In [1]:
```
import pandas as pd
import numpy as np
from collections import Counter
import json
import ast
import pickle
```

Importing the data

The data are going to be explored in this jupyter notebook file, have been stored in a dictionary with a .pkl format. You can download the data through the DataverseNL.

It should be noted that the current data sample has been enriched to include company unique identifier from a publicly available company database, namely 7+ Million Company Dataset by People Data Labs. You can merge the sample of Bigprod data with the mentioned dataset using the column **index** in the Companies_data dataframe. The following cells explain how the BigProd data can be merged into 7+ Million Company Dataset.

In [2]:
```
# import the sample of BigProd data
with open("BIGPROD_Public_Data.pkl", 'rb') as handle: # please modify the name of data have been stored in your system
    data = pickle.load(handle)
```

In [ ]:
```
''' companies data are stored in a dataframe named Companies_data'''
''' using the index column in Companies_data, you can indetify the instances from People Data Labs have been brought in t
data
```

The following function will be using frequently for converting dict files to list. Therefore, it is executed here before going through the data.

In [ ]:
```
def getlist(dict): # This function make a list of keys in a dict. It enables extraction of data stored in a json format.
    if dict is not None:
        l = list(dict.keys())
    else:
        l = []
    return l
```

Table of Datagroup

Companies Table

The following cells show missing values and basic descriptive stats of the Companies dataset.

In [ ]:
```
data_companies = data['Companies_data']

#missing values stored in dataframe format
missvalue_companies = data_companies.isnull().sum()
print('The number of missing values is:','\n{}'.format(missvalue_companies))
```

In [ ]:
```
data_companies
```

Descriptive stats for *name, website, country_code, nace_code* (categorical variables)

In [ ]:
```
data_companies[['name','website','country','industry']].describe()
```

Number of unique instances for *iso and linked_countries*

In [ ]:
```
# finding the number of unique keywords,iso, linked_countries and fos_ids
unique_iso = len(list(dict.fromkeys(data_companies["iso"].apply(getlist).sum())))
unique_linkedcountry = len(list(dict.fromkeys(data_companies["linked_countries"].apply(getlist).sum())))
```

In [ ]:
```
print('number of unique iso: {}'.format(unique_iso),'\n'
'number of unique linked_countries: {}'.format(unique_linkedcountry))
```

In [ ]:
```
top_freq_name = Counter(data_companies["name"]).most_common()[:3]
top_freq_website = Counter(data_companies["website"]).most_common()[:3]
```

In [ ]:
```
print('Top frequent names are: {}'.format(top_freq_name),'\n'
'top frequent websites are: {}'.format(top_freq_website))
```

Top X frequent *nace_code, country, iso and linked_countries*.

In [ ]:
```
x = 3

top_freq_industry = Counter(data_companies["industry"]).most_common()[:x]
top_freq_country = Counter(data_companies["country"]).most_common()[:x]

top_freq_iso = Counter(data_companies["iso"].apply(getlist).sum()).most_common()[:x]
top_freq_linkedcountry = Counter(data_companies["linked_countries"].apply(getlist).sum()).most_common()[:x]

# top_freq_keywords = Counter(data_companies["keywords"].apply(getlist).sum()).most_common()[:x]
# top_freq_fosids = Counter(data_companies["fos_ids"].apply(getlist).sum()).most_common()[:x]
```

In [ ]:
```
print('Top {} frequent Nace codes are (Nace code, frequency): {}'.format(x, top_freq_nacecode),'\n\n')
print('Top {} frequent country codes are (Country code, frequency): {}'.format(x, top_freq_countrycode),'\n\n')
print('Top {} frequent ISO codes are (ISO code, frequency): {}'.format(x, top_freq_iso),'\n\n')
print('Top {} frequent linked countries are (Linked country, frequency): {}'.format(x, top_freq_linkedcountry))
```

Here, we measure the number of unique keywords and fos_ids and their freqeuncy.
It should be noted that although some keywords and fos_ids might appear more than one time in each bvd_id, **we have not considered any weight** for the corresponded keywords and fos_ids allocated. This index indicates that to what extent the presented keywords/ fos_ids are popular among all instances.

In [ ]:
```
C1 = Counter(data_companies["keywords"].apply(getlist).sum())

print('Number of unique keywords:\n {}'.format(len(C1)))
print('Top frequent keywords are:','\n{}'.format(C1.most_common()[:3]))
```

In [ ]:
```
C2 = Counter(data_companies["fos_ids"].apply(getlist).sum())

print('Number of unique FOS ids:\n {}'.format(len(C2)))
print('Top frequent FOS ids are:','\n{}'.format(C2.most_common()[:3]))
```

The following cell presents basic descriptive stats for the number of allocated *iso, keywords, linked_countries and fos_id* to each bvd_ids

In [ ]:
```
def len_of_dict(dict): # This funtion builds another variable from the dict format data, which shows the number of keywor
    if dict is not None:
        length =  len(dict)

        return length


data_companies["#iso"] = data_companies["iso"].apply(len_of_dict)
data_companies["#keywords"] = data_companies["keywords"].apply(len_of_dict)
data_companies["#fos_ids"] = data_companies["fos_ids"].apply(len_of_dict)
data_companies["#linked_countries"] = data_companies["linked_countries"].apply(len_of_dict)


pd.options.display.float_format = "{:.2f}".format #showing numbers with 2 decimals

print('This descriptive statistics for the mentioned variables are:','\n\n{}'
.format(data_companies.describe(include = [np.number])))
```

Top x Companies with the most *iso and linked_countries*.
Please note that as the number of keywords, and fos_ids for each bvd_ids have been limited, respectively to 2000 and 100 records, we therefore only show the companies with the broadest *iso and linked_countries* records.

In [ ]:
```
x = 5
companies_most_iso = data_companies.nlargest(X, '#iso')['name']
companies_most_linkedcountries = data_companies.nlargest(X, '#linked_countries')['name']

print('Top {} companies in terms of number of iso:'.format(x),'\n\n{}'.format(list(companies_most_iso)),'\n\n')
print('Top {} companies in terms of number of linked_countries:'.format(x),'\n\n{}'.format(list(companies_most_linkedcour
```

Visualization by Altair

Here, you can find the distribution of companies by their country code.

In [ ]:
```
!pip install altair
import altair as alt
alt.data_transformers.disable_max_rows()
```

In [ ]:
```
altair_data = data_companies[['index','country']].groupby('country').size().reset_index().rename(columns = {0:'Count'})
```

In [ ]:
```
bars = alt.Chart(altair_data).mark_bar().encode(x = alt.X('Count',title = 'Count of records'), y = alt.Y('country_code',s

text = bars.mark_text(align = 'left', baseline = 'middle',dx = 3).encode(text ="Count")

(bars + text).properties(height = 600, width = 800)

alt.layer(bars, text).configure_view(strokeOpacity=0)
```

Collaboration Table

The following cells explain basic descriptive stats of the Collaboration dataset

In [ ]:
```
data_collaboration = data["Collaboration_data"].copy()

missvalue_collaboration = data_collaboration.isnull().sum()
print('The number of missing values is:','\n{}'.format(missvalue_collaboration))
```

In [ ]:
```
data_collaboration.describe()
```

As the Collaboration data include only categorical variables, the descriptive statistics presents count of instances, number of unique instances and the most frequent instance in this dataset.

Top frequent *bvd_ids, name and country_code* appeared in the collaboration data.

In [ ]:
```
# To find the most frequent entities in the collaboration data, first we need to identify the most frequent bvd_ids
# and then using the Companies data, find the entities name.

top_freq_index = Counter(data_collaboration["index"]).most_common()[:3]
top_freq_entityname = [data_companies['name'][data_companies['index'] == i[0]].to_list() for i in top_freq_bvdid]

top_freq_name = Counter(data_collaboration["name"]).most_common()[:3]
top_freq_countrycode = Counter(data_collaboration["country"]).most_common()[:3]
```

In [ ]:
```
print('Top {} frequent entities are:'.format(x),'\n\n{}'.format(top_freq_index))
```

In [ ]:
```
print('Top {} frequent collaborators are:'.format(x),'\n\n{}'.format(top_freq_name))
```

Please note that some of collaborators such as **INSTITUTE OF TECHNOLOGY,UNIVERSITY OF TECHNOLOGY and UNIVERSITY OF APPLIED SCIENCES** have missed their identifiers in the current dataset. Such instances can be partially identidified using their country.

In [ ]:
```
print('The collaborators were mostly from the following countries:','\n\n{}'.format(top_freq_countrycode))
```

Finding national and international level of collaboration

In [ ]:
```
data_collaboration = data_collaboration.dropna( how = 'any', subset = ['country'])

new_data_collaboration = pd.merge(data_companies,data_collaboration, how = 'inner', on = 'index')[['index','name_x','cour
new_data_collaboration = new_data_collaboration.rename(columns = {'name_x':'entity name','name_y':'collaborator name','cc

def f1(a,b):
    if a == b:
        return 'National'
    else:
        return 'Inernational'


new_data_collaboration['collaboration type'] = new_data_collaboration[['country code','collaborator country']].apply(lamb

pd.options.display.float_format = "{:.2f}".format #showing numbers with 2 decimals
collaboration = new_data_collaboration.groupby(['country code','collaboration type']).size()/new_data_collaboration.group
# collaboration.to_excel('collaboration national internationl.xlsx')
print('The rates of national and international collaboration by country are:','\n\n{}'.format(collaboration))
```

Patent Table

In [ ]:
```
data_patents = data["Patents_data"].copy()
```

In [ ]:
```
missvalue_patent = data_patents.isnull().sum()
print('The number of missing values in patent data is:','\n{}'.format(missvalue_patent))
```

Basic descriptive statitics for patent data, which incldues only categorical variables.

In [ ]:
```
data_patents.describe()
```

Top frequent index and their corresponded entity name in the Patents data.

In [ ]:
```
top_freq_index = Counter(data_patents["bvd_id"]).most_common()[:3]
top_freq_entityname = [data_companies['name'][data_patents['index'] == i[0]].to_list() for i in top_freq_index]

print('Top {} frequent indexes are:'.format(x),'\n\n{}'.format(top_freq_index))
```

In [ ]:
```
top_freq_name = Counter(data_patents["patent_name"]).most_common()[:3]
top_freq_name
```

Publication Table

Descriptive statistics for Publication data includes number of unique and top frequent FOS ids and companies.

In [ ]:
```
data_publications = data["Publications_data"].copy()
print('There are {} unique doi in the publication data'.format(data_publications["doi"].nunique()))
```

In [ ]:
```
missvalue_publications = data_publications.isnull().sum()
print('The number of missing values in the pubclication data is:','\n{}'.format(missvalue_publications))
```

In [ ]:
```
data_publications.describe()
```

Number of unique *fos_ids* and *companies* in the Publications table

In [ ]:
```
unique_fosids = list(dict.fromkeys(data_publications["fos_ids"].sum()))
print('The number of unique fos ids is: {}'.format(len(unique_fosids)))
```

In [ ]:
```
data_publications["similar_companies"] = data_publications["similar_companies"].apply(getlist)
unique_companies = list(dict.fromkeys(data_publications["similar_companies"].sum()))
```

In [ ]:
```
print('The number of unique companies is: {}'.format(len(unique_companies)))
```

Finding the most frequent fos_ids and companies in terms of publications.

In [ ]:
```
X = 30
```

```
top_freq_fosids = Counter(data_publications["fos_ids"].sum()).most_common()
print('Top {} frequent fos-ids among the publication samples are:'.format(x),'\n\n{}'.format(top_freq_fosids))
```

In [ ]:
```
top_freq_companies_bvdids = Counter(data_publications["similar_companies"].sum()).most_common()[:X]
# top_freq_companies_name = [sample_data_companies['name'][sample_data_companies['bvd_id'] == i[0]].to_list() for i in to
print('Top {} frequent entities in terms of publication are:'.format(x),'\n\n{}'.format(top_freq_companies_bvdids))
```

Product Table

First, we identify the duplicated instances, as well as missing values in the product table.

In [ ]:
```
data_products = data["Products_data"].copy() #storing the product data in a separate dataframe


#missing values stored in dataframe format
missvalue_product = data_products.isnull().sum()
print('The number of missing values in the product data is:','\n{}'.format(missvalue_product))
```

In [ ]:
```
data_products['index'].describe()
```

In [ ]:
```
top_freq_index = Counter(list(data_products['index'])).most_common()
print('The most frequent companies in the product data are:','\n{}'.format(top_freq_index))
```

In [ ]:
```
data_products[["product_name","index","is_trademark"]].describe() #descriptive statistics including two categorical varia
```

In [ ]:
```
def len_of_dict(dict): # This funtion builds another variable from the dict format data, which shows the number of keywor
    if dict is not None:
        return len(dict)


data_products["#product_keywords"] = data_products["product_keywords"].apply(len_of_dict) #number of keywords for each pr
data_products["#fos_ids"] = data_products["product_fos"].apply(len_of_dict) #number of fos_ids for each product

data_products.describe(include = [np.number])
```

The following cells presents the top frequent fos_ids and keywords in the Product data.

In [ ]:
```
fosids_freq = Counter(data_products["product_fos"].apply(getlist).sum()).most_common()
```

```
print('Number of unique fos_ids in the product data is:\n {}'.format(len(fosids_freq)))
print('Top frequent FOS ids in the product data are:','\n{}'.format(fosids_freq))
```

In [ ]:
```
keywords_freq = Counter(data_products["product_keywords"].sum()).most_common()

print('Number of unique keywords in the product data is:\n {}'.format(len(keywords_freq)))
print('Frequency(%) of keywords in the product data:','\n{}'.format(keywords_freq))
```

Here, we find top companies in terms of number of products.

In [ ]:
```
top_freq_index = Counter(data_products["index"]).most_common()[:3]
top_freq_entityname = [data_companies['name'][data_companies['index'] == i[0]].to_list() for i in top_freq_index]

print('These entities have most number of products (descending format):'.format(top_freq_entityname))
```

FOS Table

This table presents the details of FoS_ids, in terms of FoS name, FOS level in hierarchical structure and Count of publications in Microsoft Academics database with specified FOS.

In [ ]:
```
data_fos = data["FOS"].copy() #storing the product data in a separate dataframe
missvalue_fos = data_fos.isnull().sum()
print('The number of missing values in the FOS data is:','\n{}'.format(missvalue_fos))
```

In [ ]:
```
data_fos['level'] = data_fos['level'].apply(lambda x:str(x))
```

In [ ]:
```
data_fos.describe(include = 'object')
```

In [ ]:
```
Counter(list(data_fos['fos_name'])).most_common()
```

In [ ]:
```
## You can find your desired FOS ids using the following code.
string = 'enter the input'
data_fos['fos_name'][data_fos['fos_name'].str.lower().str.contains(pat = string) == True]
```

FOS Relations Table

In [ ]:
```
data_fosrelations = data["FOS_relations"].copy() #storing the product data in a separate dataframe
missvalue_fosrelations = data_fosrelations.isnull().sum()
print('The number of missing values in the FOS data is:','\n{}'.format(missvalue_fosrelations))
```

In [ ]:
```
data_fosrelations.describe()
```

Finding the most frequent parent/child fos_ids.

In [ ]:
```
x = 5
C5 = Counter(data_fosrelations['parent_id']).most_common()
most_freq_parentFOS = [data_fos['fos_name'][data_fos['fos_id']==i].to_list() for i,counter in C5]
print('Top {} frequent Parents ids are:'.format(x),'\n\n{}'.format(most_freq_parentFOS))
```