Prescribing Data

In [1]:

```
import pandas as pd
import numpy as np
#1. load practice information data file to pandas dataframe
#https://files.digital.nhs.uk/71/B59D99/gp-reg-pat-prac-all.csv**********************
******

#url = "https://files.digital.nhs.uk/71/B59D99/gp-reg-pat-prac-all.csv"
#local_file = r"C:\Data folder\patientinfo_data.csv"
#patientinfo_data = pd.read_csv(url)

#This cell is greyed out to avoid loading the dataa twice and saving memory
#See file loaded below with custom names.
```

In [2]:

```
#2a. Eplore data; what columns are in the data
#patientinfo_data.columns
```

In [3]:

```
#2b. Explore data; what type of objects are in the dataframe
#patientinfo_data.dtypes
```

In [4]:

```
#3. Create custom column names and read in file again

import pandas as pd
import numpy as np

cols = [
    'publication',
    'extract_date',
    'type',
    'ccg_code',
    'ons_ccg_code',
    'code',
    'postcode',
    'sex',
    'age',
    'no_of_patients'
]

#patientinfo_data url = "https://files.digital.nhs.uk/71/B59D99/gp-reg-pat-prac-all.csv"

patientinfo_data = pd.read_csv(r'https://files.digital.nhs.uk/71/B59D99/gp-reg-pat-prac-all.csv', header=None, names=cols, index_col=False,skiprows=1)
patientinfo_data.head()
```

Out[4]:

	publication	extract_date	type	ccg_code	ons_ccg_code	code	postcode	sex
0	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83005	DL1 3RT	ALL
1	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83006	DL3 6HZ	ALL
2	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83010	DL3 9JP	ALL
3	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83013	DL1 4YL	ALL
4	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83031	DL3 8SQ	ALL

In [5]:

```
#4. Extract postcode area (first two characters from the postcodes string) for all prac
tices

patientinfo_data.dropna(inplace = True)

patientinfo_data['postcode_area'] = patientinfo_data["postcode"].str[:2]

patientinfo_data.head()
```

Out[5]:

	publication	extract_date	type	ccg_code	ons_ccg_code	code	postcode	sex
0	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83005	DL1 3RT	ALL
1	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83006	DL3 6HZ	ALL
2	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83010	DL3 9JP	ALL
3	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83013	DL1 4YL	ALL
4	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83031	DL3 8SQ	ALL

In [6]:

```
#Load UK prescribing dataset into pandas dataframe
cols2 = [
    'sha',
    'pct',
    'practice',
    'bnf_code',
    'bnf_name',
    'items',
    'nic',
    'act_cost',
    'quantity',
    'period'
]


presc_data_url = 'http://datagov.ic.nhs.uk/presentation/2018_04_April/T201804PDPI+BNFT.CSV'


presc_data = pd.read_csv(r'http://datagov.ic.nhs.uk/presentation/2018_04_April/T201804PDPI+BNFT.CSV',header=None, names=cols2, index_col=False, skiprows=1)
presc_data.head()
```

Out[6]:

	sha	pct	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	Q44	RTV	Y04937	0401010Z0AAAAAA	Zopiclone_Tab 7.5mg	6	1.56	2.12	63
1	Q44	RTV	Y04937	0401020K0AAAHAH	Diazepam_Tab 2mg	4	0.87	1.15	73
2	Q44	RTV	Y04937	0401020K0AAAIAI	Diazepam_Tab 5mg	2	0.46	0.56	35
3	Q44	RTV	Y04937	0402010ABAAABAB	Quetiapine_Tab 25mg	1	2.60	2.52	14
4	Q44	RTV	Y04937	0402010ADAAAAAA	Aripiprazole_Tab 10mg	1	1.53	1.53	14

In [7]:

```
patientinfo_data["code"]= patientinfo_data["code"].astype(str)
patientinfo_data = patientinfo_data[['code', 'postcode','postcode_area','no_of_patient
s']]

patientinfo_data.head()
```

Out[7]:

	code	postcode	postcode_area	no_of_patients
0	A83005	DL1 3RT	DL	11826
1	A83006	DL3 6HZ	DL	8044
2	A83010	DL3 9JP	DL	14070
3	A83013	DL1 4YL	DL	11298
4	A83031	DL3 8SQ	DL	10109

In [8]:

```
# Merge patientinfo_data with prescribing data

presc_patient_data = pd.DataFrame.merge(patientinfo_data, presc_data, left_on='code', r
ight_on='practice', how = 'right')
presc_patient_data.head()
```

Out[8]:

	code	postcode	postcode_area	no_of_patients	sha	pct	practice	bnf_cc
0	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0101010G0AAAB
1	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0101010G0BCAB
2	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0101012B0AAAB
3	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0101021B0AAAH
4	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0101021B0AAAL

In [9]:

```
## To identify GP practice prescribing in the London and Cambridge postcode areas:

#There are 8 london postal areas(excluding those in the Greater London area) and 1 Camb
ridge post code area
#london_postcodes = ['SE', 'E', 'SW', 'W', 'NW', 'N', 'EC', 'WC']
#Cambridge_postcodes = ['CB']
```

In [10]:

```
#Filter out London and Cambridge practices from new dataset (presc_patient_data)

postcodes = ['SE', 'E', 'SW', 'W', 'NW', 'N', 'EC', 'WC','CB']

#Combined London/Cambridge data
cb_ldn_patient_data = presc_patient_data.loc[presc_patient_data['postcode_area'].isin(p
ostcodes)]

# convert act_cost to float (otherwise it is rounded)
cb_ldn_patient_data["act_cost"]= cb_ldn_patient_data["act_cost"].astype(float)
cb_ldn_patient_data.head()
```

```
C:\Users\fouad\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

Out[10]:

	code	postcode	postcode_area	no_of_patients	sha	pct	practice	bn
4113292	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0A
4113293	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0A
4113294	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0BI
4113295	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0B
4113296	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0B

In [11]:

```
cb_ldn_patient_data.tail()
```

Out[11]:

	code	postcode	postcode_area	no_of_patients	sha	pct	practice	bnf_code
6357585	Y02260	SW1W 8NA	SW	3716.0	Q62	09A	Y02260	23803378003
6357586	Y02260	SW1W 8NA	SW	3716.0	Q62	09A	Y02260	23850108508
6357587	Y02260	SW1W 8NA	SW	3716.0	Q62	09A	Y02260	23850108513
6357588	Y02260	SW1W 8NA	SW	3716.0	Q62	09A	Y02260	23850708519
6357589	Y02260	SW1W 8NA	SW	3716.0	Q62	09A	Y02260	23960109756

In [12]:

```
# Derive total cost of each prescription item prescribed in the London/Cambridge practi
ces

cb_ldn_patient_data["total_cost"] = cb_ldn_patient_data["act_cost"] * cb_ldn_patient_da
ta["items"]
cb_ldn_patient_data
cb_ldn_patient_data.head()
```

```
C:\Users\fouad\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing import
s until
```

Out[12]:

	code	postcode	postcode_area	no_of_patients	sha	pct	practice	bn
4113292	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0A
4113293	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0A
4113294	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0BI
4113295	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0B
4113296	D81001	CB2 1EH	CB	12057.0	Q56	06H	D81001	0101021B0B

In [13]:

```
# Derive prescribing data for London patients only
ldn_postcodes = ['SE', 'E', 'SW', 'W', 'NW', 'N', 'EC', 'WC']

# filter for london patients only
ldn_patients = cb_ldn_patient_data.loc[cb_ldn_patient_data['postcode_area'].isin(ldn_po
stcodes)]
```

In [14]:

```
ldn_patients = ldn_patients[['code','postcode_area','items','total_cost','no_of_patients']]
ldn_patients.head()
```

Out[14]:

	code	postcode_area	items	total_cost	no_of_patients
4997090	E83006	NW	1	2.79	6885.0
4997091	E83006	NW	2	19.04	6885.0
4997092	E83006	NW	2	11.26	6885.0
4997093	E83006	NW	2	44.74	6885.0
4997094	E83006	NW	23	2910.65	6885.0

In [15]:

```
# As seen above, practice code and no_of_patients occur multiple times for the different items prescribed
# To avoid duplication, group and aggregate these variables

unique_ldn = ldn_patients.groupby("code").agg({"no_of_patients": np.unique,"items": np.sum, "total_cost": np.sum})

unique_ldn.head()
```

Out[15]:

code	no_of_patients	items	total_cost
E83006	6885.0	6327	541939.37
E83009	10822.0	10664	1349763.75
E83011	8116.0	11056	1577413.91
E83016	18356.0	16596	3953910.30
E83020	10855.0	12158	3304519.70

In [16]:

```
#Total number of patients registered in London
all_london_patients = unique_ldn['no_of_patients'].sum()
print (all_london_patients)
```

```
3096869.0
```

In [17]:

```
#Total number of prescriptions in London
total_london_prescriptions = unique_ldn['items'].sum()
print (total_london_prescriptions)
```

```
2875268
```

In [18]:

```
#Total cost of prescriptions issued in London
total_london_prescriptions_cost = unique_ldn['total_cost'].sum()
print (total_london_prescriptions_cost)
```

```
478504601.2899997
```

In [19]:

```
## Top 10 most frequent drugs prescribed in London

# filter for london data from ldn_patients

ldn_patients = cb_ldn_patient_data.loc[cb_ldn_patient_data['postcode_area'].isin(ldn_po
stcodes)]

# do a count of all BNF names in london data
ldn_patients['bnf_name'].value_counts()
london_freq_presc = ldn_patients['bnf_name'].value_counts()
london_freq_presc.head(10)
```

Out[19]:

```
GlucoRX FinePoint Needles Pen Inj Screw     414
Salbutamol_Inha 100mcg (200 D) CFF          364
Amlodipine_Tab 10mg                         364
Sertraline HCl_Tab 50mg                     363
Levothyrox Sod_Tab 100mcg                   363
Cetirizine HCl_Tab 10mg                     363
Amlodipine_Tab 5mg                          363
Aspirin Disper_Tab 75mg                     363
Atorvastatin_Tab 20mg                       363
Metformin HCl_Tab 500mg                     363
Name: bnf_name, dtype: int64
```

In [20]:

```
# Bottom 10 less frequently prescribed drugs in London
london_freq_presc.tail(10)
```

Out[20]:

```
Hydrosorb 10cm x 10cm Wound Dress H/Gel     1
ConjOestro/Bazedoxifene_Tab450mcg/20mgMR    1
Imperm Plas 2.5cm x 3m Surg Adh Tape        1
Actico 6cm x 6m Short Stch Compress Band    1
Diamorph HCl_Inj 10mg Amp                   1
Quinapril HCl_Tab 5mg                       1
Sterilance Lite II Safety Lancets 1.8mm/    1
Flexicare_Discreet Leg Bag Ster Short Tu    1
Acticoat 7 5cm x 5cm Wound Dress Silver     1
365 Film 4cm x 5cm VP Adh Film Dress        1
Name: bnf_name, dtype: int64
```

In [21]:

```
## Derive prescibing data for Cambridge patients only

cam_postcodes = ['CB']

# filter for cambridge patients only
cam_patients = cb_ldn_patient_data.loc[cb_ldn_patient_data['postcode_area'].isin(cam_po
stcodes)]
cam_patients = cam_patients[['code','postcode_area','items','total_cost','no_of_patient
s']]
cam_patients.head()
```

Out[21]:

	code	postcode_area	items	total_cost	no_of_patients
4113292	D81001	CB	3	33.87	12057.0
4113293	D81001	CB	3	41.73	12057.0
4113294	D81001	CB	1	11.10	12057.0
4113295	D81001	CB	2	17.92	12057.0
4113296	D81001	CB	2	55.14	12057.0

In [22]:

```
## Group and aggregate dataset to avoid duplication

unique_cam = cam_patients.groupby("code").agg({"no_of_patients": np.unique,"items": np.
sum, "total_cost": np.sum})
unique_cam.head()
```

Out[22]:

code	no_of_patients	items	total_cost
D81001	12057.0	6679	674090.97
D81002	16939.0	13885	2387339.20
D81003	9927.0	11572	1239615.72
D81005	14941.0	6918	729344.38
D81009	9071.0	11378	1465175.04

In [23]:

```
#Total number of patients registered in Cambridge
all_cambridge_patients = unique_cam['no_of_patients'].sum()
print (all_cambridge_patients)
```

```
508816.0
```

In [24]:

```
#Total number of prescriptions issued in Cambridge
total_cambridge_prescriptions = unique_cam['items'].sum()
print (total_cambridge_prescriptions)
```

```
658365
```

In [25]:

```
#Total cost of prescription items issued in Cambridge
total_cambridge_prescriptions_cost = unique_cam['total_cost'].sum()
print (total_cambridge_prescriptions_cost)
```

```
142430420.8100001
```

In [26]:

```
# Top 10 most frequent drugs prescribed in Cambridge
# filter for Cambridge data from merged table
cam_patients = cb_ldn_patient_data.loc[cb_ldn_patient_data['postcode_area'].isin(cam_po
stcodes)]

# do a count of all BNF names in Cambridge data
cam_patients['bnf_name'].value_counts()
cambridge_freq_presc = cam_patients['bnf_name'].value_counts()
cambridge_freq_presc.head(10)
```

Out[26]:

```
GlucoRX FinePoint Needles Pen Inj Screw     134
3m Health Care_Cavilon Durable Barrier C     83
Ramipril_Cap 5mg                             50
Zopiclone_Tab 3.75mg                         50
Codeine Phos_Tab 15mg                        50
Sertraline HCl_Tab 100mg                     50
Liquifilm 1.4% Polyvinyl Alcohol Eye Dps     50
Citalopram Hydrob_Tab 10mg                   50
Ramipril_Cap 1.25mg                          50
Losartan Pot_Tab 100mg                       50
Name: bnf_name, dtype: int64
```

In [27]:

```
# Top 10 bottom frequent drugs prescribed in Cambridge
cambridge_freq_presc.tail(10)
```

Out[27]:

```
ActiLymph Class 2 B/Knee Closed Toe Sand    1
Tacrolimus_Cap 1mg                          1
Allevyn Ag Heel 10.5cm x 13.5cm Wound Dr    1
InsDegludec/Liraglutide_100u/3.6mg/mlPfP    1
Skinnies Viscose Leggings 5-8 Yrs Elasct    1
Clexane_Inj 100mg/ml 0.8ml Pfs              1
Jobst Elvarex Acc For U/Extrem 1finger C    1
Canesten_Vag Tab 200mg + Applic             1
Tamurex_Cap 400mcg M/R                      1
Flomax Relief MR_Cap 400mcg                 1
Name: bnf_name, dtype: int64
```

In [28]:

```
## Analysis summary

#From analysis of the practice information and prescribing datasets, it can be seen tha
t there are significantly greater number
#of patients registered in London practices when compared to Cambridge.
#Consequently, higher number of prescriptions were prescribed in April 2018 ;and at a h
igher cost to the NHS
```

In [29]:

```
# Use descriptive statistics to compare London and Cambridge prescribing data

unique_cam.describe()
```

Out[29]:

	no_of_patients	items	total_cost
count	50.000000	50.000000	5.000000e+01
mean	10176.320000	13167.300000	2.848608e+06
std	5659.268149	9153.732296	4.520173e+06
min	568.000000	1622.000000	7.383125e+04
25%	6326.000000	7228.000000	6.879043e+05
50%	9499.000000	11674.000000	1.528521e+06
75%	12474.000000	14905.500000	2.527372e+06
max	33501.000000	54589.000000	2.768612e+07

In [30]:

```
unique_ldn.describe()
```

Out[30]:

	no_of_patients	items	total_cost
count	365.000000	365.000000	3.650000e+02
mean	8484.572603	7877.446575	1.310972e+06
std	5880.075963	6033.933067	3.902328e+06
min	183.000000	3.000000	5.940000e+00
25%	5032.000000	4572.000000	3.400041e+05
50%	7483.000000	6648.000000	7.045388e+05
75%	10822.000000	10125.000000	1.383768e+06
max	72227.000000	77054.000000	7.039040e+07

In [31]:

```
## Cardiovascular drugs data

presc_patient_data['drug_code'] = presc_patient_data['bnf_code'].str[:2]
cv_drugs = presc_patient_data[presc_patient_data['drug_code'] == '02']
cv_drugs.head()
```

Out[31]:

	code	postcode	postcode_area	no_of_patients	sha	pct	practice	bnf_co
106	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0201010F0AAADA
107	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0201010F0AAAEA
108	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0201010F0AAAFA
109	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0202010B0AAABA
110	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0202010B0AAACA

In [32]:

```
# Derive total cost for all cardiovascular drug prescriptions in every England practice


cv_drugs['total_cost'] = cv_drugs['act_cost'] * cv_drugs['items']
total_cv_costs = cv_drugs['total_cost'].sum()
print (total_cv_costs)
```

```
5448656348.369999

C:\Users\fouad\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing import
s until
```

In [33]:

```
# Derive total number of all cardiovascular drug prescriptions in every England practic
e

total_cv_no = cv_drugs['items'].sum()
print (total_cv_no)
```

```
26449832
```

In [34]:

```
## Antidepressant drugs data

presc_patient_data['drug_code'] = presc_patient_data['bnf_code'].str[:4]
antidep_drugs = presc_patient_data[presc_patient_data['drug_code'] == '0403']
antidep_drugs.head()
```

Out[34]:

	code	postcode	postcode_area	no_of_patients	sha	pct	practice	bnf_co
480	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0403010B0AAAG/
481	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0403010B0AAAH/
482	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0403010B0AAA
483	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0403010B0AAAN/
484	A83005	DL1 3RT	DL	11826.0	Q45	00C	A83005	0403010F0AAAA/

In [35]:

```
# Derive total cost for all antidepressant prescriptions in every England practice

presc_patient_data['drug_code'] = presc_patient_data['bnf_code'].str[:4]
antidep_drugs = presc_patient_data[presc_patient_data['drug_code'] == '0403']

antidep_drugs['total_cost'] = antidep_drugs['act_cost'] * antidep_drugs['items']
total_antidep_costs = antidep_drugs['total_cost'].sum()
print (total_antidep_costs)
```

```
925174735.9200002

C:\Users\fouad\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
```

In [36]:

```
# Derive total number of all antideprtessant drug prescriptions in every England practi
ce

total_antidep_no = antidep_drugs['items'].sum()
print (total_antidep_no)
```

```
5715873
```

In [37]:

```
#Describe the total spending and the relative costs per patient across all practices fo
r April 2018
```

In [38]:

```
##Describe total spending and the relative costs per patient across all practices for A
pril 2018 using a scatter plot

import matplotlib.pyplot as plt

presc_patient_data["total_cost"] = presc_patient_data["act_cost"] * presc_patient_data[
"items"]
scatter_data = presc_patient_data[['code','postcode_area','items','total_cost','no_of_p
atients']]
scatter_chart_data = scatter_data.groupby("code").agg({"no_of_patients": np.unique,"ite
ms": np.sum, "total_cost": np.sum})


ax3=scatter_chart_data.plot(kind='scatter', x='total_cost',y='no_of_patients', title=
'Scatter plot: monthly total spend per registered patients',figsize=(8,6))
ax3.set_xlabel("monthly total spending per surgery",fontsize=12)
ax3.set_ylabel("total number of registered patients",fontsize=12)
ax3.set_xlim(0, 900000)
ax3.set_ylim(0, 15000)
```

Out[38]:

```
(0, 15000)
```

WHO Mortality data

In [39]:

```
# Load WHO ICD mortality data 1 AND 2 into pandas dataframe

import pandas as pd
import numpy as np

#mortality_data_1 url = r"https://www.who.int/healthinfo/statistics/Morticd10_part1.zip?ua=1"
#mortality_data_2 url = r"https://www.who.int/healthinfo/statistics/Morticd10_part2.zip?ua=1"


# load mortality data 1 file
mortality_data1 = pd.read_csv(r"https://www.who.int/healthinfo/statistics/Morticd10_part1.zip?ua=1",compression='zip',index_col=False, skiprows=0)

# load second mortality data file
mortality_data2 = pd.read_csv(r"https://www.who.int/healthinfo/statistics/Morticd10_part2.zip?ua=1",compression='zip',index_col=False, skiprows=0)

#Append mortality data 1 to mortality data 2
mortality_data = mortality_data1.append(mortality_data2, ignore_index = True)

mortality_data.head()
```

```
C:\Users\fouad\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2785: DtypeWarning: Columns (4) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
C:\Users\fouad\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2785: DtypeWarning: Columns (2,4) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[39]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Deaths1	...	Deaths
0	1400	NaN	NaN	2001	101	1000	1	7	8	332	...	9
1	1400	NaN	NaN	2001	101	1000	2	7	8	222	...	11
2	1400	NaN	NaN	2001	101	1001	1	7	8	24	...	
3	1400	NaN	NaN	2001	101	1001	2	7	8	14	...	
4	1400	NaN	NaN	2001	101	1002	1	7	8	0	...	

5 rows × 39 columns

In [40]:

```
# Load population data

who_pop_data = pd.read_csv(r'https://www.who.int/healthinfo/Pop.zip?ua=1',compression=
'zip', low_memory=False)
who_pop_data.head()
```

Out[40]:

	Country	Admin1	SubDiv	Year	Sex	Frmat	Pop1	Pop2	Pop3	Pop4	...	
0	1060	NaN	NaN	1980	1	7	137100.0	3400.0	15800.0	NaN	...	
1	1060	NaN	NaN	1980	2	7	159000.0	4000.0	18400.0	NaN	...	
2	1125	NaN	NaN	1955	1	2	5051500.0	150300.0	543400.0	NaN	...	1
3	1125	NaN	NaN	1955	2	2	5049400.0	145200.0	551000.0	NaN	...	1
4	1125	NaN	NaN	1956	1	2	5353700.0	158700.0	576600.0	NaN	...	1

5 rows × 33 columns

In [41]:

```
# Load country code lookup data


country_lookup_data = pd.read_csv(r'https://www.who.int/healthinfo/statistics/country_c
odes.zip?ua=1',compression='zip',index_col=False, skiprows=0)
country_lookup_data.head()
```

Out[41]:

	country	name
0	1010	Algeria
1	1020	Angola
2	1025	Benin
3	1030	Botswana
4	1035	Burkina Faso

In [42]:

```
#Total deaths for all years

who_total_deaths = pd.DataFrame.merge(mortality_data, country_lookup_data, left_on='Cou
ntry', right_on='country', how = 'inner')
who_total_deaths.head()
```

Out[42]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Deaths1	...	Deaths
0	1400	NaN	NaN	2001	101	1000	1	7	8	332	...	N
1	1400	NaN	NaN	2001	101	1000	2	7	8	222	...	N
2	1400	NaN	NaN	2001	101	1001	1	7	8	24	...	N
3	1400	NaN	NaN	2001	101	1001	2	7	8	14	...	N
4	1400	NaN	NaN	2001	101	1002	1	7	8	0	...	N

5 rows × 41 columns

In [50]:

```
#Total deaths in 2010 for Iceland, Italy and New Zealand

mort_2010 =  who_total_deaths['Year']==2010
who_total_deaths_2010 = who_total_deaths[mort_2010]
countries = ['Iceland', 'Italy', 'New Zealand']

# filter for these countries only
who_total_deaths_2010_df = who_total_deaths_2010.loc[who_total_deaths_2010['name'].isin
(countries)]
who_deaths = who_total_deaths_2010_df.groupby('name').agg({"Deaths1": np.sum})

print (who_deaths)
```

```
            Deaths1
name
Iceland        4038
Italy       1169230
New Zealand   57298
```

In [51]:

```
# Population in 2010 for Iceland, Italy and New Zealand

who_pop_data1 = pd.DataFrame.merge(who_pop_data, country_lookup_data, left_on='Country'
, right_on='country', how = 'inner')
who_pop_2010 = who_pop_data1.query('Year == "2010"')

countries = ['Iceland', 'Italy', 'New Zealand']

# filter for these countries only
who_pop_2010b = who_pop_2010.loc[who_pop_2010['name'].isin(countries)]
who_pop = who_pop_2010b.groupby('name').agg({"Pop1": np.sum})

who_pop.head()
```

Out[51]:

name	Pop1
Iceland	318041.0
Italy	60483386.0
New Zealand	4367360.0

In [52]:

```
#Distribution of deaths (all causes, all years) by age group in Italy


is_italy = who_total_deaths['name']=='Italy'
italy_death = who_total_deaths[is_italy]
italy_death_grouping = italy_death.iloc[:, 10:39]
italy_death_grouping.head()
```

Out[52]:

	Deaths2	Deaths3	Deaths4	Deaths5	Deaths6	Deaths7	Deaths8	Deaths9	Deat
2618040	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2618041	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2618042	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2618043	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2618044	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 29 columns

In [53]:

```
#Histogram to visualise distribution of deaths in Italy

#import matplotlib.pyplot as plt

#x= italy_death_grouping
#plt.hist(x, bins=20)
#plt.xlabel("Number of Deaths")
#plt.ylabel("Frequency")

#min_x = floor(x.quantile(.01))
#max_x = floor(x.quantile(.99))
#plt.xlim(min_x,max_x)
#plt.title("Italy - Death by Age Group")
```

In [54]:

```
#Deaths in Italy due to neoplasms ICD10-category (C00-D48)

#Filter deaths in Italy from the all causes
is_italy = who_total_deaths['name']=='Italy'
italy_death = who_total_deaths[is_italy]

italy_death.head()
```

Out[54]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Deaths1	.
2618040	4180	NaN	NaN	2003	104	A010	1	0	1	1	.
2618041	4180	NaN	NaN	2003	104	A020	1	0	1	11	.
2618042	4180	NaN	NaN	2003	104	A020	2	0	1	5	.
2618043	4180	NaN	NaN	2003	104	A021	1	0	1	1	.
2618044	4180	NaN	NaN	2003	104	A021	2	0	1	1	.

5 rows × 41 columns

In [55]:

```
#Derive data for all deaths in Italy in 2010

is_2010 =  italy_death['Year']==2010
italy_death_2010 = italy_death[is_2010]
italy_death_2010.head()
```

Out[55]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Deaths1	...
2652376	4180	NaN	NaN	2010	104	A010	1	0	1	1	...
2652377	4180	NaN	NaN	2010	104	A020	1	0	1	3	...
2652378	4180	NaN	NaN	2010	104	A020	2	0	1	1	...
2652379	4180	NaN	NaN	2010	104	A021	1	0	1	6	...
2652380	4180	NaN	NaN	2010	104	A021	2	0	1	2	...

5 rows × 41 columns

In [56]:

```
#Derive data for deaths in Italy in 2010 due to neoplasms (ICD C00-D48)

neoplasm_italy = italy_death_2010.loc[italy_death_2010['Cause'].between('C00','D48')]
neoplasm_italy.head()
```

Out[56]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Deaths1	.
2652668	4180	NaN	NaN	2010	104	C000	1	0	1	3	.
2652669	4180	NaN	NaN	2010	104	C000	2	0	1	4	.
2652670	4180	NaN	NaN	2010	104	C001	1	0	1	17	.
2652671	4180	NaN	NaN	2010	104	C001	2	0	1	10	.
2652672	4180	NaN	NaN	2010	104	C006	1	0	1	1	.

5 rows × 41 columns

In [57]:

```
#Generate a table with the cause of death, the number of deaths, and the proportion of
 overall deaths

neoplasm_italy_data = neoplasm_italy.groupby('Cause').agg({'Deaths1':'count',})
print(neoplasm_italy_data)
```

Cause	Deaths1
C000	2
C001	2
C006	1
C009	2
C01	2
C021	1
C022	1
C023	1
C024	1
C029	2
C030	2
C031	2
C039	2
C040	1
C049	2
C050	2
C051	2
C052	2
C059	2
C060	2
C061	1
C062	2
C068	2
C069	2
C07	2
C080	2
C081	1
C089	2
C090	1
C091	1
...	...
D414	2
D417	1
D419	2
D421	1
D429	2
D430	2
D431	2
D432	2
D433	1
D434	1
D437	2
D439	1
D440	2
D441	2
D443	2
D444	2
D445	2
D446	1
D448	1
D449	2
D45	2
D462	2
D464	2
D467	2
D469	2
D471	2
D472	2
D473	2

```
D477          2
D479          2

[440 rows x 1 columns]
```

In [58]:

```
#Deaths in Australia in 2010 due to Neoplasms
is_australia = who_total_deaths['name']=='Australia'
australia_death = who_total_deaths[is_australia]

is_2010 =  australia_death['Year']==2010
australia_death_2010 = australia_death[is_2010]
australia_death_2010.head()
```

Out[58]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Deaths1	.
3458547	5020	NaN	NaN	2010	104	A020	1	0	1	1	.
3458548	5020	NaN	NaN	2010	104	A020	2	0	1	4	.
3458549	5020	NaN	NaN	2010	104	A021	1	0	1	3	.
3458550	5020	NaN	NaN	2010	104	A021	2	0	1	1	.
3458551	5020	NaN	NaN	2010	104	A047	1	0	1	16	.

5 rows × 41 columns

In [59]:

```
#Derive data for deaths in Australia due to neoplasms
neoplasm_australia = australia_death_2010.loc[australia_death_2010['Cause'].between('C00','D48')]
neoplasm_australia.head()
```

Out[59]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Deaths1	.
3458743	5020	NaN	NaN	2010	104	C001	1	0	1	2	.
3458744	5020	NaN	NaN	2010	104	C001	2	0	1	2	.
3458745	5020	NaN	NaN	2010	104	C009	1	0	1	2	.
3458746	5020	NaN	NaN	2010	104	C009	2	0	1	5	.
3458747	5020	NaN	NaN	2010	104	C01	1	0	1	20	.

5 rows × 41 columns