In [8]:

```
data = [['E001', 'M', 34, 123, 'Normal', 350],
        ['E002', 'F', 40, 114, 'Overweight', 450],
        ['E003', 'F', 37, 135, 'Obesity', 169],
        ['E004', 'M', 30, 139, 'Underweight', 189],
        ['E005', 'F', 44, 117, 'Underweight', 183],
        ['E006', 'M', 36, 121, 'Normal', 80],
        ['E007', 'M', 32, 133, 'Obesity', 166],
        ['E008', 'F', 26, 140, 'Normal', 120],
        ['E009', 'M', 32, 133, 'Normal', 75],
        ['E010', 'M', 36, 133, 'Underweight', 40]]
```

In [9]:

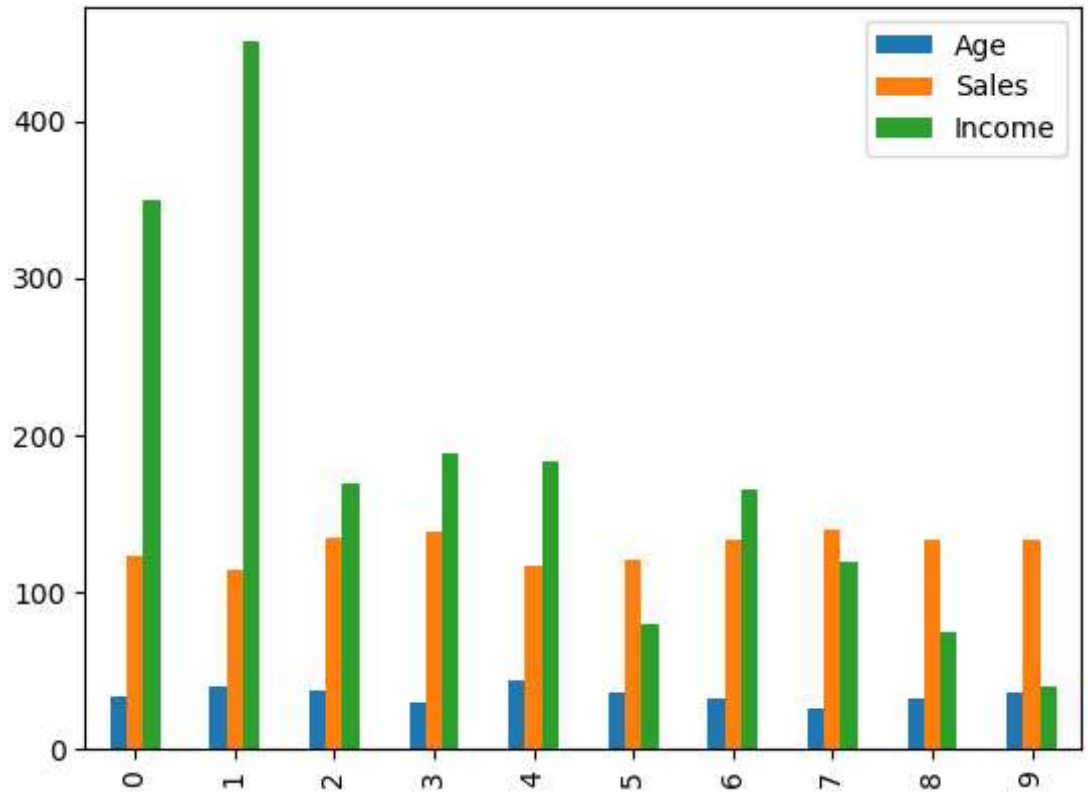
```
# HISTOGRAM

import pandas as pd
import matplotlib.pyplot as plt


df=pd.DataFrame(data,columns=['EMPID', 'Gender','Age','Sales','BMI','Income'])

df.hist()
plt.show()
```

In [10]:

```
df.plot.bar()
```

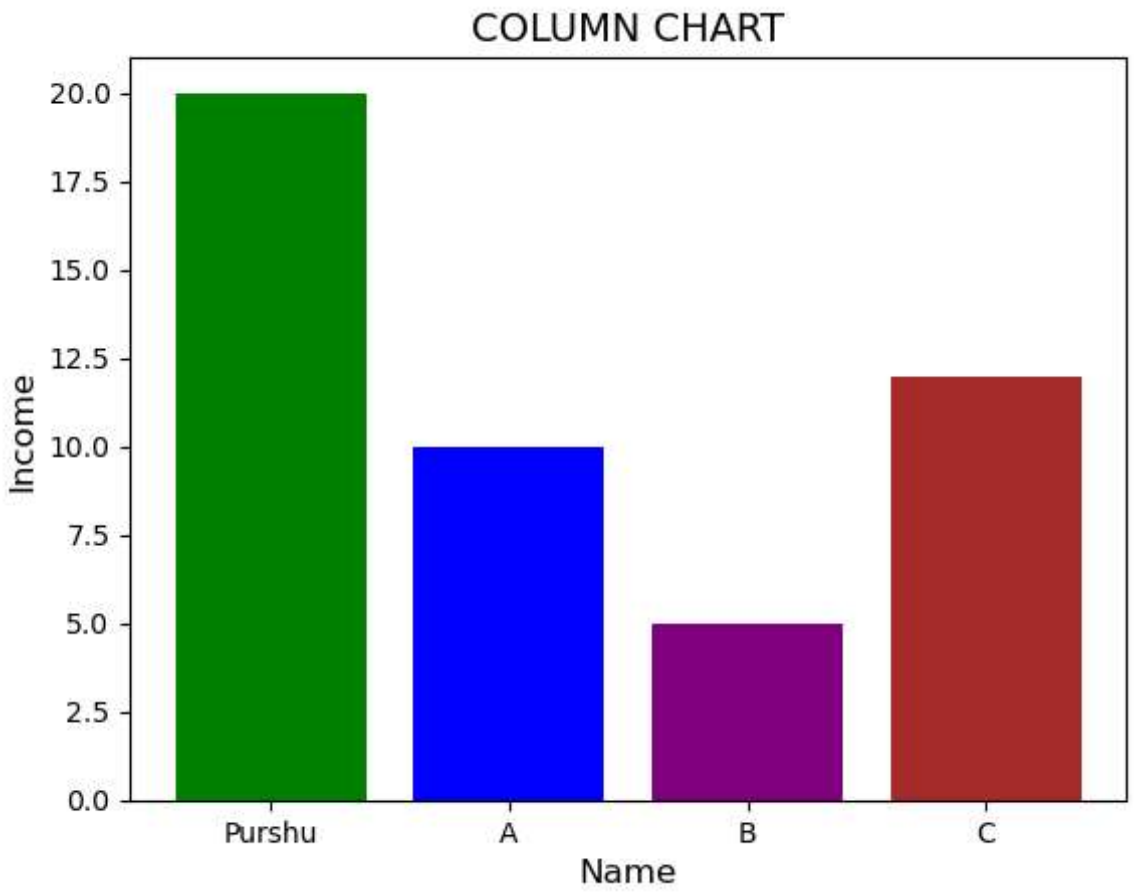
Out[10]:

<AxesSubplot:>



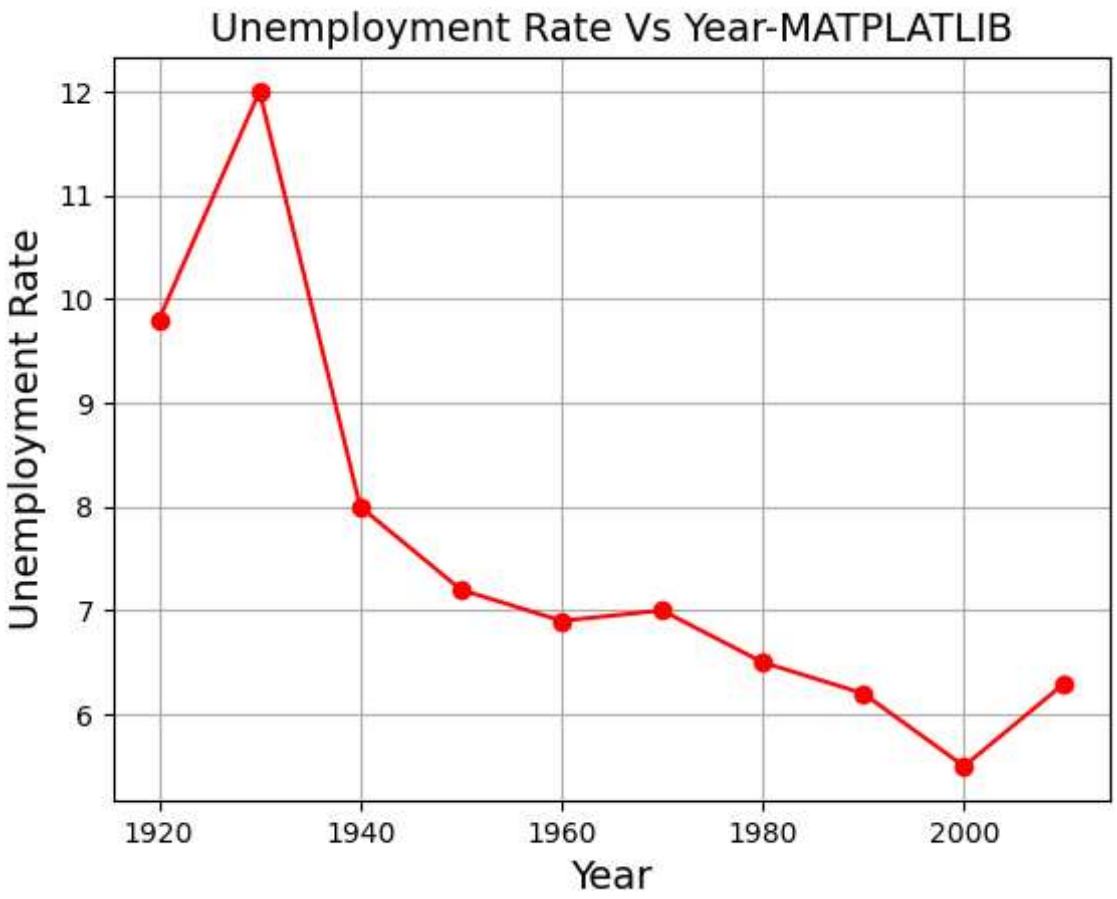
In [43]:

```
import matplotlib.pyplot as plt

NAME = ['Purshu','A','B','C']
INCOME=[20,10,5,12]
New_Colors = ['green','blue','purple','brown','teal']

plt.bar(NAME,INCOME,color=New_Colors)
plt.title('COLUMN CHART',fontsize=14)
plt.xlabel('Name',fontsize=12)
plt.ylabel('Income',fontsize=12)
plt.show()
```



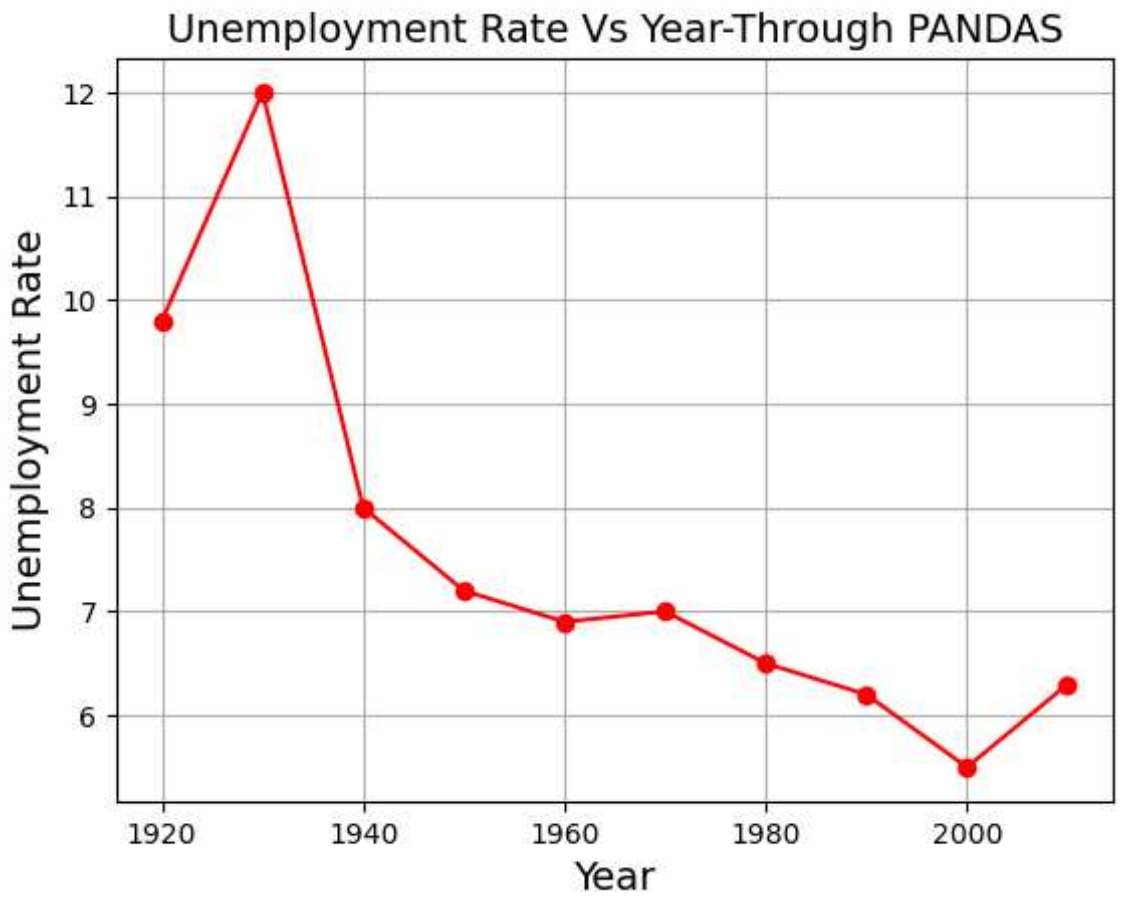
In [58]:

```
import matplotlib.pyplot as plt

Year = [1920,1930,1940,1950,1960,1970,1980,1990,2000,2010]
Unemployment_Rate = [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]

plt.plot(Year, Unemployment_Rate, color='red', marker='o')
plt.title('Unemployment Rate Vs Year-MATPLATLIB', fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Unemployment Rate', fontsize=14)
plt.grid(True)
plt.show()
```



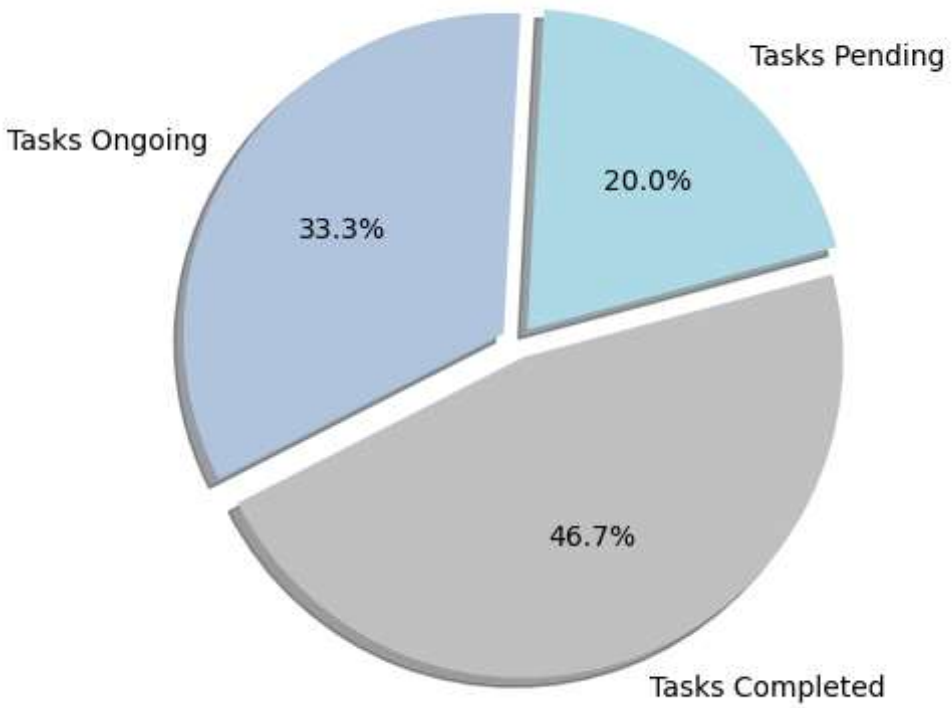
In [57]:

```
import pandas as pd
import matplotlib.pyplot as plt

Data = {'Year': [1920,1930,1940,1950,1960,1970,1980,1990,2000,2010],
        'Unemployment_Rate': [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]
       }

df = pd.DataFrame(Data,columns=['Year','Unemployment_Rate'])

plt.plot(df['Year'], df['Unemployment_Rate'], color='red', marker='o')
plt.title('Unemployment Rate Vs Year-Through PANDAS', fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Unemployment Rate', fontsize=14)
plt.grid(True)
plt.show()
```



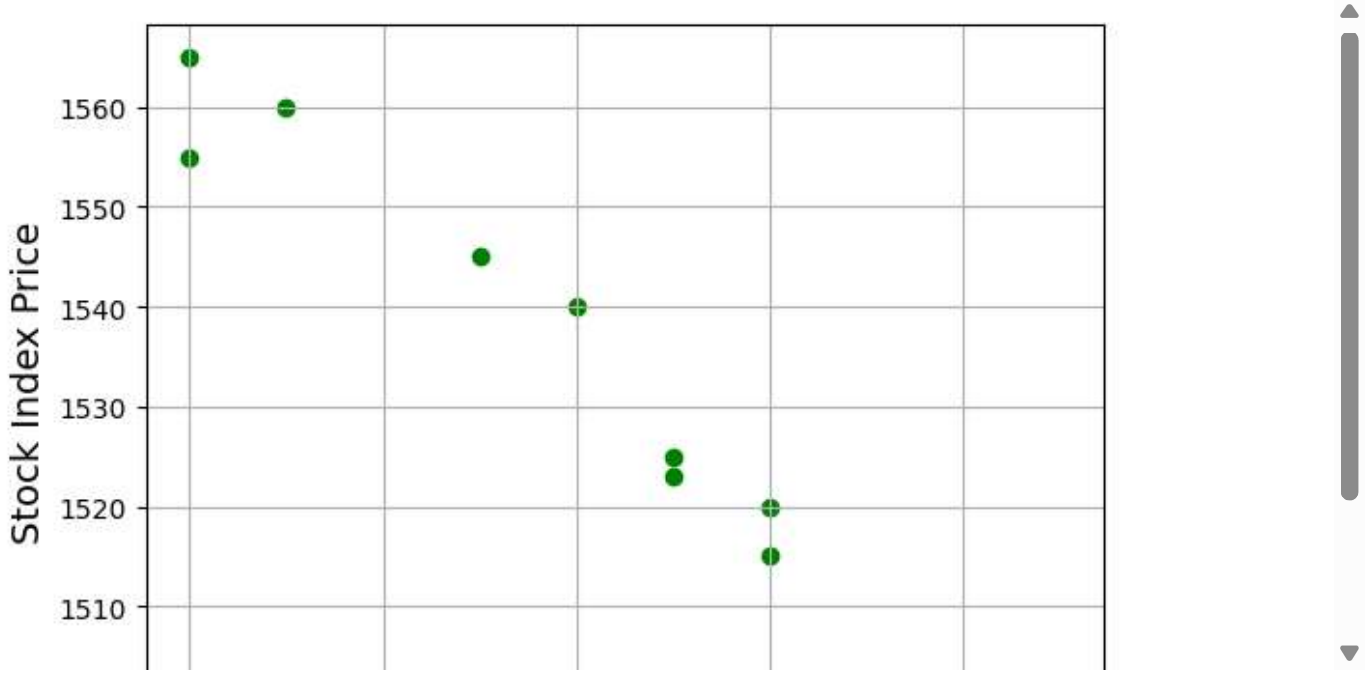
In [32]:

```
import matplotlib.pyplot as plt

Tasks = [300,500,700]

my_labels = 'Tasks Pending','Tasks Ongoing','Tasks Completed'
my_colors = ['lightblue','lightsteelblue','silver']
my_explode = (0.05,0.05,0.05)
plt.pie(Tasks, labels=my_labels, autopct='%1.1f%%', startangle=15, shadow = True, colors=
plt.axis('equal')
plt.show()
```



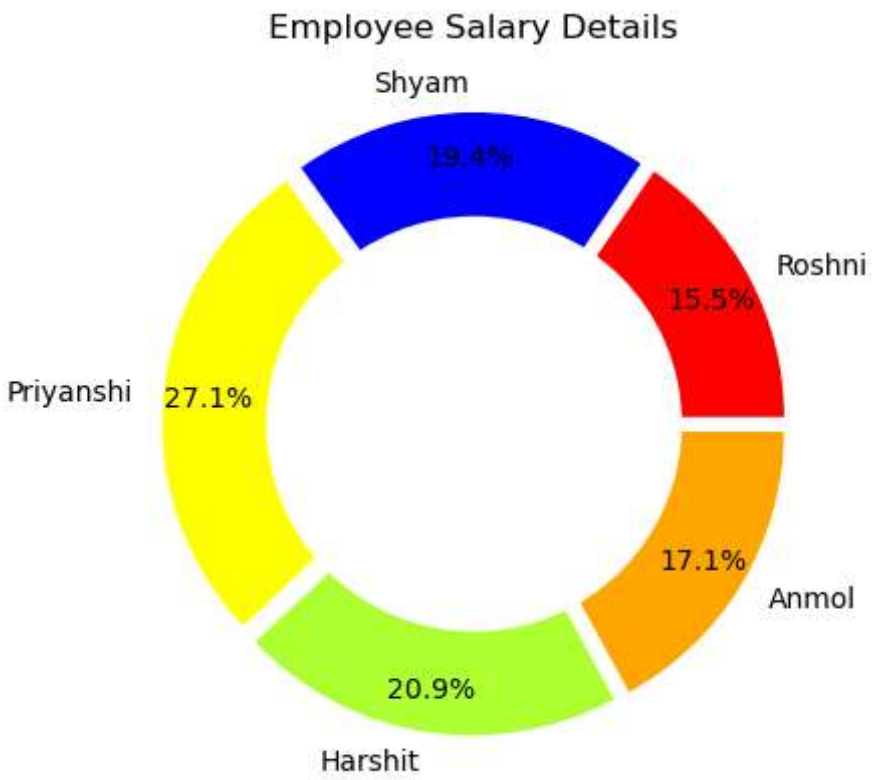
In [92]:

```
# SCATTERPLOT CHART
import matplotlib.pyplot as plt

Unemployment_Rate = [6.1,5.8,5.7,5.7,5.8,5.6,5.5,5.3,5.2,5.2]
Stock_Index_Price = [1500,1520,1525,1523,1515,1540,1545,1560,1555,1565]

plt.scatter(Unemployment_Rate, Stock_Index_Price, color='green')
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```



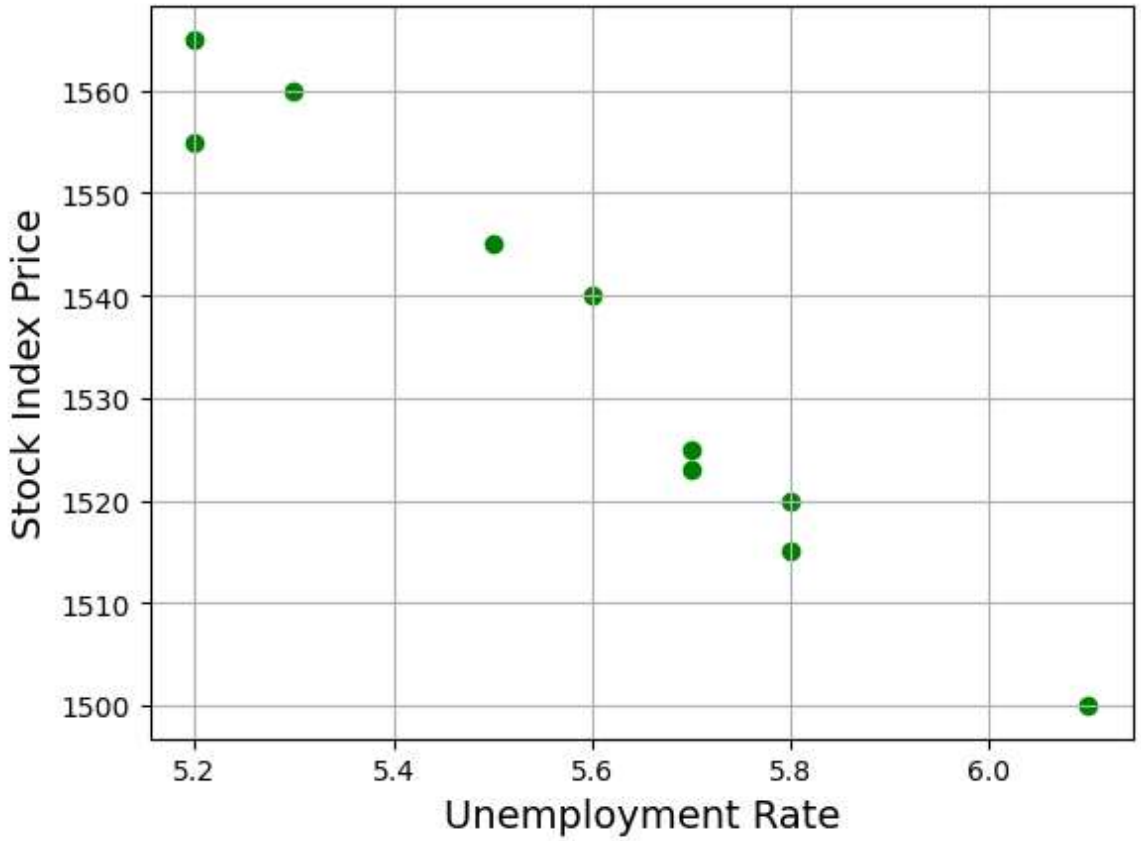
In [33]:

```
import matplotlib.pyplot as plt

Employee = ['Roshni', 'Shyam', 'Priyanshi','Harshit', 'Anmol']

Salary = [40000, 50000, 70000, 54000, 44000]

colors = ['#FF0000', '#0000FF', '#FFFF00','#ADFF2F', '#FFA500']

explode = (0.05, 0.05, 0.05, 0.05, 0.05)

plt.pie(Salary, colors=colors, labels=Employee,autopct='%1.1f%%', pctdistance=0.85,explod

# draw circle
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()

# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)

# Adding Title of chart
plt.title('Employee Salary Details')

# Displaying Chart
plt.show()
```



In [94]:

```
# SCATTERPLOT CHART THROUGH PANDAS

import pandas as pd
import matplotlib.pyplot as plt

data = {'Unemployment_Rate': [6.1,5.8,5.7,5.7,5.8,5.6,5.5,5.3,5.2,5.2],
        'Stock_Index_Price': [1500,1520,1525,1523,1515,1540,1545,1560,1555,1565]
       }

df = pd.DataFrame(data,columns=['Unemployment_Rate','Stock_Index_Price'])

plt.scatter(df['Unemployment_Rate'], df['Stock_Index_Price'], color='green')
#plt.title('Unemployment Rate Vs Stock Index Price', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```



In [98]:

```
import pandas as pd

df = pd.read_excel(r'C:\Users\dhpur\OneDrive\Desktop\lap sales.xlsx')

print(df)
```

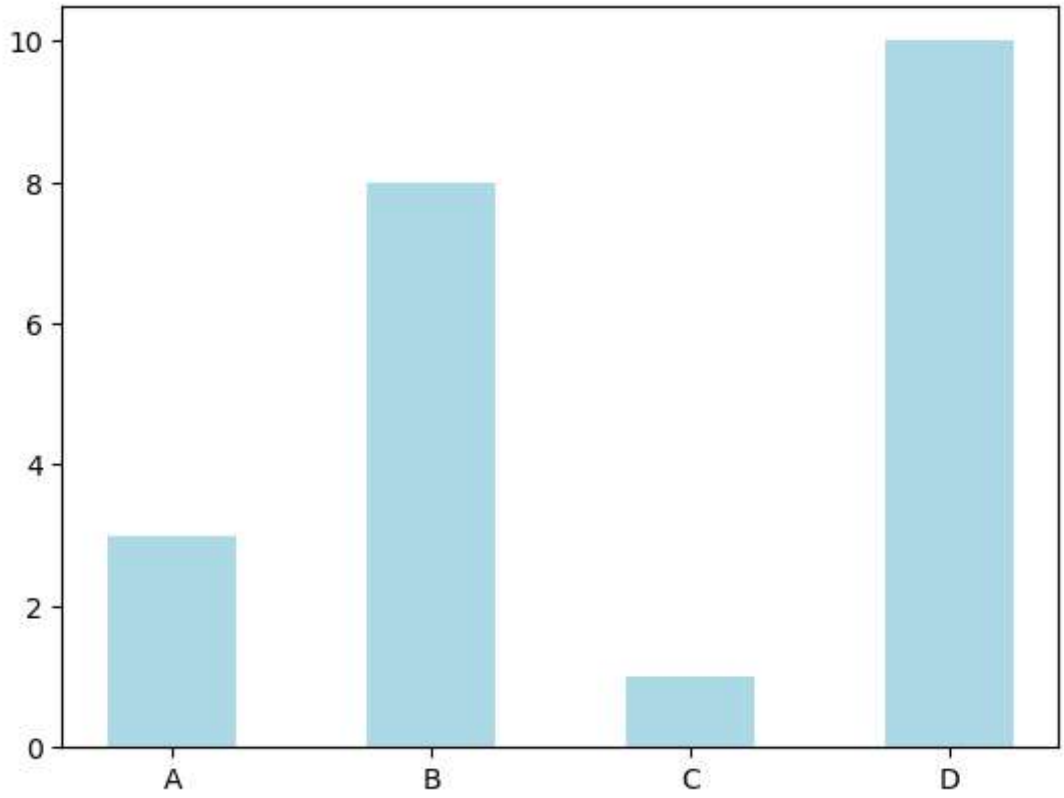
```
     laptop_ID  Price_euros Company            TypeName  Inches Resolution
\
0            2       898.94   Apple           Ultrabook    13.3         HD
1            3       575.00      HP            Notebook    15.6        FHD
2            4      2537.45   Apple           Ultrabook    15.4        QHD
3            5      1803.60   Apple           Ultrabook    13.3        QHD
4            6       400.00    Acer            Notebook    15.6         HD
..         ...          ...     ...                 ...     ...        ...
781       1286       209.00      HP             Netbook    11.6         HD
782       1288       638.00  Lenovo  2 in 1 Convertible    14.0        FHD
783       1289      1499.00  Lenovo  2 in 1 Convertible    13.3        QHD
784       1290       229.00  Lenovo            Notebook    14.0         HD
785       1291       764.00      HP            Notebook    15.6         HD

     Screen.Type Cpu.Vendor Cpu.Series  Cpu.Speed..GHz.  Ram..GB.  \
0    Unspecified      Intel         i5              1.8         8
1    Unspecified      Intel         i5              2.5         8
2            IPS      Intel         i7              2.7        16
3            IPS      Intel         i5              3.1         8
4    Unspecified        AMD       9420              3.0         4
..           ...        ...        ...              ...       ...
781  Unspecified      Intel  Dual Core              1.6         2
782          IPS      Intel         i7              2.5         4
783          IPS      Intel         i7              2.5        16
784  Unspecified      Intel  Dual Core              1.6         2
785  Unspecified      Intel         i7              2.5         6

    Storage.Type  Memory..GB. Gpu.Vendor OpSys  Weight..Kg.  average_ratin
g  \
0            SSD          128      Intel   mac         1.34             4.
4
1            SSD          256      Intel   no          1.86             5.
0
2            SSD          512        AMD   mac         1.83             3.
9
3            SSD          256      Intel   mac         1.37             4.
6
4            HDD          500        AMD   win         2.10             3.
8
..           ...          ...        ...   ...          ...
...
781          SSD           32      Intel   win         1.17             4.
1
782          SSD          128      Intel   win         1.80             3.
3
783          SSD          512      Intel   win         1.30             3.
4
784          SSD           64      Intel   win         1.50             3.
6
785          HDD         1024        AMD   win         2.19             3.
8

     rating_count
0            2385
1               6
2              15
3             125
4              98
..            ...
781          2032
782            49
```

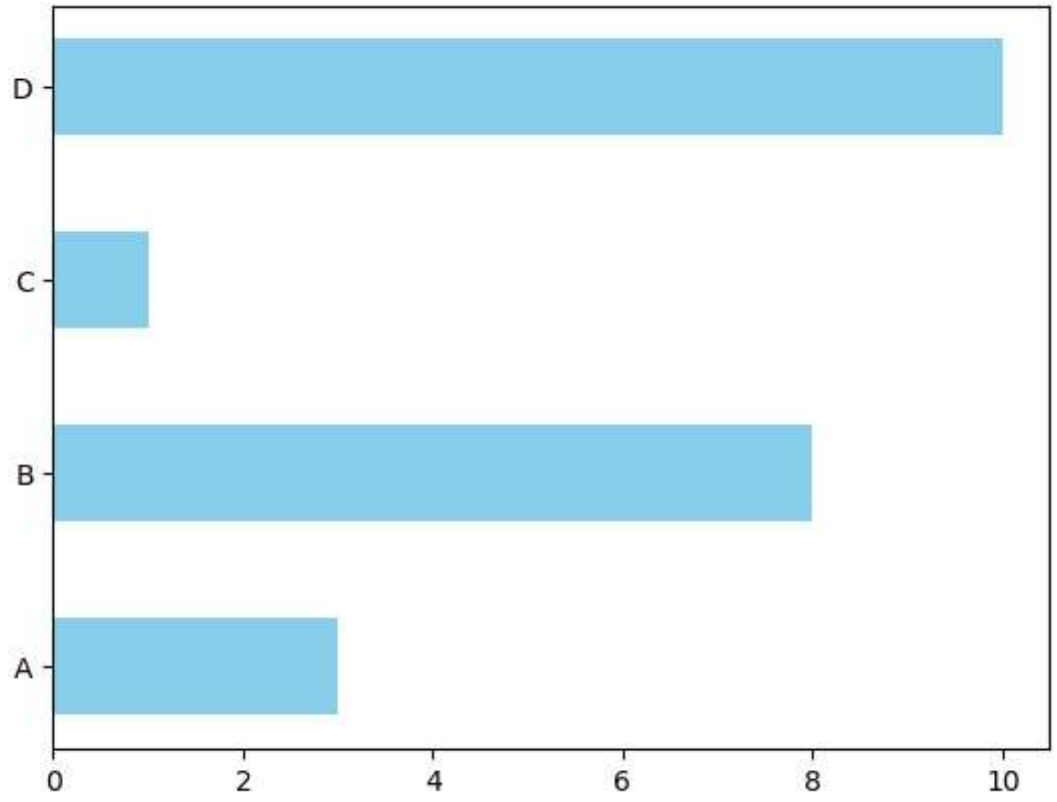
```
783            71
784            70
785            46

[786 rows x 18 columns]
```

#AREA CHART

In [36]:

```
import matplotlib.pyplot as plt
import numpy as np
import mkl

x=np.array(["A","B","C","D"])
y=np.array([3,8,1,10])

plt.bar(x,y,color="LightBlue",width=0.5)
plt.show()

plt.barh(x,y,color="SkyBlue",height=0.5)
plt.show()
```




D
C
B
A
0
2
4
6
8
10

In [27]:

```
import numpy as np

x=np.random.normal(170,10,250)

print(x)
```

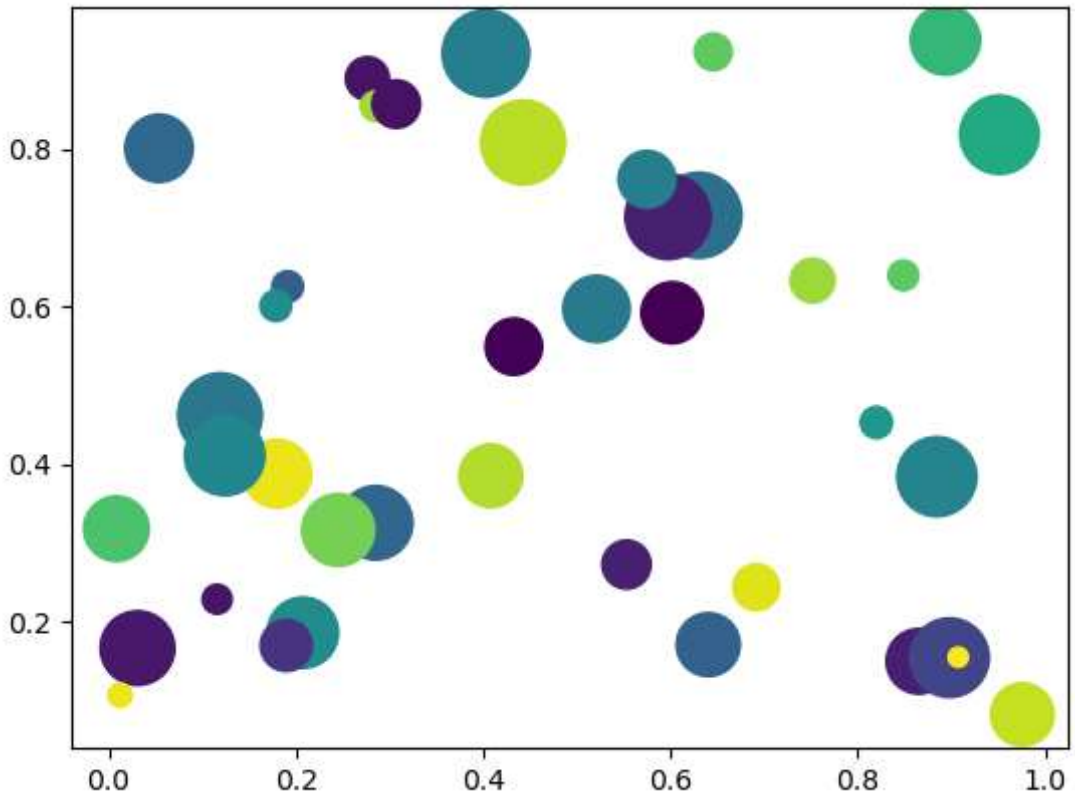
```
[174.74779528 158.08526168 167.83119159 153.06224746 172.35041303
 183.47567832 171.59452916 175.04703494 178.08894181 175.22708281
 161.76142453 187.09201995 194.05905156 176.1659066  164.5755972
 200.48864182 149.22303153 171.5326741  173.21999394 181.33842903
 170.23297849 177.63755787 174.98806722 170.88738144 172.98675062
 166.54618172 157.52881399 174.59064698 170.89098379 188.95198497
 179.45197    159.3073028  163.04353967 191.7017628  175.7032148
 180.11992007 169.54528851 157.36955931 165.61149186 189.53478577
 178.87739068 168.02014766 163.11284161 180.3952378  177.50374745
 187.20437612 150.98532661 183.81676717 158.93419949 184.643027
 174.62858232 153.2618526  176.58932008 173.47246692 174.91255016
 155.68867065 160.34416939 185.53192806 181.16954306 180.89307236
 164.91116444 191.09073498 186.44687534 175.08259251 166.32477978
 170.79136709 168.88053272 186.5583103  180.39687069 166.15231112
 179.30099201 176.94603257 160.52429301 170.12500248 196.44001487
 177.44645224 176.0354858  177.57387167 167.35374611 167.31126303
 176.67828654 186.13655208 164.42878804 161.6553956  177.30150527
 185.83156017 180.9964804  176.93981778 142.15419205 166.48441466
 160.02893954 163.53865418 178.28100947 173.72765128 171.64809771
 174.24778168 150.15342192 172.72956721 176.79863371 165.78530021
 169.64862735 173.94594284 157.01884111 171.38952971 182.65615984
 173.50000123 202.66429894 167.06728966 160.66729416 170.15935104
 184.19886254 177.68948542 163.76930167 159.41182862 173.80062467
 148.87345812 162.06951319 174.42701331 178.47444989 174.79122874
 169.65144176 156.9239145  178.12917482 173.4184064  181.6334223
 157.03140414 164.61986715 159.2695066  155.89508776 168.29208266
 179.10281845 174.18218785 177.61037356 162.45541268 163.62708523
 161.59930461 195.30569423 171.09417595 186.0611195  163.99558141
 172.22545819 170.40301122 167.6010443  169.17193129 167.46588571
 188.73317241 172.46529455 169.05930299 170.60832476 170.39014194
 181.71764884 166.72761625 184.20137132 161.00186716 178.44285137
 185.72817924 182.6251489  170.00200262 176.99612901 156.23436808
 170.18552314 174.57918499 176.64611461 181.28040718 159.08838677
 163.88592065 184.05929613 153.57328282 168.2537637  158.38563187
 171.91194994 165.19284413 178.250098   187.16727955 164.21086575
 184.91859261 180.34711212 168.27864703 166.39982734 179.99910998
 159.304236   173.22847688 165.84551293 170.47167989 155.79005387
 154.48853758 168.273248   160.20107897 176.39608049 164.31770043
 178.10007848 164.84817639 177.23534647 165.48711561 179.07266214
 151.81537865 179.34227463 191.25068598 176.86351106 171.41014114
 166.53306237 197.06281736 177.53830302 162.6579455  163.99100887
 169.14205736 175.65433905 162.98581569 161.63149333 152.80498982
 170.33806003 172.56314046 162.04006408 183.38126192 170.61099988
 162.30228322 160.1154444  165.39644181 162.35058645 146.43624114
 173.97075403 182.28399242 166.20124054 172.35843317 170.97942439
 168.31646306 169.76593713 174.05742648 173.59623856 160.41419045
 178.33684536 172.82952092 166.88234224 170.36950627 173.0444376
 187.21637959 169.85154461 161.24934784 173.21347831 168.43805075
 157.14672277 180.09701469 182.22883529 172.46315121 142.36609418
 175.43282345 167.35558632 164.75250264 172.00566314 162.71142247]
```

```
plt.hist(x)
plt.show()
```

BUBBLE CHARTS

In [48]:

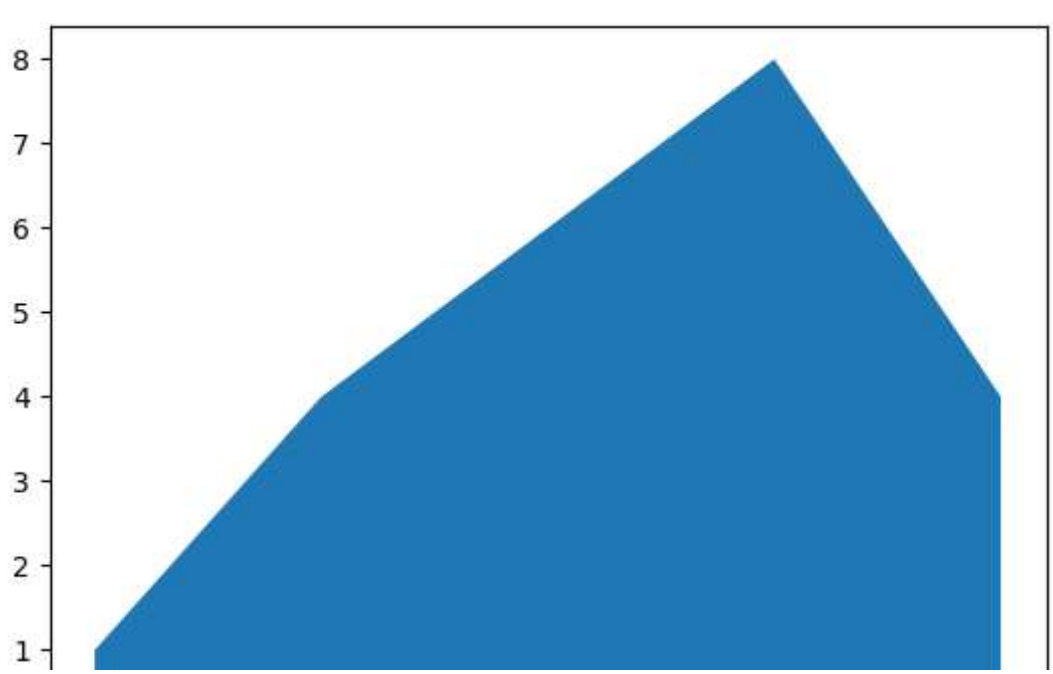
```
import matplotlib.pyplot as plt
import numpy as np

# create data
x = np.random.rand(40)
y = np.random.rand(40)
z = np.random.rand(40)
Varnas = np.random.rand(40)
# use the scatter function
plt.scatter(x,y,s=z*1000,c=Varnas)
plt.show()
```



AREA CHARTS

In [4]:

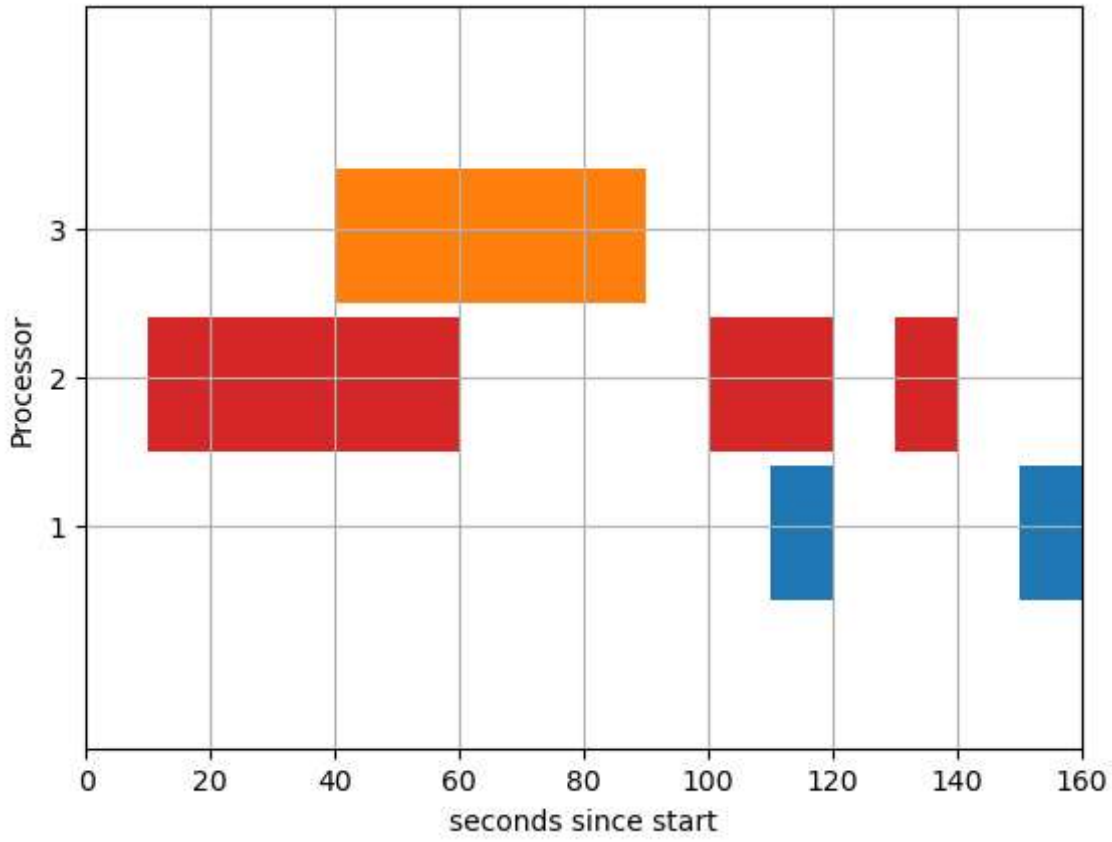
```
import numpy as np
import matplotlib.pyplot as plt

x=range(1,6)
y=[1,4,6,8,4]

plt.fill_between(x,y)
plot.show()
```



Gaant Chart

In [1]:

```
# Importing the matplotlib.pyplot
import matplotlib.pyplot as plt

# Declaring a figure "gnt"
fig, gnt = plt.subplots()

# Setting Y-axis limits
gnt.set_ylim(0, 50)

# Setting X-axis limits
gnt.set_xlim(0, 160)

# Setting labels for x-axis and y-axis
gnt.set_xlabel('seconds since start')
gnt.set_ylabel('Processor')

# Setting ticks on y-axis
gnt.set_yticks([15, 25, 35])
# Labelling tickes of y-axis
gnt.set_yticklabels(['1', '2', '3'])

# Setting graph attribute
gnt.grid(True)

# Declaring a bar in schedule
gnt.broken_barh([(40, 50)], (30, 9), facecolors =('tab:orange'))

# Declaring multiple bars in at same level and same width
gnt.broken_barh([(110, 10), (150, 10)], (10, 9),
                        facecolors ='tab:blue')

gnt.broken_barh([(10, 50), (100, 20), (130, 10)], (20, 9),
                                facecolors =('tab:red'))

plt.savefig("gantt1.png")
```



TREE MAPS

In [10]:

```
import seaborn as sns
import squarify
import matplotlib.pyplot as plt

titanic = sns.load_dataset('titanic')

a = titanic.groupby('class')[['survived']].sum().index.get_level_values(0).tolist()

d = titanic.groupby('class')[['survived']].sum().reset_index().survived.values.tolist()

squarify.plot(sizes=d,label=a, alpha=.8 )
plt.axis('off')
plt.show()
```

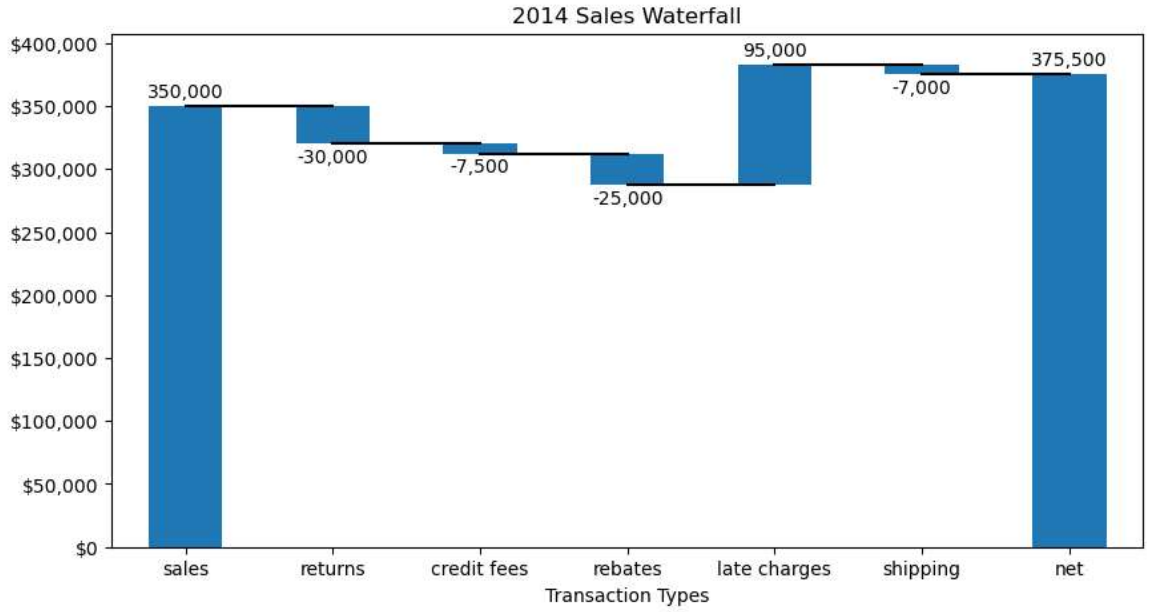
```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_15264\3852048875.py in <module>
      9 d = titanic.groupby('class')[['survived']].sum().reset_index().sur
vived.values.tolist()
     10
---> 11 squarify.plot(sizes=d,label=a, alpha=.8 )
     12 plt.axis('off')
     13 plt.show()

NameError: name 'squarify' is not defined
```

Water Fall chart

In [13]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

#Use python 2.7+ syntax to format currency
def money(x, pos):
    'The two args are the value and tick position'
    return "${:,.0f}".format(x)
formatter = FuncFormatter(money)

#Data to plot. Do not include a total, it will be calculated
index = ['sales','returns','credit fees','rebates','late charges','shipping']
data = {'amount': [350000,-30000,-7500,-25000,95000,-7000]}

#Store data and create a blank series to use for the waterfall
trans = pd.DataFrame(data=data,index=index)
blank = trans.amount.cumsum().shift(1).fillna(0)

#Get the net total number for the final element in the waterfall
total = trans.sum().amount
trans.loc["net"]= total
blank.loc["net"] = total

#The steps graphically show the levels as well as used for label placement
step = blank.reset_index(drop=True).repeat(3).shift(-1)
step[1::3] = np.nan

#When plotting the last element, we want to show the full bar,
#Set the blank to 0
blank.loc["net"] = 0

#Plot and label
my_plot = trans.plot(kind='bar', stacked=True, bottom=blank,legend=None, figsize=(10, 5),
my_plot.plot(step.index, step.values,'k')
my_plot.set_xlabel("Transaction Types")

#Format the axis for dollars
my_plot.yaxis.set_major_formatter(formatter)

#Get the y-axis position for the labels
y_height = trans.amount.cumsum().shift(1).fillna(0)

#Get an offset so labels don't sit right on top of the bar
max = trans.max()
neg_offset = max / 25
pos_offset = max / 50
plot_offset = int(max / 15)

#Start label loop
loop = 0
for index, row in trans.iterrows():
    # For the last item in the list, we don't want to double count
    if row['amount'] == total:
        y = y_height[loop]
    else:
        y = y_height[loop] + row['amount']
    # Determine if we want a neg or pos offset
    if row['amount'] > 0:
```

```
        y += pos_offset
    else:
        y -= neg_offset
    my_plot.annotate("{:,.0f}".format(row['amount']),(loop,y),ha="center")
    loop+=1

#Scale up the y axis so there is room for the labels
my_plot.set_ylim(0,blank.max()+int(plot_offset))
#Rotate the labels
my_plot.set_xticklabels(trans.index,rotation=0)
my_plot.get_figure().savefig("waterfall.png",dpi=200,bbox_inches='tight')
```



In [ ]: