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

plt.style.use('ggplot')
```

Colored Scatterplots

In [2]:
```
iris = pd.read_csv("iris.csv")
iris.sample(5)
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
91	92	6.1	3.0	4.6	1.4	Iris-versicolor
108	109	6.7	2.5	5.8	1.8	Iris-virginica
93	94	5.0	2.3	3.3	1.0	Iris-versicolor
147	148	6.5	3.0	5.2	2.0	Iris-virginica
139	140	6.9	3.1	5.4	2.1	Iris-virginica

In [3]:
```
# Replacing

iris['Species'] = iris['Species'].replace({'Iris-setosa':0 ,
                                           'Iris-versicolor':1 ,
                                           'Iris-virginica':2})

iris.sample(5)
```

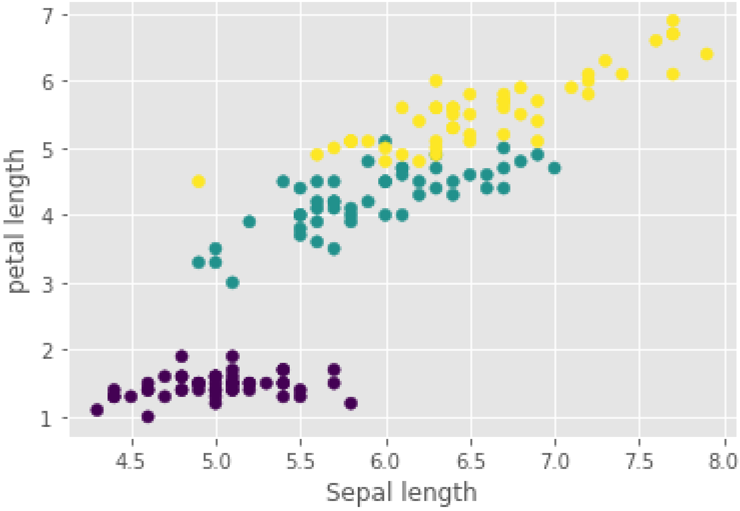
Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
11	12	4.8	3.4	1.6	0.2	0
119	120	6.0	2.2	5.0	1.5	2
112	113	6.8	3.0	5.5	2.1	2
135	136	7.7	3.0	6.1	2.3	2
88	89	5.6	3.0	4.1	1.3	1

C - colors

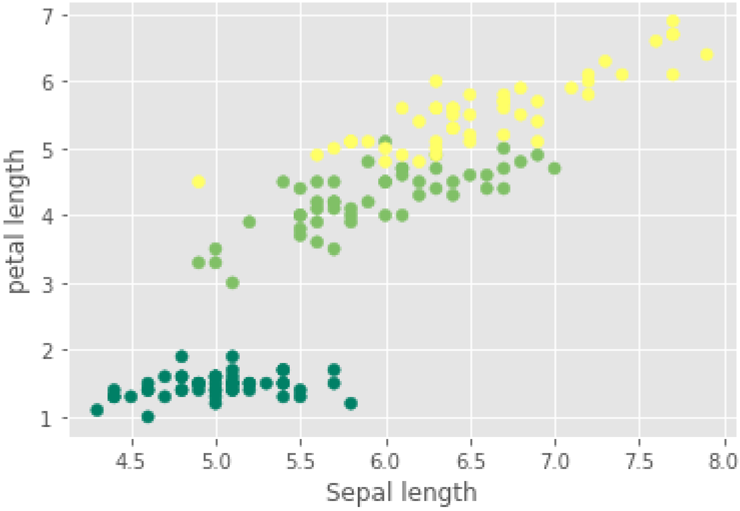
In [4]:

```
plt.scatter( iris['SepalLengthCm'],iris['PetalLengthCm'],
            c= iris['Species']) # c

plt.xlabel('Sepal length')
plt.ylabel('petal length')

plt.show()
```

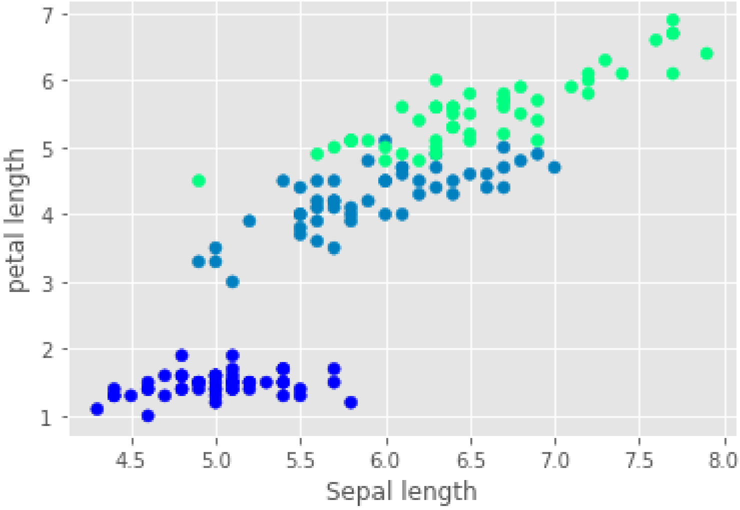


cmap

In [5]:
```
# different themes - 'cmap'

plt.scatter( iris['SepalLengthCm'],iris['PetalLengthCm'],
            c= iris['Species'] ,cmap ='summer') # cmap

plt.xlabel('Sepal length')
plt.ylabel('petal length')

plt.show()
```

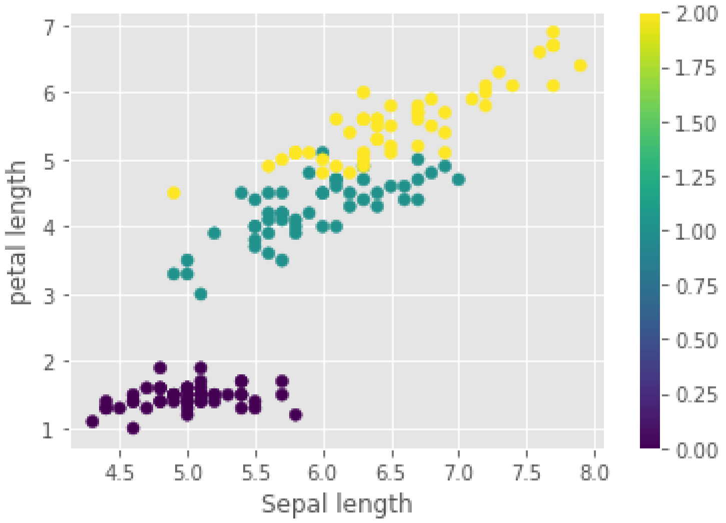


In [6]:
```
plt.scatter( iris['SepalLengthCm'],iris['PetalLengthCm'],
            c= iris['Species'] ,cmap ='winter') # winter

plt.xlabel('Sepal length')
plt.ylabel('petal length')

plt.show()
```



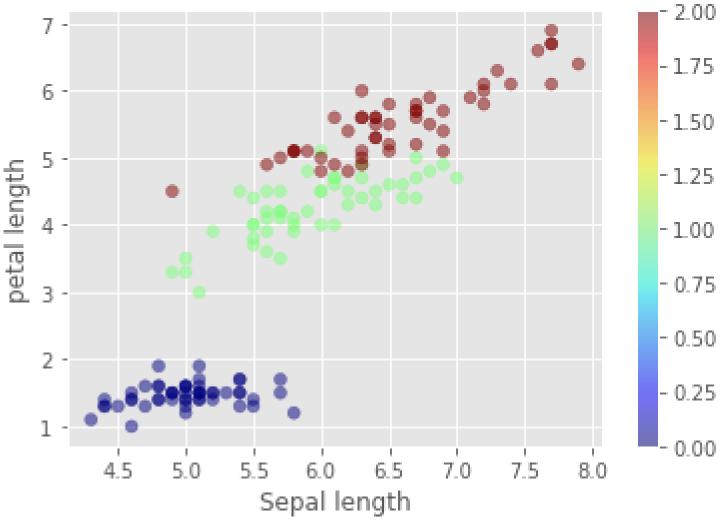
color bar

In [7]:
```
plt.scatter( iris['SepalLengthCm'],iris['PetalLengthCm'],
            c= iris['Species'] ,cmap ='viridis')

plt.xlabel('Sepal length')
plt.ylabel('petal length')

plt.colorbar() # colorbar shows variations levels

plt.show()
```



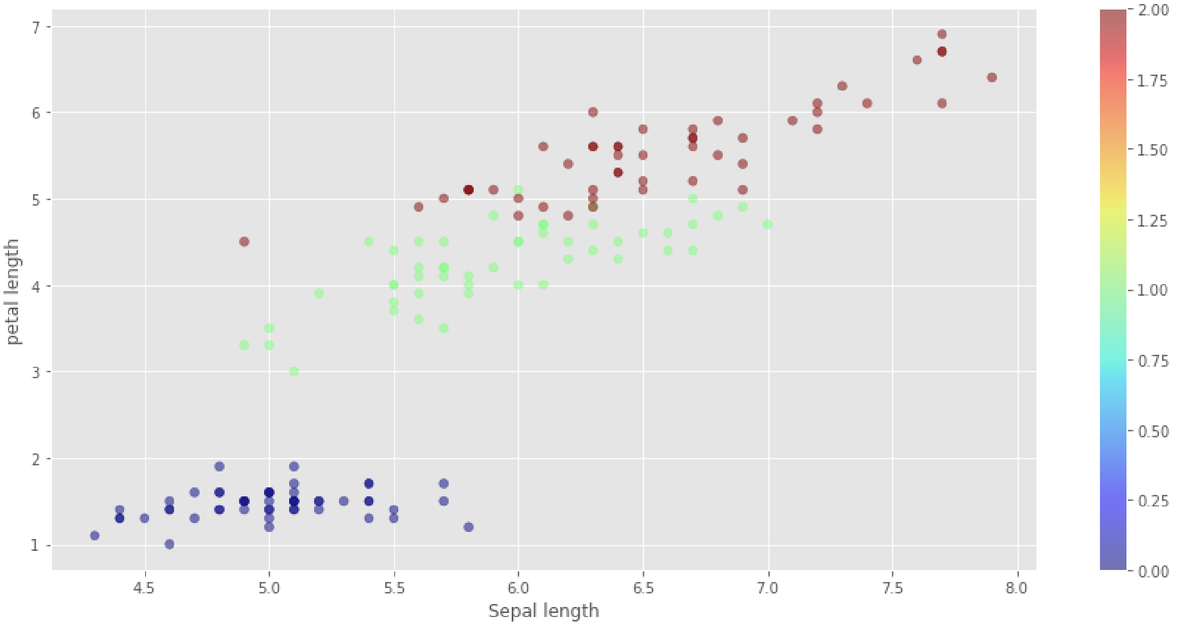
alpha

In [8]:
```
plt.scatter( iris['SepalLengthCm'],iris['PetalLengthCm'],
            c= iris['Species'] ,cmap ='jet',
            alpha =0.5) # reduces,increase color

plt.xlabel('Sepal length')
plt.ylabel('petal length')

plt.colorbar()

plt.show()
```



Plot size

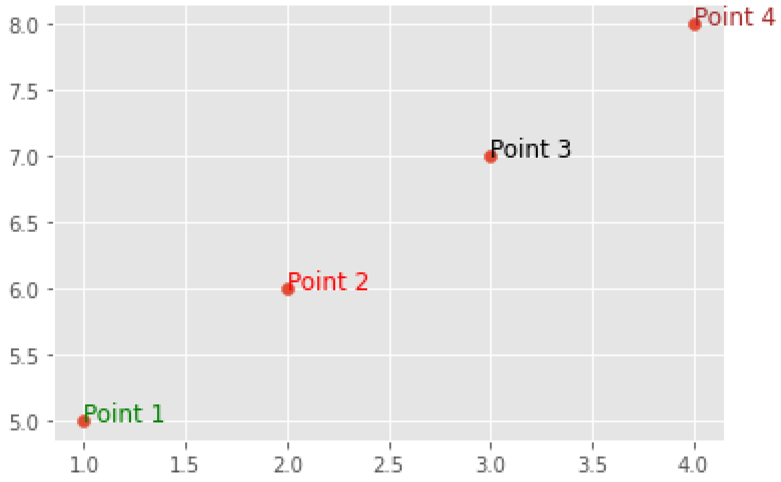
In [9]:
```
plt.figure(figsize=(15,7)) # 15 -width , 7 -height

plt.scatter( iris['SepalLengthCm'],iris['PetalLengthCm'],
            c= iris['Species'] ,cmap ='jet', alpha =0.5)   # reduces color

plt.xlabel('Sepal length')
plt.ylabel('petal length')

plt.colorbar()

plt.show()
```



Annotations

In [10]:
```
# sample annotation - naming

x = [1,2,3,4]
y = [5,6,7,8]

plt.scatter(x,y)
plt.text(1,5,'Point 1',fontdict={'size':12,'color':'green'})
plt.text(2,6,'Point 2',fontdict={'size':12,'color':'red'})
plt.text(3,7,'Point 3',fontdict={'size':12,'color':'black'})
plt.text(4,8,'Point 4',fontdict={'size':12,'color':'brown'})
```

Out[10]:
```
Text(4, 8, 'Point 4')
```



In [11]:
```
batter =pd.read_csv("batter.csv")
batter.shape
```

Out[11]:
```
(605, 4)
```

In [12]:
```
# sample =25

sample_df =batter.head(100).sample(25,random_state=29)
```

In [13]:
```
sample_df.shape
```

Out[13]:
```
(25, 4)
```

In [14]:

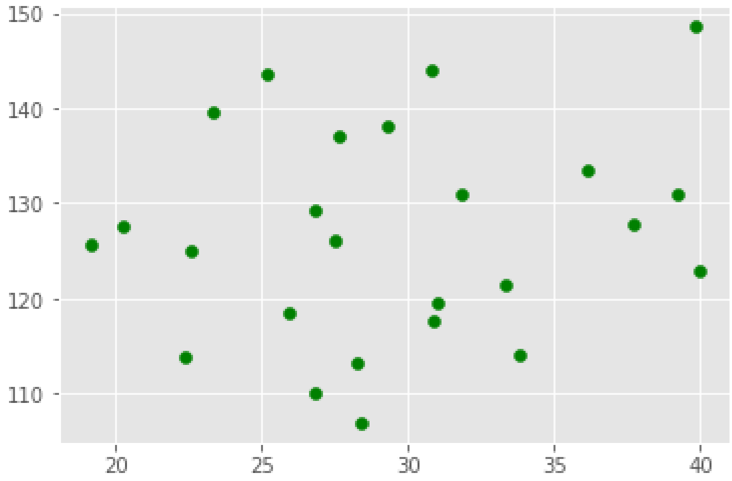
```
sample_df
```

Out[14]:

	batter	runs	avg	strike_rate
73	TM Dilshan	1153	26.813953	110.124164
8	RV Uthappa	4954	27.522222	126.152279
10	G Gambhir	4217	31.007353	119.665153
7	MS Dhoni	4978	39.196850	130.931089
19	YK Pathan	3222	29.290909	138.046272
71	RD Gaikwad	1207	37.718750	127.724868
92	STR Binny	880	19.130435	125.714286
25	Q de Kock	2767	31.804598	130.951254
38	SR Tendulkar	2334	33.826087	114.187867
12	AM Rahane	4074	30.863636	117.575758
57	DJ Bravo	1560	22.608696	125.100241
52	RA Tripathi	1798	27.661538	137.042683
77	LMP Simmons	1079	39.962963	122.892938
5	AB de Villiers	5181	39.853846	148.580442
41	R Dravid	2174	28.233766	113.347237
98	GC Smith	739	28.423077	106.946454
70	DJ Hooda	1237	20.278689	127.525773
33	DA Miller	2455	36.102941	133.569097
84	Y Venugopal Rao	985	22.386364	113.872832
55	KC Sangakkara	1687	25.953846	118.469101
63	BJ Hodge	1400	33.333333	121.422376
90	MM Ali	910	23.333333	139.570552
94	SO Hetmyer	831	30.777778	144.020797
56	PP Shaw	1588	25.206349	143.580470
9	KD Karthik	4377	26.852761	129.267572

```
In [41]: plt.scatter(sample_df['avg'],sample_df['strike_rate'], color='green')
```

```
Out[41]: <matplotlib.collections.PathCollection at 0x25b49b5d190>
```



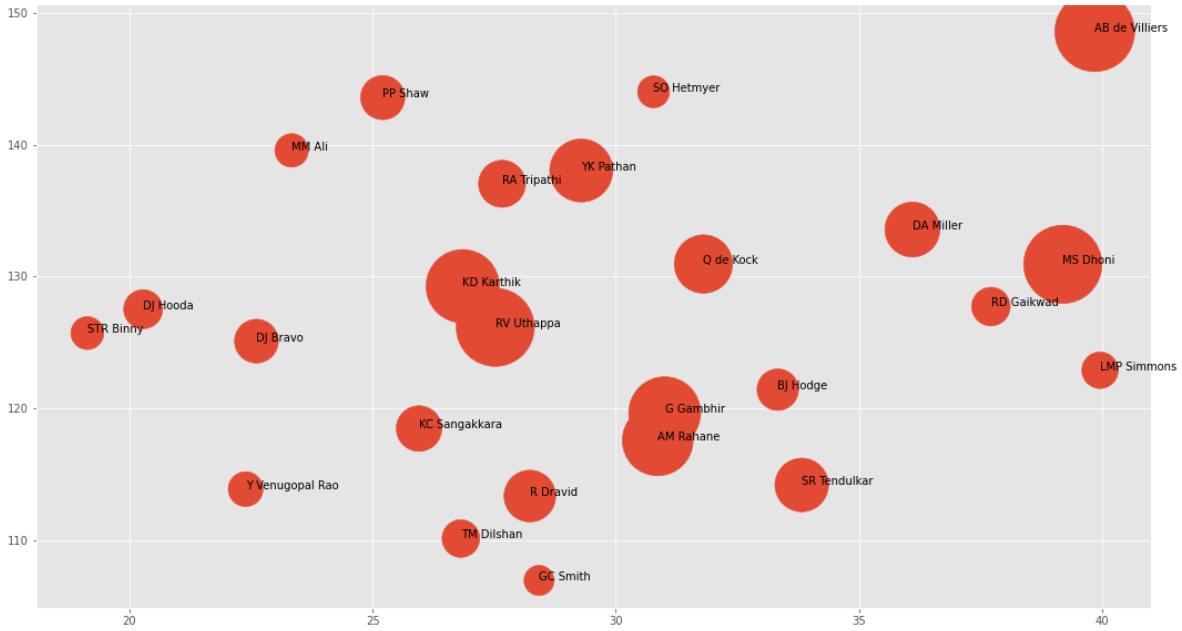
In [16]:

```
sample_df
```

Out[16]:

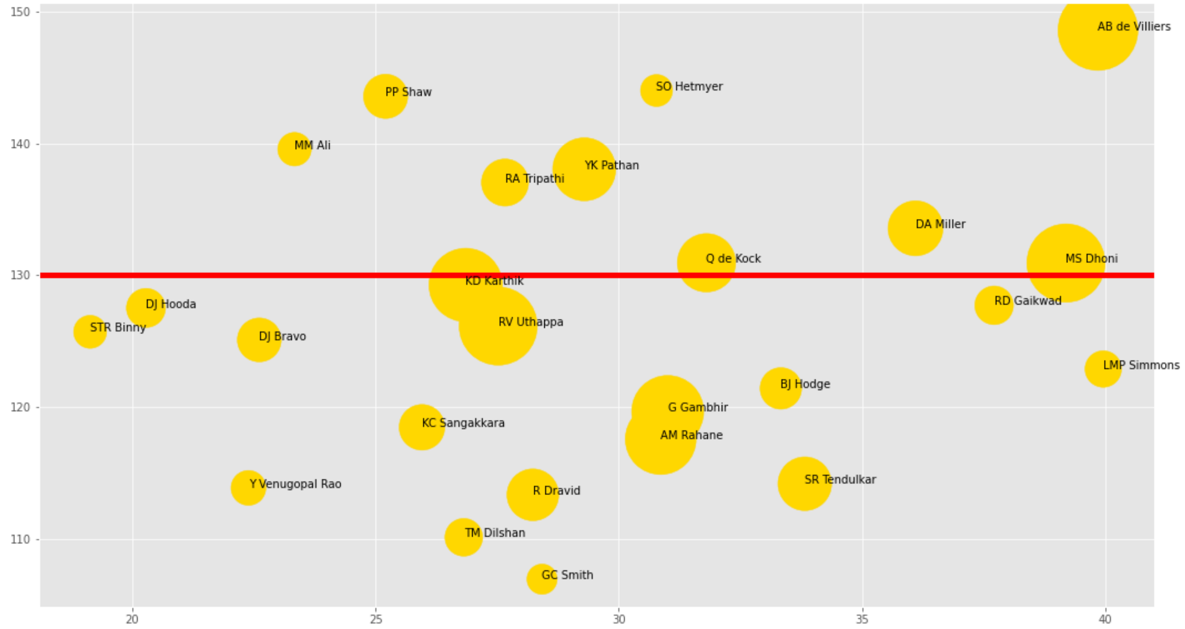
	batter	runs	avg	strike_rate
73	TM Dilshan	1153	26.813953	110.124164
8	RV Uthappa	4954	27.522222	126.152279
10	G Gambhir	4217	31.007353	119.665153
7	MS Dhoni	4978	39.196850	130.931089
19	YK Pathan	3222	29.290909	138.046272
71	RD Gaikwad	1207	37.718750	127.724868
92	STR Binny	880	19.130435	125.714286
25	Q de Kock	2767	31.804598	130.951254
38	SR Tendulkar	2334	33.826087	114.187867
12	AM Rahane	4074	30.863636	117.575758
57	DJ Bravo	1560	22.608696	125.100241
52	RA Tripathi	1798	27.661538	137.042683
77	LMP Simmons	1079	39.962963	122.892938
5	AB de Villiers	5181	39.853846	148.580442
41	R Dravid	2174	28.233766	113.347237
98	GC Smith	739	28.423077	106.946454
70	DJ Hooda	1237	20.278689	127.525773
33	DA Miller	2455	36.102941	133.569097
84	Y Venugopal Rao	985	22.386364	113.872832
55	KC Sangakkara	1687	25.953846	118.469101
63	BJ Hodge	1400	33.333333	121.422376
90	MM Ali	910	23.333333	139.570552
94	SO Hetmyer	831	30.777778	144.020797
56	PP Shaw	1588	25.206349	143.580470
9	KD Karthik	4377	26.852761	129.267572

In [17]:
```
plt.figure(figsize=(18,10))

plt.scatter(sample_df['avg'],sample_df['strike_rate'] ,
            s =sample_df['runs']) # s = size

for i in range(sample_df.shape[0]):
    plt.text(sample_df['avg'].values[i],
            sample_df['strike_rate'].values[i],
            sample_df['batter'].values[i])
```

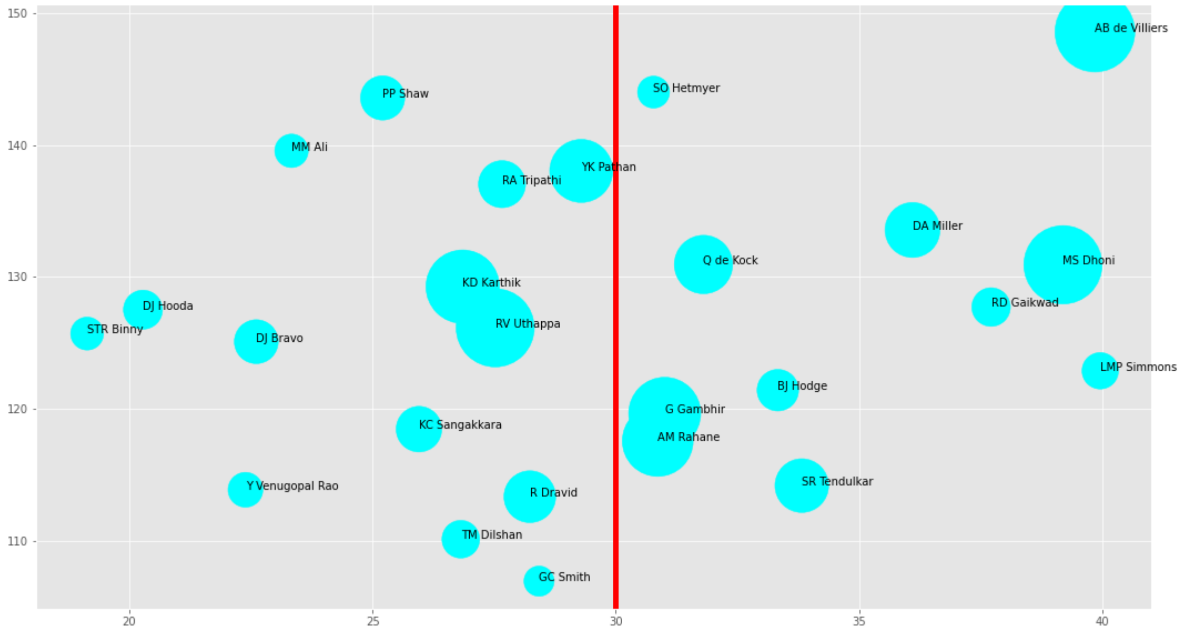


Horizontal and Vertical Lines

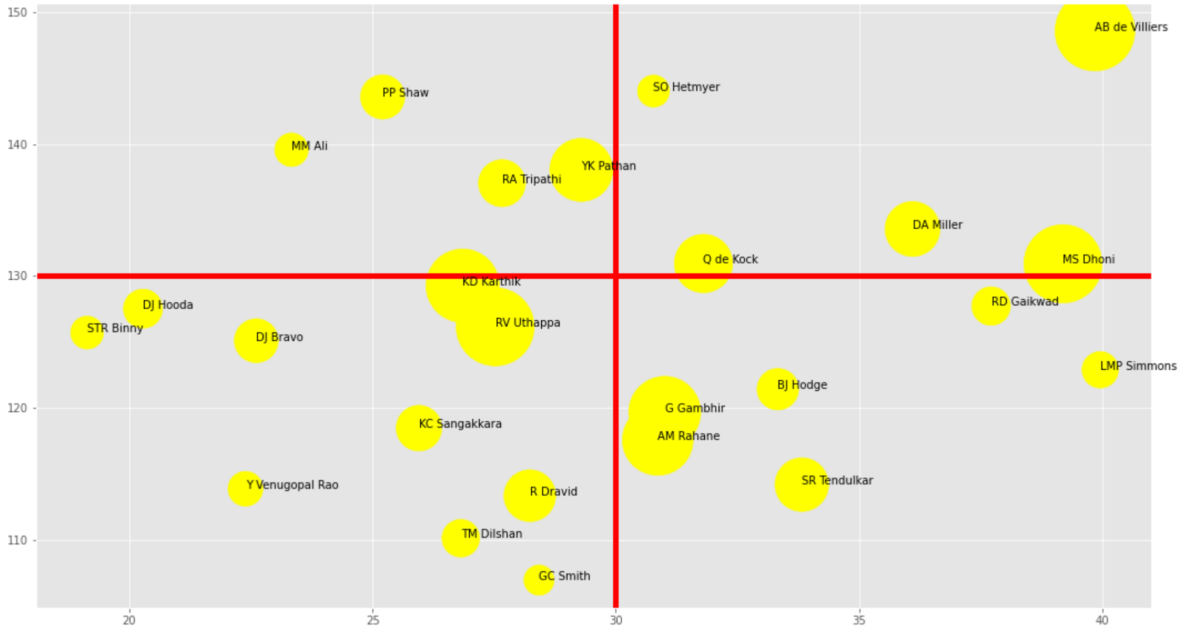
In [147]:
```
plt.figure(figsize=(18,10))

plt.axhline(130, color ='red', linewidth=5) #  Horizontal line

plt.scatter(sample_df['avg'],sample_df['strike_rate'] ,
           s =sample_df['runs'], color='gold')

for i in range(sample_df.shape[0]):
    plt.text(sample_df['avg'].values[i],
            sample_df['strike_rate'].values[i],
            sample_df['batter'].values[i])
```



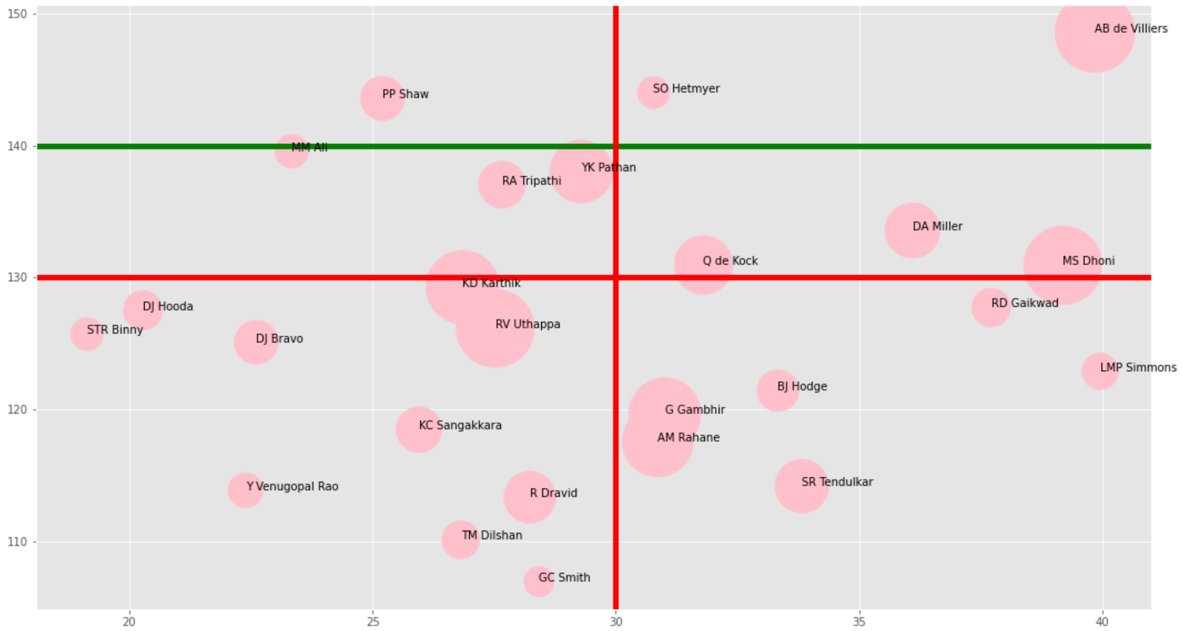
In [148]:
```
plt.figure(figsize=(18,10))

plt.axvline(30, color ='red',linewidth=5) # Vertical line

plt.scatter(sample_df['avg'],sample_df['strike_rate'] ,
           s =sample_df['runs'], color ='cyan')

for i in range(sample_df.shape[0]):
    plt.text(sample_df['avg'].values[i],
            sample_df['strike_rate'].values[i],
            sample_df['batter'].values[i])
```



In [149]:
```
plt.figure(figsize=(18,10))

plt.axhline(130, color ='red',linewidth=5) #  Horizontal line
plt.axvline(30, color ='red',linewidth=5) #   Vertical line

plt.scatter(sample_df['avg'],sample_df['strike_rate'] ,
           s =sample_df['runs'], color ='yellow')

for i in range(sample_df.shape[0]):
    plt.text(sample_df['avg'].values[i],
            sample_df['strike_rate'].values[i],
            sample_df['batter'].values[i])
```



In [150]:
```
plt.figure(figsize=(18,10))

plt.axhline(130, color ='red',linewidth=5) #  Horizontal line
plt.axhline(140, color ='green',linewidth=5) # EXTRA Horizontal line
plt.axvline(30, color ='red',linewidth=5)

plt.scatter(sample_df['avg'],sample_df['strike_rate'] ,
           s =sample_df['runs'],color='pink')

for i in range(sample_df.shape[0]):
    plt.text(sample_df['avg'].values[i],
            sample_df['strike_rate'].values[i],
            sample_df['batter'].values[i])
```



Subplots

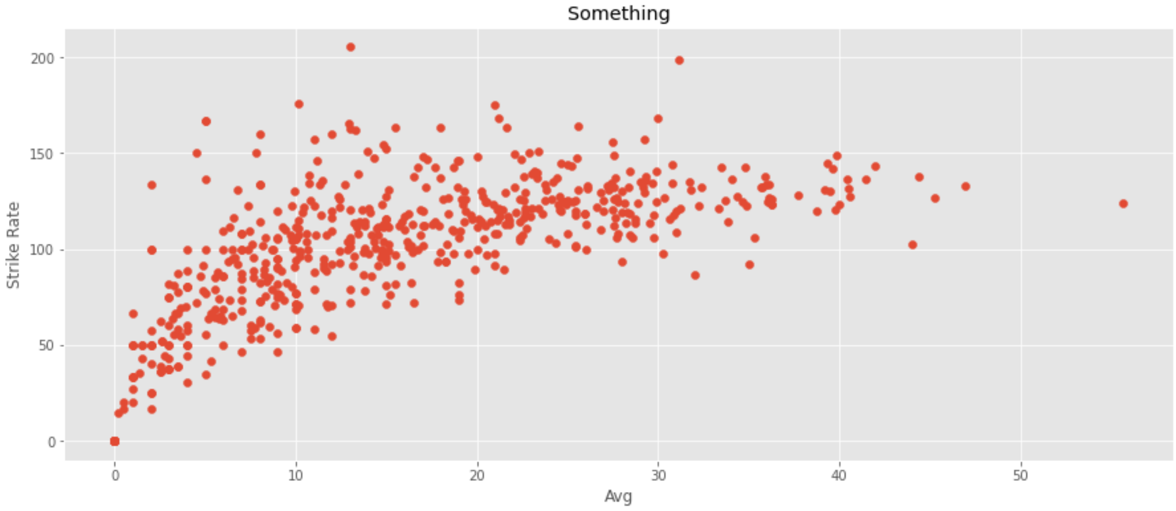
In [22]:
```
batter.head()
```

Out[22]:

	batter	runs	avg	strike_rate
0	V Kohli	6634	36.251366	125.977972
1	S Dhawan	6244	34.882682	122.840842
2	DA Warner	5883	41.429577	136.401577
3	RG Sharma	5881	30.314433	126.964594
4	SK Raina	5536	32.374269	132.535312

In [23]:

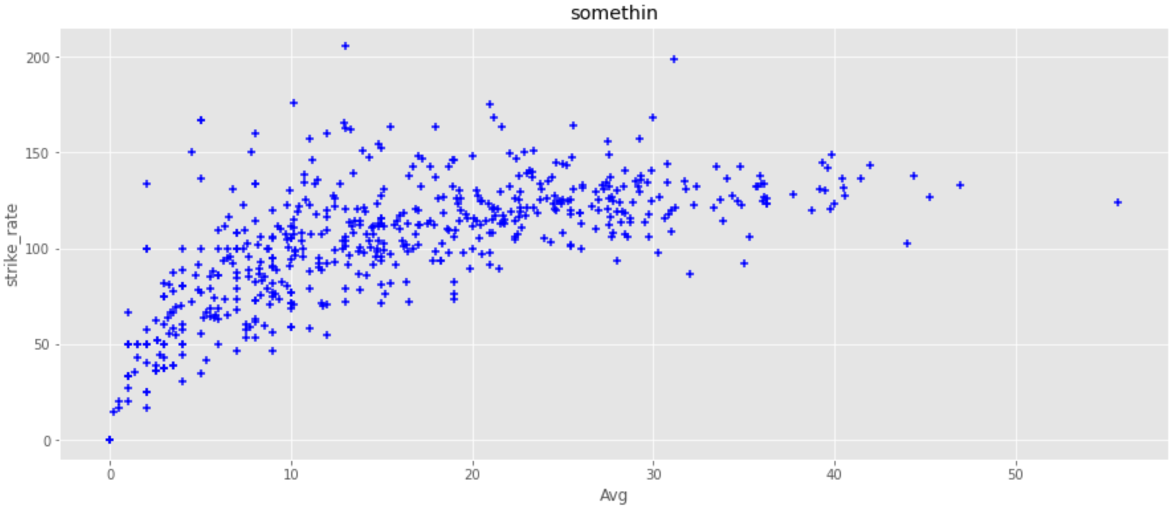
```
# Normal way

plt.figure(figsize=(15,6))
plt.scatter(batter['avg'],batter['strike_rate'])
plt.title('Something')
plt.xlabel('Avg')
plt.ylabel('Strike Rate')

plt.show()
```



In [43]:

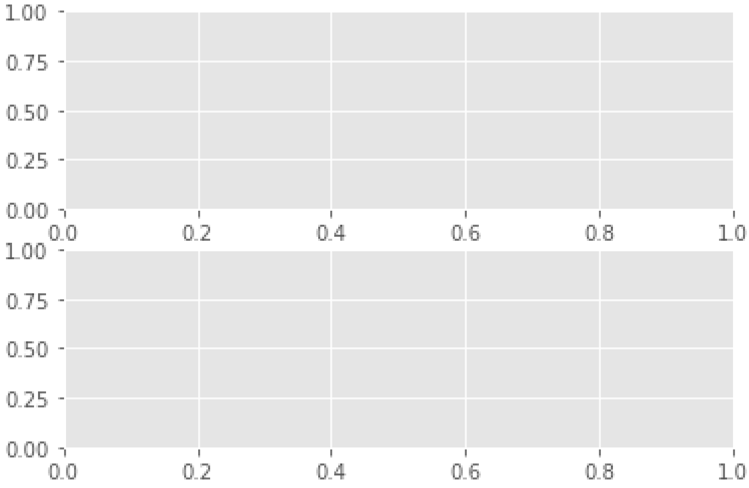
```
fig, ax = plt.subplots(figsize=(15,6))

ax.scatter(batter['avg'],batter['strike_rate'] , color ='blue' , marker = '+')
ax.set_title('somethin')
ax.set_xlabel('Avg')
ax.set_ylabel('strike_rate')

fig.show()
```

```
C:\Users\user\AppData\Local\Temp/ipykernel_14332/2843873373.py:8: UserWarnin
g: Matplotlib is currently using module://matplotlib_inline.backend_inline, w
hich is a non-GUI backend, so cannot show the figure.
  fig.show()
```



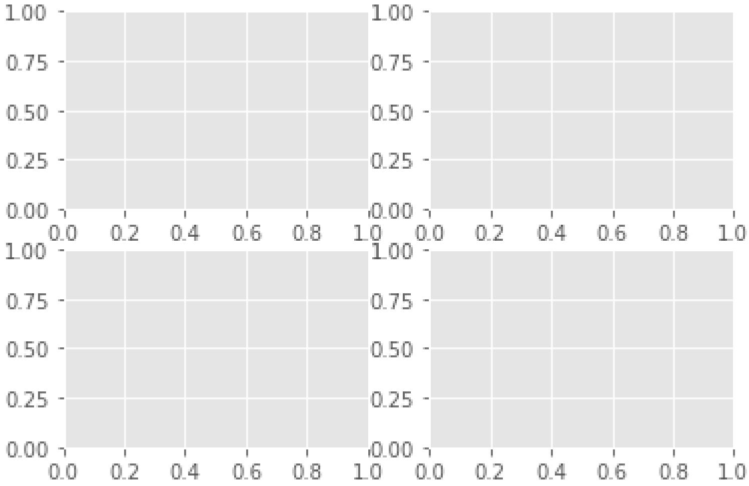
In [25]:
```
# we can plot 2 graphs

plt.subplots(nrows=2, ncols =1)
```

Out[25]:
```
(<Figure size 432x288 with 2 Axes>,
 array([<AxesSubplot:>, <AxesSubplot:>], dtype=object))
```



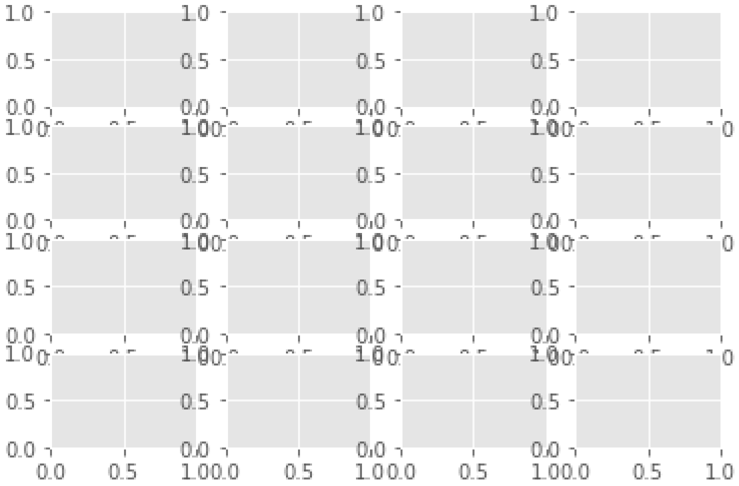
In [26]:
```
# we can want plot 4 graphs

plt.subplots(nrows=2, ncols =2)
```

Out[26]:
```
(<Figure size 432x288 with 4 Axes>,
 array([[<AxesSubplot:>, <AxesSubplot:>],
        [<AxesSubplot:>, <AxesSubplot:>]], dtype=object))
```



In [27]:
```
# we can want plot 16 graphs

plt.subplots(nrows=4, ncols =4)
```

Out[27]:
```
(<Figure size 432x288 with 16 Axes>,
 array([[<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
        [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
        [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
        [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]],
       dtype=object))
```

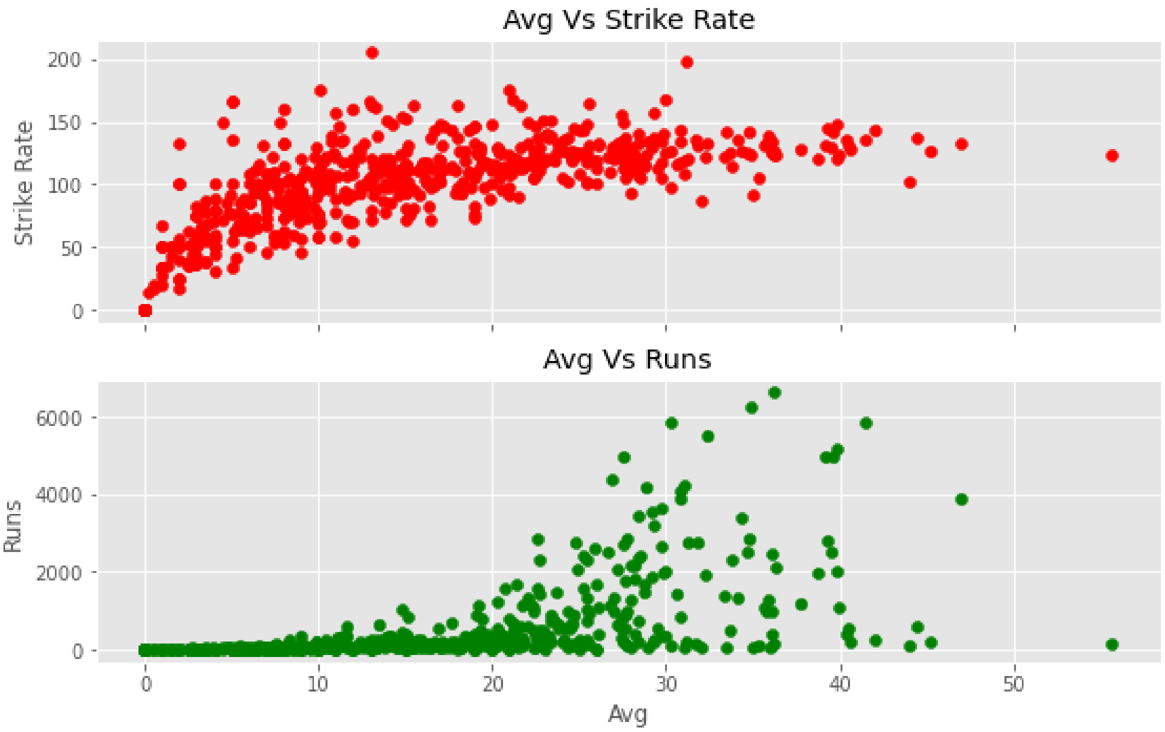


In [37]:

```
# on Data

fig, ax = plt.subplots(nrows=2,ncols=1,sharex=True,figsize=(10,6))

#sharex = Controls sharing of properties among x (*sharex*) or y (*sharey*)


# axis
ax[0].scatter(batter['avg'],batter['strike_rate'],color='red')
ax[1].scatter(batter['avg'],batter['runs'],color ='green')

ax[0].set_title('Avg Vs Strike Rate')
ax[0].set_ylabel('Strike Rate')


ax[1].set_title('Avg Vs Runs')
ax[1].set_ylabel('Runs')
ax[1].set_xlabel('Avg')
```

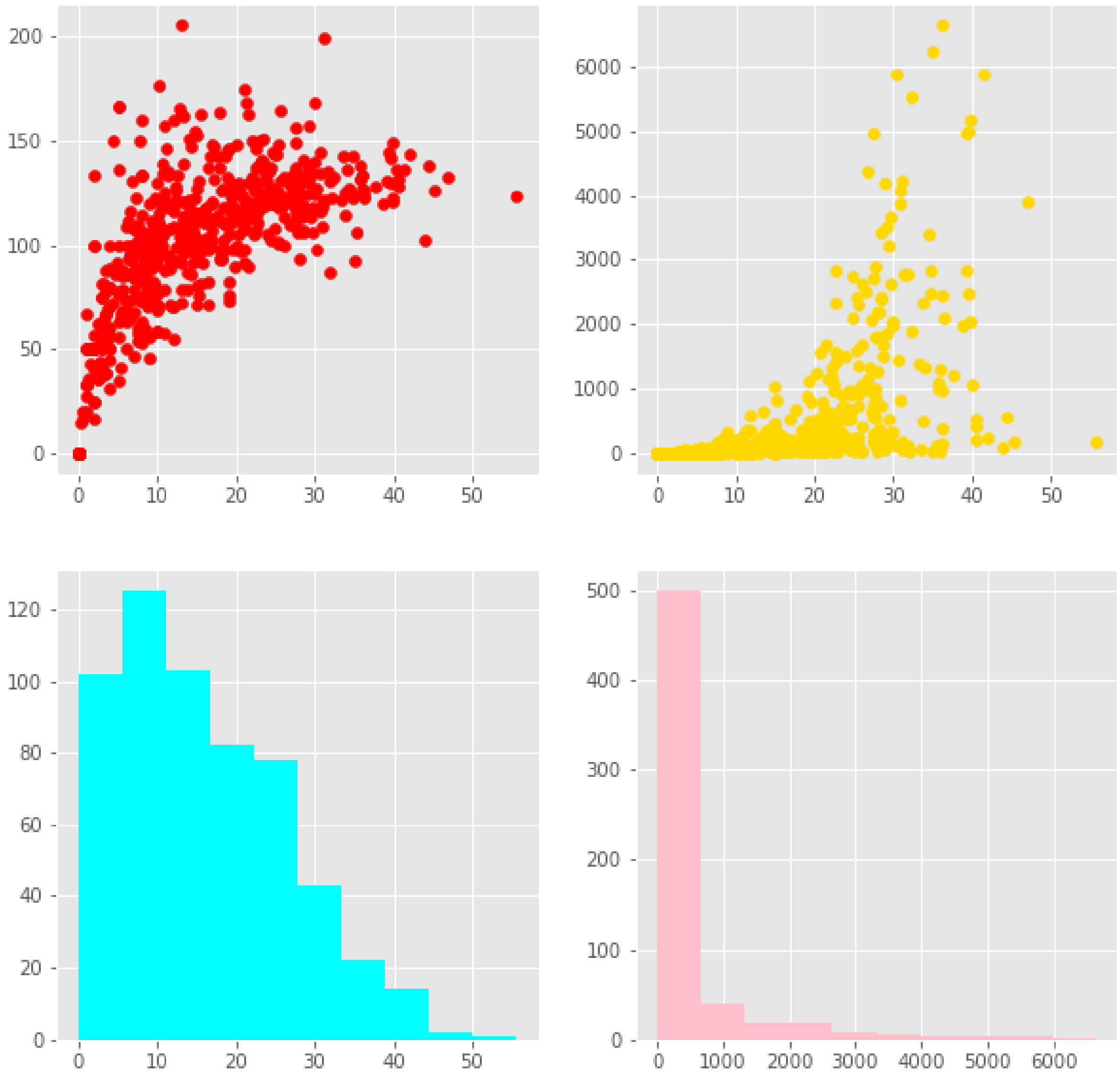
Out[37]:

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



In [29]:
```
fig, ax = plt.subplots(nrows=2,ncols=2,figsize=(10,10))

# its 2D array we have to metion (0,0)(0,1)(1,0)(1,1)

ax[0,0].scatter(batter['avg'],batter['strike_rate']
                ,color='red') # gives first set of axis

ax[0,1].scatter(batter['avg'],
                batter['runs']
               ,color='gold') # gives  second set of axis

ax[1,0].hist(batter['avg'],color='cyan') # gives third set of axis

ax[1,1].hist(batter['runs'],color='pink') # gives fourth set of axis

fig.show()
```

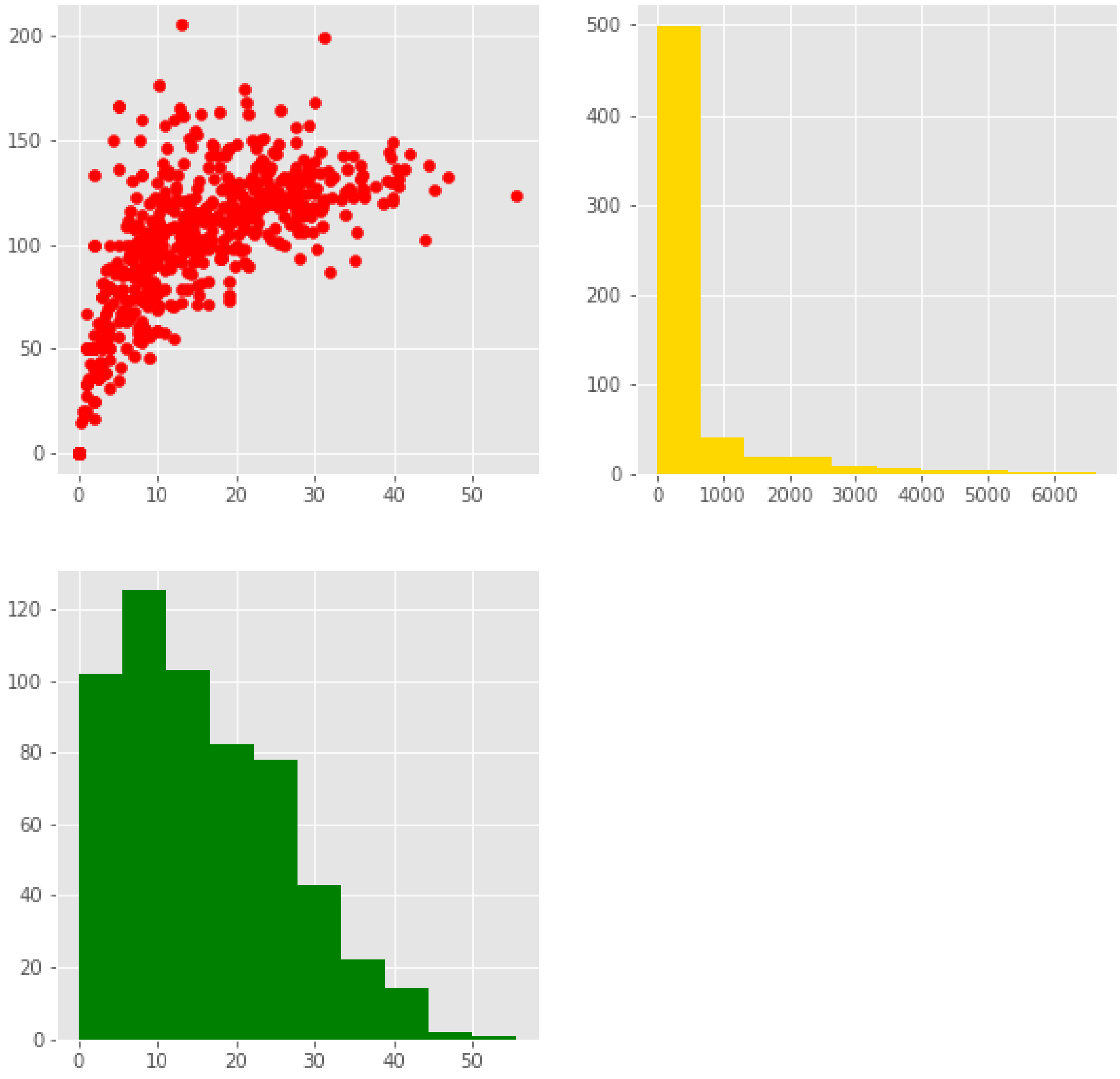
```
C:\Users\user\AppData\Local\Temp/ipykernel_14332/3464065883.py:16: UserWarnin
g: Matplotlib is currently using module://matplotlib_inline.backend_inline, w
hich is a non-GUI backend, so cannot show the figure.
  fig.show()
```



In [36]:

```
# Manual way of adding subplots

fig = plt.figure(figsize=(10,10))

ax1 = fig.add_subplot(2,2,1) #2 rows , 2 column , 1 graph
ax1.scatter(batter['avg'],batter['strike_rate'],color='red')

ax2 = fig.add_subplot(2,2,2) #2 rows , 2 column , 2 graph
ax2.hist(batter['runs'],color='gold')

ax3 = fig.add_subplot(2,2,3) #2 rows , 2 column , 3 graph
ax3.hist(batter['avg'],color ='green')

fig.show()
```

```
C:\Users\user\AppData\Local\Temp/ipykernel_14332/2368418411.py:14: UserWarnin
g: Matplotlib is currently using module://matplotlib_inline.backend_inline, w
hich is a non-GUI backend, so cannot show the figure.
  fig.show()
```



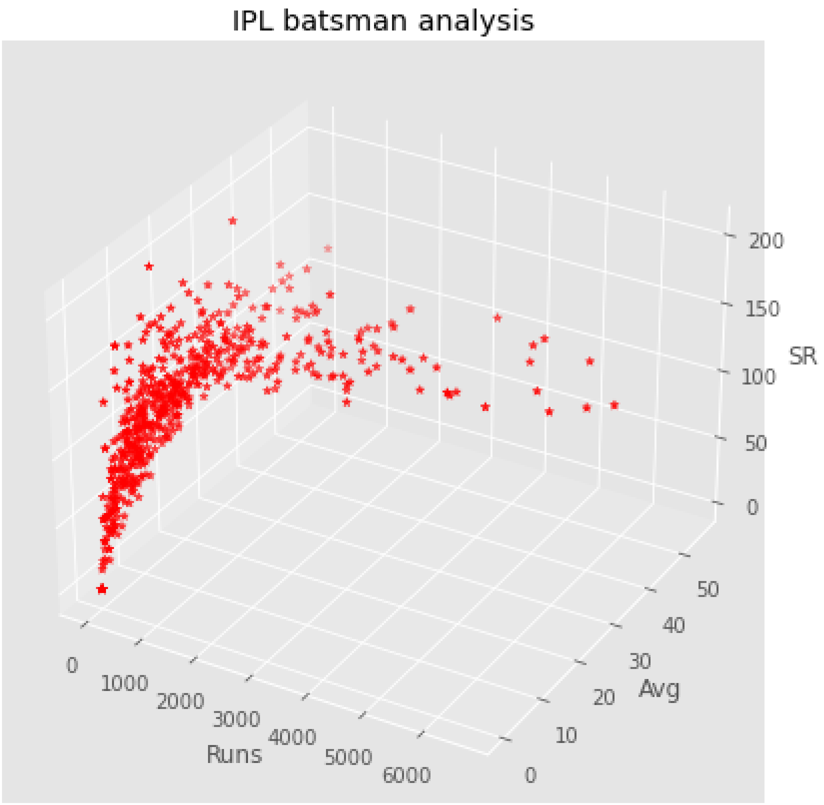
3D scatter plots

In [62]:

```
batter

fig = plt.figure(figsize=(10,7))

ax = plt.subplot(projection ='3d')

ax.scatter3D(batter['runs'],batter['avg'],batter['strike_rate'],
             color='red' , marker = '*')

ax.set_title('IPL batsman analysis')

ax.set_xlabel('Runs')
ax.set_ylabel('Avg')
ax.set_zlabel('SR')
```

Out[62]:

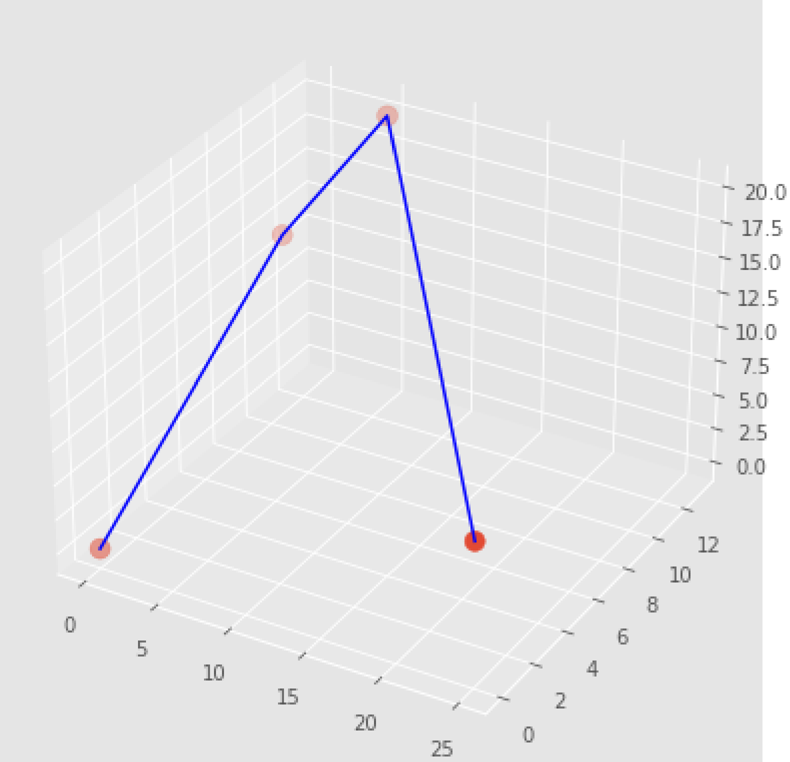
```
Text(0.5, 0, 'SR')
```



3D Line plot

In [61]:
```
x = [0,1,5,25]
y = [0,10,13,0]
z = [0,13,20,9]

fig = plt.figure(figsize=(10,7))

ax = plt.subplot(projection='3d')

ax.scatter3D(x,y,z,s=[100,100,100,100])

ax.plot3D(x,y,z,color='blue')
```

Out[61]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x25b48431250>]
```



3D surface Plot

In [63]:
```
# Loss function x2 + y2

# Helpful in Machine Learning

x = np.linspace(-10,10,100)
y = np.linspace(-10,10,100)
```

In [64]:
```
x
```

Out[64]:
```
array([-10.        ,  -9.7979798 ,  -9.5959596 ,  -9.39393939,
        -9.19191919,  -8.98989899,  -8.78787879,  -8.58585859,
        -8.38383838,  -8.18181818,  -7.97979798,  -7.77777778,
        -7.57575758,  -7.37373737,  -7.17171717,  -6.96969697,
        -6.76767677,  -6.56565657,  -6.36363636,  -6.16161616,
        -5.95959596,  -5.75757576,  -5.55555556,  -5.35353535,
        -5.15151515,  -4.94949495,  -4.74747475,  -4.54545455,
        -4.34343434,  -4.14141414,  -3.93939394,  -3.73737374,
        -3.53535354,  -3.33333333,  -3.13131313,  -2.92929293,
        -2.72727273,  -2.52525253,  -2.32323232,  -2.12121212,
        -1.91919192,  -1.71717172,  -1.51515152,  -1.31313131,
        -1.11111111,  -0.90909091,  -0.70707071,  -0.50505051,
        -0.3030303 ,  -0.1010101 ,   0.1010101 ,   0.3030303 ,
         0.50505051,   0.70707071,   0.90909091,   1.11111111,
         1.31313131,   1.51515152,   1.71717172,   1.91919192,
         2.12121212,   2.32323232,   2.52525253,   2.72727273,
         2.92929293,   3.13131313,   3.33333333,   3.53535354,
         3.73737374,   3.93939394,   4.14141414,   4.34343434,
         4.54545455,   4.74747475,   4.94949495,   5.15151515,
         5.35353535,   5.55555556,   5.75757576,   5.95959596,
         6.16161616,   6.36363636,   6.56565657,   6.76767677,
         6.96969697,   7.17171717,   7.37373737,   7.57575758,
         7.77777778,   7.97979798,   8.18181818,   8.38383838,
         8.58585859,   8.78787879,   8.98989899,   9.19191919,
         9.39393939,   9.5959596 ,   9.7979798 ,  10.        ])
```

In [65]:
```
y
```

Out[65]:
```
array([-10.        ,  -9.7979798 ,  -9.5959596 ,  -9.39393939,
        -9.19191919,  -8.98989899,  -8.78787879,  -8.58585859,
        -8.38383838,  -8.18181818,  -7.97979798,  -7.77777778,
        -7.57575758,  -7.37373737,  -7.17171717,  -6.96969697,
        -6.76767677,  -6.56565657,  -6.36363636,  -6.16161616,
        -5.95959596,  -5.75757576,  -5.55555556,  -5.35353535,
        -5.15151515,  -4.94949495,  -4.74747475,  -4.54545455,
        -4.34343434,  -4.14141414,  -3.93939394,  -3.73737374,
        -3.53535354,  -3.33333333,  -3.13131313,  -2.92929293,
        -2.72727273,  -2.52525253,  -2.32323232,  -2.12121212,
        -1.91919192,  -1.71717172,  -1.51515152,  -1.31313131,
        -1.11111111,  -0.90909091,  -0.70707071,  -0.50505051,
        -0.3030303 ,  -0.1010101 ,   0.1010101 ,   0.3030303 ,
         0.50505051,   0.70707071,   0.90909091,   1.11111111,
         1.31313131,   1.51515152,   1.71717172,   1.91919192,
         2.12121212,   2.32323232,   2.52525253,   2.72727273,
         2.92929293,   3.13131313,   3.33333333,   3.53535354,
         3.73737374,   3.93939394,   4.14141414,   4.34343434,
         4.54545455,   4.74747475,   4.94949495,   5.15151515,
         5.35353535,   5.55555556,   5.75757576,   5.95959596,
         6.16161616,   6.36363636,   6.56565657,   6.76767677,
         6.96969697,   7.17171717,   7.37373737,   7.57575758,
         7.77777778,   7.97979798,   8.18181818,   8.38383838,
         8.58585859,   8.78787879,   8.98989899,   9.19191919,
         9.39393939,   9.5959596 ,   9.7979798 ,  10.        ])
```

In [66]:
```
np.meshgrid(x,y)
```

Out[66]:
```
[array([[-10.        ,  -9.7979798,  -9.5959596, ...,   9.5959596,
           9.7979798,  10.        ],
        [-10.        ,  -9.7979798,  -9.5959596, ...,   9.5959596,
           9.7979798,  10.        ],
        [-10.        ,  -9.7979798,  -9.5959596, ...,   9.5959596,
           9.7979798,  10.        ],
        ...,
        [-10.        ,  -9.7979798,  -9.5959596, ...,   9.5959596,
           9.7979798,  10.        ],
        [-10.        ,  -9.7979798,  -9.5959596, ...,   9.5959596,
           9.7979798,  10.        ],
        [-10.        ,  -9.7979798,  -9.5959596, ...,   9.5959596,
           9.7979798,  10.        ]]),
 array([[-10.        , -10.        , -10.        , ..., -10.        ,
         -10.        , -10.        ],
        [ -9.7979798,  -9.7979798,  -9.7979798, ...,  -9.7979798,
          -9.7979798,  -9.7979798],
        [ -9.5959596,  -9.5959596,  -9.5959596, ...,  -9.5959596,
          -9.5959596,  -9.5959596],
        ...,
        [  9.5959596,   9.5959596,   9.5959596, ...,   9.5959596,
           9.5959596,   9.5959596],
        [  9.7979798,   9.7979798,   9.7979798, ...,   9.7979798,
           9.7979798,   9.7979798],
        [ 10.        ,  10.        ,  10.        , ...,  10.        ,
          10.        ,  10.        ]])]
```

In [67]:
```
# Mesh grid

xx,yy = np.meshgrid(x,y)
```

In [68]:
```
xx.shape
```

Out[68]:
```
(100, 100)
```

In [69]:
```
yy.shape
```

Out[69]:
```
(100, 100)
```

In [83]:
```
z = xx**2 + yy**2
```

In [71]:
```
z.shape
```

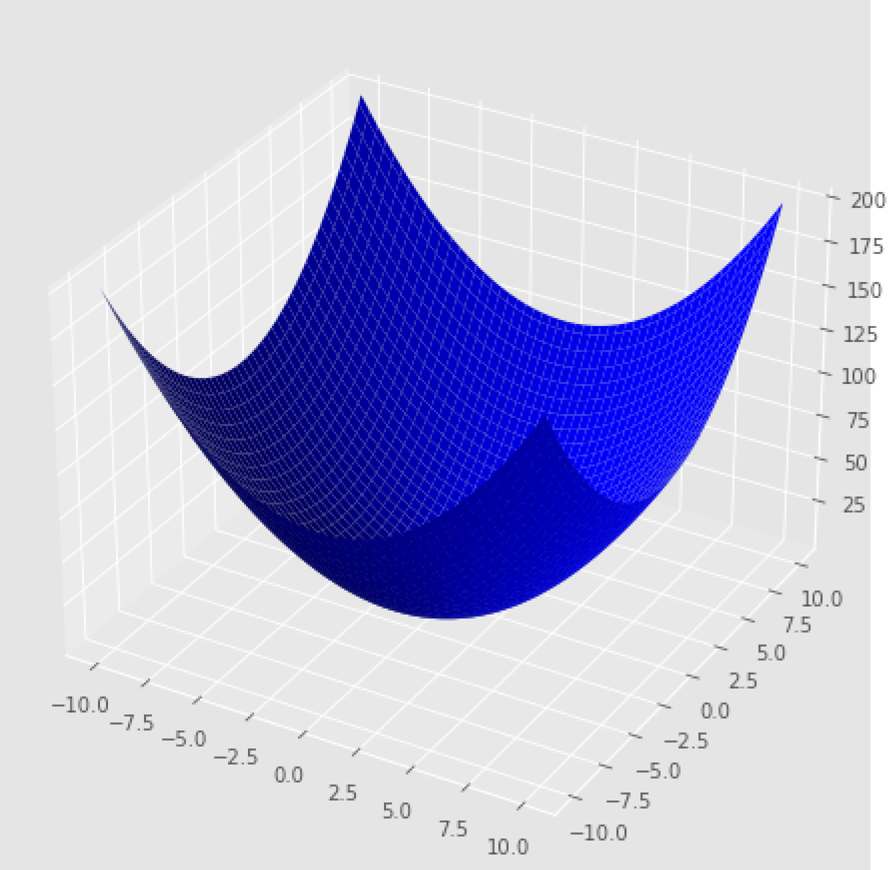
Out[71]:
```
(100, 100)
```

In [76]:

```
# 3D Surface Plot

fig =plt.figure(figsize =(12,8))

ax = plt.subplot(projection = '3d')

ax.plot_surface(xx,yy,z, color ='blue')
```

Out[76]:

```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x25b4c7d0c40>
```

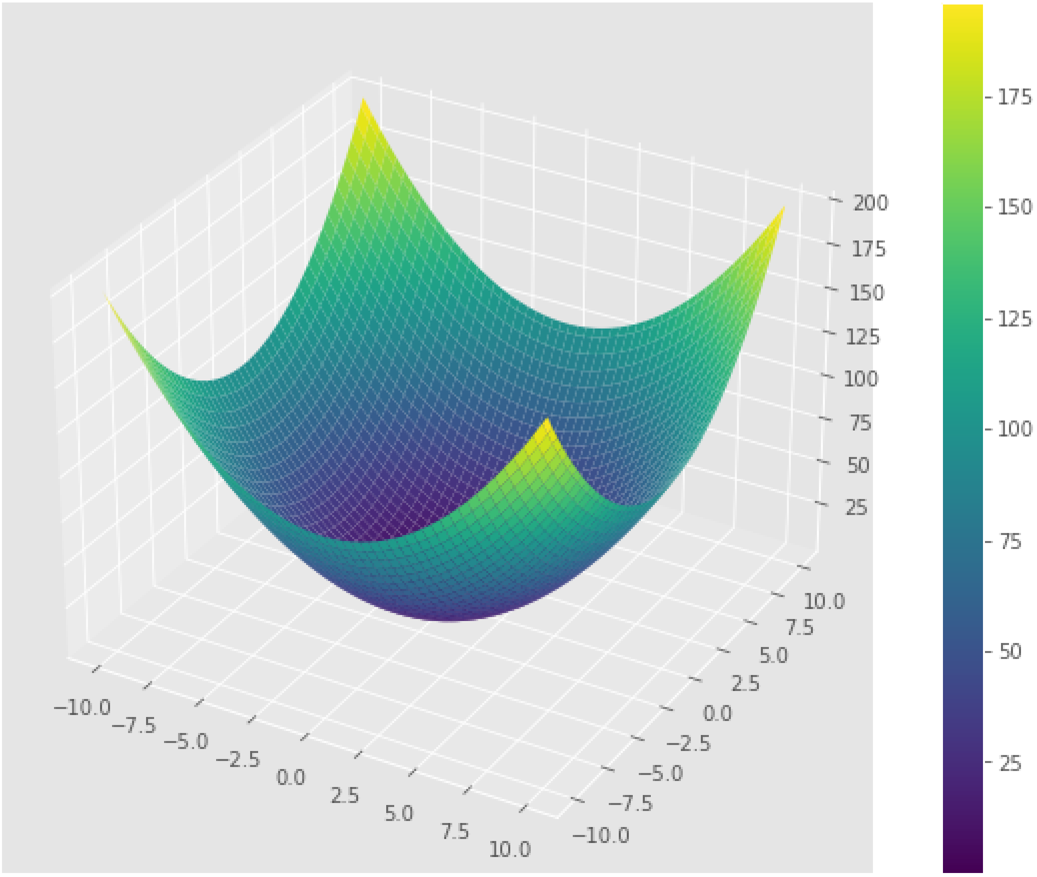


In [74]:

```
# Adding theme

fig = plt.figure(figsize=(12,8))

ax = plt.subplot(projection='3d')

p = ax.plot_surface(xx,yy,z,cmap='viridis')
fig.colorbar(p)
```

Out[74]:

```
<matplotlib.colorbar.Colorbar at 0x25b48466d90>
```

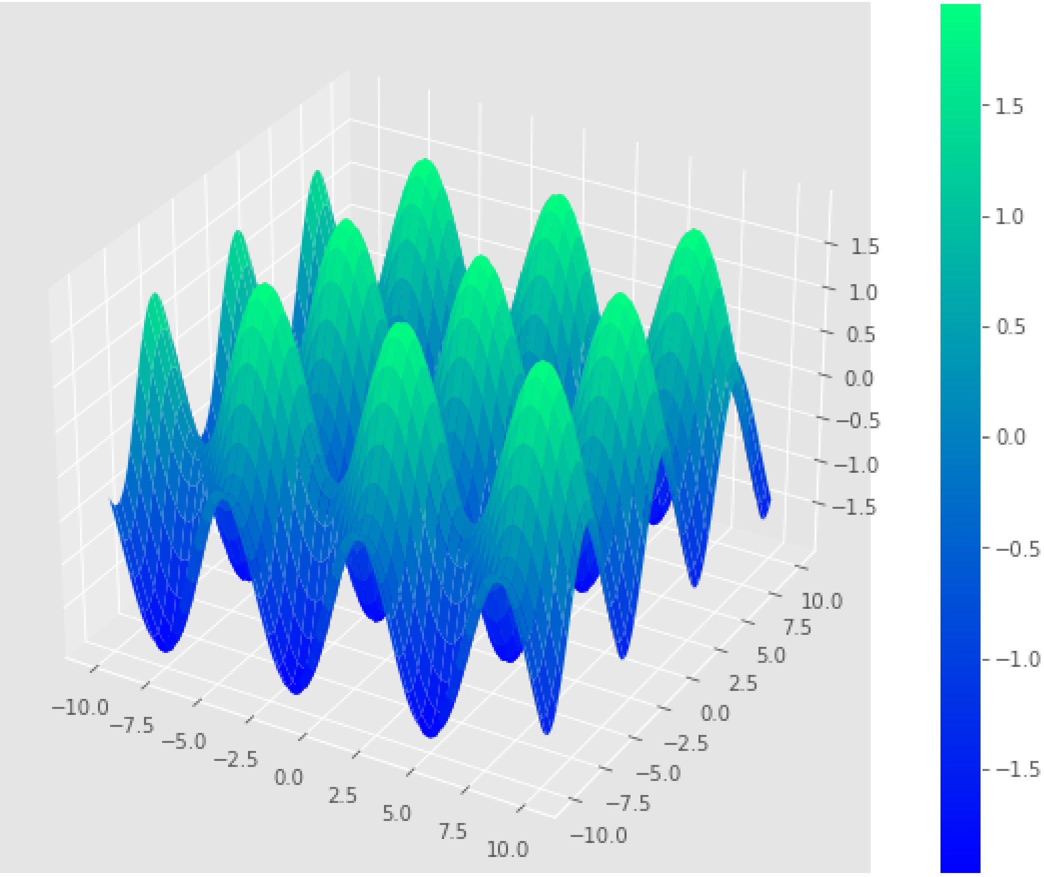


In [79]:

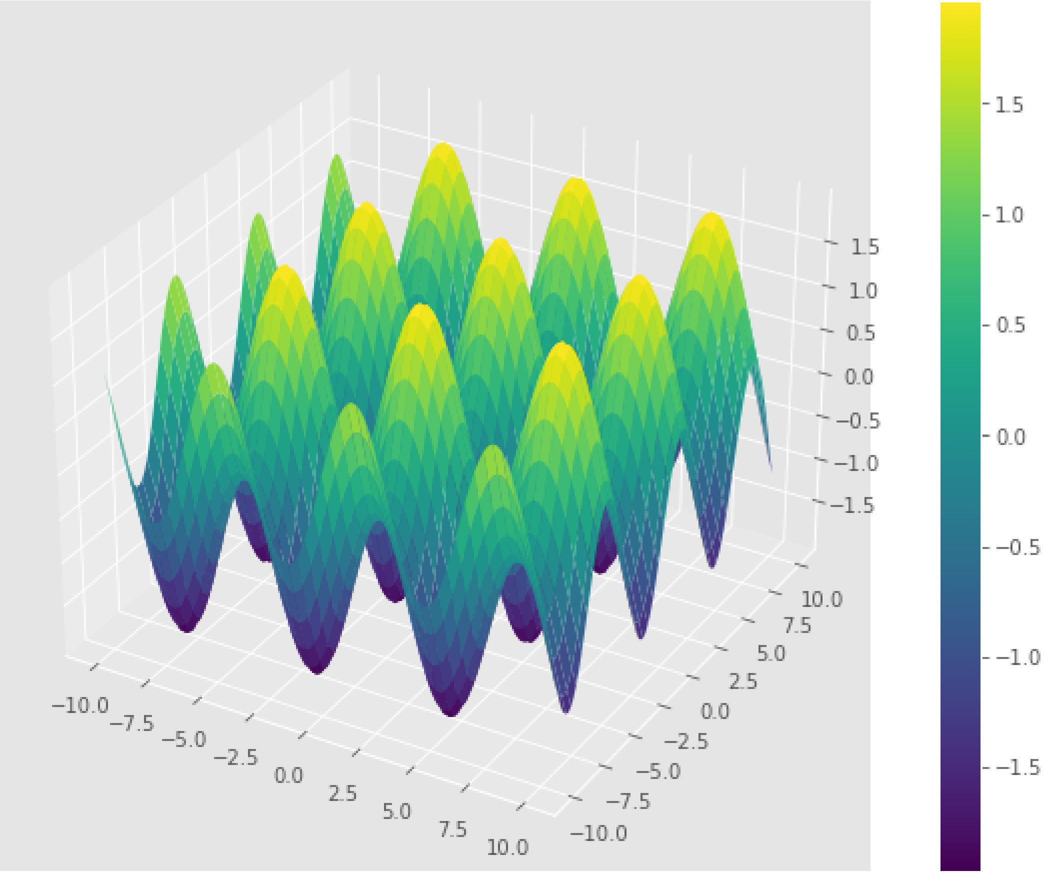
```
z = np.sin(xx) + np.cos(yy)

fig = plt.figure(figsize=(12,8))

ax = plt.subplot(projection='3d')

p = ax.plot_surface(xx,yy,z,cmap='winter')

fig.colorbar(p)
```

Out[79]:

```
<matplotlib.colorbar.Colorbar at 0x25b4e03a220>
```



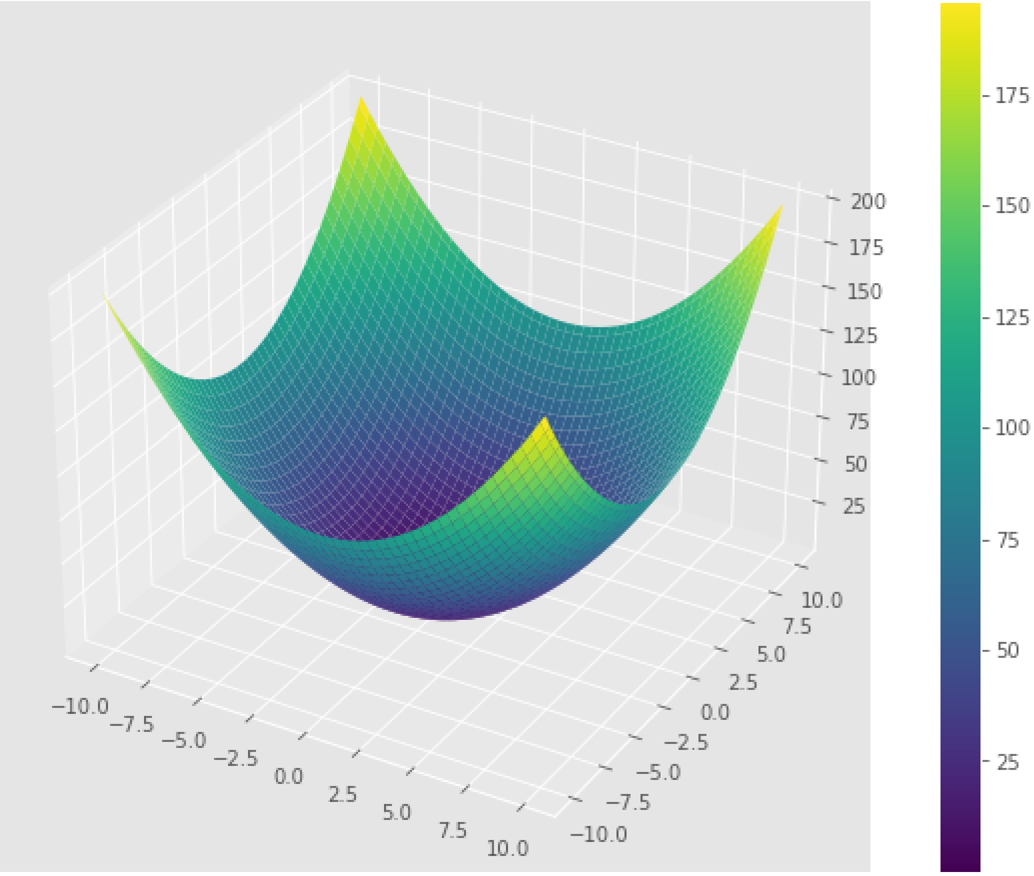
```
In [82]: z = np.sin(xx) + np.sin(yy)

         fig = plt.figure(figsize=(12,8))

         ax = plt.subplot(projection='3d')

         p = ax.plot_surface(xx,yy,z,cmap='viridis')
         fig.colorbar(p)

Out[82]: <matplotlib.colorbar.Colorbar at 0x25b4e877a30>
```



Contour Plots

In [84]:
```
# Representing 3D to 2d

#3D graph
fig = plt.figure(figsize=(12,8))

ax = plt.subplot(projection='3d')

p = ax.plot_surface(xx,yy,z,cmap='viridis')
fig.colorbar(p)
```

Out[84]:
```
<matplotlib.colorbar.Colorbar at 0x25b4ebe94f0>
```

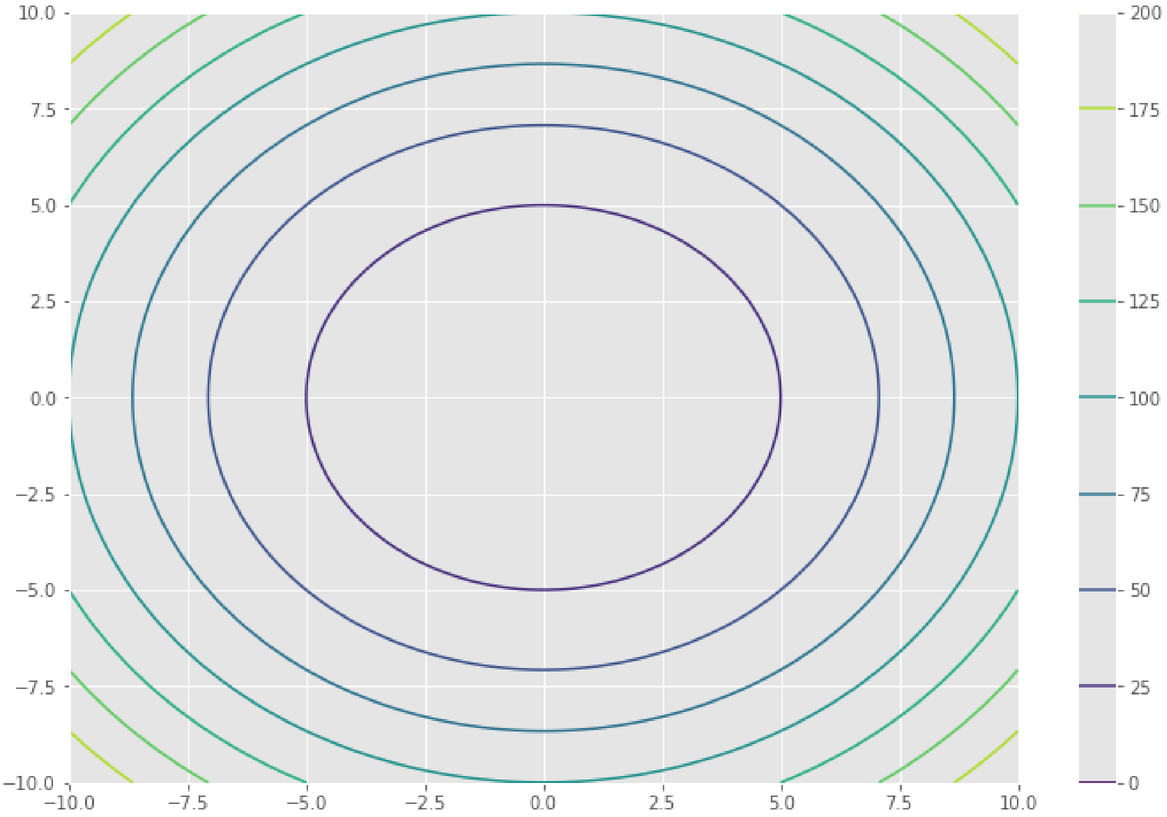


In [86]:

```
# Contour Plot

fig = plt.figure(figsize=(12,8))

ax = plt.subplot()

p = ax.contour(xx,yy,z,cmap='viridis')
fig.colorbar(p)
```

Out[86]:

```
<matplotlib.colorbar.Colorbar at 0x25b4fe9bc40>
```



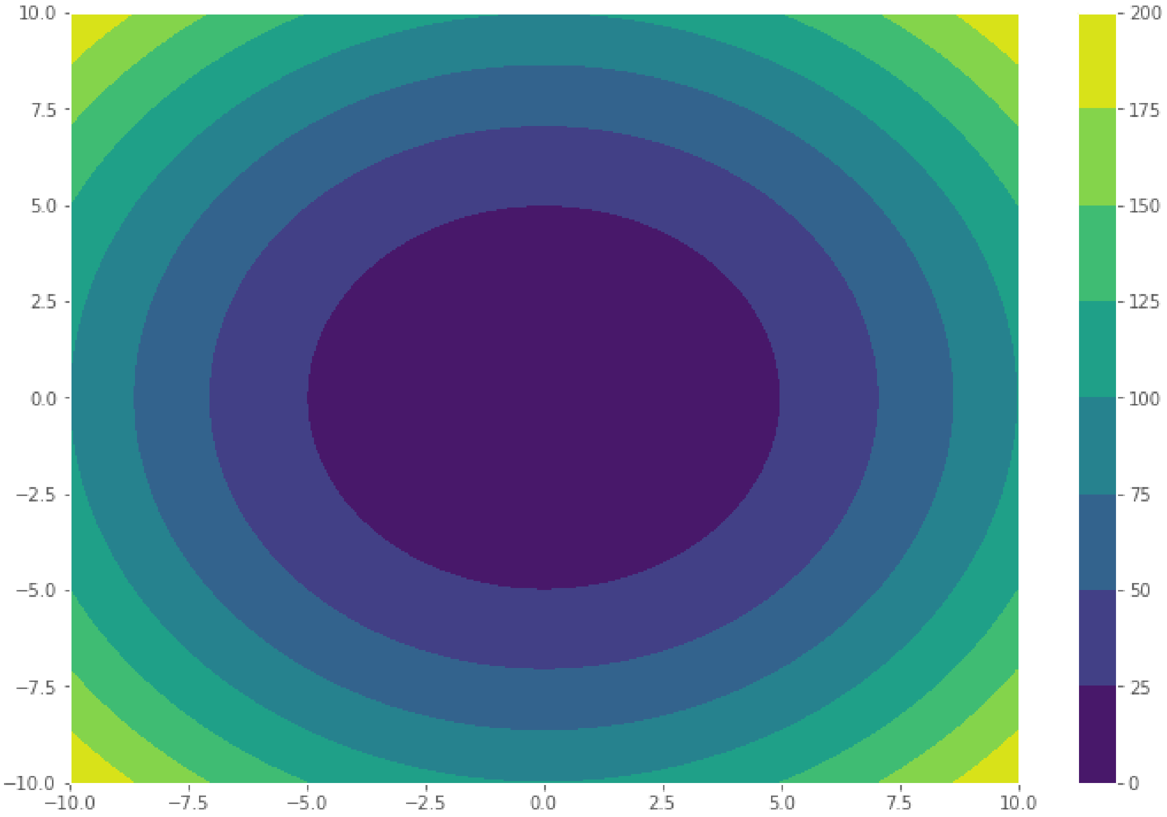
Contourf Plot

In [87]:

```
fig = plt.figure(figsize=(12,8))

ax = plt.subplot()

p = ax.contourf(xx,yy,z,cmap='viridis')
fig.colorbar(p)

# Here blue colour is bend/shallow and yellow is up/top in 3D
```

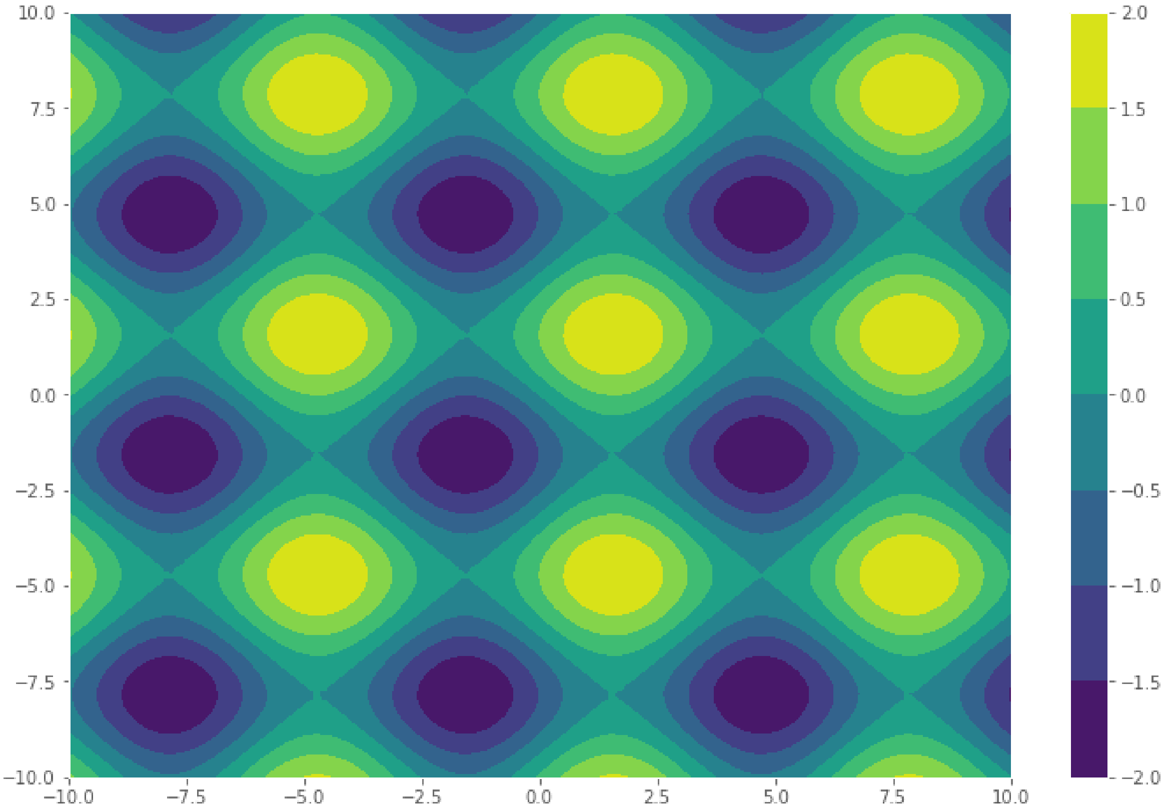
Out[87]:

```
<matplotlib.colorbar.Colorbar at 0x25b502ccb80>
```



In [88]:
```
z = np.sin(xx) + np.sin(yy)

fig = plt.figure(figsize=(12,8))

ax = plt.subplot()

p = ax.contourf(xx,yy,z,cmap='viridis')
fig.colorbar(p)
```

Out[88]:
```
<matplotlib.colorbar.Colorbar at 0x25b506709a0>
```



Heat map

In [90]:
```
delivery = pd.read_csv("IPL_Ball_by_Ball_2008_2022.csv")
```

In [91]:
```
delivery.head()
```

Out[91]:

	ID	innings	overs	ballnumber	batter	bowler	non-striker	extra_type	batsman_run	e
0	1312200	1	0	1	YBK Jaiswal	Mohammed Shami	JC Buttler	NaN	0	
1	1312200	1	0	2	YBK Jaiswal	Mohammed Shami	JC Buttler	legbyes	0	
2	1312200	1	0	3	JC Buttler	Mohammed Shami	YBK Jaiswal	NaN	1	
3	1312200	1	0	4	YBK Jaiswal	Mohammed Shami	JC Buttler	NaN	0	
4	1312200	1	0	5	YBK Jaiswal	Mohammed Shami	JC Buttler	NaN	0	

In [92]:
```
temp_df = delivery[(delivery['ballnumber'].isin([1,2,3,4,5,6]))
                   & (delivery['batsman_run']==6)]
```

In [94]:
```
temp_df.sample()
```

Out[94]:

	ID	innings	overs	ballnumber	batter	bowler	non-striker	extra_type	batsman_run	ex
134037	598062	1	15	3	MS Dhoni	A Nehra	RA Jadeja	NaN	6	

In [95]:
```
grid = temp_df.pivot_table(index='overs',columns='ballnumber',
                           values='batsman_run',aggfunc='count')
```

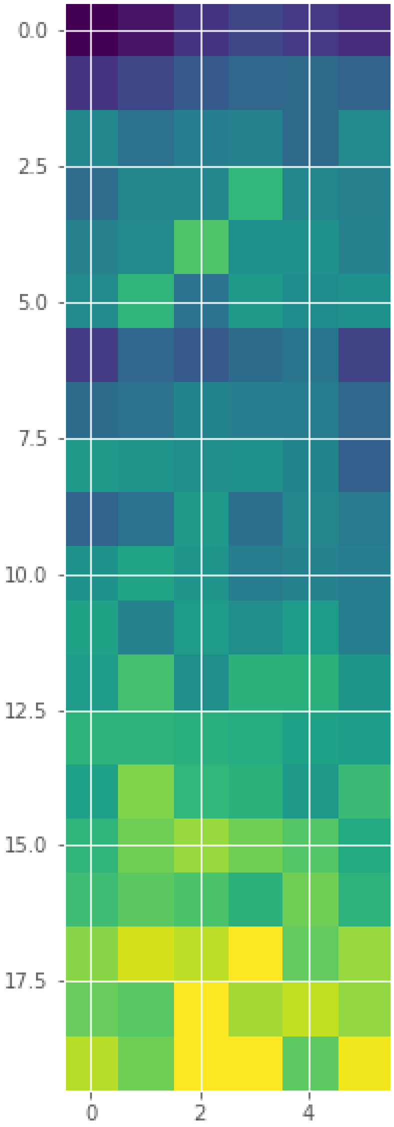
In [96]:
```
grid
```

Out[96]:

ballnumber	1	2	3	4	5	6
overs						
0	9	17	31	39	33	27
1	31	40	49	56	58	54
2	75	62	70	72	58	76
3	60	74	74	103	74	71
4	71	76	112	80	81	72
5	77	102	63	86	78	80
6	34	56	49	59	64	38
7	59	62	73	70	69	56
8	86	83	79	81	73	52
9	54	62	86	61	74	67
10	82	92	83	69	72	70
11	91	72	87	79	87	70
12	87	109	79	100	100	84
13	101	101	99	97	90	88
14	90	124	103	100	86	106
15	102	120	129	121	113	96
16	107	115	111	100	120	101
17	126	142	137	151	117	129
18	118	114	151	132	138	128
19	136	120	151	151	116	148

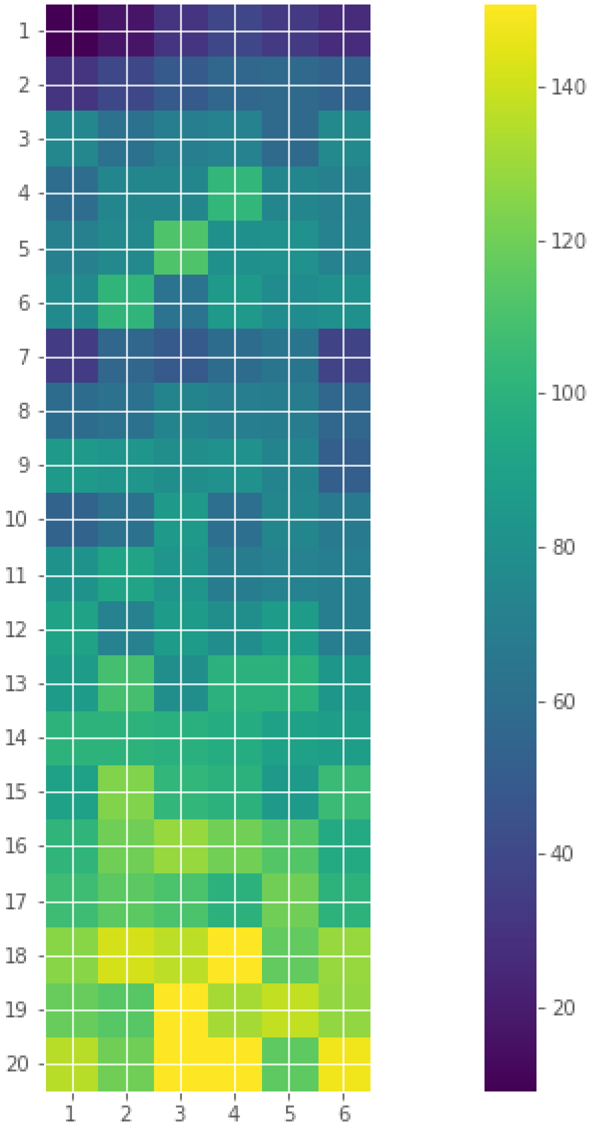
In [99]:
```
plt.figure(figsize=(20,10))
plt.imshow(grid)
```

Out[99]:
```
<matplotlib.image.AxesImage at 0x25b506e4760>
```



In [100]:
```
plt.figure(figsize=(20,10))
plt.imshow(grid)

plt.yticks(delivery['overs'].unique(), list(range(1,21)))

plt.xticks(np.arange(0,6), list(range(1,7)))

plt.colorbar()
```

Out[100]:
```
<matplotlib.colorbar.Colorbar at 0x25b50c9e100>
```



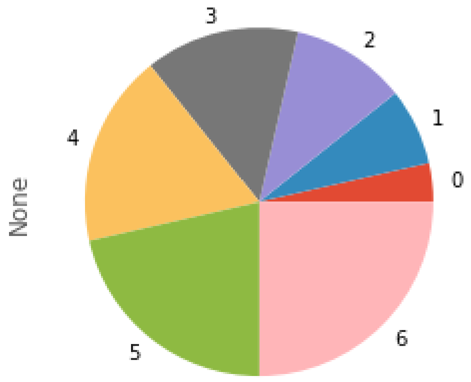
Pandas Plot

In [101]:

```
# on a series

s = pd.Series([1,2,3,4,5,6,7])
s.plot(kind='pie')
```

Out[101]:

```
<AxesSubplot:ylabel='None'>
```



In [106]:

```
# can be used on a dataframe as well

import seaborn as sns
tips = sns.load_dataset('tips')
tips.head()
```

Out[106]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [108]:

```
tips['size'] = tips['size'] * 100
```

In [109]:

```
tips.head()
```

Out[109]:

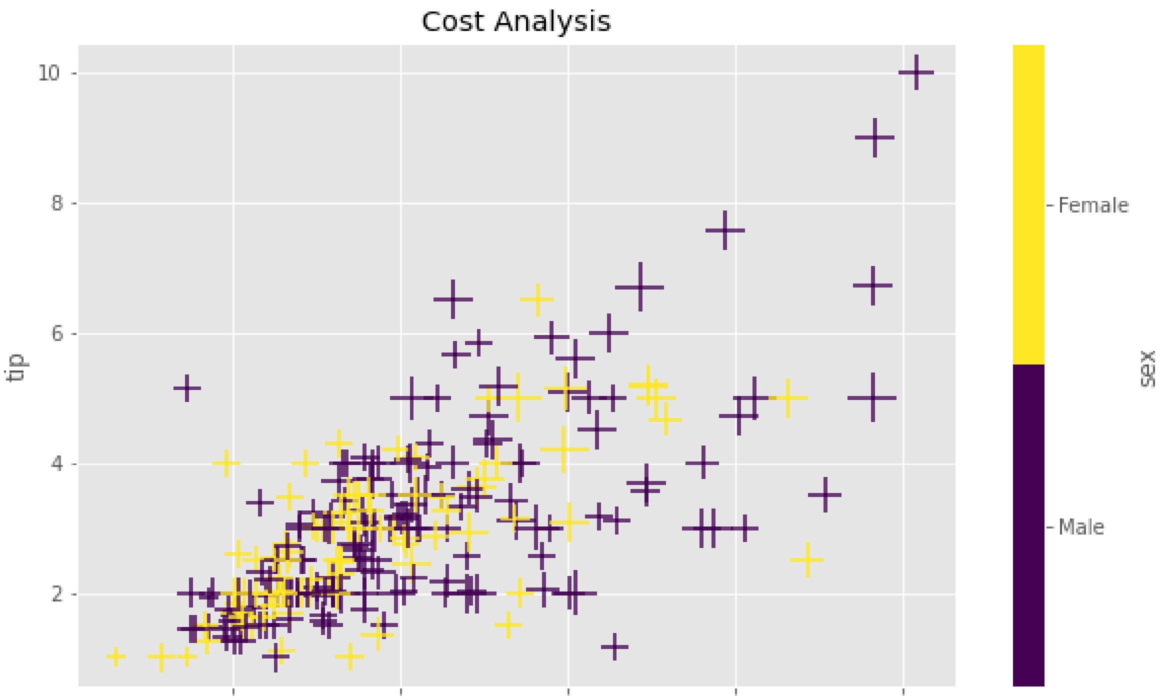
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	200
1	10.34	1.66	Male	No	Sun	Dinner	300
2	21.01	3.50	Male	No	Sun	Dinner	300
3	23.68	3.31	Male	No	Sun	Dinner	200
4	24.59	3.61	Female	No	Sun	Dinner	400

In [110]:

```
# Scatter plot -> labels -> markers -> figsize -> color -> cmap

tips.plot(kind='scatter',x='total_bill',y='tip',
          title='Cost Analysis',marker='+',
          figsize=(10,6),s='size',c='sex',
          cmap='viridis')
```

Out[110]:

```
<AxesSubplot:title={'center':'Cost Analysis'}, xlabel='total_bill', ylabel='t
ip'>
```



2d plot

In [111]:

```

# dataset = 'https://raw.githubusercontent.com/m-mehdi/pandas_tutorials/main/w

stocks = pd.read_csv('https://raw.githubusercontent.com/m-mehdi/pandas_tutoria
stocks.head()
```

Out[111]:

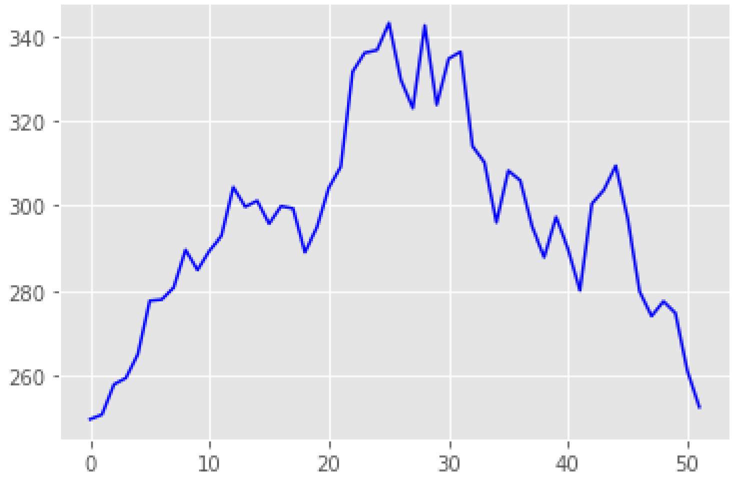
	Date	MSFT	FB	AAPL
0	2021-05-24	249.679993	328.730011	124.610001
1	2021-05-31	250.789993	330.350006	125.889999
2	2021-06-07	257.890015	331.260010	127.349998
3	2021-06-14	259.429993	329.660004	130.460007
4	2021-06-21	265.019989	341.369995	133.110001

In [118]:

```
# line plot

stocks['MSFT'].plot(kind='line', color ='blue')
```

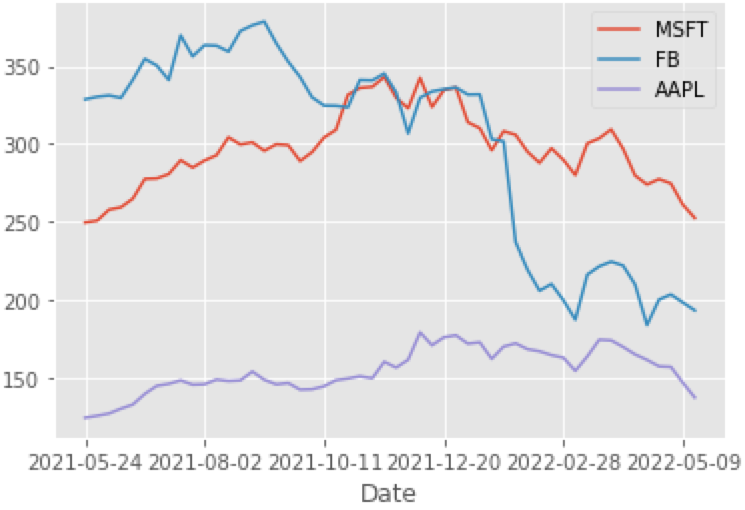
Out[118]:

```
<AxesSubplot:>
```



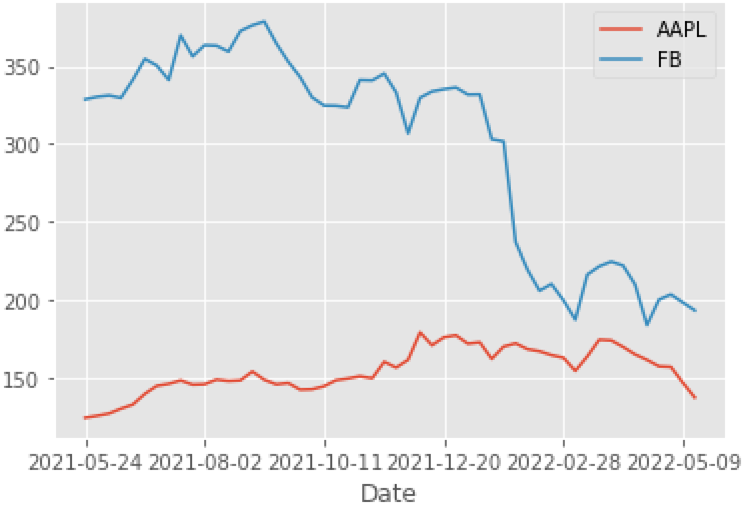
In [113]:
```
stocks.plot(kind='line',x='Date')
```

Out[113]:
```
<AxesSubplot:xlabel='Date'>
```



In [114]:
```
stocks[['Date','AAPL','FB']].plot(kind='line',x='Date')
```

Out[114]:
```
<AxesSubplot:xlabel='Date'>
```



bar chart

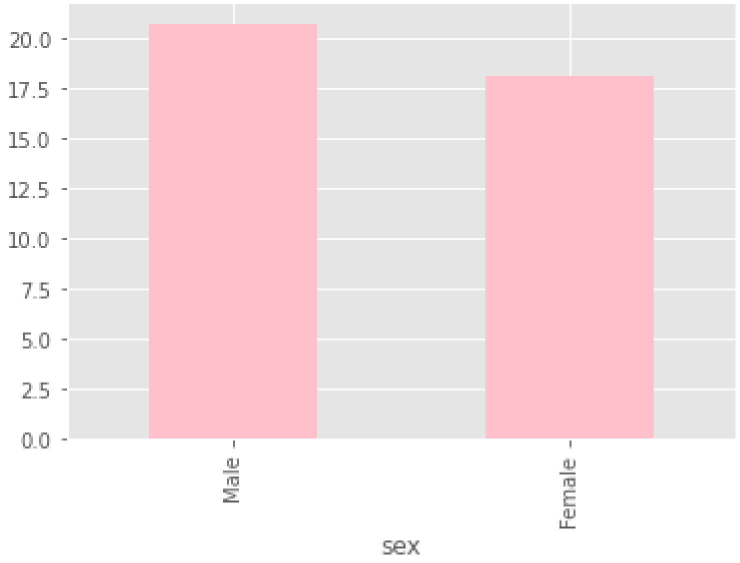
In [115]:
```
# bar chart -> single -> horizontal -> multiple
# using tips
temp = pd.read_csv("batsman_season_record.csv")
temp.head()
```

Out[115]:

	batsman	2015	2016	2017
0	AB de Villiers	513	687	216
1	DA Warner	562	848	641
2	MS Dhoni	372	284	290
3	RG Sharma	482	489	333
4	V Kohli	505	973	308

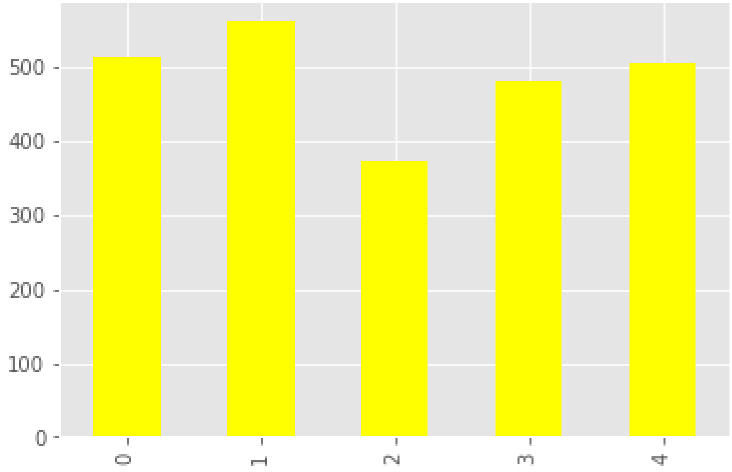
In [119]:
```
tips.groupby('sex')['total_bill'].mean().plot(kind='bar', color ='pink')
```

Out[119]:
```
<AxesSubplot:xlabel='sex'>
```



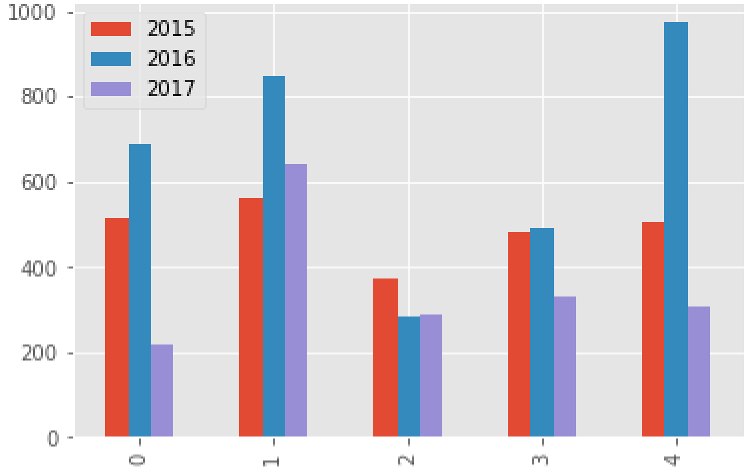
```
In [117]: temp['2015'].plot(kind='bar',color ='yellow')
```

```
Out[117]: <AxesSubplot:>
```



```
In [120]: temp.plot(kind='bar')
```

```
Out[120]: <AxesSubplot:>
```

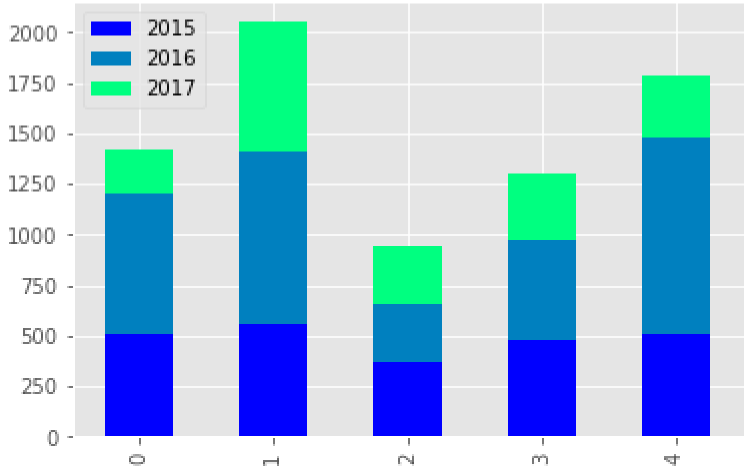


In [126]:

```
# stacked bar chart

temp.plot(kind='bar',stacked=True , colormap ='winter')
```

Out[126]:

```
<AxesSubplot:>
```



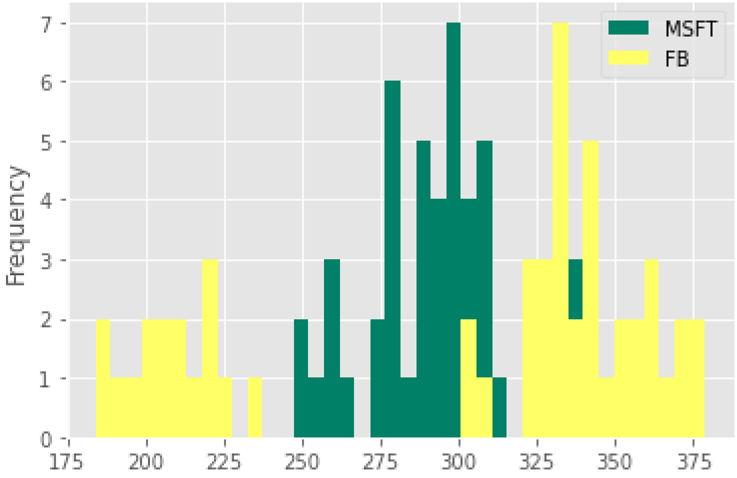
Histogram

In [125]:

```
# using stocks

stocks[['MSFT','FB']].plot(kind='hist',bins=40 , colormap ='summer')
```

Out[125]:

```
<AxesSubplot:ylabel='Frequency'>
```



Pie chart

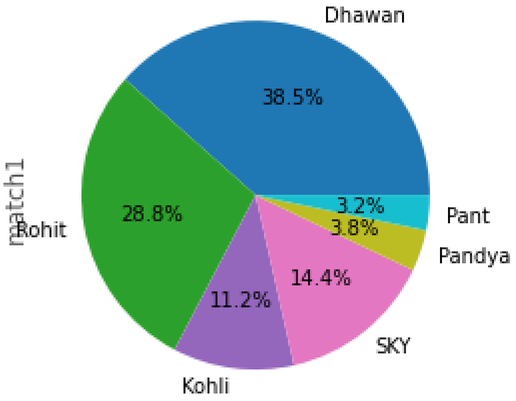
In [127]:
```
# single and multiple

df = pd.DataFrame(
    {
        'batsman':['Dhawan','Rohit','Kohli','SKY','Pandya','Pant'],
        'match1':[120,90,35,45,12,10],
        'match2':[0,1,123,130,34,45],
        'match3':[50,24,145,45,10,90]
    }
)

df.head()
```

Out[127]:

	batsman	match1	match2	match3
0	Dhawan	120	0	50
1	Rohit	90	1	24
2	Kohli	35	123	145
3	SKY	45	130	45
4	Pandya	12	34	10

In [130]:
```
df['match1'].plot(kind='pie',
                  labels=df['batsman'].values,autopct='%0.1f%%',
                  colormap= 'tab10')
```

Out[130]:
```
<AxesSubplot:ylabel='match1'>
```

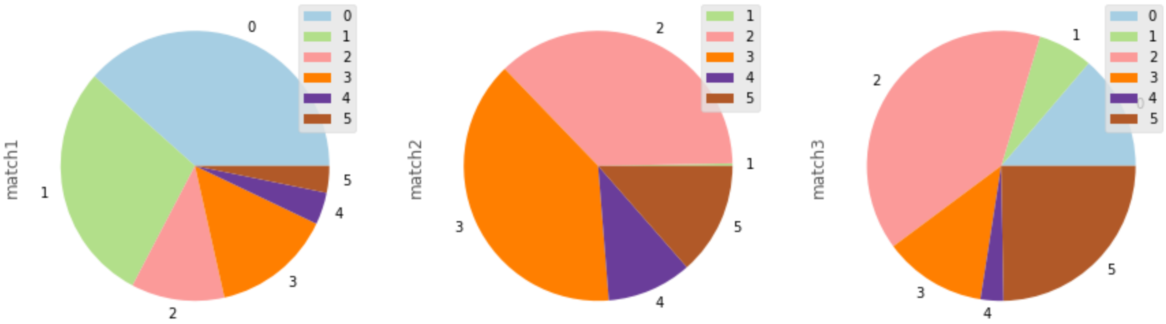


In [132]:

```
# multiple pie charts

df[['match1','match2','match3']].plot(kind='pie',
                                      subplots=True,
                                      figsize=(15,8),
                                     colormap='Paired')
```

Out[132]:

```
array([<AxesSubplot:ylabel='match1'>, <AxesSubplot:ylabel='match2'>,
       <AxesSubplot:ylabel='match3'>], dtype=object)
```



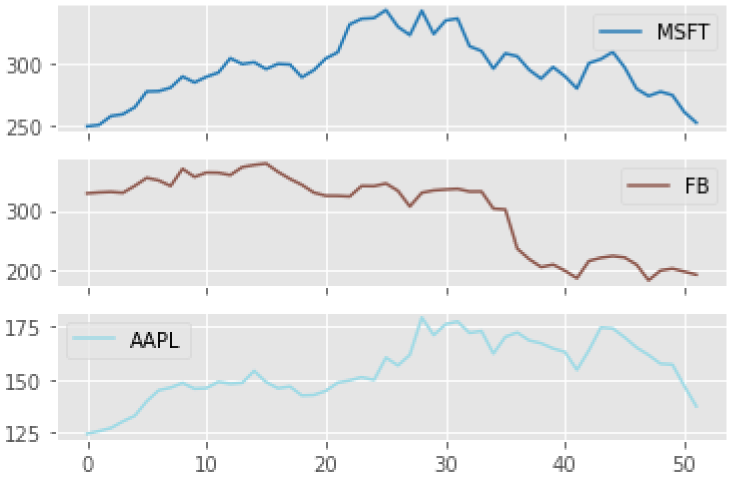
multiple separate graphs together

In [138]:

```
# using stocks

stocks.plot(kind='line',subplots=True , colormap ='tab20')
```

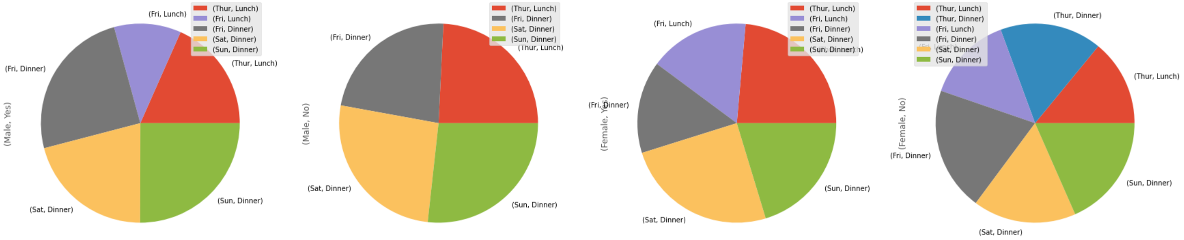
Out[138]:

```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



on multiindex dataframes

In [144]:
```
# using tips

tips.pivot_table(index=['day','time'],
                 columns=['sex','smoker'],
                 values='total_bill',
                 aggfunc='mean').plot(kind='pie',
                                      subplots=True,
                                      figsize=(30,20))
plt.show()
```



In [ ]:
```
```