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 [1]:   # standard library
          import datetime
          import math
```

```
In [2]:   # 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 [3]:   today = datetime.datetime.today().strftime('%Y-%m-%d')
          today
```

```
Out[3]:   '2021-05-22'
```

Set Defaults

```
In [4]:   # style like ggplot in R
          plt.style.use('ggplot')
```

```
In [5]:   # 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 [6]:    population_germany = 83_200_000
```

Get and Transform Data

```
In [7]:    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

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

```
In [8]:    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 [9]:    more_cols_to_drop = ['dosen_biontech_erst_kumulativ', 'dosen_biontech_zweit_kumulativ',
                                'dosen_moderna_erst_kumulativ', 'dosen_moderna_zweit_kumulativ',
                                'dosen_astrazeneca_erst_kumulativ', 'dosen_astrazeneca_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 [10]:   vaccinations.drop(columns=['impf_quote_erst', 'impf_quote_voll'], inplace=True)
```

Convert datatype of date column

```
In [11]:   vaccinations.iloc[ : , [0]] = vaccinations.iloc[ : , [0]].apply(pd.to_datetime)
```

Show Data

```
In [12]:   vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 145 entries, 0 to 144
Data columns (total 13 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   date                            145 non-null    datetime64[ns]
 1   dosen_kumulativ                 145 non-null    int64
 2   dosen_differenz_zum_vortag      145 non-null    int64
 3   dosen_erst_differenz_zum_vortag 145 non-null    int64
 4   dosen_zweit_differenz_zum_vortag 145 non-null   int64
 5   dosen_biontech_kumulativ        145 non-null    int64
 6   dosen_moderna_kumulativ         145 non-null    int64
 7   dosen_astrazeneca_kumulativ     145 non-null    int64
 8   personen_erst_kumulativ         145 non-null    int64
 9   personen_voll_kumulativ         145 non-null    int64
 10  dosen_dim_kumulativ             145 non-null    int64
 11  dosen_kbv_kumulativ             145 non-null    int64
 12  dosen_johnson_kumulativ         145 non-null    int64
dtypes: datetime64[ns](1), int64(12)
memory usage: 14.9 KB
```

In [13]: `vaccinations.tail(3)`

Out[13]:

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
142	2021-05-18	41567941	862946	514216	348730	3053867
143	2021-05-19	42639298	1071357	555973	515384	3132982
144	2021-05-20	43549672	910374	444696	465678	3199015

Check Validity

In [14]:
```
# get the last row / the newest available data
last_row = vaccinations.tail(1)
```

In [15]:
```
doses_used = last_row['dosen_kumulativ']
doses_used
```

Out[15]:
```
144    43549672
Name: dosen_kumulativ, dtype: int64
```

In [16]:
```python
# 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 [17]:
```python
# Must be exactly 0
doses_used - partially_vaccinated_people - (fully_vaccinated_people - johnson_doses) * 2 - johnson_doses ==  0
```

Out[17]:
```
144     True
dtype: bool
```

Calculate columns

In [18]:
```python
vaccinations['partly vaccinated'] = round(
    (vaccinations['personen_erst_kumulativ'] - vaccinations['personen_voll_kumulativ']) * 100 / population_germany,
    2)
```

In [19]:
```python
vaccinations['fully vaccinated'] = round(
    vaccinations['personen_voll_kumulativ'] * 100 / population_germany,
    2)
```

In [20]:
```python
vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145 entries, 0 to 144
Data columns (total 15 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   date                          145 non-null     datetime64[ns]
 1   dosen_kumulativ               145 non-null     int64
 2   dosen_differenz_zum_vortag    145 non-null     int64
 3   dosen_erst_differenz_zum_vortag   145 non-null     int64
 4   dosen_zweit_differenz_zum_vortag  145 non-null     int64
 5   dosen_biontech_kumulativ      145 non-null     int64
 6   dosen_moderna_kumulativ       145 non-null     int64
 7   dosen_astrazeneca_kumulativ   145 non-null     int64
 8   personen_erst_kumulativ       145 non-null     int64
 9   personen_voll_kumulativ       145 non-null     int64
 10  dosen_dim_kumulativ           145 non-null     int64
 11  dosen_kbv_kumulativ           145 non-null     int64
 12  dosen_johnson_kumulativ       145 non-null     int64
 13  partly vaccinated             145 non-null     float64
```

```
  14   fully vaccinated                    145 non-null       float64
dtypes: datetime64[ns](1), float64(2), int64(12)
memory usage: 17.1 KB
```

In [21]:
```
vaccinations.tail(3)
```

Out[21]:

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
142	2021-05-18	41567941	862946	514216	348730	3053867
143	2021-05-19	42639298	1071357	555973	515384	3132982
144	2021-05-20	43549672	910374	444696	465678	3199015

◄ ▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬ ►

Last Update

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

In [22]:
```
last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
last_update
```

Out[22]: `'2021-05-20'`

Doses Used

In [23]:
```
doses = vaccinations.loc[ : , ['date', 'dosen_differenz_zum_vortag']]
# Rename columns
doses.columns = ['date', 'doses used']
```

In [24]:
```
# Scale number of doses as millions
doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

In [25]:
```
doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```
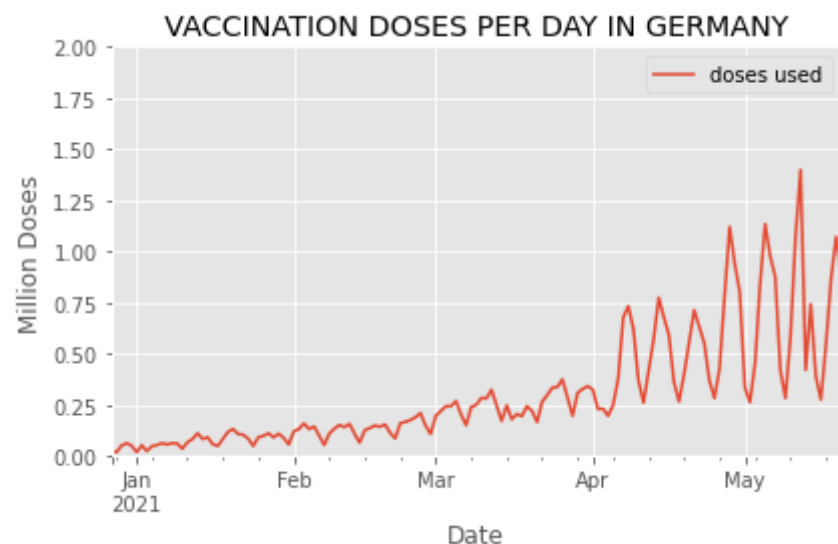
Out[25]:

	doses used
date	
2021-05-20	0.910374

In [26]:
```python
# What is the highest number of doses used in a day?
max_doses_daily = max(doses_daily['doses used'])
max_doses_daily
```

Out[26]: 1.39941

In [27]:
```python
doses_daily.plot(
    ylim=(0,math.ceil(max_doses_daily)),
    xlabel='Date',
    ylabel='Million Doses',
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

Out[27]: <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>

Doses per Weekday (in the last 6 weeks)

In [28]:
```python
last_6_weeks = doses.tail(42)
```

In [29]:
```python
# 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-29-45013977109e>: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 [30]:
```python
# check:
last_6_weeks.tail(3)
```

Out[30]:

	date	doses used	weekday
142	2021-05-18	0.862946	Tuesday
143	2021-05-19	1.071357	Wednesday
144	2021-05-20	0.910374	Thursday

In [31]:
```python
# drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

In [32]:
```python
#last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

Out[32]:

	doses used	weekday
142	0.862946	Tuesday
143	1.071357	Wednesday
144	0.910374	Thursday

In [33]:
```python
pivot_table =last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

Out[33]:

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
140	NaN	NaN	NaN	0.276121	NaN	NaN	NaN
141	NaN	0.542053	NaN	NaN	NaN	NaN	NaN

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
142	NaN	NaN	NaN	NaN	NaN	0.862946	NaN
143	NaN	NaN	NaN	NaN	NaN	NaN	1.071357
144	NaN	NaN	NaN	NaN	0.910374	NaN	NaN

In [34]:
```python
# 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[34]:

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
140	NaN	NaN	NaN	NaN	NaN	NaN	0.276121
141	0.542053	NaN	NaN	NaN	NaN	NaN	NaN
142	NaN	0.862946	NaN	NaN	NaN	NaN	NaN
143	NaN	NaN	1.071357	NaN	NaN	NaN	NaN
144	NaN	NaN	NaN	0.910374	NaN	NaN	NaN

In [35]:
```python
weekday_boxplot = pivot_table.boxplot()
```
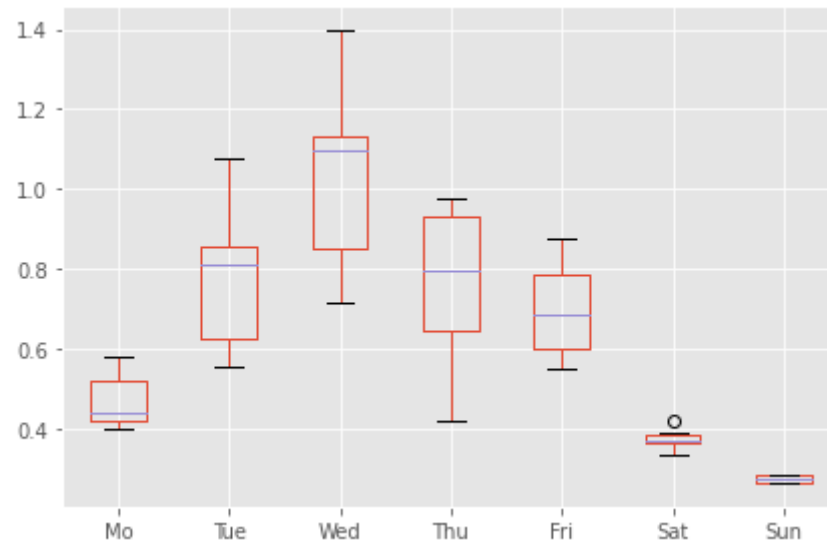
```
In [36]:  fig = weekday_boxplot.get_figure()
          fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

```
In [37]:  # 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[37]:

| date	million doses used
2021-04-26	3.534467
2021-05-03	4.692721
2021-05-10	5.112350
2021-05-17	4.845280
2021-05-24	2.844677

```
In [38]:  # What is the highest number of doses used in a week?
```

```python
max_million_doses_weekly = max(doses_weekly['million doses used'])
max_million_doses_weekly
```

Out[38]: 5.11235

In [39]:
```python
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[39]: `<AxesSubplot:title={'center':'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-05-20)'}, xlabel='Date', ylabel='Million Doses'>`
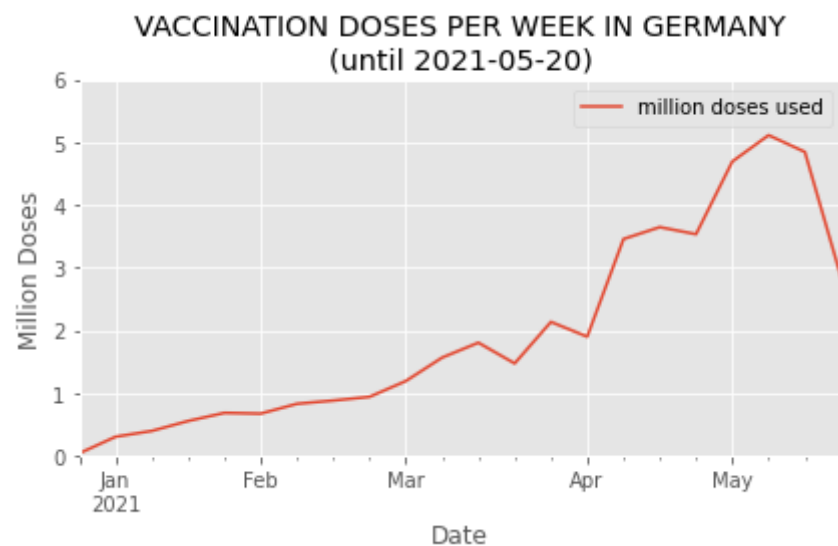
Doses per Month

In [40]:
```python
# M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date',freq='M')).sum()
doses_monthly.tail()
```

Out[40]:

date	doses used
2021-01-31	2.344695

	doses used
date	
2021-02-28	3.779238
2021-03-31	7.853167
2021-04-30	15.516301
2021-05-31	13.849828

In [41]:
```python
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[41]:

	doses used
label	
December 2020	0.206443
January 2021	2.344695
February 2021	3.779238
March 2021	7.853167
April 2021	15.516301
May 2021	13.849828

In [42]:
```python
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})")
```
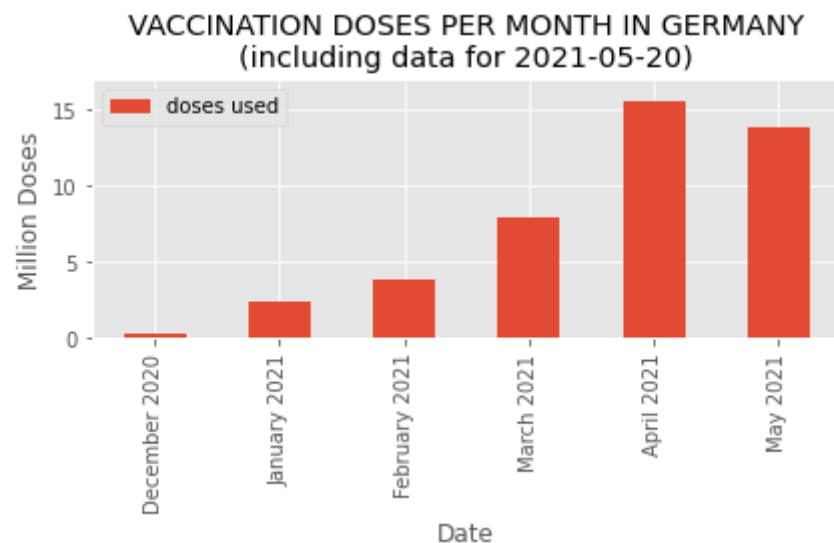
```
In [43]:   fig = monthly_plot.get_figure()
           fig.savefig('img/monthly_doses_germany.png')
```

Vaccination Campaign Progress

```
In [44]:   doses_cumulative = vaccinations.loc[ : , ['date', 'partly vaccinated', 'fully vaccinated']]
           doses_cumulative.set_index('date', inplace=True)
           doses_cumulative.tail(3)
```

Out[44]:

date	partly vaccinated	fully vaccinated
2021-05-18	26.16	11.94
2021-05-19	26.23	12.56
2021-05-20	26.21	13.12

```
In [45]:   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})")
```
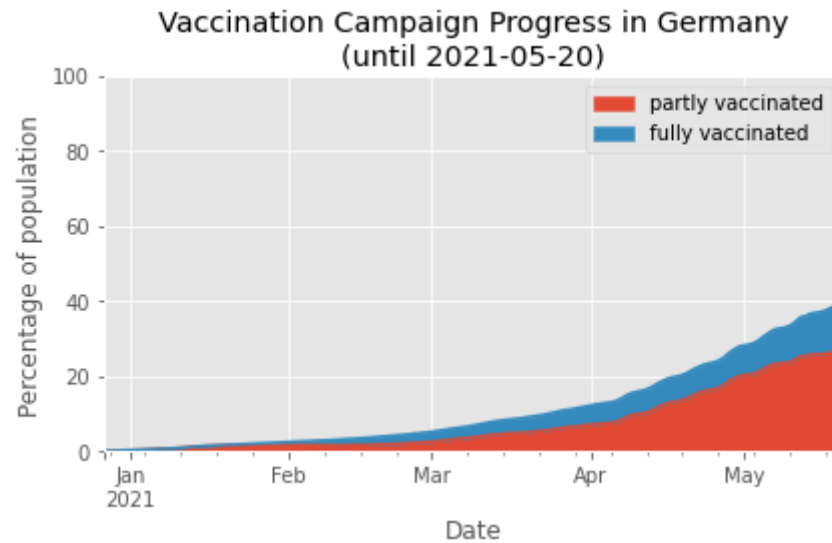
```
In [46]:   fig = doses_area_plot.get_figure()
           fig.savefig('img/vaccinations_germany_area_plot.png')
```

As of Today

```
In [47]:   # get the last line of the data
           current_state = doses_cumulative.iloc[-1]
           current_state
```

```
Out[47]:   partly vaccinated    26.21
           fully vaccinated     13.12
           Name: 2021-05-20 00:00:00, dtype: float64
```

```
In [48]:   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()
```


Vaccination Progress in Germany
as of 2021-05-20

- not vaccinated (60.7%)
- first dose (26.2%)
- fully vaccinated (13.1%)
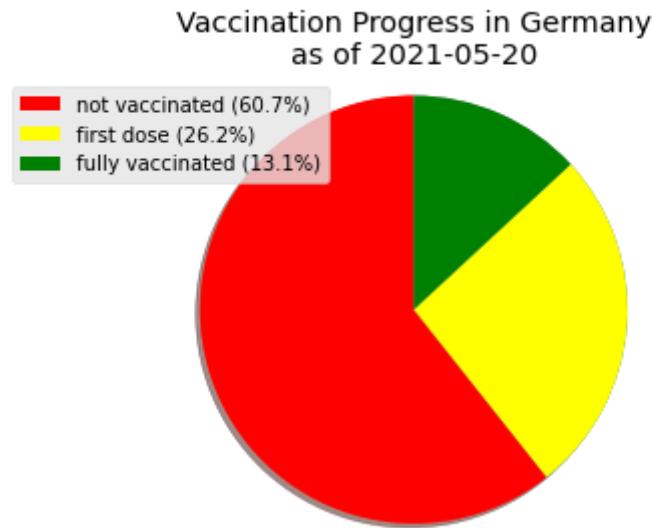
Vaccines in Use

```
In [49]:  vaccine_use = vaccinations.loc[ : , ['date', 'dosen_biontech_kumulativ',
                                               'dosen_moderna_kumulativ',
                                               'dosen_astrazeneca_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[49]:

	BioNTech	Moderna	AstraZeneca	Johnson & Johnson
date				
2021-05-18	30.538677	2.996634	7.962746	0.069884
2021-05-19	31.329821	3.087631	8.141466	0.080380
2021-05-20	31.990150	3.175990	8.293287	0.090245

In [50]:
```python
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]))+1),
    xlabel='Date',
    ylabel='Million Doses (cumulative)',
    title=f"VACCINES USED IN GERMANY\n(until {last_update})")
```
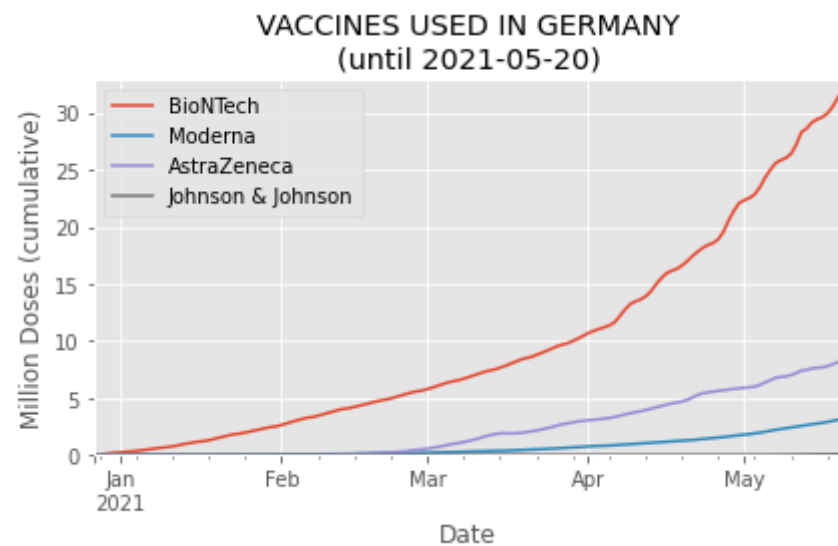

In [51]:
```python
fig = vaccines_used.get_figure()
fig.savefig('img/vaccines_used_in_germany.png')
```

Vaccination Centers versus Doctor's Practices

In [52]:
```python
by_place = vaccinations.loc[ : , ['date', 'dosen_dim_kumulativ', 'dosen_kbv_kumulativ']]
```

```
by_place.columns = ['date', 'vaccination centers', 'practices']
```

In [53]:
```
by_place['vaccination centers daily'] = by_place['vaccination centers'].diff()
by_place['practices daily'] = by_place['practices'].diff()
```

In [54]:
```
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 [55]:
```
# make 'date' an index
by_place.set_index('date', inplace=True)
```

In [56]:
```
by_place
```

Out[56]:

date	vaccination centers	practices	vaccination centers daily	practices daily	percentage practices	percentage centers
2020-12-27	24087	0	NaN	NaN	NaN	NaN
2020-12-28	42647	0	18560.0	0.0	0.00	100.00
2020-12-29	93470	0	50823.0	0.0	0.00	100.00
2020-12-30	156252	0	62782.0	0.0	0.00	100.00
2020-12-31	206443	0	50191.0	0.0	0.00	100.00
...	...	...	...	...	...	...
2021-05-16	30070471	10092471	260116.0	16005.0	5.80	94.20
2021-05-17	30443206	10261789	372735.0	169318.0	31.24	68.76
2021-05-18	30851148	10716793	407942.0	455004.0	52.73	47.27
2021-05-19	31242727	11396571	391579.0	679778.0	63.45	36.55
2021-05-20	31622158	11927514	379431.0	530943.0	58.32	41.68

145 rows × 6 columns

In [57]:
```
share = by_place.loc[ : , ['percentage centers', 'percentage practices']]
```

In [58]:
```python
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})")
```
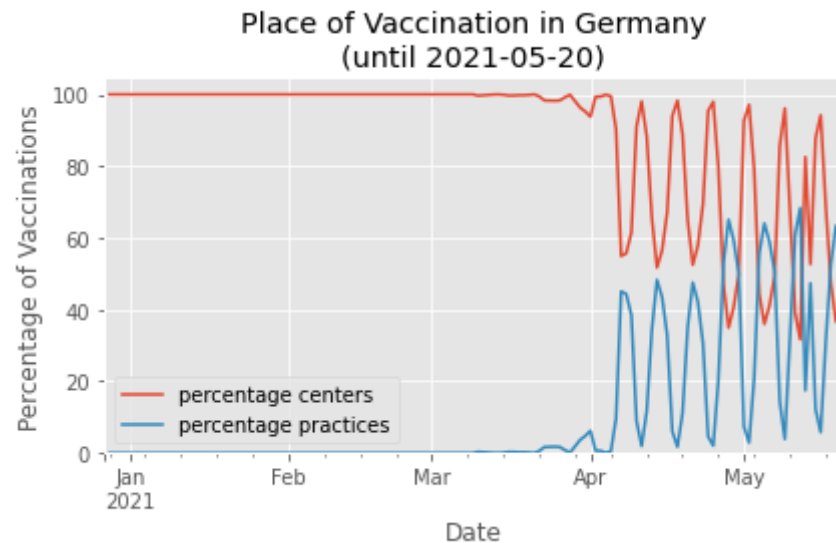

In [59]:
```python
fig = vacc_shares.get_figure()
fig.savefig('img/vaccinations_germany_by_place.png')
```

Other units of Time

In [60]:
```python
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 [61]:
```python
by_place_monthly = by_place_daily.groupby(pd.Grouper(key='date',freq='M')).sum()
by_place_monthly.tail()
```

Out[61]:

	vaccination centers	practices
date		

	vaccination centers	practices
date		
2021-01-31	2344695.0	0.0
2021-02-28	3779238.0	0.0
2021-03-31	7786933.0	66234.0
2021-04-30	10187161.0	5329140.0
2021-05-31	7317688.0	6532140.0

Scale:

In [62]:
```python
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 [63]:
```python
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[63]:

	vaccination centers	practices
label		
December 2020	0.182356	0.000000
January 2021	2.344695	0.000000
February 2021	3.779238	0.000000
March 2021	7.786933	0.066234
April 2021	10.187161	5.329140
May 2021	7.317688	6.532140

In [64]:
```python
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})")
```
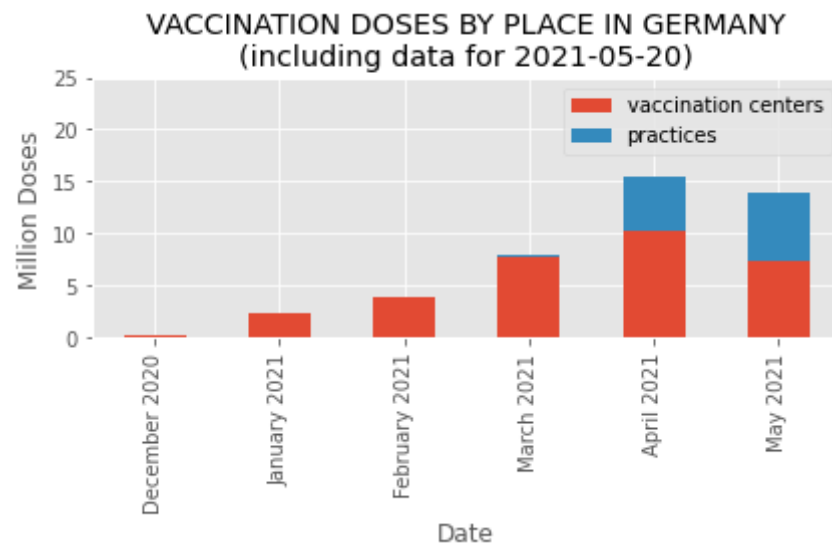

In [65]:
```
fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_by_place_germany.png')
```