Replacing Null with Median Value of sold_qty

[20]:
```
df_nike.sold_qty.fillna(df_nike_median, inplace = True)
df_nike.sold_qty
```

```
C:\Users\bharg\AppData\Local\Temp\ipykernel_9948\1033834991.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
ined assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves a
s a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instea
d, to perform the operation inplace on the original object.


  df_nike.sold_qty.fillna(df_nike_median, inplace = True)
C:\Users\bharg\AppData\Local\Temp\ipykernel_9948\1033834991.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_nike.sold_qty.fillna(df_nike_median, inplace = True)
```

[20]:
```
0     24.0
2     21.0
4     18.0
6     22.0
8     20.0
10    23.0
12    19.0
14    17.0
```

```
16    25.0
18    14.0
20    23.0
22    19.0
24    16.0
26    22.0
28    17.0
30    21.0
32    20.0
34    18.0
36    15.0
38    24.0
40    24.0
42    16.0
44    20.0
46    15.0
48    20.0
50    19.0
52    22.0
54    21.0
56    17.0
58    18.0
Name: sold_qty, dtype: float64
```

[26]:
```
df_nike.loc[[32,48]]
```

[26]:

	date	brand	sold_qty
32	9/17/2023	Nike	20.0
48	9/25/2023	Nike	20.0

Total Sale of Nike Shoes

[28]:
```
print(f"Total Sales of Nike Shoes is {df_nike.sold_qty.sum()}")
```

```
Total Sales of Nike Shoes is 590.0
```

2. Analysis of Adidas Shoes

[17]:
```
df_adidas = df[df.brand == "Adidas"]
df_adidas
```

[17]:

	date	brand	sold_qty
1	9/1/2023	Adidas	14.0
3	9/2/2023	Adidas	12.0
5	9/3/2023	Adidas	11.0
7	9/4/2023	Adidas	13.0
9	9/5/2023	Adidas	10.0
11	9/6/2023	Adidas	15.0
13	9/7/2023	Adidas	16.0
15	9/8/2023	Adidas	8.0
17	9/9/2023	Adidas	17.0
19	9/10/2023	Adidas	7.0
21	9/11/2023	Adidas	9.0
23	9/12/2023	Adidas	689.0
25	9/13/2023	Adidas	11.0
27	9/14/2023	Adidas	13.0
29	9/15/2023	Adidas	10.0
31	9/16/2023	Adidas	14.0
33	9/17/2023	Adidas	8.0
35	9/18/2023	Adidas	15.0
37	9/19/2023	Adidas	7.0
39	9/20/2023	Adidas	19.0
41	9/21/2023	Adidas	18.0
43	9/22/2023	Adidas	12.0
45	9/23/2023	Adidas	11.0
47	9/24/2023	Adidas	14.0
49	9/25/2023	Adidas	10.0
51	9/26/2023	Adidas	9.0
53	9/27/2023	Adidas	13.0
55	9/28/2023	Adidas	15.0
57	9/29/2023	Adidas	8.0
59	9/30/2023	Adidas	16.0

Quick Statistical Analysis of Adidas Shoes

[18]:
```
df_adidas.describe()
```

[18]:

	sold_qty
count	30.000000
mean	34.800000
std	123.602366
min	7.000000
25%	10.000000
50%	12.500000
75%	15.000000
max	689.000000

Medain of Total sold_qty of Adidas Shoes

[25]:
```
df_adidas_median = df_adidas.sold_qty.median()
df_adidas_median
```

[25]:
```
np.float64(12.5)
```

Finding Outlier

Finding 90% is less than ?

[36]:
```
df_adidas.sold_qty.quantile([0.90])
```

[36]:
```
0.9    17.1
Name: sold_qty, dtype: float64
```

[38]:
```
df_adidas[df_adidas.sold_qty > 18]
```

[38]:

	date	brand	sold_qty
23	9/12/2023	Adidas	689.0

```
39  9/20/2023  Adidas  19.0
```

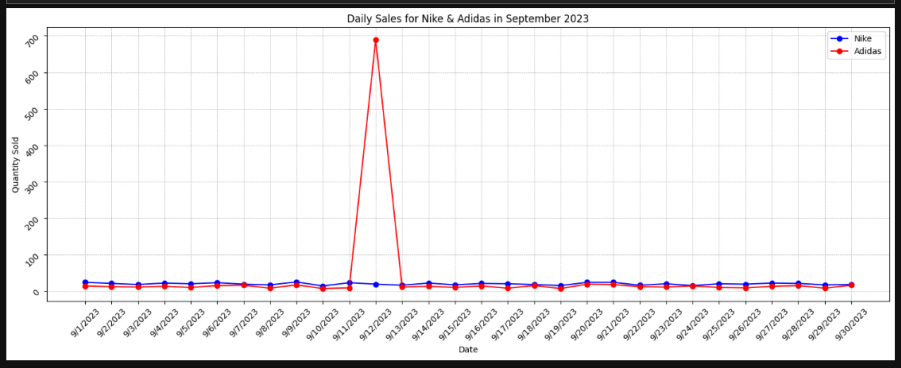
[45]:
```
from matplotlib import pyplot as plt
def plot_qty():
    plt.figure(figsize=(15,6))

    dates = df_nike["date"]

    plt.plot(dates, df_nike['sold_qty'], marker='o', label='Nike', color='blue')
    plt.plot(dates, df_adidas['sold_qty'], marker='o', label='Adidas', color='red')

    plt.xlabel('Date')
    plt.ylabel('Quantity Sold')
    plt.title('Daily Sales for Nike & Adidas in September 2023')
    plt.xticks(rotation=45)
    plt.yticks(rotation=45)
    plt.legend()
    plt.tight_layout()
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.show()
```

[46]:
```
plot_qty()
```



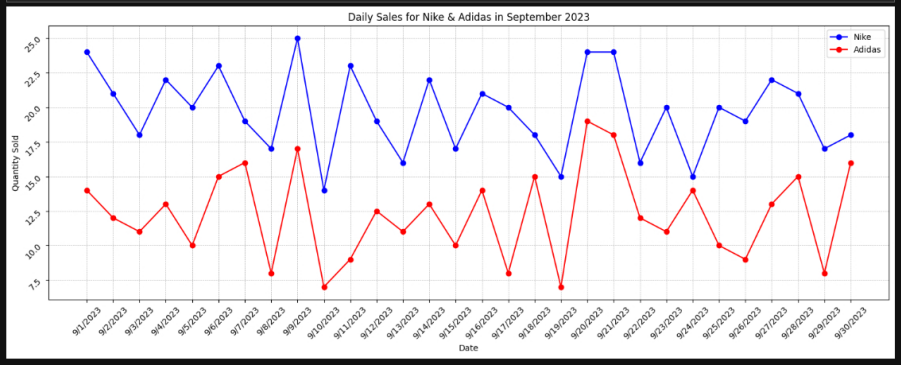
Replacing Outlier with Median

[50]:
```
df_adidas.sold_qty.replace(689, df_adidas_median, inplace=True)
```

```
C:\Users\bharg\AppData\Local\Temp\ipykernel_9948\3201625330.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_adidas.sold_qty.replace(689, df_adidas_median, inplace=True)
```

[51]:
```
plot_qty()
```



[ ]: