DMHR MODULE ASSIGNMENT

Assignment A:

The NHS has been challenged to make “efficiency savings” and you have been commissioned by an NHS executive to review, document and assess GP prescribing costs. Using data from the GP Practice Prescribing dataset (April 2018) address the following queries using a combination of narrative, tables, figures, and descriptive statistics:

Questions

1. Identify all GP practices located in London. For those practices, describe:

   - the total number of patients registered
   - the total number of prescriptions
   - the total actual cost of these prescriptions (using the ACT COST column)
   - the top 10 most frequent drugs prescribed
   - the bottom 10 less frequent drugs prescribed

2. Repeat the previous instructions, this time for the city of Cambridge. Discuss and compare your findings with the answers for London in question 1 above using descriptive statistics.
3. Describe total number of prescriptions and their total actual cost (using the ACT COST column) across all practices for drugs related to:

   - cardiovascular disease (British National Formulary chapter 2)
   - antidepressants (British National Formulary chapter 4.3)

4. Describe the total spending and the relative costs per patient across all practices for the month of April 2018:

   - visualize the monthly total spending per registered patients using a scatterplot and provide a trend line
   - generate a histogram for relative spending for all practices and fit a Gaussian (normal) curve

WHO Mortality Database

The WHO Mortality Database is a database of registered deaths compiled by WHO from data given by national authorities around the world. The cause of each death is classified by the circumstances that led to death. For this exercise, you will use data which report the cause of death using the 10th revision of the International Classification of Diseases (ICD-10). All of this information is collated into a number of Comma Separated Value (CSV) files, which can be found on the WHO Mortality Database website. The year of interest is 2010.
Each country in the database is uniquely identified all WHO datasets by a four digit numeric code. The mapping between countries and identifier codes is located in the "Country codes" lookup file. Information on the population of each country is found in the "Population and live births" file.

Questions

1. What was the population and the total number of deaths (from all causes, all ages) in 2010 for:

   - Iceland
   - Italy
   - New Zealand

2. What was the distribution of deaths (all causes, all years) by age group in Italy?

   - Visualise the results using a histogram.

3. What were the top five causes of death (top five ICD-10 terms) in Italy across all years for the Neoplasm ICD10-category (C00-D48)?

   - Generate a table with the cause of death, the number of deaths, and the proportion of overall deaths.
   - Generate a pie chart to visualize the proportion of deaths.

4. Are there differences by age group for deaths from Neoplasms (C00-D48) in Australia for 2010?

- Identify the top five age groups in Australia dying with a Neoplasms cause of death.

5. Compare and contrast the frequency of deaths by Neoplasms in Italy and Australia in 2010.

- Combine information on the population and deaths and describe your logic.
- Use descriptive statistics and plots.

Technical Report

Candidate number: BLBV5

Assignment A: NHS Data

1. Background and Introduction

The work for this assignment was carried out in order to make efficiency savings for prescribing in General Practices (GP). As part of the NHS five year forward view a number of measures are being put into place to ensure that there are enough facilities and resources to care for an aging population [1]. One of these measures includes plans to get the best value out of medicines and pharmacy: in 2017 the NHS was spending approximately £16 billion a year on drugs, of which about £9 billion arose from GP prescribing [2], hence reducing this number will allow resources to be reallocated.

2. Data used

The data used to generate figures and tables in this report are the GP Practice Prescribing (April/2018) and NHS Digital GP Practice Demographics (April/2018) which are both available from NHS digital [3, 4].

3. Results and discussion

3.1 London vs Cambridge

As London has the highest population of any city, we analysed the data related to London and used the city of Cambridge for comparison. Postcodes based on a publication by the ONS [5] were used to cross reference the postcodes given in the GP demographics and GP prescribing datasets. Postcodes starting with N, NE, SE, SW, E, EC, W and WC and CB1-CB5 were used as indicators of London and Cambridge city postcodes, respectively. Effective deselection measures were taken to ensure that the algorithm used to select for postcodes did not include non-London postcodes, for example Newcastle Upon Type which begins with NE and to ensure that the Cambridge city data did not contain CB postcodes after CB5. An important note to consider regarding the postcodes is that not all of the practices listed in the prescribing data had a related practice code and postcode in the GP demographic data file. This means that not all of the prescribing data was used in the analysis of the costs for each practice etc hence the figures may in fact underestimate the numbers of patients and costs.

Table 1 below shows the measures that were compared between the London and Cambridge.

TABLE 1. Comparison between London and Cambridge

Measure	London	Cambridge
Total number of practices	783	17
Number of patients registered	6026746	191931
Mean number of patients per practice +/- standard deviation	7697 +/- 5079	11290 +/- 4782
Number of prescribed items	5992400	160494
Cost of prescriptions	£44,142,367.58	£1,227,048.96
Cost per patient	£7.32	£6.39

Next, the 10 most and least frequently prescribed drugs were identified. This was done by selecting for BNF codes from the prescribing data. As this task is specifically related to drugs, BNF codes relating all chapters except 9, 18, 20, 21,22, 23 were used to selection of data. The chapters that were excluded relate to nutritional products which NICE have described as “borderline” substances [6] and appliances and dressings. In order to be able to easily identify the drugs and group them accordingly, the chemical and substances file which contains 9-digit BNF codes was used to cross reference the 15-digit BNF codes found in the prescribing file.
The top 10 most prescribed drugs for London and Cambridge are listed in Table 2 and Table4 below.

The data were relatively similar with most of the same drugs found on the top 10 list for both cities. Manual cross referencing of these drugs on the NICE website indicate that seven of the drugs listed in the top 10 most prescribed drugs are related to treatments for cardiac conditions. Other conditions listed related to diabetes, Heliobacter pylori infections and respiratory disease. The presence of these drugs also correlates to the leading causes of death (cardiac and respiratory) in both men and women from the 2017 data for deaths registered in England and Wales [7]. Figure1 shows the number of prescriptions for each of these drugs as a fraction of the total number of patients registered. From the graph, we can infer that for the 8 drugs that were common to both the London and Cambridge top 10 lists London still prescribes more of each drug per patient compared to Cambridge.

The least frequently prescribed drugs were also identified for London and Cambridge. Table 3 and Table 5 contains the information for the least prescribed drugs in London and Cambridge, respectively. There was a mixture of drugs from nearly every chapter for both London and Cambridge with medicines from BNF chapters 4 and 5 most frequently represented in London data (figure 2) and medicines from chapter 2 most highly represented in the Cambridge data (figure 3).

As the goal of this exercise is to make efficiency savings, the top 10 most expensive drugs for London and Cambridge were calculated and tabulated in table 6 and table 7. Table 8 shows the drugs that were common to both. The most expensive drug in both cases was Beclometasone Dipropionate which is a indicted as a prophylaxis for asthma but is also prescribed for hayfever, which is unsurprising for April. Other respiratory medicines such as Fluticasone Propionate and Budesonide were also on the top 10 most expensive list for Cambridge and London.

Metformin Hydrochloride, a treatment for type II diabetes, was found on both the most expensive drug's list and the most prescribed list for London. Moreover, diabetes-related product Glucose Blood Testing Reagents was also on the top 10 most expensive drugs list for both Cambridge and London. This could be one area where efficiency savings could be made as lifestyle modifications to diet and exercise resulting in weight loss can prevent and reverse the symptoms of Type II diabetes [8]. However, complete eradication of pharmacological intervention for Type II Diabetes is currently impractical as further study is required to determine the optimal lifestyle changes required [9].

3.2 Cardiac drugs and antidepressants

Heart disease was the second leading cause of death in the UK and death from suicide has been reported to have been increased for certain sexes and age groups [7]. The prescribing figures for medicines related to cardiovascular disease (CV) and antidepressants (AD) from GP practices across the UK are listed s follows: The total number of prescriptions and total cost for cardiovascular medicines amounted to 26,449,832 and £90,193,834.02, respectively. The total number of prescriptions and total cost for antidepressants medicines amounted to 5,715,873 and £16,853,470.86, respectively.
Given then aforementioned statistics on illnesses related to these medicines, costs such as these are likely to increase or be maintained. The cost per item prescribed for CV and AD medicines averages £11.45 and £12.33, respectively. While this is lower than the average prescription cost of £21.05, if the trend from 2017 continues, these costs will continue to grow also.

3.3 Total Spending and relative costs per patient

The total spending across all GP practices identifiable in both the prescribing data and the GP demographic data for April 2018 was £631,677,358.40. There was also an additional £11,510,031.80 identified in the practices that did not have a corresponding entry in the GP demographic file (see Appendix 1 in jupyter notebook for workings). Hence the total spend for prescribing from GP practices in the UK in April 2018 was £643,187,390.20.

The monthly spending per registered patient is displayed in figure 4. From the figure we can see a correlation between total spend and number of patients registered at a practice. The average spend per practice in this cohort was £87,842.77 with a standard deviation of £59133.29. The greatest total spend for a single practice was £842,838.17 which was for a practice of 66,502 registered patients in Birmingham. The histogram in figure 5 visualises the total monthly spend per patient for all practices. From the histogram, we can see that the average monthly spend per patient per practice is £11.

Assignment B: WHO data

1. Introduction and background

The World Health Organisation collate international mortality data based on clinical codes known as ICD10 codes which relates to a specific cause of death [10].

2. Data

The WHO data used to generate the information used on this section of the report is as follows:

- Country codes data to identify the countries.
- Mortality data
- Population data

Before addressing the points in the assignment, the data was processed to make it easier to work with. First, the two mortality datasets were combined into one file. I created dictionaries of the age groups and replaced the columns in the mortality and population datasets. Also, as there would be multiple commands which required the selection of data for all of the age groups I specified a variable related to these.

3. Results and discussions

3.1 Population and total number of deaths for Iceland, Italy and New Zealand in 2010

The information for this section was generated by finding the country code in the country code file and then using it to select for the country code and year from the population and mortality datasets. The outputs generated are as follows:

Iceland:

- Population 318041
- Deaths 4038

Italy:

- Population 60483386
- Deaths 1169230

New Zealand:

- Population 4367360
- Deaths 57298

3.2 The distribution of deaths by age group in Italy

The data for this section was generated by selecting all of the data related to Italy from the mortality data. Figure 6 shows a bar graph was created for each age group which represents the proportion of the total number of deaths that are represented by each age group. From the histogram we can see that 80-84 years old and 85-89 year old represent the highest proportion of the total deaths in Italy indicating that most people die aged 80-89.

3.3 Top 5 causes of death in Italy for neoplasms

The top 5 causes of deaths related to Neoplasms in Italy were calculated by isolating the data for ICD codes C00-D48 and grouping the data according to Neoplasm type and calculating the total deaths and the percentage that each type of neoplasm account for. Table 9 displays the top 5 causes of death for neoplasms and the percentage of the total number of neoplasm deaths that they represent. The percentage that each subcategory of Neoplasm accounts for is depicted in the pie chart in Figure 7. The ICD10 codes relate to:

```
C349    Malignant neoplasm: Bronchus or lung, unspecified
C509    Malignant neoplasm: Breast, unspecified
C189    Malignant neoplasm: Colon, unspecified
C169    Malignant neoplasm: Stomach, unspecified
C259    Malignant neoplasm: Pancreas, unspecified
```

The fact that each of the top 5 neoplasms are “unspecified” highlights one of the key issues that researchers face when using ICD10 codes for analysis as the information is somewhat limited.

3.4. Differences in age groups for Neoplasm deaths in Australia

To investigate whether in 2010 there were differences between age groups for the deaths that are due to neoplasms in Australia the data for this was isolated from the data for all deaths. Figure 8 shows a bar graph of this data. From this, we can observe that the highest number of deaths due to neoplasms occurred in the 80-84 year age bracket.

1. 80-84
2. 75-79
3. 70-74
4. 85-89
5. 65-69

This follows trends that have been reported for other countries such as the UK and the US [11, 12]

3.5 Differences in neoplasm deaths between Italy and Australia for 2010

The frequency of deaths or death rate per 100,000 of the population in both Italy and Australia was calculated by dividing the total number of deaths caused by neoplasms by the total number of deaths from all causes in each age group. In Italy, the death rate due to neoplasms per 100,000 of the population was 289.41 while in Australia it was 194.26 In figure 9, we can see that the total number of deaths due to neoplasms was much greater in Italy compared to Australia. Even when the deaths caused by neoplasms are expressed relative to the population for each country (figure 10), neoplasms account for a higher percentage of all deaths in Italy, particularly for age groups less 55 years of age. This was surprising given that recent figures have shown that cancer rates in Australia are the highest in the world [13]. Previous work has also shown that the relative survival rates for cancer are higher in Australia than Italy [14]. Hence, the higher mortality rate in Italy could be due to more aggressive types of cancer being present in Italian populations or better standards of care in the Australian system.

REFERENCES

1. NHS. NHS: Five year forward view. 2016; Available from: https://www.england.nhs.uk/five-year-forward-view/next-steps-on-the-nhs-five-year-forward-view/executive-summary/ (https://www.england.nhs.uk/five-year-forward-view/next-steps-on-the-nhs-five-year-forward-view/executive-summary/).
2. NHS. NHS: five-year-forward-view funding-and-efficiency. 2016; Available from: https://www.england.nhs.uk/five-year-forward-view/next-steps-on-the-nhs-five-year-forward-view/funding-and-efficiency/ (https://www.england.nhs.uk/five-year-forward-view/next-steps-on-the-nhs-five-year-forward-view/funding-and-efficiency/).
3. NHS. Practice Level Prescribing - presentation level data - April 2018. 2018; Available from: https://digital.nhs.uk/data-and-information/publications/statistical/practice-level-prescribing-data/april-2018 (https://digital.nhs.uk/data-and-information/publications/statistical/practice-level-prescribing-data/april-2018).
4. NHS. Patients Registered at a GP Practice, April 2018; Special Topic - Registered patients compared to the projected resident population in England. 2018; Available from: https://digital.nhs.uk/data-and-information/publications/statistical/patients-registered-at-a-gp-practice/patients-registered-at-a-gp-practice-april-2018-special-topic---registered-patients-compared-to-the-projected-resident-population-in-england (https://digital.nhs.uk/data-and-information/publications/statistical/patients-registered-at-a-gp-practice/patients-registered-at-a-gp-practice-april-2018-special-topic---registered-patients-compared-to-the-projected-resident-population-in-england).
5. ONS. ONS Postcode Directory (May 2018) User Guide. 2018; Available from: https://data.gov.uk/dataset/224cbd39-ba59-4356-a24f-2a142d57375d/ons-postcode-directory-may-2018-user-guide (https://data.gov.uk/dataset/224cbd39-ba59-4356-a24f-2a142d57375d/ons-postcode-directory-may-2018-user-guide).
6. NICE. How to use BNF Publications online. 2019; Available from: https://bnf.nice.org.uk/about/how-to-use-bnf-publications-online.html (https://bnf.nice.org.uk/about/how-to-use-bnf-publications-online.html).
7. ONS. Home People, population and community Births, deaths and marriages Deaths Deaths registered in England and Wales (series DR) Deaths registered in England and Wales (series DR): 2017. 2018; Available from: https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/bulletins/deathsregisteredinengland
(https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/bulletins/deathsregisteredinenglanc
8. Guess, N.D., Dietary Interventions for the Prevention of Type 2 Diabetes in High-Risk Groups: Current State of Evidence and Future Research Needs. Nutrients, 2018. 10(9).
9. Ried-Larsen, M., et al., Why prescribe exercise as therapy in type 2 diabetes? We have a pill for that! Diabetes Metab Res Rev, 2018. 34(5): p. e2999.
10. Cimino, J.J., Review paper: coding systems in health care. Methods Inf Med, 1996. 35(4-5): p. 273-84.
11. UK, C.R. Cancer Mortality by age. 2017; Available from: https://www.cancerresearchuk.org/health-professional/cancer-statistics/mortality/age (https://www.cancerresearchuk.org/health-professional/cancer-statistics/mortality/age).
12. White, M.C., et al., Age and cancer risk: a potentially modifiable relationship. Am J Prev Med, 2014. 46(3 Suppl 1): p. S7-15.
13. Fund, W.C.R. Global cancer data by country. 2018; Available from: https://www.wcrf.org/dietandcancer/cancer-trends/data-cancer-frequency-country (https://www.wcrf.org/dietandcancer/cancer-trends/data-cancer-frequency-country).
14. Crocetti, E., et al., Cancer prevalence in United States, Nordic Countries, Italy, Australia, and France: an analysis of geographic variability. Br J Cancer, 2013. 109(1): p. 219-28.

Before starting, make sure everything you need is installed on your computer

First ensure that you have the latest pip by upgrading it and then install all of the libraries you will need. It is assumed that python is already installed. Alternatively you can download **Anaconda (https://www.anaconda.com)** which has these libraries already installed.

Once the libraries have been installed you can import them.

In [20]:

```
! pip install --upgrade pip
! pip install pandas
! pip install pandasql
! pip install numpy
! pip install scipy
! pip install scikit-learn
! pip install matplotlib
! pip install blackcellmagic

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pandasql import PandaSQL
pdsql = PandaSQL()
pd.set_option("display.max_colwidth", -1) #
%load_ext blackcellmagic
```

```
Requirement already up-to-date: pip in ./anaconda3/lib/python3.7/site-packages (18.1)
Requirement already satisfied: pandas in ./anaconda3/lib/python3.7/site-packages (0.23.4)
Requirement already satisfied: python-dateutil>=2.5.0 in ./anaconda3/lib/python3.7/site-package
s (from pandas) (2.7.3)
Requirement already satisfied: pytz>=2011k in ./anaconda3/lib/python3.7/site-packages (from pan
das) (2018.5)
```

```
Requirement already satisfied: numpy>=1.9.0 in ./anaconda3/lib/python3.7/site-packages (from pa
ndas) (1.15.1)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.7/site-packages (from python
-dateutil>=2.5.0->pandas) (1.11.0)
Requirement already satisfied: pandasql in ./anaconda3/lib/python3.7/site-packages (0.7.3)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.7/site-packages (from pandasql)
 (1.15.1)
Requirement already satisfied: pandas in ./anaconda3/lib/python3.7/site-packages (from pandasql
) (0.23.4)
Requirement already satisfied: sqlalchemy in ./anaconda3/lib/python3.7/site-packages (from pand
asql) (1.2.11)
Requirement already satisfied: python-dateutil>=2.5.0 in ./anaconda3/lib/python3.7/site-package
s (from pandas->pandasql) (2.7.3)
Requirement already satisfied: pytz>=2011k in ./anaconda3/lib/python3.7/site-packages (from pan
das->pandasql) (2018.5)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.7/site-packages (from python
-dateutil>=2.5.0->pandas->pandasql) (1.11.0)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.7/site-packages (1.15.1)
Requirement already satisfied: scipy in ./anaconda3/lib/python3.7/site-packages (1.1.0)
Requirement already satisfied: scikit-learn in ./anaconda3/lib/python3.7/site-packages (0.19.2)
Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.7/site-packages (2.2.3)
Requirement already satisfied: numpy>=1.7.1 in ./anaconda3/lib/python3.7/site-packages (from ma
tplotlib) (1.15.1)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.7/site-packages (from ma
tplotlib) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in ./anaconda3/lib/pyth
on3.7/site-packages (from matplotlib) (2.2.0)
Requirement already satisfied: python-dateutil>=2.1 in ./anaconda3/lib/python3.7/site-packages
(from matplotlib) (2.7.3)
Requirement already satisfied: pytz in ./anaconda3/lib/python3.7/site-packages (from matplotlib
) (2018.5)
Requirement already satisfied: six>=1.10 in ./anaconda3/lib/python3.7/site-packages (from matpl
otlib) (1.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.7/site-packages (fr
om matplotlib) (1.0.1)
Requirement already satisfied: setuptools in ./anaconda3/lib/python3.7/site-packages (from kiwi
solver>=1.0.1->matplotlib) (40.2.0)
Requirement already satisfied: blackcellmagic in ./anaconda3/lib/python3.7/site-packages (0.0.1
)
Requirement already satisfied: ipython in ./anaconda3/lib/python3.7/site-packages (from blackce
llmagic) (6.5.0)
Requirement already satisfied: black in ./anaconda3/lib/python3.7/site-packages (from blackcell
magic) (18.9b0)
Requirement already satisfied: simplegeneric>0.8 in ./anaconda3/lib/python3.7/site-packages (fr
om ipython->blackcellmagic) (0.8.1)
Requirement already satisfied: pexpect; sys_platform != "win32" in ./anaconda3/lib/python3.7/si
te-packages (from ipython->blackcellmagic) (4.6.0)
Requirement already satisfied: traitlets>=4.2 in ./anaconda3/lib/python3.7/site-packages (from
ipython->blackcellmagic) (4.3.2)
Requirement already satisfied: setuptools>=18.5 in ./anaconda3/lib/python3.7/site-packages (fro
m ipython->blackcellmagic) (40.2.0)
Requirement already satisfied: pickleshare in ./anaconda3/lib/python3.7/site-packages (from ipy
thon->blackcellmagic) (0.7.4)
Requirement already satisfied: decorator in ./anaconda3/lib/python3.7/site-packages (from ipyth
on->blackcellmagic) (4.3.0)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.15 in ./anaconda3/lib/python3.7/site-
packages (from ipython->blackcellmagic) (1.0.15)
Requirement already satisfied: jedi>=0.10 in ./anaconda3/lib/python3.7/site-packages (from ipyt
hon->blackcellmagic) (0.12.1)
Requirement already satisfied: pygments in ./anaconda3/lib/python3.7/site-packages (from ipytho
n->blackcellmagic) (2.2.0)
Requirement already satisfied: backcall in ./anaconda3/lib/python3.7/site-packages (from ipytho
n->blackcellmagic) (0.1.0)
Requirement already satisfied: appnope; sys_platform == "darwin" in ./anaconda3/lib/python3.7/s
ite-packages (from ipython->blackcellmagic) (0.1.0)
Requirement already satisfied: toml>=0.9.4 in ./anaconda3/lib/python3.7/site-packages (from bla
ck->blackcellmagic) (0.10.0)
Requirement already satisfied: attrs>=17.4.0 in ./anaconda3/lib/python3.7/site-packages (from b
lack->blackcellmagic) (18.2.0)
Requirement already satisfied: appdirs in ./anaconda3/lib/python3.7/site-packages (from black->
blackcellmagic) (1.4.3)
Requirement already satisfied: click>=6.5 in ./anaconda3/lib/python3.7/site-packages (from blac
k->blackcellmagic) (6.7)
Requirement already satisfied: ptyprocess>=0.5 in ./anaconda3/lib/python3.7/site-packages (from
 pexpect; sys_platform != "win32"->ipython->blackcellmagic) (0.6.0)
Requirement already satisfied: ipython-genutils in ./anaconda3/lib/python3.7/site-packages (fro
m traitlets>=4.2->ipython->blackcellmagic) (0.2.0)
Requirement already satisfied: six in ./anaconda3/lib/python3.7/site-packages (from traitlets>=
4.2->ipython->blackcellmagic) (1.11.0)
Requirement already satisfied: wcwidth in ./anaconda3/lib/python3.7/site-packages (from prompt-
```

```
toolkit<2.0.0,>=1.0.15->ipython->blackcellmagic) (0.1.7)
Requirement already satisfied: parso>=0.3.0 in ./anaconda3/lib/python3.7/site-packages (from je
di>=0.10->ipython->blackcellmagic) (0.3.1)
The blackcellmagic extension is already loaded. To reload it, use:
  %reload_ext blackcellmagic
```

QUESTION 1

- Identify the total number of patients registered

Step 1. Extract the data for the GP practices in London from the UK-wide file

First create a pandas dataframe (df) of all of the gp dempgraphics for the UK and check the output.

In [21]:

```
# Create df.
gp_prac = pd.read_csv("https://files.digital.nhs.uk/71/B59D99/gp-reg-pat-prac-all.csv", header=0)
```

In [22]:

```
# Check df - formatting
gp_prac.head(5)  # check formatting by showing first 10 rows
```

Out[22]:

	PUBLICATION	EXTRACT_DATE	TYPE	CCG_CODE	ONS_CCG_CODE	CODE	POSTCODE	SEX	AGE	NUMBER_O
0	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83005	DL1 3RT	ALL	ALL	
1	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83006	DL3 6HZ	ALL	ALL	
2	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83010	DL3 9JP	ALL	ALL	
3	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83013	DL1 4YL	ALL	ALL	
4	GP_PRAC_PAT_LIST	01APR2018	GP	00C	E38000042	A83031	DL3 8SQ	ALL	ALL	

In [23]:

```
# Check df - shape (info on number of rows and columns)
gp_prac.shape
```

Out[23]:

```
(7241, 10)
```

In [24]:

```
# Check df - check for duplicated - should have same number of rows as original
gp_prac_duplicated = gp_prac.duplicated(subset="CODE", keep="first")
gp_prac_duplicated.shape
```

Out[24]:

```
(7241,)
```

Step 2. Remove any white space that may be present around the headings or data within the dataframe

In [25]:

```
# remove any white space around the headings
gp_prac_trimmed = gp_prac.apply(lambda x: x.str.strip() if x.dtype == "object" else x)

# remove white space from left and right of all data within columns
for col in gp_prac_trimmed.keys():
    if type(gp_prac_trimmed[col].iloc[0]) == str:
        gp_prac_trimmed[col] = gp_prac_trimmed[col].str.strip()
```

Step 3. Extract data for London practices

From gp_prac_trimmed create a new dataframe with just the data for London using postcode **[information (https://data.gov.uk/dataset/224cbd39-ba59-4356-a24f-2a142d57375d/ons-postcode-directory-may-2018-user-guide)](https://data.gov.uk/dataset/224cbd39-ba59-4356-a24f-2a142d57375d/ons-postcode-directory-may-2018-user-guide)** that is available to download from the Office of National Statistics (ONS).
From ONS information, represented in the table below, we can see that London postcodes begin with: N, NW, SE, SW, E, EC, W and WC. However there are also other postcodes that begin with N, E and W so these need to be accounted for.

Beginning with	London Postcodes	Other postcodes
N	London N, London NW	Newcastle upon Tyne NE, Nottingham NG, Northampton NN, Newport NP, Norwich NR
S	London SW London SE	n/a
E	London E, London EC	Edinburgh EH, Enfield EN, Exeter EX
W	London W, London WC	Warrington WA, Watford WD, Wakefield WF, Wigan WN, Worcester WR, Walsall WS, Wolverhampton WV

In [26]:

```
# Create a new df with all of the postcodes, including the other non-London postcodes that begin with N, E a
nd W.
gp_prac_londonlike = gp_prac_trimmed[gp_prac_trimmed.POSTCODE.str.startswith(('N', 'SE', 'SW', 'E', 'W'))]
```

In [27]:

```
# Check df - first 5 rows
gp_prac_londonlike.head(5)
```

Out[27]:

	PUBLICATION	EXTRACT_DATE	TYPE	CCG_CODE	ONS_CCG_CODE	CODE	POSTCODE	SEX	AGE	NUMBER
64	GP_PRAC_PAT_LIST	01APR2018	GP	00J	E38000116	A83038	NE16 6HU	ALL	ALL	
72	GP_PRAC_PAT_LIST	01APR2018	GP	00J	E38000116	A83618	NE17 7SB	ALL	ALL	
114	GP_PRAC_PAT_LIST	01APR2018	GP	00L	E38000130	A84002	NE65 7UW	ALL	ALL	
115	GP_PRAC_PAT_LIST	01APR2018	GP	00L	E38000130	A84003	NE63 9UT	ALL	ALL	
116	GP_PRAC_PAT_LIST	01APR2018	GP	00L	E38000130	A84005	NE22 6JX	ALL	ALL	

In [28]:

```
# Check df - shape
gp_prac_londonlike.shape
```

Out[28]:

```
(1804, 10)
```

In [29]:

```
# Create df with london-only postcodes
gp_prac_londontrue = gp_prac_londonlike[~gp_prac_londonlike.POSTCODE.str.startswith
                                                              (('NE','NG', 'NN', 'NP', 'NR', 'EH', 'EN', 'EX
', 'WA',
                                                                'WD','WF', 'WN', 'WR', 'WS', 'WV'))]
```

In [30]:

```
# Check df - shape
gp_prac_londontrue.shape
```

Out[30]:

```
(783, 10)
```

Step 4. Use the dataframe created in step 1 to answer the bullet points for question 1

Calculate the total numbers of patients registered:

In [31]:

```
gp_prac_londontrue['NUMBER_OF_PATIENTS'].sum()
```

Out[31]:

```
6026746
```

Question 1

- Identify the total number of prescriptions
- Identify the total actual cost of these prescriptions (using the ACT COST column)

Step 1. First identify the Prescribing data that relates to London GP practices

In [32]:

```
# Create df of prescribing data and remove any white spaces in the data

prescribing = pd.read_csv(
    "https://files.digital.nhs.uk/38/03EC1C/T201804PDPI%20BNFT.CSV",
    header=0,
)

# remove the white spaces in headers
prescribing = prescribing.rename(columns=lambda x: x.strip())

# remove white space from left and right of all data within columns
for col in prescribing.keys():
    if type(prescribing[col].iloc[0]) == str:
        prescribing[col] = prescribing[col].str.strip()
```

In [33]:

```
# Select only London GP prescribing data
prescribing_london = prescribing[prescribing.PRACTICE.isin(gp_prac_londontrue.CODE)]
prescribing_london.shape  # check shape of london-only prescribing data
```

Out[33]:

```
(814140, 11)
```

Step 2: Total number of prescriptions is found by summing the "ITEMS" column

In [34]:

```
# total number of prescriptions ofr London
prescribing_london['ITEMS'].sum()
```

Out[34]:

```
5992400
```

Step 3: Actual cost of these prescriptions

In [35]:

```
# total cost
prescribing_london['ACT COST'].sum()
```

Out[35]:

```
44142367.580000006
```

Question 1

- Identify the top 10 most frequent drugs prescribed
- Identify the bottom 10 less frequent drugs prescribed

Step 1. Select rows that only contain BNF codes relating to drugs (ie: not nutritional products / dressings / appliances) in chapters 1-15 of the BNF.

In [36]:

```
# Create df
drugs_only = prescribing_london[
    prescribing_london["BNF CODE"].str.startswith(
        ("01", "02", "03", "04", "05", "06", "07", "08", "10", "11", "12", "13", "14", "15")
    )
]

# sort data by "BNF code" and show the last 5 rows to make sure that the last values do not begin > "15..."
drugs_only.sort_values("BNF CODE").tail(10)
```

Out[36]:

	SHA	PCT	PRACTICE	BNF CODE	BNF NAME	ITEMS	NIC	ACT COST	QUANTITY	PERIOD
7305069	Q63	08Q	G85050	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	2	184.62	171.24	90	201804
7307303	Q63	08Q	G85052	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	3	143.59	133.29	70	201804
7308189	Q63	08Q	G85082	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	2	123.08	114.17	60	201804
7309177	Q63	08Q	G85084	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	7	492.32	456.66	240	201804
7240288	Q63	08L	G85076	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	1	123.08	114.16	60	201804
6396294	Q61	08M	F84070	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	3	184.62	171.25	90	201804
7424640	Q63	08X	H85075	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	1	61.54	57.08	30	201804
7299660	Q63	08Q	G85031	1502010J0BWAAEL	Ralvo_Medic Plastr 700mg	5	369.24	342.50	180	201804
6124401	Q61	07R	F83017	1502010P0BCACA0	Scandonest_Plain Inj 3% 2.2ml Cart	3	1.32	1.23	3	201804
7439151	Q63	08X	Y01132	1502010V0AAAAAA	Levobupivac HCl_Inj 2.5mg/ml 10ml Amp	1	14.11	13.10	10	201804

Step 2. Add drug name information column to this df by merging with chemical substances info available on NHS digital website

In [37]:

```python
# Make a new data frame of bnf chemical substances
bnf_chem_subs = pd.read_csv('https://files.digital.nhs.uk/79/6D58A8/T201804CHEM%20SUBS.CSV', header=0)


# Remove any white space around column names and string columns
bnf_chem_subs = bnf_chem_subs.rename(columns=lambda x: x.strip())

for col in bnf_chem_subs.keys():
   if type(bnf_chem_subs[col].iloc[0])==str:
     bnf_chem_subs[col] = bnf_chem_subs[col].str.strip()

# Make a new column in the df that lists the first 9 digits of the BNF code
drugs_only['BNF_CODE_9']=drugs_only['BNF CODE'].str[:9]

# Check new columns added correctly
drugs_only.head(5)
```

```
/Users/clairemooney/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:13: SettingWith
CopyWarning:
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-docs/stable/indexing.html
#indexing-view-versus-copy
  del sys.path[0]
```

Out[37]:

	SHA	PCT	PRACTICE	BNF CODE	BNF NAME	ITEMS	NIC	ACT COST	QUANTITY	PERIOD	BI
6050139	Q61	07M	E83003	0101010G0BCABAB	Mucogel_Susp 195mg/220mg/5ml S/F	1	2.99	2.79	500	201804	
6050140	Q61	07M	E83003	0101010L0BEAAAI	Maalox Plus_Susp	1	2.91	2.71	250	201804	
6050141	Q61	07M	E83003	0101021B0AAALAL	Sod Algin/Pot Bicarb_Susp S/F	12	66.56	61.88	6500	201804	
6050142	Q61	07M	E83003	0101021B0AAAPAP	Sod Alginate/Pot Bicarb_Tab Chble 500mg	1	0.61	0.68	12	201804	
6050143	Q61	07M	E83003	0101021B0BEADAJ	Gaviscon Infant_Sach 2g (Dual Pack) S/F	5	67.48	62.64	210	201804	

In [38]:

```python
# Merge files so that data frame contains drug name
BNF9 = drugs_only.merge(bnf_chem_subs, how='inner', left_on=['BNF_CODE_9'], right_on=['CHEM SUB'])
```

In [39]:

```python
# Check new df
BNF9.shape
```

Out[39]:

```
(661579, 16)
```

In [40]:

```
# Check new df
BNF9.head(5)
```

Out[40]:

	SHA	PCT	PRACTICE	BNF CODE	BNF NAME	ITEMS	NIC	ACT COST	QUANTITY	PERIOD	_x	BNF_COI
0	Q61	07M	E83003	0101010G0BCABAB	Mucogel_Susp 195mg/220mg/5ml S/F	1	2.99	2.79	500	201804		010101
1	Q61	07M	E83006	0101010G0BCABAB	Mucogel_Susp 195mg/220mg/5ml S/F	1	2.99	2.79	500	201804		010101
2	Q61	07M	E83011	0101010G0BCABAB	Mucogel_Susp 195mg/220mg/5ml S/F	1	2.99	2.79	500	201804		010101
3	Q61	07M	E83021	0101010G0BCABAB	Mucogel_Susp 195mg/220mg/5ml S/F	1	2.99	2.79	500	201804		010101
4	Q61	07M	E83035	0101010G0BCABAB	Mucogel_Susp 195mg/220mg/5ml S/F	1	2.99	2.79	500	201804		010101

Step 3. Group the data according to drug name and count the number of items (indicted by the "ITEMS" column) in each group ie: for each drug name

In [41]:

```
# group the merged data by drug name and sum the number of items in each group
BNF9_grouped = BNF9.groupby(['NAME'])
```

In [42]:

```
# check how many groups were created
len(BNF9_grouped)
```

Out[42]:

1067

In [43]:

```
# sum number of prescriptions in the "ITEMS" in each group, rename the "sum" colum "TOTAL_ITEMS" and sort th
e values
counts = BNF9_grouped["ITEMS"].sum().reset_index(name="TOTAL_ITEMS").sort_values(["TOTAL_ITEMS"], ascending=
False)
```

Step 4. Identify the top 10 most frequently prescribed drugs

Table 2 Top 10 most prescribed drugs London

In [44]:

```
counts.nlargest(10, ['TOTAL_ITEMS'])
```

Out[44]:

	NAME	TOTAL_ITEMS
77	Atorvastatin	262472
54	Amlodipine	205497
627	Metformin Hydrochloride	174110
560	Levothyroxine Sodium	149900
870	Ramipril	149128
725	Omeprazole	143935
543	Lansoprazole	134012
72	Aspirin	128978
915	Simvastatin	116354
118	Bisoprolol Fumarate	115176

In [45]:

```
# check that the number of items for Atorvastatin = 262,472
BNF9[BNF9.NAME == "Atorvastatin"].ITEMS.sum()
```

Out[45]:

262472

Step 5. Identify the top 10 least frequently prescribed drugs

From the sorted "counts" dataframe in step 3 we can see that there are more than 10 drugs that were prescribed only once. First we need to find out exactly how many drug are only prescribed once. We can then investigate further to see what chapters they come from as there may be a trend towards least prescibing from different chapters.

In [46]:

```
# create new df containing only drugs prescibed once and then get the number of rows
least = counts[counts['TOTAL_ITEMS'] == 1]
least.shape
```

Out[46]:

(54, 2)

In [47]:

```
# to get all of the information related to the least presecribed drugs merge least df with BNF9 df
least_alldata = least.merge(BNF9, how='inner', on=['NAME'])
```

Table 3

In [48]:

```
# table 3
least_alldata.sort_values(['BNF_CODE_9'], ascending=True)
```

Out[48]:

	NAME	TOTAL_ITEMS	SHA	PCT	PRACTICE	BNF CODE	
15	Macrogol 4000	1	Q61	08D	Y03035	0106040X0AAAAAA	Macrogol 4000_
33	Timolol With Diuretic	1	Q61	08W	F86666	020400030AAAEAE	Timolol/Bendroflumeth_
50	Phenindione	1	Q61	08M	F84730	0208020N0AAAAAA	Phenir
0	Alirocumab	1	Q63	08X	H85114	0212000AIAAABAB	Alirocumab_Inj 15
7	Other Expectorant& Demulcent Cough Preps	1	Q62	08Y	E87722	030902000BEDSA0	Buttercup_Bronch
4	Dextromethorphan Hydrobrom Comp Prep's	1	Q61	07T	F84080	0309020R0AAABAB	Diphenhyd/Dextrometh_Ora

43	Caffeine Citrate	1	Q61	08V	F84079	0404000E0AAAMAM	Caffeine Cit_Lic
13	Nabilone	1	Q61	08M	F84735	0406000R0AAAAAA	Na
20	Ketamine Hydrochloride	1	Q62	07W	E85116	0406000W0AAAPAP	Ketamine_Ora
36	Aspirin & Caffeine	1	Q62	07W	E85628	0407010AABCAAAB	Anadin Orig_
42	Paracetamol & Ibuprofen	1	Q61	08D	F85034	0407010ADAAAAAA	Paracet/Ibuprofen_Ta
49	Phenobarbital Sod	1	Q63	08A	G83026	0408010P0AAAVAV	Phenobarb Sod_Lic
37	Benzatropine Mesilate	1	Q61	08W	F86650	0409020E0AAABAB	Benzatropir
14	Meropenem	1	Q61	08D	F85640	0501022A0AAAAAA	Meropenem_Ir
29	Amikacin	1	Q61	08D	F85014	0501040C0AAADAD	Amikacin_Inj Paec
45	Pyrazinamide	1	Q61	08W	F86666	0501090N0AACJCJ	Pyrazina
41	Rifabutin	1	Q61	07R	F83018	0501090Q0AAABAB	Rifab
6	Other HIV Infection Preps	1	Q61	07T	F84072	050301000BBAEA0	Genvoya_Tab 150mg/150r
44	Raltegravir	1	Q63	08K	G85021	0503010AEAAAAAA	Ralteg
19	Efavirenz/Emtricitabine/Tenofovir Disop	1	Q61	07R	F83050	0503010ANAAAAAA	Efavirenz/Emtricitabi
11	Nevirapine	1	Q61	07X	F85002	0503010B0AAAAAA	Nevira
53	Abacavir & Lamivudine	1	Q61	07X	F85002	0503010Z0AAAAAA	Abacavir/Lamivudine_Ta
48	Chloroquine Phosphate with Proguanil HCl	1	Q63	08Q	G85712	0504010Z0AAAAAA	Chloroquine/Proguani
52	Paromomycin Sulfate	1	Q61	07R	F83023	0504050T0AAADAD	Paromomycin Sulf_
24	Insulin Human	1	Q61	08H	F83027	0601011R0BEAAAE	Humulin R KwikPen_Inj 50
23	Insulin Zinc Suspension	1	Q63	08L	G85089	0601012G0BCAAAH	Ins Hypurin Bov Lente_
46	Protamine Zinc Insulin	1	Q63	08R	H85051	0601012U0AAAAAA	Ins Prot Zn_(Bov)
51	Chorionic Gonadotrophin	1	Q63	08Q	G85042	0605010D0AAACAC	Chorion Gonadotroph_Inj 1
34	Tetracosactide	1	Q62	08Y	E87043	0605010T0BBAAAA	Synacthen_Inj (
35	Teriparatide	1	Q62	09A	E87754	0606010U0BBABAC	Forsteo_Inj 250mcg
10	Norethisterone Enantate	1	Q63	08Q	G85034	0703022N0BBAAAA	Noristerat_Inj 20(
16	Other Urological Pain Preps	1	Q61	08M	F84047	070403000AAAEAE	
1	Yohimbine Hydrochloride	1	Q61	08D	F85640	0704050Y0AAAVAV	Yohimb
22	Estramustine Phosphate	1	Q63	08L	G85055	0801010J0AAAAAA	Estramustine I
32	Tioguanine	1	Q63	08R	H85028	0801030T0AAAAAA	Tiogu
39	Bexarotene	1	Q61	08H	F83027	0801050ABAAAAAA	Bexar
17	Dimethyl Fumar	1	Q61	07X	F85015	0802040AKAAABAB	Dimethyl Fumar
2	Fulvestrant	1	Q61	08W	F86078	0803041AABBAAAA	Faslodex_Ir
30	Buserelin	1	Q62	09A	E87745	0803042B0AAABAB	Buserelin_Nsl Spy 10
38	Benzbromarone	1	Q61	08M	F84017	1001040A0AAAAAA	Benzbroma
26	Glucosamine Sulf (Rheumatic)	1	Q61	08W	F86627	1001050B0BCABAB	
12	Neostigmine Bromide	1	Q62	08Y	E87738	1002010M0AAAAAA	Neostig
28	Oxybuprocaine Hydrochloride	1	Q61	08D	F85007	1107000M0AAAAAA	Oxybuprocaine HCl_
5	Other Oral Ulceration&Inflammation Preps	1	Q61	07T	F84686	120301000BCBCAM	Frado
3	Tretinoin	1	Q62	08C	Y02589	1306010V0AAANAN	Tret
25	Formaldehyde	1	Q61	08D	F85675	1307000C0BBAAAB	Verac
31	Titanium Dioxide	1	Q63	08X	H85069	1308010U0AAAJAJ	Reflectant_Sunsc
40	Bifonazole	1	Q62	09A	E87045	1310020E0AAAAAA	Bif
47	Chlorhex HCl	1	Q63	08A	G83039	1310050W0AAAAAA	Chlorh
8	Other Antiperspirant Preps	1	Q61	08V	F84034	131200000BBACA0	Triple Dry Anti-p
27	Other Topical Circulatory Preps	1	Q63	07V	Y05317	131400000BLAAA0	Boo
9	Normal Immunoglobulin (Gamma Globulin)	1	Q62	07W	E85026	1405010A0BQAAAE	Subgam_
21	Ketamine	1	Q61	08W	F86638	1501010F0AAAAAA	Ketamine_Inj
18	Levobupivacaine Hydrochloride	1	Q63	08X	Y01132	1502010V0AAAAAA	Levobupivac HCl_Inj 2.5r

Question 2: Repeat the previous instructions, this time for the city of Cambridge.

Question 2

- Identify the total number of patients registered

Step 1. Identify Cambridge GP practice data

From previously created gp_prac file, isolate the Cambridge GP practices.
Cambridge city postcodes are CB1, CB2, CB3, CB4, CB5. There are also other postcodes in Cambridge from CB6-25 which encompases the surrounding area so these need to be accounted for.

In [49]:

```
# extract rows that contain postcodes starting with CB1-5
gp_prac_cb = gp_prac[gp_prac.POSTCODE.str.startswith(("CB1", "CB2", "CB3", "CB4", "CB5"))]

# split the POSTCODE column by the white space
gp_prac_cb_split = gp_prac_cb["POSTCODE"].str.split(" ", n=1, expand=True)

# add columns of split postcodes to the dataframe
gp_prac_cb["POSTCODE_1"] = gp_prac_cb_split[0]
gp_prac_cb["POSTCODE_2"] = gp_prac_cb_split[1]

gp_prac_cb
gp_prac_cbcity = gp_prac_cb[gp_prac_cb.POSTCODE_1.isin(("CB1", "CB2", "CB3", "CB4", "CB5"))]
```

```
/Users/clairemooney/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingWithC
opyWarning:
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-docs/stable/indexing.html
#indexing-view-versus-copy

/Users/clairemooney/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: SettingWithC
opyWarning:
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-docs/stable/indexing.html
#indexing-view-versus-copy
  if __name__ == '__main__':
```

In [50]:

```
# Check df - rows will be the number of practices
gp_prac_cbcity.shape
```

Out[50]:

```
(17, 12)
```

In [51]:

```
### Step 2. Calculate the total number of patients
```

In [52]:

```
gp_prac_cbcity['NUMBER_OF_PATIENTS'].sum()
```

Out[52]:

```
191931
```

Question 2

- Identify the total number of prescriptions
- Identify the total actual cost of these prescriptions (using the ACT COST column)

Step 1. First extract the prescribing data that relates to Cambridge city GP practices and clean the data

In [54]:

```
# Create df of prescribing data and remove any white spaces in the data
prescribing_cb = pd.read_csv(
    'https://files.digital.nhs.uk/38/03EC1C/T201804PDPI%20BNFT.CSV',
    header=0,
)

# remove the white spaces in headers
prescribing_cb = prescribing_cb.rename(columns=lambda x: x.strip())

# remove white space from left and right of all data within columns
for col in prescribing.keys():
    if type(prescribing_cb[col].iloc[0]) == str:
        prescribing_cb[col] = prescribing_cb[col].str.strip()


# Select only Cambridge city GP prescribing data
prescribing_cb = prescribing_cb[prescribing_cb.PRACTICE.isin(gp_prac_cbcity.CODE)]
prescribing_cb.shape  # check shape of london-only prescribing data
```

Out[54]:

```
(21360, 11)
```

Step 2: Total number of prescriptions is found by summing the "ITEMS" column

In [55]:

```
prescribing_cb['ITEMS'].sum()
```

Out[55]:

```
160494
```

Step 3: Actual cost of these prescriptions

In [56]:

```
# total cost
prescribing_cb['ACT COST'].sum()
```

Out[56]:

```
1227048.9600000002
```

Question 2

- Identify the top 10 most frequent drugs prescribed
- Identify the bottom 10 less frequent drugs prescribed

Step 1. Carry out same process as for London in Question 1

In [57]:

```
# Create df
cb_drugs_only = prescribing_cb[
    prescribing_cb["BNF CODE"].str.startswith(
        ("01", "02", "03", "04", "05", "06", "07", "08", "10", "11", "12", "13", "14", "15")
    )
]

# sort data by "BNF code" and show the last 5 rows to make sure that the last values do not begin > "15..."
cb_drugs_only.sort_values("BNF CODE").tail(10)


# Make a new column in the df that lists the first 9 digits of the BNF code
cb_drugs_only['BNF_CODE_9']=cb_drugs_only['BNF CODE'].str[:9]

# Merge files to create file with drug name and BNF_CODE_9 column
cb_BNF9 = cb_drugs_only.merge(bnf_chem_subs, how='inner', left_on=['BNF_CODE_9'], right_on=['CHEM SUB'])

# group the merged data by drug name and sum the number of items in each group
cb_BNF9_grouped = cb_BNF9.groupby(['NAME'])

# sum number of prescriptions in the "ITEMS" in each group, rename the "sum" column "TOTAL_ITEMS" and sort t
he values
cb_counts = cb_BNF9_grouped["ITEMS"].sum().reset_index(name="TOTAL_ITEMS").sort_values(["TOTAL_ITEMS"], asce
nding=False)
```

```
/Users/clairemooney/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:13: SettingWith
CopyWarning:
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-docs/stable/indexing.html
#indexing-view-versus-copy
  del sys.path[0]
```

In [58]:

```
len(cb_BNF9_grouped)
```

Out[58]:

758

Step 2. Identify the top 10 most frequently prescribed drugs

Table 4

In [59]:

```
cb_counts.nlargest(10, ['TOTAL_ITEMS'])
```

Out[59]:

	NAME	TOTAL_ITEMS
51	Atorvastatin	5658
524	Omeprazole	5569
401	Levothyroxine Sodium	4855
36	Amlodipine	3763
48	Aspirin	3329
648	Simvastatin	2987
80	Bisoprolol Fumarate	2977
411	Lisinopril	2935
634	Salbutamol	2860
451	Metformin Hydrochloride	2784

In [60]:

```python
# check that the number of items for Atorvastatin = 5658
cb_BNF9[cb_BNF9.NAME == "Atorvastatin"].ITEMS.sum()
```

Out[60]:

5658

Step 3. Identify the 10 least frequently presecribed drugs

Table 5

In [61]:

```python
# create new df containing only drugs prescibed once and then get the number of rows
cb_least = cb_counts[cb_counts['TOTAL_ITEMS'] == 1]

# see the list of least prescribed drugs
cb_least
```

Out[61]:

	NAME	TOTAL_ITEMS
695	Tiagabine	1
86	Brivaracetam	1
52	Atovaquone	1
60	Balsalazide Sodium	1
739	Vancomycin Hydrochloride	1
2	Acenocoumarol	1
718	Trimetazidine Hydrochloride	1
702	Tinidazole	1
20	Aliskiren	1
726	Ulipristal Acet	1
723	Tropicamide	1
30	Amiloride HCl With Loop Diuretics	1
748	Xipamide	1
738	Valsartan/Amlodipine	1
26	Aluminium & Magnesium & Act Simeticone	1
728	Umeclidinium Brom	1
427	Lubiprostone	1
105	Carbomer	1
536	Oxazepam	1
518	Oils For The Ear	1
296	Fondaparinux Sodium	1
522	Olsalazine Sodium	1
288	Flurbiprofen	1
529	Other Bisphosphonate & Other Preps	1
533	Other Non-Opioid Analgesic Preps	1
538	Oxprenolol Hydrochloride	1
322	Heparin Flushes	1
281	Fluocortolone	1
258	Ethinylestradiol	1
550	Penicillamine	1
...	...	...
646	Simeticone	1
649	Simvastatin & Ezetimibe	1
652	Sodium Aurothiomalate	1

132	Cilostazol	1
128	Choline Salicylate	1
570	Pilocarpine Nitrate	1
669	Sucralfate	1
123	Chloroxylenol	1
118	Chlohexidine Gluconate (Emollient)	1
676	Tafluprost & Timolol	1
117	Cetrimide	1
112	Cefradine	1
152	Cloral Betaine	1
163	Co-Proxamol (Dextroprop HCl/Paracet)	1
632	Rutosides	1
189	Degarelix	1
613	Ramipril with Calcium Channel Blocker	1
201	Diazoxide	1
604	Pseudoephedrine Hydrochloride	1
205	Diethylamine Salicylate	1
210	Diphenhydramine Hydrochloride	1
213	Disopyramide	1
227	Dronabinol/Cannabidiol	1
589	Prednisolone Sodium Metasulphobenzoate	1
585	Prasugrel	1
583	Povidone Iodine	1
582	Potassium Permanganate	1
573	Piracetam	1
241	Entacapone	1
224	Doxepin Hydrochloride	1

87 rows × 2 columns

In [62]:

```
# to get all of the information related to the least presecribed drugs merge least df with BNF9 df
cb_least_alldata = cb_least.merge(cb_BNF9, how='inner', on=['NAME'])

# sort the data by BNF code in ascending order to get an idea of which chapters are represented in the least
 group
cb_least_alldata.sort_values(['BNF_CODE_9'], ascending=True)
```

Out[62]:

	NAME	TOTAL_ITEMS	SHA	PCT	PRACTICE	BNF CODE	BNF NAME	ITEMS
14	Aluminium & Magnesium & Act Simeticone	1	Q56	06H	D81037	0101010L0BEAAAI	Maalox Plus_Susp	1
57	Simeticone	1	Q56	06H	D81013	0101010R0BCAAAB	Infacol_Susp 40mg/ml S/F	1
63	Sucralfate	1	Q56	06H	D81016	0103030S0AAAAAA	Sucralfate_Tab 1g	1
38	Loperamide Hydrochloride & Simeticone	1	Q56	06H	D81001	0104020P0AAAAAA	Loperamide HCl/Simeticone_Tab 2mg/125mg	1
21	Olsalazine Sodium	1	Q56	06H	D81037	0105010C0AAADAD	Olsalazine Sod_Tab 500mg	1
3	Balsalazide Sodium	1	Q56	06H	D81066	0105010D0AAABAB	Balsalazide Sod_Cap 750mg	1
80	Prednisolone Sodium Metasulphobenzoate	1	Q56	06H	D81070	0105020D0AAACAC	Prednisolone_20mg/Applic Foam Enema(14D)	1
16	Lubiprostone	1	Q56	06H	D81001	0106070C0AAAAAA	Lubiprostone_Cap 24mcg	1
27	Fluocortolone	1	Q56	06H	D81066	0107020F0BBAAAA	Ultraproct_Oint	1
12	Xipamide	1	Q56	06H	D81012	0202010Y0AAAAAA	Xipamide_Tab 20mg	1
11	Amiloride HCl With Loop Diuretics	1	Q56	06H	D81017	0202040D0AAAAAA	Amiloride HCl/Bumetanide_Tab	1

							5mg/1mg	
78	Disopyramide	1	Q56	06H	D81012	0203020F0AAABAB	Disopyramide_Cap 100mg	1
49	Nadolol	1	Q56	06H	D81037	0204000M0AAABAB	Nadolol_Tab 80mg	1
25	Oxprenolol Hydrochloride	1	Q56	06H	D81002	0204000N0AAACAC	Oxprenolol HCl_Tab 40mg	1
55	Sildenafil(Vasodilator Antihypertensive)	1	Q56	06H	D81012	0205010Y0AAABAB	Sildenafil_Susp 10mg/1ml S/F	1
73	Ramipril with Calcium Channel Blocker	1	Q56	06H	D81017	0205051S0AAABAB	Felodipine/Ramipril_Tab 5mg/5mg M/R	1
32	Perindopril Arginine with Diuretic	1	Q56	06H	D81002	0205051Z0AAAAAA	Perindopril Argin/Indapam_Tab 5mg/1.25mg	1
53	Eprosartan	1	Q56	06H	D81002	0205052W0AAACAC	Eprosartan_Tab 600mg	1
8	Aliskiren	1	Q56	06H	D81012	0205053A0AAABAB	Aliskiren_Tab 300mg	1
6	Trimetazidine Hydrochloride	1	Q56	06H	D81017	0206020B0AAABAB	Trimetazidine HCl_Tab 35mg M/R	1
13	Valsartan/Amlodipine	1	Q56	06H	D81013	0206020Z0AAACAC	Amlodipine/Valsartan_Tab 10mg/160mg	1
71	Rutosides	1	Q56	06H	D81054	0206040AHAAAAAA	Oxerutins_Cap 250mg	1
60	Cilostazol	1	Q56	06H	D81056	0206040X0AAAAAA	Cilostazol_Tab 100mg	1
20	Fondaparinux Sodium	1	Q56	06H	D81003	0208010ABAAABAB	Fondaparinux Sod_Inj 12.5mg/ml 0.4ml Pfs	1
26	Heparin Flushes	1	Q56	06H	D81066	0208010P0AAADAD	Heparin Sod_Soln 10u/ml 5ml Amp	1
5	Acenocoumarol	1	Q56	06H	D81025	0208020H0AAAAAA	Acenocoumarol_Tab 1mg	1
81	Prasugrel	1	Q56	06H	D81016	0209000Y0AAAAAA	Prasugrel_Tab 5mg	1
58	Simvastatin & Ezetimibe	1	Q56	06H	D81066	0212000ACAAABAB	Simvastatin/Ezetimibe_Tab 40mg/10mg	1
15	Umeclidinium Brom	1	Q56	06H	D81025	0301020T0BBAAAA	Incruse Ellipta_Inh 55mcg (30D)	1
54	Ciclesonide	1	Q56	06H	D81086	0302000U0AAACAC	Ciclesonide_Inh 160mcg (60 D) CFF	1
...	...	...	...	...	...	...	...	...
41	Mesterolone	1	Q56	06H	D81002	0604020F0AAAAAA	Mesterolone_Tab 25mg	1
23	Other Bisphosphonate & Other Preps	1	Q56	06H	D81002	060602000BBAAA0	Actonel Combi_Tab 35mg/Gran Eff 1g/800u	1
51	Estradiol & Nomegestrol	1	Q56	06H	D81012	0703010S0BBAAAA	Zoely_Tab 2.5mg/1.5mg	1
72	Degarelix	1	Q56	06H	D81025	0803042R0AAAAAA	Degarelix_Inj 80mg Vl + Dil	1
22	Flurbiprofen	1	Q56	06H	D81066	1001010I0AAABAB	Flurbiprofen_Tab 50mg	1
29	Penicillamine	1	Q56	06H	D81013	1001030F0AAAFAF	Penicillamine_Tab 250mg	1
59	Sodium Aurothiomalate	1	Q56	06H	D81086	1001030J0AAAEAE	Sod Aurothiomalate_Inj 100mg/ml .5ml Amp	1
79	Dronabinol/Cannabidiol	1	Q56	06H	D81013	1002020Y0BBABAB	Sativex_Oromucosal P/Spy 10ml (90D)	1
76	Diethylamine Salicylate	1	Q56	06H	D81066	1003020I0AAAAAA	Diethylamine Sal_Crm 10% BP	1
37	Loteprednol Etabonate	1	Q56	06H	D81017	1104010W0BBAAAA	Lotemax_Eye Dps 0.5%	1
10	Tropicamide	1	Q56	06H	D81025	1105000S0AAABAB	Tropicamide_Eye Dps 1%	1
66	Tafluprost & Timolol	1	Q56	06H	D81012	1106000AMBBAAAA	Taptiqom_Eye Dps 15mcg/5mg/ml 0.3ml Ud	1
42	Levobunolol Hydrochloride	1	Q56	06H	D81070	1106000T0AAAAAA	Levobunolol HCl_Eye Dps 0.5%	1
62	Pilocarpine Nitrate	1	Q56	06H	D81086	1106000Y0AAABAB	Piloc Nit_Eye Dps 2% Ud	1
82	Povidone Iodine	1	Q56	06H	D81013	1108020AHAAABAB	Povidone-Iodine_Eye Dps 5% P/F 0.4ml Ud	1
48	Moxifloxacin Hydrochloride	1	Q56	06H	D81070	1108020U0AAABAB	Moxifloxacin HCl_Eye Dps 0.5%	1
17	Carbomer	1	Q56	06H	D81012	1108030AAAAABAB	Carbomer_Eye Gel 0.36% P/F	1
19	Oils For The Ear	1	Q56	06H	D81054	1201030H0BBAAAA	Cerumol_Ear Dps	1
61	Choline Salicylate	1	Q56	06H	D81044	1203010K0AAABAB	Choline Sal_Dental Gel 8.7% S/F	1

65	Chlohexidine Gluconate (Emollient)	1	Q56	06H	D81013	1302010Z0AAAAAA	Chlorhex Glucon_Emollient/Crm 1%	1
86	Doxepin Hydrochloride	1	Q56	06H	D81086	1303000F0BCAAAA	Xepin_Crm 5%	1
40	Mepyramine Maleate	1	Q56	06H	D81054	1303000M0BBAAAA	Anthisan_Crm 2%	1
34	Nicotinamide	1	Q56	06H	D81056	1306010N0AAAAAA	Nicotinamide_Gel 4%	1
56	Silver Nitrate	1	Q56	06H	D81003	1307000Q0AAAAAA	Silver Nit Caustic_Pencil 95% BP 1980	1
47	Ingenol Mebutate	1	Q56	06H	D81037	1308010Z0AAAAAA	Ingenol Mebutate_Gel 150mcg/g	1
67	Cetrimide	1	Q56	06H	D81044	1310050D0BIAAAD	Savlon_Antis Crm	1
64	Chloroxylenol	1	Q56	06H	D81070	1311050E0BBABAC	Dettol_Liq	1
83	Potassium Permanganate	1	Q56	06H	Y00056	1311060Q0AAACAC	Pot Permanganate_Cutaneous Soln Tab400mg	1
44	Human Papillomavirus (Type 6 11 16 18)	1	Q56	06H	D81002	1404000AHAAAAAA	HPV (Type 6 11 16 18)_Vac 0.5ml Pfs	1
39	Meningococcal A + C + W135 + Y Vaccine	1	Q56	06H	D81056	1404000X0BJAAAG	Nimenrix_Vac 0.5ml Dil + Pfs	1

87 rows × 17 columns

QUESTION 2: Descriptive Statistics of London vs Cambridge

In [63]:

```
%precision 2
# mean number of patients per pratice in London
gp_prac_londontrue['NUMBER_OF_PATIENTS'].mean()
```

Out[63]:

7696.99

In [64]:

```
# std dev of the number of patients per pratice in London
gp_prac_londontrue['NUMBER_OF_PATIENTS'].std()
```

Out[64]:

5079.25

In [65]:

```
# mean number of patients per pratice in Cambridge
gp_prac_cbcity['NUMBER_OF_PATIENTS'].mean()
```

Out[65]:

11290.06

In [66]:

```
# std dev of the number of patients per pratice in Cambridge
gp_prac_cbcity['NUMBER_OF_PATIENTS'].std()
```

Out[66]:

4782.13

Figure1

In [67]:

```
# Figure 1: bar graph of top 10 most prescribed drugs relative to number of patients
top10_london = counts.nlargest(10, ['TOTAL_ITEMS'])
top10_london['LONDON_PER_PATIENT'] = top10_london['TOTAL_ITEMS']/gp_prac_londontrue['NUMBER_OF_PATIENTS'].su
m()

top10_cambridge = cb_counts.nlargest(10, ['TOTAL_ITEMS'])
top10_cambridge['CAMBRIDGE_PER_PATIENT'] = top10_cambridge['TOTAL_ITEMS']/gp_prac_cb['NUMBER_OF_PATIENTS'].s
um()

top10_merge = top10_london.merge(top10_cambridge, on='NAME')
top10_merge.set_index('NAME')[['LONDON_PER_PATIENT', 'CAMBRIDGE_PER_PATIENT']].plot.bar()
```

Out[67]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a434eadd8>
```

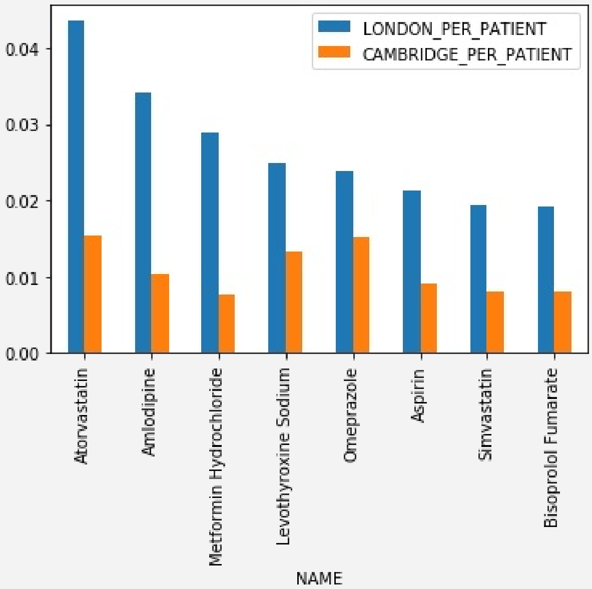


Figure2

In [68]:

```
# Figure 2: pie chart of the chapters related to the least prescribed drugs for Cambridge

# create a column of with the least data set with the first two digits of BNf and group and plot them
least_alldata['BNF2'] = least_alldata['BNF CODE'].str[:2]
least_alldata_chapter_counts = least_alldata.groupby('BNF2').agg({'BNF2': 'count'})['BNF2']
ax = plt.subplot(111)
wedges, texts = ax.pie(least_alldata_chapter_counts.values, labels=least_alldata_chapter_counts.keys())

for w in wedges:
    w.set_linewidth(1)
    w.set_edgecolor('black')

plt.title('Chapters for least prescribed drugs for London')
```

Out[68]:

```
Text(0.5,1,'Chapters for least prescribed drugs for London')
```

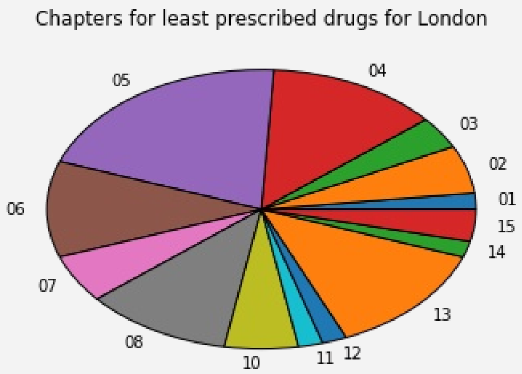


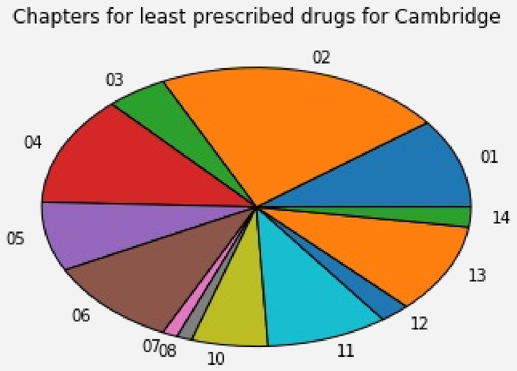
Figure3

In [69]:

```
# Figure 3: Pie chart of chapters related to least prescribed drugs in Cambridge

#As above but for Cambridge data
cb_least_alldata['BNF2'] = cb_least_alldata['BNF CODE'].str[:2]
cb_least_chapter_counts = cb_least_alldata.groupby('BNF2').agg({'BNF2': 'count'})['BNF2']
ax = plt.subplot(111)
wedges, texts = ax.pie(cb_least_chapter_counts.values, labels=cb_least_chapter_counts.keys())

for w in wedges:
    w.set_linewidth(1)
    w.set_edgecolor('black')

plt.title('Chapters for least prescribed drugs for Cambridge')
```

Out[69]:

```
Text(0.5,1,'Chapters for least prescribed drugs for Cambridge')
```



In [70]:

```
# Cost per patient for London
prescribing_london['ACT COST'].sum()/gp_prac_londontrue['NUMBER_OF_PATIENTS'].sum()
```

Out[70]:

```
7.324411478432973
```

In [71]:

```
# Cost per patient for Cambridge
prescribing_cb['ACT COST'].sum()/gp_prac_cbcity['NUMBER_OF_PATIENTS'].sum()
```

Out[71]:

```
6.393177548181379
```

Table 6

In [72]:

```
# London's top 10 most expensive drugs Table 6
london_costs = BNF9_grouped["ACT COST"].sum().reset_index(name="TOTAL_COST").sort_values(["TOTAL_COST"], asc
ending=False)
london_top10_costs = london_costs.head(10)
london_top10_costs
```

Out[72]:

	NAME	TOTAL_COST
417	Fluticasone Propionate (Inh)	1127957.66
918	Sitagliptin	1085861.87
447	Glucose Blood Testing Reagents	954292.56
89	Beclometasone Dipropionate	898492.91
884	Rivaroxaban	788403.64
129	Budesonide	734729.18
991	Tiotropium	711898.31
63	Apixaban	707637.17
627	Metformin Hydrochloride	688632.97
625	Mesalazine (Systemic)	521519.27

Table 7

In [73]:

```
# Cambridge's top 10 most expensive drugs Table 7
cb_costs = cb_BNF9_grouped["ACT COST"].sum().reset_index(name="TOTAL_COST").sort_values(["TOTAL_COST"], asce
nding=False)
cb_top10_costs = cb_costs.head(10)
cb_top10_costs
```

Out[73]:

	NAME	TOTAL_COST
62	Beclometasone Dipropionate	45224.61
625	Rivaroxaban	36966.49
313	Glucose Blood Testing Reagents	30535.43
292	Fluticasone Propionate (Inh)	26933.44
704	Tiotropium	22832.36
43	Apixaban	21438.81
87	Budesonide	18764.81
356	Insulin Aspart	17329.73
606	Quetiapine	14894.52
663	Somatropin	12431.13

Table 8

In [74]:

```
# 7 drugs are in common between the top Table 8
shared_top10_drugs = pd.merge(cb_top10_costs, london_top10_costs, on='NAME')
shared_top10_drugs.columns = ['NAME', 'CAMB_COST', 'LONDON_COST']
shared_top10_drugs
```

Out[74]:

	NAME	CAMB_COST	LONDON_COST
0	Beclometasone Dipropionate	45224.61	898492.91
1	Rivaroxaban	36966.49	788403.64
2	Glucose Blood Testing Reagents	30535.43	954292.56
3	Fluticasone Propionate (Inh)	26933.44	1127957.66
4	Tiotropium	22832.36	711898.31
5	Apixaban	21438.81	707637.17
6	Budesonide	18764.81	734729.18

In [75]:

```
# London has 3 drugs in it's top 10 not in Cambridge's top 10. Sitagliptin is expensive
london_top10_costs[~(london_top10_costs.NAME.isin(cb_top10_costs.NAME))]
```

Out[75]:

	NAME	TOTAL_COST
918	Sitagliptin	1085861.87
627	Metformin Hydrochloride	688632.97
625	Mesalazine (Systemic)	521519.27

Question 3

1. Describe total number of prescriptions and their total actual cost (using the ACT COST column) across all practices for drugs related to:

    > - cardiovascular disease (British National Formulary chapter 2)
    > - antidepressants (British National Formulary chapter 4.3)

In [76]:

```
# First select the presecriptions from the relevant chapter

cv_prescrip = prescribing[prescribing['BNF CODE'].str.startswith('02')]
ad_prescrip = prescribing[prescribing['BNF CODE'].str.startswith('0403')]
```

In [77]:

```
# Duplicate the prescribing df and add a column which contains the first 2 letter of the BNF code
# Grouping the data by the BNF2 column allows us to check that there is only one group
cv_prescrip['BNF2'] = cv_prescrip['BNF CODE'].str[:2]
cv_prescrip_grouped = cv_prescrip.groupby(['BNF2'])
len(cv_prescrip_grouped)
```

```
/Users/clairemooney/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: SettingWithC
opyWarning:
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-docs/stable/indexing.html
#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[77]:

```
1
```

In [78]:

```
#Duplicate the prescribing df and add a column which contains the first 4 letters of the BNF code
# Grouping the data by the BNF4 column allows us to check that there is only one group
ad_prescrip['BNF4'] = ad_prescrip['BNF CODE'].str[:4]
ad_prescrip_grouped = ad_prescrip.groupby(['BNF4'])
len(cv_prescrip_grouped)
```

```
/Users/clairemooney/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: SettingWithC
opyWarning:
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-docs/stable/indexing.html
#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[78]:

1

In [79]:

```
# calcuate the number of prescriptions for cardiovascular medicines
cv_prescrip.ITEMS.sum()
```

Out[79]:

26449832

In [80]:

```
# calculate the cost of these prescriptions
cv_prescrip['ACT COST'].sum()
```

Out[80]:

90193834.01999994

In [81]:

```
# calculate the number of prescriptons related to Antidepressants
ad_prescrip.ITEMS.sum()
```

Out[81]:

5715873

In [82]:

```
# calculate the cost of antidepressants
ad_prescrip['ACT COST'].sum()
```

Out[82]:

16853470.86

In [83]:

```
# calculate the average presecription cost for a given treatment item
avg_medicine_cost = (prescribing['ACT COST'] / prescribing['ITEMS']).mean()
avg_medicine_cost
```

Out[83]:

21.05

In [84]:

```
# calculate the average cost of prescriptions related to cv medicines
avg_cv_prescript = (cv_prescrip['ACT COST'] / cv_prescrip['ITEMS']).mean()
avg_cv_prescript
```

Out[84]:

11.45

In [85]:

```
# calculate the average cost of prescriptions related to ad medicines
avg_ad_prescript = (ad_prescrip['ACT COST'] / ad_prescrip['ITEMS']).mean()
avg_ad_prescript
```

Out[85]:

12.33

Question 4

1. Describe the total spending and the relative costs per patient across all practices for the month of April 2018:

> - visualize the monthly total spending per registered patients using a scatterplot and provide a trend line
> - generate a histogram for relative spending for all practices and fit a Gaussian (normal) curve

Step 1. Combine prescribing data with GP prac demographic data to get the number of patients within the same df

In [86]:

```
# create new df of combining prescrribing and gp prac demographics
prescribing_pat_num = prescribing.merge(gp_prac, how='inner', left_on=['PRACTICE'], right_on=['CODE'])

# check the new dataframe that is created
prescribing_pat_num.head()
```

Out[86]:

	SHA	PCT	PRACTICE	BNF CODE	BNF NAME	ITEMS	NIC	ACT COST	QUANTITY	PERIOD	...	PUBLICAT
0	Q44	RXA	Y04664	0101021B0BEAIAL	Gaviscon Advance_Liq (Aniseed) (Reckitt)	9	61.44	57.09	6000	201804	...	GP_PRAC_PAT_
1	Q44	RXA	Y04664	0101021B0BEAQAP	Gaviscon Advance_Tab Chble Mint(Reckitt)	2	15.35	14.26	300	201804	...	GP_PRAC_PAT_
2	Q44	RXA	Y04664	0101021B0BEBEAL	Gaviscon Advance_Liq (Peppermint) S/F	7	33.98	31.60	3050	201804	...	GP_PRAC_PAT_
3	Q44	RXA	Y04664	0102000A0AAAAAA	Alverine Cit_Cap 60mg	4	4.00	4.16	84	201804	...	GP_PRAC_PAT_
4	Q44	RXA	Y04664	0102000A0BBABAB	Spasmonal Fte_Cap 120mg	1	29.13	27.13	90	201804	...	GP_PRAC_PAT_

5 rows × 21 columns

Step . Get total spend per practice

Group the merged df by CODE and get the total within each group

In [87]:

```
# group the new df by CODE column
prescribing_grouped = prescribing_pat_num.groupby(['CODE'])
len(prescribing_grouped)

# See Appendix 1 for information on practices omitted during the selection process
```

Out[87]:

7191

In [88]:

```
# sum number of prescriptions in the "ITEMS" in each group, rename the "sum" column "TOTAL_ITEMS" and sort t
he values
practice_spend = prescribing_grouped['ACT COST'].sum().reset_index(name='TOTAL SPEND')

# check output
practice_spend.head(5)
```

Out[88]:

	CODE	TOTAL SPEND
0	A81001	52194.63
1	A81002	268607.26
2	A81004	139115.40
3	A81005	102914.06
4	A81006	183226.79

Step 3. Get total spend per patient for each practice

Merge the total spend per practice df with the gp_prac data and calculate necessary outputs

In [89]:

```
# merge the practice spend file with the gp_prac file in order to calculate the spend per patient
spend_vs_numPat = practice_spend.merge(gp_prac, how='inner', on=['CODE'])

#check output
spend_vs_numPat.head(5)
```

Out[89]:

	CODE	TOTAL SPEND	PUBLICATION	EXTRACT_DATE	TYPE	CCG_CODE	ONS_CCG_CODE	POSTCODE	SEX	AGE
0	A81001	52194.63	GP_PRAC_PAT_LIST	01APR2018	GP	00K	E38000075	TS18 1HU	ALL	ALL
1	A81002	268607.26	GP_PRAC_PAT_LIST	01APR2018	GP	00K	E38000075	TS18 2AW	ALL	ALL
2	A81004	139115.40	GP_PRAC_PAT_LIST	01APR2018	GP	00M	E38000162	TS5 8SB	ALL	ALL
3	A81005	102914.06	GP_PRAC_PAT_LIST	01APR2018	GP	00M	E38000162	TS14 7DJ	ALL	ALL
4	A81006	183226.79	GP_PRAC_PAT_LIST	01APR2018	GP	00K	E38000075	TS18 2AT	ALL	ALL

In [90]:

```
# mean total spend per practice
spend_vs_numPat['TOTAL SPEND'].mean()
```

Out[90]:

87842.77

In [91]:

```
# maximum spend
spend_vs_numPat['TOTAL SPEND'].max()
```

Out[91]:

842838.1799999974

In [92]:

```
## minimum spend
spend_vs_numPat['TOTAL SPEND'].min()
```

Out[92]:

3.25

In [93]:

```
spend_vs_numPat['TOTAL SPEND'].std()
```

Out[93]:

```
59133.29
```

In [94]:

```
spend_vs_numPat['TOTAL SPEND'].sum()
```

Out[94]:

```
631677358.4000001
```

In [95]:

```
spend_vs_numPat['NUMBER_OF_PATIENTS'].max()
```

Out[95]:

```
72227
```

In [96]:

```
spend_vs_numPat['NUMBER_OF_PATIENTS'].min()
```

Out[96]:

```
3
```

In [97]:

```
# to find out where the location of practice with the highest spend
spend_vs_numPat[spend_vs_numPat['TOTAL SPEND'] == 842838.1799999974]
```

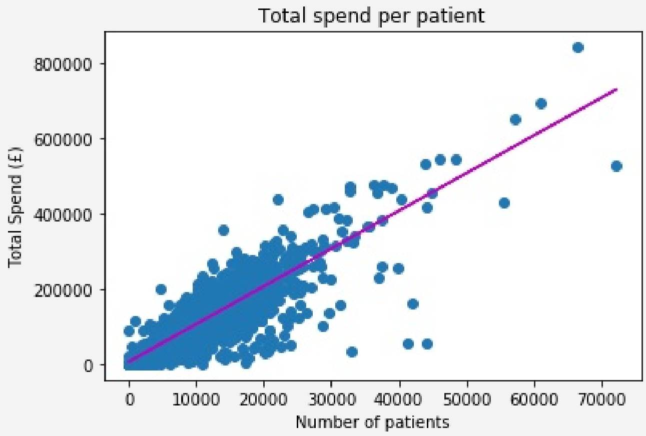
Out[97]:

	CODE	TOTAL SPEND	PUBLICATION	EXTRACT_DATE	TYPE	CCG_CODE	ONS_CCG_CODE	POSTCODE	SEX	A
5491	M85063	842838.18	GP_PRAC_PAT_LIST	01APR2018	GP	15E	E38000220	B24 0SY	ALL	A

Figure 4

In [98]:

```python
# Create scatter plot. FIGURE 4
plt.scatter(spend_vs_numPat['NUMBER_OF_PATIENTS'], spend_vs_numPat['TOTAL SPEND'])

# Add trendline
x = spend_vs_numPat['NUMBER_OF_PATIENTS'].values
y = spend_vs_numPat['TOTAL SPEND'].values

z = np.polyfit(x, y, 1)
p = np.poly1d(z)

plt.plot(x, p(x), 'm-')

plt.xlabel('Number of patients')
plt.ylabel('Total Spend (£)')
plt.title('Total spend per patient')
plt.figure(figsize=(30,20))
plt.show()
```




```
<Figure size 2160x1440 with 0 Axes>
```

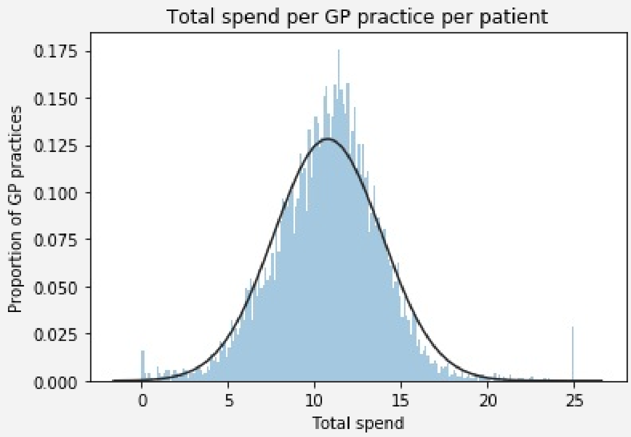
Figure 5

In [99]:

```python
# Create histogram Figure 5

from scipy.stats import norm

x = spend_vs_numPat['TOTAL SPEND']/spend_vs_numPat['NUMBER_OF_PATIENTS']

plt.xlabel('Total Spend per patient')
plt.ylabel('Proportion of GP practices')
plt.title('Total spend per GP practice per patient')

sns.distplot(x.clip(0,25), bins=200, fit=norm, kde=False, axlabel='Total spend', norm_hist=True)
```

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1f083780>
```



ASSIGNMENT B

WHO Mortality Database

The WHO Mortality Database is a database of registered deaths compiled by WHO from data given by national authorities around the world. The cause of each death is classified by the circumstances that led to death. For this exercise, you will use data which report the cause of death using the 10th revision of the International Classification of Diseases (ICD-10). All of this information is collated into a number of Comma Separated Value (CSV) files, which can be found on the WHO Mortality Database website. The year of interest is 2010.
Each country in the database is uniquely identified all WHO datasets by a four digit numeric code. The mapping between countries and identifier codes is located in the "Country codes" lookup file. Information on the population of each country is found in the "Population and live births" file.

In [100]:

```
import urllib
import zipfile
urls = ['https://www.who.int/healthinfo/statistics/Morticd10_part1.zip', 'https://www.who.int/healthinfo/sta
tistics/Morticd10_part2.zip',
        'https://www.who.int/healthinfo/statistics/country_codes.zip', 'https://www.who.int/healthinfo/Pop.z
ip']
for url in urls:
  urllib.request.urlretrieve(url, url.split('/')[-1])

for filepath in ['country_codes.zip', 'Morticd10_part1.zip', 'Morticd10_part2.zip', 'Pop.zip']:
  zip_ref = zipfile.ZipFile(filepath, 'r')
  zip_ref.extractall()
  zip_ref.close()

mortality1 = pd.read_csv('Morticd10_part1')
mortality2 = pd.read_csv('Morticd10_part2')
country_codes = pd.read_csv('country_codes')
pop = pd.read_csv('pop')
```

```
/Users/clairemooney/anaconda3/lib/python3.7/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)
/Users/clairemooney/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py:2785
: DtypeWarning: Columns (2,4) have mixed types. Specify dtype option on import or set low_memor
y=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Question 1.

What was the population and the total number of deaths (from all causes, all ages) in 2010 for:

- Iceland
- Italy
- New Zealand

Step 1. Set up data

In [101]:

```
# combine mortality data files make files headers more more readable by changing the headers

all_deaths = pd.concat((mortality1, mortality2))

age_dict = {'Deaths1':'Total', 'Deaths2':'0', 'Deaths3':'1', 'Deaths4':'2', 'Deaths5':'3', 'Deaths6':'4',
            'Deaths7':'5-9', 'Deaths8':'10-14', 'Deaths9':'15-19', 'Deaths10':'20-24', 'Deaths11':'25-29',
            'Deaths12':'30-34', 'Deaths13':'35-39', 'Deaths14':'40-44', 'Deaths15':'45-49', 'Deaths16':'50-54',
            'Deaths17':'55-59', 'Deaths18':'60-64', 'Deaths19':'65-69', 'Deaths20':'70-74', 'Deaths21':'75-79',
            'Deaths22':'80-84', 'Deaths23':'85-89', 'Deaths24':'90-94', 'Deaths25':'95+', 'Deaths26':'Unspecified'}

pop_dict = {'Pop1':'Total', 'Pop2':'0', 'Pop3':'1', 'Pop4':'2', 'Pop5':'3', 'Pop6':'4',
            'Pop7':'5-9', 'Pop8':'10-14', 'Pop9':'15-19', 'Pop10':'20-24', 'Pop11':'25-29',
            'Pop12':'30-34', 'Pop13':'35-39', 'Pop14':'40-44', 'Pop15':'45-49', 'Pop16':'50-54',
            'Pop17':'55-59', 'Pop18':'60-64', 'Pop19':'65-69', 'Pop20':'70-74', 'Pop21':'75-79',
            'Pop22':'80-84', 'Pop23':'85-89', 'Pop24':'90-94', 'Pop25':'95+', 'Pop26':'Unspecified'}

specified_age_groups = ['0-4','5-9','10-14','15-19','20-24','25-29','30-34','35-39','40-44','45-49',
                        '50-54','55-59','60-64','65-69','70-74','75-79','80-84','85-89','90-94','95+']
```

In [102]:

```
all_deaths = all_deaths.rename(index=str, columns=age_dict)
pop = pop.rename(index=str, columns=pop_dict)
all_deaths['0-4'] = all_deaths[['0', '1', '2', '3', '4']].sum(axis=1)
pop['0-4'] = pop[['0', '1', '2', '3', '4']].sum(axis=1)
```

In [103]:

```
# Check new keys for pop
pop.head(5)
```

Out[103]:

	Country	Admin1	SubDiv	Year	Sex	Frmat	Total	0	1	2	...	65-69	70-74	75-79
0	1060	NaN	NaN	1980	1	7	137100.0	3400.0	15800.0	NaN	...	5300.0	NaN	2900.0
1	1060	NaN	NaN	1980	2	7	159000.0	4000.0	18400.0	NaN	...	6200.0	NaN	3400.0
2	1125	NaN	NaN	1955	1	2	5051500.0	150300.0	543400.0	NaN	...	51100.0	41600.0	14300.0
3	1125	NaN	NaN	1955	2	2	5049400.0	145200.0	551000.0	NaN	...	51100.0	50700.0	15800.0
4	1125	NaN	NaN	1956	1	2	5353700.0	158700.0	576600.0	NaN	...	54100.0	44000.0	14900.0

5 rows × 34 columns

In [104]:

```
# Optional Check new keys for all_deaths
all_deaths.head(5)
```

Out[104]:

	Country	Admin1	SubDiv	Year	List	Cause	Sex	Frmat	IM_Frmat	Total	...	80-84	85-89	90-94	95+	Unspecifie
0	1400	NaN	NaN	2001	101	1000	1	7	8	332	...	NaN	NaN	NaN	NaN	0.
1	1400	NaN	NaN	2001	101	1000	2	7	8	222	...	NaN	NaN	NaN	NaN	0.
2	1400	NaN	NaN	2001	101	1001	1	7	8	24	...	NaN	NaN	NaN	NaN	0.
3	1400	NaN	NaN	2001	101	1001	2	7	8	14	...	NaN	NaN	NaN	NaN	0.
4	1400	NaN	NaN	2001	101	1002	1	7	8	0	...	NaN	NaN	NaN	NaN	0.

5 rows × 40 columns

Calculate the population and total number deaths for iceland

In [105]:

```
# first find the country code for Iceland
country_codes[country_codes['name'] == 'Iceland'].values
```

Out[105]:

```
array([[4160, 'Iceland']], dtype=object)
```

In [106]:

```
# find population data for 2010 in Iceland

# this give 2 rows - one for male and one for female
iceland_pop = pop[(pop['Year'] == 2010) & (pop['Country'] == 4160)]
iceland_deaths = all_deaths[(all_deaths['Year'] == 2010) & (all_deaths['Country'] == 4160)]
```

In [107]:

```
iceland_pop['Total'].sum()
```

Out[107]:

```
318041.0
```

In [108]:

```
iceland_deaths['Total'].sum()
```

Out[108]:

```
4038
```

Calculate the population and total number deaths for Italy

In [109]:

```
# find country code for Italy
country_codes[country_codes.name == 'Italy'].values
```

Out[109]:

```
array([[4180, 'Italy']], dtype=object)
```

In [110]:

```
# find population data for 2010 in Italy

# this give 2 rows - one for male and one for female
italy_pop_2010 = pop[(pop['Year'] == 2010) & (pop['Country'] == 4180)]
italy_deaths_2010 = all_deaths[(all_deaths['Year'] == 2010) & (all_deaths['Country'] == 4180)]
```

In [111]:

```
# Total population in 2010
italy_pop_2010['Total'].sum()
```

Out[111]:

```
60483386.0
```

In [112]:

```
# Total number deaths in 2010
italy_deaths_2010['Total'].sum()
```

Out[112]:

```
1169230
```

Calculate the population and total number deaths for New Zealand

In [113]:

```
country_codes[country_codes['name'] == 'New Zealand'].values
```

Out[113]:

```
array([[5150, 'New Zealand']], dtype=object)
```

In [114]:

```
# find population data for 2010 in Italy

# this give 2 rows - one for male and one for female
nz_pop = pop[(pop['Year'] == 2010) & (pop['Country'] == 5150)]
nz_deaths = all_deaths[(all_deaths['Year'] == 2010) & (all_deaths['Country'] == 5150)]
```

In [115]:

```
# Population of New Zealand
nz_pop['Total'].sum()
```

Out[115]:

4367360.0

In [116]:

```
# Total number deaths
nz_deaths['Total'].sum()
```

Out[116]:

57298

Question 2

What was the distribution of deaths (all causes, all years) by age group in Italy?

- Visualise the results using a histogram.

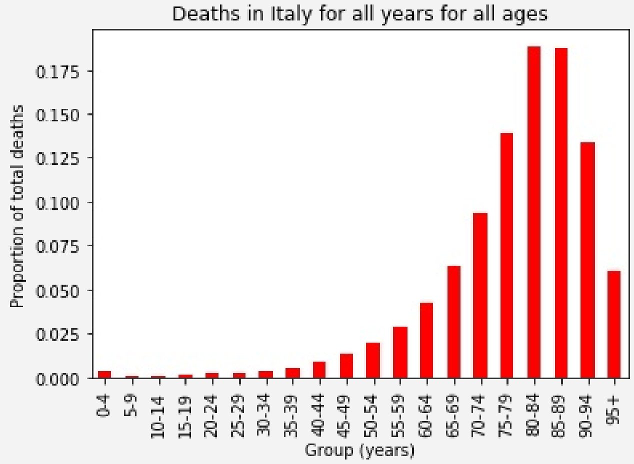
In [117]:

```
# First select all of the deaths for Italy
italy_all_deaths = all_deaths[all_deaths['Country'] == 4180]

# Now sum the totals for the specified age groups (dictionary created previously) and divide by the total of
 each column
italy_all_deaths_hist = italy_all_deaths[specified_age_groups].sum() / italy_all_deaths[specified_age_groups
].sum().sum()
```

Figure 6

In [118]:

```
# Figure 6
# Create a plot of the summed data and modify aspects of the graph
italy_all_deaths_hist.plot.bar(x=[0], y=[1], color = 'r')
plt.title('Deaths in Italy for all years for all ages')
plt.ylabel('Proportion of total deaths')
plt.xlabel('Group (years)')
plt.show()
```



Questions 3

What were the top five causes of death (top five ICD-10 terms) in Italy across all years for the Neoplasm ICD10-category (C00-D48)?

- Generate a table with the cause of death, the number of deaths, and the proportion of overall deaths.
- Generate a pie chart to visualize the proportion of deaths.

Step 1. Generate table for top 5 neoplasm causes of death

In [119]:

```
# Extract the data for C00-D48 codes first
italy_neoplasms = italy_all_deaths[italy_all_deaths['Cause'].str.startswith(('C', 'D0', 'D1', 'D2', 'D3', 'D
4'))]

# group them by cause and sum the values in each group
italy_neoplasms_grouped = italy_neoplasms.groupby('Cause').agg({'Total' : 'sum'}).reset_index()

# create a new column in the grouped df with the percentage that each cause accounts for
italy_neoplasms_grouped['Percentage'] = italy_neoplasms_grouped['Total']/ italy_neoplasms_grouped['Total'].s
um()*100
```

Table 9

In [120]:

```
# Table 9
# sort the data in descending order and generate table from the
italy_neoplasms_sorted = italy_neoplasms_grouped.sort_values('Total', ascending=False)
italy_neoplasms_top5 = italy_neoplasms_sorted.head(5)
italy_neoplasms_top5
```

Out[120]:

	Cause	Total	Percentage
143	C349	426451	18.964664
227	C509	155895	6.932792
92	C189	143188	6.367701
76	C169	125679	5.589059
118	C259	120070	5.339622

Step 2. Generate table with top 5 causes + all other causes

In [121]:

```
# first calculate what percentage the other ICD10 codes account for
Total_minus_top5 = italy_neoplasms_grouped['Total'].sum() - italy_neoplasms_top5["Total"].sum()

# create another df for other neoplasms and add it to the top 5 df
other_neoplasms = {'Cause' : 'Other neoplasms', 'Total' : [Total_minus_top5], 'Percentage' : 100-italy_neopl
asms_top5.Percentage.sum()}
other_neoplasms_df = pd.DataFrame(other_neoplasms)

all_neoplasms = pd.concat ((italy_neoplasms_top5, other_neoplasms_df))
all_neoplasms
```

Out[121]:

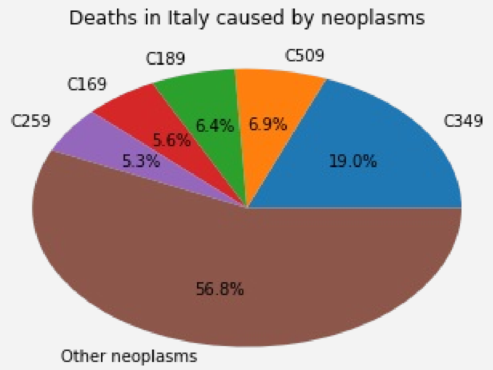
	Cause	Total	Percentage
143	C349	426451	18.964664
227	C509	155895	6.932792
92	C189	143188	6.367701
76	C169	125679	5.589059
118	C259	120070	5.339622
0	Other neoplasms	1277378	56.806162

Step 3. Generate pie chart of results

Figure 7

In [122]:

```
# Figure 7

# plot a pie chart with the data from the all_neoplasms df
plt.pie(all_neoplasms.Percentage, labels=all_neoplasms.Cause, autopct='%1.1f%%')
plt.title('Deaths in Italy caused by neoplasms')
plt.show()
```



Question 4

Are there differences by age group for deaths from Neoplasms (C00-D48) in Australia for 2010?

- Identify the top five age groups in Australia dying with a Neoplasms cause of death.

Step 1. Get Australia country code, use it to extract relevant info for Australia from mortality data

In [123]:

```
# First get Australia country code
country_codes[country_codes['name'] == 'Australia'].values
```

Out[123]:

```
array([[5020, 'Australia']], dtype=object)
```

In [124]:

```
# extract australia information
aus_all_deaths = all_deaths[all_deaths['Country'] == 5020]

# extract rows for deaths attributed to neoplasms
aus_neoplasms = aus_all_deaths[aus_all_deaths['Cause'].str.startswith(('C', 'D0', 'D1', 'D2', 'D3', 'D4'))]

# extract rows data related to 2010
aus_neoplasms_2010 = aus_neoplasms[aus_neoplasms['Year'] == 2010]

# sum the age group columns
aus_neoplasms_2010_by_age = aus_neoplasms_2010[specified_age_groups].sum()

# check the output
aus_neoplasms_2010_by_age
```

Out[124]:

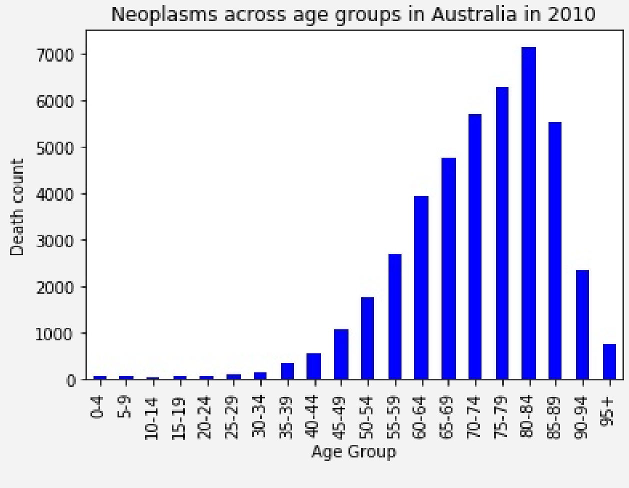
```
0-4      44.0
5-9      46.0
10-14    31.0
15-19    50.0
20-24    52.0
25-29    97.0
30-34    138.0
35-39    323.0
40-44    548.0
45-49    1065.0
50-54    1756.0
55-59    2695.0
60-64    3938.0
65-69    4768.0
70-74    5713.0
75-79    6291.0
80-84    7167.0
85-89    5520.0
90-94    2336.0
95+      735.0
dtype: float64
```

Step 2. Create bar chart to visulaise comparison between age groups

Figure 8

In [125]:

```
#Figure 8: Neoplasms across age groups in Australia in 2010

# create bar chart to see if there are differences in ages
aus_neoplasms_2010_by_age.plot.bar(x=[0], y=[1], color = 'b')
plt.title('Neoplasms across age groups in Australia in 2010')
plt.ylabel('Death count')
plt.xlabel('Age Group')
plt.show()
```



Step 3. Identify top 5 age groups for neoplasms

In [126]:

```
# identify the top 5 age groups where neoplasms are the cause of death
aus_neoplasms_2010_by_age.nlargest(5)
```

Out[126]:

```
80-84    7167.0
75-79    6291.0
70-74    5713.0
85-89    5520.0
65-69    4768.0
dtype: float64
```

Question 5

Compare and contrast the frequency of deaths by Neoplasms in Italy and Australia in 2010.

- Combine information on the population and deaths and describe your logic.
- Use descriptive statistics and plots.

Step 1. Calculate the frequency of deaths by neoplasms

In [127]:

```
# First get the 2010 data for Italy neoplasms. Already have population data for italy 2010
italy_neoplasms_2010 = italy_neoplasms[italy_neoplasms['Year'] == 2010]
```

In [128]:

```
# Get population information that we need for Australia
aus_pop_2010 = pop[(pop['Year'] == 2010) & (pop['Country'] == 5020)]
```

In [129]:

```
# Calculate the frequency of deaths for each country
aus_death_freq = aus_neoplasms_2010.Total.sum() / aus_pop_2010.Total.sum() * 100000
italy_death_freq = italy_neoplasms_2010.Total.sum() / italy_pop_2010.Total.sum() * 100000

# Declare information
print('Italy')
print('Deaths caused by neoplasms per 100,000 of the population: ' + str(italy_death_freq))
print('-----------------')
print('Australia')
print('Deaths caused by neoplasms per 100,000 of the population: ' + str(aus_death_freq))
```

```
Italy
Deaths caused by neoplasms per 100,000 of the population: 289.41170720832326
-----------------
Australia
Deaths caused by neoplasms per 100,000 of the population: 194.26380024859273
```

Step 2. Get information required to make a bar chart for Italy v Australia. Do as have done for australia in question 4.

In [130]:

```
# sum the age group columns
italy_neoplasms_2010_by_age = italy_neoplasms_2010[specified_age_groups].sum()
```

In [131]:

```
# get all deaths for australia in 2010
aus_all_deaths_2010 = all_deaths[(all_deaths['Country'] == 5020) & (all_deaths['Year'] == 2010)]
```

Step 3. Create a bar chart to compare the number deaths caused by neoplasms in 2010 in Italy and Australia

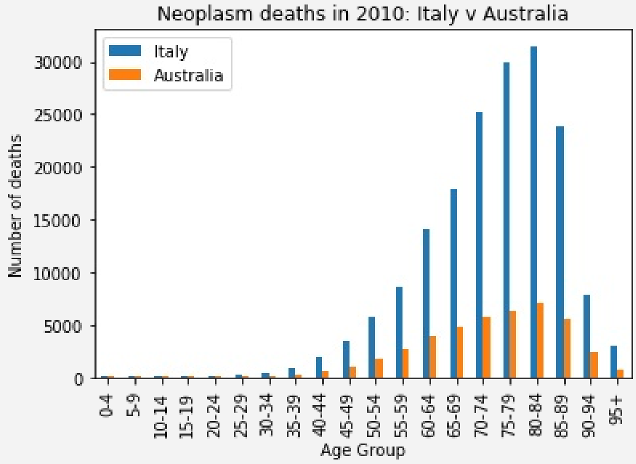
In [132]:

```
# calculate the number of deaths for each age group, make a new df and create a bar chart
italy_counts = np.array(italy_neoplasms_2010[specified_age_groups].sum())
aus_counts = np.array(aus_neoplasms_2010[specified_age_groups].sum())
counts_index = italy_neoplasms_2010[specified_age_groups].sum().index
italy_vs_aus_count = pd.DataFrame({'Italy': italy_counts, 'Australia':aus_counts}).set_index(counts_index)
```

Figure 9

In [133]:

```
# Figure 9
# plot a bar graph to illustrate
italy_vs_aus_count.plot.bar()
plt.title('Neoplasm deaths in 2010: Italy v Australia')
plt.ylabel('Number of deaths')
plt.xlabel('Age Group')
plt.show()
```



Step 4. Create a bar chart of the neoplasm deaths in 2010 as a percentage of all deaths for a given age group. Plot Italy and Austrlia data on graph for comparison

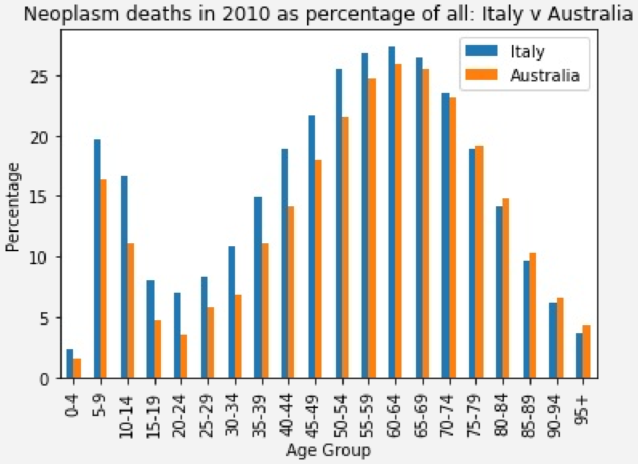
In [134]:

```
# calculate neoplasm deaths as a percentage of all deaths for each age group
italy_percentage_neoplasm_deaths = italy_neoplasms_2010[specified_age_groups].sum() / italy_deaths_2010[spec
ified_age_groups].sum() *100
aus_percentage_neoplasm_deaths = aus_neoplasms_2010[specified_age_groups].sum() / aus_all_deaths_2010[specif
ied_age_groups].sum() *100

# create a new df with the percentage neoplasm data
age_groups = np.array(aus_percentage_neoplasm_deaths.index)
italy_array = np.array(italy_percentage_neoplasm_deaths)
aus_array = np.array(aus_percentage_neoplasm_deaths)
italy_vs_aus = pd.DataFrame({'Italy': italy_array, 'Australia':aus_array}).set_index(age_groups)
```

Figure 10

In [135]:

```
#Figure 10
# plot a bar graph to illustrate
italy_vs_aus.plot.bar()
plt.title('Neoplasm deaths in 2010 as percentage of all: Italy v Australia')
plt.ylabel('Percentage')
plt.xlabel('Age Group')
plt.show()
```



Appendix 1 - difference between prescribing and gp demographic data

In [136]:

```
# group prescribing data by practice
prescribing_grouped = prescribing.groupby(['PRACTICE'])
len(prescribing_grouped)
```

Out[136]:

9578

In [137]:

```
# group prescribing data by practice (CODE column)
gp_prac_grouped = gp_prac.groupby(['CODE'])
len(gp_prac_grouped)
```

Out[137]:

7241

In [138]:

```
# what practices in prescibing data but not in gp demographic data
uncommon = prescribing[~prescribing['PRACTICE'].isin(gp_prac['CODE'])].dropna()
uncommon
uncommon_grouped = uncommon.groupby(['PRACTICE'])
len(uncommon_grouped)
```

Out[138]:

2387

In [139]:

```
# what practices in gp demogrpahic data but not in prescribing data
uncommon2 = gp_prac[~gp_prac['CODE'].isin(prescribing['PRACTICE'])].dropna()
uncommon2_grouped = uncommon2.groupby(['CODE'])
len(uncommon2_grouped)
```

Out[139]:

50

In [140]:

```
# total cost of prescriptions for GPs not in GP demographic data file
uncommon['ACT COST'].sum()
```

Out[140]:

```
11510031.8
```