Actividad Final

Eduardo Rodríguez Gil - A01274913, Jose Manuel Neri Villeda - A01706450, Héctor Javier Calderón González - A01067542

Actividad Evaluable: Obtención de estadisticas descriptivas

In [4]:
```
import pandas as pd
from matplotlib import pyplot as plt

data = pd.read_csv('Bitcoin.csv')
```

In [5]:
```
data
```

Out[5]:

	Date	Price	Open	High	Low
0	Apr 25, 2021	49561.9	50088.2	50438.8	49226.5
1	Apr 24, 2021	50088.9	51140.8	51183.0	48775.2
2	Apr 23, 2021	51143.6	51707.1	52099.9	47659.4
3	Apr 22, 2021	51729.5	53821.3	55408.4	50590.9
4	Apr 21, 2021	53820.2	56479.5	56764.4	53657.6
...	...	...	...	...	...
416	Mar 05, 2020	9060.3	8757.9	9147.3	8751.5
417	Mar 04, 2020	8757.9	8761.3	8840.3	8679.7
418	Mar 03, 2020	8761.4	8906.1	8911.7	8669.3
419	Mar 02, 2020	8904.8	8537.5	8961.8	8503.1
420	Mar 01, 2020	8540.0	8543.8	8737.2	8437.2

421 rows × 5 columns

In [6]:
```
filas=len(data)
filas
```

Out[6]:
```
421
```

En esta parte de la documentación, mostramos los datos cargados mediante el uso de la librería de Pandas, por lo que al revisar esta tabla se muestra que es lo que contiene nuestra base de datos que cuenta con 421 filas y 5 columnas.

In [7]:
```
data.columns
```

Out[7]:
```
Index(['Date', 'Price', 'Open', 'High', 'Low'], dtype='object')
```

In [9]:
```
type(data)
```

Out[9]:
```
pandas.core.frame.DataFrame
```

In [10]:
```
type(data.Date)
```

Out[10]:
```
pandas.core.series.Series
```

In [11]:
```
type(data.Price)
```

Out[11]:
```
pandas.core.series.Series
```

In [12]:
```
type(data.Open)
```

Out[12]:
```
pandas.core.series.Series
```

In [13]:
```
type(data.High)
```

Out[13]:
```
pandas.core.series.Series
```

In [14]:
```
type(data.Low)
```

Out[14]:
```
pandas.core.series.Series
```

Por medio de estos comandos, encontramos las 5 columnas con sus respectivos tipos de datos como lo es DataFrame y series.

In [16]:
```
pd.unique(data['Price']) #como se comporta el dato
```

Out[16]:
```
array([49561.9, 50088.9, 51143.6, 51729.5, 53820.2, 56483.2, 55646.1,
       56207.1, 60041.9, 61379.7, 63216. , 62980.4, 63540.9, 59863.8,
       59978.7, 59748.4, 58118.7, 58077.4, 55948.7, 57996.3, 58993.4,
       58199.9, 57059.9, 58977.3, 58718.3, 58763.7, 58771.3, 57616.2,
       55765.2, 55862.9, 55036.1, 51322.3, 52325.4, 54452.5, 54158.3,
       57383.8, 58093.4, 58088. , 57656. , 58913.5, 56889.7, 55791.3,
       59113.7, 61195.3, 57265.1, 57799.5, 55851.9, 54879. , 52311. ,
       50982.3, 48855.6, 48792.5, 48428. , 50395.1, 48424.2, 49595.5,
       45164. , 46136.7, 46345.6, 46928.5, 49697.5, 48911.2, 54111.8,
       57433.8, 55923.7, 55906.6, 51582.2, 52079.2, 49169.7, 47936.3,
       48643.4, 47168.7, 47371.7, 47990.7, 44836. , 46508.6, 46395.7,
       38852.9, 39256.6, 38297.6, 36982.1, 37646.8, 35485.2, 33515.7,
       33108.1, 34283.1, 34301.8, 33374.8, 30404. , 32502.1, 32252.3,
       32241.3, 32088.9, 33000.5, 30842.1, 35476.3, 36002.9, 36613.2,
       35839.6, 36019.5, 36845.8, 39175.7, 37382.2, 34076.1, 35544.3,
       38192.2, 40151.9, 40599.3, 39460.2, 36793.2, 33991.5, 32022.6,
       32958.9, 32193.3, 29359.9, 28949.4, 28868.7, 27376. , 27057.8,
       26261.3, 26454.4, 24689.6, 23736.5, 23257.9, 23823.2, 22728.5,
       23474.9, 23844. , 23127.9, 22825.4, 21352.2, 19434.9, 19273.8,
       19176.8, 18808.9, 18023.6, 18247.2, 18546. , 18326.6, 19170.7,
       19379.9, 19146.5, 18658.1, 19433.3, 19218.8, 18770.7, 19698.1,
       18185.5, 17730.7, 17127.1, 17162. , 18723. , 19152.6, 18379.6,
       18412.9, 18687.2, 18675.2, 17803.5, 17774.6, 17662.3, 16715.8,
       15953. , 16071. , 16324.2, 16294.7, 15695.8, 15303.6, 15327.2,
       15483.7, 14828.4, 15577.9, 15587.1, 14145.6, 14019.9, 13561.4,
       13759.4, 13797.3, 13559.9, 13457.2, 13278.9, 13657.8, 13061.6,
       13032.2, 13117.2, 12934.1, 12974.6, 12808.7, 11913.5, 11753.4,
       11506.9, 11362.1, 11322. , 11503. , 11420.4, 11423.8, 11533.9,
       11371. , 11298.4, 11054.2, 10924.1, 10670.9, 10602.6, 10789.5,
       10672.9, 10544.2, 10572.3, 10620.5, 10776.1, 10840.9, 10693.2,
```

```
       10776.2, 10727.9, 10688.8, 10739.4, 10237.3, 10531.5, 10416.8,
       10921.5, 11081.8, 10933. , 10941.3, 10949.5, 10785.3, 10675.3,
       10326. , 10441.9, 10390.2, 10339.7, 10224.6, 10126.6, 10376.9,
       10296.4, 10092.2, 10472.5, 10168.8, 11413.3, 11908.5, 11644.2,
       11702. , 11468.1, 11527.4, 11327.4, 11462.3, 11324.8, 11753.5,
       11641.6, 11661.3, 11529.2, 11856.9, 11750.2, 11947.6, 12282.6,
       11899. , 11845.3, 11750.8, 11770.9, 11557.2, 11390.4, 11889.2,
       11681.2, 11764.3, 11592. , 11757.1, 11735.1, 11184.7, 11224.4,
       11066.8, 11803.1, 11333.4, 11096.2, 11105.9, 10908.5, 11022.8,
        9932.5,  9704.1,  9546.4,  9599.6,  9513.7,  9387.3,  9162.4,
        9208. ,  9170.2,  9155.8,  9135.3,  9198.7,  9253.4,  9243.6,
        9300.8,  9233.3,  9285.1,  9235.7,  9429.9,  9256. ,  9339. ,
        9081. ,  9134.4,  9067.1,  9085.1,  9229.9,  9135.4,  9185.4,
        9124. ,  9008.3,  9160. ,  9247.5,  9302. ,  9624.6,  9683.7,
        9296.4,  9358.8,  9314. ,  9388.1,  9464.6,  9523.5,  9425.4,
        9345.3,  9471.3,  9466.6,  9283.2,  9878.8,  9768.8,  9777.9,
        9742.6,  9669.6,  9631.2,  9794.4,  9667.2,  9527.6, 10189.3,
        9454.8,  9692.5,  9424.8,  9572.2,  9199.1,  8842.5,  8898.2,
        8728.2,  9177. ,  9169.7,  9059. ,  9512.3,  9773.3,  9730.7,
        9677.7,  9379.5,  9318. ,  9778.4,  9298.7,  8813.8,  8579.8,
        8738.8,  9554.6,  9806.2,  9979.8,  9151.4,  9001. ,  8874.7,
        8885.5,  8966.3,  8821.6,  8629. ,  8770.9,  7746.9,  7766. ,
        7678.9,  7540.4,  7503.8,  7488.5,  7112.9,  6842.5,  6833.5,
        7122.9,  7230.8,  7035.8,  7085.6,  6629.1,  6850.9,  6841.3,
        6917.6,  6867.8,  6863.1,  7289. ,  7361.2,  7185.2,  7332.3,
        6772.7,  6857.4,  6735.9,  6800.5,  6638.5,  6412.5,  6391. ,
        5890.4,  6233.7,  6373.4,  6725.1,  6678.9,  6744.6,  6468.9,
        5822.1,  6186.2,  6205.3,  6172. ,  5361.4,  5261.1,  5030. ,
        5366.3,  5182.7,  5584.3,  4826. ,  7935.1,  7891.2,  7933. ,
        8034.1,  8887.8,  9134.8,  9060.3,  8757.9,  8761.4,  8904.8,
        8540. ])
```

In [17]:
```
data['Price'].describe()
```

Out[17]:
```
count      421.000000
mean     21471.073872
std      17492.702670
min       4826.000000
25%       9314.000000
50%      11557.200000
75%      32958.900000
max      63540.900000
Name: Price, dtype: float64
```

In [18]:
```
data['Price'].std()
```

Out[18]:
```
17492.702669813672
```

In [19]:
```
data['Price'].median()
```

Out[19]:
```
11557.2
```

In [20]:
```
data['Price'].mean()
```

Out[20]:
```
21471.07387173399
```

Al analizar estos datos tenemos una desviación estándar muy alta por lo que me lleva a deducir que su precio es muy cambiante respecto a esta criptomoneda, como media se tiene que el valor medio es de 11557.2. Por último, se tiene un promedio de sus 421 datos que se tiene de registro desde marzo de 2020 dando un valor de 21471.07387173399.

Actividad Evaluable: Mapas de calor y boxplots

In [21]:
```
import pandas as pd
import seaborn as sb
import numpy as np; np.random.seed(0)
import matplotlib.pyplot as plt
data = pd.read_csv('Bitcoin.csv')
from matplotlib import cm
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
# Voy a revisar dimensiones
data.shape
```

Out[21]:
```
(421, 5)
```

Son 421 registros con 5 columnas que no se me olvide que puedo intercalar texto y fórmulas.

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

Out[22]:

	Date	Price	Open	High	Low
0	Apr 25, 2021	49561.9	50088.2	50438.8	49226.5
1	Apr 24, 2021	50088.9	51140.8	51183.0	48775.2
2	Apr 23, 2021	51143.6	51707.1	52099.9	47659.4
3	Apr 22, 2021	51729.5	53821.3	55408.4	50590.9
4	Apr 21, 2021	53820.2	56479.5	56764.4	53657.6

In [23]:
```
data.describe()
```

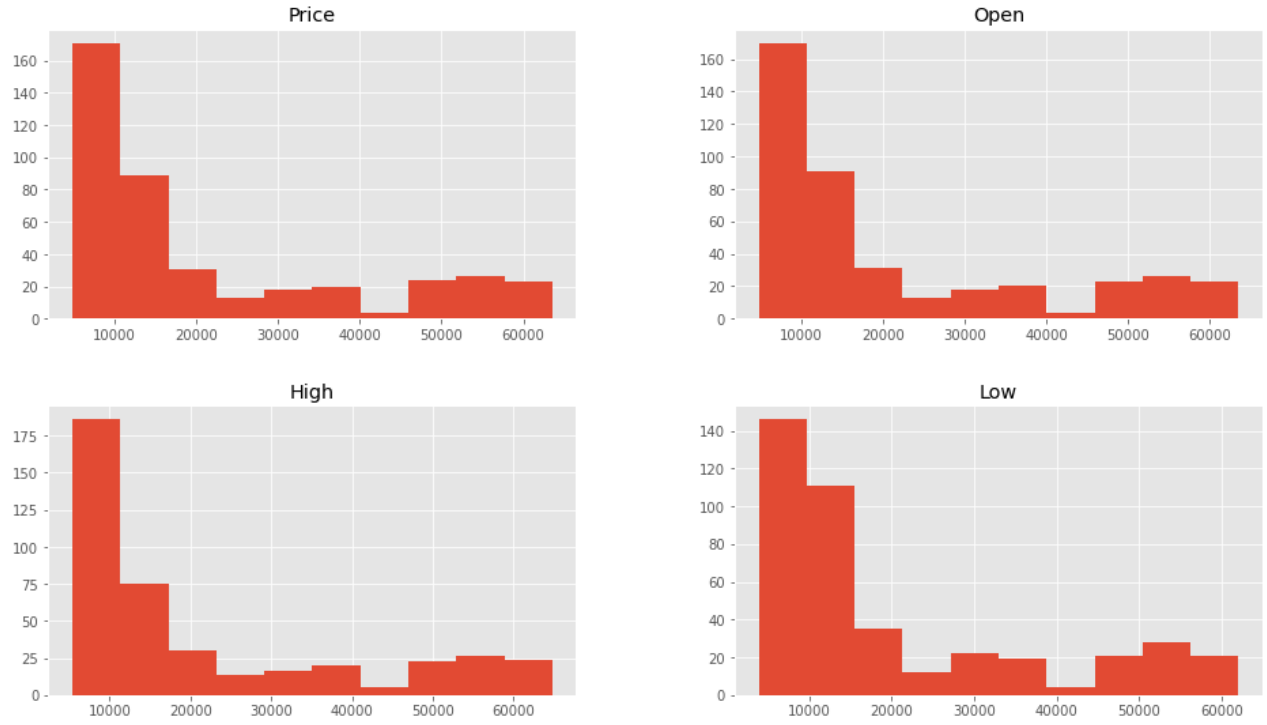
Out[23]:

	Price	Open	High	Low
count	421.000000	421.000000	421.000000	421.000000
mean	21471.073872	21372.344181	22028.754869	20687.659857
std	17492.702670	17448.718099	18024.928136	16785.882734
min	4826.000000	4815.200000	5369.300000	3869.500000
25%	9314.000000	9300.800000	9458.300000	9184.200000
50%	11557.200000	11533.500000	11766.900000	11315.900000
75%	32958.900000	32499.600000	34348.300000	30850.000000
max	63540.900000	63544.200000	64778.000000	62067.500000

Visualización general

Eliminar etiquetas de filas o columnas

In [25]:
```
data.drop ([0, 1]).hist()
plt.show ()
```



Filtros

In [26]:
```
mas_de_40 = data[data['Price'] > 4000]
mas_de_40
```

Out[26]:

	Date	Price	Open	High	Low
0	Apr 25, 2021	49561.9	50088.2	50438.8	49226.5
1	Apr 24, 2021	50088.9	51140.8	51183.0	48775.2
2	Apr 23, 2021	51143.6	51707.1	52099.9	47659.4
3	Apr 22, 2021	51729.5	53821.3	55408.4	50590.9
4	Apr 21, 2021	53820.2	56479.5	56764.4	53657.6
...	...	...	...	...	...
416	Mar 05, 2020	9060.3	8757.9	9147.3	8751.5
417	Mar 04, 2020	8757.9	8761.3	8840.3	8679.7
418	Mar 03, 2020	8761.4	8906.1	8911.7	8669.3
419	Mar 02, 2020	8904.8	8537.5	8961.8	8503.1
420	Mar 01, 2020	8540.0	8543.8	8737.2	8437.2

421 rows × 5 columns

Doble filtro

In [27]:
```
doble_filtro = data[(data['Price'] > 4000) & (data['High'] > 35000)]
doble_filtro
```

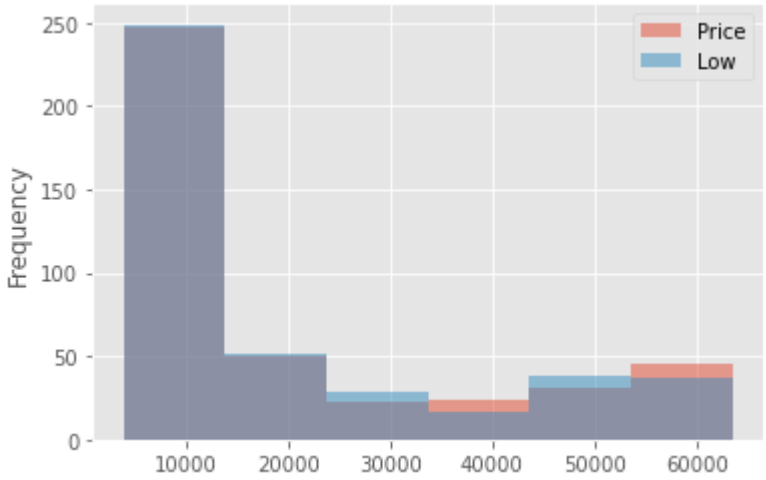
Out[27]:

	Date	Price	Open	High	Low
0	Apr 25, 2021	49561.9	50088.2	50438.8	49226.5
1	Apr 24, 2021	50088.9	51140.8	51183.0	48775.2
2	Apr 23, 2021	51143.6	51707.1	52099.9	47659.4
3	Apr 22, 2021	51729.5	53821.3	55408.4	50590.9
4	Apr 21, 2021	53820.2	56479.5	56764.4	53657.6
...	...	...	...	...	...
105	Jan 10, 2021	38192.2	40149.7	41362.4	35141.6
106	Jan 09, 2021	40151.9	40607.2	41363.5	38775.1
107	Jan 08, 2021	40599.3	39466.4	41921.7	36613.4
108	Jan 07, 2021	39460.2	36798.5	40340.9	36361.2
109	Jan 06, 2021	36793.2	33999.3	36934.8	33408.3

100 rows × 5 columns

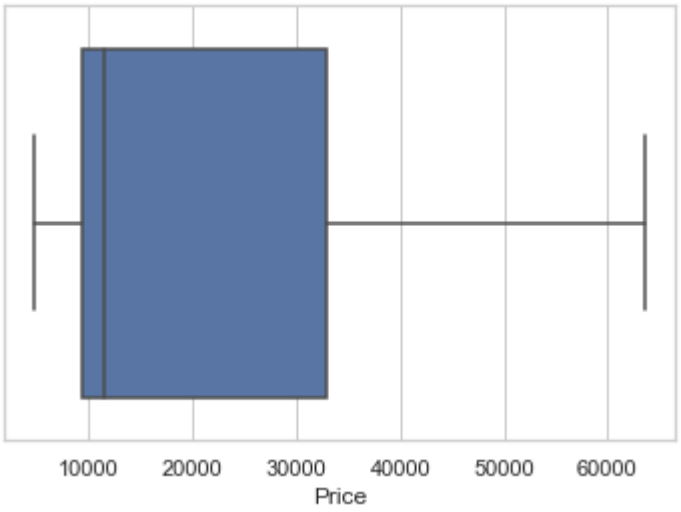
Visualización

In [28]:
```
%matplotlib inline
data[['Price', 'Low']].plot.hist(bins=6, alpha=0.5)
plt.show()
```

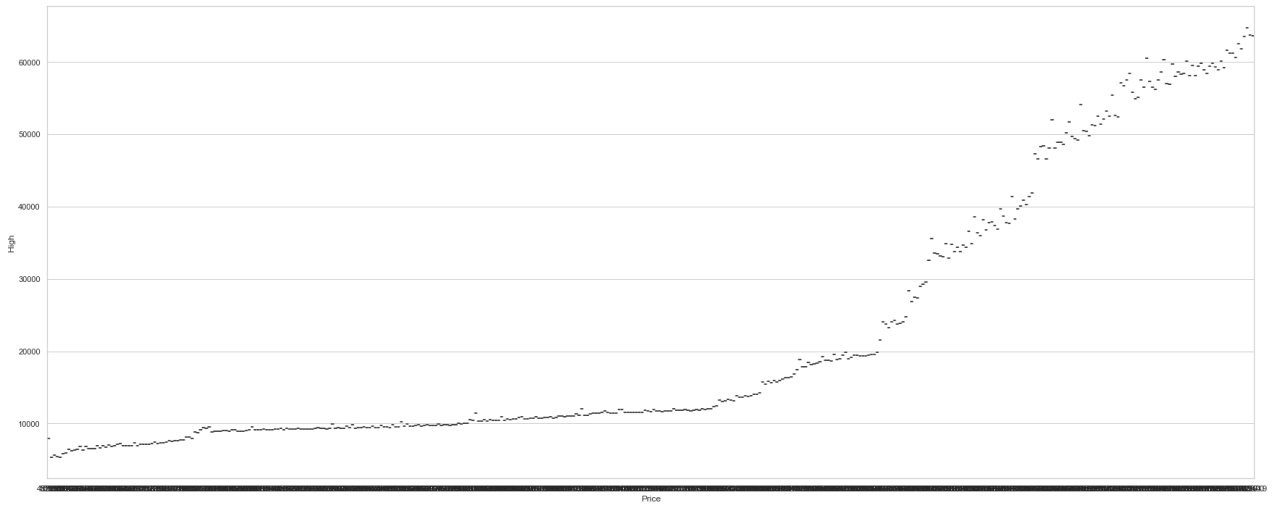


Boxplot para obtener un diagrama de cajas y bigotes

In [41]:
```
sb.set_theme(style = "whitegrid")
Bitcoin = pd.read_csv('Data Bitcoin.csv')
ax = sb.boxplot(x = Bitcoin["Price"])
```

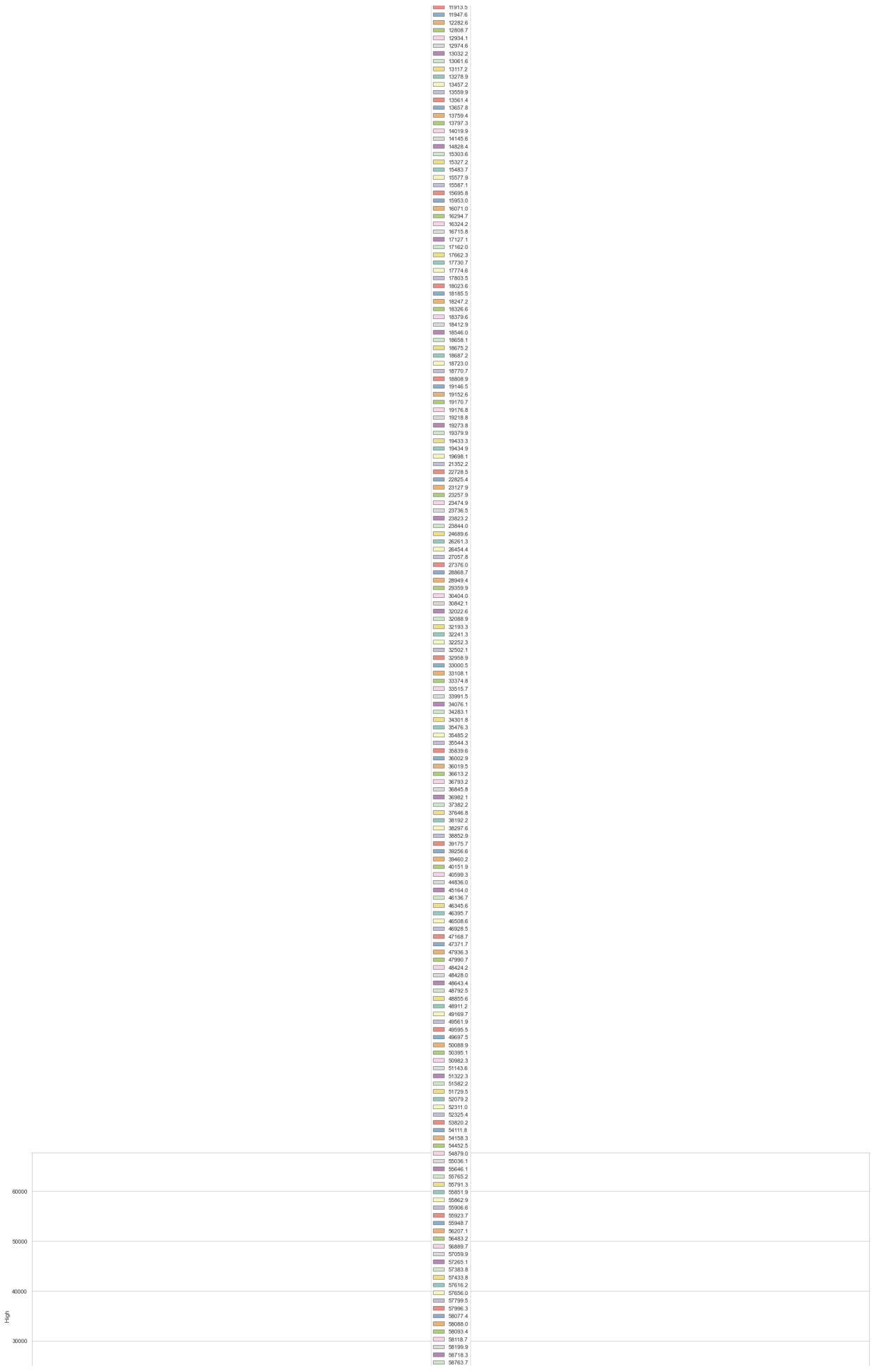


In [30]:
```
plt.figure(figsize=(30,12))
ax = sb.boxplot(x = "Price", y = "High", data = Bitcoin)
```

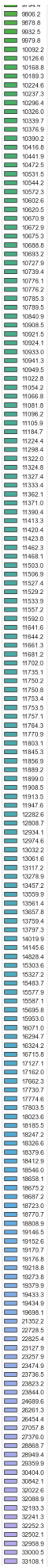


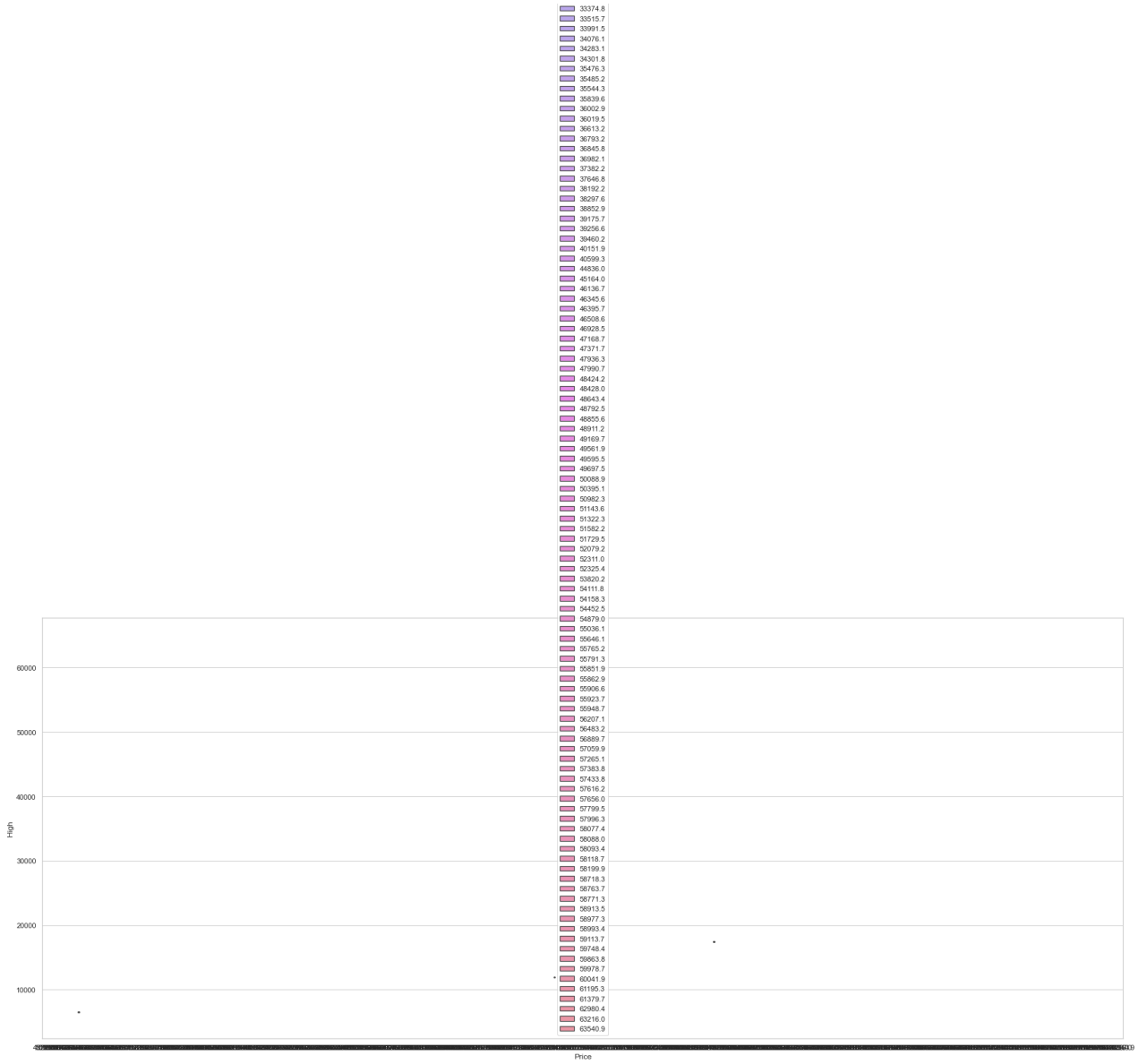
In [31]:
```
plt.figure(figsize=(30,12))
ax = sb.boxplot(x = "Price", y = "High", hue = "Price",
                data = Bitcoin, palette = "Set3")
```


8738.8
8757.9
8761.4
8770.9
8813.8
8821.6
8842.5
8874.7
8885.5
8887.8
8898.2
8904.6
8966.3
9001.0
9008.3
9059.0
9060.3
9067.1
9081.0
9085.1
9124.0
9134.4
9134.8
9135.3
9135.4
9151.4
9155.8
9160.0
9162.4
9169.7
9170.2
9177.0
9185.4
9198.7
9199.1
9208.0
9229.9
9233.3
9235.7
9243.6
9247.5
9253.4
9256.0
9283.2
9285.1
9296.4
9298.7
9300.8
9302.0
9314.0
9318.0
9339.0
9345.3
9358.8
9379.5
9387.3
9388.1
9424.8
9425.4
9429.9
9454.8
9464.6
9466.6
9471.3
9512.3
9513.7
9523.5
9527.6
9546.4
9554.6
9572.2
9599.6
9624.6
9631.2
9667.2
9669.6
9677.7
9683.7
9692.5
9704.1
9730.7
9742.6
9768.8
9773.3
9777.9
9778.4
9794.4
9806.2
9878.8
9932.5
9979.8
10092.2
10126.6
10168.8
10189.3
10224.6
10237.3
10296.4
10326.0
10339.7
10376.9
10390.2
10416.8
10441.9
10472.5
10531.5
10544.2
10572.3
10602.6
10620.5
10670.9
10672.9
10675.3
10688.8
10693.2
10727.9
10739.4
10776.1
10776.2
10785.3
10789.5
10840.9
10908.5
10921.5
10924.1
10933.0
10941.3
10949.5
11022.8
11054.2
11066.8
11081.8
11096.2
11105.9
11184.7
11224.4
11298.4
11322.0
11324.8
11327.4
11333.4
11362.1
11371.0
11390.4
11413.3
11420.4
11423.8
11462.3
11468.1
11503.0
11506.9
11527.4
11529.2
11533.9
11557.2
11592.0
11641.6
11644.2
11661.3
11681.2
11702.0
11735.1
11750.2
11750.8
11753.4
11753.5
11757.1
11764.3
11770.9
11803.1
11845.3
11856.9
11889.2
11899.0
11908.5


11913.5
11947.6
12282.6
12808.7
12934.1
12974.6
13032.2
13061.6
13117.2
13278.9
13457.2
13559.9
13561.4
13657.8
13759.4
13797.3
14019.9
14145.6
14828.4
15303.6
15327.2
15483.7
15577.9
15587.1
15695.8
15953.0
16071.0
16294.7
16324.2
16715.8
17127.1
17162.0
17662.3
17730.7
17774.6
17803.5
18023.6
18185.5
18247.2
18326.6
18379.6
18412.9
18546.0
18658.1
18675.2
18687.2
18723.0
18770.7
18808.9
19146.5
19152.6
19170.7
19176.8
19218.8
19273.8
19379.9
19433.3
19434.9
19698.1
21352.2
22728.5
22825.4
23127.9
23257.9
23474.9
23736.5
23823.2
23844.0
24689.6
26261.3
26454.4
27057.8
27376.0
28868.7
28949.4
29359.9
30404.0
30842.1
32022.6
32088.9
32193.3
32241.3
32252.3
32502.1
32958.9
33000.5
33108.1
33374.8
33515.7
33991.5
34076.1
34283.1
34301.8
35476.3
35485.2
35544.3
35839.6
36002.9
36019.5
36613.2
36793.2
36845.8
36982.1
37382.2
37646.8
38192.2
38297.6
38852.9
39175.7
39256.6
39460.2
40151.9
40599.3
44836.0
45164.0
46136.7
46345.6
46395.7
46508.6
46928.5
47168.7
47371.7
47936.3
47990.7
48424.2
48428.0
48643.4
48792.5
48855.6
48911.2
49169.7
49561.9
49595.5
49697.5
50088.9
50395.1
50982.3
51143.6
51322.3
51582.2
51729.5
52079.2
52311.0
52325.4
53820.2
54111.8
54158.3
54452.5
54879.0
55036.1
55646.1
55765.2
55791.3
55851.9
55862.9
55906.6
55923.7
55948.7
56207.1
56483.2
56889.7
57059.9
57265.1
57383.8
57433.8
57616.2
57656.0
57799.5
57996.3
58077.4
58088.0
58093.4
58118.7
58199.9
58718.3
58763.7
60000
50000
40000
30000
High

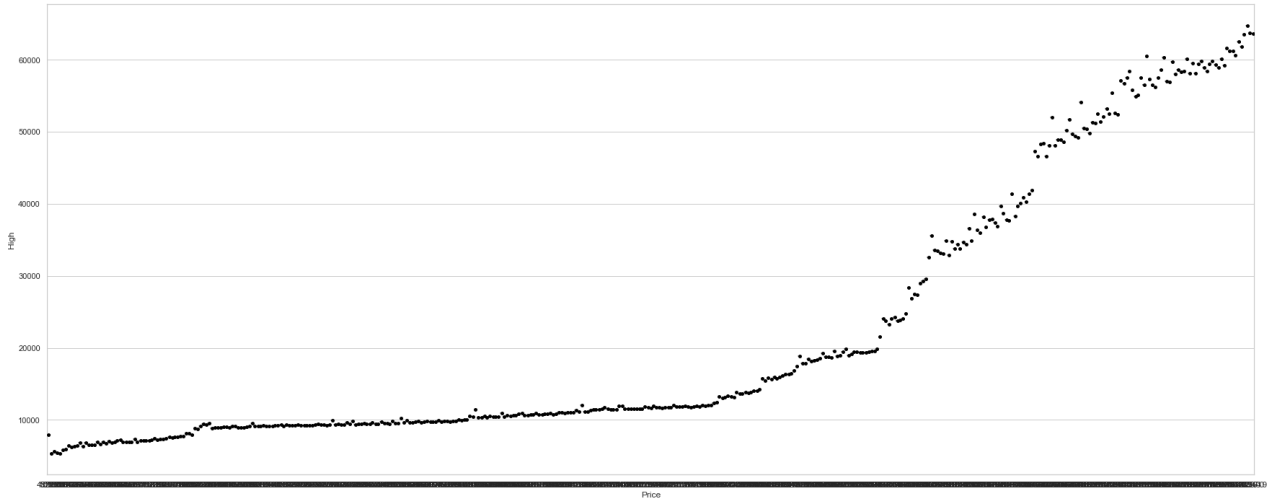
In [42]:
```
plt.figure(figsize=(30,12))
ax = sb.boxplot(x = "Price", y = "High", hue = "Price",
                data = Bitcoin, linewidth = 2.5)
```


9806.2
9678.8
9932.5
9979.8
10092.2
10126.6
10168.8
10189.3
10224.6
10237.3
10296.4
10326.0
10339.7
10376.9
10390.2
10416.8
10441.9
10472.5
10531.5
10544.2
10572.3
10602.6
10620.5
10670.9
10672.9
10675.3
10688.8
10693.2
10727.9
10739.4
10776.1
10776.2
10785.3
10789.5
10840.9
10908.5
10921.5
10924.1
10933.0
10941.3
10949.5
11022.8
11054.2
11066.8
11081.8
11096.2
11105.9
11184.7
11224.4
11298.4
11322.0
11324.8
11327.4
11333.4
11362.1
11371.0
11390.4
11413.3
11420.4
11423.8
11462.3
11468.1
11503.0
11506.9
11527.4
11529.2
11533.9
11557.2
11592.0
11641.6
11644.2
11681.3
11681.2
11702.0
11735.1
11750.2
11750.8
11753.4
11753.5
11757.1
11764.3
11770.9
11803.1
11845.3
11856.9
11889.2
11899.0
11908.5
11913.5
11947.6
12282.6
12808.7
12934.1
12974.6
13032.2
13061.6
13117.2
13278.9
13457.2
13559.9
13561.4
13657.8
13759.4
13797.3
14019.9
14145.6
14828.4
15303.6
15327.2
15483.7
15577.9
15587.1
15695.8
15953.0
16071.0
16294.7
16324.2
16715.8
17127.1
17162.0
17662.3
17730.7
17774.6
17803.5
18023.6
18185.5
18247.2
18326.6
18379.6
18412.9
18546.0
18658.1
18675.2
18687.2
18723.0
18770.7
18808.9
19146.5
19152.6
19170.7
19176.8
19218.8
19273.8
19379.9
19433.3
19434.9
19698.1
21352.2
22728.5
22825.4
23127.9
23257.9
23474.9
23736.5
23823.2
23844.0
24689.6
26261.3
26454.4
27057.8
27376.0
28868.7
28949.4
29359.9
30404.0
30842.1
32022.6
32088.9
32193.3
32241.3
32252.3
32502.1
32958.9
33000.5
33108.1



Cajas

In [43]:
```
plt.figure(figsize=(30,12))
ax = sb.boxplot(x = "Price", y = "High", data = Bitcoin)
ax = sb.stripplot(x = "Price", y = "High", data = Bitcoin, color = "0.0000005")
```



Correlación

In [32]:
```
data.corr(method = 'pearson')
```

Out[32]:

	Price	Open	High	Low
Price	1.000000	0.997997	0.999207	0.998767
Open	0.997997	1.000000	0.999032	0.998379
High	0.999207	0.999032	1.000000	0.998287
Low	0.998767	0.998379	0.998287	1.000000

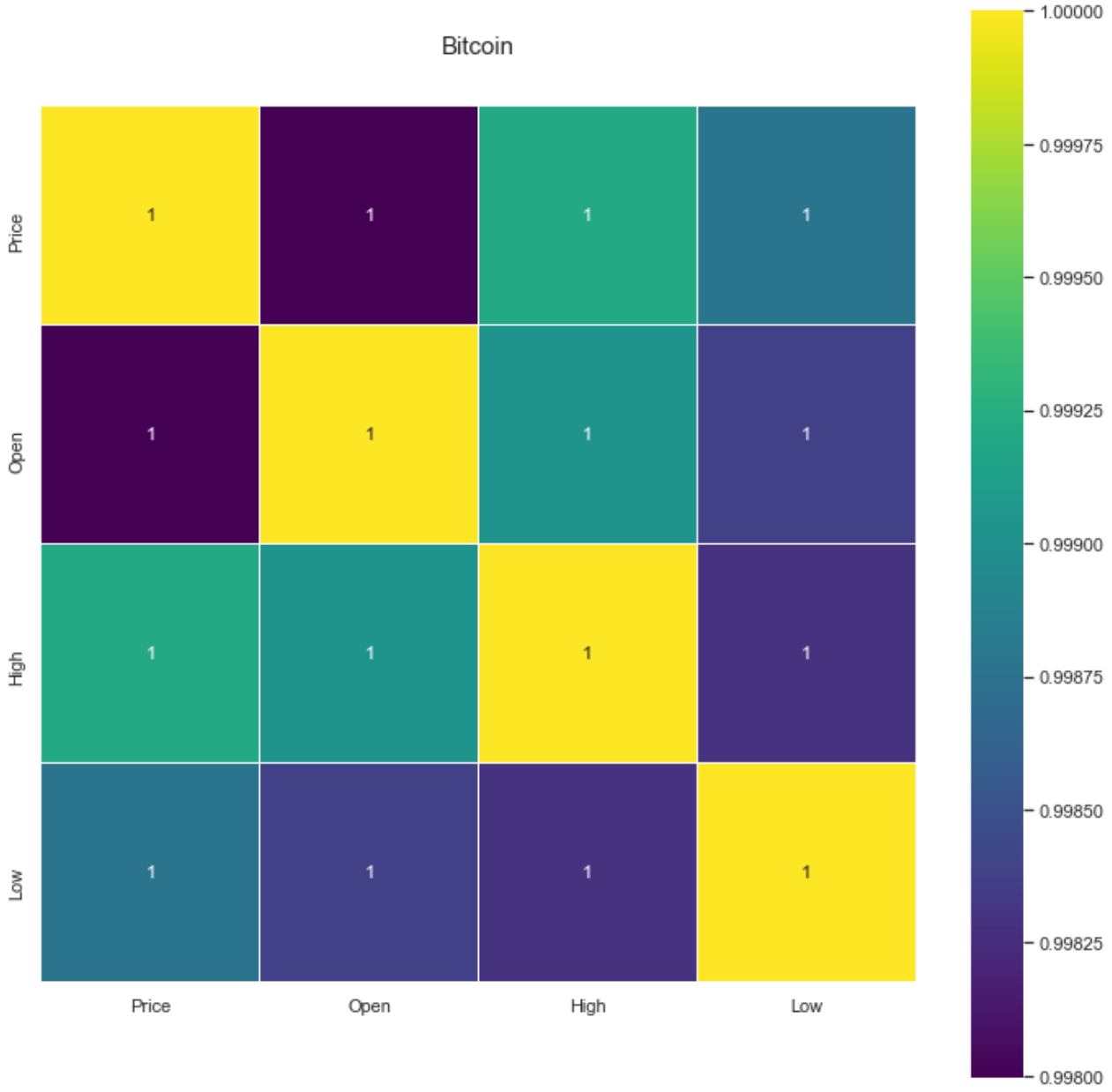
In [33]:
```
data.corr(method = 'kendall')
```

Out[33]:

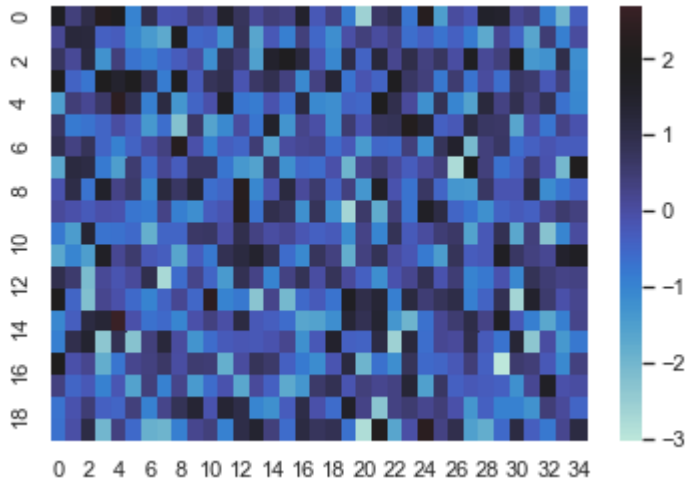
	Price	Open	High	Low
Price	1.000000	0.945413	0.966768	0.969574
Open	0.945413	1.000000	0.965298	0.962402
High	0.966768	0.965298	1.000000	0.957154
Low	0.969574	0.962402	0.957154	1.000000

Visualizar mapa de calor

In [34]:
```
colormap = plt.cm.viridis
plt.figure(figsize = (12, 12))
plt.title('Bitcoin', y = 1.05, size = 15)
sb.heatmap(data.corr(), linewidths = 0.1, vmax = 1.0, square = True, cmap = colormap, l
```

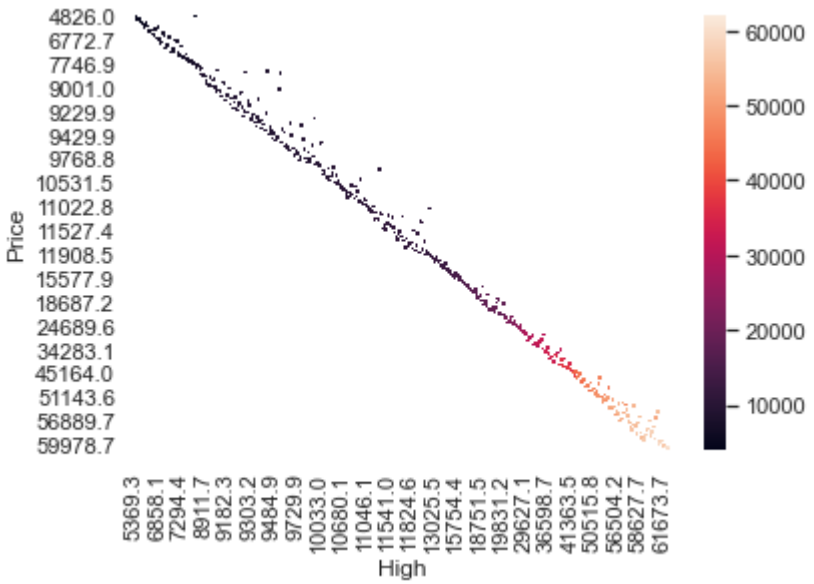


In [35]:
```
normal_data = np.random.randn(20, 35)
ax = sb.heatmap(normal_data, center = 2)
```



In [37]:
```
Bitcoin = pd.read_csv('Bitcoin.csv')
```

```
Bitcoin = Bitcoin.pivot("Price", "High", "Low")
ax = sb.heatmap(Bitcoin)
```



Actividad Evaluable: Patrones con K-means

In [44]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import sklearn
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import scale
import sklearn.metrics as sm
from sklearn import datasets


%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

In [45]:
```
dataframe = pd.read_csv(r"Bitcoin.csv") # Base de datos
dataframe.head()
```

Out[45]:

	Date	Price	Open	High	Low
0	Apr 25, 2021	49561.9	50088.2	50438.8	49226.5
1	Apr 24, 2021	50088.9	51140.8	51183.0	48775.2
2	Apr 23, 2021	51143.6	51707.1	52099.9	47659.4
3	Apr 22, 2021	51729.5	53821.3	55408.4	50590.9
4	Apr 21, 2021	53820.2	56479.5	56764.4	53657.6

Para este punto en nuestra base de datos quitamos dos variables la de Volume y la de Change, ya

que no hacías un gran cambio en nuestros datos, ya que al momento de graficar no los tomábamos en cuenta al no ser unos valores numéricos.

In [46]:
```
dataframe.describe()
```

Out[46]:

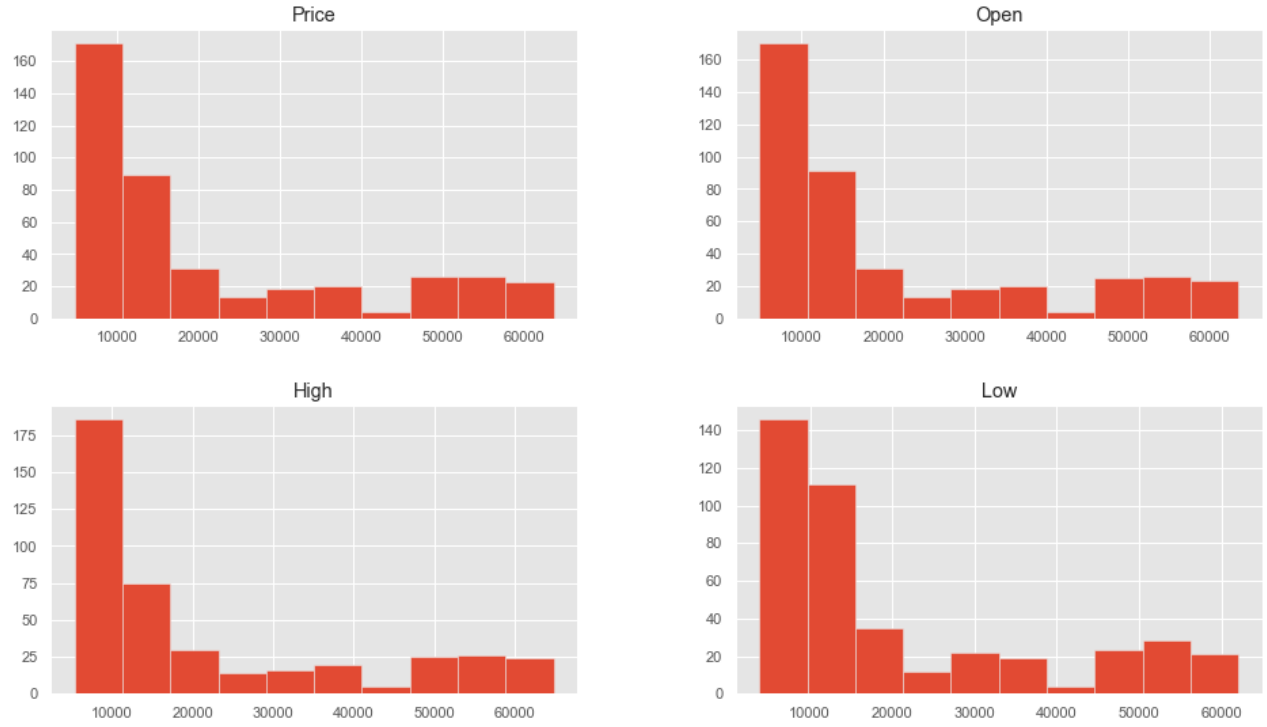
	Price	Open	High	Low
count	421.000000	421.000000	421.000000	421.000000
mean	21471.073872	21372.344181	22028.754869	20687.659857
std	17492.702670	17448.718099	18024.928136	16785.882734
min	4826.000000	4815.200000	5369.300000	3869.500000
25%	9314.000000	9300.800000	9458.300000	9184.200000
50%	11557.200000	11533.500000	11766.900000	11315.900000
75%	32958.900000	32499.600000	34348.300000	30850.000000
max	63540.900000	63544.200000	64778.000000	62067.500000

In [47]:
```
# Vemos en cuanto esta el Precio de la Bitcoin
print(dataframe.groupby('Date').size())
```

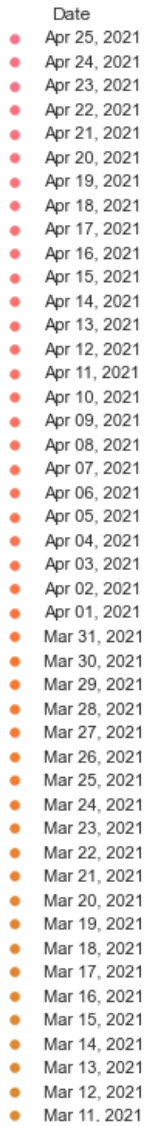
```
Date
Apr 01, 2020    1
Apr 01, 2021    1
Apr 02, 2020    1
Apr 02, 2021    1
Apr 03, 2020    1
               ..
Sep 26, 2020    1
Sep 27, 2020    1
Sep 28, 2020    1
Sep 29, 2020    1
Sep 30, 2020    1
Length: 421, dtype: int64
```

Visualizamos los datos

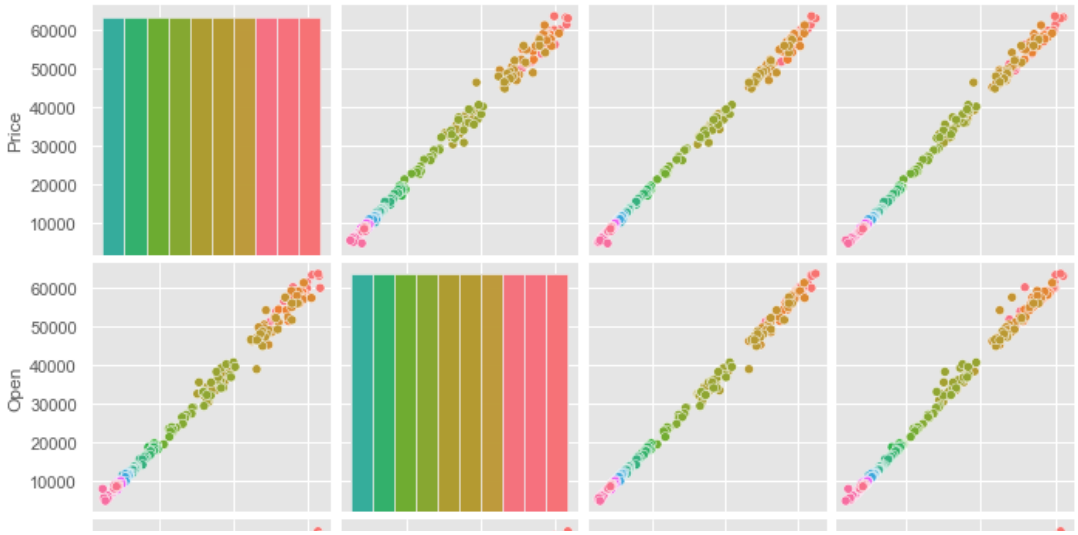
In [48]:
```
dataframe.drop(['Date'], 1).hist()
plt.show()
```



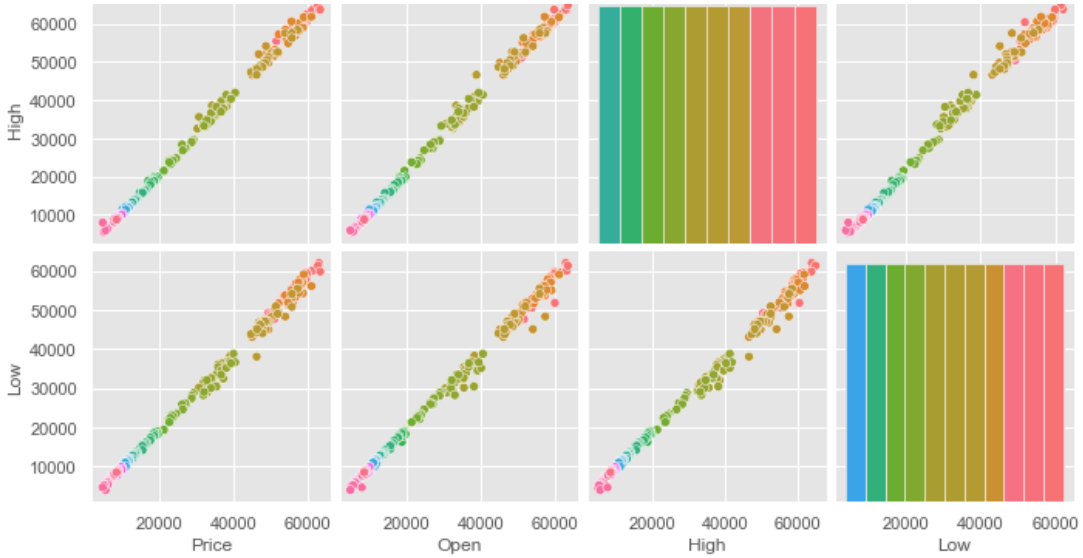
In [49]:
```
sb.pairplot(dataframe, hue = "Date", diag_kind = "hist");
```



Mar 10, 2021
Mar 09, 2021
Mar 08, 2021
Mar 07, 2021
Mar 06, 2021
Mar 05, 2021
Mar 04, 2021
Mar 03, 2021
Mar 02, 2021
Mar 01, 2021
Feb 28, 2021
Feb 27, 2021
Feb 26, 2021
Feb 25, 2021
Feb 24, 2021
Feb 23, 2021
Feb 22, 2021
Feb 21, 2021
Feb 20, 2021
Feb 19, 2021
Feb 18, 2021
Feb 17, 2021
Feb 16, 2021
Feb 15, 2021
Feb 14, 2021
Feb 13, 2021
Feb 12, 2021
Feb 11, 2021
Feb 10, 2021
Feb 09, 2021
Feb 08, 2021
Feb 07, 2021
Feb 06, 2021
Feb 05, 2021
Feb 04, 2021
Feb 03, 2021
Feb 02, 2021
Feb 01, 2021
Jan 31, 2021
Jan 30, 2021
Jan 29, 2021
Jan 28, 2021
Jan 27, 2021
Jan 26, 2021
Jan 25, 2021
Jan 24, 2021
Jan 23, 2021
Jan 22, 2021
Jan 21, 2021
Jan 20, 2021
Jan 19, 2021
Jan 18, 2021
Jan 17, 2021
Jan 16, 2021
Jan 15, 2021
Jan 14, 2021
Jan 13, 2021
Jan 12, 2021
Jan 11, 2021
Jan 10, 2021
Jan 09, 2021
Jan 08, 2021
Jan 07, 2021
Jan 06, 2021
Jan 05, 2021
Jan 04, 2021
Jan 03, 2021
Jan 02, 2021
Jan 01, 2021
Dec 31, 2020
Dec 30, 2020
Dec 29, 2020
Dec 28, 2020
Dec 27, 2020
Dec 26, 2020
Dec 25, 2020
Dec 24, 2020
Dec 23, 2020
Dec 22, 2020
Dec 21, 2020
Dec 20, 2020
Dec 19, 2020



Dec 18, 2020
Dec 17, 2020
Dec 16, 2020
Dec 15, 2020
Dec 14, 2020
Dec 13, 2020
Dec 12, 2020
Dec 11, 2020
Dec 10, 2020
Dec 09, 2020
Dec 08, 2020
Dec 07, 2020
Dec 06, 2020
Dec 05, 2020
Dec 04, 2020
Dec 03, 2020
Dec 02, 2020
Dec 01, 2020
Nov 30, 2020
Nov 29, 2020
Nov 28, 2020
Nov 27, 2020
Nov 26, 2020
Nov 25, 2020
Nov 24, 2020
Nov 23, 2020
Nov 22, 2020
Nov 21, 2020
Nov 20, 2020
Nov 19, 2020
Nov 18, 2020
Nov 17, 2020
Nov 16, 2020
Nov 15, 2020
Nov 14, 2020
Nov 13, 2020
Nov 12, 2020
Nov 11, 2020
Nov 10, 2020
Nov 09, 2020
Nov 08, 2020
Nov 07, 2020
Nov 06, 2020
Nov 05, 2020
Nov 04, 2020
Nov 03, 2020
Nov 02, 2020
Nov 01, 2020
Oct 31, 2020
Oct 30, 2020
Oct 29, 2020
Oct 28, 2020
Oct 27, 2020
Oct 26, 2020
Oct 25, 2020
Oct 24, 2020
Oct 23, 2020
Oct 22, 2020
Oct 21, 2020
Oct 20, 2020
Oct 19, 2020
Oct 18, 2020
Oct 17, 2020
Oct 16, 2020
Oct 15, 2020
Oct 14, 2020
Oct 13, 2020
Oct 12, 2020
Oct 11, 2020
Oct 10, 2020
Oct 09, 2020
Oct 08, 2020
Oct 07, 2020
Oct 06, 2020
Oct 05, 2020
Oct 04, 2020
Oct 03, 2020
Oct 02, 2020
Oct 01, 2020
Sep 30, 2020
Sep 29, 2020



Sep 28, 2020
Sep 27, 2020
Sep 26, 2020
Sep 25, 2020
Sep 24, 2020
Sep 23, 2020
Sep 22, 2020
Sep 21, 2020
Sep 20, 2020
Sep 19, 2020
Sep 18, 2020
Sep 17, 2020
Sep 16, 2020
Sep 15, 2020
Sep 14, 2020
Sep 13, 2020
Sep 12, 2020
Sep 11, 2020
Sep 10, 2020
Sep 09, 2020
Sep 08, 2020
Sep 07, 2020
Sep 06, 2020
Sep 05, 2020
Sep 04, 2020
Sep 03, 2020
Sep 02, 2020
Sep 01, 2020
Aug 31, 2020
Aug 30, 2020
Aug 29, 2020
Aug 28, 2020
Aug 27, 2020
Aug 26, 2020
Aug 25, 2020
Aug 24, 2020
Aug 23, 2020
Aug 22, 2020
Aug 21, 2020
Aug 20, 2020
Aug 19, 2020
Aug 18, 2020
Aug 17, 2020
Aug 16, 2020
Aug 15, 2020
Aug 14, 2020
Aug 13, 2020
Aug 12, 2020
Aug 11, 2020
Aug 10, 2020
Aug 09, 2020
Aug 08, 2020
Aug 07, 2020
Aug 06, 2020
Aug 05, 2020
Aug 04, 2020
Aug 03, 2020
Aug 02, 2020
Aug 01, 2020
Jul 31, 2020
Jul 30, 2020
Jul 29, 2020
Jul 28, 2020
Jul 27, 2020
Jul 26, 2020
Jul 25, 2020
Jul 24, 2020
Jul 23, 2020
Jul 22, 2020
Jul 21, 2020
Jul 20, 2020
Jul 19, 2020
Jul 18, 2020
Jul 17, 2020
Jul 16, 2020
Jul 15, 2020
Jul 14, 2020
Jul 13, 2020
Jul 12, 2020
Jul 11, 2020
Jul 10, 2020
Jul 09, 2020

Jul 08, 2020
Jul 07, 2020
Jul 06, 2020
Jul 05, 2020
Jul 04, 2020
Jul 03, 2020
Jul 02, 2020
Jul 01, 2020
Jun 30, 2020
Jun 29, 2020
Jun 28, 2020
Jun 27, 2020
Jun 26, 2020
Jun 25, 2020
Jun 24, 2020
Jun 23, 2020
Jun 22, 2020
Jun 21, 2020
Jun 20, 2020
Jun 19, 2020
Jun 18, 2020
Jun 17, 2020
Jun 16, 2020
Jun 15, 2020
Jun 14, 2020
Jun 13, 2020
Jun 12, 2020
Jun 11, 2020
Jun 10, 2020
Jun 09, 2020
Jun 08, 2020
Jun 07, 2020
Jun 06, 2020
Jun 05, 2020
Jun 04, 2020
Jun 03, 2020
Jun 02, 2020
Jun 01, 2020
May 31, 2020
May 30, 2020
May 29, 2020
May 28, 2020
May 27, 2020
May 26, 2020
May 25, 2020
May 24, 2020
May 23, 2020
May 22, 2020
May 21, 2020
May 20, 2020
May 19, 2020
May 18, 2020
May 17, 2020
May 16, 2020
May 15, 2020
May 14, 2020
May 13, 2020
May 12, 2020
May 11, 2020
May 10, 2020
May 09, 2020
May 08, 2020
May 07, 2020
May 06, 2020
May 05, 2020
May 04, 2020
May 03, 2020
May 02, 2020
May 01, 2020
Apr 30, 2020
Apr 29, 2020
Apr 28, 2020
Apr 27, 2020
Apr 26, 2020
Apr 25, 2020
Apr 24, 2020
Apr 23, 2020
Apr 22, 2020
Apr 21, 2020
Apr 20, 2020
Apr 19, 2020
Apr 18, 2020

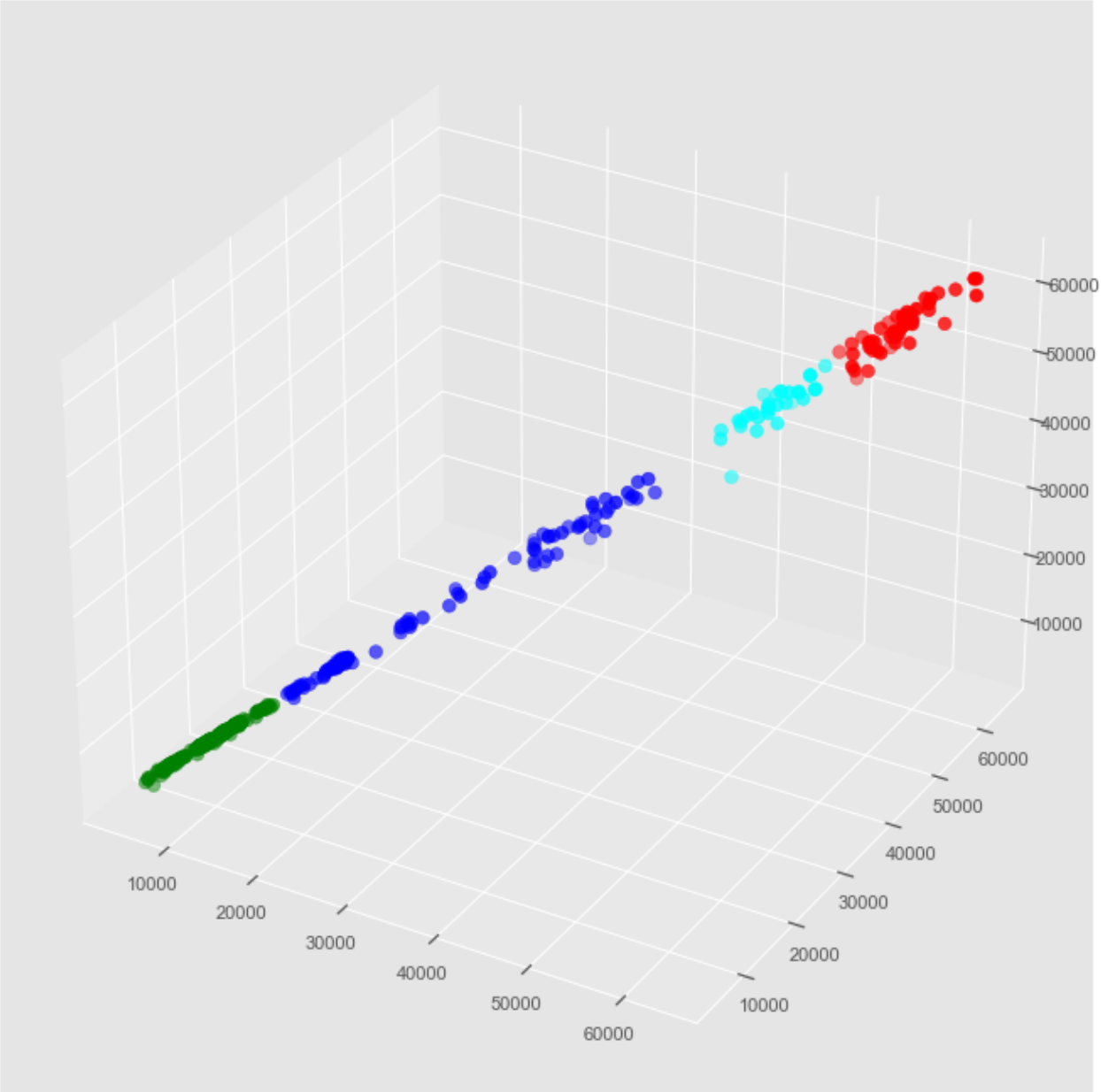
- Apr 17, 2020
- Apr 16, 2020
- Apr 15, 2020
- Apr 14, 2020
- Apr 13, 2020
- Apr 12, 2020
- Apr 11, 2020
- Apr 10, 2020
- Apr 09, 2020
- Apr 08, 2020
- Apr 07, 2020
- Apr 06, 2020
- Apr 05, 2020
- Apr 04, 2020
- Apr 03, 2020
- Apr 02, 2020
- Apr 01, 2020
- Mar 31, 2020
- Mar 30, 2020
- Mar 29, 2020
- Mar 28, 2020
- Mar 27, 2020
- Mar 26, 2020
- Mar 25, 2020
- Mar 24, 2020
- Mar 23, 2020
- Mar 22, 2020
- Mar 21, 2020
- Mar 20, 2020
- Mar 19, 2020
- Mar 18, 2020
- Mar 17, 2020
- Mar 16, 2020
- Mar 15, 2020
- Mar 14, 2020
- Mar 13, 2020
- Mar 12, 2020
- Mar 11, 2020
- Mar 10, 2020
- Mar 09, 2020
- Mar 08, 2020
- Mar 07, 2020
- Mar 06, 2020
- Mar 05, 2020
- Mar 04, 2020
- Mar 03, 2020
- Mar 02, 2020
- Mar 01, 2020

Creamos el modelo

In [74]:
```
# Para el ejercicio, sólo seleccionamos 3 dimensiones, para poder graficarlo
X = np.array(dataframe[["Price", "High", "Low"]])
y = np.array(dataframe['Date'])
X.shape
```

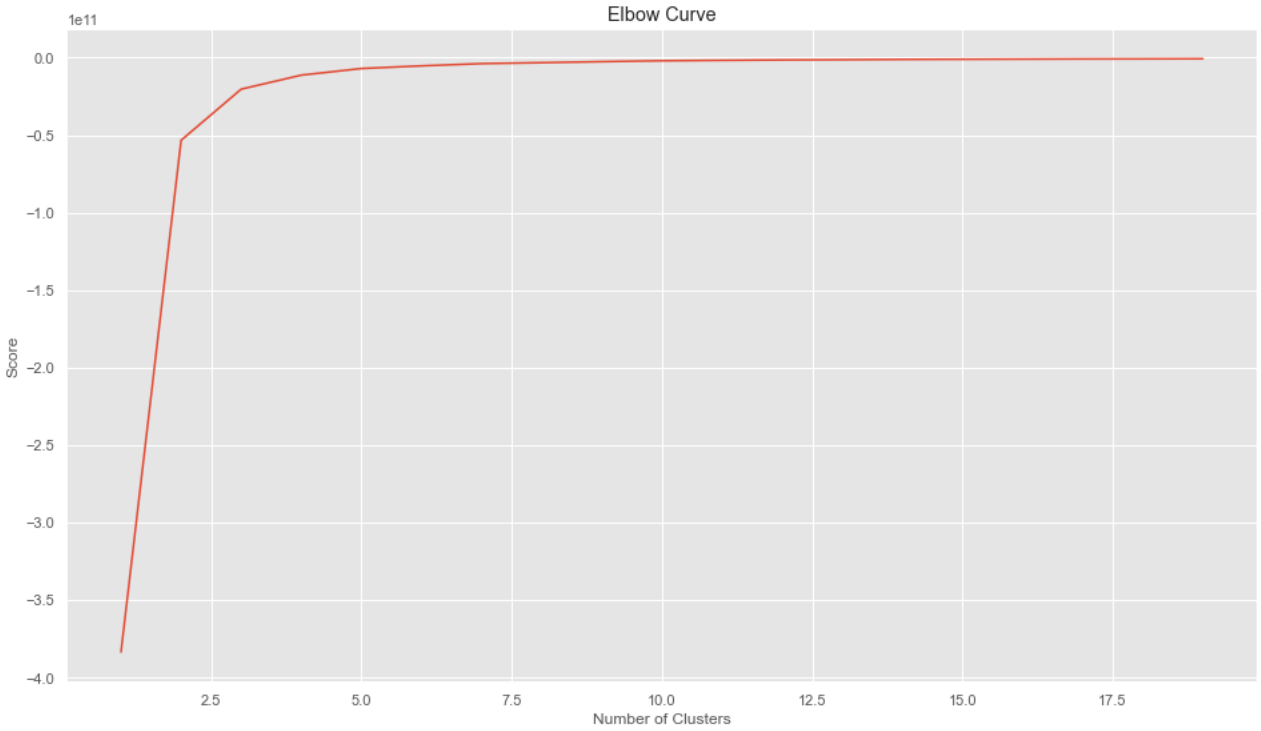
Out[74]: (421, 3)

In [88]:
```
fig = plt.figure()
ax = Axes3D(fig)
colores = ['blue', 'red', 'green','blue', 'cyan', 'yellow', 'orange', 'black', 'pink',
asignar = []
for row in labels:
    asignar.append(colores[row])
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c = asignar, s = 60);
```



Buscamos el valor K

In [75]:
```
Nc = range(1, 20)
kmeans = [KMeans(n_clusters = i) for i in Nc]
kmeans
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
score
plt.plot(Nc, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

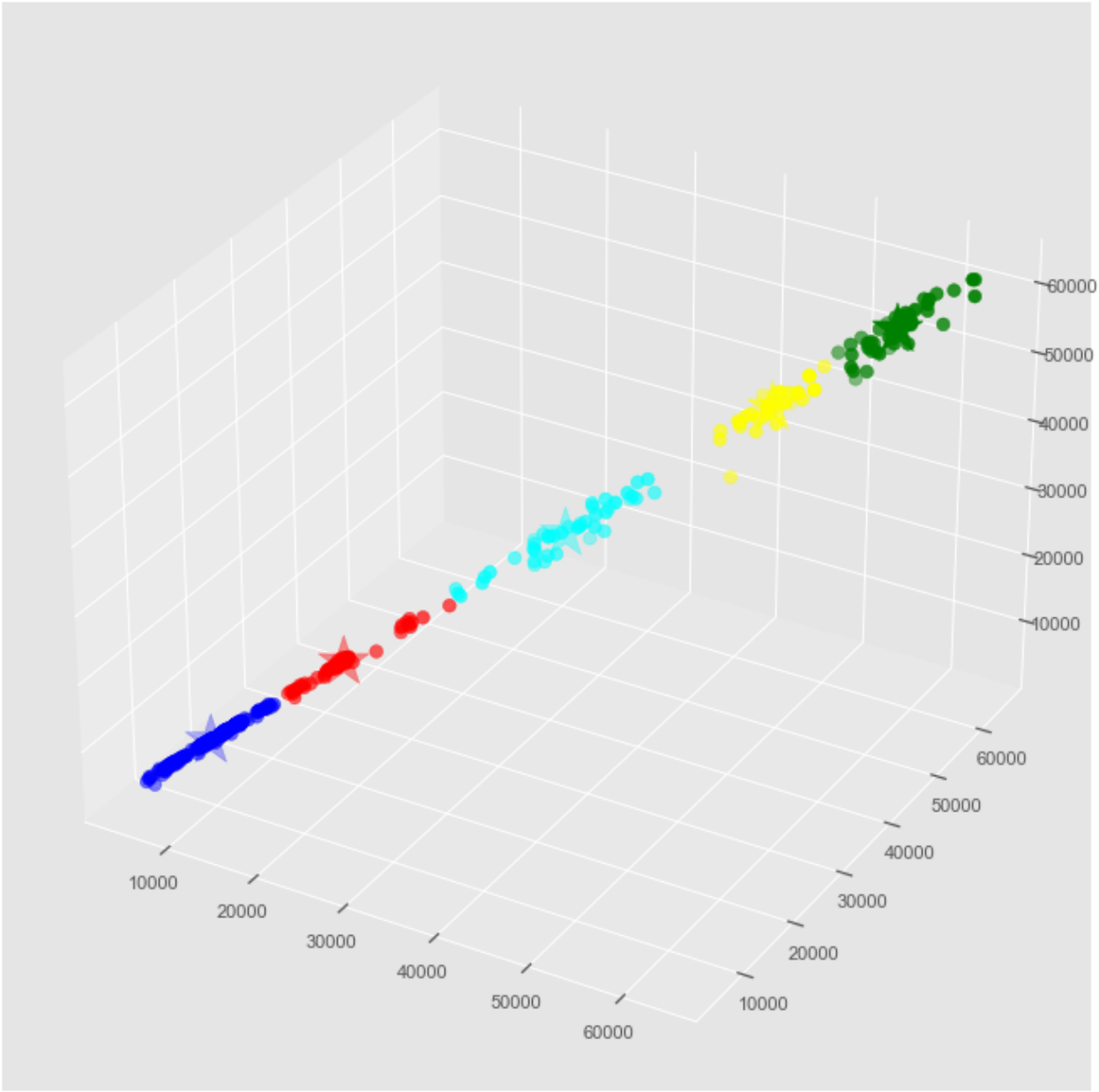


In [83]:
```
# Para el ejercicio, elijo 5 como un buen valor de K, pero podría ser otro.
kmeans = KMeans(n_clusters = 5).fit(X)
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[18985.90384615 19351.15192308 18280.81538462]
 [57660.5106383  59107.12978723 55776.62553191]
 [ 9646.05341365  9825.5686747   9415.36787149]
 [34317.38837209 35616.37906977 32374.81860465]
 [48816.53666667 50391.33       46695.24      ]]
```

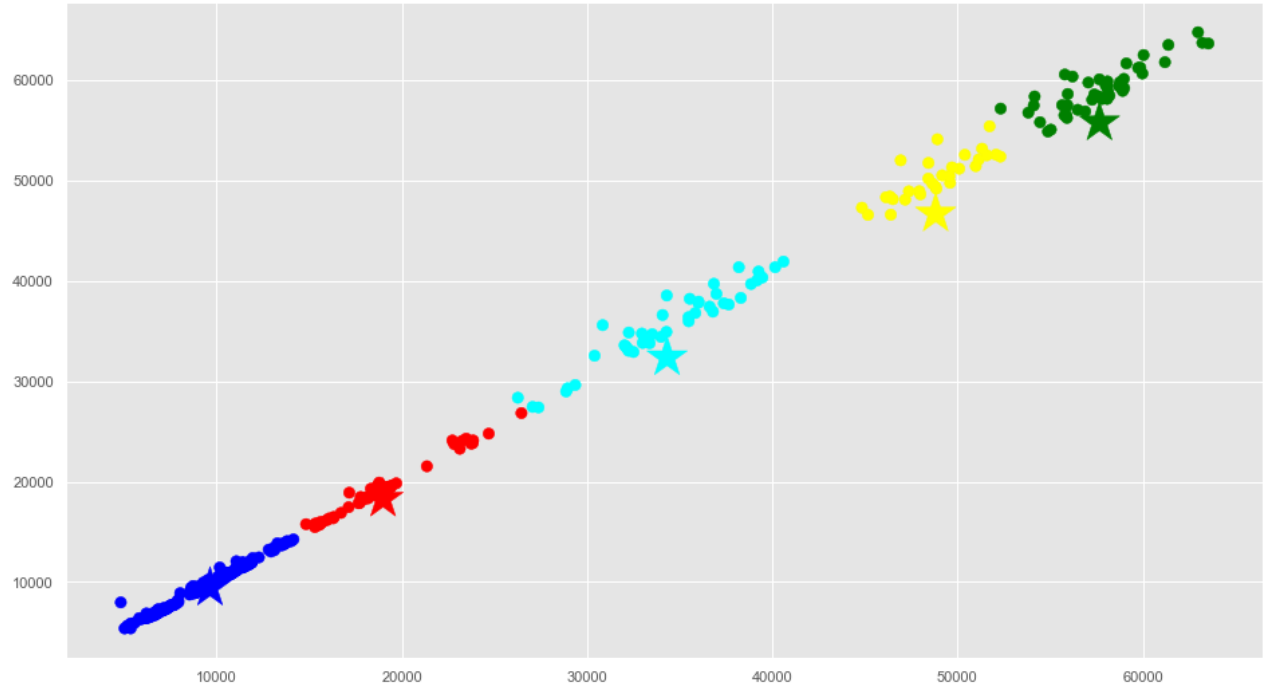
In [84]:
```
# Obtener las etiquetas de cada punto de nuestros datos
labels = kmeans.predict(X)
# Obtenemos los centroids
C = kmeans.cluster_centers_
colores = ['red', 'green', 'blue', 'cyan', 'yellow']
asignar = []
for row in labels:
    asignar.append(colores[row])

fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c = asignar, s = 60)
ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker = '*', c = colores, s = 1000);
```



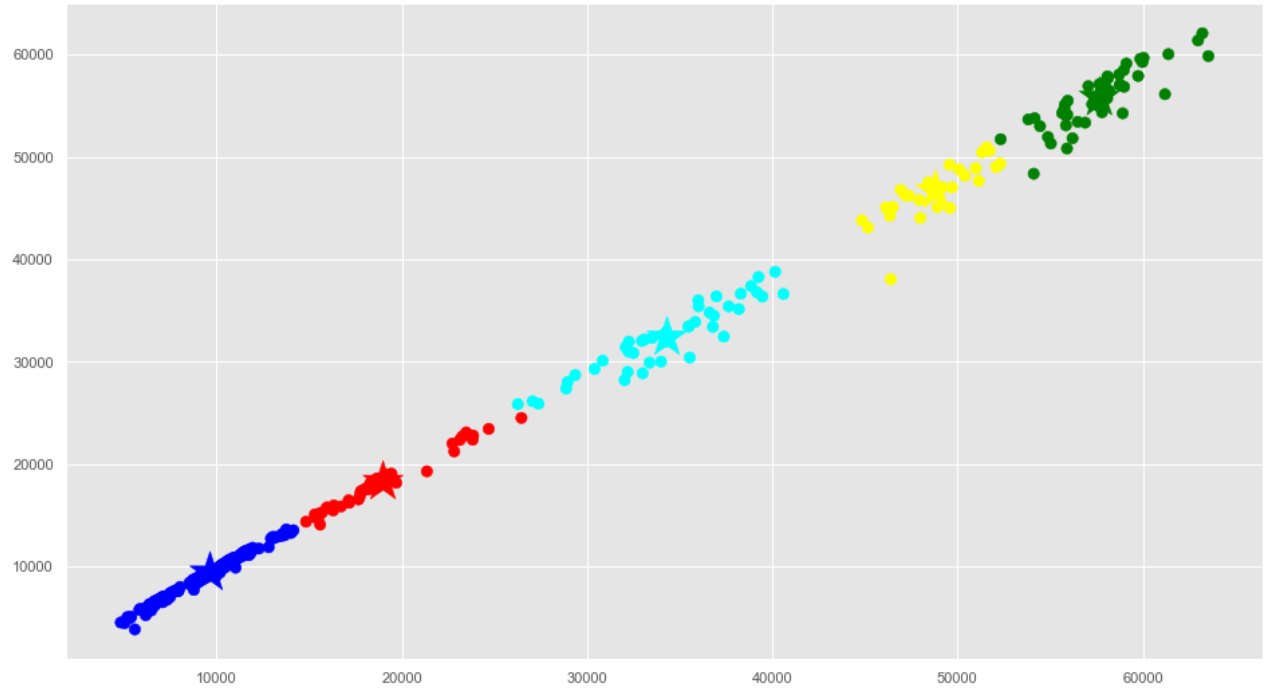
In [85]:
```
# Hacemos una proyección a 2D con los diversos ejes
f1 = dataframe['Price'].values
f2 = dataframe['High'].values

plt.scatter(f1, f2, c = asignar, s = 70)
plt.scatter(C[:, 0], C[:, 2], marker = '*', c = colores, s = 1000);
plt.show()
```



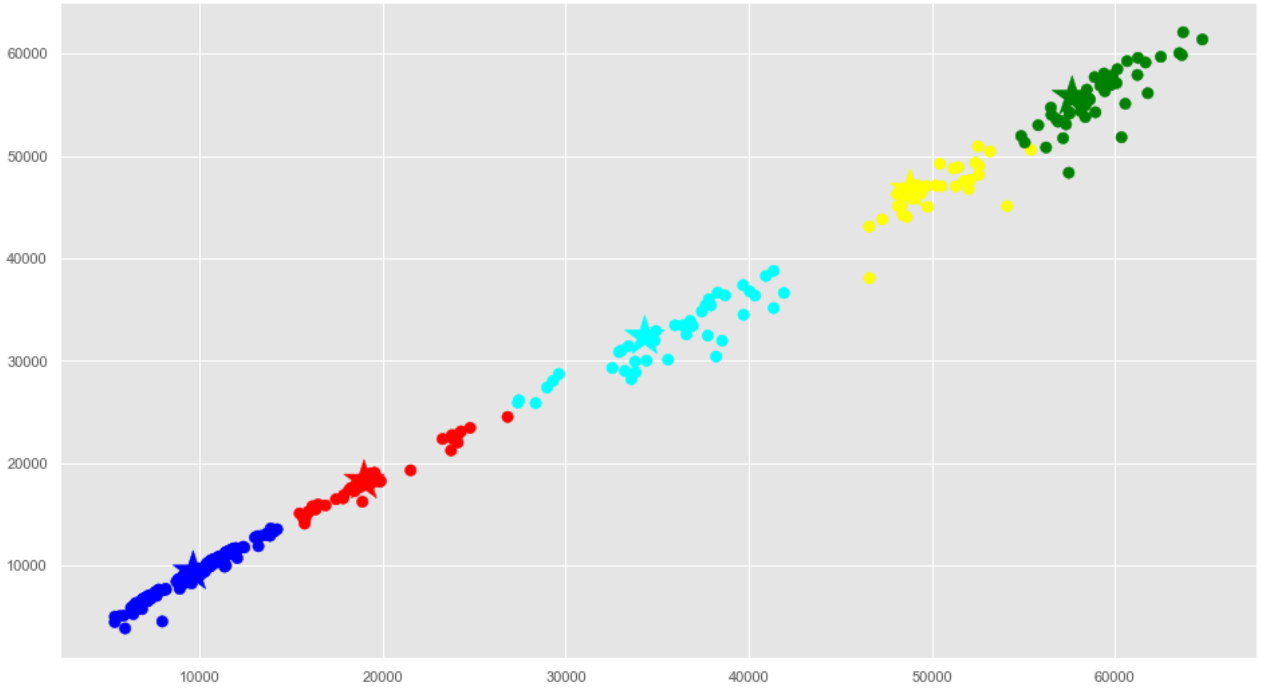
In [86]:
```
f1 = dataframe['Price'].values
f2 = dataframe['Low'].values

plt.scatter(f1, f2, c = asignar, s = 70)
plt.scatter(C[:, 0], C[:, 2], marker = '*', c = colores, s = 1000);
plt.show()
```



In [87]:
```
f1 = dataframe['High'].values
f2 = dataframe['Low'].values

plt.scatter(f1, f2, c = asignar, s = 70)
plt.scatter(C[:, 0], C[:, 2], marker = '*', c = colores, s = 1000);
plt.show()
```



Evaluando los resultados

In [89]:
```
print (classification_report(labels, labels));
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        52
           1       1.00      1.00      1.00        47
           2       1.00      1.00      1.00       249
           3       1.00      1.00      1.00        43
           4       1.00      1.00      1.00        30

    accuracy                           1.00       421
   macro avg       1.00      1.00      1.00       421
weighted avg       1.00      1.00      1.00       421
```