Python for Data Science LAB Session - 13

Basics of Data Visualization

Q-1) Read the data from company_sales_data.csv file. Concider all the "?" and "n.a." values as NaN values.

a. Read all the profit data of each month and display it using the scatter plot

b. Create a line on the scatterplot that displays the general direction of the profit using polyfit()

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
```

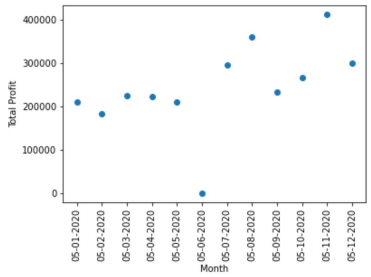
In [2]:
```
df = pd.read_csv("company_sales_data.csv", index_col=0, na_values=['?', 'n.a.'])
df = df.fillna(0)   #load the data and it fill as 0 ,where value ? or n.a.
```

In [3]:
```
df
```

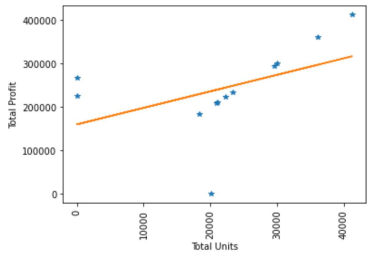
Out[3]:

month_number	Date	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	total_profit
1	05-01-2020	2500	1500	5200	9200.0	1200.0	1500	21100.0	211000.0
2	05-02-2020	2630	1200	5100	6100.0	2100.0	1200	18330.0	183300.0
3	05-03-2020	2140	1340	4550	9550.0	0.0	1340	0.0	224700.0
4	05-04-2020	3400	1130	5870	8870.0	1870.0	1130	22270.0	222700.0
5	05-05-2020	3600	1740	4560	0.0	1560.0	1740	20960.0	209600.0
6	05-06-2020	2760	1555	4890	7490.0	1890.0	1555	20140.0	0.0
7	05-07-2020	2980	1120	4780	8980.0	1780.0	1120	29550.0	295500.0
8	05-08-2020	3700	1400	5860	9960.0	2860.0	1400	36140.0	361400.0
9	05-09-2020	3540	1780	6100	8100.0	2100.0	1780	23400.0	234000.0
10	05-10-2020	1990	1890	8300	0.0	2300.0	1890	0.0	266700.0
11	05-11-2020	2340	2100	7300	13300.0	2400.0	2100	41280.0	412800.0
12	05-12-2020	2900	1760	7400	14400.0	0.0	1760	30020.0	300200.0

In [4]:
```
plt.scatter(df.Date, df.total_profit)      #Scatter plot
plt.xticks(rotation=90)  #It rotate x value to 90
plt.xlabel("Month")
plt.ylabel("Total Profit")
plt.show()
```



In [5]:
```
fig, con = np.polyfit(df.total_units, df.total_profit, 1)
plt.plot(df.total_units, df.total_profit, '*')
plt.plot(df.total_units, fig * df.total_units + con)  #that make's the y= m*x + c line equation
plt.xticks(rotation=90)
plt.xlabel("Total Units")
plt.ylabel("Total Profit")
plt.show()
```



Q-2)Create a dataframe taking facecream and toothpaste sales data from the original dataframe with date as index column.

a. Create a time series graph that shows facecream and toothpaste sales data on each data. Use different style and markers for facecream and toothpaste data

b. Create two columns facecreame-trend and toothpaste-trend that contains the value of general trends over time and display both trends as lines on top of the timesries graph.

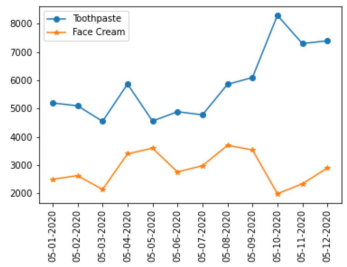
In [6]:
```
df1 = pd.DataFrame({"face_cream": df.facecream.to_list(), "toothpaste": df.toothpaste.to_list()}, index=df.Date)
# here we arrange face cream and toothpaste date to dataframe by date wise
```

In [7]:
```
df1
```

Out[7]:

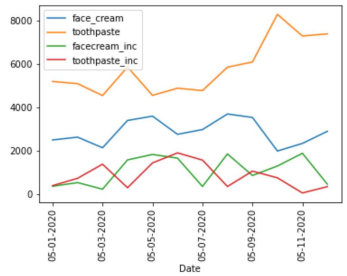
Date	face_cream	toothpaste
05-01-2020	2500	5200
05-02-2020	2630	5100
05-03-2020	2140	4550
05-04-2020	3400	5870
05-05-2020	3600	4560
05-06-2020	2760	4890
05-07-2020	2980	4780
05-08-2020	3700	5860
05-09-2020	3540	6100
05-10-2020	1990	8300
05-11-2020	2340	7300
05-12-2020	2900	7400

In [8]:
```
plt.plot(df1.index, df1.toothpaste, label='Toothpaste',marker='o')  #plor the data
plt.plot(df1.index, df1.face_cream, label='Face Cream',marker='*')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



In [9]:
```
facecream_inc = np.array(np.random.rand(12) * 2000, dtype=np.int32) # these both varible genrate randome value between 2000
toothpaste_inc = np.array(np.random.rand(12) * 2000, dtype=np.int32)

trend = df1
trend['facecream_inc'] = facecream_inc
trend['toothpaste_inc'] = toothpaste_inc

trend.plot()
plt.xticks(rotation=90)
```

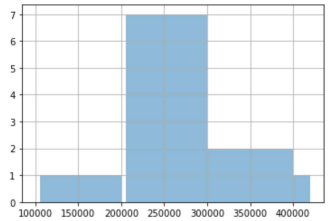
Out[9]:
```
(array([-2.,  0.,  2.,  4.,  6.,  8., 10., 12.]),
 [Text(-2.0, 0, '05-11-2020'),
  Text(0.0, 0, '05-01-2020'),
  Text(2.0, 0, '05-03-2020'),
  Text(4.0, 0, '05-05-2020'),
  Text(6.0, 0, '05-07-2020'),
  Text(8.0, 0, '05-09-2020'),
  Text(10.0, 0, '05-11-2020'),
  Text(12.0, 0, '')])
```



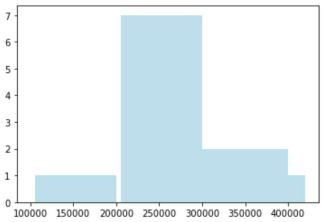
Q-3)Read the total profit of each month and show it using the histogram to see most common profit ranges.

Explore different parameters of hist() and its uses.

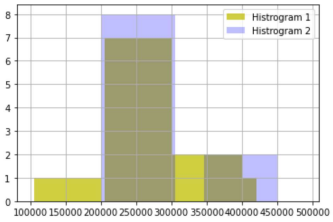
In [10]:
```
total = df.total_profit.to_list()
rang = [100000, 105000, 200000, 205000, 300000, 400000, 420000]

plt.hist(total, rang,alpha=0.5)
plt.grid() #grid
plt.show()
```



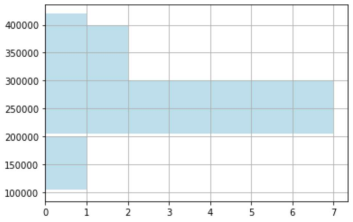
In [11]:
```
rang0 = [100000, 105000, 200000, 205000, 300000,350000, 400000, 420000]
plt.hist(total, rang, color='lightblue', alpha=0.8) #color used for give color and alpha for opacity
plt.show()
```



In [12]:
```
rang1 = [200000,305000, 346000, 450000, 490000]
plt.hist(total, rang, facecolor='y', alpha=0.75, label="Histrogram 1")
plt.hist(total, rang1, facecolor='b', alpha=0.25, label="Histrogram 2")
plt.grid(True)
plt.legend()
plt.show()
```

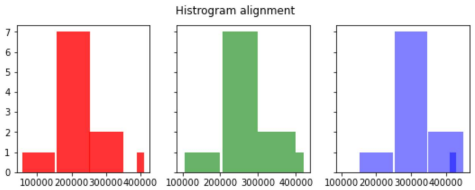


In [13]:
```
plt.hist(total, rang, color='lightblue', alpha=0.8, orientation='horizontal') #oreientation based horizantal/vertical
plt.grid()
plt.show()
```



In [14]:
```
fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)  #shray for all the y axis are same value
axs[0].hist(total, rang, facecolor='r', alpha=.8, align='left')   #alignment from left right center side
axs[1].hist(total, rang, facecolor='g', alpha=.6, align='mid')
axs[2].hist(total, rang, facecolor='b', alpha=.5, align='right')
fig.suptitle('Histrogram alignment')
```

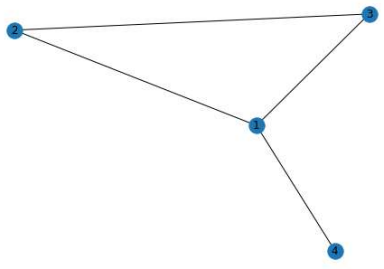
Out[14]: `Text(0.5, 0.98, 'Histrogram alignment')`



Q-4) Create an undirected graph from below adjacency matrix using networkx.

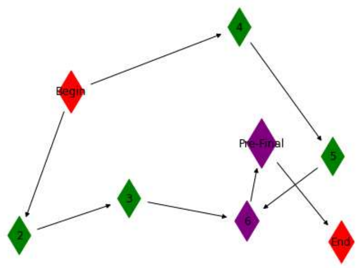
In [15]:
```
Graph = nx.Graph([(1,2), (1,3), (1,4), (2,1), (2,3), (3,1), (3,2), (4,1)])
```

In [16]:
```
nx.draw(Graph, with_labels = True)
```



Q-5) Create below Directed graph with appropriate labels and colors using networkx

In [17]:
```
grp = nx.DiGraph()
colors = ['green', 'green', 'green', 'green', 'purple', 'red', 'purple', 'red']

grp.add_nodes_from([2, 3, 4, 5, 6, "Begin", "Pre-Final", "End"]) # added all value
grp.add_edges_from([(4,5), ("Pre-Final", "End"), ("Begin", 2), ("Begin", 4),(6, "Pre-Final"), (5, 6), (3, 6), (2, 3)]) #show all path

nx.draw(grp, with_labels=True, node_color=colors,node_size=[900, 900, 900, 900, 1000, 1100, 1500, 1100],node_shape='d')#draw graph
```



Q-6)Perform below operations to create a box plot

a. Create a numpy array of size 100 that contains random numbers in the range of 50 to 200

b. Create a numpy array of size 75 that contains all 1s

c. Create a numpy array of size 12 that contains random numbers in the range of 200 to 240

d. Create a numpy array of size 10 that contains random numbers in the range of -10 to -70

e. Concatenate all these arrays and create a single array

f. Draw a default boxplot and display all outliers as red color x mark

g. Create a notched boxplot using same data and observe the difference.

In [18]:
```
arr = np.random.uniform(50, 200, 100)
```

In [19]:
```
arr
```

Out[19]:
```
array([113.15056793, 140.77590247,  60.5034521 , 197.22488   ,
        51.15614227, 170.29391859, 152.91087957, 187.39701841,
        55.86099951, 164.79006063,  69.22318367, 181.03859641,
       199.14759506, 137.23805344, 172.62993564, 173.48442914,
       180.96435542,  53.60544163,  75.13988941, 184.06386345,
       135.47333829, 129.11072049, 114.36612423, 168.75342476,
       190.9918443 , 110.34148697, 123.69300417, 132.35997835,
       164.55178846, 157.01017192,  71.17567739, 153.91356572,
       165.62090968, 102.69938877,  68.1002667 ,  96.70351594,
       142.57762451,  97.86636698, 135.09085239,  97.26792568,
       185.72053692, 117.98275519,  58.63892761, 187.02238867,
       167.89192305, 133.10488793,  72.50736343, 109.53255117,
       148.61216382,  93.06422446,  77.73135218, 181.85093558,
       128.05109076,  58.86842493, 166.11825082, 169.20979258,
       113.30632012, 143.60021659, 161.60294145, 156.70385674,
        50.58784487, 141.16265257, 170.98598691, 112.98759453,
       196.20683907, 184.16194723, 120.28577441, 171.26471206,
       178.28017799, 161.89949743,  91.94606712,  68.37333884,
       156.11848099, 149.47242942,  90.33746749, 129.8902996 ,
       148.95875111, 164.5224345 , 182.5713298 , 115.06778499,
       192.53565123,  57.6974238 ,  60.30473873, 195.7891697 ,
        70.59475063, 157.13744429, 163.16189897, 177.57884183,
       124.93011239, 179.7283863 , 118.70474584, 140.34948659,
       102.20296611, 180.91179053,  84.17927362, 108.3259271 ,
       113.68585379,  71.90073507, 138.66019836, 157.93815018])
```

In [20]:
```
arr1 = np.ones(75) * 100
arr1
```

Out[20]:
```
array([100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100., 100., 100.,
       100., 100., 100., 100., 100., 100., 100., 100., 100.])
```

In [21]:
```
arr2 = np.random.uniform(200, 240, 12)
arr2
```

Out[21]:
```
array([207.23703817, 235.26988763, 217.45213723, 230.79327922,
       210.50274499, 213.98164139, 216.14676086, 205.41623759,
       219.33600333, 222.06102462, 227.65764844, 212.00982013])
```

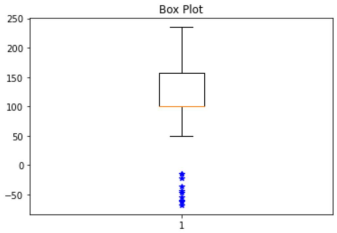
In [22]:
```
arr3 = np.random.uniform(-70, -10, 10)
arr3
```

Out[22]:
```
array([-46.75494237, -13.89462009, -62.3127355 , -61.1265556 ,
       -44.42635847, -53.91531869, -21.99466615, -36.08196174,
       -68.27524554, -14.59624308])
```

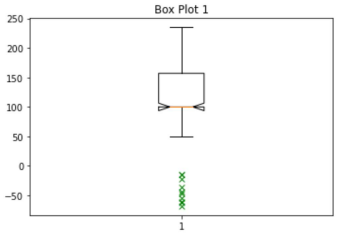
In [23]:
```
concate = np.concatenate((arr,arr1,arr2,arr3))
concate
```

Out[23]:
```
array([113.15056793, 140.77590247,  60.5034521 , 197.22488   ,
        51.15614227, 170.29391859, 152.91087957, 187.39701841,
        55.86099951, 164.79006063,  69.22318367, 181.03859641,
       199.14759506, 137.23805344, 172.62993564, 173.48442914,
       180.96435542,  53.60544163,  75.13988941, 184.06386345,
       135.47333829, 129.11072049, 114.36612423, 168.75342476,
       190.9918443 , 110.34148697, 123.69300417, 132.35997835,
       164.55178846, 157.01017192,  71.17567739, 153.91356572,
       165.62090968, 102.69938877,  68.1002667 ,  96.70351594,
       142.57762451,  97.86636698, 135.09085239,  97.26792568,
       185.72053692, 117.98275519,  58.63892761, 187.02238867,
       167.89192305, 133.10488793,  72.50736343, 109.53255117,
       148.61216382,  93.06422446,  77.73135218, 181.85093558,
       128.05109076,  58.86842493, 166.11825082, 169.20979258,
       113.30632012, 143.60021659, 161.60294145, 156.70385674,
        50.58784487, 141.16265257, 170.98598691, 112.98759453,
       196.20683907, 184.16194723, 120.28577441, 171.26471206,
       178.28017799, 161.89949743,  91.94606712,  68.37333884,
       156.11848099, 149.47242942,  90.33746749, 129.8902996 ,
       148.95875111, 164.5224345 , 182.5713298 , 115.06778499,
       192.53565123,  57.6974238 ,  60.30473873, 195.7891697 ,
        70.59475063, 157.13744429, 163.16189897, 177.57884183,
       124.93011239, 179.7283863 , 118.70474584, 140.34948659,
       102.20296611, 180.91179053,  84.17927362, 108.3259271 ,
       113.68585379,  71.90073507, 138.66019836, 157.93815018,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 100.        ,
       100.        , 100.        , 100.        , 207.23703817,
       235.26988763, 217.45213723, 230.79327922, 210.50274499,
       213.98164139, 216.14676086, 205.41623759, 219.33600333,
       222.06102462, 227.65764844, 212.00982013, -46.75494237,
       -13.89462009, -62.3127355 , -61.1265556 , -44.42635847,
       -53.91531869, -21.99466615, -36.08196174, -68.27524554,
       -14.59624308])
```

In [24]:
```
plt.boxplot(concate, 0, 'b*')
plt.title('Box Plot')
plt.show()
```



In [25]:
```
plt.boxplot(concate, 1, 'gx')
plt.title('Box Plot 1')
plt.show()
```



In [ ]: