Women's Shoe Prices

In [169]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 5GB to the current directory (/kaggle/working/) that gets preserved as output when you create
 a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/womens-shoes-prices/7210_1.csv
/kaggle/input/womens-shoes-prices/Datafiniti_Womens_Shoes_Jun19.csv
/kaggle/input/womens-shoes-prices/Datafiniti_Womens_Shoes.csv
```

1. Importing Libraries and Packages

We will use these packages to help us manipulate the data and visualize the features/labels as well as measure how well our model performed.

In [175]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Plotting data
import seaborn as sns # Advanced visualization
import pandas as pd
import sqlite3


pd.set_option('display.max_columns', 999) # it helps to see all columns
```

2. Loading and Viewing Data Set

Before we begin, we should take a look at our data table to see the values that we'll be working with. We can use the head and describe function to look at some sample data and statistics.

In [176]:
```
# choose data
data = pd.read_csv('../input/womens-shoes-prices/Datafiniti_Womens_Shoes.csv')
```

In [177]:
```
#first 5 rows of dataset
data.head()
```

Out[177]:

	id	dateAdded	dateUpdated	asins	brand	categories	primaryCategories	colors	dimension	ean	
0	AVpfEf_hLJeJML431ueH	2015-05-04T12:13:08Z	2018-01-29T04:38:43Z	NaN	Naturalizer	Clothing,Shoes,Women's Shoes,All Women's Shoes...	Shoes	Silver,Cream Watercolor Floral	NaN	NaN	ht
1	AVpi74XfLJeJML43qZAc	2017-01-27T01:23:39Z	2018-01-03T05:21:54Z	NaN	MUK LUKS	Clothing,Shoes,Women's Shoes,Women's Casual Sh...	Shoes	Grey	NaN	3.397705e+10	htt
2	AVpi74XfLJeJML43qZAc	2017-01-27T01:23:39Z	2018-01-03T05:21:54Z	NaN	MUK LUKS	Clothing,Shoes,Women's Shoes,Women's Casual Sh...	Shoes	Grey	NaN	3.397705e+10	htt
3	AVpjXyCc1cnluZ0-V-Gj	2017-01-27T01:25:56Z	2018-01-04T11:52:35Z	NaN	MUK LUKS	Clothing,Shoes,Women's Shoes,All Women's Shoes...	Shoes,Shoes	Black	6.0 in x 6.0 in x 1.0 in	3.397705e+10	htt
4	AVphGKLPilAPnD_x1Nrm	2017-01-27T01:25:56Z	2018-01-18T03:55:18Z	NaN	MUK LUKS	Clothing,Shoes,Women's Shoes,All Women's Shoes...	Shoes	Grey	6.0 in x 6.0 in x 1.0 in	3.397705e+10	htt

In [178]:
```
#connect to a database
conn = sqlite3.connect("Shoes_database.db") #if the db does not exist, this creates a Any_Database_Name.db file in t
he current directory
#store your table in the database:
data.to_sql('Shoes_Price_Analysis', conn)
```

In [179]:
```
#read a SQL Query out of your database and into a pandas dataframe
sql_string ="""SELECT brand,
                        Count,
                        pr.MaxPrice/pr.Count as Price
                        FROM (SELECT
                                brand,
                                sum([prices.amountMax]) as maxPrice,
                                count(id) as Count
                                FROM Shoes_Price_Analysis
                                group by brand) as pr
                        where pr.Count >= 50
                                and pr.Count <=600
                                and brand is not null
                                and brand <> ""
                        order by pr.Count asc;"""

data = pd.read_sql(sql_string, conn)
data
```

Out[179]:

	brand	Count	Price
0	unionbay	55	47.990000
1	eastland	56	96.071429
2	dolce by mojo moxy	58	59.990000
3	lc lauren conrad	69	50.279855
4	adidas	74	65.462973
5	spring step	79	62.774810
6	madden nyc	84	50.466190
7	naturalsoul by naturalizer	107	71.485327
8	asics	111	77.737748
9	ryka	129	62.286124
10	Lifestride	136	59.990000
11	candies	144	57.351111
12	SKECHERS	151	66.314702
13	simply vera vera wang	164	68.648537
14	Nike	179	80.805587
15	apt. 9	183	52.694918
16	skechers	185	64.611622
17	sonoma goods for life	229	69.815328
18	new balance	255	69.159922
19	nike	285	67.929825
20	croft barrow	315	67.323333
21	dr. scholls	337	77.091988
22	so	422	56.265047
23	style charles by charles david	478	88.853556
24	easy street	556	54.045755

In [189]:
```
#is any row NULL
? data.isnull().any().any(), data.shape
```

```
Object ` data.isnull().any().any(), data.shape` not found.
```

In [190]:
```
data.info
```

Out[190]:
```
<bound method DataFrame.info of                              brand  Count      Price
0                         unionbay     55  47.990000
1                         eastland     56  96.071429
2               dolce by mojo moxy     58  59.990000
3                 lc lauren conrad     69  50.279855
4                           adidas     74  65.462973
5                      spring step     79  62.774810
6                       madden nyc     84  50.466190
7       naturalsoul by naturalizer    107  71.485327
8                            asics    111  77.737748
9                             ryka    129  62.286124
10                      Lifestride    136  59.990000
11                         candies    144  57.351111
12                        SKECHERS    151  66.314702
13           simply vera vera wang    164  68.648537
14                            Nike    179  80.805587
15                          apt. 9    183  52.694918
16                        skechers    185  64.611622
17           sonoma goods for life    229  69.815328
18                     new balance    255  69.159922
19                            nike    285  67.929825
20                    croft barrow    315  67.323333
21                     dr. scholls    337  77.091988
22                              so    422  56.265047
23  style charles by charles david    478  88.853556
24                     easy street    556  54.045755>
```

In [191]:
```
data.describe()
```

Out[191]:

	Count	Price
count	25.000000	25.000000
mean	193.640000	65.817827
std	137.224233	11.856632
min	55.000000	47.990000
25%	84.000000	57.351111
50%	151.000000	65.462973
75%	255.000000	69.815328
max	556.000000	96.071429

In [182]:
```
data.head()
```

Out[182]:

	brand	Count	Price
0	unionbay	55	47.990000
1	eastland	56	96.071429
2	dolce by mojo moxy	58	59.990000
3	lc lauren conrad	69	50.279855
4	adidas	74	65.462973

In [183]:
```
data.tail()
```

Out[183]:

	brand	Count	Price
20	croft barrow	315	67.323333
21	dr. scholls	337	77.091988
22	so	422	56.265047
23	style charles by charles david	478	88.853556
24	easy street	556	54.045755

In [184]:
```
data.corr()
```

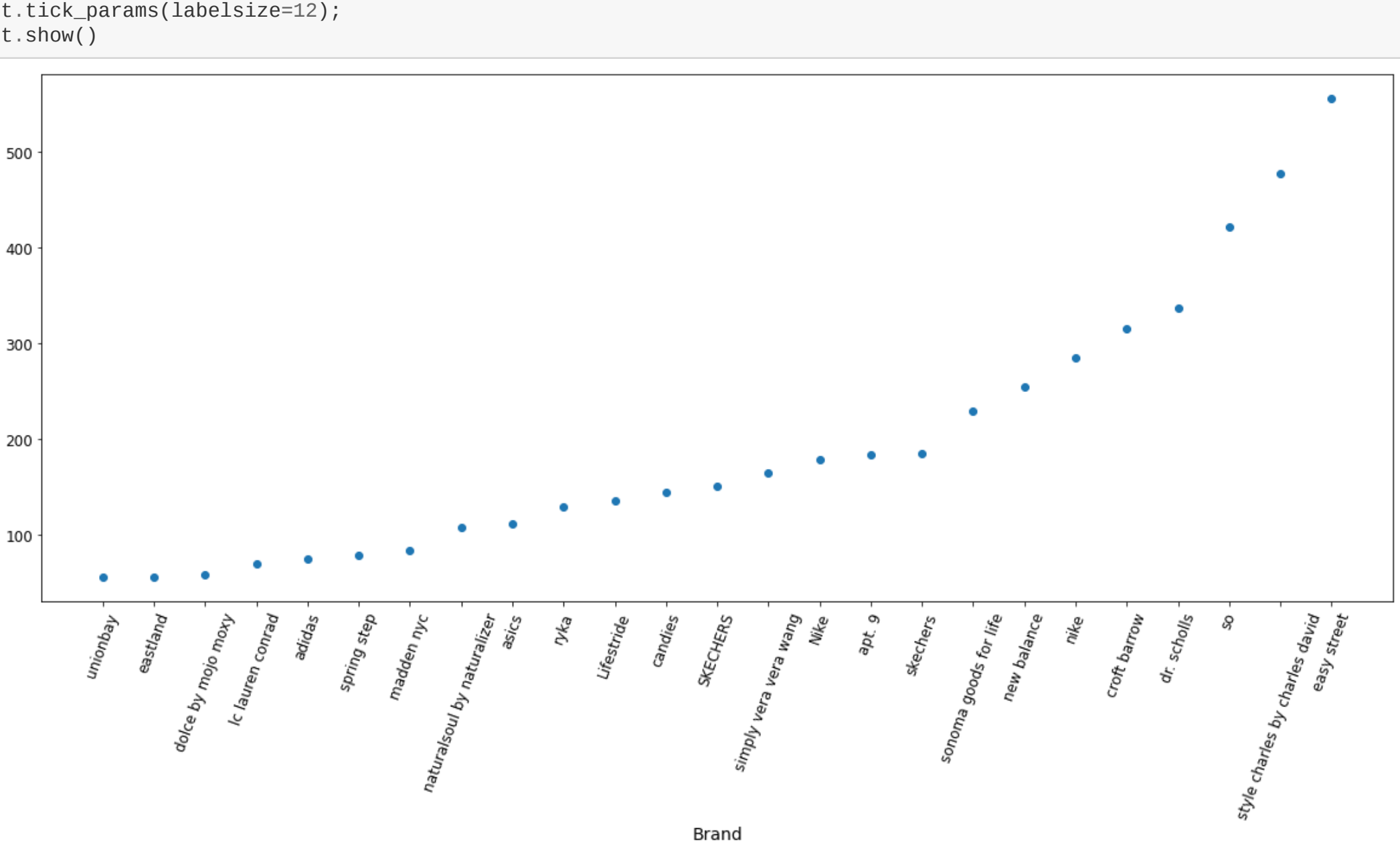
Out[184]:

	Count	Price
Count	1.000000	0.125093
Price	0.125093	1.000000

3. Plotting and Visualizing Data

Next we will start plotting. Firstly we will use matplotlib scatter to show trending What shoes brand is the most popular to buy women's shoes.

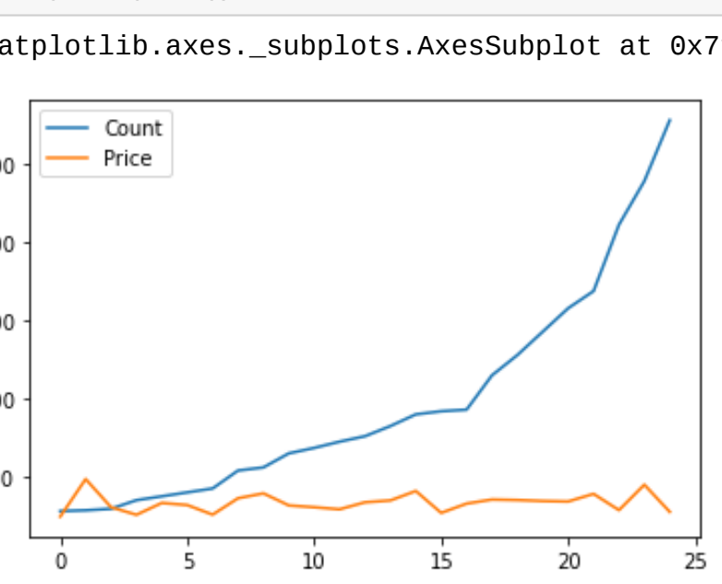
In [194]:
```
plt.figure(figsize=(20, 8))
plt.scatter(data.brand, data.Count)
plt.xlabel("Brand",fontsize = 14)
plt.xticks(rotation=70)
plt.ylabel("Count", fontsize = 14)
plt.tick_params(labelsize=12);
plt.show()
```



Second chart will show to us what relationships are between price and custumers.

In [195]:
```
data_plot = data.loc[:,["Count","Price"]]
data_plot.plot()
```

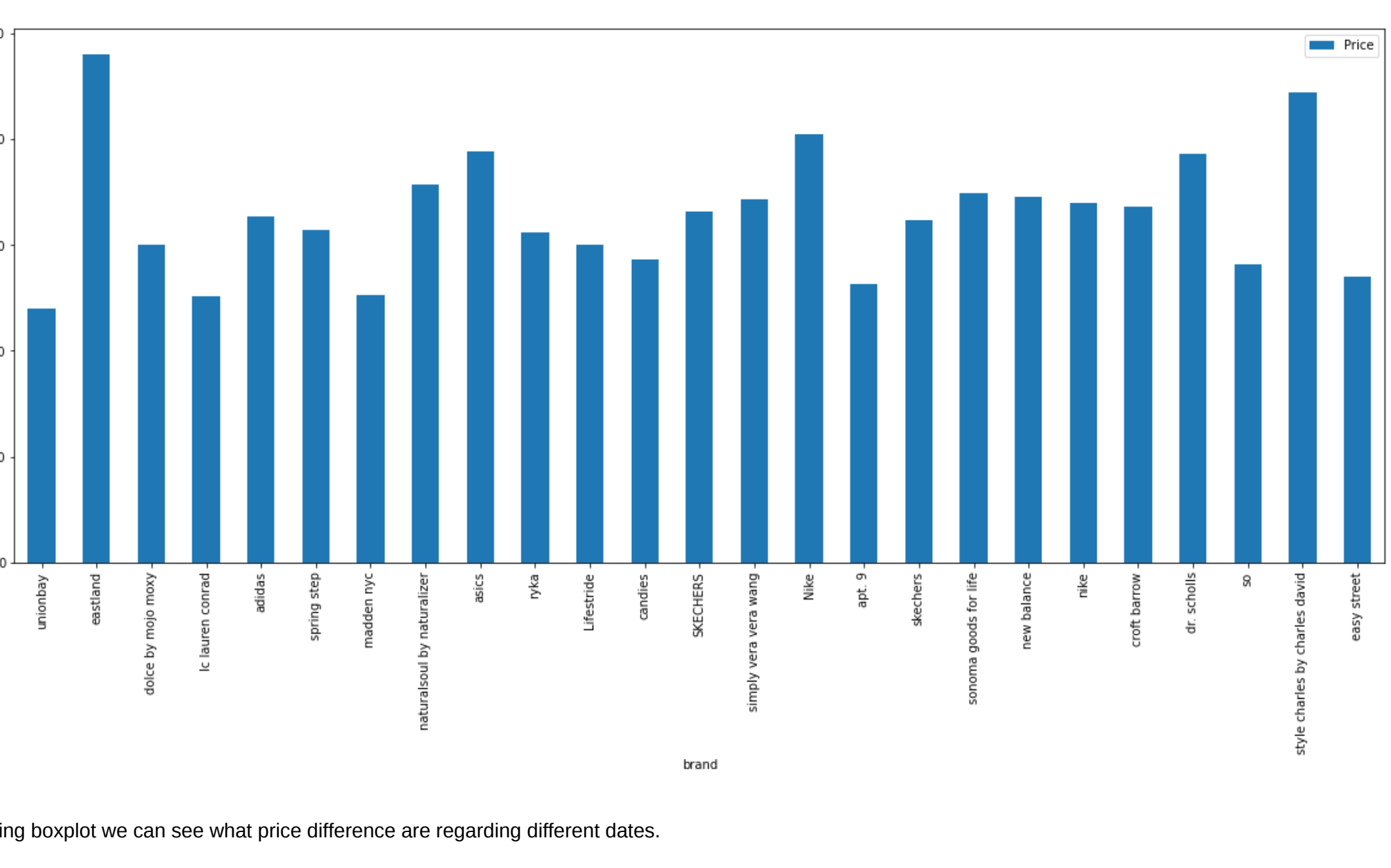
Out[195]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f25fdd99150>
```



Prices per each brand.

In [197]:
```
data.plot(kind = "bar",x = "brand", y = "Price",figsize=(20,8))
```

Out[197]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f25fdbd1950>
```

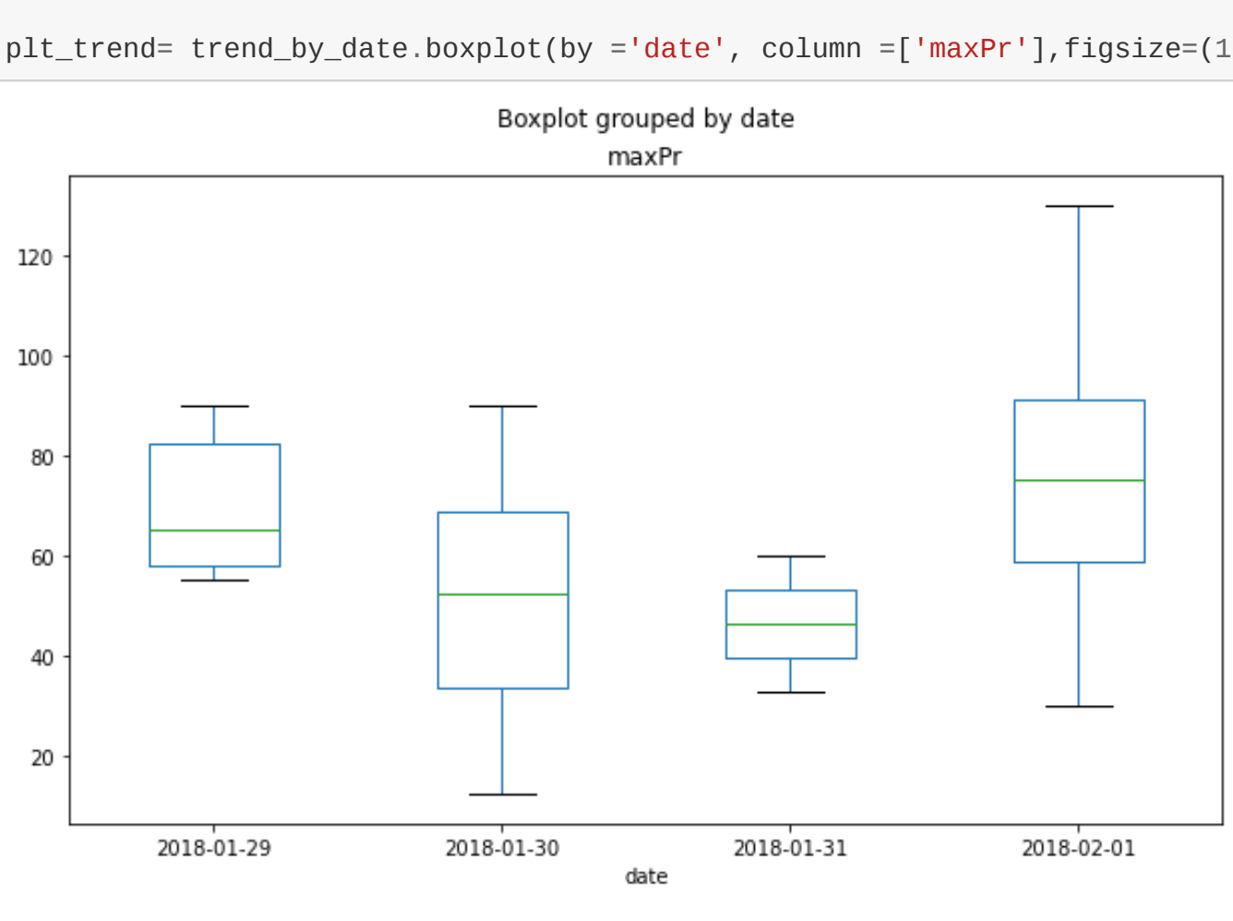


Using boxplot we can see what price difference are regarding different dates.

In [198]:
```
sql_trends = """SELECT *
                FROM (SELECT DISTINCT DATE([dateUpdated]) as date,
                        [prices.amountMax] as maxPr
                      FROM Shoes_Price_Analysis) as pr
                where date between "2018-01-29" and "2018-02-01";"""

trend_by_date = pd.read_sql(sql_trends, conn)
trend_by_date

plt_trend= trend_by_date.boxplot(by ='date', column =['maxPr'],figsize=(10,6), grid = False)
```



In [200]:
```
sql_prices = """SELECT price,
                    minprice
                FROM (SELECT DATE([dateUpdated]) as date,
                        sum([prices.amountMax]) as price,
                        sum([prices.amountMin]) as minprice
                      FROM Shoes_Price_Analysis
                      where date between "2018-01-29" and "2018-01-29"
                      group by dateUpdated) as pr;"""


prices = pd.read_sql(sql_prices, conn)
prices

f,ax = plt.subplots(figsize=(20, 10))
sns.heatmap(data2, annot=True, linewidths=0.5,linecolor="red", fmt= '.1f',ax=ax)
plt.show()
```

4. Model Fitting, Optimizing, and Predicting

Now that our data has been processed and formmated properly, and that we understand the general data we're working with as well as the trends and associations, we can start to build our model. We can import different classifiers from sklearn.

In [201]:
```
from sklearn.linear_model import LinearRegression
```

In [203]:
```
#our data
data
```

Out[203]:

	brand	Count	Price
0	unionbay	55	47.990000
1	eastland	56	96.071429
2	dolce by mojo moxy	58	59.990000
3	lc lauren conrad	69	50.279855
4	adidas	74	65.462973
5	spring step	79	62.774810
6	madden nyc	84	50.466190
7	naturalsoul by naturalizer	107	71.485327
8	asics	111	77.737748
9	ryka	129	62.286124
10	Lifestride	136	59.990000
11	candies	144	57.351111
12	SKECHERS	151	66.314702
13	simply vera vera wang	164	68.648537
14	Nike	179	80.805587
15	apt. 9	183	52.694918
16	skechers	185	64.611622
17	sonoma goods for life	229	69.815328
18	new balance	255	69.159922
19	nike	285	67.929825
20	croft barrow	315	67.323333
21	dr. scholls	337	77.091988
22	so	422	56.265047
23	style charles by charles david	478	88.853556
24	easy street	556	54.045755

In [204]:
```
linear_reg = LinearRegression()
x = data.Price.values.reshape(-1,1)
y = data.Count.values.reshape(-1,1)
```

In [205]:
```
linear_reg.fit(x,y)
```

Out[205]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

What the price will be then the brand will have 1000 customers?

In [206]:
```
next_price = linear_reg.predict([[1000]])
print(next_salary)
```

```
[[74.53330324]]
```