Covid-19 Vaccination Campaign in Germany

The data used here were provided by Robert Koch Institute and the German federal ministry of Health.

These institutions publish the datasets and some analysis on the page impfdashboard.de.

Setup

Imports

In [71]:
```
# standard library
import datetime
import math
```

In [72]:
```
# third party
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
import seaborn
```

Date this Notebook was run

In [73]:
```
today = datetime.datetime.today().strftime('%Y-%m-%d')
today
```

Out[73]:
```
'2021-09-05'
```

Set Defaults

In [74]:
```
# style like ggplot in R
plt.style.use('ggplot')
```

In [75]:
```
# Avoid cutting off part of the axis labels, see:
# https://stackoverflow.com/questions/6774086/why-is-my-xlabel-cut-off-in-my-matplotlib-plot
plt.rcParams.update({'figure.autolayout': True})
```

In [76]:
```
population_germany = 83_200_000
```

Get and Transform Data

In [77]:
```
vaccination_data_permalink = 'https://impfdashboard.de/static/data/germany_vaccinations_timeseries_v2.tsv'
vaccinations = pd.read_csv(
    vaccination_data_permalink,
    sep="\t")
```

Drop unnecessary / misleading columns

List all columns:

In [78]:
```
vaccinations.columns
```

Out[78]:
```
Index(['date', 'dosen_kumulativ', 'dosen_biontech_kumulativ',
       'dosen_biontech_erst_kumulativ', 'dosen_biontech_zweit_kumulativ',
       'dosen_moderna_kumulativ', 'dosen_moderna_erst_kumulativ',
       'dosen_moderna_zweit_kumulativ', 'dosen_astra_kumulativ',
       'dosen_astra_erst_kumulativ', 'dosen_astra_zweit_kumulativ',
       'dosen_johnson_kumulativ', 'dosen_erst_kumulativ',
       'dosen_zweit_kumulativ', 'dosen_differenz_zum_vortag',
       'dosen_erst_differenz_zum_vortag', 'dosen_zweit_differenz_zum_vortag',
       'personen_erst_kumulativ', 'personen_voll_kumulativ', 'impf_quote_erst',
       'impf_quote_voll', 'dosen_dim_kumulativ', 'dosen_kbv_kumulativ',
       'indikation_alter_dosen', 'indikation_beruf_dosen',
       'indikation_medizinisch_dosen', 'indikation_pflegeheim_dosen',
       'indikation_alter_erst', 'indikation_beruf_erst',
       'indikation_medizinisch_erst', 'indikation_pflegeheim_erst',
       'indikation_alter_voll', 'indikation_beruf_voll',
       'indikation_medizinisch_voll', 'indikation_pflegeheim_voll'],
      dtype='object')
```

Columns with names starting with 'indikation_' will not be analyzed as the data providers stopped updating them.

In [79]:
```
cols_to_drop = vaccinations.columns[vaccinations.columns.str.contains('indikation_')]
vaccinations.drop(columns=cols_to_drop, inplace=True)
```

Some more columns can be dropped, as there is no interest in analyzing differences on a vaccine level - especially since in some cases vaccines were mixed.

In [80]:
```
more_cols_to_drop = ['dosen_biontech_erst_kumulativ', 'dosen_biontech_zweit_kumulativ',
```

```
                      'dosen_moderna_erst_kumulativ', 'dosen_moderna_zweit_kumulativ',
                      'dosen_astra_erst_kumulativ', 'dosen_astra_zweit_kumulativ']
vaccinations.drop(columns=more_cols_to_drop, inplace=True)
```

Some columns are labeled misleadingly. As stated by the data provider the columns `personen_erst_kumulativ` and `impf_quote_erst` contain people vaccinated with the Johnson & Johnson vaccine. As this requires only one shot. the same persons are included in `personen_voll_kumulativ` . Therefore more columns are dropped and recalculated later.

In [81]:
```
vaccinations.drop(columns=['impf_quote_erst', 'impf_quote_voll'], inplace=True)
```

Convert datatype of date column

In [82]:
```
vaccinations.iloc[ : , [0]] = vaccinations.iloc[ : , [0]].apply(pd.to_datetime)
```

Show Data

In [83]:
```
vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 15 columns):
 #   Column                           Non-Null Count  Dtype         
---  ------                           --------------  -----         
 0   date                             250 non-null    datetime64[ns]
 1   dosen_kumulativ                  250 non-null    int64         
 2   dosen_biontech_kumulativ         250 non-null    int64         
 3   dosen_moderna_kumulativ          250 non-null    int64         
 4   dosen_astra_kumulativ            250 non-null    int64         
 5   dosen_johnson_kumulativ          250 non-null    int64         
 6   dosen_erst_kumulativ             250 non-null    int64         
 7   dosen_zweit_kumulativ            250 non-null    int64         
 8   dosen_differenz_zum_vortag       250 non-null    int64         
 9   dosen_erst_differenz_zum_vortag  250 non-null    int64         
 10  dosen_zweit_differenz_zum_vortag 250 non-null    int64         
 11  personen_erst_kumulativ          250 non-null    int64         
 12  personen_voll_kumulativ          250 non-null    int64         
 13  dosen_dim_kumulativ              250 non-null    int64         
 14  dosen_kbv_kumulativ              250 non-null    int64         
dtypes: datetime64[ns](1), int64(14)
memory usage: 29.4 KB
```

In [84]:
```
vaccinations.tail(3)
```

Out[84]:

	date	dosen_kumulativ	dosen_biontech_kumulativ	dosen_moderna_kumulativ	dosen_astra_kumulativ	dosen_johnson_kumulativ	dosen_erst_kumu
247	2021-08-31	101921487	77015789	9398975	12645995	2856281	5431
248	2021-09-01	102198167	77254708	9412177	12648398	2869614	5443
249	2021-09-02	102437852	77458405	9423676	12650336	2881838	5454

Check Validity

In [85]:
```
# get the last row / the newest available data
last_row = vaccinations.tail(1)
```

In [86]:
```
doses_used = last_row['dosen_kumulativ']
doses_used
```

Out[86]:
```
249    102437852
Name: dosen_kumulativ, dtype: int64
```

In [87]:
```
# The number of person having been vaccinated at least once, includes those fully vaccinated
at_least_once = last_row['personen_erst_kumulativ']
fully_vaccinated_people = last_row['personen_voll_kumulativ']
partially_vaccinated_people = at_least_once - fully_vaccinated_people
# The johnson & Johnson vaccine is the only one used in Germany that only needs a single shot:
johnson_doses = last_row['dosen_johnson_kumulativ']
```

In [88]:
```
# Must be exactly 0
result_substraction = doses_used - partially_vaccinated_people - (fully_vaccinated_people - johnson_doses) * 2 - johnsc
result_substraction
```

Out[88]:
```
249    23597
dtype: int64
```

In [89]:
```
result_substraction ==  0
```

Out[89]:
```
249    False
dtype: bool
```

Calculate columns

In [90]:
```
vaccinations['partly vaccinated'] = round(
    (vaccinations['personen_erst_kumulativ'] - vaccinations['personen_voll_kumulativ']) * 100 / population_germany,
    2)
```

In [91]:
```
vaccinations['fully vaccinated'] = round(
    vaccinations['personen_voll_kumulativ'] * 100 / population_germany,
    2)
```

In [92]:
```
vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 17 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   date                             250 non-null    datetime64[ns]
 1   dosen_kumulativ                  250 non-null    int64
 2   dosen_biontech_kumulativ         250 non-null    int64
 3   dosen_moderna_kumulativ          250 non-null    int64
 4   dosen_astra_kumulativ            250 non-null    int64
 5   dosen_johnson_kumulativ          250 non-null    int64
 6   dosen_erst_kumulativ             250 non-null    int64
 7   dosen_zweit_kumulativ            250 non-null    int64
 8   dosen_differenz_zum_vortag       250 non-null    int64
 9   dosen_erst_differenz_zum_vortag  250 non-null    int64
 10  dosen_zweit_differenz_zum_vortag 250 non-null    int64
 11  personen_erst_kumulativ          250 non-null    int64
 12  personen_voll_kumulativ          250 non-null    int64
 13  dosen_dim_kumulativ              250 non-null    int64
 14  dosen_kbv_kumulativ              250 non-null    int64
 15  partly vaccinated                250 non-null    float64
 16  fully vaccinated                 250 non-null    float64
dtypes: datetime64[ns](1), float64(2), int64(14)
memory usage: 33.3 KB
```

In [93]:
```
vaccinations.tail(3)
```

Out[93]:

	date	dosen_kumulativ	dosen_biontech_kumulativ	dosen_moderna_kumulativ	dosen_astra_kumulativ	dosen_johnson_kumulativ	dosen_erst_kumu
247	2021-08-31	101921487	77015789	9398975	12645995	2856281	5431

	date	dosen_kumulativ	dosen_biontech_kumulativ	dosen_moderna_kumulativ	dosen_astra_kumulativ	dosen_johnson_kumulativ	dosen_erst_kumu
248	2021-09-01	102198167	77254708	9412177	12648398	2869614	5443(
249	2021-09-02	102437852	77458405	9423676	12650336	2881838	5454:

Last Update

Often the data is not updated on weekends, so get the highest date in the dataset.

In [94]:
```
last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
last_update
```

Out[94]: '2021-09-02'

Doses Used

In [95]:
```
doses = vaccinations.loc[ : , ['date', 'dosen_differenz_zum_vortag']]
# Rename columns
doses.columns = ['date', 'doses used']
```

In [96]:
```
# Scale number of doses as millions
doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

In [97]:
```
doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```

Out[97]:

	doses used
date	
2021-09-02	0.239685

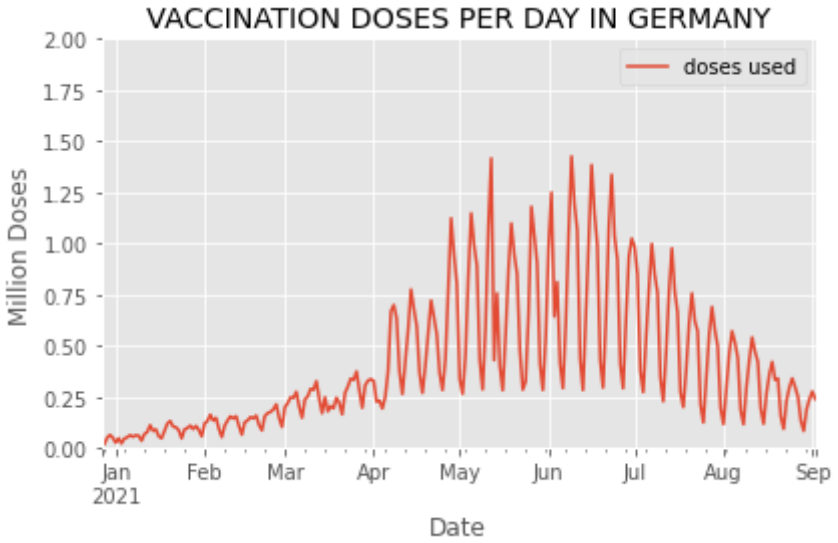
In [98]:
```
# What is the highest number of doses used in a day?
```

```
max_doses_daily = max(doses_daily['doses used'])
max_doses_daily
```

Out[98]: 1.426177

In [99]:
```
doses_daily.plot(
    ylim=(0,math.ceil(max_doses_daily)),
    xlabel='Date',
    ylabel='Million Doses',
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

Out[99]: <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>



Doses per Weekday (in the last 6 weeks)

In [100…
```
last_6_weeks = doses.tail(42)
```

In [101…
```
# Yields a warning, but exactly like the docs prescribe and it works
# https://pandas.pydata.org/docs/getting_started/intro_tutorials/05_add_columns.html
last_6_weeks['weekday'] = last_6_weeks['date'].dt.day_name()
```

```
<ipython-input-101-45013977109e>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  last_6_weeks['weekday'] = last_6_weeks['date'].dt.day_name()
```

In [102…
```
# check:
last_6_weeks.tail(3)
```

Out[102…

	date	doses used	weekday
247	2021-08-31	0.239112	Tuesday
248	2021-09-01	0.276680	Wednesday
249	2021-09-02	0.239685	Thursday

In [103…
```
# drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

In [104…
```
#last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

Out[104…

	doses used	weekday
247	0.239112	Tuesday
248	0.276680	Wednesday
249	0.239685	Thursday

In [105…
```
pivot_table =last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

Out[105…

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
245	NaN	NaN	NaN	0.083448	NaN	NaN	NaN
246	NaN	0.189108	NaN	NaN	NaN	NaN	NaN
247	NaN	NaN	NaN	NaN	NaN	0.239112	NaN
248	NaN	NaN	NaN	NaN	NaN	NaN	0.27668
249	NaN	NaN	NaN	NaN	0.239685	NaN	NaN

In [106…
```
# Reorder the columns
```

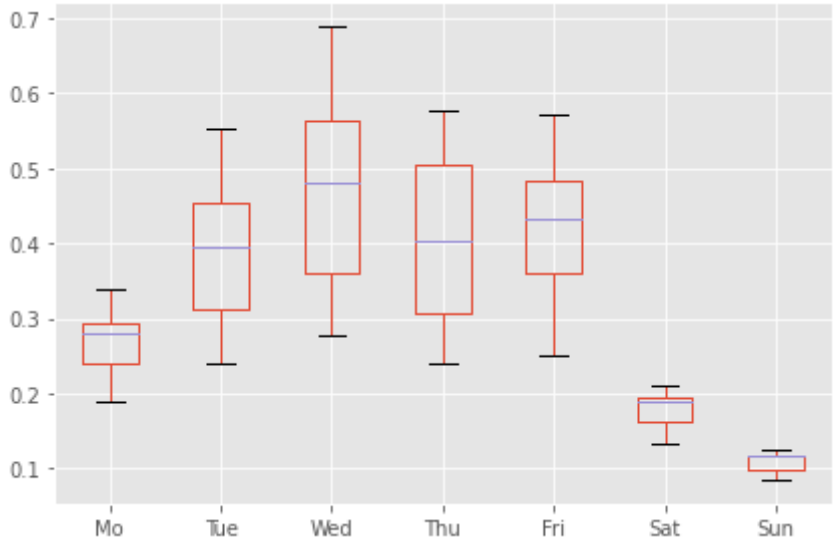
```
pivot_table = pivot_table[['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']]
# Rename the columns
pivot_table.columns=['Mo', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
pivot_table.tail()
```

Out[106…

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
245	NaN	NaN	NaN	NaN	NaN	NaN	0.083448
246	0.189108	NaN	NaN	NaN	NaN	NaN	NaN
247	NaN	0.239112	NaN	NaN	NaN	NaN	NaN
248	NaN	NaN	0.27668	NaN	NaN	NaN	NaN
249	NaN	NaN	NaN	0.239685	NaN	NaN	NaN

In [107…

```
weekday_boxplot = pivot_table.boxplot()
```



In [108…

```
fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

In [109…

```
# W-Mon in order to start the week on a Monday, see:
# https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#anchored-offsets
```

```
doses_weekly = doses.groupby(pd.Grouper(key='date',freq='W-Mon')).sum()
doses_weekly.columns = ['million doses used']
doses_weekly.tail()
```

Out[109…

date	million doses used
2021-08-09	2.584778
2021-08-16	2.440778
2021-08-23	1.935928
2021-08-30	1.591766
2021-09-06	0.755477

In [110…

```
# What is the highest number of doses used in a week?
max_million_doses_weekly = max(doses_weekly['million doses used'])
max_million_doses_weekly
```

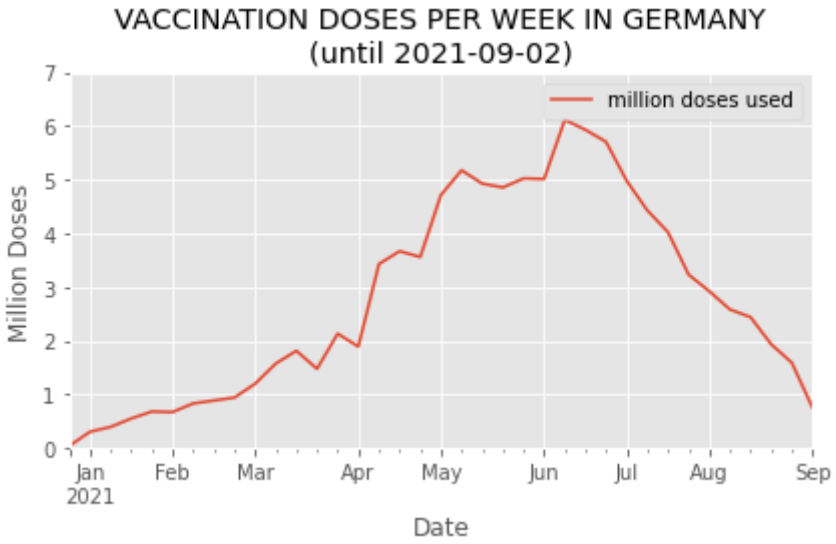
Out[110… 6.116577

In [111…

```
doses_weekly.plot(
    ylim=(0, math.ceil(max_million_doses_weekly)),
    xlabel='Date',
    ylabel='Million Doses',
    title=f"VACCINATION DOSES PER WEEK IN GERMANY\n(until {last_update})")
```

Out[111…

```
<AxesSubplot:title={'center':'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-09-02)'}, xlabel='Date', ylabel='Million Doses'>
```



Doses per Month

In [112…
```
# M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date',freq='M')).sum()
doses_monthly.tail()
```

Out[112…

date	doses used
2021-05-31	21.039689
2021-06-30	24.726359
2021-07-31	17.220404
2021-08-31	9.202679
2021-09-30	0.516365

In [113…
```
max_doses_monthly = max(doses_monthly['doses used'])
max_doses_monthly
doses_monthly['month'] = doses_monthly.index.strftime('%B')
doses_monthly['year'] =  doses_monthly.index.strftime('%Y')
doses_monthly['label'] = doses_monthly['month'] + ' ' + doses_monthly['year']
doses_monthly.drop(columns=['month', 'year'], inplace=True)
```

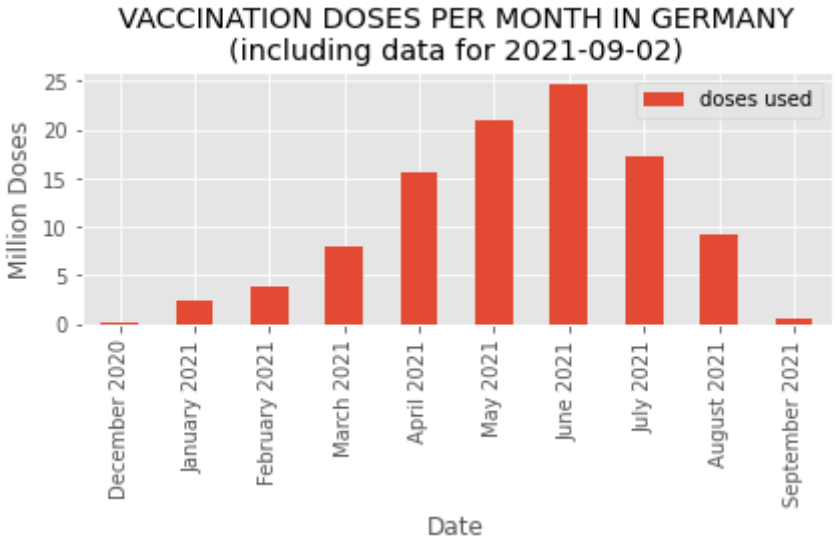
```
doses_monthly.set_index('label', inplace=True)
doses_monthly.tail(6)
```

Out[113…

label	doses used
April 2021	15.548745
May 2021	21.039689
June 2021	24.726359
July 2021	17.220404
August 2021	9.202679
September 2021	0.516365

In [114…

```
monthly_plot = doses_monthly.plot.bar(
    ylim=(0,math.ceil(max_doses_monthly) + 1),
    xlabel='Date',
    ylabel='Million Doses',
    title=f"VACCINATION DOSES PER MONTH IN GERMANY\n(including data for {last_update})")
```



In [115…

```
fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_germany.png')
```

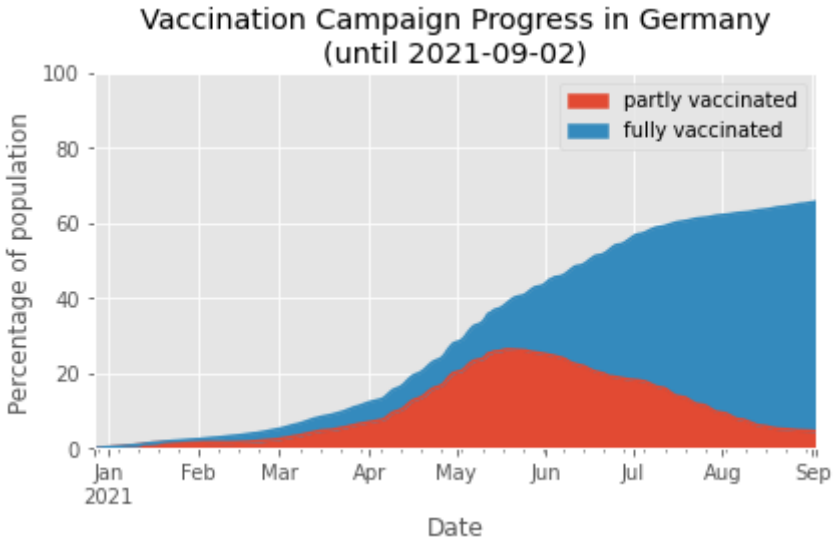
Vaccination Campaign Progress

In [116…
```
doses_cumulative = vaccinations.loc[ : , ['date', 'partly vaccinated', 'fully vaccinated']]
doses_cumulative.set_index('date', inplace=True)
doses_cumulative.tail(3)
```

Out[116…

date	partly vaccinated	fully vaccinated
2021-08-31	4.64	60.64
2021-09-01	4.59	60.84
2021-09-02	4.55	61.00

In [117…
```
doses_area_plot = doses_cumulative.plot.area(
    ylim=(0,100),
    xlabel='Date',
    ylabel='Percentage of population',
    title=f"Vaccination Campaign Progress in Germany\n(until {last_update})")
```



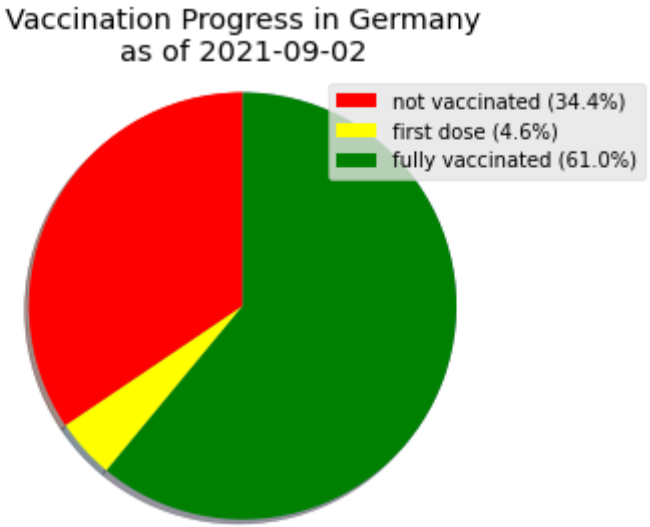
In [118…
```
fig = doses_area_plot.get_figure()
fig.savefig('img/vaccinations_germany_area_plot.png')
```

As of Today

In [119…
```
# get the last line of the data
current_state = doses_cumulative.iloc[-1]
current_state
```

Out[119…
```
partly vaccinated     4.55
fully vaccinated     61.00
Name: 2021-09-02 00:00:00, dtype: float64
```

In [120…
```
percentage_not_vacc = 100 - current_state['partly vaccinated'] - current_state['fully vaccinated']
labels = [f"not vaccinated ({round(percentage_not_vacc, 1)}%)",
          f"first dose ({round(current_state['partly vaccinated'], 1)}%)",
          f"fully vaccinated ({round(current_state['fully vaccinated'], 1)}%)"]
colors = ['red', 'yellow', 'green']
sizes = [percentage_not_vacc,
         current_state['partly vaccinated'],
         current_state['fully vaccinated']]
fig1, ax1 = plt.subplots()
ax1.pie(sizes, shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
patches, texts = plt.pie(sizes, colors=colors, startangle=90)
plt.legend(patches, labels, loc="best")
plt.title(f"Vaccination Progress in Germany\nas of {last_update}")
# plt.savefig must be before show()
# BEWARE plt.savefig must be in the same Jupyter code cell that creates the graph!
# See comment by ijoseph here:
# https://stackoverflow.com/questions/9012487/matplotlib-pyplot-savefig-outputs-blank-image
plt.savefig('img/vaccination_in_germany_pie.png', bbox_inches='tight')
plt.show()
```



Vaccines in Use

In [121…
```
vaccinations.columns
```

Out[121…
```
Index(['date', 'dosen_kumulativ', 'dosen_biontech_kumulativ',
       'dosen_moderna_kumulativ', 'dosen_astra_kumulativ',
       'dosen_johnson_kumulativ', 'dosen_erst_kumulativ',
       'dosen_zweit_kumulativ', 'dosen_differenz_zum_vortag',
       'dosen_erst_differenz_zum_vortag', 'dosen_zweit_differenz_zum_vortag',
       'personen_erst_kumulativ', 'personen_voll_kumulativ',
       'dosen_dim_kumulativ', 'dosen_kbv_kumulativ', 'partly vaccinated',
       'fully vaccinated'],
      dtype='object')
```

In [122…
```
vaccine_use = vaccinations.loc[ : , ['date', 'dosen_biontech_kumulativ',
                                     'dosen_moderna_kumulativ',
                                     'dosen_astra_kumulativ',
                                     'dosen_johnson_kumulativ']]
# Rename columns
vaccine_use.columns = ['date', 'BioNTech', 'Moderna', 'AstraZeneca', 'Johnson & Johnson']
# make 'date' an index
vaccine_use.set_index('date', inplace=True)
# divide columns by 1 million
vaccine_use["BioNTech"] = vaccine_use["BioNTech"] / 1_000_000
vaccine_use["Moderna"] = vaccine_use["Moderna"] / 1_000_000
```

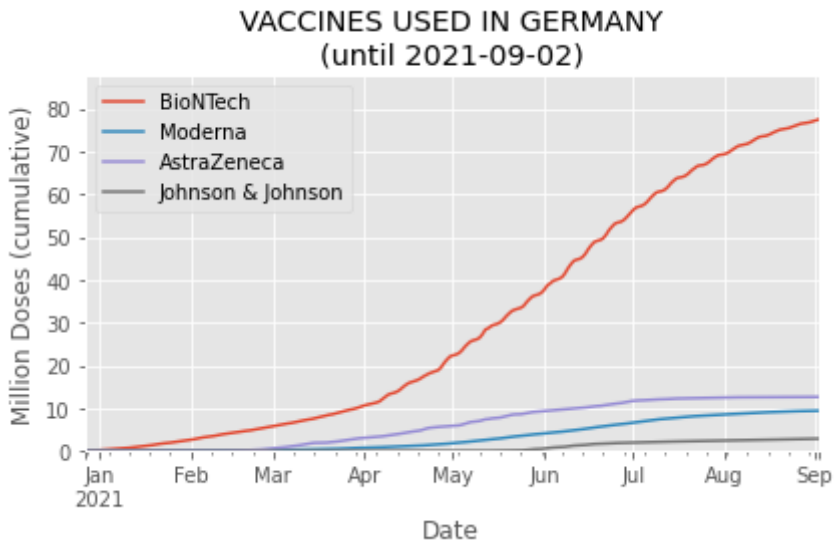
```
vaccine_use["AstraZeneca"] = vaccine_use["AstraZeneca"] / 1_000_000
vaccine_use["Johnson & Johnson"] = vaccine_use["Johnson & Johnson"] / 1_000_000
vaccine_use.tail(3)
```

Out[122…

date	BioNTech	Moderna	AstraZeneca	Johnson & Johnson
2021-08-31	77.015789	9.398975	12.645995	2.856281
2021-09-01	77.254708	9.412177	12.648398	2.869614
2021-09-02	77.458405	9.423676	12.650336	2.881838

In [123…

```
vaccines_used = vaccine_use.plot(
    # as it is cumulative, the last row must contain the single highest number
    ylim=(0,math.ceil(max(vaccine_use.iloc[-1]))+10),
    xlabel='Date',
    ylabel='Million Doses (cumulative)',
    title=f"VACCINES USED IN GERMANY\n(until {last_update})")
```



In [124…

```
fig = vaccines_used.get_figure()
fig.savefig('img/vaccines_used_in_germany.png')
```

Vaccination Centers versus Doctor’s Practices

In [125…
```
vaccinations.tail()
```

Out[125…

	date	dosen_kumulativ	dosen_biontech_kumulativ	dosen_moderna_kumulativ	dosen_astra_kumulativ	dosen_johnson_kumulativ	dosen_erst_kumu
245	2021-08-29	101493267	76653993	9366275	12642454	2830466	5412
246	2021-08-30	101682375	76810151	9383233	12644336	2842978	5420
247	2021-08-31	101921487	77015789	9398975	12645995	2856281	5431
248	2021-09-01	102198167	77254708	9412177	12648398	2869614	5443
249	2021-09-02	102437852	77458405	9423676	12650336	2881838	5454

In [126…
```
by_place = vaccinations.loc[ : , ['date', 'dosen_dim_kumulativ', 'dosen_kbv_kumulativ']]
by_place.columns = ['date', 'vaccination centers', 'practices']
```

In [127…
```
by_place['vaccination centers daily'] = by_place['vaccination centers'].diff()
by_place['practices daily'] = by_place['practices'].diff()
```

In [128…
```
by_place['percentage practices'] =  round(
    by_place['practices daily'] * 100 /
    (by_place['vaccination centers daily'] + by_place['practices daily']), 2)

by_place['percentage centers'] = 100 - by_place['percentage practices']
```

In [129…
```
# make 'date' an index
by_place.set_index('date', inplace=True)
```

In [130…
```
by_place
```

Out[130…

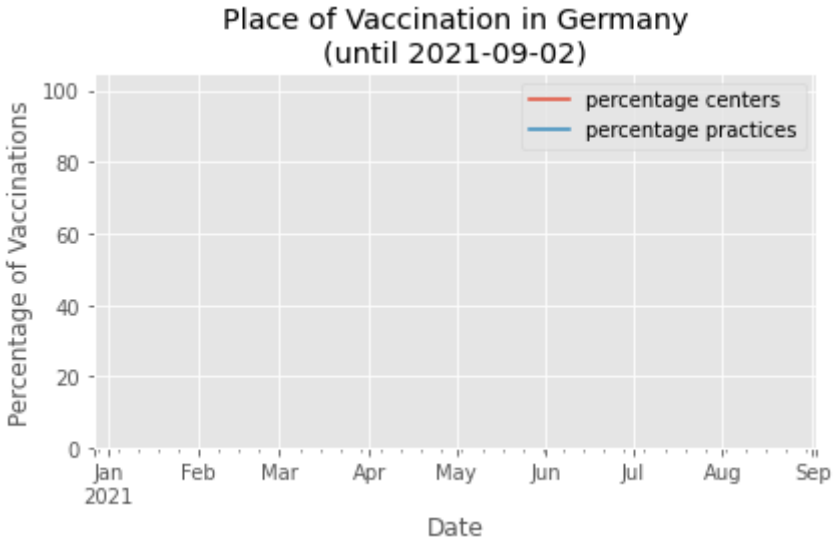
date	vaccination centers	practices	vaccination centers daily	practices daily	percentage practices	percentage centers
2020-12-27	0	0	NaN	NaN	NaN	NaN

date	vaccination centers	practices	vaccination centers daily	practices daily	percentage practices	percentage centers
2020-12-28	0	0	0.0	0.0	NaN	NaN
2020-12-29	0	0	0.0	0.0	NaN	NaN
2020-12-30	0	0	0.0	0.0	NaN	NaN
2020-12-31	0	0	0.0	0.0	NaN	NaN
...	...	...	...	...	...	...
2021-08-29	0	0	0.0	0.0	NaN	NaN
2021-08-30	0	0	0.0	0.0	NaN	NaN
2021-08-31	0	0	0.0	0.0	NaN	NaN
2021-09-01	0	0	0.0	0.0	NaN	NaN
2021-09-02	0	0	0.0	0.0	NaN	NaN

250 rows × 6 columns

In [131…
```
share = by_place.loc[ : , ['percentage centers', 'percentage practices']]
```

In [132…
```
vacc_shares = share.plot(
    # as it is cumulative, the last row must contain the single highest number
    ylim=(0, 105),  # above 100 to see the line
    xlabel='Date',
    ylabel='Percentage of Vaccinations',
    title=f"Place of Vaccination in Germany\n(until {last_update})")
```



In [133…
```
fig = vacc_shares.get_figure()
fig.savefig('img/vaccinations_germany_by_place.png')
```

Other units of Time

In [134…
```
by_place_daily = by_place.loc[ : , ['vaccination centers daily', 'practices daily']]
by_place_daily.columns = ['vaccination centers', 'practices']
by_place_daily.reset_index(inplace=True)
```

Monthly

In [135…
```
by_place_monthly = by_place_daily.groupby(pd.Grouper(key='date',freq='M')).sum()
by_place_monthly.tail()
```

Out[135…

date	vaccination centers	practices
2021-05-31	0.0	0.0
2021-06-30	0.0	0.0
2021-07-31	0.0	0.0
2021-08-31	0.0	0.0

date	vaccination centers	practices
2021-09-30	0.0	0.0

Scale:

In [136…
```
by_place_monthly['vaccination centers'] = by_place_monthly['vaccination centers'] / 1_000_000
by_place_monthly['practices'] = by_place_monthly['practices'] / 1_000_000
```

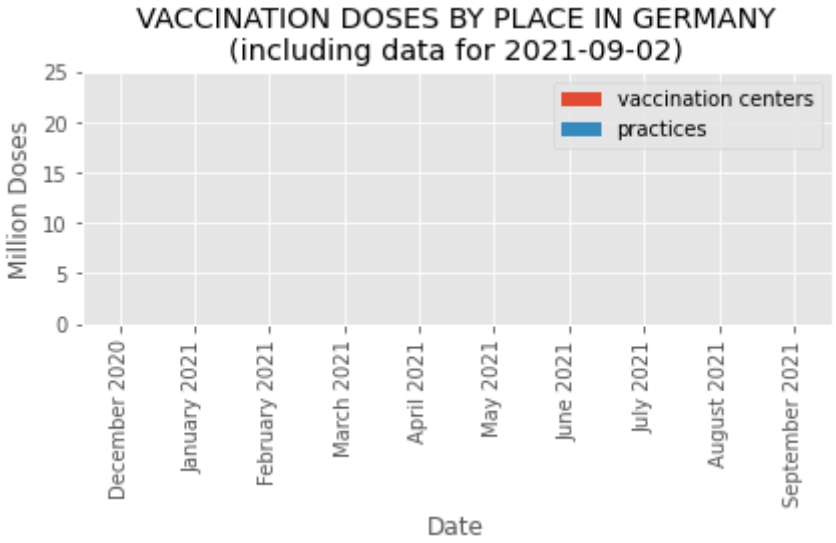
Rename the columns

In [137…
```
by_place_monthly['month'] = by_place_monthly.index.strftime('%B')
by_place_monthly['year'] =  by_place_monthly.index.strftime('%Y')
by_place_monthly['label'] = by_place_monthly['month'] + ' ' + by_place_monthly['year']
by_place_monthly.drop(columns=['month', 'year'], inplace=True)
by_place_monthly.set_index('label', inplace=True)
by_place_monthly.tail(6)
```

Out[137…

label	vaccination centers	practices
April 2021	0.0	0.0
May 2021	0.0	0.0
June 2021	0.0	0.0
July 2021	0.0	0.0
August 2021	0.0	0.0
September 2021	0.0	0.0

In [138…
```
monthly_plot = by_place_monthly.plot.bar(
    stacked=True,
    ylim=(0, 25),
    xlabel='Date',
    ylabel='Million Doses',
    title=f"VACCINATION DOSES BY PLACE IN GERMANY\n(including data for {last_update})")
```



In [139…

```
fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_by_place_germany.png')
```