In [9]:

```
#https://github.com/imasha1/DMHR #Github URL

import pandas
import pandasql
from datetime import datetime
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib.dates as dates
from scipy.stats import norm
import numpy as np
sql = pandasql.PandaSQL()

#Need to import Pandas library, PandaSQL, Matplotlib, Numpys etc in Jupyter Notebook so it’s loade
d into the memory and is available to work with
#Add ‘as plt’ and ‘as np’ so can access Plotting library and Numpys just by writing ‘plt.command’
etc every time you need to use it.
#A date in Python is not a data type of its own, so need to import it as datetime

def find_population_and_deaths_neoplasms(population_data, mortality_data, country_code):
  return sql('select p.pop1 as population, sum(m.deaths1) as deaths from population_data p left jo
in mortality_data m on p.country = m.country where m.year=2010 and p.year=2010 and m.cause >= "C00
" and m.cause <= "D48" and p.country = ' + str(country_code) + ' group by population')

#Def defines a funtion, in this case function is defined as to find population and deaths neoplasm
s, with the parameters being population data, mortality data and country code
#This code is using and Pandas and PandaSQL
#The return statement causes your find population and deaths neoplasms function to exit and hand b
ack a value to its caller

def find_population_and_deaths(population_data, mortality_data, country_code):
  return sql('select p.pop1 as population, sum(m.deaths1) as deaths from population_data p left jo
in mortality_data m on p.country = m.country where m.year=2010 and p.year=2010 and p.country = ' +
str(country_code) + ' group by population')

#Def defines a funtion, in this case function is defined as to find population and deaths, with th
e parameters being population data, mortality data and country code
#This code is using and Pandas and PandaSQL
#The return statement causes your find population and deaths function to exit and hand back a valu
e to its caller


def find_country_code(country_code_data, country_name):
  return sql('select country from country_code_data where name = "' + country_name + '"').country[0
]

#Def defines a funtion, in this case function is defined as to find country code, with the paramet
ers being country code data and country name
#This code is using and Pandas and PandaSQL
#The return statement causes your find country code function to exit and hand back a value to its
caller


low_memory=False
```

In [10]:

```
mortality_data = pandas.read_csv('Morticd10') # combine part 1 and part 2 mortality datan files
manually beforehand on disk then import mortality data from WHO as panda and naming it as mortalit
y_data
population_data = pandas.read_csv('pop') #importing population data from WHO as panda and naming i
t as population_data
country_code_data = pandas.read_csv('country_codes') #importing country codes data from WHO as
panda and naming it as country_code_data
iceland_country_code = find_country_code(country_code_data, 'Iceland') #Identifying country code u
sing Panda and PandaSQL
italy_country_code = find_country_code(country_code_data, 'Italy')
new_zealand_country_code = find_country_code(country_code_data, 'New Zealand')
australia_country_code = find_country_code(country_code_data, 'Australia')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3020: DtypeWarning:
```

```
Columns
(0,1,2,3,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
38) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [11]:

```python
# identifyin iceland population and the total number of deaths using pandasql
find_population_and_deaths(population_data, mortality_data, iceland_country_code)
```

Out[11]:

	population	deaths
0	158070.0	4038
1	159971.0	4038

In [12]:

```python
# identifying italy population and the total number of deaths using pandasql
find_population_and_deaths(population_data, mortality_data, italy_country_code)
```

Out[12]:

	population	deaths
0	29350339.0	1169230
1	31133047.0	1169230

In [14]:

```python
# identifying new_zealand population and the total number of deaths using pandasql
find_population_and_deaths(population_data, mortality_data, new_zealand_country_code)
```

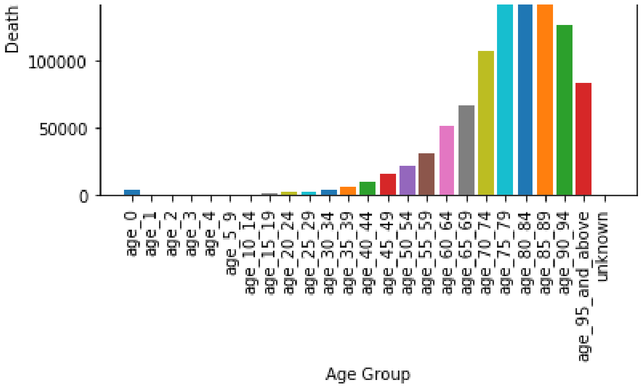
Out[14]:

	population	deaths
0	2144390.0	57298
1	2222970.0	57298

In [15]:

```python
# What was the distribution of deaths (all causes, all years) by age group in Italy
plot_data = sql('select sum(deaths2) as age_0, sum(deaths3) as age_1, sum(deaths4) as age_2, sum(d
eaths5) as age_3, sum(deaths6) as age_4, sum(deaths7) as age_5_9, sum(deaths8) as age_10_14, sum(d
eaths9) as age_15_19, sum(deaths10) as age_20_24, sum(deaths11) as age_25_29, sum(deaths12) as
age_30_34, sum(deaths13) as age_35_39, sum(deaths14) as age_40_44, sum(deaths15) as age_45_49,
sum(deaths16) as age_50_54, sum(deaths17) as age_55_59, sum(deaths18) as age_60_64, sum(deaths19)
as age_65_69, sum(deaths20) as age_70_74, sum(deaths21) as age_75_79, sum(deaths22) as age_80_84,
sum(deaths23) as age_85_89, sum(deaths24) as age_90_94, sum(deaths25) as age_95_and_above,
sum(deaths26) as unknown from mortality_data where year=2010 and country = ' +
str(italy_country_code))
for age_group in plot_data:
  plt.bar(age_group, plot_data[age_group])
plt.ylabel('Deaths')
plt.xlabel('Age Group')
plt.xticks(rotation=90)
plt.show()

#below is visualisation of distribution of death by age group in italy.
```



In [16]:

```
# Generate a table with the cause of death, the number of deaths, and the proportion of overall de
aths
table_data = sql('select cause as cause, sum(deaths1) as deaths from mortality_data where cause >=
"C00" and cause <= "D48" and year = 2010 and country = ' + str(italy_country_code) + ' group by
cause order by deaths DESC')
total_deaths = float(sql('select sum(deaths) as total_deaths from table_data').total_deaths)
table_data = sql('select *, (deaths / ' + str(total_deaths) + ') as proportion from table_data')
```

In [18]:

```
print(table_data) # print table_data for table
```

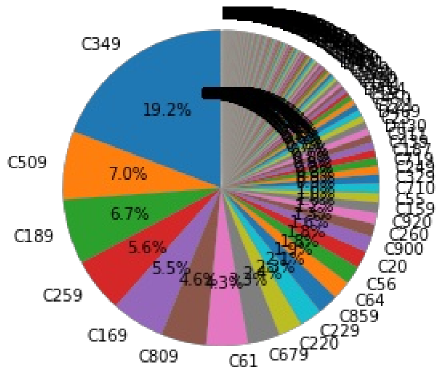
```
     cause  deaths  proportion
0     C349   33416    0.191600
1     C509   12231    0.070130
2     C189   11638    0.066730
3     C259    9683    0.055520
4     C169    9523    0.054603
5     C809    8036    0.046077
6      C61    7509    0.043055
7     C679    5675    0.032539
8     C220    4257    0.024409
9     C229    4018    0.023038
10    C859    3660    0.020986
11     C64    3361    0.019271
12     C56    3193    0.018308
13     C20    3101    0.017780
14    C900    2831    0.016232
15    C260    2240    0.012844
16    C920    2067    0.011852
17    C159    1823    0.010453
18     C55    1691    0.009696
19    C710    1687    0.009673
20    C329    1591    0.009122
21    C249    1532    0.008784
22    C719    1482    0.008497
23    C187    1406    0.008062
24    C439    1377    0.007895
25    C911    1317    0.007551
26    D430    1307    0.007494
27     C23    1199    0.006875
28    D469    1192    0.006835
29    C221    1139    0.006531
..     ...     ...         ...
410   C811       1    0.000006
411   C812       1    0.000006
412   C820       1    0.000006
413   C827       1    0.000006
414   C917       1    0.000006
415   C922       1    0.000006
416   C960       1    0.000006
417   D049       1    0.000006
418   D075       1    0.000006
419   D125       1    0.000006
420   D130       1    0.000006
421   D152       1    0.000006
422   D160       1    0.000006
423   D164       1    0.000006
```

```
424   D165        1     0.000006
425   D171        1     0.000006
426   D172        1     0.000006
427   D179        1     0.000006
428   D213        1     0.000006
429    D24        1     0.000006
430   D331        1     0.000006
431    D34        1     0.000006
432   D351        1     0.000006
433   D369        1     0.000006
434   D409        1     0.000006
435   D417        1     0.000006
436   D421        1     0.000006
437   D433        1     0.000006
438   D446        1     0.000006
439   D448        1     0.000006

[440 rows x 3 columns]
```

In [19]:

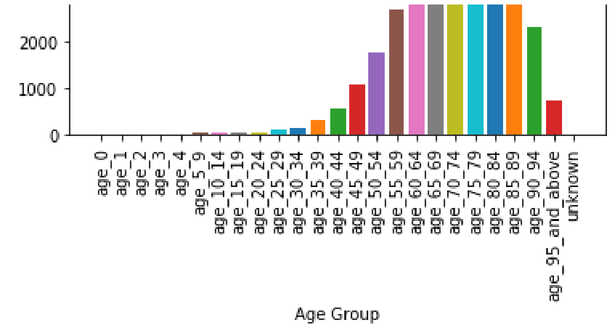
```python
# Generate a pie chart to visualize the proportion of deaths
_fig1, ax1 = plt.subplots()
ax1.pie(table_data.deaths, labels=table_data.cause, autopct='%1.1f%%', startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



In [20]:

```python
# Are there differences by age group for deaths from Neoplasms (C00-D48) in Australia for 2010
plot_data = sql('select sum(deaths2) as age_0, sum(deaths3) as age_1, sum(deaths4) as age_2, sum(d
eaths5) as age_3, sum(deaths6) as age_4, sum(deaths7) as age_5_9, sum(deaths8) as age_10_14, sum(d
eaths9) as age_15_19, sum(deaths10) as age_20_24, sum(deaths11) as age_25_29, sum(deaths12) as
age_30_34, sum(deaths13) as age_35_39, sum(deaths14) as age_40_44, sum(deaths15) as age_45_49,
sum(deaths16) as age_50_54, sum(deaths17) as age_55_59, sum(deaths18) as age_60_64, sum(deaths19)
as age_65_69, sum(deaths20) as age_70_74, sum(deaths21) as age_75_79, sum(deaths22) as age_80_84,
sum(deaths23) as age_85_89, sum(deaths24) as age_90_94, sum(deaths25) as age_95_and_above,
sum(deaths26) as unknown from mortality_data where cause >= "C00" and cause <= "D48" and year=2010
and country = ' + str(australia_country_code))
for age_group in plot_data:
  plt.bar(age_group, plot_data[age_group])
plt.ylabel('Deaths')
plt.xlabel('Age Group')
plt.xticks(rotation=90)
plt.show()


#Identify the top five age groups in Australia dying with a Neoplasms cause of death.
#looking at graph, top age groups are from most to least, ages 80-84, ages 75-79, ages 70-74, and
ages 85-89.
```



In [21]:

```
find_population_and_deaths_neoplasms(population_data, mortality_data, australia_country_code)
#using pandasql, finding population and deaths neoplasms using population data, mortality data and australia country code as parameters
```

Out[21]:

	population	deaths
0	11100244.0	43276
1	11197271.0	43276

In [22]:

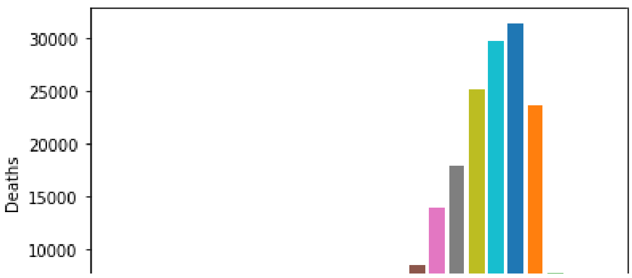
```
find_population_and_deaths_neoplasms(population_data, mortality_data, italy_country_code)
#using pandasql, finding population and deaths neoplasms using population data, mortality data and italy country code as parameters
```

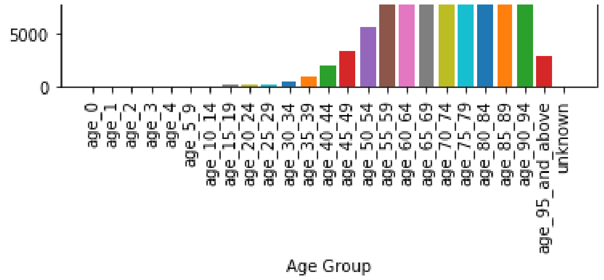
Out[22]:

	population	deaths
0	29350339.0	174405
1	31133047.0	174405

In [23]:

```
# Are there differences by age group for deaths from Neoplasms (C00-D48) in Italy for 2010
plot_data = sql('select sum(deaths2) as age_0, sum(deaths3) as age_1, sum(deaths4) as age_2, sum(deaths5) as age_3, sum(deaths6) as age_4, sum(deaths7) as age_5_9, sum(deaths8) as age_10_14, sum(deaths9) as age_15_19, sum(deaths10) as age_20_24, sum(deaths11) as age_25_29, sum(deaths12) as age_30_34, sum(deaths13) as age_35_39, sum(deaths14) as age_40_44, sum(deaths15) as age_45_49, sum(deaths16) as age_50_54, sum(deaths17) as age_55_59, sum(deaths18) as age_60_64, sum(deaths19) as age_65_69, sum(deaths20) as age_70_74, sum(deaths21) as age_75_79, sum(deaths22) as age_80_84, sum(deaths23) as age_85_89, sum(deaths24) as age_90_94, sum(deaths25) as age_95_and_above, sum(deaths26) as unknown from mortality_data where cause >= "C00" and cause <= "D48" and year=2010 and country = ' + str(italy_country_code))
for age_group in plot_data:
  plt.bar(age_group, plot_data[age_group])
plt.ylabel('Deaths')
plt.xlabel('Age Group')
plt.xticks(rotation=90)
plt.show()
#Identify the top five age groups in Italy dying with a Neoplasms cause of death.
#looking at graph, top age groups are from most to least, ages 80-84, ages 75-79, ages 70-74, and ages 85-89, so same as Australia.
```




5000
0
age_0
age_1
age_2
age_3
age_4
age_5_9
age_10_14
age_15_19
age_20_24
age_25_29
age_30_34
age_35_39
age_40_44
age_45_49
age_50_54
age_55_59
age_60_64
age_65_69
age_70_74
age_75_79
age_80_84
age_85_89
age_90_94
age_95_and_above
unknown
Age Group

In [ ]: