In [2]:
```
#install library
pip install pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in c:\users\girija\anaconda3\l
ib\site-packages (0.10.0)
Requirement already satisfied: requests>=2.19.0 in c:\users\girija\anaconda3\li
b\site-packages (from pandas-datareader) (2.26.0)
Requirement already satisfied: lxml in c:\users\girija\anaconda3\lib\site-packa
ges (from pandas-datareader) (4.6.4)
Requirement already satisfied: pandas>=0.23 in c:\users\girija\anaconda3\lib\si
te-packages (from pandas-datareader) (1.1.5)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\girija\anacon
da3\lib\site-packages (from pandas>=0.23->pandas-datareader) (2.8.2)
Requirement already satisfied: numpy>=1.15.4 in c:\users\girija\anaconda3\lib\s
ite-packages (from pandas>=0.23->pandas-datareader) (1.20.3)
Requirement already satisfied: pytz>=2017.2 in c:\users\girija\anaconda3\lib\si
te-packages (from pandas>=0.23->pandas-datareader) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\girija\anaconda3\lib\site-p
ackages (from python-dateutil>=2.7.3->pandas>=0.23->pandas-datareader) (1.16.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\girija\anaconda3
\lib\site-packages (from requests>=2.19.0->pandas-datareader) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\girija\ana
conda3\lib\site-packages (from requests>=2.19.0->pandas-datareader) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\girija\anaconda3\lib\si
te-packages (from requests>=2.19.0->pandas-datareader) (3.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\girija\anacond
a3\lib\site-packages (from requests>=2.19.0->pandas-datareader) (1.26.7)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
#import libraries for reading stock data from yahoo
import pandas_datareader.data as web
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
#set up start and end time for data grab
start = datetime.datetime(2020,1,1)
end = datetime.datetime(2022,7,16)
```

In [4]:
```
#Fetching the data
ferrari = web.DataReader("RACE",'yahoo',start,end)
benz = web.DataReader("DDAIF",'yahoo',start,end)
```

In [5]:
```
#saving the data in csv format
ferrari.to_csv("Ferrari_stock.csv")
benz.to_csv("Benz_stock.csv")
```

In [6]:
```
ferrari.head(10)
```

Out[6]:

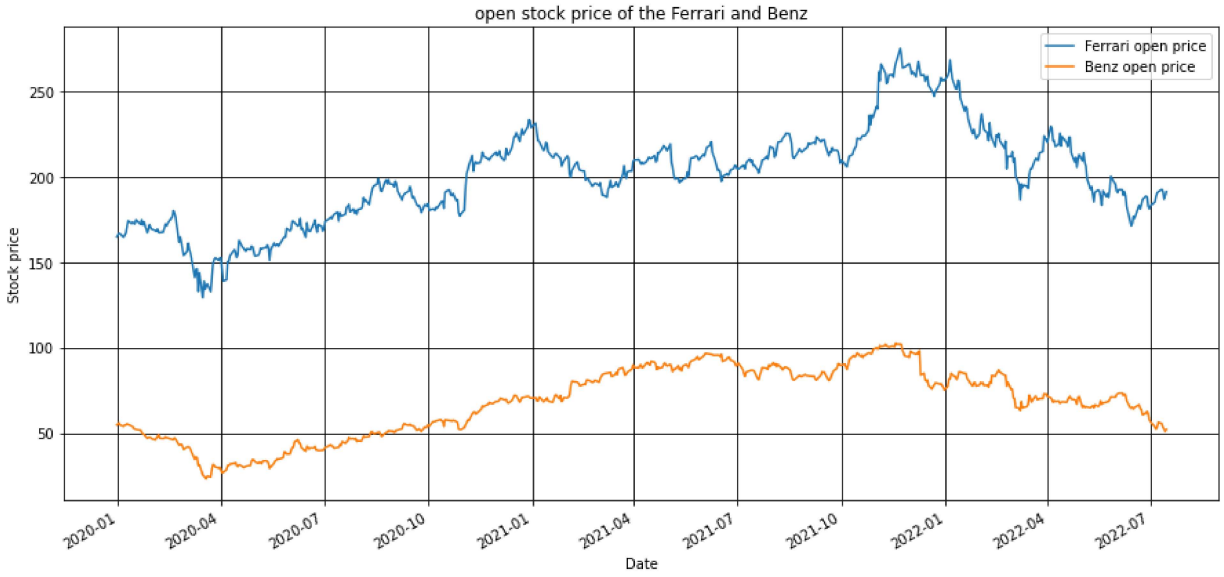
Date	High	Low	Open	Close	Volume	Adj Close
2019-12-31	165.839996	164.240005	165.000000	165.539993	128200.0	162.384796
2020-01-02	168.774002	166.630005	167.050003	168.509995	195800.0	165.298203
2020-01-03	167.750000	166.268005	166.759995	166.729996	134600.0	163.552109
2020-01-06	166.289993	164.759995	164.759995	165.850006	169900.0	162.688889
2020-01-07	166.119995	165.179993	165.979996	165.220001	309300.0	162.070892
2020-01-08	168.970001	166.750000	167.039993	168.539993	261400.0	165.327606
2020-01-09	171.000000	169.000000	170.759995	169.910004	283500.0	166.671509
2020-01-10	174.520004	170.330002	174.419998	171.309998	602200.0	168.044815
2020-01-13	174.289993	172.160004	172.710007	174.289993	375600.0	170.968002
2020-01-14	174.074997	171.199997	173.669998	171.550003	483700.0	168.280243

In [7]:
```
benz.head(10)
```

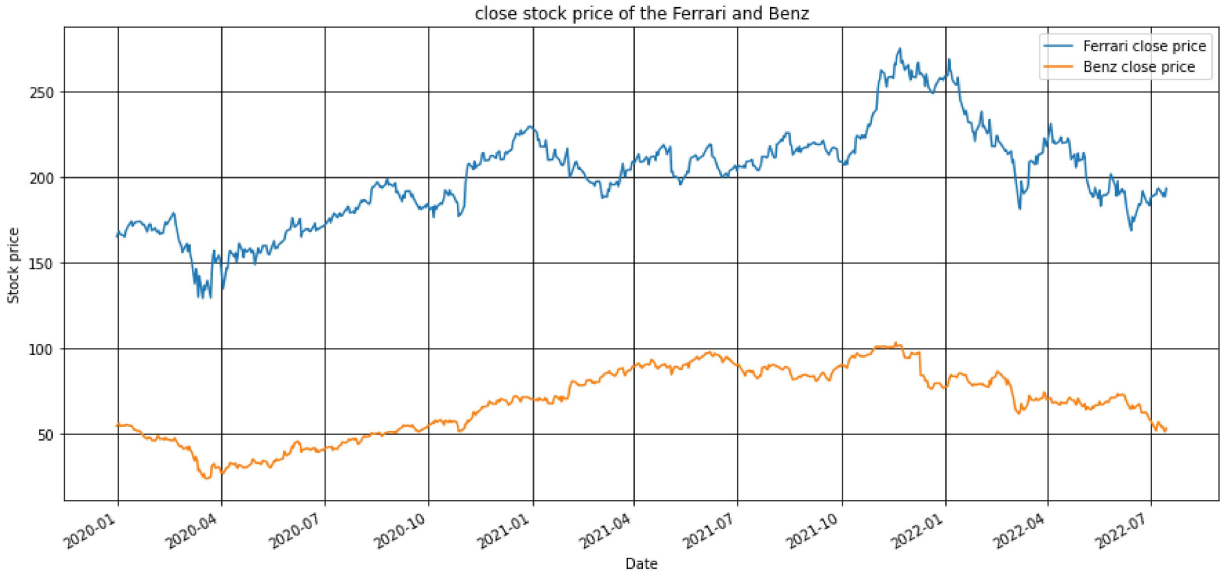
Out[7]:

Date	High	Low	Open	Close	Volume	Adj Close
2019-12-31	54.700001	54.349998	54.700001	54.480000	34700.0	48.244125
2020-01-02	56.169998	55.770000	55.939999	56.090000	61100.0	49.669842
2020-01-03	54.950001	54.500000	54.680000	54.500000	58700.0	48.261837
2020-01-06	54.779999	53.820000	53.840000	54.730000	60900.0	48.465508
2020-01-07	54.849998	54.599998	54.730000	54.599998	15800.0	48.350388
2020-01-08	55.360001	54.689999	54.689999	55.119999	18500.0	48.810871
2020-01-09	55.380001	54.990002	55.380001	55.310001	22300.0	48.979126
2020-01-10	55.130001	54.799999	55.090000	54.799999	33800.0	48.527500
2020-01-13	54.630001	54.009998	54.029999	54.610001	21700.0	48.359245
2020-01-14	54.040001	53.680000	53.810001	53.930000	38700.0	47.757080

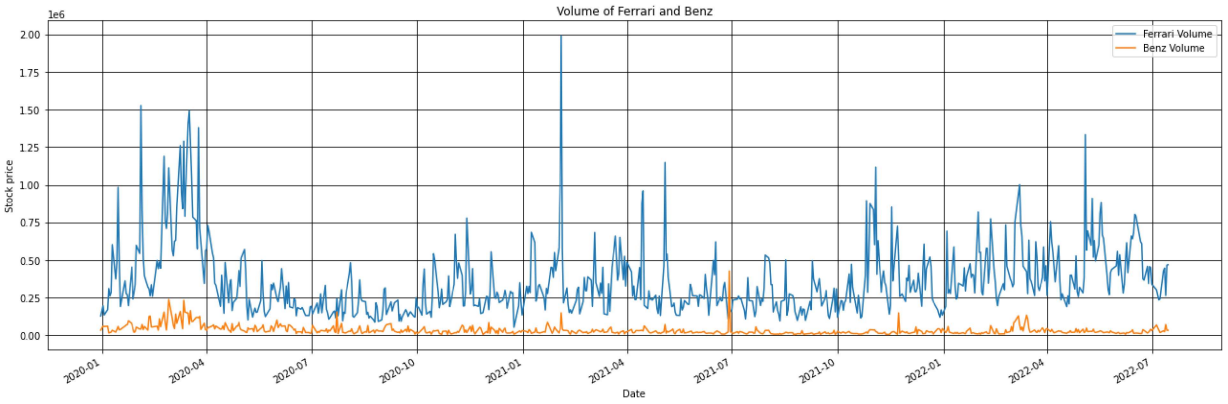
In [8]:
```
ferrari["Open"].plot(label="Ferrari open price", figsize=(15,7))
benz["Open"].plot(label="Benz open price")
plt.title('open stock price of the Ferrari and Benz')
plt.legend()
plt.xlabel("Date")
plt.ylabel("Stock price")
plt.grid(which="major", color='k',linestyle='-')
plt.show()
```



In [9]:
```
ferrari["Close"].plot(label="Ferrari close price", figsize=(15,7))
benz["Close"].plot(label="Benz close price")
plt.title('close stock price of the Ferrari and Benz')
plt.legend()
plt.xlabel("Date")
plt.ylabel("Stock price")
plt.grid(which="major", color='k',linestyle='-')
plt.show()
```



In [10]:
```
ferrari["Volume"].plot(label="Ferrari Volume", figsize=(22,7))
benz["Volume"].plot(label="Benz Volume")
plt.title('Volume of Ferrari and Benz')
plt.legend()
plt.xlabel("Date")
plt.ylabel("Stock price")
plt.grid(which="major", color='k',linestyle='-')
plt.show()
```



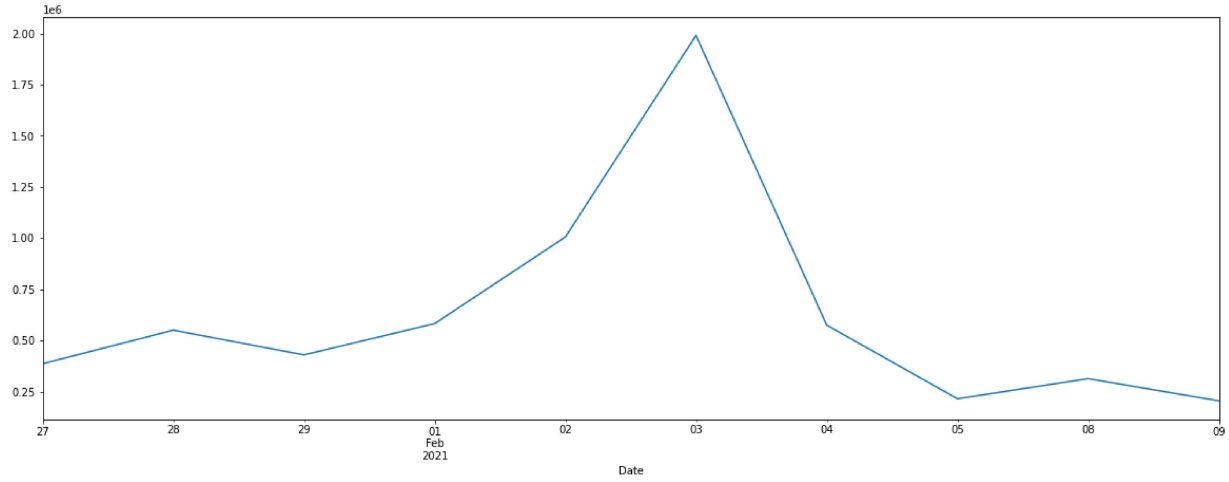
In [11]:
```
ferrari['Open'].plot(label='Open',figsize=(20,7))
ferrari['High'].plot(label='High')
ferrari['Low'].plot(label='Low')
ferrari['Close'].plot(label='Close')
plt.legend()
plt.grid(which="major", color='k',linestyle='-')
plt.show()
```

In [12]:
```
ferrari['Volume'].argmax()
```

Out[12]: 275

In [13]:
```
ferrari['Volume'].iloc[270:280].plot(figsize=(20,7))
```

Out[13]: <AxesSubplot:xlabel='Date'>



In [14]:
```
ferrari['Total Traded']=ferrari['Open']*ferrari['Volume']
benz['Total Traded']=benz['Open']*benz['Volume']
```

In [15]:
```
ferrari.head()
```

Out[15]:

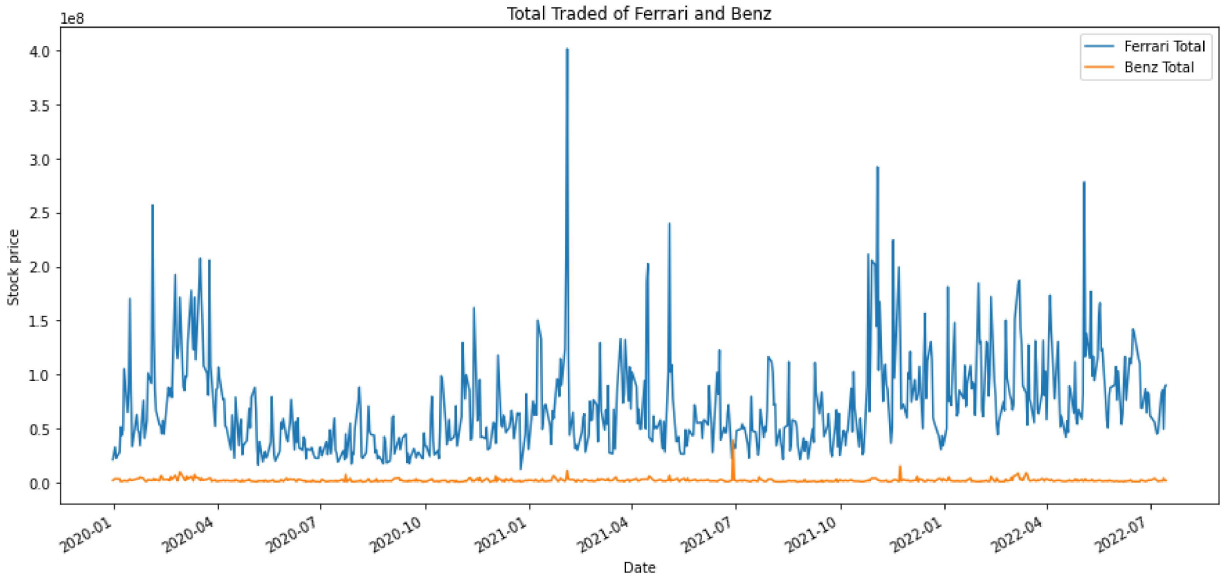
Date	High	Low	Open	Close	Volume	Adj Close	Total Traded
2019-12-31	165.839996	164.240005	165.000000	165.539993	128200.0	162.384796	2.115300e+07
2020-01-02	168.774002	166.630005	167.050003	168.509995	195800.0	165.298203	3.270839e+07
2020-01-03	167.750000	166.268005	166.759995	166.729996	134600.0	163.552109	2.244590e+07
2020-01-06	166.289993	164.759995	164.759995	165.850006	169900.0	162.688889	2.799272e+07
2020-01-07	166.119995	165.179993	165.979996	165.220001	309300.0	162.070892	5.133761e+07

In [16]:
```
benz.head()
```

Out[16]:

Date	High	Low	Open	Close	Volume	Adj Close	Total Traded
2019-12-31	54.700001	54.349998	54.700001	54.480000	34700.0	48.244125	1.898090e+06
2020-01-02	56.169998	55.770000	55.939999	56.090000	61100.0	49.669842	3.417934e+06
2020-01-03	54.950001	54.500000	54.680000	54.500000	58700.0	48.261837	3.209716e+06
2020-01-06	54.779999	53.820000	53.840000	54.730000	60900.0	48.465508	3.278856e+06
2020-01-07	54.849998	54.599998	54.730000	54.599998	15800.0	48.350388	8.647340e+05

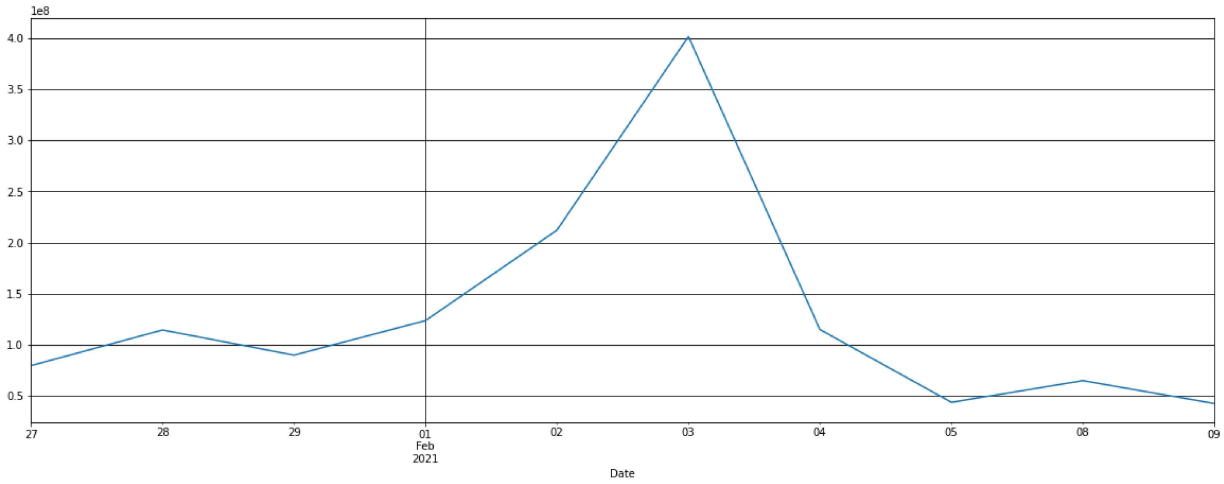
In [17]:
```
ferrari["Total Traded"].plot(label="Ferrari Total", figsize=(15,7))
benz["Total Traded"].plot(label="Benz Total")
plt.title('Total Traded of Ferrari and Benz')
plt.legend()
plt.xlabel("Date")
plt.ylabel("Stock price")

plt.show()
```



In [18]:
```
ferrari['Total Traded'].argmax()
```

Out[18]: 275

In [19]:
```
ferrari['Total Traded'].iloc[270:280].plot(figsize=(20,7))
plt.grid(which="major", color='k',linestyle='-')
```

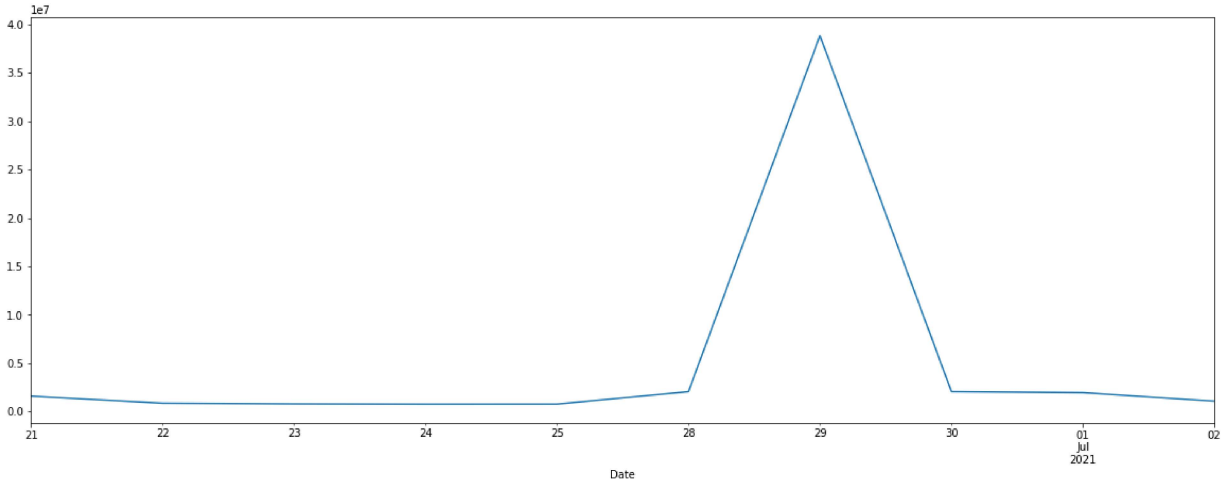


In [20]:
```
benz['Total Traded'].argmax()
```

Out[20]: 376

In [21]:
```
benz['Total Traded'].iloc[370:380].plot(figsize=(20,7))
```

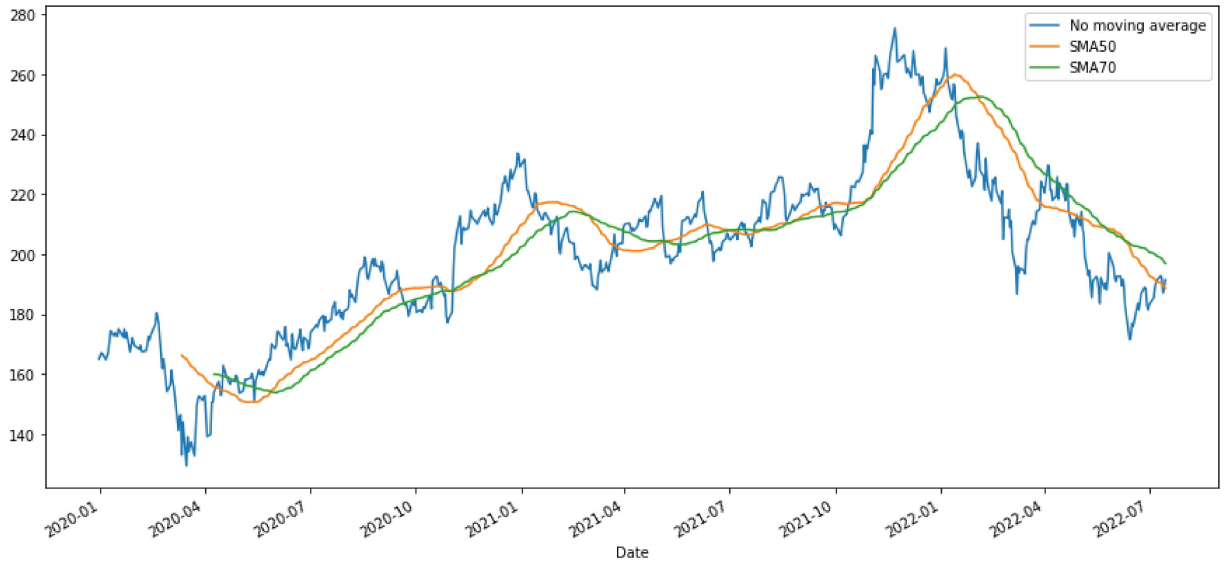
Out[21]: <AxesSubplot:xlabel='Date'>



Simple Moving Average of Ferrari

In [22]:
```
ferrari['Open'].plot(label='No moving average',figsize=(15,7))
ferrari['SMA50']=ferrari['Open'].rolling(50).mean()
ferrari['SMA50'].plot(label='SMA50')
ferrari['SMA70']=ferrari['Open'].rolling(70).mean()
ferrari['SMA70'].plot(label='SMA70')
plt.legend()
```

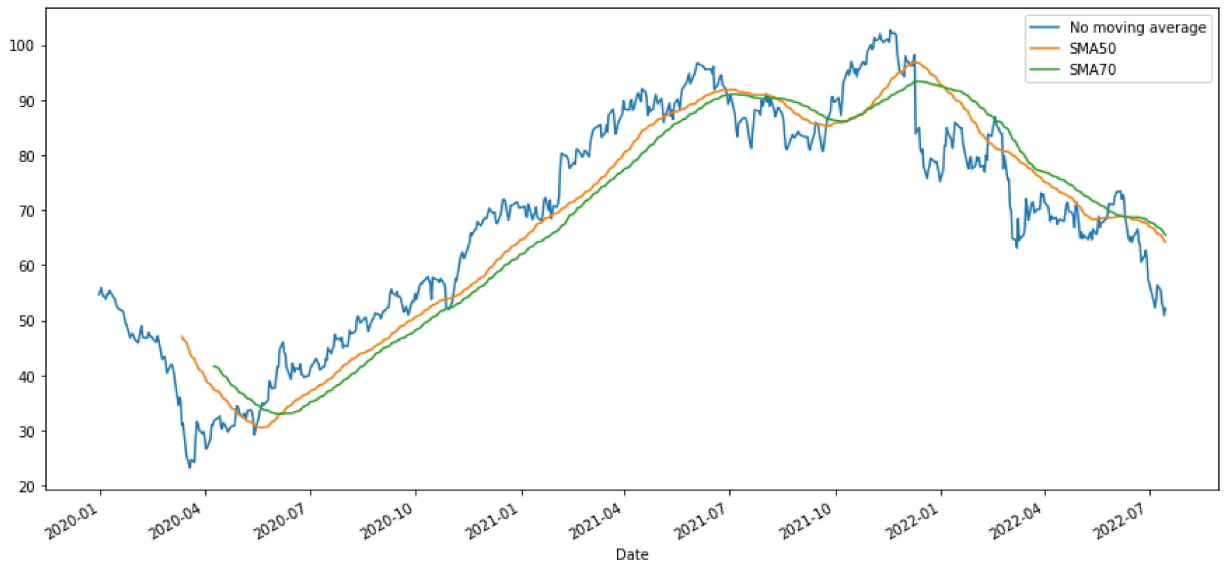
Out[22]:
```
<matplotlib.legend.Legend at 0x1b784743340>
```



Simple Moving Average of Benz

In [23]:
```
benz['Open'].plot(label='No moving average',figsize=(15,7))
benz['SMA50']=benz['Open'].rolling(50).mean()
benz['SMA50'].plot(label='SMA50')
benz['SMA70']=benz['Open'].rolling(70).mean()
benz['SMA70'].plot(label='SMA70')
plt.legend()
```

Out[23]:
```
<matplotlib.legend.Legend at 0x1b7851eb340>
```



In [ ]:
```
```

In [24]:
```
ferrari['Open'].plot(label='No moving average',figsize=(15,7))
ferrari['SMA50']=ferrari['Open'].ewm(span=5, adjust=False).mean()
ferrari['SMA50'].plot(label='SMA50')
ferrari['SMA70']=ferrari['Open'].ewm(span=10, adjust=False).mean()
ferrari['SMA70'].plot(label='SMA70')
plt.legend()
```

Out[24]:
```
<matplotlib.legend.Legend at 0x1b784411a00>
```

Checking correlation between Ferrari and Benz

In [25]:
```
from pandas.plotting import scatter_matrix
```

In [26]:
```
car_company = pd.concat([ferrari['Open'],benz['Open']], axis=1)

car_company.columns = ['Ferrari Open','Benz Open']
```

In [27]:
```
scatter_matrix(car_company, figsize=(20,7), hist_kwds={'bins':100})
```

```
C:\Users\Girija\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.p
y:331: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed
two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```

Out[27]:
```
array([[<AxesSubplot:xlabel='Ferrari Open', ylabel='Ferrari Open'>,
        <AxesSubplot:xlabel='Benz Open', ylabel='Ferrari Open'>],
       [<AxesSubplot:xlabel='Ferrari Open', ylabel='Benz Open'>,
        <AxesSubplot:xlabel='Benz Open', ylabel='Benz Open'>]],
      dtype=object)
```