```
In [1]: import numpy as np, pandas as pd
        from IPython.display import Image
        import matplotlib.pyplot as plt, seaborn as sns
        import scipy
        import warnings
        import plotly.express as px
        from itertools import product
        import statsmodels.api as sm
        import datetime
        from tqdm import tqdm
        from scipy.special import boxcox1p
        warnings.filterwarnings('ignore')
```

```
In [2]: #Loading Data
        data = pd.read_csv('/Users/user/Desktop/idsFinalProject/country_vaccinations.csv')
```

```
In [3]: data
```

Out[3]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	da
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	
...	...	...	...	...	...	...	
26233	Zimbabwe	ZWE	2021-06-18	1131397.0	700244.0	431153.0	
26234	Zimbabwe	ZWE	2021-06-19	1133920.0	701348.0	432572.0	
26235	Zimbabwe	ZWE	2021-06-20	1138733.0	703065.0	435668.0	
26236	Zimbabwe	ZWE	2021-06-21	1140852.0	704001.0	436851.0	
26237	Zimbabwe	ZWE	2021-06-22	1146378.0	706158.0	440220.0	

26238 rows × 15 columns

```
In [4]: data.shape

Out[4]: (26238, 15)
```

```
In [5]: #Checking Missing Data
        data.isna().sum()

Out[5]: country                                    0
        iso_code                                   0
        date                                       0
        total_vaccinations                     11490
        people_vaccinated                      12267
        people_fully_vaccinated                14959
        daily_vaccinations_raw                 14048
        daily_vaccinations                       273
        total_vaccinations_per_hundred         11490
        people_vaccinated_per_hundred          12267
        people_fully_vaccinated_per_hundred    14959
        daily_vaccinations_per_million           273
        vaccines                                   0
        source_name                                0
        source_website                             0
        dtype: int64
```

```
In [6]: data = data.drop(data[data.total_vaccinations.isna()].index)
```

```
In [7]: data.isna().sum()

Out[7]: country                                   0
        iso_code                                  0
        date                                      0
        total_vaccinations                        0
        people_vaccinated                       921
        people_fully_vaccinated                3509
        daily_vaccinations_raw                 2558
        daily_vaccinations                      217
        total_vaccinations_per_hundred            0
        people_vaccinated_per_hundred           921
        people_fully_vaccinated_per_hundred    3509
        daily_vaccinations_per_million          217
        vaccines                                  0
        source_name                               0
        source_website                            0
        dtype: int64
```

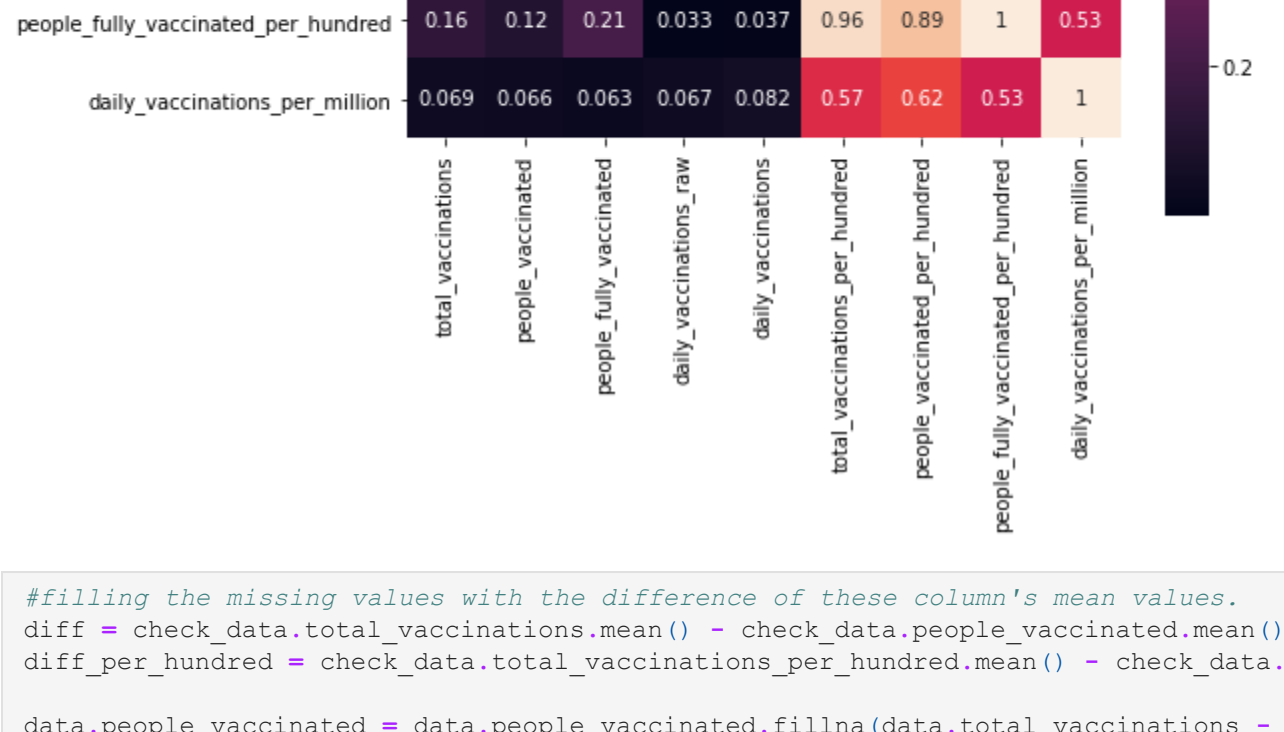
```
In [8]: check_data = data.drop(data[data.people_vaccinated.isna()].index)
```

```
In [9]: check_data.head()
```

Out[9]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	
6	Afghanistan	AFG	2021-02-28	8200.0	8200.0	NaN	
22	Afghanistan	AFG	2021-03-16	54000.0	54000.0	NaN	
44	Afghanistan	AFG	2021-04-07	120000.0	120000.0	NaN	
59	Afghanistan	AFG	2021-04-22	240000.0	240000.0	NaN	

```
In [10]: #As can bee seen from data, the values of total_vaccinations column are mostly the sa
         #Also can bee seen from the heatmap, these features have almost ideal correlation.
         plt.subplots(figsize=(8, 8))
         sns.heatmap(check_data.corr(), annot=True, square=True)
         plt.show()
```



```
In [11]: #filling the missing values with the difference of these column's mean values.
         diff = check_data.total_vaccinations.mean() - check_data.people_vaccinated.mean()
         diff_per_hundred = check_data.total_vaccinations_per_hundred.mean() - check_data.peop

         data.people_vaccinated = data.people_vaccinated.fillna(data.total_vaccinations - diff
         data.people_vaccinated_per_hundred = data.people_vaccinated_per_hundred.fillna(data.t
```

```
In [12]: data.isna().sum()

Out[12]: country                                   0
         iso_code                                  0
         date                                      0
         total_vaccinations                        0
         people_vaccinated                         0
         people_fully_vaccinated                3509
         daily_vaccinations_raw                 2558
         daily_vaccinations                      217
         total_vaccinations_per_hundred            0
         people_vaccinated_per_hundred             0
         people_fully_vaccinated_per_hundred    3509
         daily_vaccinations_per_million          217
         vaccines                                  0
         source_name                               0
         source_website                            0
         dtype: int64
```

```
In [13]: #daily_vaccinations and daily_vaccinations_per_million greatly correlates with people_
         #So, just fill missing values with zeros.
         data.daily_vaccinations = data.daily_vaccinations.fillna(0)
         data.daily_vaccinations_per_million = data.daily_vaccinations_per_million.fillna(0)
```

```
In [14]: data.isna().sum()

Out[14]: country                                   0
         iso_code                                  0
         date                                      0
         total_vaccinations                        0
         people_vaccinated                         0
         people_fully_vaccinated                3509
         daily_vaccinations_raw                 2558
         daily_vaccinations                        0
         total_vaccinations_per_hundred            0
         people_vaccinated_per_hundred             0
         people_fully_vaccinated_per_hundred    3509
         daily_vaccinations_per_million            0
         vaccines                                  0
         source_name                               0
         source_website                            0
         dtype: int64
```

```
In [15]: #people_fully_vaccinated and people_fully_vaccinated_per_hundred greatly correlates w
         #Just filling missing values with 0.
         data.people_fully_vaccinated = data.people_fully_vaccinated.fillna(0)
         data.people_fully_vaccinated_per_hundred = data.people_fully_vaccinated_per_hundred.f
```

```
In [16]: data.isna().sum()

Out[16]: country                                   0
         iso_code                                  0
         date                                      0
         total_vaccinations                        0
         people_vaccinated                         0
         people_fully_vaccinated                   0
         daily_vaccinations_raw                 2558
         daily_vaccinations                        0
         total_vaccinations_per_hundred            0
         people_vaccinated_per_hundred             0
         people_fully_vaccinated_per_hundred       0
         daily_vaccinations_per_million            0
         vaccines                                  0
         source_name                               0
         source_website                            0
         dtype: int64
```

```
In [17]: #daily_vaccinations_raw greatly correlates with daily_vaccinations.
         #Just filling missing values with 0.
         data.daily_vaccinations_raw = data.daily_vaccinations_raw.fillna(0)
```

```
In [18]: data.isna().sum()

Out[18]: country                                0
         iso_code                               0
         date                                   0
         total_vaccinations                     0
         people_vaccinated                      0
         people_fully_vaccinated                0
         daily_vaccinations_raw                 0
         daily_vaccinations                     0
         total_vaccinations_per_hundred         0
         people_vaccinated_per_hundred          0
         people_fully_vaccinated_per_hundred    0
         daily_vaccinations_per_million         0
         vaccines                               0
         source_name                            0
         source_website                         0
         dtype: int64
```

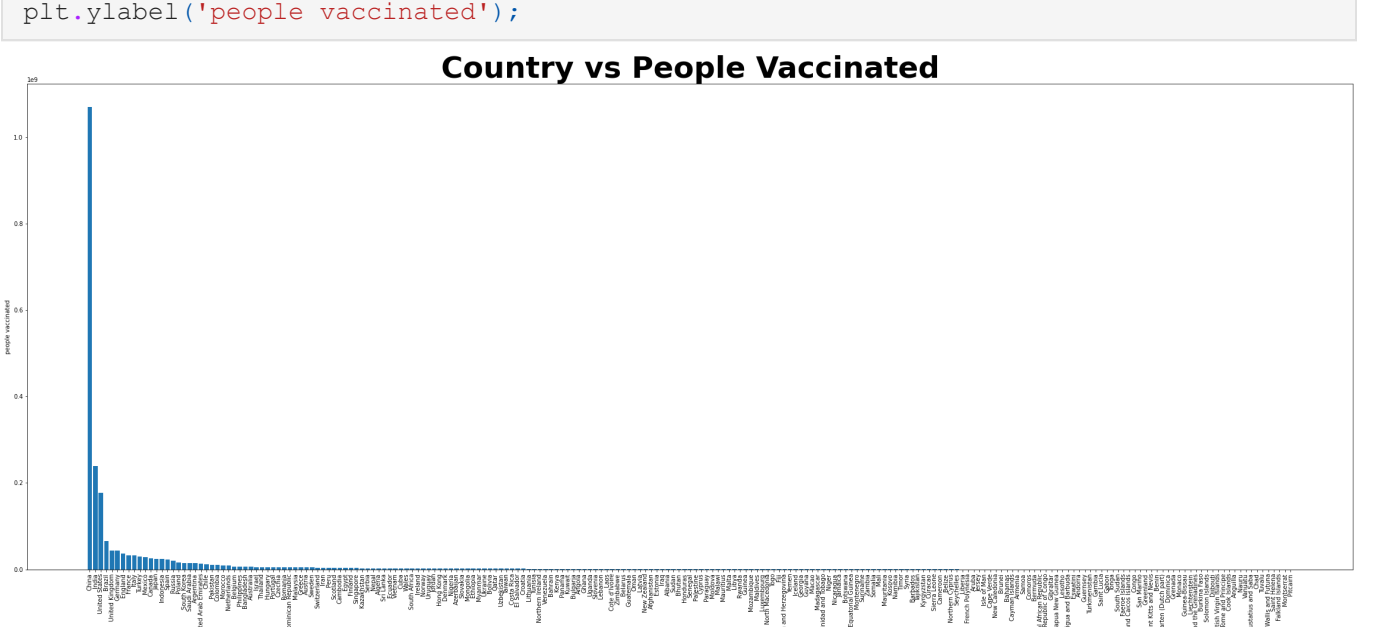
```
In [19]: #finding out which countries have missing iso-code.
         data[data.iso_code.isna()].country.unique()

Out[19]: array([], dtype=object)
```

```
In [20]: #filling missing iso-codes with appropriate ones.
         data[data.country == 'England'] = data[data.country == 'England'].fillna('GB-ENG')
         data[data.country == 'Northern Ireland'] == data[data.country == 'Northern Ireland'].
         data[data.country == 'Scotland'] = data[data.country == 'Scotland'].fillna('GB-SCT')
         data[data.country == 'Wales'] = data[data.country == 'Wales'].fillna('GB-WLS')
         data = data.fillna('NC')
```

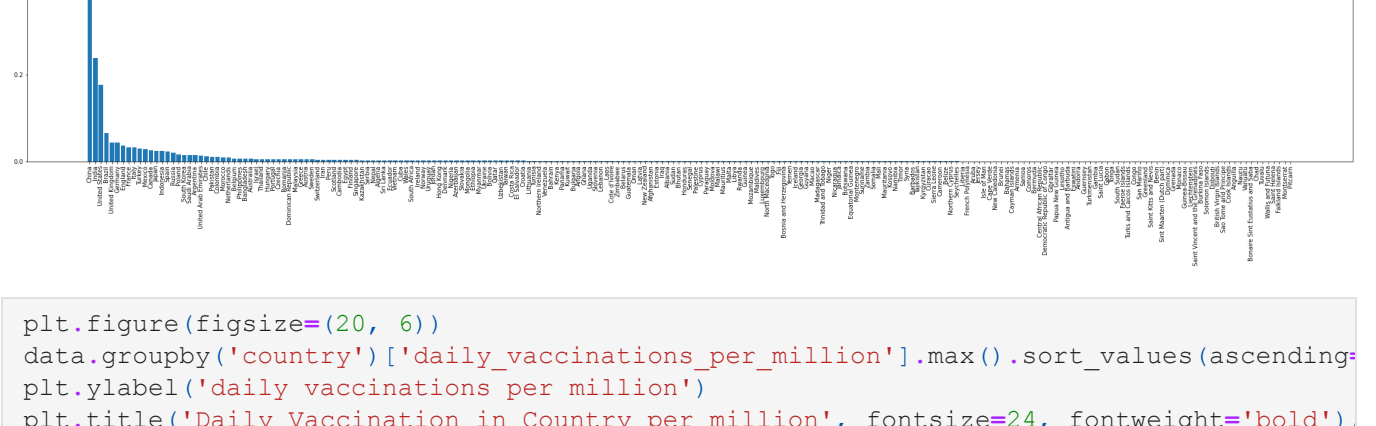
```
In [21]: data.isna().sum()

Out[21]: country                                0
         iso_code                               0
         date                                   0
         total_vaccinations                     0
         people_vaccinated                      0
         people_fully_vaccinated                0
         daily_vaccinations_raw                 0
         daily_vaccinations                     0
         total_vaccinations_per_hundred         0
         people_vaccinated_per_hundred          0
         people_fully_vaccinated_per_hundred    0
         daily_vaccinations_per_million         0
         vaccines                               0
         source_name                            0
         source_website                         0
         dtype: int64
```

Visualizations

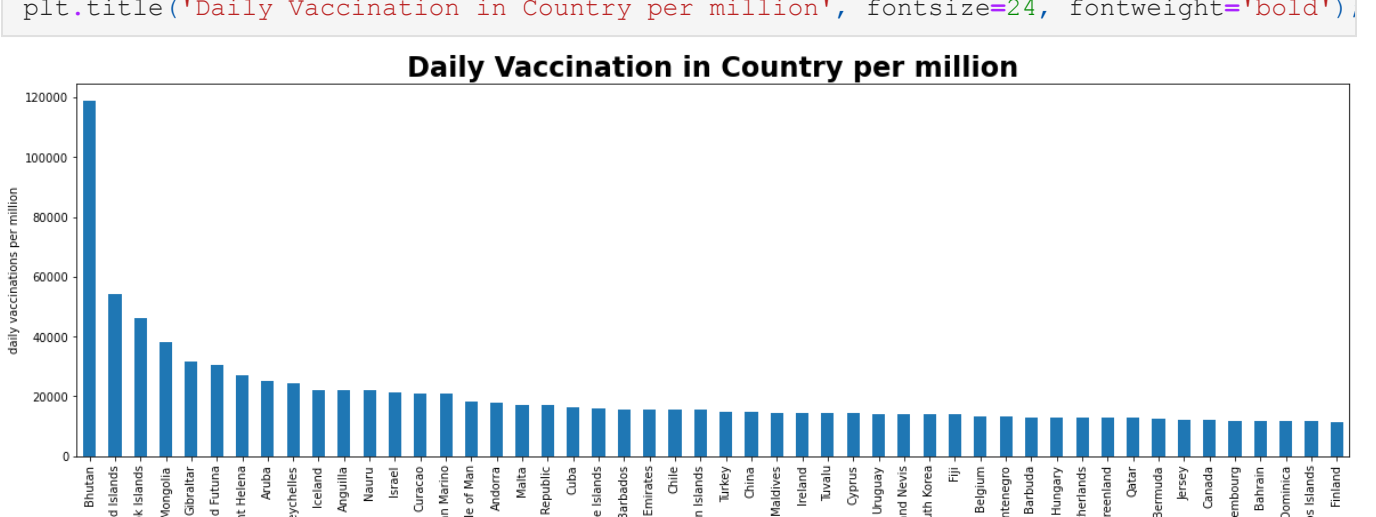
```
In [22]: plt.figure(figsize=(20, 6))
         data.groupby('country')['total_vaccinations'].max().sort_values(ascending=False)[:50]
         plt.title('Country vs Total Vaccination', fontsize=24, fontweight='bold')
         plt.ylabel('Total Vaccinations');
```



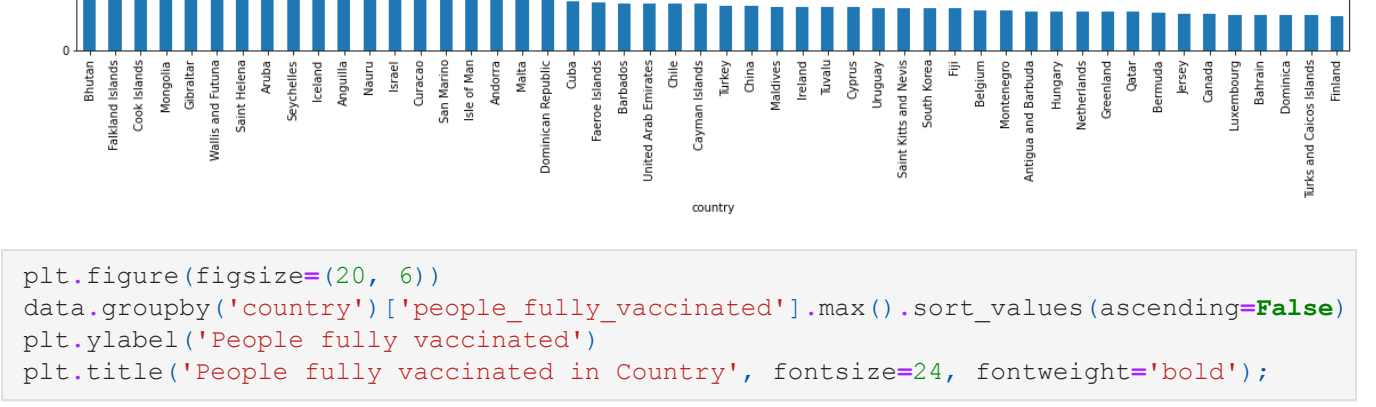

```
In [23]: plt.figure(figsize=(40, 15))
         data1 = data.groupby('country')['people_vaccinated'].max().sort_values(ascending=False
         plt.bar(data1.index, data1);
         plt.xticks(rotation='90')
         plt.yticks()
         plt.title('Country vs People Vaccinated', fontsize=50, fontweight='bold')
         plt.ylabel('people vaccinated');
```




```
In [24]: plt.figure(figsize=(20, 6))
         data.groupby('country')['daily_vaccinations_per_million'].max().sort_values(ascending
         plt.ylabel('daily vaccinations per million')
         plt.title('Daily Vaccination in Country per million', fontsize=24, fontweight='bold')
```




```
In [25]: plt.figure(figsize=(20, 6))
         data.groupby('country')['people_fully_vaccinated'].max().sort_values(ascending=False)
         plt.ylabel('People fully vaccinated')
         plt.title('People fully vaccinated in Country', fontsize=24, fontweight='bold');
```



Working on just Data related to Pakistan

```
In [26]: #Sorting out Pakistan's Data from the complete dataset
         df_pak = data[ data['country'] == 'Pakistan' ]
```

```
In [27]: df_pak.head()
```

Out[27]:

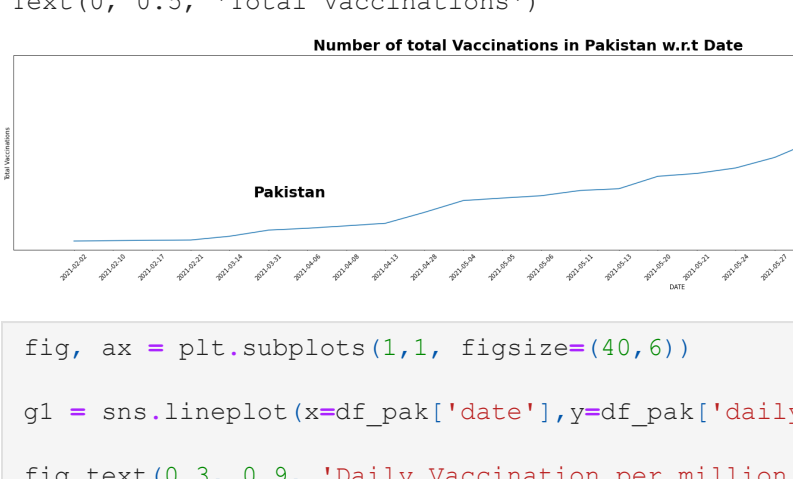
	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_
18114	Pakistan	PAK	2021-02-02	0.0	0.0	0.0	
18122	Pakistan	PAK	2021-02-10	27228.0	27228.0	0.0	
18129	Pakistan	PAK	2021-02-17	52768.0	52768.0	0.0	
18133	Pakistan	PAK	2021-02-21	72882.0	72882.0	0.0	
18154	Pakistan	PAK	2021-03-14	350000.0	350000.0	0.0	

```
In [28]: df_pak.isnull().sum()

Out[28]: country                                0
         iso_code                               0
         date                                   0
         total_vaccinations                     0
         people_vaccinated                      0
         people_fully_vaccinated                0
         daily_vaccinations_raw                 0
         daily_vaccinations                     0
         total_vaccinations_per_hundred         0
         people_vaccinated_per_hundred          0
         people_fully_vaccinated_per_hundred    0
         daily_vaccinations_per_million         0
         vaccines                               0
         source_name                            0
         source_website                         0
         dtype: int64
```

No need to drop any values as no Na values

Visualizations of Pakistan's Dataset

```
In [29]: df_pak[['people_vaccinated', 'people_fully_vaccinated']].plot(ylabel = '(in 1000) vac
         plt.grid()
         plt.title('People fully vaccinated Vs Patially Vaccinated', fontsize=10, fontweight='
```

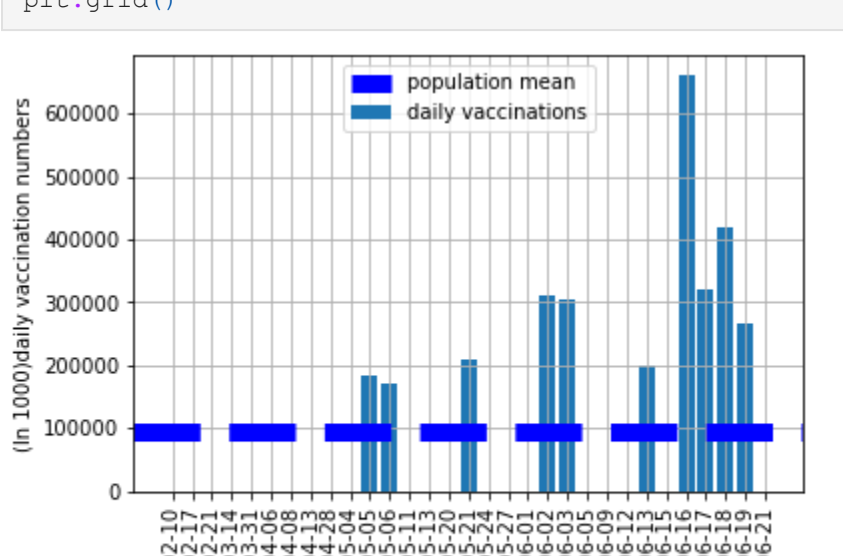



```
In [30]: fig, ax = plt.subplots(1,1, figsize=(40,6))

         g1 = sns.lineplot(x=df_pak['date'],y=df_pak['total_vaccinations'])

         fig.text(0.3, 0.9, 'Number of total Vaccinations in Pakistan w.r.t Date',
                 fontsize=25, fontweight='bold',color='black')
         fig.text(0.265, 0.33, 'Pakistan',
                  fontsize=25, fontweight='bold')
         ax.yaxis.tick_right()
         ax.tick_params(length=1)
         plt.xlabel('DATE')
         plt.xticks(rotation=45)
         plt.ylabel('Total Vaccinations')

Out[30]: Text(0, 0.5, 'Total Vaccinations')
```


```
In [31]: fig, ax = plt.subplots(1,1, figsize=(40,6))

         g1 = sns.lineplot(x=df_pak['date'],y=df_pak['daily_vaccinations_per_million'])

         fig.text(0.3, 0.9, 'Daily Vaccination per million in Pakistan',
                 fontsize=25, fontweight='bold',color='black')
         fig.text(0.265, 0.33, 'Pakistan',
                  fontsize=25, fontweight='bold')
         plt.xticks()
         ax.yaxis.tick_right()
         ax.tick_params(length=1)
         plt.xlabel('DATE')
         plt.ylabel('Daily Vaccinations per Million')

Out[31]: Text(0, 0.5, 'Daily Vaccinations per Million')
```



```
In [32]: plt.bar(df_pak['date'][1:], df_pak['daily_vaccinations_raw'][1:], label = 'daily vacc
         plt.xlabel('Date')
         plt.ylabel('(In 1000)daily vaccination numbers')
         plt.axhline(y = np.mean(df_pak['daily_vaccinations_raw']), color = 'blue', ls = '--',
         plt.legend(loc = 9)
         plt.xticks(rotation = 90)
         plt.grid()
```



Prediction

```
In [33]: t_cols = ['date', 'total_vaccinations']
         timeseries_cov = data[t_cols].groupby('date').sum()[4:-1]
         print('p-value : {}'.format(sm.tsa.stattools.adfuller(timeseries_cov)[1]))

         p-value : 1.0
```

p-value is extremely high as = 1 so using Box-Cox transformation.

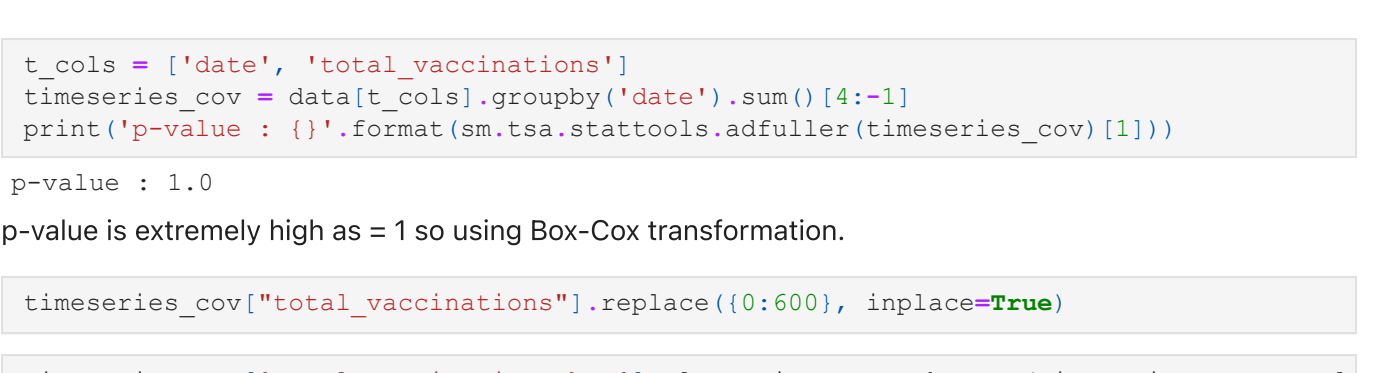
```
In [34]: timeseries_cov["total_vaccinations"].replace({0:600}, inplace=True)
```

```
In [35]: timeseries_cov['total_vaccinations_box'], l = scipy.stats.boxcox(timeseries_cov.total_
```

```
In [36]: print('p-value : {}'.format(sm.tsa.stattools.adfuller(timeseries_cov.drop(columns=['t

         p-value : 0.04694406645626073
```

p-value is less than 0.5, so now the timeseries is stationary

```
In [37]: plt.figure(figsize=(20,7))
         timeseries_cov.total_vaccinations_box.plot()
         plt.xticks(rotation=45)
         plt.show()
```

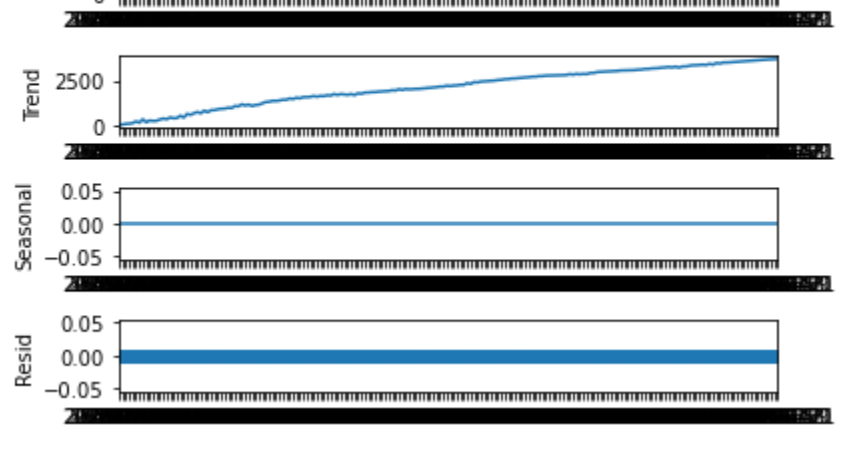


```
In [38]: timeseries_cov['total_vaccinations_box_diff1int2'] = timeseries_cov.total_vaccination
```

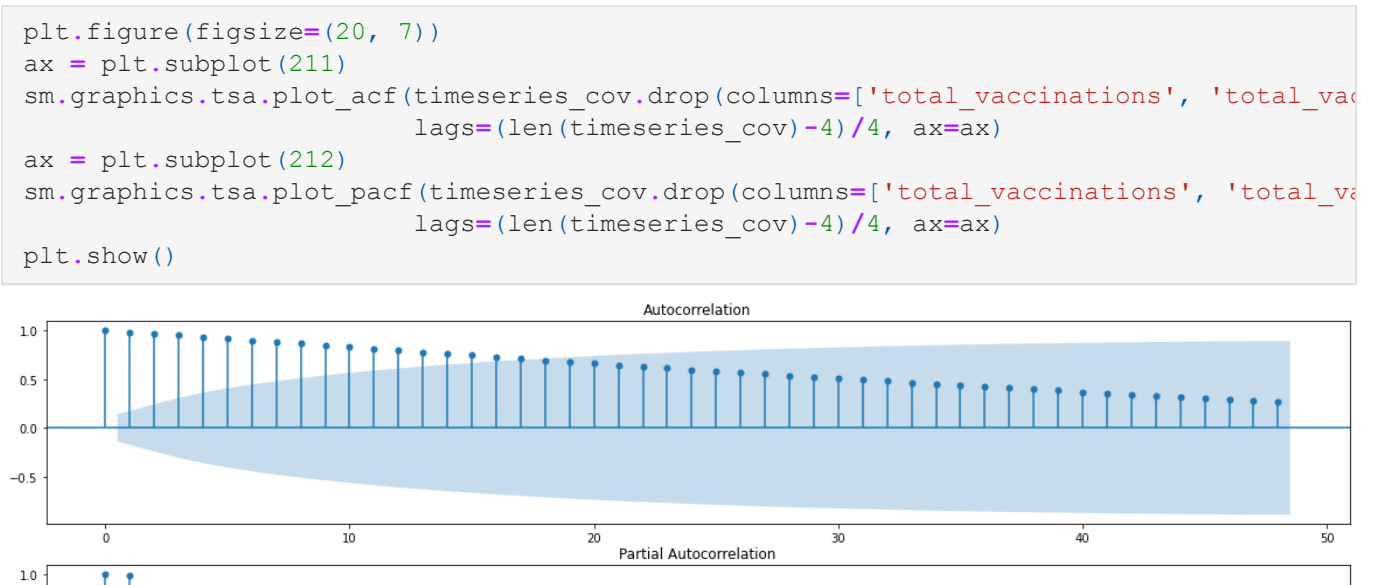
```
In [39]: timeseries_cov['total_vaccinations_box_diff2int2'] = timeseries_cov['total_vaccinatio
```

```
In [40]: print('p-value : {}'.format(sm.tsa.stattools.adfuller(timeseries_cov.drop(columns=['t

         p-value : 0.04175462855609853
```

```
In [41]: sm.tsa.seasonal_decompose(timeseries_cov.total_vaccinations_box_diff2int2[4:], period
         plt.show()
```




```
In [42]: plt.figure(figsize=(20, 7))
         ax = plt.subplot(211)
         sm.graphics.tsa.plot_acf(timeseries_cov.drop(columns=['total_vaccinations', 'total_va
                                  lags=(len(timeseries_cov)-4)/4, ax=ax)
         ax = plt.subplot(212)
         sm.graphics.tsa.plot_pacf(timeseries_cov.drop(columns=['total_vaccinations', 'total_v
                                  lags=(len(timeseries_cov)-4)/4, ax=ax)
         plt.show()
```



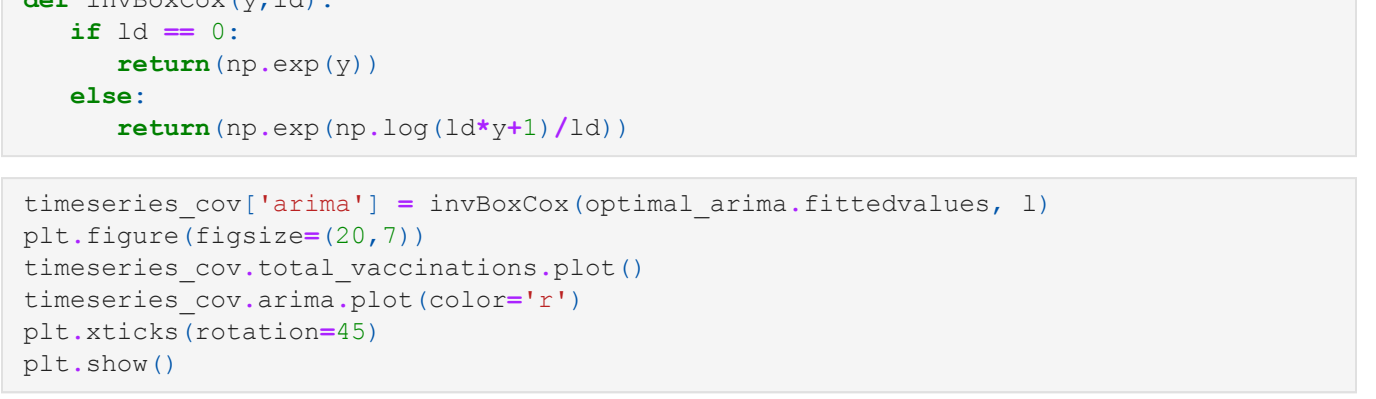

```
In [43]: d = 0
         D = 2
```

```
In [44]: %%time
         results = []
         best_aic = float('inf')

         parameters = list(product(np.arange(0, 14), np.arange(0, 14), np.arange(0, 14), np.ar

         for param in tqdm(parameters):
             try:
                 arima = sm.tsa.statespace.SARIMAX(timeseries_cov.total_vaccinations_box, orde
                                                   seasonal_order=(param[2], D, param[3], 2)).
             except:
                 continue
             aic = arima.aic
             if aic < best_aic:
                 optimal_arima = arima
                 best_aic = aic
                 best_param = param
             results.append([param, optimal_arima.aic])

         100%|██████████| 38416/38416 [49:17<00:00, 12.99it/s]
         CPU times: user 4h 56min 32s, sys: 39min 8s, total: 5h 35min 41s
         Wall time: 49min 17s
```

```
In [45]: print(optimal_arima.summary())

                                      SARIMAX Results
         ==============================================================================
         ====
         Dep. Variable:           total_vaccinations_box   No. Observations:
         198
         Model:             SARIMAX(0, 0, 8)x(3, 2, [], 2)   Log Likelihood                 -80
         5.681
         Date:                          Tue, 29 Jun 2021   AIC                           163
         5.362
         Time:                                  23:02:34   BIC                           167
         4.576
         Sample:                              12-06-2020   HQIC                          165
         1.241
                                            - 06-21-2021

         Covariance Type:                            opg

         ==============================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
         ------------------------------------------------------------------------------
         ma.L1          0.2380      0.118      2.021      0.043       0.007       0.469
         ma.L2         -1.2260      0.119    -10.337      0.000      -1.458      -0.994
         ma.L3         -0.1373      0.090     -1.518      0.129      -0.314       0.040
         ma.L4          0.8138      0.130      6.248      0.000       0.559       1.069
         ma.L5          0.2305      0.101      2.271      0.023       0.032       0.429
         ma.L6         -1.1297      0.148     -7.628      0.000      -1.420      -0.839
         ma.L7         -0.2195      0.117     -1.882      0.060      -0.448       0.009
         ma.L8          0.6682      0.128      5.240      0.000       0.418       0.918
         ar.S.L2       -0.4718      0.076     -6.174      0.000      -0.622      -0.322
         ar.S.L4       -0.9521      0.050    -18.952      0.000      -1.051      -0.854
         ar.S.L6       -0.0843      0.063     -1.337      0.181      -0.208       0.039
         sigma2       206.9624     28.774      7.193      0.000     150.567     263.358
         ===================================================================================
         Ljung-Box (L1) (Q):                   0.15   Jarque-Bera (JB):                58.34
         Prob(Q):                              0.70   Prob(JB):                         0.00
         Heteroskedasticity (H):               0.22   Skew:                            -0.12
         Prob(H) (two-sided):                  0.00   Kurtosis:                         5.68
         ===================================================================================

         Warnings:
         [1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
In [62]: def invBoxCox(y,ld):
            if ld == 0:
               return(np.exp(y))
            else:
               return(np.exp(np.log(ld*y+1)/ld))
```

```
In [63]: timeseries_cov['arima'] = invBoxCox(optimal_arima.fittedvalues, l)
         plt.figure(figsize=(20,7))
         timeseries_cov.total_vaccinations.plot()
         timeseries_cov.arima.plot(color='r')
         plt.xticks(rotation=45)
         plt.show()
```



Making Prediction that how many people will be vaccinated in the next 15 days?

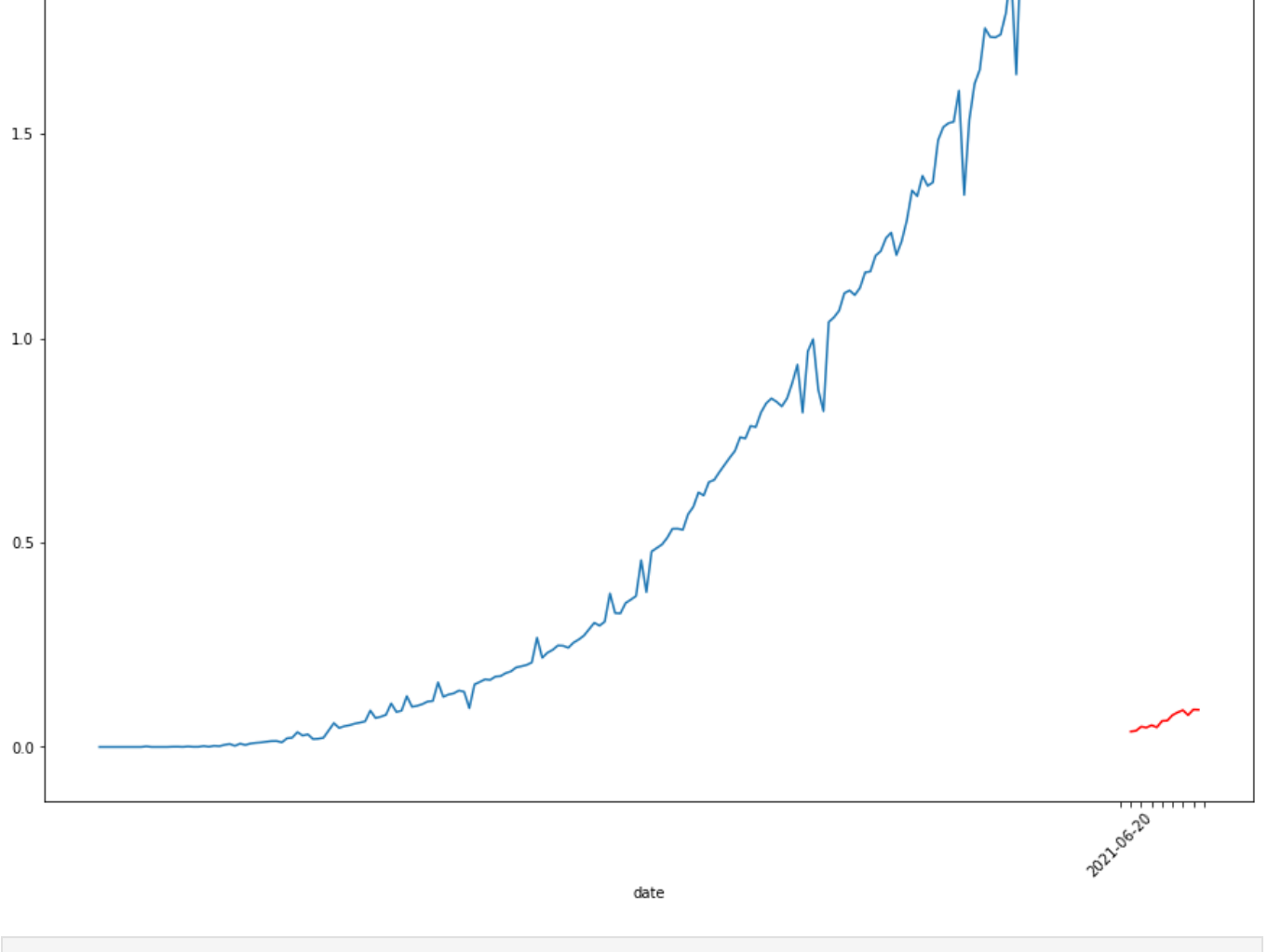
```
In [64]: date = ['2021-07-'+str(x) for x in range(1,15)]
         timeseries = timeseries_cov['total_vaccinations']
         pred_df = pd.DataFrame(index=date)
         pred_df['total_vaccinations'] = invBoxCox(optimal_arima.predict(start=44, end=57).val
         timeseries = pd.concat([timeseries, pred_df])
```

```
In [66]: timeseries.drop(columns=[0])[-14:]
```

Out[66]:

	total_vaccinations
2021-07-1	3.789318e+07
2021-07-2	3.971775e+07
2021-07-3	4.993734e+07
2021-07-4	4.755530e+07
2021-07-5	5.334872e+07
2021-07-6	4.817139e+07
2021-07-7	6.375136e+07
2021-07-8	6.451675e+07
2021-07-9	7.803701e+07
2021-07-10	8.486346e+07
2021-07-11	9.013140e+07
2021-07-12	7.812185e+07
2021-07-13	9.168727e+07
2021-07-14	9.117121e+07

```
In [67]: timeseries_cov['arima'] = invBoxCox(optimal_arima.fittedvalues, l)
         plt.figure(figsize=(15,15))
         timeseries.total_vaccinations.plot(color='r')
         timeseries_cov.total_vaccinations.plot()
         plt.xticks(rotation=45)
         plt.show()
```



```
In [ ]:
```