CORD-19 Analyse fetched coredata

In general, this jupyter notebook is designated to analyse fetched core data.

First, relevant packages must be imported to the notebook.

In [1]:

```
import numpy as np
import pandas as pd
import csv
import ast
import collections
import matplotlib.pyplot as plt
import datetime
import re
import time
from urllib.parse import urlparse
from collections import Counter
from pybtex.database import parse_file, BibliographyData, Entry
```

Thusly, the fetched core data is read from the PKL-file and incorporated into a DataFrame for further processing.

In [2]:

```
read_coredata = pd.read_pickle('extra_info_CS5099.pkl')
df_current_extra_info = pd.DataFrame()
df_current_extra_info['coredata'] = read_coredata['coredata']
df_current_extra_info
```

Out[2]:

	coredata
0	{'srctype': 'j', 'eid': '2-s2.0-85083266658', ...
1	{'srctype': 'j', 'prism:issueIdentifier': '7',...
2	{'srctype': 'j', 'prism:issueIdentifier': '8',...
3	{'srctype': 'j', 'prism:issueIdentifier': '9',...
4	{'srctype': 'j', 'prism:issueIdentifier': '11'...
...	...
74297	{'srctype': 'j', 'eid': '2-s2.0-85092678139', ...
74298	{'srctype': 'j', 'eid': '2-s2.0-85087468210', ...
74299	{'srctype': 'j', 'eid': '2-s2.0-85092677974', ...
74300	{'srctype': 'j', 'prism:issueIdentifier': '4',...
74301	None

74302 rows × 1 columns

For expressiveness purposes, the number of None values within the DataFrame must be considered.

In [3]:

```
df_current_extra_info.isnull().sum()
```

Out[3]:

```
coredata    13997
dtype: int64
```

In [4]:

```
no_return_value = round(df_current_extra_info.isnull().sum() / len(df_current_extra_info) *
print(no_return_value)
```

```
coredata    18.84
dtype: float64
```

Compared to the length of the dataset, ~18.8% of fetched SCOPUS data has no return value. Thusly, all rows which contain "None" values are dropped and the DataFrame is reindexed.

In [5]:

```
df_combined = df_current_extra_info.dropna()
df_combined = df_combined.reset_index(drop=True)
df_combined
```

Out[5]:

	coredata
0	{'srctype': 'j', 'eid': '2-s2.0-85083266658', ...
1	{'srctype': 'j', 'prism:issueIdentifier': '7',...
2	{'srctype': 'j', 'prism:issueIdentifier': '8',...
3	{'srctype': 'j', 'prism:issueIdentifier': '9',...
4	{'srctype': 'j', 'prism:issueIdentifier': '11'...
...	...
60300	{'srctype': 'j', 'eid': '2-s2.0-85092679086', ...
60301	{'srctype': 'j', 'eid': '2-s2.0-85092678139', ...
60302	{'srctype': 'j', 'eid': '2-s2.0-85087468210', ...
60303	{'srctype': 'j', 'eid': '2-s2.0-85092677974', ...
60304	{'srctype': 'j', 'prism:issueIdentifier': '4',...

60305 rows × 1 columns

The following functions support the creation of DataFrames based on the columns affiliation and core data.

In [6]:

```
def get_one_entry(dic):
    """
    This function receives a dictionary with one entry and returns it as transformed DataFr
    """
    df_affiliation_holder = pd.DataFrame(dic.items()).T
    df_affiliation_holder.columns = df_affiliation_holder.iloc[0]
    df_affiliation_holder = df_affiliation_holder.drop(df_affiliation_holder.index[0])
    return df_affiliation_holder
```

In [7]:

```
def get_various_entries(dic):
    """
    This function receives a dictionary with more than one entry and returns it as transfor
    """
    df_affiliation_holder = pd.DataFrame.from_dict(dic, orient='columns')
    return df_affiliation_holder
```

The next cell creates the DataFrame which focus on core data.

In [8]:

```
%%time
df_coredata = pd.DataFrame()
df_coredata_holder = pd.DataFrame()


for i in df_combined['coredata']:
    string_holder = str(i)
    if string_holder[0] == "[":
        df_coredata_holder = get_various_entries(i)
    else:
        df_coredata_holder = get_one_entry(i)
    df_coredata = pd.concat([df_coredata_holder, df_coredata],ignore_index=True)
    print(len(df_coredata))
df_coredata
```

```
57360
57361
57362
57363
57364
57365
57366
57367
57368
57369
57370
57371
57372
57373
57374
57375
57376
57377
```

Subsequently, the publications literature types are counted.

In [9]:

```
ser_type = df_coredata['prism:aggregationType']
ser_type_counted = ser_type.value_counts()
ser_type_counted
```

Out[9]:

```
Journal                  57000
Book Series               2504
Book                       561
Conference Proceeding      226
Trade Journal                9
Name: prism:aggregationType, dtype: int64
```

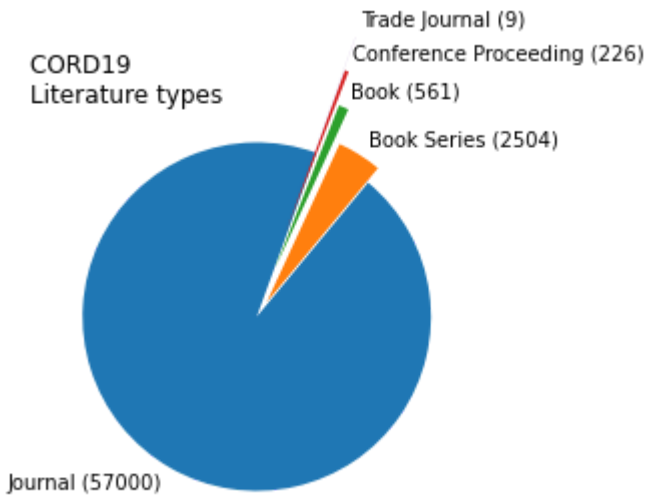
Next, the literature types are visualized with a pie chart.

In [10]:

```
ser_type_dict = {}
len_ser_type_counted  = len(ser_type_counted)
i = 0
while i < len_ser_type_counted:
    ser_type_dict[i] = ser_type_counted.index[i] + " (" + str(ser_type_counted.values[i]) +
    i = i + 1

y = ser_type_counted.values
labelling = ser_type_dict.values()
shift = [-0.1, 0.2, 0.4, 0.6, 0.8]

plt.pie(y, labels = labelling, startangle = 70, explode = shift)
plt.title("CORD19 \nLiterature types", loc="left")
plt.show()
```



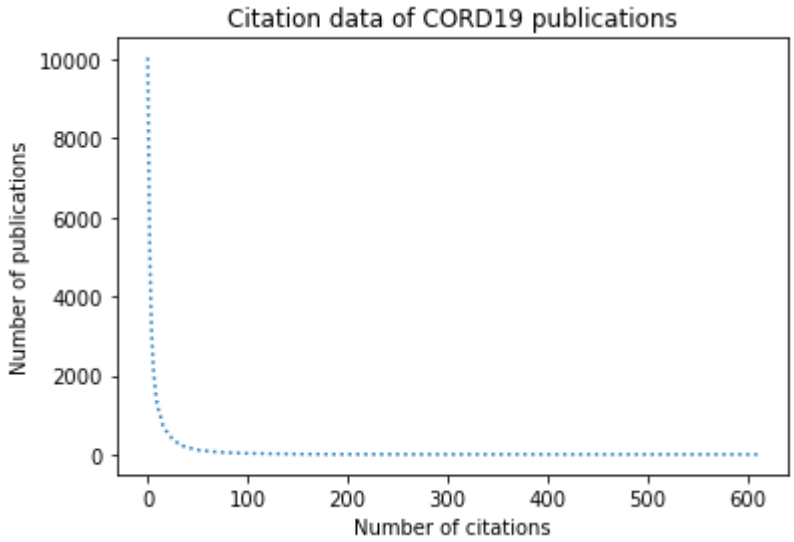
The pie chart shows that journal is the dominating literature type followed noticeable by books. Conference proceedings and trade journals are less occurring in this dataset.

The cited count of CORD19 publications is presented with a scatterplot.

In [11]:

```
ser_citedcount = df_coredata['citedby-count']
ser_citedcount_counted = ser_citedcount.value_counts()
ypoints = ser_citedcount_counted.values

plt.plot(ypoints, linestyle = 'dotted')
plt.xlabel("Number of citations")
plt.ylabel("Number of publications")
plt.title("Citation data of CORD19 publications")
plt.show()
```



The scatterplot shows that most of the publications are not referenced or solely have a few citations. A small selection of publications is cited often.

In [12]:

```
ser_citedcount_counted
```

Out[12]:

```
0       10039
1        7328
2        5235
3        3855
4        2999
        ...
279         1
516         1
957         1
884         1
1669        1
Name: citedby-count, Length: 611, dtype: int64
```