In [1]:

```
#Séries Temporais
#Covid-19
#FiqueEmCasa
#Aula: Especial 04
#Data: 11/04/2020
#Professor: Victor Venites
```

In [2]:

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

In [3]:

```
!dir
```

```
 O volume na unidade C não tem nome.
 O Número de Série do Volume é 3A4A-84B2

 Pasta de C:\Users\thiag\Documents\Aulas-Python\School of AI\Ano de 2020\a
ula 4 especial\COVID-19-master\csse_covid_19_data\csse_covid_19_time_serie
s

13/04/2020  14:33    <DIR>          .
13/04/2020  14:33    <DIR>          ..
10/04/2020  20:55                 9 .gitignore
13/04/2020  14:33    <DIR>          .ipynb_checkpoints
10/04/2020  20:55               668 README.md
13/04/2020  14:33               831 Recuperados-Covid19.ipynb
11/04/2020  22:09           421.527 Serie Temporal_Covid-19.ipynb
10/04/2020  20:55            65.331 time_series_covid19_confirmed_global.c
sv
10/04/2020  20:55           859.468 time_series_covid19_confirmed_US.csv
10/04/2020  20:55            52.985 time_series_covid19_deaths_global.csv
10/04/2020  20:55           859.093 time_series_covid19_deaths_US.csv
10/04/2020  20:55            55.120 time_series_covid19_recovered_global.c
sv
13/04/2020  14:32               831 Untitled.ipynb
              10 arquivo(s)      2.315.863 bytes
               3 pasta(s)   65.764.401.152 bytes disponíveis
```

In [4]:

```
pwd
```

Out[4]:

```
'C:\\Users\\thiag\\Documents\\Aulas-Python\\School of AI\\Ano de 2020\\aul
a 4 especial\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_seri
es'
```

In [5]:

```
DataFrame = pd.read_csv("time_series_covid19_recovered_global.csv")
DataFrame.head()
```

Out[5]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/2
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	
1	NaN	Albania	41.1533	20.1683	0	0	0	0	
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	

5 rows × 84 columns

In [6]:

```
DataFrame.tail()
```

Out[6]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20
245	Saint Pierre and Miquelon	France	46.885200	-56.315900	0	0	0	0
246	NaN	South Sudan	6.877000	31.307000	0	0	0	0
247	NaN	Western Sahara	24.215500	-12.885800	0	0	0	0
248	NaN	Sao Tome and Principe	0.186360	6.613081	0	0	0	0
249	NaN	Yemen	15.552727	48.516388	0	0	0	0

5 rows × 84 columns

In [7]:

```
DataFrame.iloc[:3,:3]
```

Out[7]:

	Province/State	Country/Region	Lat
0	NaN	Afghanistan	33.0000
1	NaN	Albania	41.1533
2	NaN	Algeria	28.0339

In [8]:

```
DataFrame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 84 columns):
Province/State    67 non-null object
Country/Region    250 non-null object
Lat               250 non-null float64
Long              250 non-null float64
1/22/20           250 non-null int64
1/23/20           250 non-null int64
1/24/20           250 non-null int64
1/25/20           250 non-null int64
1/26/20           250 non-null int64
1/27/20           250 non-null int64
1/28/20           250 non-null int64
1/29/20           250 non-null int64
1/30/20           250 non-null int64
1/31/20           250 non-null int64
2/1/20            250 non-null int64
2/2/20            250 non-null int64
2/3/20            250 non-null int64
2/4/20            250 non-null int64
2/5/20            250 non-null int64
2/6/20            250 non-null int64
2/7/20            250 non-null int64
2/8/20            250 non-null int64
2/9/20            250 non-null int64
2/10/20           250 non-null int64
2/11/20           250 non-null int64
2/12/20           250 non-null int64
2/13/20           250 non-null int64
2/14/20           250 non-null int64
2/15/20           250 non-null int64
2/16/20           250 non-null int64
2/17/20           250 non-null int64
2/18/20           250 non-null int64
2/19/20           250 non-null int64
2/20/20           250 non-null int64
2/21/20           250 non-null int64
2/22/20           250 non-null int64
2/23/20           250 non-null int64
2/24/20           250 non-null int64
2/25/20           250 non-null int64
2/26/20           250 non-null int64
2/27/20           250 non-null int64
2/28/20           250 non-null int64
2/29/20           250 non-null int64
3/1/20            250 non-null int64
3/2/20            250 non-null int64
3/3/20            250 non-null int64
3/4/20            250 non-null int64
3/5/20            250 non-null int64
3/6/20            250 non-null int64
3/7/20            250 non-null int64
3/8/20            250 non-null int64
3/9/20            250 non-null int64
3/10/20           250 non-null int64
3/11/20           250 non-null int64
3/12/20           250 non-null int64
3/13/20           250 non-null int64
3/14/20           250 non-null int64
3/15/20           250 non-null int64
```

```
3/16/20          250 non-null int64
3/17/20          250 non-null int64
3/18/20          250 non-null int64
3/19/20          250 non-null int64
3/20/20          250 non-null int64
3/21/20          250 non-null int64
3/22/20          250 non-null int64
3/23/20          250 non-null int64
3/24/20          250 non-null int64
3/25/20          250 non-null int64
3/26/20          250 non-null int64
3/27/20          250 non-null int64
3/28/20          250 non-null int64
3/29/20          250 non-null int64
3/30/20          250 non-null int64
3/31/20          250 non-null int64
4/1/20           250 non-null int64
4/2/20           250 non-null int64
4/3/20           250 non-null int64
4/4/20           250 non-null int64
4/5/20           250 non-null int64
4/6/20           250 non-null int64
4/7/20           250 non-null int64
4/8/20           250 non-null int64
4/9/20           250 non-null int64
4/10/20          250 non-null int64
dtypes: float64(2), int64(80), object(2)
memory usage: 164.1+ KB
```

In [9]:

```
DataFrame.describe()
```

Out[9]:

	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1
count	250.000000	250.000000	250.000000	250.000000	250.0000	250.00000	250.00000	250.
mean	20.048575	27.934869	0.112000	0.120000	0.1440	0.15600	0.20800	0.
std	24.394560	67.432156	1.770875	1.774881	1.9664	2.03278	2.66555	2.
min	-51.796300	-106.346800	0.000000	0.000000	0.0000	0.00000	0.00000	0.
25%	6.677575	-8.091400	0.000000	0.000000	0.0000	0.00000	0.00000	0.
50%	21.805100	22.380900	0.000000	0.000000	0.0000	0.00000	0.00000	0.
75%	39.376275	87.379325	0.000000	0.000000	0.0000	0.00000	0.00000	0.
max	71.706900	178.065000	28.000000	28.000000	31.0000	32.00000	42.00000	45.

8 rows × 82 columns

In [10]:

```
DataFrame.sum()
```

Out[10]:

```
Lat          5012.143754
Long         6983.717345
1/22/20        28.000000
1/23/20        30.000000
1/24/20        36.000000
1/25/20        39.000000
1/26/20        52.000000
1/27/20        61.000000
1/28/20       107.000000
1/29/20       126.000000
1/30/20       143.000000
1/31/20       222.000000
2/1/20        284.000000
2/2/20        472.000000
2/3/20        623.000000
2/4/20        852.000000
2/5/20       1124.000000
2/6/20       1487.000000
2/7/20       2011.000000
2/8/20       2616.000000
2/9/20       3244.000000
2/10/20      3946.000000
2/11/20      4683.000000
2/12/20      5150.000000
2/13/20      6295.000000
2/14/20      8058.000000
2/15/20      9395.000000
2/16/20     10865.000000
2/17/20     12583.000000
2/18/20     14352.000000
                ...
3/12/20     68324.000000
3/13/20     70251.000000
3/14/20     72624.000000
3/15/20     76034.000000
3/16/20     78088.000000
3/17/20     80840.000000
3/18/20     83312.000000
3/19/20     84975.000000
3/20/20     87420.000000
3/21/20     91692.000000
3/22/20     97899.000000
3/23/20     98351.000000
3/24/20    108000.000000
3/25/20    113787.000000
3/26/20    122150.000000
3/27/20    130915.000000
3/28/20    139415.000000
3/29/20    149082.000000
3/30/20    164566.000000
3/31/20    178034.000000
4/1/20     193177.000000
4/2/20     210263.000000
4/3/20     225796.000000
4/4/20     246152.000000
4/5/20     260012.000000
4/6/20     276515.000000
4/7/20     300054.000000
4/8/20     328661.000000
```

```
4/9/20     353975.000000
4/10/20    376096.000000
Length: 82, dtype: float64
```

In [11]:

```
Serie_Temporal = DataFrame.copy()
```

In [12]:

```
Serie_Temporal
```

Out[12]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/2(
0	NaN	Afghanistan	33.000000	65.000000	0	0	0	(
1	NaN	Albania	41.153300	20.168300	0	0	0	(
2	NaN	Algeria	28.033900	1.659600	0	0	0	(
3	NaN	Andorra	42.506300	1.521800	0	0	0	(
4	NaN	Angola	-11.202700	17.873900	0	0	0	(
5	NaN	Antigua and Barbuda	17.060800	-61.796400	0	0	0	(
6	NaN	Argentina	-38.416100	-63.616700	0	0	0	(
7	NaN	Armenia	40.069100	45.038200	0	0	0	(
8	Australian Capital Territory	Australia	-35.473500	149.012400	0	0	0	(
9	New South Wales	Australia	-33.868800	151.209300	0	0	0	(
10	Northern Territory	Australia	-12.463400	130.845600	0	0	0	(
11	Queensland	Australia	-28.016700	153.400000	0	0	0	(
12	South Australia	Australia	-34.928500	138.600700	0	0	0	(
13	Tasmania	Australia	-41.454500	145.970700	0	0	0	(
14	Victoria	Australia	-37.813600	144.963100	0	0	0	(
15	Western Australia	Australia	-31.950500	115.860500	0	0	0	(
16	NaN	Austria	47.516200	14.550100	0	0	0	(
17	NaN	Azerbaijan	40.143100	47.576900	0	0	0	(
18	NaN	Bahamas	25.034300	-77.396300	0	0	0	(
19	NaN	Bahrain	26.027500	50.550000	0	0	0	(
20	NaN	Bangladesh	23.685000	90.356300	0	0	0	(
21	NaN	Barbados	13.193900	-59.543200	0	0	0	(
22	NaN	Belarus	53.709800	27.953400	0	0	0	(
23	NaN	Belgium	50.833300	4.000000	0	0	0	(
24	NaN	Belize	13.193900	-59.543200	0	0	0	(
25	NaN	Benin	9.307700	2.315800	0	0	0	(
26	NaN	Bhutan	27.514200	90.433600	0	0	0	(
27	NaN	Bolivia	-16.290200	-63.588700	0	0	0	(
28	NaN	Bosnia and Herzegovina	43.915900	17.679100	0	0	0	(
29	NaN	Brazil	-14.235000	-51.925300	0	0	0	(
...	...	...	...	...	...	...	...	..
220	Gibraltar	United Kingdom	36.140800	-5.353600	0	0	0	(
221	Isle of Man	United Kingdom	54.236100	-4.548100	0	0	0	(

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/2(
222	Montserrat	United Kingdom	16.742500	-62.187400	0	0	0	(
223	NaN	United Kingdom	55.378100	-3.436000	0	0	0	(
224	NaN	Uruguay	-32.522800	-55.765800	0	0	0	(
225	NaN	US	37.090200	-95.712900	0	0	0	(
226	NaN	Uzbekistan	41.377500	64.585300	0	0	0	(
227	NaN	Venezuela	6.423800	-66.589700	0	0	0	(
228	NaN	Vietnam	16.000000	108.000000	0	0	0	(
229	NaN	Zambia	-15.416700	28.283300	0	0	0	(
230	NaN	Zimbabwe	-20.000000	30.000000	0	0	0	(
231	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	(
232	NaN	Laos	19.856270	102.495496	0	0	0	(
233	NaN	Kosovo	42.602636	20.902977	0	0	0	(
234	NaN	Burma	21.916200	95.956000	0	0	0	(
235	Anguilla	United Kingdom	18.220600	-63.068600	0	0	0	(
236	British Virgin Islands	United Kingdom	18.420700	-64.640000	0	0	0	(
237	Turks and Caicos Islands	United Kingdom	21.694000	-71.797900	0	0	0	(
238	NaN	MS Zaandam	0.000000	0.000000	0	0	0	(
239	NaN	Botswana	-22.328500	24.684900	0	0	0	(
240	NaN	Burundi	-3.373100	29.918900	0	0	0	(
241	NaN	Sierra Leone	8.460555	-11.779889	0	0	0	(
242	Bonaire, Sint Eustatius and Saba	Netherlands	12.178400	-68.238500	0	0	0	(
243	NaN	Malawi	-13.254308	34.301525	0	0	0	(
244	Falkland Islands (Malvinas)	United Kingdom	-51.796300	-59.523600	0	0	0	(
245	Saint Pierre and Miquelon	France	46.885200	-56.315900	0	0	0	(
246	NaN	South Sudan	6.877000	31.307000	0	0	0	(
247	NaN	Western Sahara	24.215500	-12.885800	0	0	0	(
248	NaN	Sao Tome and Principe	0.186360	6.613081	0	0	0	(
249	NaN	Yemen	15.552727	48.516388	0	0	0	(

250 rows × 84 columns

In [19]:

```
Serie_Temporal
```

Out[19]:

	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...
5	0	0	0	0	0	0	0	0	0	0	...
6	0	0	0	0	0	0	0	0	0	0	...
7	0	0	0	0	0	0	0	0	0	0	...
8	0	0	0	0	0	0	0	0	0	0	...
9	0	0	0	0	0	0	0	0	2	2	...
10	0	0	0	0	0	0	0	0	0	0	...
11	0	0	0	0	0	0	0	0	0	0	...
12	0	0	0	0	0	0	0	0	0	0	...
13	0	0	0	0	0	0	0	0	0	0	...
14	0	0	0	0	0	0	0	0	0	0	...
15	0	0	0	0	0	0	0	0	0	0	...
16	0	0	0	0	0	0	0	0	0	0	...
17	0	0	0	0	0	0	0	0	0	0	...
18	0	0	0	0	0	0	0	0	0	0	...
19	0	0	0	0	0	0	0	0	0	0	...
20	0	0	0	0	0	0	0	0	0	0	...
21	0	0	0	0	0	0	0	0	0	0	...
22	0	0	0	0	0	0	0	0	0	0	...
23	0	0	0	0	0	0	0	0	0	0	...
24	0	0	0	0	0	0	0	0	0	0	...
25	0	0	0	0	0	0	0	0	0	0	...
26	0	0	0	0	0	0	0	0	0	0	...
27	0	0	0	0	0	0	0	0	0	0	...
28	0	0	0	0	0	0	0	0	0	0	...
29	0	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
220	0	0	0	0	0	0	0	0	0	0	...
221	0	0	0	0	0	0	0	0	0	0	...
222	0	0	0	0	0	0	0	0	0	0	...
223	0	0	0	0	0	0	0	0	0	0	...
224	0	0	0	0	0	0	0	0	0	0	...
225	0	0	0	0	0	0	0	0	0	0	...

	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...
226	0	0	0	0	0	0	0	0	0	0	...
227	0	0	0	0	0	0	0	0	0	0	...
228	0	0	0	0	0	0	0	0	0	0	...
229	0	0	0	0	0	0	0	0	0	0	...
230	0	0	0	0	0	0	0	0	0	0	...
231	0	0	0	0	0	0	0	0	0	0	...
232	0	0	0	0	0	0	0	0	0	0	...
233	0	0	0	0	0	0	0	0	0	0	...
234	0	0	0	0	0	0	0	0	0	0	...
235	0	0	0	0	0	0	0	0	0	0	...
236	0	0	0	0	0	0	0	0	0	0	...
237	0	0	0	0	0	0	0	0	0	0	...
238	0	0	0	0	0	0	0	0	0	0	...
239	0	0	0	0	0	0	0	0	0	0	...
240	0	0	0	0	0	0	0	0	0	0	...
241	0	0	0	0	0	0	0	0	0	0	...
242	0	0	0	0	0	0	0	0	0	0	...
243	0	0	0	0	0	0	0	0	0	0	...
244	0	0	0	0	0	0	0	0	0	0	...
245	0	0	0	0	0	0	0	0	0	0	...
246	0	0	0	0	0	0	0	0	0	0	...
247	0	0	0	0	0	0	0	0	0	0	...
248	0	0	0	0	0	0	0	0	0	0	...
249	0	0	0	0	0	0	0	0	0	0	...

250 rows × 80 columns

In [20]:

```
Serie_Temporal.iloc[:3,:3]
```

Out[20]:

	1/22/20	1/23/20	1/24/20
0	0	0	0
1	0	0	0
2	0	0	0

In [21]:

```
Serie_Temporal.sum()
```

Out[21]:

```
1/22/20        28
1/23/20        30
1/24/20        36
1/25/20        39
1/26/20        52
1/27/20        61
1/28/20       107
1/29/20       126
1/30/20       143
1/31/20       222
2/1/20        284
2/2/20        472
2/3/20        623
2/4/20        852
2/5/20       1124
2/6/20       1487
2/7/20       2011
2/8/20       2616
2/9/20       3244
2/10/20      3946
2/11/20      4683
2/12/20      5150
2/13/20      6295
2/14/20      8058
2/15/20      9395
2/16/20     10865
2/17/20     12583
2/18/20     14352
2/19/20     16121
2/20/20     18177
            ...
3/12/20     68324
3/13/20     70251
3/14/20     72624
3/15/20     76034
3/16/20     78088
3/17/20     80840
3/18/20     83312
3/19/20     84975
3/20/20     87420
3/21/20     91692
3/22/20     97899
3/23/20     98351
3/24/20    108000
3/25/20    113787
3/26/20    122150
3/27/20    130915
3/28/20    139415
3/29/20    149082
3/30/20    164566
3/31/20    178034
4/1/20     193177
4/2/20     210263
4/3/20     225796
4/4/20     246152
4/5/20     260012
4/6/20     276515
4/7/20     300054
4/8/20     328661
```

```
4/9/20     353975
4/10/20    376096
Length: 80, dtype: int64
```

In [22]:

```
Serie_Geral = Serie_Temporal.sum().copy()
```

In [23]:

```
Serie_Geral
```

Out[23]:

```
1/22/20        28
1/23/20        30
1/24/20        36
1/25/20        39
1/26/20        52
1/27/20        61
1/28/20       107
1/29/20       126
1/30/20       143
1/31/20       222
2/1/20        284
2/2/20        472
2/3/20        623
2/4/20        852
2/5/20       1124
2/6/20       1487
2/7/20       2011
2/8/20       2616
2/9/20       3244
2/10/20      3946
2/11/20      4683
2/12/20      5150
2/13/20      6295
2/14/20      8058
2/15/20      9395
2/16/20     10865
2/17/20     12583
2/18/20     14352
2/19/20     16121
2/20/20     18177
            ...
3/12/20     68324
3/13/20     70251
3/14/20     72624
3/15/20     76034
3/16/20     78088
3/17/20     80840
3/18/20     83312
3/19/20     84975
3/20/20     87420
3/21/20     91692
3/22/20     97899
3/23/20     98351
3/24/20    108000
3/25/20    113787
3/26/20    122150
3/27/20    130915
3/28/20    139415
3/29/20    149082
3/30/20    164566
3/31/20    178034
4/1/20     193177
4/2/20     210263
4/3/20     225796
4/4/20     246152
4/5/20     260012
4/6/20     276515
4/7/20     300054
4/8/20     328661
```

```
4/9/20     353975
4/10/20    376096
Length: 80, dtype: int64
```

In [24]:

```
Serie_Geral = pd.DataFrame(Serie_Geral)
Serie_Geral.rename(columns = {0 :"y"}, inplace = True)
```

In [25]:

```
Serie_Geral.iloc[:3,:3]
```

Out[25]:

	y
1/22/20	28
1/23/20	30
1/24/20	36

In [26]:

```
Serie_Geral
```

Out[26]:

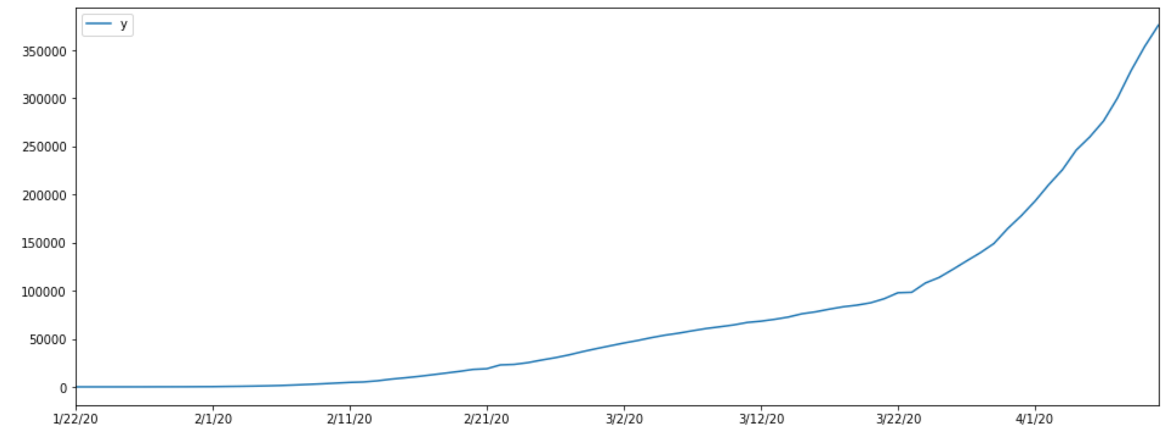
	y
1/22/20	28
1/23/20	30
1/24/20	36
1/25/20	39
1/26/20	52
1/27/20	61
1/28/20	107
1/29/20	126
1/30/20	143
1/31/20	222
2/1/20	284
2/2/20	472
2/3/20	623
2/4/20	852
2/5/20	1124
2/6/20	1487
2/7/20	2011
2/8/20	2616
2/9/20	3244
2/10/20	3946
2/11/20	4683
2/12/20	5150
2/13/20	6295
2/14/20	8058
2/15/20	9395
2/16/20	10865
2/17/20	12583
2/18/20	14352
2/19/20	16121
2/20/20	18177
...	...
3/12/20	68324
3/13/20	70251
3/14/20	72624
3/15/20	76034
3/16/20	78088
3/17/20	80840

	y
3/18/20	83312
3/19/20	84975
3/20/20	87420
3/21/20	91692
3/22/20	97899
3/23/20	98351
3/24/20	108000
3/25/20	113787
3/26/20	122150
3/27/20	130915
3/28/20	139415
3/29/20	149082
3/30/20	164566
3/31/20	178034
4/1/20	193177
4/2/20	210263
4/3/20	225796
4/4/20	246152
4/5/20	260012
4/6/20	276515
4/7/20	300054
4/8/20	328661
4/9/20	353975
4/10/20	376096

80 rows × 1 columns

In [27]:

```
Serie_Geral.plot(figsize=(16,6))
plt.show()
```



In [28]:

```
Serie_Geral.index = pd.to_datetime(Serie_Geral.index)
```

In [29]:

```
Serie_Geral
```

Out[29]:

	y
2020-01-22	28
2020-01-23	30
2020-01-24	36
2020-01-25	39
2020-01-26	52
2020-01-27	61
2020-01-28	107
2020-01-29	126
2020-01-30	143
2020-01-31	222
2020-02-01	284
2020-02-02	472
2020-02-03	623
2020-02-04	852
2020-02-05	1124
2020-02-06	1487
2020-02-07	2011
2020-02-08	2616
2020-02-09	3244
2020-02-10	3946
2020-02-11	4683
2020-02-12	5150
2020-02-13	6295
2020-02-14	8058
2020-02-15	9395
2020-02-16	10865
2020-02-17	12583
2020-02-18	14352
2020-02-19	16121
2020-02-20	18177
...	...
2020-03-12	68324
2020-03-13	70251
2020-03-14	72624
2020-03-15	76034
2020-03-16	78088
2020-03-17	80840

	y
2020-03-18	83312
2020-03-19	84975
2020-03-20	87420
2020-03-21	91692
2020-03-22	97899
2020-03-23	98351
2020-03-24	108000
2020-03-25	113787
2020-03-26	122150
2020-03-27	130915
2020-03-28	139415
2020-03-29	149082
2020-03-30	164566
2020-03-31	178034
2020-04-01	193177
2020-04-02	210263
2020-04-03	225796
2020-04-04	246152
2020-04-05	260012
2020-04-06	276515
2020-04-07	300054
2020-04-08	328661
2020-04-09	353975
2020-04-10	376096

80 rows × 1 columns

In [30]:

```
Serie_Geral.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 80 entries, 2020-01-22 to 2020-04-10
Data columns (total 1 columns):
y    80 non-null int64
dtypes: int64(1)
memory usage: 1.2 KB
```

In [31]:

```
tempos = list(range(Serie_Geral.shape[0]))
Serie_Geral["tempos"] = tempos
```

In [32]:

```
Serie_Geral.index.day
```

Out[32]:

```
Int64Index([22, 23, 24, 25, 26, 27, 28, 29, 30, 31,  1,  2,  3,  4,  5,
6,  7,
             8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 2
3, 24,
            25, 26, 27, 28, 29,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 1
1, 12,
            13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 2
8, 29,
            30, 31,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10],
           dtype='int64')
```

In [34]:

```
Serie_Geral["Dia"] = Serie_Geral.index.day
```

In [35]:

```
Serie_Geral
```

Out[35]:

	y	tempos	Dia
2020-01-22	28	0	22
2020-01-23	30	1	23
2020-01-24	36	2	24
2020-01-25	39	3	25
2020-01-26	52	4	26
2020-01-27	61	5	27
2020-01-28	107	6	28
2020-01-29	126	7	29
2020-01-30	143	8	30
2020-01-31	222	9	31
2020-02-01	284	10	1
2020-02-02	472	11	2
2020-02-03	623	12	3
2020-02-04	852	13	4
2020-02-05	1124	14	5
2020-02-06	1487	15	6
2020-02-07	2011	16	7
2020-02-08	2616	17	8
2020-02-09	3244	18	9
2020-02-10	3946	19	10
2020-02-11	4683	20	11
2020-02-12	5150	21	12
2020-02-13	6295	22	13
2020-02-14	8058	23	14
2020-02-15	9395	24	15
2020-02-16	10865	25	16
2020-02-17	12583	26	17
2020-02-18	14352	27	18
2020-02-19	16121	28	19
2020-02-20	18177	29	20
...	...	...	...
2020-03-12	68324	50	12
2020-03-13	70251	51	13
2020-03-14	72624	52	14
2020-03-15	76034	53	15
2020-03-16	78088	54	16
2020-03-17	80840	55	17

	y	tempos	Dia
2020-03-18	83312	56	18
2020-03-19	84975	57	19
2020-03-20	87420	58	20
2020-03-21	91692	59	21
2020-03-22	97899	60	22
2020-03-23	98351	61	23
2020-03-24	108000	62	24
2020-03-25	113787	63	25
2020-03-26	122150	64	26
2020-03-27	130915	65	27
2020-03-28	139415	66	28
2020-03-29	149082	67	29
2020-03-30	164566	68	30
2020-03-31	178034	69	31
2020-04-01	193177	70	1
2020-04-02	210263	71	2
2020-04-03	225796	72	3
2020-04-04	246152	73	4
2020-04-05	260012	74	5
2020-04-06	276515	75	6
2020-04-07	300054	76	7
2020-04-08	328661	77	8
2020-04-09	353975	78	9
2020-04-10	376096	79	10

80 rows × 3 columns

In [36]:

```
Serie_Geral["Dia da Semana"] = Serie_Geral.index.dayofweek
Serie_Geral
```

Out[36]:

	y	tempos	Dia	Dia da Semana
2020-01-22	28	0	22	2
2020-01-23	30	1	23	3
2020-01-24	36	2	24	4
2020-01-25	39	3	25	5
2020-01-26	52	4	26	6
2020-01-27	61	5	27	0
2020-01-28	107	6	28	1
2020-01-29	126	7	29	2
2020-01-30	143	8	30	3
2020-01-31	222	9	31	4
2020-02-01	284	10	1	5
2020-02-02	472	11	2	6
2020-02-03	623	12	3	0
2020-02-04	852	13	4	1
2020-02-05	1124	14	5	2
2020-02-06	1487	15	6	3
2020-02-07	2011	16	7	4
2020-02-08	2616	17	8	5
2020-02-09	3244	18	9	6
2020-02-10	3946	19	10	0
2020-02-11	4683	20	11	1
2020-02-12	5150	21	12	2
2020-02-13	6295	22	13	3
2020-02-14	8058	23	14	4
2020-02-15	9395	24	15	5
2020-02-16	10865	25	16	6
2020-02-17	12583	26	17	0
2020-02-18	14352	27	18	1
2020-02-19	16121	28	19	2
2020-02-20	18177	29	20	3
...	...	...	...	...
2020-03-12	68324	50	12	3
2020-03-13	70251	51	13	4
2020-03-14	72624	52	14	5
2020-03-15	76034	53	15	6
2020-03-16	78088	54	16	0
2020-03-17	80840	55	17	1

	y	tempos	Dia	Dia da Semana
2020-03-18	83312	56	18	2
2020-03-19	84975	57	19	3
2020-03-20	87420	58	20	4
2020-03-21	91692	59	21	5
2020-03-22	97899	60	22	6
2020-03-23	98351	61	23	0
2020-03-24	108000	62	24	1
2020-03-25	113787	63	25	2
2020-03-26	122150	64	26	3
2020-03-27	130915	65	27	4
2020-03-28	139415	66	28	5
2020-03-29	149082	67	29	6
2020-03-30	164566	68	30	0
2020-03-31	178034	69	31	1
2020-04-01	193177	70	1	2
2020-04-02	210263	71	2	3
2020-04-03	225796	72	3	4
2020-04-04	246152	73	4	5
2020-04-05	260012	74	5	6
2020-04-06	276515	75	6	0
2020-04-07	300054	76	7	1
2020-04-08	328661	77	8	2
2020-04-09	353975	78	9	3
2020-04-10	376096	79	10	4

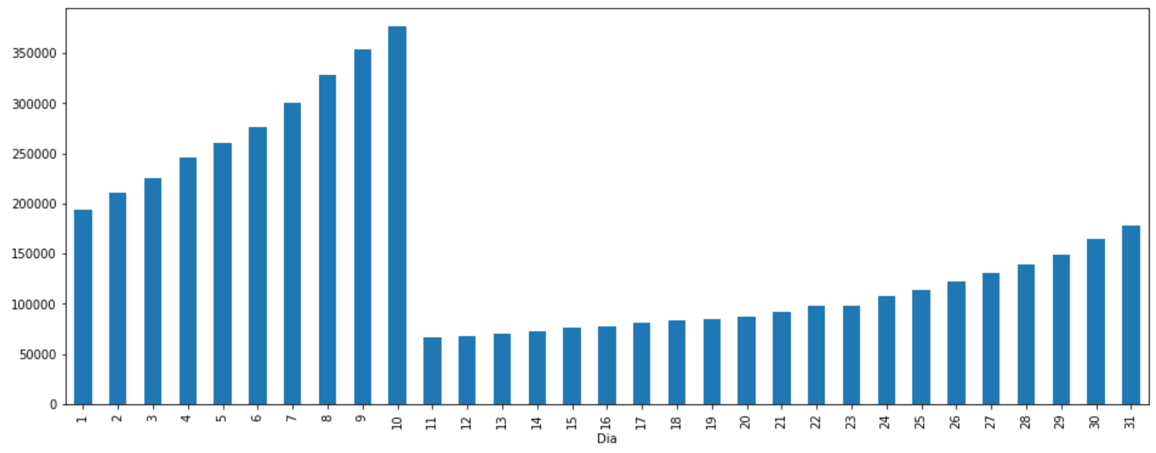
80 rows × 4 columns

In [37]:

```
Serie_Geral.groupby("Dia")["y"].max().plot.bar(figsize=(16,6))
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2241ab44320>
```

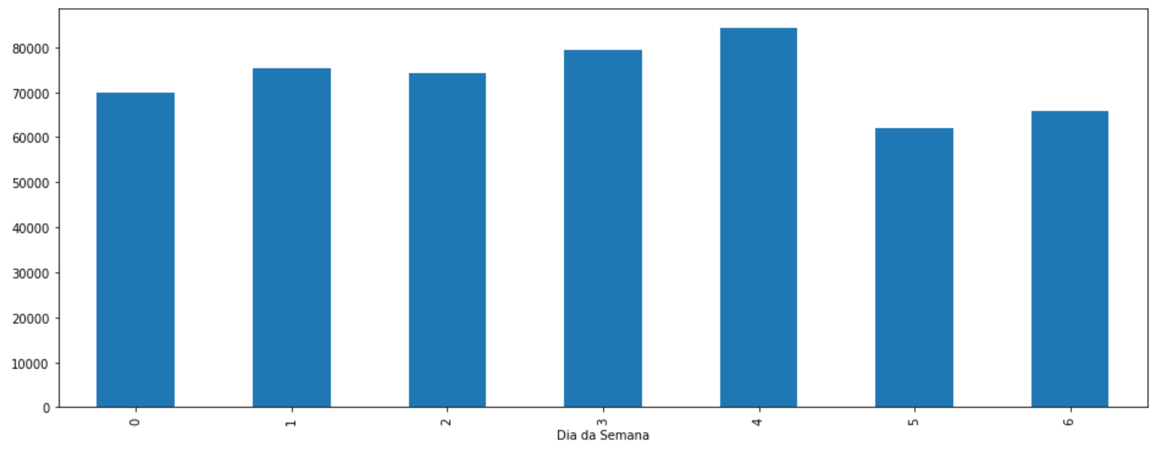


In [38]:

```
Serie_Geral.groupby("Dia da Semana")["y"].mean().plot.bar(figsize=(16,6))
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2241ac626d8>
```



In [39]:

```
Serie_Geral.shape[0]
```

Out[39]:

```
80
```

In [40]:

```
Taxa_Treino = 0.9
X_Treino = Serie_Geral.iloc[:round(Serie_Geral.shape[0] * Taxa_Treino), 1:]
X_Treino
```

Out[40]:

	tempos	Dia	Dia da Semana
2020-01-22	0	22	2
2020-01-23	1	23	3
2020-01-24	2	24	4
2020-01-25	3	25	5
2020-01-26	4	26	6
2020-01-27	5	27	0
2020-01-28	6	28	1
2020-01-29	7	29	2
2020-01-30	8	30	3
2020-01-31	9	31	4
2020-02-01	10	1	5
2020-02-02	11	2	6
2020-02-03	12	3	0
2020-02-04	13	4	1
2020-02-05	14	5	2
2020-02-06	15	6	3
2020-02-07	16	7	4
2020-02-08	17	8	5
2020-02-09	18	9	6
2020-02-10	19	10	0
2020-02-11	20	11	1
2020-02-12	21	12	2
2020-02-13	22	13	3
2020-02-14	23	14	4
2020-02-15	24	15	5
2020-02-16	25	16	6
2020-02-17	26	17	0
2020-02-18	27	18	1
2020-02-19	28	19	2
2020-02-20	29	20	3
...	...	...	...
2020-03-04	42	4	2
2020-03-05	43	5	3
2020-03-06	44	6	4
2020-03-07	45	7	5
2020-03-08	46	8	6
2020-03-09	47	9	0

	tempos	Dia	Dia da Semana
2020-03-10	48	10	1
2020-03-11	49	11	2
2020-03-12	50	12	3
2020-03-13	51	13	4
2020-03-14	52	14	5
2020-03-15	53	15	6
2020-03-16	54	16	0
2020-03-17	55	17	1
2020-03-18	56	18	2
2020-03-19	57	19	3
2020-03-20	58	20	4
2020-03-21	59	21	5
2020-03-22	60	22	6
2020-03-23	61	23	0
2020-03-24	62	24	1
2020-03-25	63	25	2
2020-03-26	64	26	3
2020-03-27	65	27	4
2020-03-28	66	28	5
2020-03-29	67	29	6
2020-03-30	68	30	0
2020-03-31	69	31	1
2020-04-01	70	1	2
2020-04-02	71	2	3

72 rows × 3 columns

In [41]:

```
Serie_Geral["Anterior"] = Serie_Geral["y"].shift(5)
Serie_Geral = Serie_Geral.iloc[5:]
Serie_Geral
```

Out[41]:

	y	tempos	Dia	Dia da Semana	Anterior
2020-01-27	61	5	27	0	28.0
2020-01-28	107	6	28	1	30.0
2020-01-29	126	7	29	2	36.0
2020-01-30	143	8	30	3	39.0
2020-01-31	222	9	31	4	52.0
2020-02-01	284	10	1	5	61.0
2020-02-02	472	11	2	6	107.0
2020-02-03	623	12	3	0	126.0
2020-02-04	852	13	4	1	143.0
2020-02-05	1124	14	5	2	222.0
2020-02-06	1487	15	6	3	284.0
2020-02-07	2011	16	7	4	472.0
2020-02-08	2616	17	8	5	623.0
2020-02-09	3244	18	9	6	852.0
2020-02-10	3946	19	10	0	1124.0
2020-02-11	4683	20	11	1	1487.0
2020-02-12	5150	21	12	2	2011.0
2020-02-13	6295	22	13	3	2616.0
2020-02-14	8058	23	14	4	3244.0
2020-02-15	9395	24	15	5	3946.0
2020-02-16	10865	25	16	6	4683.0
2020-02-17	12583	26	17	0	5150.0
2020-02-18	14352	27	18	1	6295.0
2020-02-19	16121	28	19	2	8058.0
2020-02-20	18177	29	20	3	9395.0
2020-02-21	18890	30	21	4	10865.0
2020-02-22	22886	31	22	5	12583.0
2020-02-23	23394	32	23	6	14352.0
2020-02-24	25227	33	24	0	16121.0
2020-02-25	27905	34	25	1	18177.0
...	...	...	...	...	...
2020-03-12	68324	50	12	3	58358.0
2020-03-13	70251	51	13	4	60694.0
2020-03-14	72624	52	14	5	62494.0
2020-03-15	76034	53	15	6	64404.0
2020-03-16	78088	54	16	0	67003.0
2020-03-17	80840	55	17	1	68324.0

	y	tempos	Dia	Dia da Semana	Anterior
2020-03-18	83312	56	18	2	70251.0
2020-03-19	84975	57	19	3	72624.0
2020-03-20	87420	58	20	4	76034.0
2020-03-21	91692	59	21	5	78088.0
2020-03-22	97899	60	22	6	80840.0
2020-03-23	98351	61	23	0	83312.0
2020-03-24	108000	62	24	1	84975.0
2020-03-25	113787	63	25	2	87420.0
2020-03-26	122150	64	26	3	91692.0
2020-03-27	130915	65	27	4	97899.0
2020-03-28	139415	66	28	5	98351.0
2020-03-29	149082	67	29	6	108000.0
2020-03-30	164566	68	30	0	113787.0
2020-03-31	178034	69	31	1	122150.0
2020-04-01	193177	70	1	2	130915.0
2020-04-02	210263	71	2	3	139415.0
2020-04-03	225796	72	3	4	149082.0
2020-04-04	246152	73	4	5	164566.0
2020-04-05	260012	74	5	6	178034.0
2020-04-06	276515	75	6	0	193177.0
2020-04-07	300054	76	7	1	210263.0
2020-04-08	328661	77	8	2	225796.0
2020-04-09	353975	78	9	3	246152.0
2020-04-10	376096	79	10	4	260012.0

75 rows × 5 columns

In [43]:

```
Taxa_Treino = 0.95
X_Treino = Serie_Geral.iloc[:round(Serie_Geral.shape[0] * Taxa_Treino), 1:]
X_Teste = Serie_Geral.iloc[round(Serie_Geral.shape[0] * Taxa_Treino):, 1:]
Y_Treino = Serie_Geral.iloc[:round(Serie_Geral.shape[0] * Taxa_Treino), 0]
Y_Teste = Serie_Geral.iloc[round(Serie_Geral.shape[0] * Taxa_Treino):, 0]
```

In [44]:

```
len(X_Teste)
```

Out[44]:

```
4
```

In [45]:

```
Y_Treino
```

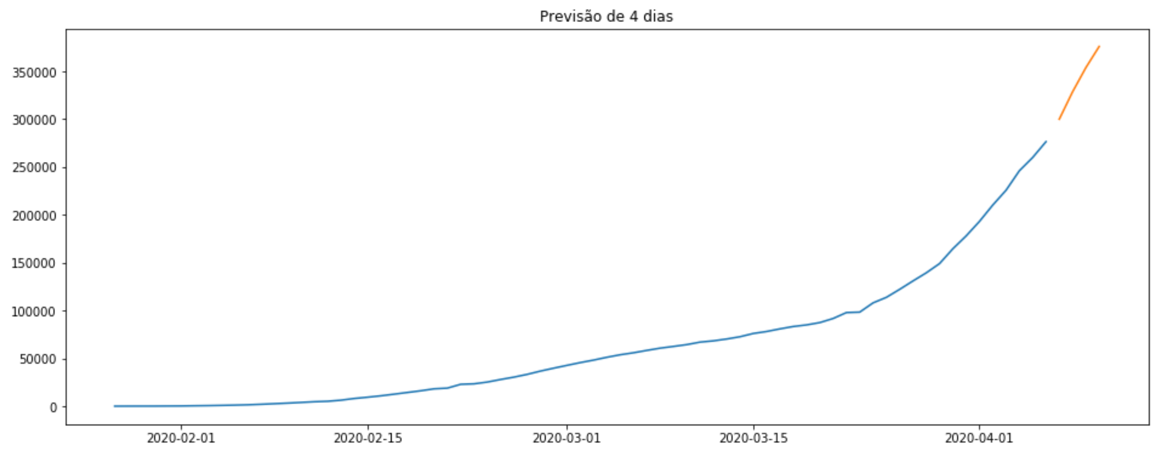
Out[45]:

```
2020-01-27        61
2020-01-28       107
2020-01-29       126
2020-01-30       143
2020-01-31       222
2020-02-01       284
2020-02-02       472
2020-02-03       623
2020-02-04       852
2020-02-05      1124
2020-02-06      1487
2020-02-07      2011
2020-02-08      2616
2020-02-09      3244
2020-02-10      3946
2020-02-11      4683
2020-02-12      5150
2020-02-13      6295
2020-02-14      8058
2020-02-15      9395
2020-02-16     10865
2020-02-17     12583
2020-02-18     14352
2020-02-19     16121
2020-02-20     18177
2020-02-21     18890
2020-02-22     22886
2020-02-23     23394
2020-02-24     25227
2020-02-25     27905
                ...
2020-03-08     60694
2020-03-09     62494
2020-03-10     64404
2020-03-11     67003
2020-03-12     68324
2020-03-13     70251
2020-03-14     72624
2020-03-15     76034
2020-03-16     78088
2020-03-17     80840
2020-03-18     83312
2020-03-19     84975
2020-03-20     87420
2020-03-21     91692
2020-03-22     97899
2020-03-23     98351
2020-03-24    108000
2020-03-25    113787
2020-03-26    122150
2020-03-27    130915
2020-03-28    139415
2020-03-29    149082
2020-03-30    164566
2020-03-31    178034
2020-04-01    193177
2020-04-02    210263
2020-04-03    225796
2020-04-04    246152
```

```
2020-04-05    260012
2020-04-06    276515
Name: y, Length: 71, dtype: int64
```

In [46]:

```
plt.figure(figsize = (16,6))
plt.title(f'Previsão de {len(Y_Teste)} dias')
plt.plot(Y_Treino)
plt.plot(Y_Teste)
plt.show()
```



In [47]:

```
from sklearn.linear_model import LinearRegression
```

In [48]:

```
modelo = LinearRegression().fit(X_Treino, Y_Treino)
modelo
```

Out[48]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [49]:

```
modelo.score(X_Treino,Y_Treino)
```

Out[49]:

```
0.9886887338264854
```

In [50]:

```
modelo.coef_, modelo.intercept_
```

Out[50]:

```
(array([-520.42783918,  -75.34214462,  192.91990499,    1.5991983 ]),
 10884.410128643642)
```

In [51]:

```
Previsto = modelo.predict(X_Teste)
```

In [52]:

```
Previsto.round()
```

Out[52]:

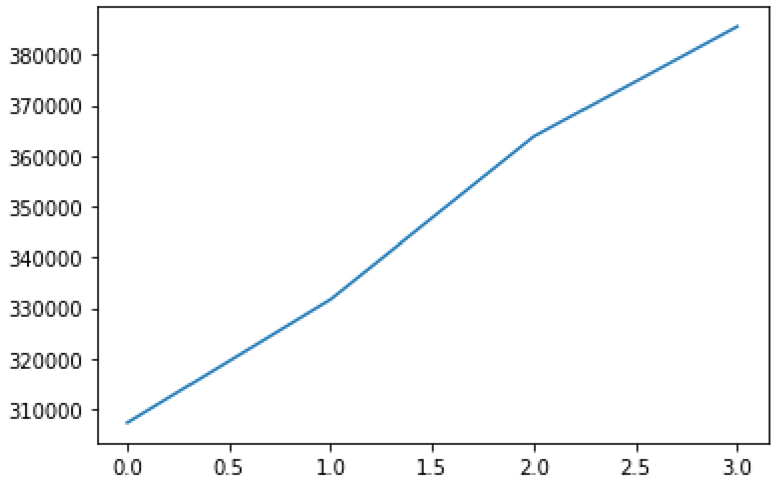
```
array([307250., 331687., 363838., 385600.])
```

In [53]:

```
plt.plot(Previsto)
```

Out[53]:

```
[<matplotlib.lines.Line2D at 0x2241d419588>]
```



In [54]:

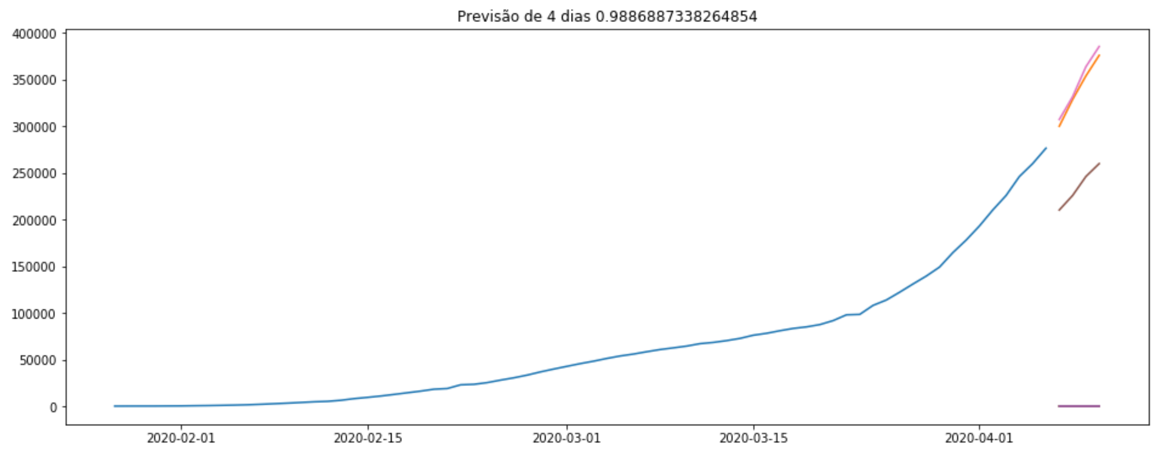
```
df_Previsao = X_Teste.copy()
df_Previsao[0] = Previsto
df_Previsao
```

Out[54]:

	tempos	Dia	Dia da Semana	Anterior	0
2020-04-07	76	7	1	210263.0	307249.651234
2020-04-08	77	8	2	225796.0	331687.148337
2020-04-09	78	9	3	246152.0	363837.578838
2020-04-10	79	10	4	260012.0	385599.617186

In [55]:

```
plt.figure(figsize = (16,6))
plt.title(f'Previsão de {len(Y_Teste)} dias {modelo.score(X_Treino, Y_Treino)}')
plt.plot(Y_Treino)
plt.plot(Y_Teste)
plt.plot(df_Previsao)
plt.show()
```



In [ ]:

```

```