In [1]:
```
%matplotlib inline
import matplotlib
import seaborn as sns
matplotlib.rcParams['savefig.dpi'] = 144
```

In [2]:
```
from static_grader import grader
```

DW Miniproject

Introduction

The objective of this miniproject is to exercise your ability to wrangle tabular data set and aggregate large data sets into meaningful summary statistics. We will be working with the same medical data used in the `pw` miniproject, but will be leveraging the power of Pandas to more efficiently represent and act on our data.

Downloading the data

We first need to download the data we'll be using from Amazon S3:

In [81]:
```
!mkdir dw-data
!aws s3 sync s3://dataincubator-wqu/dwdata-ease/ ./dw-data
```

```
mkdir: cannot create directory 'dw-data': File exists
```

Loading the data

Similar to the `PW` miniproject, the first step is to read in the data. The data files are stored as compressed CSV files. You can load the data into a Pandas DataFrame by making use of the `gzip` package to decompress the files and Panda's `read_csv` methods to parse the data into a DataFrame. You may want to check the Pandas documentation for parsing [CSV (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html) files for reference.

For a description of the data set please, refer to the [PW miniproject (./pw.ipynb)](./pw.ipynb).

In [3]:
```
import pandas as pd
import numpy as np
import gzip
```

In [4]:
```
scripts_file = gzip.open('./dw-data/201701scripts_sample.csv.gz', 'r')

scripts = pd.read_csv(scripts_file)
scripts.head()
```

Out[4]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28

In [5]:
```
scripts_file_16 = gzip.open('./dw-data/201606scripts_sample.csv.gz', 'r')

scripts_16 = pd.read_csv(scripts_file_16)
scripts_16.head()
```

Out[5]:

	Unnamed: 0	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	0	P84046	0101010R0	Simeticone_Dps 21mg/2.5ml	1	3.46	3.22	200
1	1	N81623	040201030	Risperidone_Tab 500mcg	1	1.15	1.18	28
2	2	E85658	0407020A0	Fentanyl_Transdermal Patch 12mcg/hr	2	50.36	46.65	20
3	3	E82106	212300001	Sylk Vag Moist 40g Tube	2	10.32	9.58	2
4	4	L83015	0403010N0	Imipramine HCl_Tab 10mg	1	2.10	1.96	56

In [6]:
```
practices_file = gzip.open('./dw-data/practices.csv.gz', 'r')

col_names=[ 'code', 'name', 'addr_1', 'addr_2', 'borough', 'village', 'post_code'
practices = pd.read_csv(practices_file, names = col_names)
practices.head()
```

Out[6]:

	code	name	addr_1	addr_2	borough	village	post_code
0	A81001	THE DENSHAM SURGERY	THE HEALTH CENTRE	LAWSON STREET	STOCKTON ON TEES	CLEVELAND	TS18 1HU
1	A81002	QUEENS PARK MEDICAL CENTRE	QUEENS PARK MEDICAL CTR	FARRER STREET	STOCKTON ON TEES	CLEVELAND	TS18 2AW
2	A81003	VICTORIA MEDICAL PRACTICE	THE HEALTH CENTRE	VICTORIA ROAD	HARTLEPOOL	CLEVELAND	TS26 8DB
3	A81004	WOODLANDS ROAD SURGERY	6 WOODLANDS ROAD	NaN	MIDDLESBROUGH	CLEVELAND	TS1 3BE
4	A81005	SPRINGWOOD SURGERY	SPRINGWOOD SURGERY	RECTORY LANE	GUISBOROUGH	NaN	TS14 7DJ

In [7]:
```
chem = pd.read_csv(gzip.open('./dw-data/chem.csv.gz', 'r'))
chem.head()
```

Out[7]:

	CHEM SUB	NAME
0	0101010A0	Alexitol Sodium
1	0101010B0	Almasilate
2	0101010C0	Aluminium Hydroxide
3	0101010D0	Aluminium Hydroxide With Magnesium
4	0101010E0	Hydrotalcite

now that we've loaded in the data, let's first replicate our results from the `PW` miniproject. Note that we are now working with a larger data set so the answers will be different than in the `PW` miniproject even if the analysis is the same.

Question 1: summary_statistics

In the `PW` miniproject we first calculated the total, mean, standard deviation, and quartile statistics of the `'items'` , `'quantity'` ', `'nic'` , and `'act_cost'` fields. To do this we had to write some functions to calculate the statistics and apply the functions to our data structure. The DataFrame has a `describe` method that will calculate most (not all) of these things for us.

Submit the summary statistics to the grader as a list of tuples: [('act_cost', (total, mean, std, q25, median, q75)), ...]

In [7]:
```
scripts.head()
```

Out[7]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28

In [10]:
```
scripts_sum = scripts[['items','nic','act_cost','quantity']].agg(['sum'])
scripts_sum
```

Out[10]:

	items	nic	act_cost	quantity
sum	8982877	72490066.94	67440722.67	725345509

In [12]:
```
scripts_des = scripts.describe()
scripts_des
```

Out[12]:

	items	nic	act_cost	quantity
count	1.001634e+06	1.001634e+06	1.001634e+06	1.001634e+06
mean	8.968223e+00	7.237181e+01	6.733070e+01	7.241622e+02
std	2.966303e+01	1.913729e+02	1.774985e+02	3.882238e+03
min	1.000000e+00	0.000000e+00	1.000000e-02	0.000000e+00
25%	1.000000e+00	7.840000e+00	7.350000e+00	2.800000e+01
50%	2.000000e+00	2.256000e+01	2.113000e+01	1.000000e+02
75%	6.000000e+00	6.440000e+01	5.993000e+01	3.360000e+02
max	3.333000e+03	1.963500e+04	1.817733e+04	6.526240e+05

In [17]:
```
scripts_output = pd.concat([scripts_sum, scripts_des.drop(['count','min','max'])]
scripts_output
```

Out[17]:

	items	nic	act_cost	quantity
sum	8.982877e+06	7.249007e+07	6.744072e+07	7.253455e+08
mean	8.968223e+00	7.237181e+01	6.733070e+01	7.241622e+02
std	2.966303e+01	1.913729e+02	1.774985e+02	3.882238e+03
25%	1.000000e+00	7.840000e+00	7.350000e+00	2.800000e+01
50%	2.000000e+00	2.256000e+01	2.113000e+01	1.000000e+02
75%	6.000000e+00	6.440000e+01	5.993000e+01	3.360000e+02

In [20]:
```
act_cost_l = []
for i in scripts_output['act_cost']:
    act_cost_l.append(i)

nic_l = []
for i in scripts_output['nic']:
    nic_l.append(i)

items_l = []
for i in scripts_output['items']:
    items_l.append(i)

quantity_l = []
for i in scripts_output['quantity']:
    quantity_l.append(i)

#print(output_list_of_tuples)
output_list = [('items', tuple(items_l)), ('quantity', tuple(quantity_l)), ('nic'
```

In [21]:
```
def summary_stats():
    return output_list
    #return [('items', (1,) * 6), ('quantity', (1,) * 6), ('nic', (1,) * 6), ('ac
```

In [22]:
```
grader.score('dw__summary_statistics', summary_stats)
```

```
==================
Your score:  1.0
==================
```

Question 2: most_common_item

We can also easily compute summary statistics on groups within the data. In the `pw` miniproject we had to explicitly construct the groups based on the values of a particular field. Pandas will handle that for us via the `groupby` method. This process is [detailed in the Pandas documentation (https://pandas.pydata.org/pandas-docs/stable/groupby.html)](https://pandas.pydata.org/pandas-docs/stable/groupby.html).

Use `groupby` to calculate the total number of items dispensed for each `'bnf_name'`. Find the item with the highest total and return the result as `(bnf_name, total)`.

In [9]:
```
scripts.head()
```

Out[9]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28

In [15]:
```
scripts_gp = scripts.groupby('bnf_name')['items'].count()
scripts_gp
```

Out[15]:
```
bnf_name
365 Film 4cm x 5cm VP Adh Film Dress           1
365 IV Transpt IV 10cm x 12cm VP Adh Fil       1
365 Non Adherent 10cm x 10cm Pfa Plas Fa       1
365 Non Woven Island 10cm x 10cm Adh Dre       2
3M Micropore Silicone 2.5cm x 5m Surg Ad      14
3M Micropore Silicone 5cm x 5m Surg Adh        7
3m Health Care_Cavilon Durable Barrier C     913
3m Health Care_Cavilon No Sting 1ml Barr     209
3m Health Care_Cavilon No Sting 3ml Barr      88
3m Health Care_Cavilon No Sting Barrier      466
4Head_Top Headache Relief Stick                4
A & P_Infants Pdrs                             4
A.S Saliva Orthana Spy 50ml (App)            116
A.S Saliva Orthana Spy Refill 500ml (App       6
A2A Spacer                                    30
A2A Spacer + Sml/Med Mask                     73
AAA_Sore Throat A/Spy 7.5g                     3
AMI_Corsinel Suportx Ab Tube Wte ExLge (       1
AMI_Corsinel Suportx Ab Tube Wte Lge (94       1
AMI_Corsinel Suportx Ab Tube Wte Med (86       2
AMI_Corsinel Suportx Easy Panel Belt Wte       1
AMI_Corsinel Suportx M/M Fle                   1
AMI_Corsinel Suportx StomaSafe Plus Belt       1
AMI_Suportx Hernia Support Girdles Fle H       7
AMI_Suportx Hernia Support Girdles Fle L      21
AMI_Suportx Hernia Support Girdles Male        5
AMI_Suportx Ostomy & A/Support Ladies Br       3
AMI_Suportx Ostomy & A/Support Ladies Sh       1
AMI_Suportx Ostomy Support Mens Shorts B       2
AMI_Suportx Ostomy Support Mens Shorts D       2
                                            ...
iMEDicare_Brief XL                             1
iMEDicare_Collection Bag Stabilizer (Hig       1
iMEDicare_Core Supporter Brief Med             1
iMEDicare_High Receptacle                      4
iMEDicare_Urine Collection Bag 1200ml          3
iMEDicare_Urine Collection Bag 2000ml          1
iMEDicare_Urine Collection Bag 500ml           5
imuDERM Urea Emollient                        12
kliniderm Foam Lite Border 10cm x 10cm W       3
kliniderm Foam Lite Border 15cm x 15cm W       1
kliniderm Foam Lite Border 7.5cmx7.5cm W       3
kliniderm Foam Slc 10cm x 10cm Wound Dre      15
kliniderm Foam Slc 5cm x 5cm Wound Dress       4
kliniderm Foam Slc Border 10cm x 10cm Wo      35
kliniderm Foam Slc Border 10cm x 20cm Wo       3
kliniderm Foam Slc Border 12.5cmx12.5cm        9
kliniderm Foam Slc Border 15cm x 15cm Wo       6
kliniderm Foam Slc Border 15cm x 20cm Wo       4
kliniderm Foam Slc Border 7.5cm x 7.5cm       19
kliniderm Foam Slc Sacrum Border 18cmx18       2
kliniderm superabsorbent 10cm x 10cm Pfa      15
kliniderm superabsorbent 10cm x 15cm Pfa      14
```

```
klinidern superabsorbent 20cm x 20cm Pfa    21
kliniderm superabsorbent 20cm x 30cm Pfa    22
nSpire PiKo-1 Stnd Range Peak Flow Meter     2
nSpire Pocket Peak Low Range Peak Flow M     5
nSpire Pocket Peak Stnd Range Peak Flow      3
nSpire Rep Mthpce Stnd Size Peak Flow Me     2
oraNurse_Toothpaste Orig (1450ppm)           4
palmdoc (Reagent)_Strips                     8
Name: items, Length: 13674, dtype: int64
```

In [22]:
```
sums = scripts.groupby('bnf_name').sum()['items']
sums.head()
```

Out[22]:
```
bnf_name
365 Film 4cm x 5cm VP Adh Film Dress        1
365 IV Transpt IV 10cm x 12cm VP Adh Fil    1
365 Non Adherent 10cm x 10cm Pfa Plas Fa    1
365 Non Woven Island 10cm x 10cm Adh Dre    4
3M Micropore Silicone 2.5cm x 5m Surg Ad   16
Name: items, dtype: int64
```

In [16]:
```
def most_common_item():
    sums = scripts.groupby('bnf_name').sum()['items']
    item = (sums.idxmax(), sums.max())
    return [item]
```

In [18]:
```
answer = most_common_item()
answer
```

Out[18]:
```
[('Omeprazole_Cap E/C 20mg', 222007)]
```

In [54]:
```
def most_common_item():
    return (0, 1643)
```

In [17]:
```
grader.score('dw__most_common_item', most_common_item)
```

```
==================
Your score:  1.0
==================
```

Question 3: items_by_region

Now let's find the most common item by post code. The post code information is in the `practices` DataFrame, and we'll need to `merge` it into the `scripts` DataFrame. Pandas provides [extensive documentation (https://pandas.pydata.org/pandas-docs/stable/merging.html)](https://pandas.pydata.org/pandas-docs/stable/merging.html) with diagrammed examples on different methods and approaches for joining data. The `merge` method is only one of many possible options.

Return your results as a list of tuples `(post code, item name, amount dispensed as % of total)`. Sort your results ascending alphabetically by post code and take only results from the first 100 post codes.

NOTE: Some practices have multiple postal codes associated with them. Use the alphabetically first postal code. Note some postal codes may have multiple `'bnf_name'` with the same prescription rate for the maximum. In this case, take the alphabetically first `'bnf_name'` (as in the PW miniproject).

In [8]:
```
practices.head()
```

Out[8]:

	code	name	addr_1	addr_2	borough	village	post_code
0	A81001	THE DENSHAM SURGERY	THE HEALTH CENTRE	LAWSON STREET	STOCKTON ON TEES	CLEVELAND	TS18 1HU
1	A81002	QUEENS PARK MEDICAL CENTRE	QUEENS PARK MEDICAL CTR	FARRER STREET	STOCKTON ON TEES	CLEVELAND	TS18 2AW
2	A81003	VICTORIA MEDICAL PRACTICE	THE HEALTH CENTRE	VICTORIA ROAD	HARTLEPOOL	CLEVELAND	TS26 8DB
3	A81004	WOODLANDS ROAD SURGERY	6 WOODLANDS ROAD	NaN	MIDDLESBROUGH	CLEVELAND	TS1 3BE
4	A81005	SPRINGWOOD SURGERY	SPRINGWOOD SURGERY	RECTORY LANE	GUISBOROUGH	NaN	TS14 7DJ

In [9]:
```
scripts.head()
```

Out[9]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28

In [10]:
```
srt = practices.sort_values('post_code').drop_duplicates('code', keep='first')
srt.head()
```

Out[10]:

	code	name	addr_1	addr_2	borough	village	post_co
1896	E82060	PARKBURY HOUSE SURGERY	PARKBURY HOUSE	ST.PETERS STREET	ST.ALBANS	HERTFORDSHIRE	AL1 3I
1871	E82031	MALTINGS SURGERY	THE MALTINGS SURGERY	8-14 VICTORIA STREET	ST ALBANS	HERTFORDSHIRE	AL1 3
1849	E82004	HATFIELD ROAD SURGERY	61 HATFIELD ROAD	NaN	ST.ALBANS	HERTFORDSHIRE	AL1 4
1848	E82002	WRAFTON HOUSE SURGERY	WRAFTON HOUSE SURGERY	9/11 WELLFIELD ROAD	HATFIELD	HERTFORDSHIRE	AL10 0
9812	Y05146	HCT LYMPHOEDEMA AT WEST ESSEX CCG	QUEENSWAY HEALTH CENTRE	QUEENSWAY	HATFIELD	HERTFORDSHIRE	AL10 0

In [11]:
```
merged = scripts.merge(srt, left_on='practice', right_on='code')
merged.head()
```

Out[11]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity	code	name
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60	B87016	WHITE ROSE SURGERY
1	B87016	0704020N0	Detrusitol_Tab 2mg	2	61.12	56.60	112	B87016	WHITE ROSE SURGERY
2	B87016	0604011G0	Elleste Solo_Tab 2mg	7	35.42	32.89	588	B87016	WHITE ROSE SURGERY
3	B87016	0407010F0	Co-Codamol Eff_Tab 30mg/500mg	16	169.54	158.06	1958	B87016	WHITE ROSE SURGERY
4	B87016	091000000	Centrum Advance 50+_Multivit/Mineral Tab	1	3.66	3.40	30	B87016	WHITE ROSE SURGERY

In [12]:
```
total_items_by_postcode = merged.groupby(['post_code','bnf_name'])[['items']].sum
total_items_by_postcode.head()
```

Out[12]:

		items
post_code	**bnf_name**	
AL1 3HD	**Acetic Acid_Ear Spy 2% 5ml**	2
	ActivHeal 10cm x 10cm Non-Adh Foam Wound	1
	Adcal-D3_Capl 750mg/200u	35
	AgaMatrix Ultra-Thin Lancets 0.35mm/28 G	5
	Amiodarone HCl_Tab 200mg	5

In [13]:
```
total_items_by_postcode.reset_index(inplace=True)
total_items_by_postcode.head()
```

Out[13]:

	post_code	bnf_name	items
0	AL1 3HD	Acetic Acid_Ear Spy 2% 5ml	2
1	AL1 3HD	ActivHeal 10cm x 10cm Non-Adh Foam Wound	1
2	AL1 3HD	Adcal-D3_Capl 750mg/200u	35
3	AL1 3HD	AgaMatrix Ultra-Thin Lancets 0.35mm/28 G	5
4	AL1 3HD	Amiodarone HCl_Tab 200mg	5

In [14]:
```
total_items_max = total_items_by_postcode.groupby('post_code')[['items']].max()
total_items_max.head()
```

Out[14]:

	items
post_code	
AL1 3HD	187
AL1 3JB	225
AL1 4JE	70
AL10 0BS	180
AL10 0LF	1

In [15]:
```
total_items_max.reset_index(inplace=True)
total_items_max.head()
```

Out[15]:

	post_code	items
0	AL1 3HD	187
1	AL1 3JB	225
2	AL1 4JE	70
3	AL10 0BS	180
4	AL10 0LF	1

In [16]:
```
total_items = total_items_max.merge(total_items_by_postcode, on=['post_code','ite
```

In [17]:
```
total_items.head()
```

Out[17]:

	post_code	items	bnf_name
0	AL1 3HD	187	Amoxicillin_Cap 500mg
1	AL1 3HD	187	Levothyrox Sod_Tab 25mcg
2	AL1 3JB	225	Bendroflumethiazide_Tab 2.5mg
3	AL1 4JE	70	Aspirin_Tab 75mg
4	AL10 0BS	180	Amoxicillin_Cap 500mg

In [18]:
```
total_items_sorted = total_items.sort_values('bnf_name').drop_duplicates(['post_c
total_items_sorted.head()
```

Out[18]:

	post_code	items	bnf_name
2558	GL53 9QU	4	3m Health Care_Cavilon Durable Barrier C
67	B14 7AG	53	3m Health Care_Cavilon Durable Barrier C
111	B24 9JN	162	3m Health Care_Cavilon Durable Barrier C
6439	SR4 7TP	1	3m Health Care_Cavilon No Sting 1ml Barr
7647	WN1 1NJ	2	3m Health Care_Cavilon No Sting 1ml Barr

In [19]:
```
total_items_sorted.sort_values('post_code', inplace=True)
total_items_sorted.head()
```

Out[19]:

	post_code	items	bnf_name
0	AL1 3HD	187	Amoxicillin_Cap 500mg
2	AL1 3JB	225	Bendroflumethiazide_Tab 2.5mg
3	AL1 4JE	70	Aspirin_Tab 75mg
4	AL10 0BS	180	Amoxicillin_Cap 500mg
5	AL10 0LF	1	ActiLymph Class 1 Combined Armsleeve + T

In [35]:
```
total_items_sorted.shape
```

Out[35]: (7572, 3)

In [30]:
```
s = total_items_by_postcode.groupby('post_code')[['items']].sum().reset_index()
s.head()
```

Out[30]:

	post_code	items
0	AL1 3HD	1822
1	AL1 3JB	1778
2	AL1 4JE	364
3	AL10 0BS	1451
4	AL10 0LF	3

In [36]:
```
s.shape
```

Out[36]: (7572, 2)

In [37]:
```
final_df = s.merge(total_items_sorted, on='post_code')
final_df.head()
```

Out[37]:

	post_code	items_x	items_y	bnf_name
0	AL1 3HD	1822	187	Amoxicillin_Cap 500mg
1	AL1 3JB	1778	225	Bendroflumethiazide_Tab 2.5mg
2	AL1 4JE	364	70	Aspirin_Tab 75mg
3	AL10 0BS	1451	180	Amoxicillin_Cap 500mg
4	AL10 0LF	3	1	ActiLymph Class 1 Combined Armsleeve + T

In [38]:
```
final_df['amt%'] = final_df['items_y'].astype(float)/final_df['items_x'].astype(f
final_df['amt%'].head()
```

Out[38]:
```
0    0.102634
1    0.126547
2    0.192308
3    0.124052
4    0.333333
Name: amt%, dtype: float64
```

In [39]:
```
final_df.head()
```

Out[39]:

	post_code	items_x	items_y	bnf_name	amt%
0	AL1 3HD	1822	187	Amoxicillin_Cap 500mg	0.102634
1	AL1 3JB	1778	225	Bendroflumethiazide_Tab 2.5mg	0.126547
2	AL1 4JE	364	70	Aspirin_Tab 75mg	0.192308
3	AL10 0BS	1451	180	Amoxicillin_Cap 500mg	0.124052
4	AL10 0LF	3	1	ActiLymph Class 1 Combined Armsleeve + T	0.333333

In [53]:
```
result = final_df[['post_code','bnf_name','amt%']]
result.head()
```

Out[53]:

	post_code	bnf_name	amt%
0	AL1 3HD	Amoxicillin_Cap 500mg	0.102634
1	AL1 3JB	Bendroflumethiazide_Tab 2.5mg	0.126547
2	AL1 4JE	Aspirin_Tab 75mg	0.192308
3	AL10 0BS	Amoxicillin_Cap 500mg	0.124052
4	AL10 0LF	ActiLymph Class 1 Combined Armsleeve + T	0.333333

In [57]:
```
result = result.head(100)
values = result.get_values().tolist()
```

In [58]:
```
final=[]

for item in values:
    final.append(tuple(item))
```

In [72]:
```
def items_by_region():
    return final
```

In [73]:
```
print items_by_region()
```

```
[('AL1 3HD', 'Amoxicillin_Cap 500mg', 0.1026344676180022), ('AL1 3JB', 'Bendrof
lumethiazide_Tab 2.5mg', 0.1265466816647919), ('AL1 4JE', 'Aspirin_Tab 75mg',
0.19230769230769232), ('AL10 0BS', 'Amoxicillin_Cap 500mg', 0.1240523776705720
2), ('AL10 0LF', 'ActiLymph Class 1 Combined Armsleeve + T', 0.333333333333333
3), ('AL10 0NL', 'Amitriptyline HCl_Tab 10mg', 0.0639686684073107), ('AL10 0U
R', 'Diazepam_Tab 10mg', 0.5434782608695652), ('AL10 8HP', 'Sertraline HCl_Tab
50mg', 0.10324129651860744), ('AL2 1ES', 'Levothyrox Sod_Tab 100mcg', 0.1307420
4946996468), ('AL2 3JX', 'Simvastatin_Tab 40mg', 0.0847231487658439), ('AL3 5E
R', 'Bisoprolol Fumar_Tab 2.5mg', 0.11428571428571428), ('AL3 5HB', 'Omeprazole
_Cap E/C 20mg', 0.16846758349705304), ('AL3 5JB', 'Alimemazine Tart_Tab 10mg',
1.0), ('AL3 5NF', 'Ramipril_Cap 10mg', 0.09449465899753492), ('AL3 5NP', 'Clopi
dogrel_Tab 75mg', 0.09023255813953489), ('AL3 7BL', 'Bendroflumethiazide_Tab 2.
5mg', 0.08917197452229299), ('AL3 8LJ', 'Aspirin Disper_Tab 75mg', 0.1789772727
2727273), ('AL5 2BT', 'Bisoprolol Fumar_Tab 2.5mg', 0.137660485021398), ('AL5 4
HX', 'Metformin HCl_Tab 500mg M/R', 0.07671601615074024), ('AL5 4QA', 'Lansopra
zole_Cap 30mg (E/C Gran)', 0.14298480786416443), ('AL6 9EF', 'Atorvastatin_Tab
20mg', 0.17326732673267325), ('AL6 9SB', 'Mometasone Fur_Oint 0.1%', 0.28260869
56521739), ('AL7 1BW', 'Irripod Sod Chlor Top Irrig 20ml', 0.1583710407239819),
('AL7 3UJ', 'Levothyrox Sod_Tab 50mcg', 0.13861386138613863), ('AL7 4HL', 'Clar
ithromycin_Tab 500mg', 0.07758094074526573), ('AL7 4PL', 'Levothyrox Sod_Tab 25
mcg', 0.11315136476426799), ('AL8 6JL', 'Latanoprost_Eye Dps 50mcg/ml', 0.71428
57142857143), ('AL8 7QG', 'Salbutamol_Inha 100mcg (200 D) CFF', 0.1581422692533
8037), ('AL9 7SN', 'Salbutamol_Inha 100mcg (200 D) CFF', 0.14134542705971279),
('B1 1EQ', 'Loperamide HCl_Cap 2mg', 0.5384615384615384), ('B1 3AL', 'Citalopra
m Hydrob_Tab 20mg', 0.11314475873544093), ('B1 3RA', 'Quetiapine_Tab 25mg', 0.2
173913043478260 8), ('B10 0BS', 'Salbutamol_Inha 100mcg (200 D) CFF', 0.17847769
02887139), ('B10 0JL', 'Desunin_Tab 800u', 0.17592592592592593), ('B10 0TU', 'A
mlodipine_Tab 5mg', 0.228310502283105), ('B10 0UG', 'Amoxicillin_Cap 500mg', 0.
10748299319727891), ('B10 9AB', 'Losartan Pot_Tab 50mg', 0.08932461873638345),
('B10 9QE', 'Fortisip Bottle_Liq (8 Flav)', 0.08923076923076922), ('B11 1LU',
'Paracet_Tab 500mg', 0.1488), ('B11 1TX', 'Fortisip Bottle_Liq (8 Flav)', 0.179
55112219451372), ('B11 3ND', 'GlucoRx Nexus (Reagent)_Strips', 0.07524271844660
194), ('B11 4AN', 'Metformin HCl_Tab 500mg', 0.16051502145922747), ('B11 4BW',
'Lansoprazole_Cap 30mg (E/C Gran)', 0.07043407043407043), ('B11 4DG', 'Paracet_
Tab 500mg', 0.3543123543123543), ('B11 4RA', 'Paracet_Tab 500mg', 0.16339869281
045752), ('B12 0UF', 'Lansoprazole_Cap 30mg (E/C Gran)', 0.1488833746898263),
('B12 0YA', 'Amoxicillin_Cap 500mg', 0.1375186846038864), ('B12 8HE', 'Atorvast
atin_Tab 40mg', 0.19387755102040816), ('B12 8QE', 'Atorvastatin_Tab 20mg', 0.12
996941896024464), ('B12 9LP', 'Aspirin Disper_Tab 75mg', 0.08866995073891626),
('B12 9RR', 'Aspirin Disper_Tab 75mg', 0.1111111111111111), ('B13 0HN', 'Amlodi
pine_Tab 5mg', 0.10548885077186965), ('B13 8JL', 'Nurse It Ster Dress Pack', 0.
31699496106275765), ('B13 8JS', 'Salbutamol_Inha 100mcg (200 D) CFF', 0.1542857
1428571428), ('B13 8QS', 'Lansoprazole_Cap 15mg (E/C Gran)', 0.115124153498871 3
3), ('B13 9HD', 'Influenza_Vac Inact 0.5ml Pfs', 0.5218037661050545), ('B13 9L
H', 'Amlodipine_Tab 5mg', 0.23478260869565218), ('B14 4DU', 'Paracet_Tab 500m
g', 0.18742985409652077), ('B14 4JU', 'Paracet_Tab 500mg', 0.1768465909090909),
('B14 5DJ', 'Atorvastatin_Tab 10mg', 0.10728476821192053), ('B14 5NG', 'Aspirin
Disper_Tab 75mg', 0.1897810218978102), ('B14 5SB', 'Amlodipine_Tab 5mg', 0.1604
3956043956045), ('B14 6AA', 'Amlodipine_Tab 10mg', 0.05718954248366013), ('B14
7AG', '3m Health Care_Cavilon Durable Barrier C', 0.08466453674121406), ('B14 7
NH', 'Omeprazole_Cap E/C 20mg', 0.12063492063492064), ('B15 1LZ', 'Levothyrox S
od_Tab 100mcg', 0.056847545219638244), ('B15 2QU', 'Salbutamol_Inha 100mcg (200
D) CFF', 0.10996563573883161), ('B15 3BU', 'Protopic_Oint 0.1%', 0.595238095238
0952), ('B15 3SJ', 'Metronidazole_Tab 400mg', 1.0), ('B16 0HH', 'Lisinopril_Tab
5mg', 0.2079207920792079), ('B16 0HZ', 'Amoxicillin_Cap 500mg', 0.1202185792349
```

```
7267), ('B16 0LU', 'Paracet_Tab 500mg', 0.21238938053097345), ('B16 8HA', 'Aspi
rin Disper_Tab 75mg', 0.19321148825065274), ('B16 9AL', 'Aspirin Disper_Tab 75m
g', 0.13713405238828968), ('B17 0HG', 'Omeprazole_Cap E/C 20mg', 0.139830508474
57626), ('B17 8DP', 'Lansoprazole_Cap 30mg (E/C Gran)', 0.15562735595045774),
('B17 8LG', 'Stexerol-D3_Tab 1 000u', 0.17080745341614906), ('B17 9DB', 'Omepra
zole_Cap E/C 20mg', 0.12826446280991735), ('B18 7AL', 'Aspirin Disper_Tab 75m
g', 0.07208765859284891), ('B18 7BA', 'Citalopram Hydrob_Tab 20mg', 0.087774294
6708464), ('B18 7EE', 'Metformin HCl_Tab 500mg', 0.3333333333333333), ('B19 1B
P', 'Aspirin Disper_Tab 75mg', 0.14380321665089876), ('B19 1HL', 'Metformin HCl
_Tab 500mg', 0.245136186770428), ('B19 1HS', 'Paracet_Tab 500mg', 0.24577572964
66974), ('B19 1TT', 'Metformin HCl_Tab 500mg', 0.26259541984732826), ('B19 2J
A', 'Amlodipine_Tab 5mg', 0.18029556650246306), ('B20 2BT', 'Simvastatin_Tab 20
mg', 0.19021739130434784), ('B20 2ES', 'GlucoRx Lancets 0.31mm/30 Gauge', 0.079
36507936507936), ('B20 2NR', 'Imuvac_Vac 0.5ml Pfs', 0.6362725450901804), ('B20
2QR', 'Bendroflumethiazide_Tab 2.5mg', 0.1571753986332574), ('B20 3HE', 'Simvas
tatin_Tab 20mg', 0.16216216216216217), ('B20 3QP', 'Ventolin_Evohaler 100mcg (2
00 D)', 0.18430034129692832), ('B21 0HL', 'Salbutamol_Inha 100mcg (200 D) CFF',
0.25), ('B21 0HR', 'Amlodipine_Tab 10mg', 0.16783216783216784), ('B21 9NH', 'Ad
cal-D3_Capl 750mg/200u', 0.17357222844344905), ('B21 9RY', 'Atorvastatin_Tab 10
mg', 0.043362495245340436), ('B23 5BX', 'Lansoprazole_Cap 30mg (E/C Gran)', 0.1
2195121951219512), ('B23 5DD', 'Ventolin_Evohaler 100mcg (200 D)', 0.2390837508
9477452), ('B23 5TJ', 'Bendroflumethiazide_Tab 2.5mg', 0.1712962962962963), ('B
23 6DJ', 'Lansoprazole_Cap 30mg (E/C Gran)', 0.11962931760741365)]
```

In [37]:
```python
def items_by_region():
    [('AL1 3HD', u'Amoxicillin_Cap 500mg', 0.1026344676180022)] * 100
```

In [61]:
```python
grader.score('dw__items_by_region', items_by_region)
```

```
==================
Your score:  1.0
==================
```

Question 4: script_anomalies

Drug abuse is a source of human and monetary costs in health care. A first step in identifying practitioners that enable drug abuse is to look for practices where commonly abused drugs are prescribed unusually often. Let's try to find practices that prescribe an unusually high amount of opioids. The opioids we'll look for are given in the list below.

In [8]:
```python
opioids = ['morphine', 'oxycodone', 'methadone', 'fentanyl', 'pethidine', 'bupren
```

In [9]:
```python
opioids_join = '|'.join(opioids)
opioids_join
```

Out[9]:
```
'morphine|oxycodone|methadone|fentanyl|pethidine|buprenorphine|propoxyphene|cod
eine'
```

In [10]:
```
chem.head()
```

Out[10]:

	CHEM SUB	NAME
0	0101010A0	Alexitol Sodium
1	0101010B0	Almasilate
2	0101010C0	Aluminium Hydroxide
3	0101010D0	Aluminium Hydroxide With Magnesium
4	0101010E0	Hydrotalcite

In [11]:
```
opioid_codes = chem.loc[chem['NAME'].str.contains(opioids_join, case=False)]['CHE
len(opioid_codes)
```

Out[11]: 35

In [12]:
```
chem_new = chem
chem_new.columns = ['bnf_code','chem_name']
chem_new.head()
```

Out[12]:

	bnf_code	chem_name
0	0101010A0	Alexitol Sodium
1	0101010B0	Almasilate
2	0101010C0	Aluminium Hydroxide
3	0101010D0	Aluminium Hydroxide With Magnesium
4	0101010E0	Hydrotalcite

In [13]:
```
scripts.head()
```

Out[13]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28

In [14]:
```
scripts.shape
```

Out[14]: (1001634, 7)

In [15]:
```
scrpt = scripts
scrpt.head()
```

Out[15]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28

In [16]:
```
scrpt['opioid_prescription'] = scrpt['bnf_code'].isin(opioid_codes)
```

In [17]:
```
scrpt.shape
```

Out[17]:
```
(1001634, 8)
```

In [18]:
```
scrpt.head()
```

Out[18]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity	opioid_prescription
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60	False
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140	False
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000	False
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2	False
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28	False

In [19]:
```
practices.head()
```

Out[19]:

	code	name	addr_1	addr_2	borough	village	post_code
0	A81001	THE DENSHAM SURGERY	THE HEALTH CENTRE	LAWSON STREET	STOCKTON ON TEES	CLEVELAND	TS18 1HU
1	A81002	QUEENS PARK MEDICAL CENTRE	QUEENS PARK MEDICAL CTR	FARRER STREET	STOCKTON ON TEES	CLEVELAND	TS18 2AW
2	A81003	VICTORIA MEDICAL PRACTICE	THE HEALTH CENTRE	VICTORIA ROAD	HARTLEPOOL	CLEVELAND	TS26 8DB
3	A81004	WOODLANDS ROAD SURGERY	6 WOODLANDS ROAD	NaN	MIDDLESBROUGH	CLEVELAND	TS1 3BE
4	A81005	SPRINGWOOD SURGERY	SPRINGWOOD SURGERY	RECTORY LANE	GUISBOROUGH	NaN	TS14 7DJ

In [20]:
```
pract = practices[['code', 'name']]
pract.columns = ['practice', 'name']
pract.head()
```

Out[20]:

	practice	name
0	A81001	THE DENSHAM SURGERY
1	A81002	QUEENS PARK MEDICAL CENTRE
2	A81003	VICTORIA MEDICAL PRACTICE
3	A81004	WOODLANDS ROAD SURGERY
4	A81005	SPRINGWOOD SURGERY

In [21]:
```
len(scrpt.loc[scrpt['opioid_prescription']])
```

Out[21]:
```
34597
```

In [22]:
```
opioids_per_practice = scrpt.groupby('practice')['opioid_prescription'].mean().re
opioids_per_practice.head()
```

Out[22]:
```
practice
A81001    0.025424
A81002    0.021834
A81004    0.048780
A81005    0.034965
A81006    0.036269
Name: frac, dtype: float64
```

In [23]:
```
opioids_per_practice_std = scrpt.groupby('practice')['opioid_prescription'].std()
opioids_per_practice_std.head()
```

Out[23]:
```
practice
A81001    0.158080
A81002    0.146462
A81004    0.216069
A81005    0.184337
A81006    0.187446
Name: frac_std, dtype: float64
```

In [24]:
```
overall_rate = scrpt['opioid_prescription'].mean()
overall_rate
```

Out[24]:
```
0.034540560723777348
```

In [25]:
```
overall_rate_std = scrpt['opioid_prescription'].std()
overall_rate_std
```

Out[25]:
```
0.18261309833034187
```

In [26]:
```
relative_opioids_per_practice = (opioids_per_practice - overall_rate).rename('rel
relative_opioids_per_practice.head()
```

Out[26]:
```
practice
A81001   -0.009117
A81002   -0.012706
A81004    0.014240
A81005    0.000424
A81006    0.001729
Name: relative, dtype: float64
```

In [27]:
```
opioid = scrpt.groupby('practice')['opioid_prescription'].sum().rename('opioid')
opioid.head()
```

Out[27]:
```
practice
A81001    3.0
A81002    5.0
A81004    8.0
A81005    5.0
A81006    7.0
Name: opioid, dtype: float64
```

In [28]:
```
total = scrpt.groupby('practice')['bnf_code'].count().rename('total')
total.head()
```

Out[28]:
```
practice
A81001    118
A81002    229
A81004    164
A81005    143
A81006    193
Name: total, dtype: int64
```

In [29]:
```
standard_error_per_practice = (overall_rate_std/(total**0.5)).rename('std_err')
standard_error_per_practice.head()
```

Out[29]:
```
practice
A81001    0.016811
A81002    0.012067
A81004    0.014260
A81005    0.015271
A81006    0.013145
Name: std_err, dtype: float64
```

In [30]:
```
opioid_scores = (relative_opioids_per_practice/standard_error_per_practice).renam
opioid_scores.head()
```

Out[30]:
```
practice
A81001   -0.542317
A81002   -1.052960
A81004    0.998614
A81005    0.027796
A81006    0.131525
Name: opioid_scores, dtype: float64
```

In [31]:
```
merged = pd.concat([opioid, total, opioids_per_practice, relative_opioids_per_pra
```

In [42]:
```
merged = merged.reset_index()
```

In [43]:
```
merged.head()
```

Out[43]:

	practice	opioid	total	frac	relative	std_err	opioid_scores
0	A81001	3.0	118	0.025424	-0.009117	0.016811	-0.542317
1	A81002	5.0	229	0.021834	-0.012706	0.012067	-1.052960
2	A81004	8.0	164	0.048780	0.014240	0.014260	0.998614
3	A81005	5.0	143	0.034965	0.000424	0.015271	0.027796
4	A81006	7.0	193	0.036269	0.001729	0.013145	0.131525

In [33]:
```
type(merged)
```

Out[33]:
```
pandas.core.frame.DataFrame
```

In [34]:
```
merged.shape
```

Out[34]:
```
(9230, 6)
```

In [84]:
```
pract.reset_index().head()
```

Out[84]:

	index	practice	name
0	0	A81001	THE DENSHAM SURGERY
1	1	A81002	QUEENS PARK MEDICAL CENTRE
2	2	A81003	VICTORIA MEDICAL PRACTICE
3	3	A81004	WOODLANDS ROAD SURGERY
4	4	A81005	SPRINGWOOD SURGERY

In [85]:
```
type(pract)
```

Out[85]: `pandas.core.frame.DataFrame`

In [86]:
```
final_df = merged.merge(pract, on='practice', how='left')
```

In [87]:
```
final_df.head()
```

Out[87]:

	practice	opioid	total	frac	relative	std_err	opioid_scores	name
0	A81001	3.0	118	0.025424	-0.009117	0.016811	-0.542317	THE DENSHAM SURGERY
1	A81001	3.0	118	0.025424	-0.009117	0.016811	-0.542317	THE DENSHAM SURGERY
2	A81002	5.0	229	0.021834	-0.012706	0.012067	-1.052960	QUEENS PARK MEDICAL CENTRE
3	A81004	8.0	164	0.048780	0.014240	0.014260	0.998614	WOODLANDS ROAD SURGERY
4	A81004	8.0	164	0.048780	0.014240	0.014260	0.998614	BLUEBELL MEDICAL CENTRE

In [88]:
```
final_df.sort_values('opioid_scores', ascending = False , inplace=True)
```

In [ ]:
```

```

In [89]:
```
final_df.head()
```

Out[89]:

	practice	opioid	total	frac	relative	std_err	opioid_scores	name
9644	Y04576	4.0	4	1.0	0.965459	0.091307	10.573825	ADDACTION NORTH DEVON
7795	P88636	4.0	4	1.0	0.965459	0.091307	10.573825	COMMUNITY DRUG TEAM
9645	Y04576	4.0	4	1.0	0.965459	0.091307	10.573825	ADDACTION NORTH DEVON
8718	Y02770	4.0	5	0.8	0.765459	0.081667	9.372928	NY HORIZONS HARROGATE
8332	Y01640	4.0	5	0.8	0.765459	0.081667	9.372928	SALFORD DRUG TEAM

In [90]:
```
final = final_df.drop_duplicates('name')
final.head()
```

Out[90]:

	practice	opioid	total	frac	relative	std_err	opioid_scores	name
9644	Y04576	4.0	4	1.0	0.965459	0.091307	10.573825	ADDACTION NORTH DEVON
7795	P88636	4.0	4	1.0	0.965459	0.091307	10.573825	COMMUNITY DRUG TEAM
8718	Y02770	4.0	5	0.8	0.765459	0.081667	9.372928	NY HORIZONS HARROGATE
8332	Y01640	4.0	5	0.8	0.765459	0.081667	9.372928	SALFORD DRUG TEAM
9079	Y03738	3.0	3	1.0	0.965459	0.105432	9.157201	CRI DRUG SERVICE

In [97]:
```
result = final[['name','opioid_scores','total']]
result.head()
```

Out[97]:

	name	opioid_scores	total
9644	ADDACTION NORTH DEVON	10.573825	4
7795	COMMUNITY DRUG TEAM	10.573825	4
8718	NY HORIZONS HARROGATE	9.372928	5
8332	SALFORD DRUG TEAM	9.372928	5
9079	CRI DRUG SERVICE	9.157201	3

In [98]:
```
result = result.head(100)
values = result.get_values().tolist()
```

In [99]:
```
answer=[]

for item in values:
    answer.append(tuple(item))
```

In [100]:
```
def script_anomalies():
    return answer
```

In [101]:
```
print script_anomalies()
```

```
[('ADDACTION NORTH DEVON', 10.573824639125657, 4), ('COMMUNITY DRUG TEAM', 10.5
73824639125657, 4), ('NY HORIZONS HARROGATE', 9.3729275495027, 5), ('SALFORD DR
UG TEAM', 9.3729275495027, 5), ('CRI DRUG SERVICE', 9.157200752644645, 3), ('CR
I CAMDEN COMMUNITY DRUG TREATMENT SERV', 9.157200752644645, 3), ('WORCESTERSHIR
E RECOVERY PARTNERSHIP', 9.157200752644645, 3), ('HAYFIELD GPWSI SERVICE', 9.15
7200752644645, 3), ('ADDICTIONS SERVICE', 9.145409115671335, 8), ('TURNING POIN
T CROYDON RECOVERY NETWORK', 8.479054497238709, 6), ('DUDLEY COMMUNITY PALLIATI
VE MEDICINE', 8.060276069984052, 10), ('MACMILLAN ST HELENS SP  PALLIATIVE CAR
E', 7.835795414652864, 4), ('LIFELINE SOUTHWARK', 7.835795414652864, 4), ('STAR
- EAST SUSSEX DRUG & ALCOHOL SERV', 7.835795414652864, 4), ('PLYMOUTH SPECIALIS
T ADDICTION SERVICE', 7.835795414652864, 4), ("ST ANN'S HOSPICE/PAL CARE", 7.83
5795414652864, 4), ('WESTMINSTER DAWS', 7.835795414652864, 4), ('COMMUNITY/SLH
DAY CASE PALLIATIVE CARE', 7.778588563496554, 7), ('DR LOCAL CARE DIRECT OOH',
7.778588563496554, 7), ('LOCAL CARE DIRECT OOH', 7.778588563496554, 7), ('HACKN
EY SUBSTANCE MISUSE SERVICE', 7.4768231054031515, 2), ('SOMERSET 999 GP CAR',
7.4768231054031515, 2), ('TURNING POINT TROWBRIDGE', 7.4768231054031515, 2),
('ADDACTION BRADFORD - BCSMS', 7.4768231054031515, 2), ('LIFELINE KIRKLEES', 7.
4768231054031515, 2), ('HACKNEY RECOVERY SERVICE', 7.4768231054031515, 2), ('RB
WM DAAT', 7.4768231054031515, 2), ('NMS CRI NOTTINGHAMSHIRE', 7.20933008395967
7, 8), ('LOROS', 6.923959761381819, 5), ('NRP SOUTH (NSFT)', 6.923959761381819,
5), ('BANES DOCTORS URGENT CARE (PAULTON)', 6.923959761381819, 5), ('LIFT SUBST
ANCEMISUSE LEICESTERSHIRE', 6.923959761381819, 5), ('TURNING POINT LEICESTER &
LEICESTERSHIRE', 6.923959761381819, 5), ('HEART OF KENT HOSPICE', 6.92395976138
1819, 5), ("ST WILFRID'S HOSPICE", 6.33700124944254, 15), ('COMPTON PALLIATIVE
CARE TEAM', 6.243462996976479, 6), ('LUTON DRUG SERVICE', 6.243462996976479,
6), ('TRAFFORD DRUG SERVICE', 5.995596933047784, 3), ('MK SUBSTANCE MISUSE SERV
ICE', 5.995596933047784, 3), ('RECOVERY CENTRE NEWTON AYCLIFFE', 5.995596933047
784, 3), ('TURNING POINT SOMERSET', 5.995596933047784, 3), ('ST OSWALDS PALLIAT
IVE CARE', 5.995596933047784, 3), ('CRIME REDUCTION INITIATIVES', 5.99559693304
7784, 3), ('PROJECT ANSWER', 5.995596933047784, 3), ('GRANTHAM MINOR INJURIES A
ND ILLNESS UNIT', 5.995596933047784, 3), ('SWANSWELL SUBSTANCEMISUSE LEICESTERS
HIRE', 5.995596933047784, 3), ('BOWTHORPE CARE VILLAGE - NPL', 5.99559693304778
4, 3), ('PALLIATIVE CARE', 5.995596933047784, 3), ('NORTH LANCASHIRE INSPIRE',
5.995596933047784, 3), ('SUFFOLK RECOVERY SERVICE - LOWESTOFT', 5.9955969330477
84, 3), ('CRI ASPIRE', 5.995596933047784, 3), ('EAST CHESHIRE SUBSTANCE MISUSE
SERVICE', 5.995596933047784, 3), ('ST CUTHBERTS HOSPICE', 5.977048242001399, 1
1), ('HUMBER NHS FT - BEVERLEY LOCALITY TEAM', 5.708833017673109, 7), ('WHEATFI
ELDS SRC HOSPICE', 5.708833017673109, 7), ('HUMBER NHS FT - POCK & GOOLE LOC TE
AM', 5.393165324296798, 13), ('CHCP - POCK & GOOLE LOC TEAM', 5.39316532429679
8, 13), ('HOSPICE IN THE WEALD', 5.286912319562829, 1), ('SPECIALIST NURSE TEAM
PARKINSON', 5.286912319562829, 1), ('HARDWICK GP ACUTE VISIT', 5.28691231956282
9, 1), ('ADDACTION NORTH SOMERSET', 5.286912319562829, 1), ('DR IRIS', 5.286912
319562829, 1), ('COMM HOSP - OAK WARD', 5.286912319562829, 1), ('SPRINGHILL HOS
PICE', 5.286912319562829, 1), ('KNOWSLEY INTEGRATED RECOVERY SERV (CRI)', 5.286
912319562829, 1), ('PALLIATIVE CARE SERVICE', 5.286912319562829, 1), ('NEW HOP
E', 5.286912319562829, 1), ('ST CLARE HOSPICE', 5.286912319562829, 1), ('PALLIA
TIVE MEDICINE', 5.286912319562829, 1), ('IDAS', 5.286912319562829, 1), ('BRANCH
ING OUT', 5.286912319562829, 1), ("ST MARGARET'S HOSPICE", 5.286912319562829,
1), ('COMMUNITY MIDWIFE SERVICES', 5.286912319562829, 1), ('CRI WIRRAL', 5.2869
12319562829, 1), ('ADDACTION', 5.286912319562829, 1), ('ADDACTION THURROCK', 5.
286912319562829, 1), ('SOUTHEND TREATMENT AND RECOVERY SERVICE', 5.286912319562
829, 1), ('BOLTON COMMUNITY DRUG AND ALCOHOL SERV', 5.286912319562829, 1), ('LI
FELINE REDCAR & CLEVELAND', 5.286912319562829, 1), ("ISIS WOMEN'S SERVICE", 5.2
86912319562829, 1), ('THE BEACON', 5.286912319562829, 1), ('SHARED CARE DRUG SE
RVICE', 5.286912319562829, 1), ('TURNING POINT CHIPPENHAM', 5.286912319562829,
```

```
1), ('LPS THE SURGERY', 5.286912319562829, 1), ('ALCOHOL SERVICES FOR THE COMMU
NITY', 5.286912319562829, 1), ('CRI BROMLEYCOMMUNITY DRUGS PROJECT', 5.28691231
9562829, 1), ('WOKING HOSPICE', 5.286912319562829, 1), ('NORTH BRADFORD DRUG SE
RVICE', 5.286912319562829, 1), ('GRANTHAM ADDACTION', 5.286912319562829, 1),
('SMART DRUG & ALCOHOL SERVICES WOKINGHAM', 5.286912319562829, 1), ('CALDERDALE
COMMUNITY DERMATOLOGY SERVICE', 5.286912319562829, 1), ('TURNING POINT MEDWAY',
5.286912319562829, 1), ('COMMUNITY MEDICAL TEAM - EAST ELMBRIDGE', 5.2869123195
62829, 1), ('EAST RIDING PARTNERSHIP', 5.286912319562829, 1), ('BARNET RECOVERY
CENTRE FINCHLEY WDP', 5.286912319562829, 1), ('CRI-NEWHAM RISE', 5.286912319562
829, 1), ('E.R.C.J.P. SERVICE (EAST)', 5.286912319562829, 1), ('ADDACTION SUBST
ANCE MISUSE SERVICE', 5.286912319562829, 1), ('MCAS', 5.286912319562829, 1),
('JOHN TAYLOR HOSPICE', 5.27325105224802, 8)]
```

In [ ]:

These are generic names for drugs, not brand names. Generic drug names can be found using the `'bnf_code'` field in `scripts` along with the `chem` table.. Use the list of opioids provided above along with these fields to make a new field in the `scripts` data that flags whether the row corresponds with a opioid prescription.

In [ ]:
```
scripts_chem = pd.merge(scripts, chem, left_on='bnf_code', right_on='CHEM SUB')
scripts_chem.head()
```

Note

Owing to helpful comments and answers, I managed to partially resolve problems. Here are my intermediate results...

Number of opioids in chem dataframe is **35**. Number of opioid prescriptions in scripts dataframe is **34597**. There is **9230** practices in total, to be analyzed. The mean value of opioid prescription rate for the whole population is 0.0345405607238. From individual practices means, I subtracted the total mean to get relative prescription rate. Standard deviation (std) for the whole population (based on binomial distribution) is 0.18261309833. For each practice I determined standard error by dividing standard deviation with the square root of total prescriptions, including opioid and non-opioid ones. Finally, I determined opioid scores by dividing relative prescription rate with standard error. Floats were used everywhere, to avoid accidental integer division.

In [ ]:

Now for each practice calculate the proportion of its prescriptions containing opioids.

Hint: Consider the following list: `[0, 1, 1, 0, 0, 0]` . What proportion of the entries are 1s? What is the mean value?

In [ ]:
```
opioids_per_practice = ...
```

In [ ]:

How do these proportions compare to the overall opioid prescription rate? Subtract off the proportion of all prescriptions that are opioids from each practice's proportion.

In [ ]:
```
relative_opioids_per_practice = ...
```

Now that we know the difference between each practice's opioid prescription rate and the overall rate, we can identify which practices prescribe opioids at above average or below average rates. However, are the differences from the overall rate important or just random deviations? In other words, are the differences from the overall rate big or small?

To answer this question we have to quantify the difference we would typically expect between a given practice's opioid prescription rate and the overall rate. This quantity is called the **standard error**, and is related to the **standard deviation**, σ. The standard error in this case is

$$\frac{\sigma}{\sqrt{n}}$$

where n is the number of prescriptions each practice made. Calculate the standard error for each practice. Then divide `relative_opioids_per_practice` by the standard errors. We'll call the final result `opioid_scores`.

In [ ]:
```
standard_error_per_practice = ...
opioid_scores = ...
```

The quantity we have calculated in `opioid_scores` is called a **z-score**:

$$\frac{\bar{X} - \mu}{\sqrt{\sigma^2/n}}$$

Here $\bar{X}$ corresponds with the proportion for each practice, μ corresponds with the proportion across all practices, σ^2 corresponds with the variance of the proportion across all practices, and n is the number of prescriptions made by each practice. Notice $\bar{X}$ and n will be different for each practice, while μ and σ are determined across all prescriptions, and so are the same for every z-score. The z-score is a useful statistical tool used for hypothesis testing, finding outliers, and comparing data about different types of objects or events.

Now that we've calculated this statistic, take the 100 practices with the largest z-score. Return your result as a list of tuples in the form `(practice_name, z-score, number_of_scripts)`. Sort your tuples by z-score in descending order. Note that some practice codes will correspond with multiple names. In this case, use the first match when sorting names alphabetically.

In [ ]:
```
unique_practices = ...
anomalies = ...
```

In [44]:
```
def script_anomalies():
    return [("ADDACTION NORTH DEVON", 10.5738246391, 4.0)] * 100
```

In [102]:
```
grader.score('dw__script_anomalies', script_anomalies)
```

```
==================
Your score:  0.94
==================
```

Question 5: script_growth

Another way to identify anomalies is by comparing current data to historical data. In the case of identifying sites of drug abuse, we might compare a practice's current rate of opioid prescription to their rate 5 or 10 years ago. Unless the nature of the practice has changed, the profile of drugs they prescribe should be relatively stable. We might also want to identify trends through time for business reasons, identifying drugs that are gaining market share. That's what we'll do in this question.

We'll load in beneficiary data from 6 months earlier, June 2016, and calculate the percent growth in prescription rate from June 2016 to January 2017 for each `bnf_name`. Normalize the percent growth in prescriptions of individual items by the percent change in total number of prescriptions (think about whether this normalization should be a division or a subtraction). We'll return the 50 items with largest growth and the 50 items with the largest shrinkage (i.e. negative percent growth) as a list of tuples sorted by growth rate in descending order in the format `(script_name, growth_rate, raw_2016_count)`. You'll notice that many of the 50 fastest growing items have low counts of prescriptions in 2016. Filter out any items that were prescribed less than 50 times.

In [206]:
```
drugs_16=scripts_16[['bnf_name', 'items']]
drugs_16=drugs_16.groupby('bnf_name').count().reset_index().drop_duplicates()
drugs_16.columns=[['bnf_name', 'count_16']]
drugs_16.head()
```

Out[206]:

	bnf_name	count_16
0	365 Film 4cm x 5cm VP Adh Film Dress	1
1	365 Non Adherent 10cm x 10cm Pfa Plas Fa	1
2	365 Non Adherent 5cm x 5cm Pfa Plas Face	2
3	365 Non Woven Island 10cm x 20cm Adh Dre	1
4	365 Non Woven Island 5cm x 7.2cm Adh Dre	1

In [207]:
```
drugs_17=scripts[['bnf_name', 'items']]
drugs_17=drugs_17.groupby('bnf_name').count().reset_index().drop_duplicates()
drugs_17.columns=[['bnf_name', 'count_17']]
drugs_17.head()
```

Out[207]:

	bnf_name	count_17
0	365 Film 4cm x 5cm VP Adh Film Dress	1
1	365 IV Transpt IV 10cm x 12cm VP Adh Fil	1
2	365 Non Adherent 10cm x 10cm Pfa Plas Fa	1
3	365 Non Woven Island 10cm x 10cm Adh Dre	2
4	3M Micropore Silicone 2.5cm x 5m Surg Ad	14

In [208]:
```
total_16=drugs_16['count_16'].sum()
total_17=drugs_17['count_17'].sum()
```

In [209]:
```
drugs_16['rate_16']=drugs_16['count_16']/total_16.astype(float)
drugs_17['rate_17']=drugs_17['count_17']/total_17.astype(float)
```

In [210]:
```
drugs=drugs_16.merge(drugs_17, on='bnf_name', how='inner').drop_duplicates('bnf_n
drugs.head()
```

Out[210]:

	bnf_name	count_16	rate_16	count_17	rate_17
0	365 Film 4cm x 5cm VP Adh Film Dress	1	9.765577e-07	1	9.983687e-07
1	365 Non Adherent 10cm x 10cm Pfa Plas Fa	1	9.765577e-07	1	9.983687e-07
2	3m Health Care_Cavilon Durable Barrier C	905	8.837847e-04	913	9.115106e-04
3	3m Health Care_Cavilon No Sting 1ml Barr	228	2.226552e-04	209	2.086591e-04
4	3m Health Care_Cavilon No Sting 3ml Barr	103	1.005854e-04	88	8.785644e-05

In [211]:
```
drugs['growth']=(drugs['rate_17']-drugs['rate_16'])/drugs['rate_16'].astype(float
drugs.head()
```

Out[211]:

	bnf_name	count_16	rate_16	count_17	rate_17	growth
0	365 Film 4cm x 5cm VP Adh Film Dress	1	9.765577e-07	1	9.983687e-07	0.022335
1	365 Non Adherent 10cm x 10cm Pfa Plas Fa	1	9.765577e-07	1	9.983687e-07	0.022335
2	3m Health Care_Cavilon Durable Barrier C	905	8.837847e-04	913	9.115106e-04	0.031372
3	3m Health Care_Cavilon No Sting 1ml Barr	228	2.226552e-04	209	2.086591e-04	-0.062860
4	3m Health Care_Cavilon No Sting 3ml Barr	103	1.005854e-04	88	8.785644e-05	-0.126549

In [212]:
```
total_growth=(total_17-total_16)/total_16.astype(float) #-0.0218465730148
total_growth
```

Out[212]: -0.021846573014780202

In [213]:
```
drugs['rel_growth']=drugs['growth']+total_growth
drugs=drugs[(drugs['count_16']>50)]
drugs = drugs[['bnf_name', 'rel_growth', 'count_16']]
drugs.head()
```

Out[213]:

	bnf_name	rel_growth	count_16
2	3m Health Care_Cavilon Durable Barrier C	0.009525	905
3	3m Health Care_Cavilon No Sting 1ml Barr	-0.084707	228
4	3m Health Care_Cavilon No Sting 3ml Barr	-0.148396	103
5	3m Health Care_Cavilon No Sting Barrier	-0.152489	548
8	A.S Saliva Orthana Spy 50ml (App)	0.176040	99

In [214]:
```
drugs.sort_values('rel_growth', ascending=False, inplace=True)
drugs.head()
```

Out[214]:

	bnf_name	rel_growth	count_16
1483	Butec_Transdermal Patch 10mcg/hr	3.946340	57
1485	Butec_Transdermal Patch 5mcg/hr	2.672189	75
4483	Fostair NEXThaler_Inh 200mcg/6mcg (120D)	1.726507	77
3673	Dulaglutide_Inj 1.5mg/0.5ml Pf Dev	1.363601	78
30	Abasaglar KwikPen_100u/ml 3ml Pf Pen	1.283053	55

In [215]:
```
final = pd.concat([drugs.iloc[0:50], drugs.iloc[len(drugs)-50: len(drugs)]], axis
final.head()
```

Out[215]:

	bnf_name	rel_growth	count_16
1483	Butec_Transdermal Patch 10mcg/hr	3.946340	57
1485	Butec_Transdermal Patch 5mcg/hr	2.672189	75
4483	Fostair NEXThaler_Inh 200mcg/6mcg (120D)	1.726507	77
3673	Dulaglutide_Inj 1.5mg/0.5ml Pf Dev	1.363601	78
30	Abasaglar KwikPen_100u/ml 3ml Pf Pen	1.283053	55

In [216]:
```
arr = final.values
arr[:5]
```

Out[216]:
```
array([['Butec_Transdermal Patch 10mcg/hr', 3.946340409455863, 57],
       ['Butec_Transdermal Patch 5mcg/hr', 2.672188773229457, 75],
       ['Fostair NEXThaler_Inh 200mcg/6mcg (120D)', 1.7265072272651727, 77],
       ['Dulaglutide_Inj 1.5mg/0.5ml Pf Dev', 1.363600606294229, 78],
       ['Abasaglar KwikPen_100u/ml 3ml Pf Pen', 1.2830530392006783, 55]], dtype
=object)
```

In [217]:
```
lst=[]

for item in arr:
    lst.append(tuple(item))
```

In [218]:
```
def script_growth():
    return lst
```

In [219]:
```
grader.score('dw__script_growth', script_growth)
```

```
==================
Your score:  1.0
==================
```

Question 6: rare_scripts

Does a practice's prescription costs originate from routine care or from reliance on rarely prescribed treatments? Commonplace treatments can carry lower costs than rare treatments because of efficiencies in large-scale production. While some specialist practices can't help but avoid prescribing rare medicines because there are no alternatives, some practices may be prescribing a unnecessary amount of brand-name products when generics are available. Let's identify practices whose costs disproportionately originate from rarely prescribed items.

First we have to identify which `'bnf_code'` are rare. To do this, find the probability p of a prescription having a particular `'bnf_code'` if the `'bnf_code'` was randomly chosen from the unique options in the beneficiary data. We will call a `'bnf_code'` rare if it is prescribed at a rate less than $0.1p$.

In [ ]:
```
p = ...
rates = ...
rare_codes = ...
scripts['rare'] = ...
```

In [27]:
```
# Calculating probability for each bnf_code
p = 1 / float(scripts['bnf_code'].nunique())
```

In [28]:
```
# Calculating rare prescription i.e ratio of count of bnf_code per prescription a
rates = scripts.groupby('bnf_code')['bnf_code'].count().rename('count_prescriptio
rates.head()
```

Out[28]:

	bnf_code	count_prescription
0	0101010C0	30
1	0101010F0	3
2	0101010G0	364
3	0101010I0	21
4	0101010J0	38

In [29]:
```
total_count = scripts['bnf_code'].count().astype(float)
total_count
```

Out[29]:
```
1001634.0
```

In [30]:
```
rare = rates['count_prescription'] / total_count
rare.head()
```

Out[30]:
```
0    0.000030
1    0.000003
2    0.000363
3    0.000021
4    0.000038
Name: count_prescription, dtype: float64
```

In [31]:
```
#Filtering the records on the basis of  rate < 0.1p

rare_codes =  rates[rare < 0.1*p]['bnf_code'].tolist()
rare_codes[:5]
```

Out[31]:
```
['0101010C0', '0101010F0', '0101010I0', '0101010J0', '0101010N0']
```

In [32]:
```
#Creating 'rare' column in scripts
scrpt = scripts
scrpt['rare'] = scrpt['bnf_code'].isin(rare_codes)
```

In [33]:
```
scrpt.head()
```

Out[33]:

	practice	bnf_code	bnf_name	items	nic	act_cost	quantity	rare
0	B87016	213200001	MucoClear Sod Chlor 6% Inh Soln 4ml	1	38.94	36.06	60	False
1	L85017	0503021C0	Aciclovir_Tab 800mg	4	13.44	12.49	140	False
2	M81001	090900000	SMA_High Energy Milk Ready To Use	1	147.60	136.65	15000	False
3	F85063	0601011L0	Ins Humalog_100u/ml 10ml Vl	1	33.22	30.77	2	False
4	B86667	1001010P0	Naproxen_Tab E/C 500mg	1	3.51	3.36	28	False

In [34]:
```
# Calculate total sum of act_cost for all treatment per prescription
treatment_sum = scrpt.groupby('practice')['act_cost'].sum()
treatment_sum.head()
```

Out[34]:
```
practice
A81001     5447.55
A81002    42759.63
A81004    11804.14
A81005     9746.88
A81006    24645.30
Name: act_cost, dtype: float64
```

In [35]:
```
# Calculate cost for rare treatment per prescription
rare_treatment_sum = scrpt[scrpt['rare']].groupby('practice')['act_cost'].sum()
rare_treatment_sum.head()
```

Out[35]:
```
practice
A81001     51.84
A81002    237.75
A81004      8.64
A81005    463.32
A81006     68.48
Name: act_cost, dtype: float64
```

In [121]:
```
# Calculate proportion of costs that originate from rare treatment i.e ratio of r

rare_cost_prop = (rare_treatment_sum / treatment_sum)
rare_cost_prop.fillna(0, inplace=True)

mean_val = scrpt[scrpt['rare']]['act_cost'].sum() / scrpt['act_cost'].sum()

relative_rare_cost_prop = rare_cost_prop - mean_val
standard_errors = relative_rare_cost_prop.std()

mean_val, standard_errors
```

Out[121]:
```
(0.01554553433731762, 0.0678173558942156)
```

In [122]:
```
rare_scores = (relative_rare_cost_prop / standard_errors).rename('z-score').reset_
rare_scores.columns =['code','z-score']
rare_scores.head()
```

Out[122]:

	code	z-score
0	A81001	-0.088905
1	A81002	-0.147239
2	A81004	-0.218434
3	A81005	0.471703
4	A81006	-0.188254

In [38]:
```
# Sorting rare_score in descendig order of z-score
rare_scores.sort_values('z-score', ascending=False, inplace=True)
rare_scores.columns =['code','z-score']
rare_scores.head()
```

Out[38]:

	code	z-score
8034	Y03106	14.516261
8760	Y04676	14.516261
7441	Y00799	14.516261
8403	Y03976	14.516261
7600	Y01731	14.516261

In [88]:
```
practices.head()
```

Out[88]:

	code	name	addr_1	addr_2	borough	village	post_code
0	A81001	THE DENSHAM SURGERY	THE HEALTH CENTRE	LAWSON STREET	STOCKTON ON TEES	CLEVELAND	TS18 1HU
1	A81002	QUEENS PARK MEDICAL CENTRE	QUEENS PARK MEDICAL CTR	FARRER STREET	STOCKTON ON TEES	CLEVELAND	TS18 2AW
2	A81003	VICTORIA MEDICAL PRACTICE	THE HEALTH CENTRE	VICTORIA ROAD	HARTLEPOOL	CLEVELAND	TS26 8DB
3	A81004	WOODLANDS ROAD SURGERY	6 WOODLANDS ROAD	NaN	MIDDLESBROUGH	CLEVELAND	TS1 3BE
4	A81005	SPRINGWOOD SURGERY	SPRINGWOOD SURGERY	RECTORY LANE	GUISBOROUGH	NaN	TS14 7DJ

In [89]:
```
pract = practices[['code','name']].drop_duplicates('code')
pract.head()
```

Out[89]:

	code	name
0	A81001	THE DENSHAM SURGERY
1	A81002	QUEENS PARK MEDICAL CENTRE
2	A81003	VICTORIA MEDICAL PRACTICE
3	A81004	WOODLANDS ROAD SURGERY
4	A81005	SPRINGWOOD SURGERY

In [90]:
```
pract.shape
```

Out[90]: (10843, 2)

In [ ]:

In [91]:
```
rare_scores.shape
```

Out[91]: (9230, 2)

In [92]:
```
#Joining rare_scores with practices to get (practice, name, z-score)
joined_data = pd.merge(rare_scores, pract, on='code', how ='inner')#.drop_duplica
joined_data.head()
```

Out[92]:

	code	z-score	name
0	A81001	-0.088905	THE DENSHAM SURGERY
1	A81002	-0.147239	QUEENS PARK MEDICAL CENTRE
2	A81004	-0.218434	WOODLANDS ROAD SURGERY
3	A81005	0.471703	SPRINGWOOD SURGERY
4	A81006	-0.188254	TENNANT STREET MEDICAL PRACTICE

In [103]:
```
joined_data.shape
```

Out[103]: (9230, 3)

In [108]:
```
joined_data.sort_values('z-score', ascending=False, inplace=True)
```

In [109]:
```
joined_data.reset_index(inplace=True)
```

In [110]:
```
required_cols = ['code', 'name', 'z-score']
final_df = joined_data[required_cols]
```

In [111]:
```
final_df.head()
```

Out[111]:

	code	name	z-score
0	Y03106	CRI GPWSI	14.516261
1	Y04704	PCOC RESPIRATORY SERVICE	14.516261
2	Y00215	ORTHOPAEDIC & RHEUMATOLOGY	14.516261
3	Y03268	CLECKHEATON DERMATOLOGY GPSI	14.516261
4	Y03572	DR WRIGHT ED CLINIC (BELMONT)	14.516261

In [114]:
```
final_df.shape
```

Out[114]: (9230, 3)

In [115]:
```
final_df = final_df.head(100)
arr = final_df.values
```

In [116]:
```
lst=[]

for item in arr:
    lst.append(tuple(item))
```

In [117]:
```
def rare_scripts():
    return lst
```

In [112]:
```
def rare_scripts():
    #return [("Y03106", "CRI GPWSI", 14.516)] * 100
    return [tuple(i) for i in final_df.values][:100]
```

Now for each practice, calculate the proportion of costs that originate from prescription of rare treatments (i.e. rare `'bnf_code'`). Use the `'act_cost'` field for this calculation.

In [ ]:
```
rare_cost_prop = ...
```

Now we will calculate a z-score for each practice based on this proportion. First take the difference of `rare_cost_prop` and the proportion of costs originating from rare treatments across all practices.

In [ ]:
```
relative_rare_cost_prop = ...
```

Now we will estimate the standard errors (i.e. the denominator of the z-score) by simply taking the standard deviation of this difference.

In [ ]:
```
standard_errors = ...
```

Finally compute the z-scores. Return the practices with the top 100 z-scores in the form `(practice_name, proportion)` . Note that some practice codes will correspond with multiple names. In this case, use the first match when sorting names alphabetically.

In [ ]:
```
rare_scores = ...
```

In [165]:
```
def rare_scripts():
    return [("Y03106", "CRI GPWSI", 14.516)] * 100
    rare_scores.sort_values('act_cost', ascending=False, inplace=True)
```

In [118]:
```
grader.score('dw__rare_scripts', rare_scripts)
```

```
==================
Your score:  0.96
==================
```