BIS 634 HW1

Exercise 1:

In [1]:
```
def temp_tester(temp):
    def case_temp(case_temp):
        if(abs(case_temp - temp) <= 1):
            print('true')
        else:
            print('false')
        return
    return case_temp



human_tester = temp_tester(37)
chicken_tester = temp_tester(41.1)

chicken_tester(42) # True — i.e. not a fever for a chicken
human_tester(42)   # False — this would be a severe fever for a human
chicken_tester(43) # False
human_tester(35)   # False — too low
human_tester(98.6) # False — normal in degrees F but our reference temp was in degrees C
```

```
true
false
false
false
false
```

A simple function to identify whether the temperature is normal or not.

Exercise 2:

1. Has columns name, age, weight, and eye color
2. 152361 cases in the total population

In [21]:
```
print(list(data.columns))
print(data['name'].count())
```

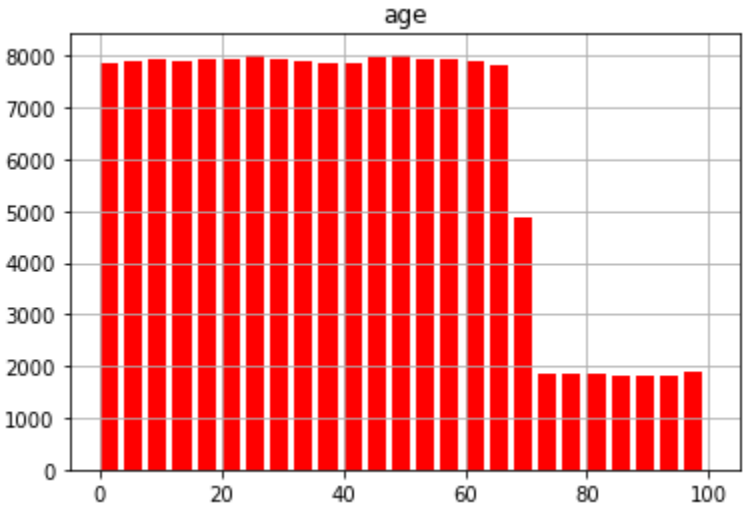
```
['name', 'age', 'weight', 'eyecolor']
152361
```

3. The distribution of age is:

```
In [22]: print(data['age'].describe())
```

```
count    152361.000000
mean         39.510528
std          24.152760
min           0.000748
25%          19.296458
50%          38.468955
75%          57.623245
max          99.991547
Name: age, dtype: float64
```

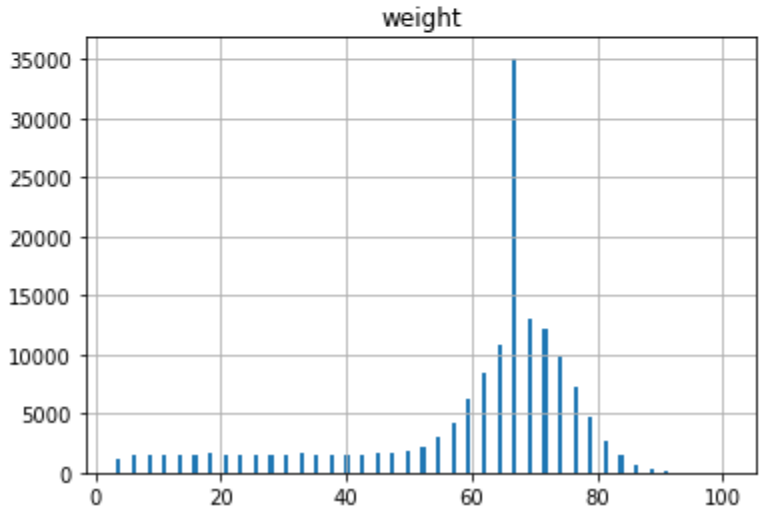
4. The number of bins is 25 as the range is 0-100. Each bin represents 4 years. The presentation of chart is clear and detailed. The outlier is around age 70, the population size drops sharply



5. The number of bins is 40 for weight as it is a continuous value. More bins can demonstrate more details. The population of 68 kg is extraordinarily large

```
print(data['weight'].describe())
weight_dist=data.hist(column=['weight'],bins=40,width=0.7);
```

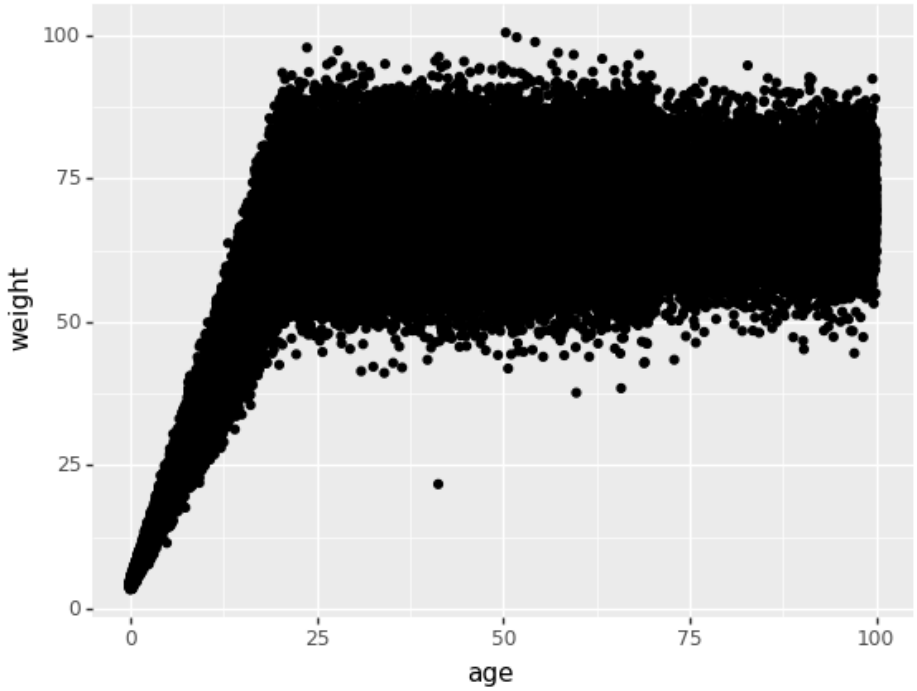
```
count    152361.000000
mean         60.884134
std          18.411824
min           3.382084
25%          58.300135
50%          68.000000
75%          71.529860
max         100.435793
Name: weight, dtype: float64
```



6. When age is less than 20, age and weight are strongly correlated. Body weight increases with age. After the age of about 20, weight stabilizes around 60-85 kg. In addition, people between the ages of 20 and 100 seem to have a greater standard deviation in body weight.

7. There is an outlier in the graph, his name is Anthony Freeman. His weight is unusually low. He can be found by the constraints.

In [85]:
```
from plotnine import *
ggplot(df, aes(x='age', y='weight')) + geom_point()
```



Out[85]: <ggplot: (154721296366)>

In [88]:
```
outlier=data.loc[data['age']>35].loc[data['weight']<25]
print(outlier)
```

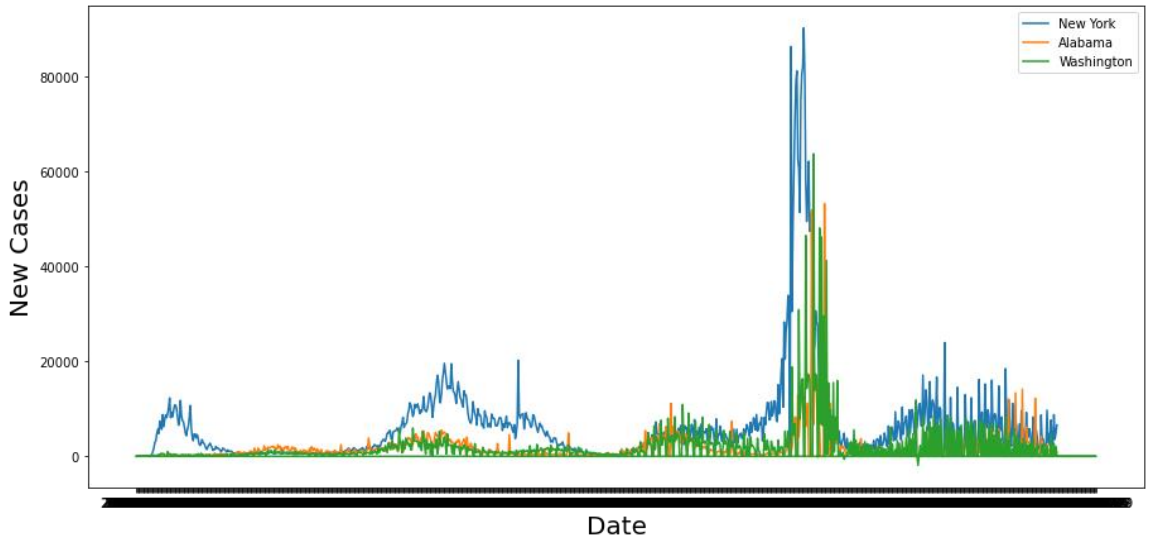
```
                name   age  weight eyecolor
537  Anthony Freeman  41.3    21.7    green
```

Exercise 3:

Data download on 9/22/2022

Data source is The New York Times

1.



Function plot_state_cases() is used to plot state new cases. Too many selected states will overlap and be difficult to read. Intensive areas can also be difficult to read. The time range is too close that the X-axis is too dense.

2. Function for identify the date of new cases peak

```
def peak(state):
    case=data[data.state==state].cases
    new_case=[case.iloc[i+1]-case.iloc[i] for i in range(case.shape[0]-1)]
    new_case=np.insert(new_case,0,case.iloc[0])
    state_case=data[data.state==state]
    new=state_case.loc[:,'new cases']=new_case

    new_case_number=state_case['new cases']
    peak_number=new_case_number.max()

    date = state_case[state_case['new cases'] == peak_number]['date']
    return date.iloc[0]

print(peak('Connecticut'))
print(peak('Washington'))
print(peak('New York'))
print(peak('New Jersey'))
print(peak('Illinois'))
```

```
2022-01-10
2022-01-18
2022-01-08
2021-01-04
2022-01-18
```

3.Peak comparision between the states

```
from dateutil.parser import parse as parse
def state_comp(state1, state2):
    peak1 = peak(state1)
    peak2 = peak(state2)

    date1 = parse(peak1)
    date2 = parse(peak2)

    if (date1 == date2):
        print(state1, 'and', state2, 'had same peak on',peak1)
    elif (date1 < date2):
        print(state1, "had its peak", (date2-date1).days, "days earlier than", state2)
    else:
        print(state2, "had its peak", (date1-date2).days, "days earlier than", state1)


state_comp('Washington', 'New York')
state_comp('Connecticut', 'Ohio')
```

```
New York had its peak 10 days earlier than Washington
Connecticut had its peak 5 days earlier than Ohio
```

Exercise 4:

1. The DescriptorName of DescriptorUI D007154 is Immune System Diseases.

2. The DescriptorUI of DescriptorName Nervous System Diseases is "D009422".

3. The DescriptorNames of items in the MeSH hierarchy that are subtypes of both Nervous System Diseases and D007154 is:

```
['Autoimmune Hypophysitis',
 'Ataxia Telangiectasia',
 'Diffuse Cerebral Sclerosis of Schilder',
 'Encephalomyelitis, Acute Disseminated',
 'Encephalomyelitis, Autoimmune, Experimental',
 'Leukoencephalitis, Acute Hemorrhagic',
 'Kernicterus',
 'Multiple Sclerosis',
 'Myasthenia Gravis',
 'Myelitis, Transverse',
 'Neuritis, Autoimmune, Experimental',
 'Neuromyelitis Optica',
 'Polyradiculoneuropathy',
 'Giant Cell Arteritis',
 'Uveomeningoencephalitic Syndrome',
 'AIDS Dementia Complex',
 'Lambert-Eaton Myasthenic Syndrome',
 'Stiff-Person Syndrome',
 'POEMS Syndrome',
 'Miller Fisher Syndrome',
 'Autoimmune Diseases of the Nervous System',
 'Guillain-Barre Syndrome',
 'Polyradiculoneuropathy, Chronic Inflammatory Demyelinating',
 'Demyelinating Autoimmune Diseases, CNS',
 'Vasculitis, Central Nervous System',
 'Multiple Sclerosis, Chronic Progressive',
 'Multiple Sclerosis, Relapsing-Remitting',
 'Myasthenia Gravis, Autoimmune, Experimental',
 'Nervous System Autoimmune Disease, Experimental',
 'Myasthenia Gravis, Neonatal',
 'AIDS Arteritis, Central Nervous System',
 'Lupus Vasculitis, Central Nervous System',
 'Mevalonate Kinase Deficiency',
 'Microscopic Polyangiitis',
 'Anti-N-Methyl-D-Aspartate Receptor Encephalitis']
```

4. These diseases are a subtype of both immune system disorders and neurological disorders. Because their tree numbers include both C20 and C10.

Code

Exercise 1:

In [1]:
```
def temp_tester(temp):
    def case_temp(case_temp):
        if(abs(case_temp - temp) <= 1):
            print('true')
        else:
            print('false')
        return
    return case_temp



human_tester = temp_tester(37)
chicken_tester = temp_tester(41.1)

chicken_tester(42) # True — i.e. not a fever for a chicken
human_tester(42)   # False — this would be a severe fever for a human
chicken_tester(43) # False
human_tester(35)   # False — too low
human_tester(98.6) # False — normal in degrees F but our reference temp was in degrees C
```

```
true
false
false
false
false
```

Exercise 2:

In [24]:
```
import pandas as pd
import sqlite3
with sqlite3.connect("hw1-population.db") as db:
    data = pd.read_sql_query("SELECT * FROM population", db)
df = pd.DataFrame(data)
df.head(5)
```

Out[24]:

	name	age	weight	eyecolor
0	Edna Phelps	88.895690	67.122450	brown
1	Cara Yasso	9.274597	29.251244	brown
2	Gail Rave	18.345613	55.347903	brown
3	Richard Adams	16.367545	70.352184	brown
4	Krista Slater	49.971604	70.563859	brown

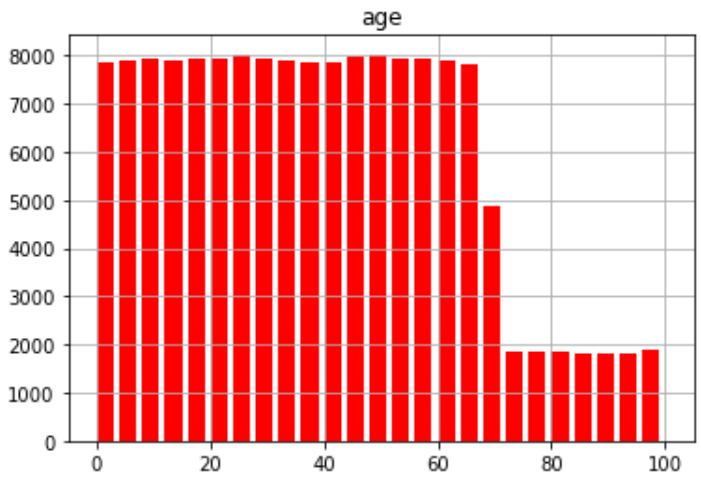
In [21]:
```
print(list(data.columns))
print(data['name'].count())
```

```
['name', 'age', 'weight', 'eyecolor']
152361
```

In [22]:
```
print(data['age'].describe())
```

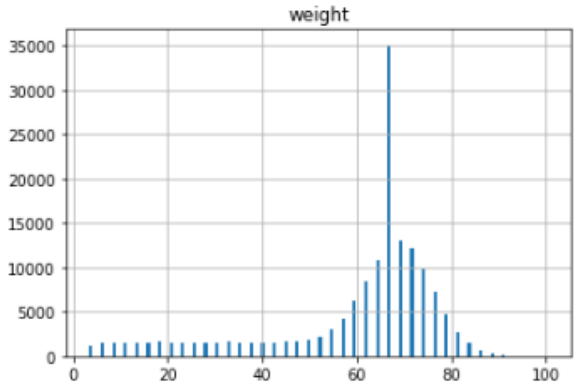
```
count    152361.000000
mean         39.510528
std          24.152760
min           0.000748
25%          19.296458
50%          38.468955
75%          57.623245
max          99.991547
Name: age, dtype: float64
```

In [75]:
```
age_dist=data.hist(column=['age'],bins=25,color='red',width=3)
```

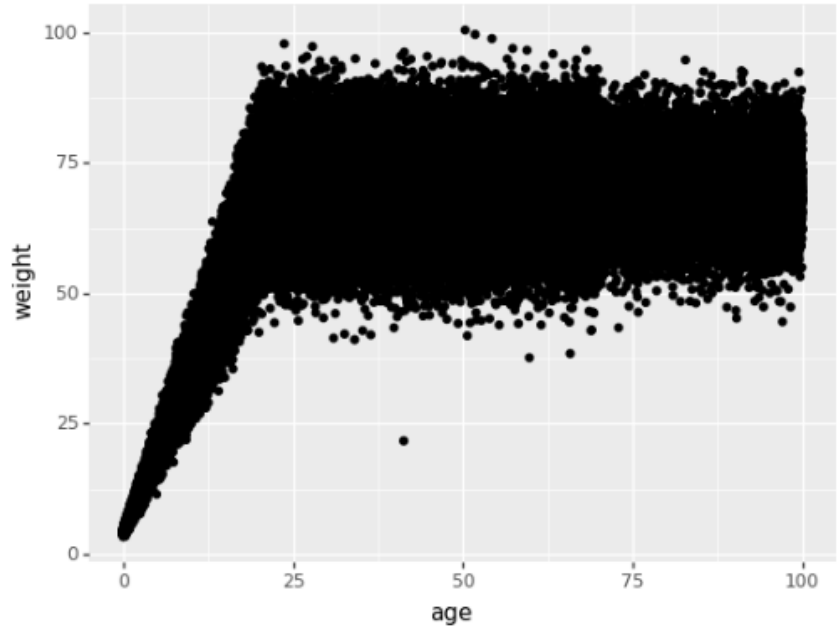


In [83]:
```
print(data['weight'].describe())
weight_dist=data.hist(column=['weight'],bins=40,width=0.7);
```

```
count    152361.000000
mean         60.884134
std          18.411824
min           3.382084
25%          58.300135
50%          68.000000
75%          71.529860
max         100.435793
Name: weight, dtype: float64
```



In [85]:
```
from plotnine import *
ggplot(df, aes(x='age', y='weight')) + geom_point()
```



Out[85]:
```
<ggplot: (154721296366)>
```

In [88]:
```
outlier=data.loc[data['age']>35].loc[data['weight']<25]
print(outlier)
```

```
                name   age  weight eyecolor
537  Anthony Freeman  41.3    21.7    green
```

Exercise 3:

In [200]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

original_data = pd.read_csv("us-states.csv")

# simplify dataset
data = original_data.drop(['fips','deaths'], axis=1)
df.head(1000)
```

Out[200]:

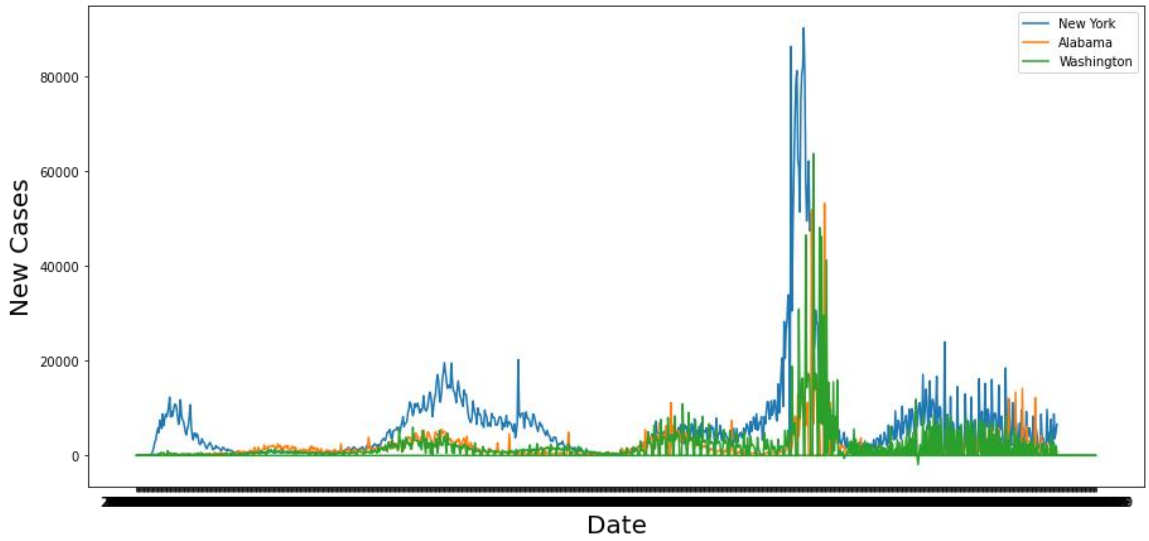
	date	state	cases
0	2020-01-21	Washington	1
1	2020-01-22	Washington	1
2	2020-01-23	Washington	1
3	2020-01-24	Illinois	1
4	2020-01-24	Washington	1
...	...	...	...
995	2020-03-20	Oregon	114
996	2020-03-20	Pennsylvania	269
997	2020-03-20	Puerto Rico	14
998	2020-03-20	Rhode Island	44
999	2020-03-20	South Carolina	126

1000 rows × 3 columns

In [231]:
```
def plot_state_cases(state):
    for i in range(len(state)):
        case=data[data.state==state[i]].cases
        new_case=[case.iloc[i+1]-case.iloc[i] for i in range(case.shape[0]-1)]
        new_case=np.insert(new_case,0,case.iloc[0])
        state_case=data[data.state==state[i]]
        new=state_case.loc[:,'new cases']=new_case
        state_case = state_case.reset_index(drop=True)

        plt.plot(state_case['date'], state_case['new cases'], label = state[i])
        plt.legend()
        plt.xlabel('Date',fontsize=20)
        plt.ylabel('New Cases',fontsize=20)

plot_state_cases(['New York','Alabama','Washington'])
```



In [230]:
```
def peak(state):
    case=data[data.state==state].cases
    new_case=[case.iloc[i+1]-case.iloc[i] for i in range(case.shape[0]-1)]
    new_case=np.insert(new_case,0,case.iloc[0])
    state_case=data[data.state==state]
    new=state_case.loc[:,'new cases']=new_case

    new_case_number=state_case['new cases']
    peak_number=new_case_number.max()

    date = state_case[state_case['new cases'] == peak_number]['date']
    return date.iloc[0]

print(peak('Connecticut'))
print(peak('Washington'))
print(peak('New York'))
print(peak('New Jersey'))
print(peak('Illinois'))
```

```
2022-01-10
2022-01-18
2022-01-08
2021-01-04
2022-01-18
```

In [237]:
```
from dateutil.parser import parse as parse
def state_comp(state1, state2):
    peak1 = peak(state1)
    peak2 = peak(state2)

    date1 = parse(peak1)
    date2 = parse(peak2)

    if (date1 == date2):
        print(state1, 'and', state2, 'had same peak on',peak1)
    elif (date1 < date2):
        print(state1, "had its peak", (date2-date1).days, "days earlier than", state2)
    else:
        print(state2, "had its peak", (date1-date2).days, "days earlier than", state1)


state_comp('Washington', 'New York')
state_comp('Connecticut', 'Ohio')
```

```
New York had its peak 10 days earlier than Washington
Connecticut had its peak 5 days earlier than Ohio
```

Exercise 4:

In [9]:
```
import xml.etree.ElementTree as ET
from pprint import pprint as pp
tree = ET.parse('desc2022.xml')
root = tree.getroot()
```

In [11]:
```
def find_des_name(UI):
    for DescRe in root:
        if (DescRe.find('DescriptorUI').text == UI):
            print(DescRe.find('DescriptorName').find('String').text)

find_des_name('D007154')
```

```
Immune System Diseases
```

In [21]:
```
def find_des_ui(Name):
    for DescRe in root:
        if (DescRe.find('DescriptorName').find('String').text == Name):
            print(DescRe.find('DescriptorUI').text)

find_des_ui('Nervous System Diseases')
```

```
D009422
```

In [21]:
```
def shared_child(parent1, parent2):
    shared_child = []
    #get treenumbers
    for child in root:
        if (child.find('DescriptorUI').text == parent1):
            tree1 = child.find('TreeNumberList').findall('TreeNumber')
        if (child.find('DescriptorUI').text == parent2):
            tree2 = child.find('TreeNumberList').findall('TreeNumber')

    def child_identifier(parent,child):
        for i in range(len(parent)):
            if (parent[i].text in child):
                return True
        return False

    for child in root:
        if (child.attrib['DescriptorClass']!= '3'):
            tree = child.find('TreeNumberList').findall('TreeNumber')
            tree_text = ''
            for i in range(len(tree)):
                tree_text = tree_text + tree[i].text
            #check if child of both parents
            if (child_identifier(tree1, tree_text) and child_identifier(tree2, tree_text)):
                shared_child.append(child.find('DescriptorName')[0].text)
    return shared_child

shared_child('D007154', 'D009422')
```