Project Summary

Task 1: Document the Data Sources
Task 2: Load Data and Perform Initial Exploration
Task 3: Further Inspect the Data Sets
Task 4: Identify Missing Values
Task 5: Identify Outliers in the Trees Dimensions
Task 6: Identify Duplicates in the Trees Data Set
Task 7: Identify Geolocation Issues
Task 8: Identify Unmatched Data
Task 9: Report back on your process

Trees in Camden Use this notebook to complete your analysis. Enter code and comments after the TODOs.

There are some code cells completed for you. These are highlighted with a TODO comment. You can use these to guide the subsequent tasks. Other cells require you to read documentation or search for answers. The markdown comments give you links to some useful documentation and articles. Read the documentation, look at the examples provided in the documentation and then try to apply them to your data.

Remember that you can find information on the pandas functions on the Pandas website https://pandas.pydata.org/pandas-docs/stable/reference/frame.html or directly in the notebook by puttin a ? before or after the function name. for instance: `?df.head()` or `df['Maturity'].value_counts()?`

Import of Python Libraries

We will import all the python libraries required for the project which will be available for use across the tasks.

`

In [1]:

```
# Data Manipulation
import pandas as pd

# Numerical Operations
import numpy as numpyp

# Data plotting
import matplotlib as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
```

Task 1: Document the Data Sources

> The data source list document has been populated accordingly.

Task 2: Load Data and Perform Initial Exploration

Load the data from the supplied data files. The files are in different **file formats**, but Pandas can handle this.

You should read the data in using the appropriate function:

- pandas.read_excel
- pandas.read_csv
- pandas.read_json

You can then inspect the first few rows of the loaded dataframe:

pandas.DataFrame.head

You can get the number of rows and columns:

pandas.DataFrame.shape

You can get the list of column names:

pandas.DataFrame.columns

And you can list the data types of the columns:

pandas.DataFrames.dtypes

I've done the first one (loading `"camden_trees.xlsx"`) for you. Please load `"camden_trees_environmental.csv"` and `"tree_common_names.json"` and analyse them in the same way.

Trees

The file `"camden_trees.xlsx"` is an Excel file, so we use the `read_excel()` function.

In [2]:
```
# Create a Pandas dataframe called trees that contains the contents of the Excel file
trees = pd.read_excel("camden_trees.xlsx")
```

We can now inspect the first few rows using `head()` . By default, `head()` displays the first 5 rows.

In [3]:
```
# Display the first few rows
trees.head()
```

Out[3]:

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	L
0	00060053	1.0	Russell Nurseries Estate	Housing	Vacant Tree Pit	NaT	NaN	NaN	NaN	NaN	E05000135	Hampstead Town	527305.0	185240.0	-
1	00057855	1.0	BRECKNOCK JMI (E)	Education	Vacant Tree Pit	2019-07-17	2022/2023	NaN	NaN	NaN	E05000131	Cantelowes	529923.0	184782.0	-
2	00059953	1.0	Estate 51 Ravenshaw Street	Housing	Ficus carica	NaT	NaN	5.0	4.0	10.0	NaN	NaN	0.0	0.0	
3	00059915	1.0	ROSARY RC JMI (E)	Education	Betula jacquemontii	NaT	NaN	4.0	1.0	6.0	E05000135	Hampstead Town	527249.0	185261.0	-
4	00010762	1.0	Holly Lodge Estate	Housing	Ilex x altaclarensis	2017-06-14	2020/2021	14.0	6.0	26.0	E05000137	Highgate	528414.0	186770.0	-

It's good to understand the size of the dataset we are dealing with. The `shape` property does this for us.

In [4]:
```
# Get the number of rows and columns
trees.shape
```

Out[4]:
```
(23444, 17)
```

If there are lots of columns we can't always see all of them in the `head()` list above. We can use the `columns` property to get a full list:

In [5]:
```
# Get a list of all the columns in the dataframe
trees.columns
```

Out[5]:
```
Index(['Identifier', 'Number Of Trees', 'Site Name', 'Contract Area',
       'Scientific Name', 'Inspection Date', 'Inspection Due Date',
       'Height In Metres', 'Spread In Metres',
       'Diameter In Centimetres At Breast Height', 'Ward Code', 'Ward Name',
       'Easting', 'Northing', 'Longitude', 'Latitude', 'Location'],
      dtype='object')
```

In order to process the data properly, we should understand the data type for each column. Pandas attempts to work this out for us, but sometimes we need to give it a bit of a hand. We can use the `dtypes` property to list the data types. Note that `object` is Pandas way of saying `string` , i.e. a text data type.

In [6]:
```
# List the data types of each column
trees.dtypes
```

Out[6]:
```
Identifier                                          object
Number Of Trees                                    float64
Site Name                                           object
Contract Area                                       object
Scientific Name                                     object
Inspection Date                             datetime64[ns]
Inspection Due Date                                 object
Height In Metres                                   float64
Spread In Metres                                   float64
Diameter In Centimetres At Breast Height           float64
Ward Code                                           object
Ward Name                                           object
Easting                                            float64
Northing                                           float64
Longitude                                          float64
Latitude                                           float64
Location                                            object
dtype: object
```

Environmental

The file `"camden_trees_environmental.csv"` is a csv file. Use the appropriate function to load it into a Pandas DataFrame.

TODO: Complete the following code cells

In [7]:
```
# Create a Pandas dataframe called trees that contains the contents of the csv file
trees_env=pd.read_csv("camden_trees_environmental.csv")
```

In [8]:
```
# Display the first few rows
trees_env.head()
```

Out[8]:

	Identifier	Maturity	Physiological Condition	Tree Set To Be Removed	Removal Reason	Capital Asset Value For Amenity Trees	Carbon Storage In Kilograms	Gross Carbon Sequestration Per Year In Kilograms	Pollution Removal Per Year In Grams
0	00055125	Juvenile	Good	No	NaN	115.07	1.6	0.5	5.7
1	00059429	Middle aged	Fair	No	NaN	7518.08	NaN	NaN	NaN
2	00018254	Mature	Fair	No	NaN	20419.63	426.4	8.8	215.2
3	00027155	Mature	Fair	No	NaN	21447.74	448.3	9.6	379.1
4	00041326	Juvenile	Good	No	NaN	524.30	9.9	1.4	12.8

In [9]:
```
# Get the number of rows and columns
trees_env.shape
```

Out[9]:
```
(23415, 9)
```

In [10]:
```
# Get a list of all the columns in the dataframe
trees_env.columns
```

Out[10]:
```
Index(['Identifier', 'Maturity', 'Physiological Condition',
       'Tree Set To Be Removed', 'Removal Reason',
       'Capital Asset Value For Amenity Trees', 'Carbon Storage In Kilograms',
       'Gross Carbon Sequestration Per Year In Kilograms',
       'Pollution Removal Per Year In Grams'],
      dtype='object')
```

In [11]:
```
# List the data types of each column
trees_env.dtypes
```

Out[11]:
```
Identifier                                           object
Maturity                                             object
Physiological Condition                              object
Tree Set To Be Removed                               object
Removal Reason                                       object
Capital Asset Value For Amenity Trees               float64
Carbon Storage In Kilograms                         float64
Gross Carbon Sequestration Per Year In Kilograms    float64
Pollution Removal Per Year In Grams                 float64
dtype: object
```

Common and Scientific Names

The file `"tree_common_names.json"` is a json file. Use the appropriate function to load it into a Pandas DataFrame.

TODO: Complete the following code cells

In [12]:
```
# Create a Pandas dataframe called trees that contains the contents of the json file
trees_com_names=pd.read_json("tree_common_names.json")
```

In [13]:
```
# Display the first few rows
trees_com_names.head()
```

Out[13]:

	Scientific Name	Common Name
0	Carpinus betulus Lucas	Hornbeam - European
1	Prunus 'Pandora'	Cherry - Ornamental
2	Tilia unidentified species	Lime
3	Rosa unidentified species	None
4	Cedrus libani	Cedar of Lebanon

In [14]:
```
# Get the number of rows and columns
trees_com_names.shape
```

Out[14]:
```
(589, 2)
```

In [15]:
```
# Get a list of all the columns in the dataframe
trees_com_names.columns
```

Out[15]:
```
Index(['Scientific Name', 'Common Name'], dtype='object')
```

In [16]:
```
# List the data types of each column
trees_com_names.dtypes
```

Out[16]:
```
Scientific Name    object
Common Name        object
dtype: object
```

Review

At the end of this task you should have a good basic understanding of the contents and overall shape of the different data files. If you don't, do back and review the outputs above.

Task 3: Further Inspect the Datasets

The initial inspection gave you a very high-level understanding of the data. We will now drill a bit deeper and try to understand the data column-by-column.

For columns with a string data type (`object` in Pandas) we have **qualitative** data. It would be good to know how many *different* values we have in the column, what those values are and the *count* how many of each different value we have. This will help us understand if the qualitative variable is **binary**, **nominal** or **ordinal**.

For columns with a numeric data type (`int` or `float`) we have **quantitative** data. Usually integer type variables can the thought of as **discrete** and float type variables can be thought of as **continuous**. It would be good to know some summary descriptive statistics for these columns.

If you are unsure of what these different data classifications mean, read this:

Types of Variables

You can get the list of values and counts for a column using this function:

pandas.Series.value_counts

You can get the descriptive statistics for a DataFrame using this function:

pandas.DataFrame.describe

Note that Pandas may treat integer columns as floats if there are null values in the columns. So if you see a float data type it might be worth checking the actual values to confirm if it really is a float or if it really is an int with nulls. You can check the actual values with:

pandas.Series.unique

Further Inspect the Trees Dataset

Let's start with the trees dataset.

Counts of Values for String Types Columns

Go through each column that is a string (object) type and count the number of rows for each value in the column. After each one, classify the data as binary, nominal or ordinal using a markdown comment.

I've done the first one for you.

In [17]:
```
# See the names of columns and data type again
trees.dtypes
```

Out[17]:
```
Identifier                                          object
Number Of Trees                                    float64
Site Name                                           object
Contract Area                                       object
Scientific Name                                     object
Inspection Date                             datetime64[ns]
Inspection Due Date                                 object
Height In Metres                                   float64
Spread In Metres                                   float64
Diameter In Centimetres At Breast Height           float64
Ward Code                                           object
Ward Name                                           object
Easting                                            float64
Northing                                           float64
Longitude                                          float64
Latitude                                           float64
Location                                            object
dtype: object
```

Site Name

In [18]:
```
# List of values in Site Name column and their counts
trees["Site Name"].value_counts()
```

Out[18]:
```
WATERLOW PARK (LS)                       920
Alexandra & Ainsworth Estate             289
Belsize nature reserve, Russell Nursery  278
Holly Lodge Estate                       272
LINCOLN'S INN FIELDS, GARDENS (LS)       193
                                        ...
GOLDINGTON CRESCENT                        1
ALLCROFT ROAD                              1
WOBURN WALK, LAND BEHIND 4-18              1
KILBURN PRIORY                             1
GOODGE PLACE                               1
Name: Site Name, Length: 1135, dtype: int64
```

`Site Name` is **qualitative nominal**.

Now do the same on the other string columns. Use `value_counts()` and then classify as binary, ordered or unordered using a markdown comment. As you do each one, stop and look at the values and counts and think about how the data in the column might be useful for supporting the council's initiatives. Don't just treat this as a mechanical copy/paste task. The objective is, after all, to get really intimate with the data!

TODO: Enter your code below. Use one code cell per column and then add a markdown cell after each one to classify the column as in the above example. Add as many cells as you need.

Contract Area

In [19]:
```
# List of values in Contract Area column and their counts
trees["Contract Area"].value_counts()
```

Out[19]:
```
Highways              10062
Housing                7500
Parks                  4330
Education              1288
Corporate Landlord      264
Name: Contract Area, dtype: int64
```

`Contract Area` is **qualitative nominal**

Scientific Name

In [20]:
```
# List of values in Scientific Name column and their counts
trees["Scientific Name"].value_counts()
```

Out[20]:
```
Platanus x hispanica                                3340
Tilia europaea                                      1468
Acer pseudoplatanus                                  941
Betula pendula                                       765
Fraxinus excelsior                                   754
                                                    ...
Vacant Tree Pit (planned: Populus tremula)             1
Liriodendron fastigiata                                1
Sequoia sempervirens                                   1
Sorbus x hybrida                                       1
Vacant Tree Pit (planned: Acer rubrum 'Amstrong')      1
Name: Scientific Name, Length: 543, dtype: int64
```

`Scientific Name` is **qualitative nominal**

Inspection Due Date

In [21]:
```
# List of values in Inspection Due Date column and their counts
trees["Inspection Due Date"].value_counts()
```

Out[21]:
```
2022/2023    7921
2021/2022    7353
2020/2021    6577
2019/2020    1157
2018/2019      16
2017/2018       5
2016/2017       4
2001/2002       4
2003/2004       2
2006/2007       1
2012/2013       1
2013/2014       1
2011/2012       1
Name: Inspection Due Date, dtype: int64
```

`Inspection Due Date` is **qualitative ordinal**

Ward Code

In [22]:
```
# List of values in Ward Code column and their counts
trees["Ward Code"].value_counts()
```

Out[22]:
```
E05000137    2799
E05000143    1832
E05000134    1541
E05000140    1540
E05000139    1463
E05000136    1424
```

```
E05000135    1340
E05000138    1293
E05000132    1284
E05000131    1231
E05000133    1229
E05000129    1008
E05000142     989
E05000144     978
E05000130     899
E05000145     853
E05000141     824
E05000128     691
Name: Ward Code, dtype: int64
```

`Ward Code` is **qualitative ordinal**

Ward Name

In [23]:
```
# List of values in Ward Name column and their counts
trees["Ward Name"].value_counts()
```

Out[23]:
```
Highgate                          2799
St Pancras and Somers Town        1832
Gospel Oak                        1541
Kilburn                           1540
Kentish Town                      1463
Haverstock                        1424
Hampstead Town                    1340
Holborn and Covent Garden         1293
Fortune Green                     1284
Cantelowes                        1231
Frognal and Fitzjohns             1229
Bloomsbury                        1008
Regent's Park                      989
Swiss Cottage                      978
Camden Town with Primrose Hill     899
West Hampstead                     853
King's Cross                       824
Belsize                            691
Name: Ward Name, dtype: int64
```

`Ward Name` is **qualitative nominal**

Location

In [24]:
```
# List of values in Location column and their counts
trees["Location"].value_counts()
```

Out[24]:
```
(51.556205, -0.173776)    3
(51.553475, -0.152668)    3
(51.548133, -0.144922)    2
(51.544482, -0.144465)    2
(51.55468, -0.164744)     2
                         ..
(51.525312, -0.128846)    1
(51.540297, -0.181512)    1
(51.556013, -0.211326)    1
(51.55969, -0.182457)     1
(51.552397, -0.173397)    1
Name: Location, Length: 23262, dtype: int64
```

`Location` is **qualitative nominal**

Descriptive Stats for Numeric Type Columns

Use the `describe()` function to get the descriptive stats for the numeric columns.

For each column, classify the column as discrete or continuous (use the data type to guide you, but check any floats to confirm whether they are really floats or just ints with null values. Use `pandas.Series.unique()` to check this).

TODO: Complete the following code cells

In [25]:
```
# Get the descriptive stats for the numeric columns
trees.describe().round(2)
```

Out[25]:

	Number Of Trees	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Easting	Northing	Longitude	Latitude
count	23422.00	23006.00	23006.00	23005.00	23444.00	23444.00	23388.00	23388.00
mean	1.10	10.31	6.00	32.60	526762.52	184085.19	-0.16	51.55
std	1.29	6.33	4.13	26.15	25835.86	9121.06	0.03	0.01
min	0.00	0.00	0.00	0.00	0.00	0.00	-0.26	51.51
25%	1.00	5.00	3.00	12.00	526583.75	183665.00	-0.18	51.54
50%	1.00	9.00	5.00	27.00	528456.50	184690.00	-0.15	51.55
75%	1.00	15.00	8.00	46.00	529369.00	185481.00	-0.14	51.55
max	67.00	127.00	88.00	228.00	531514.00	196188.00	-0.11	51.65

In [26]:
```
# Remind ourselves of the data types
trees.dtypes
```

Out[26]:
```
Identifier                                       object
Number Of Trees                                 float64
Site Name                                        object
Contract Area                                    object
Scientific Name                                  object
Inspection Date                          datetime64[ns]
Inspection Due Date                              object
Height In Metres                                float64
Spread In Metres                                float64
Diameter In Centimetres At Breast Height        float64
Ward Code                                        object
Ward Name                                        object
Easting                                         float64
Northing                                        float64
Longitude                                       float64
Latitude                                        float64
Location                                         object
dtype: object
```

Find out if floats are really floats or ints with nulls.

TODO: Enter your code below. Use one code cell per column. Add as many cells as you need.

Number of Trees

In [27]:
```
# Check the datatype of the Number of trees column
trees['Number Of Trees'].dtypes
```

Out[27]:
```
dtype('float64')
```

In [28]:
```
# Check is truly there are float values in columns
trees['Number Of Trees'].unique()
```

Out[28]:
```
array([ 1.,  2.,  3.,  0., nan,  5.,  6.,  7., 18.,  8., 65.,  4., 10.,
        9., 11., 50., 12., 15., 52., 40., 33., 13., 20., 67., 21., 32.,
       24., 26., 16., 25., 51.])
```

> Actual values in `Number of Trees` column are integers

Height In Metres

In [29]:
```
# Check the datatype of the Height In Metres column
trees['Height In Metres'].dtypes
```

Out[29]:
```
dtype('float64')
```

In [30]:
```
# Check is truly there are float values in columns
trees['Height In Metres'].unique()
```

Out[30]:
```
array([  nan,   5. ,   4. ,  14. ,   9. ,   0. ,   2. ,   2.5,   8. ,
        13. ,  17. ,  10. ,   3. ,  19. ,   7. ,   6. ,   1.8,  15. ,
        12. ,   1.5,  16. ,  20. ,  21. ,  24. ,  25. ,   2.7,  18. ,
        11. ,  26. ,  22. ,   0.5,  16.1,  27. ,  28. ,   2.3,  22.3,
        38. ,  15.5,  29. ,  23. ,  34. ,   3.5,   2.6,   1. ,   5.5,
        22.5,   2.2,  36. ,  31. ,  96. ,  13.6, 127. ,  14.7,   4.5,
        30. ,  35. ,  32. ,   9.4,  11.8,  33. ,  40. ,   2.1,   6.5,
        41. ,  23.1,  37. ,  39. ,  12.5,   7.5,  13.7,  31.6,  23.8,
         0.2,  12.3,  15.7,   6.8,   9.3,   3.8,  24.9,  17.4,   2.8,
         3.2,  13.9,  12.4,  10.2,  10.8,  24.5,  30.4,   9.8,  11.4,
        23.4,   3.7,  11.6,   7.7,   8.3,   3.6,  17.5,  19.5])
```

> Actual values in `Height In Metres` column are Floats

Spread In Metres

In [31]:
```
# Check the datatype of the Height In Metres column
trees['Spread In Metres'].dtypes
```

Out[31]:
```
dtype('float64')
```

In [32]:
```
# Check is truly there are float values in columns
trees['Spread In Metres'].unique()
```

Out[32]:
```
array([  nan,  4.  ,  1.  ,  6.  ,  7.  ,  0.  ,  1.5 ,  5.  ,  9.  ,
       10.  ,  3.  ,  2.  , 13.  ,  2.5 , 12.  , 15.  , 14.  ,  8.  ,
       11.  ,  0.6 ,  1.8 , 17.  , 20.  , 18.  , 16.  , 22.  , 19.  ,
        5.5 , 26.  ,  1.4 , 23.  , 21.  , 28.  ,  1.2 ,  0.5 ,  3.5 ,
       24.  ,  1.3 ,  1.6 , 30.  ,  1.7 , 27.  ,  4.2 , 11.02,  0.7 ,
        6.5 ,  4.5 , 31.  , 25.  ,  0.8 ,  0.3 ,  2.2 ,  2.8 ,  1.65,
       29.  , 88.  ])
```

> Actual values in `Spread In Metres` column are Floats

Diameter In Centimetres At Breast Height

In [33]:
```
# Check the datatype of the Height In Metres column
trees['Diameter In Centimetres At Breast Height'].dtypes
```

Out[33]:
```
dtype('float64')
```

In [34]:
```
# Check is truly there are float values in columns
trees['Diameter In Centimetres At Breast Height'].unique()
```

Out[34]:
```
array([  nan,  10. ,   6. ,  26. ,  29. ,   5. ,   0. ,   4. ,  12. ,
        59. ,  52. ,  23. ,  50. ,  63. ,  15. ,   3. ,  49. ,  42. ,
        14. ,  19. ,   9. ,  70. ,  32. ,  28. ,  34. ,  20. ,  17. ,
        27. ,  37. ,   8. ,  45. ,  18. , 119. ,   7. ,  38. ,  55. ,
        41. ,  75. ,  31. ,  25. ,  11. ,  30. ,  43. ,  68. ,  92. ,
        16. ,  35. ,  58. ,  72. ,  64. ,  13. ,  61. ,  69. ,  33. ,
        47. ,  67. , 109. , 106. ,  24. ,  51. ,  40. ,  22. ,  60. ,
        39. ,  46. ,  57. ,  21. ,  54. , 117. ,  44. ,  82. , 114. ,
        65. ,  84. , 118. ,   1. ,  89. ,  53. ,  36. ,  93. ,  88. ,
        86. , 163. ,  66. ,  81. ,  74. , 100. ,  48. ,  73. ,  95. ,
       129. ,  62. ,  85. ,  56. ,   3.5, 161. ,  71. , 105. , 102. ,
        80. ,  87. , 101. ,  76. , 113. , 108. , 160. , 132. ,  90. ,
       145. , 130. ,  79. , 228. , 110. ,  83. ,  78. , 122. , 170. ,
       115. ,   2. ,  77. , 107. ,  96. , 126. ,  91. , 104. , 158. ,
        99. ,  94. ,  16.5, 127. , 151. , 103. , 112. ,  98. ,  97. ,
       136. , 125. , 111. , 124. , 139. , 156. , 120. , 148. , 144. ,
       140. , 121. , 143. ,  17.5, 154. , 159. , 142. , 197. , 123. ,
       149. , 155. , 191. , 131. , 147. , 162. , 116. , 152. , 153. ,
       165. , 137. , 200. , 177. , 133. , 128. , 134. ,  11.5, 135. ,
       150. , 187. , 210. , 166. , 138. ,  10.5, 206. , 141. , 209. ,
       184. , 173. , 192. ,   7.5, 194. , 157. , 146. , 185. ])
```

> Actual values in `Diameter In Centimetres At Breast Height` column are Floats but also contains null values

Easting

In [35]:
```
# Check the datatype of the Easting column
trees['Easting'].dtypes
```

Out[35]:
```
dtype('float64')
```

In [36]:
```
# Check is truly there are float values in columns
trees['Easting'].unique()
```

Out[36]:
```
array([527305., 529923.,      0., ..., 527733., 524398., 525944.])
```

> Values in `Easting column are integers but have a period at the end of the number

Northing

In [37]:
```
# Check the datatype of the Easting column
trees['Northing'].dtypes
```

Out[37]:
```
dtype('float64')
```

In [38]:
```
# Check is truly there are float values in columns
trees['Northing'].unique()
```

Out[38]:
```
array([185240., 184782.,      0., ..., 185755., 187062., 187313.])
```

> Values in `Northing` column are integers but have a period at the end of the number

Longitude

In [39]:
```
# Check the datatype of the Easting column
trees['Longitude'].dtypes
```

Out[39]:
```
dtype('float64')
```

In [40]:
```
# Check is truly there are float values in columns
trees['Longitude'].unique()
```

Out[40]:
```
array([-0.16524 , -0.127681,       nan, ..., -0.196884, -0.204206,
       -0.173397])
```

> Values in `Longitude` column are Floats and have null values as well

Latitude

In [41]:
```
# Check the datatype of the Easting column
trees['Latitude'].dtypes
```

Out[41]:
```
dtype('float64')
```

In [42]:
```
# Check is truly there are float values in columns
trees['Latitude'].unique()
```

Out[42]:
```
array([51.551693, 51.546984,       nan, ..., 51.54329 , 51.545726,
       51.531863])
```

> Values in `Latitude` column are Floats and have null values as well

Classify the columns as discrete or continuous.

TODO: Enter markdown below.

`Number Of Tree` is **discrete - integers.** The datatype is showing as float because of the null values.

`Height In Metres` is **continuos - float.**

`Spread In Metres` is **continuos - float.**

`Diameter In Centimetres At Breast Height` is **continuos - float.**

`Easting` is **continuos - discrete.**

`Northing` is **continuos - discrete.**

`Longitude` is **continuos - float.**

`Latitude` is **continuos - float.**

Further Inspect the Environmental Dataset

Now repeat the above for the environmental dataset.

Counts of Values for String Type Columns

For each string column in the environmental dataset show the counts of the unique values.

TODO: Enter your code below. Use one code cell per column and then add a markdown cell after each one to classify the column. Add as many cells as you need.

In [43]:
```
# Confirm the names of the columns and data type
trees_env.dtypes
```

Out[43]:
```
Identifier                                       object
Maturity                                         object
Physiological Condition                          object
Tree Set To Be Removed                           object
Removal Reason                                   object
Capital Asset Value For Amenity Trees           float64
Carbon Storage In Kilograms                     float64
Gross Carbon Sequestration Per Year In Kilograms float64
Pollution Removal Per Year In Grams             float64
dtype: object
```

We will go through each column that is a string (object) type and count the number of rows for each value in the column and the classify as binary, nominal or ordinal

Maturity

In [44]:
```
# List of values in maturity column and their counts
trees_env["Maturity"].value_counts()
```

Out[44]:
```
Mature            10225
Middle aged        7779
Juvenile           4393
Not Applicable      377
Over Mature         191
Veteran              41
Name: Maturity, dtype: int64
```

`Maturity` is **qualitative ordinal**

Physiological Condition

In [45]:
```
# List of values in maturity column and their counts
trees_env["Physiological Condition"].value_counts()
```

Out[45]:
```
Good              12910
Fair               9183
Poor                357
Not applicable      249
Dead                236
Excellent             8
Name: Physiological Condition, dtype: int64
```

`Physiological Condition` is **qualitative ordinal**

Tree Set To Be Removed

In [46]:
```
# List of values in maturity column and their counts
trees_env["Tree Set To Be Removed"].value_counts()
```

Out[46]:
```
No     23331
Yes       84
Name: Tree Set To Be Removed, dtype: int64
```

`Tree Set To Be Removed` is **qualitative binary**

Removal Reason

In [47]:
```
# List of values in maturity column and their counts
trees_env["Removal Reason"].value_counts()
```

Out[47]:
```
Dead, dying                     30
Basal decay                     17
Trunk decay                     10
Tree defect                      5
Crown die-back                   5
Dog damage                       3
Unsuitable location              3
Newly planted tree failure       2
Coppiced stump                   1
Touching building/structure      1
Crown decay                      1
Split trunk                      1
Broken/split branch              1
Climber                          1
No defects - work required       1
Suppressed                       1
ATRD                             1
Name: Removal Reason, dtype: int64
```

`Removal Reason` is **qualitative nominal**

Descriptive Stats for Numeric Type Columns

For each numeric column in the environmental dataset show the descriptive stats

TODO: Complete the following code cells

In [48]:
```
# Get the descriptive stats for all numeric columns
trees_env.describe().round(2)
```

Out[48]:

	Capital Asset Value For Amenity Trees	Carbon Storage In Kilograms	Gross Carbon Sequestration Per Year In Kilograms	Pollution Removal Per Year In Grams
count	22982.00	20555.00	20555.00	20555.00
mean	14056.39	467.47	8.68	217.74
std	24803.81	844.93	8.68	306.75
min	0.00	0.50	0.00	0.30
25%	1035.65	24.80	2.20	29.30
50%	5443.66	163.90	6.10	108.10
75%	16781.42	497.30	11.70	297.60
max	504725.72	6000.00	53.80	8223.70

In [49]:
```
# Remind ourselves of the data types
trees_env.dtypes
```

Out[49]:
```
Identifier                                          object
Maturity                                            object
Physiological Condition                             object
Tree Set To Be Removed                              object
Removal Reason                                      object
Capital Asset Value For Amenity Trees              float64
Carbon Storage In Kilograms                        float64
Gross Carbon Sequestration Per Year In Kilograms   float64
Pollution Removal Per Year In Grams                float64
dtype: object
```

Find out if floats are really floats or ints with nulls.

TODO: Enter your code below. Use one code cell per column. Add as many cells as you need.

Capital Asset Value For Amenity Trees

In [50]:
```
# Check the datatype of the Easting column
trees_env['Capital Asset Value For Amenity Trees'].dtypes
```

Out[50]:
```
dtype('float64')
```

In [51]:
```
# Check is truly there are float values in columns
trees_env['Capital Asset Value For Amenity Trees'].unique()
```

Out[51]:
```
array([1.1507000e+02, 7.5180800e+03, 2.0419630e+04, ..., 3.3664130e+04,
       3.6269450e+04, 1.4801215e+05])
```

> Values in `Capital Asset Value For Amenity Trees` column are floats

Carbon Storage In Kilograms

In [52]:
```
# Check the datatype of the Easting column
trees_env['Carbon Storage In Kilograms'].dtypes
```

Out[52]:
```
dtype('float64')
```

In [53]:
```
# Check is truly there are float values in columns
trees_env['Carbon Storage In Kilograms'].unique()
```

Out[53]:
```
array([1.6000e+00,        nan, 4.2640e+02, ..., 4.7233e+03, 3.7305e+03,
       4.8100e+02])
```

> Values in `Carbon Storage In Kilograms` column are floats and it has null values as well

Gross Carbon Sequestration Per Year In Kilograms

In [54]:
```
# Check the datatype of the Easting column
trees_env['Gross Carbon Sequestration Per Year In Kilograms'].dtypes
```

Out[54]:
```
dtype('float64')
```

In [55]:
```
# Check is truly there are float values in columns
trees_env['Gross Carbon Sequestration Per Year In Kilograms'].unique()
```

Out[55]:
```
array([ 0.5,  nan,  8.8,  9.6,  1.4, 10.1,  0.8,  7.9, 24.2,  2.4,  0.9,
        3. , 14.6,  8.1,  4.1,  1.8, 18.8, 24.4, 13.4,  4.3,  6.5,  1.3,
        0.6,  2.5, 13.1,  2.1,  3.9, 15.9,  1. ,  6.6,  0.4,  3.4,  2.9,
       28.7,  0.7,  1.5, 12.5, 25.4, 11.2,  9. , 23.9,  4.4, 11.5, 11. ,
        8.4, 10.9, 10.7,  6.9,  7. ,  3.8,  3.2,  6.1,  8.6, 30.2,  3.7,
       15. , 30.4,  7.6, 20.1, 10.2,  8.3, 39.9,  0.3,  4.9, 14.3, 13.5,
       16.7,  8.9,  1.6,  4.2,  3.6,  4. ,  6.7,  0.1, 19.7, 24.6,  6.4,
        5.4,  5.9, 12.2,  7.3, 13. ,  7.1, 36.9,  9.3, 18.2, 10. ,  4.7,
        5. , 29.8, 17.8, 18.5, 17.6,  7.2,  4.8,  5.6,  5.3, 10.5, 12.9,
        9.4,  7.8,  1.1, 19.2, 37.3,  2.7, 12.8, 17. , 15.5, 27.6, 34.2,
        5.7,  5.8, 17.3, 20.4,  9.9, 15.6,  7.4, 11.8,  9.2,  2.6, 21.7,
       11.3, 10.8, 29.4, 23.1, 26.5,  1.2, 10.6, 33.6, 23.6, 11.7, 17.4,
       15.1, 16.6,  2.3, 13.3, 16.3, 18.3, 10.4, 19. ,  1.7,  3.3, 14.1,
        5.5, 12.6, 14.5,  3.5,  8.7, 36.6, 17.7,  5.1, 21.1, 20.2,  7.7,
       15.2,  7.5, 30.5, 13.8,  4.6, 39.2, 37.8, 13.2, 51. , 11.6,  6.8,
       21. ,  5.2,  8. ,  4.5, 15.3, 13.7, 18. ,  6.3,  2. ,  2.8,  9.1,
        6. , 12.4, 11.4, 18.6,  1.9, 23.5, 18.9, 26.2, 12.1, 41.7,  2.2,
       14.7, 14.4,  9.7, 23. , 22.1, 15.7, 12. , 14.8, 19.9, 29.7, 11.1,
       19.6, 12.7, 31.9, 33.2, 19.1,  6.2, 28.4, 27. , 17.5, 16.4, 27.7,
       38.4, 43.9, 35.4,  8.2, 37.1, 29.5, 22.2, 17.1, 26.3, 21.2, 20.5,
       11.9, 31.3, 14.2,  8.5, 26.8,  0.2, 16.8, 35.6, 20.8, 22.3, 19.4,
       20. , 14. , 36.1, 20.9, 15.8, 14.9,  9.5, 28.3, 16.9,  3.1, 37.6,
       34.7, 17.9, 35.3,  9.8, 16.5, 21.4, 19.3, 28.5, 29. , 33.5, 40.7,
       28.6, 13.6, 27.1, 26.7, 13.9, 21.3, 23.8, 35.1, 40.5, 32.9, 18.1,
       37. , 20.6, 35.2, 12.3, 41.8, 22.7, 21.5, 35.7, 32.4, 16.1, 31.1,
       25.3, 26. , 23.4, 30.1, 42.1, 36.5, 10.3, 44.3, 16. , 23.2, 28.8,
       21.9, 32. , 30.3, 15.4, 24.3, 26.9, 25.8, 28.2, 22.8, 42.3, 33.9,
       25.6, 38.6, 18.4, 31.2, 32.7, 28. , 30.6, 42. , 27.3, 33.8, 21.6,
       33.3, 35.9, 24. , 40.6, 42.7, 31.5, 22.5, 20.7, 33.1, 35.5, 27.5,
       24.8, 26.4, 22.6, 18.7, 32.6, 36.4, 16.2, 38.5, 22.4, 22.9, 29.2,
       25.9, 19.5, 24.5, 20.3, 27.9, 29.6, 25. , 27.4, 30.7, 25.1, 34.6,
       38. , 34.4, 36.7, 34.1, 32.5, 27.8, 32.8, 43.1, 17.2, 37.5, 39.3,
       29.1, 30.8, 34.8, 29.9, 38.3, 23.3, 21.8, 41.3, 38.7, 36.3, 30. ,
       41.5, 37.2, 24.9, 19.8, 28.9, 42.2, 39.5, 31.8, 37.4, 24.1,  0. ,
       37.7, 32.3, 36.8, 27.2, 24.7, 36.2, 39.7, 42.9, 49.5, 31.6, 38.9,
       40.9, 44.2, 32.1, 42.5, 25.7, 35.8, 38.2, 40.8, 35. , 34. , 31.7,
       40.1, 26.1, 25.2, 39. , 41.1, 33.4, 32.2, 43.3, 33.7, 26.6, 39.1,
       41.4, 43.2, 31. , 41.9, 41. , 42.4, 43. , 40.3, 40.2, 29.3, 31.4,
       22. , 42.8, 43.7, 23.7, 34.5, 41.6, 39.6, 39.8, 37.9, 43.8, 42.6,
       41.2, 43.6, 36. , 39.4, 43.5, 25.5, 38.1, 34.9, 28.1, 43.4, 53.8,
       30.9, 40.4, 34.3, 45.5, 52.7, 40. , 38.8, 44.9, 33. ])
```

> Values in `Gross Carbon Sequestration Per Year In Kilograms` column are floats and it has null values as well

Pollution Removal Per Year In Grams

In [56]:
```
# Check the datatype of the Easting column
trees_env['Pollution Removal Per Year In Grams'].dtypes
```

Out[56]:
```
dtype('float64')
```

In [57]:
```
# Check is truly there are float values in columns
trees_env['Pollution Removal Per Year In Grams'].unique()
```

Out[57]:
```
array([  5.7,   nan, 215.2, ...,   8. , 399.9,  60.1])
```

> Values in `Pollution Removal Per Year In Grams` column are floats and it has null values as well

---

Classify the columns as discrete or continuous.

TODO: Enter markdown below.

`Capital Asset Value For Amenity Trees` is **continuos - float.**

`Carbon Storage In Kilograms` is **continuos - float.**

`Gross Carbon Sequestration Per Year In Kilograms` is **continuos - float.**

`Pollution Removal Per Year In Grams` is **continuos - float.**

Further Inspect the Common Names Dataset

Now repeat the above for the common names dataset.

(Names) Counts of Values for String Type Columns

For each string column in the common names dataset show the counts of the unique values.

TODO: Enter your code below. Use one code cell per column and then add a markdown cell after each one to classify the column. Add as many cells as you need.

In [58]:
```
# Confirm the names of the columns and data type
trees_com_names.dtypes
```

Out[58]:
```
Scientific Name    object
Common Name        object
dtype: object
```

We will go through each column that is a string (object) type and count the number of rows for each value in the column and the classify as binary, nominal or ordinal

Scientific Name

In [59]:
```
# List of values in maturity column and their counts
trees_com_names["Scientific Name"].value_counts()
```

Out[59]:
```
Cupressocyparis leylandii                        2
Larix decidua                                    2
Salix fragilis                                   2
Alnus cordata                                    2
Populus nigra                                    2
                                                ..
Pyrus salicifolia 'Pendula'                      1
Chamaecyparis lawsoniana 'unid                   1
Platanus x hispanica Tremonia                    1
Vacant Tree Pit (planned: Gymnocladus dioicus)   1
Vacant Tree Pit (planned: Liquidambar styraciflua)   1
Name: Scientific Name, Length: 560, dtype: int64
```

`Scientific Name` is **qualitative ordinal**

Common Name

In [60]:
```
# List of values in maturity column and their counts
trees_com_names["Common Name"].value_counts()
```

Out[60]:
```
Cherry                        12
Rowan                         10
Magnolia                      10
Vacant Tree Pit (planned: )   10
Apple - Crab                   9
                              ..
Pittosporum                    1
Birch - Purple                 1
Maple - Column Norway          1
Maple - Crimson King Norway    1
Castlewellan gold              1
Name: Common Name, Length: 431, dtype: int64
```

`Common Name` is **qualitative ordinal**

(Names) Descriptive Stats for Numeric Type Columns

There are no numeric columns.

Task 4: Identify Missing Values

Find the number of missing values in each column. Missing values can indicate data quality issues. Missing are nulls in our data. But sometimes zero values indicate missing values. For example, a zero value for a tree height is clearly not a valid value, so should be considered missing.

Use these functions to find rows that have missing and zero values:

- pandas.DataFrame.isnull
- pandas.DataFrame.isin
- pandas.DataFrame.mean
- pandas.DataFrame.sum

As you go through this task, think about the possible impact of the missing values on the ability of the data to deliver on the council's initiatives. There is no absolute answer to "how many missing values is too many". It depends on the context of what you intend to do with the data. Try to make an interpretation based on your understanding of the requirements.

Missing Values for the Trees Dataset

I've shown you how to do this for the trees dataset.

In [61]:
```
# Percentage of null values
trees.isnull().mean()*100
```

Out[61]:
```
Identifier                                  0.000000
Number Of Trees                             0.093841
Site Name                                   0.000000
Contract Area                               0.000000
Scientific Name                             0.000000
Inspection Date                             1.710459
Inspection Due Date                         1.710459
Height In Metres                            1.868282
Spread In Metres                            1.868282
Diameter In Centimetres At Breast Height    1.872547
Ward Code                                   0.963999
Ward Name                                   0.963999
Easting                                     0.000000
Northing                                    0.000000
Longitude                                   0.238867
Latitude                                    0.238867
Location                                    0.238867
dtype: float64
```

In [62]:
```
# Number of null values
trees.isnull().sum()
```

Out[62]:
```
Identifier                                    0
Number Of Trees                              22
Site Name                                     0
Contract Area                                 0
Scientific Name                               0
Inspection Date                             401
Inspection Due Date                         401
Height In Metres                            438
Spread In Metres                            438
Diameter In Centimetres At Breast Height    439
Ward Code                                   226
Ward Name                                   226
Easting                                       0
Northing                                      0
Longitude                                    56
Latitude                                     56
Location                                     56
dtype: int64
```

In [63]:
```
# Percentage of zero values
trees.isin([0]).mean()*100
```

Out[63]:
```
Identifier                                  0.000000
Number Of Trees                             0.396690
Site Name                                   0.000000
Contract Area                               0.000000
Scientific Name                             0.000000
Inspection Date                             0.000000
Inspection Due Date                         0.000000
Height In Metres                            0.733663
Spread In Metres                            1.181539
Diameter In Centimetres At Breast Height    1.164477
Ward Code                                   0.000000
Ward Name                                   0.000000
Easting                                     0.238867
```

```
Northing                                    0.238867
Longitude                                   0.000000
Latitude                                    0.000000
Location                                    0.000000
dtype: float64
```

In [64]:
```
# Number of zero values
trees.isin([0]).sum()
```

Out[64]:
```
Identifier                                   0
Number Of Trees                             93
Site Name                                    0
Contract Area                                0
Scientific Name                              0
Inspection Date                              0
Inspection Due Date                          0
Height In Metres                           172
Spread In Metres                           277
Diameter In Centimetres At Breast Height   273
Ward Code                                    0
Ward Name                                    0
Easting                                     56
Northing                                    56
Longitude                                    0
Latitude                                     0
Location                                     0
dtype: int64
```

In [65]:
```
# Percentage of null and zero values
((trees.isnull().sum() + trees.isin([0]).sum())/trees.shape[0])*100
```

Out[65]:
```
Identifier                                  0.000000
Number Of Trees                             0.490531
Site Name                                   0.000000
Contract Area                               0.000000
Scientific Name                             0.000000
Inspection Date                             1.710459
Inspection Due Date                         1.710459
Height In Metres                            2.601945
Spread In Metres                            3.049821
Diameter In Centimetres At Breast Height    3.037024
Ward Code                                   0.963999
Ward Name                                   0.963999
Easting                                     0.238867
Northing                                    0.238867
Longitude                                   0.238867
Latitude                                    0.238867
Location                                    0.238867
dtype: float64
```

In [66]:
```
# Number of null and zero values
(trees.isnull().sum() + trees.isin([0]).sum())
```

Out[66]:
```
Identifier                                   0
Number Of Trees                            115
Site Name                                    0
Contract Area                                0
Scientific Name                              0
Inspection Date                            401
Inspection Due Date                        401
Height In Metres                           610
Spread In Metres                           715
Diameter In Centimetres At Breast Height   712
Ward Code                                  226
Ward Name                                  226
Easting                                     56
Northing                                    56
Longitude                                   56
Latitude                                    56
Location                                    56
dtype: int64
```

Missing Values for the Environmental Dataset

Now repeat the missing values check for the environmental dataset.

TODO: Complete the following code cells

In [67]:
```
# Percentage of null values
trees_env.isnull().mean()*100
```

Out[67]:
```
Identifier                                           0.000000
Maturity                                             1.746744
Physiological Condition                              2.015802
Tree Set To Be Removed                               0.000000
Removal Reason                                      99.641256
Capital Asset Value For Amenity Trees                1.849242
Carbon Storage In Kilograms                         12.214392
Gross Carbon Sequestration Per Year In Kilograms    12.214392
Pollution Removal Per Year In Grams                 12.214392
dtype: float64
```

In [68]:
```
# Number of null values
trees_env.isnull().sum()
```

Out[68]:
```
Identifier                                        0
Maturity                                        409
Physiological Condition                         472
Tree Set To Be Removed                            0
Removal Reason                                23331
Capital Asset Value For Amenity Trees           433
Carbon Storage In Kilograms                    2860
Gross Carbon Sequestration Per Year In Kilograms    2860
Pollution Removal Per Year In Grams            2860
dtype: int64
```

In [69]:
```
# Percentage of zero values
trees_env.isin([0]).mean()*100
```

Out[69]:
```
Identifier                                       0.000000
Maturity                                         0.000000
Physiological Condition                          0.000000
Tree Set To Be Removed                           0.000000
Removal Reason                                   0.000000
Capital Asset Value For Amenity Trees            1.183002
Carbon Storage In Kilograms                      0.000000
Gross Carbon Sequestration Per Year In Kilograms    0.025625
Pollution Removal Per Year In Grams              0.000000
dtype: float64
```

In [70]:
```
# Number of zero values
trees_env.isin([0]).sum()
```

Out[70]:
```
Identifier                                        0
Maturity                                          0
Physiological Condition                           0
Tree Set To Be Removed                            0
Removal Reason                                    0
Capital Asset Value For Amenity Trees           277
Carbon Storage In Kilograms                       0
Gross Carbon Sequestration Per Year In Kilograms    6
Pollution Removal Per Year In Grams               0
dtype: int64
```

In [71]:
```
# Percentage of null and zero values
((trees_env.isnull().sum() + trees_env.isin([0]).sum())/trees.shape[0])*100
```

Out[71]:
```
Identifier                                        0.000000
Maturity                                          1.744583
Physiological Condition                           2.013308
Tree Set To Be Removed                            0.000000
Removal Reason                                   99.518000
Capital Asset Value For Amenity Trees             3.028493
Carbon Storage In Kilograms                      12.199283
Gross Carbon Sequestration Per Year In Kilograms    12.224876
Pollution Removal Per Year In Grams              12.199283
dtype: float64
```

In [72]:
```
# Number of null and zero values
(trees_env.isnull().sum() + trees_env.isin([0]).sum())
```

Out[72]:
```
Identifier                                        0
Maturity                                        409
Physiological Condition                         472
Tree Set To Be Removed                            0
Removal Reason                                23331
Capital Asset Value For Amenity Trees           710
Carbon Storage In Kilograms                    2860
Gross Carbon Sequestration Per Year In Kilograms    2866
Pollution Removal Per Year In Grams            2860
dtype: int64
```

Missing Values for the Common Names Dataset

Now repeat the missing values check for the common names dataset.

TODO: Enter your code below. Add as many cells as you need.

In [73]:
```
# Percentage of null values
trees_com_names.isnull().mean()*100
```

Out[73]:
```
Scientific Name    0.000000
Common Name        4.074703
dtype: float64
```

In [74]:
```
# Number of null values
trees_com_names.isnull().sum()
```

Out[74]:
```
Scientific Name     0
Common Name        24
```

```
dtype: int64
```

In [75]:

```
# Percentage of zero values
trees_com_names.isin([0]).mean()*100
```

Out[75]:

```
Scientific Name    0.0
Common Name        0.0
dtype: float64
```

In [76]:

```
# Number of zero values
trees_com_names.isin([0]).sum()
```

Out[76]:

```
Scientific Name    0
Common Name        0
dtype: int64
```

In [77]:

```
# Percentage of null and zero values
((trees_com_names.isnull().sum() + trees_com_names.isin([0]).sum())/trees.shape[0])*100
```

Out[77]:

```
Scientific Name    0.000000
Common Name        0.102372
dtype: float64
```

In [78]:

```
# Number of null and zero values
(trees_com_names.isnull().sum() + trees_com_names.isin([0]).sum())
```

Out[78]:

```
Scientific Name     0
Common Name        24
dtype: int64
```

Observations

TODO: Write down your observation about the state of missing values below and comment on the extent to which this might impact the ability to deliver on the council's initiatives.

The missing information in the dataset would affect its ability to deliver its initiative of providing information about the trees in the council for the public. The integrity of the data and by extention the council will be called to question</br></br> **Trees Dataset**</br> The missing values in the `Number of Trees`, `Height in Metres`, `Spread In Metres``Diameter In Centimetres At Breast Height` may be the most significant. A site missing this information would not benefit the project. The amount of these being about or less than 3% of the total dataset might not be too significant not to be able to leave them out entirely from the project. Other missing values in `Ward Code` `Ward Name` `Easting` `Northing` `Longitude` `Latitude` `Location` would also affect delivering on the Tree Walk Brochure objectives to map the location of the trees. However, the missing values are less than 1% of the dataset so can be left out entirely.</br>

Environmental Dataset</br> There is an almost 100% missing values in the `Removal Reason` column so this will not be useful at all. Apart from `Carbon Storage In Kilograms`, `Gross Carbon Sequestration Per Year In Kilograms` and `Pollution Removal Per Year In Grams` which have missing values up to 12% , other missing values are 2% or less which can be simply removed. If there are other information about the location of the trees maybe not having the environmental information will not be a show stopper.However, efforts should be made to fill this gap</br> `Trees to be Removed` have 84 yes entries which matches the `Reason for Removal` value counts however there is 99% missing values in `Reason for Removal` thus, there is need to fill the missing values with e.g. NOT APPLICABLE.

Common Names Dataset</br> Trees with missing values in `Common Names` column are 0.12% of the total trees which can be easily removed to clean the dataset. Without the common names of the trees people might find it hard to relate to the information provided since these are the known names of the trees.

Task 5: Identify Outliers in the Trees Dimensions

Outliers are values that are so unusual they are possibly incorrect! We can use a boxplot to show the spread of data and any outliers. Read the following section if you are unfamiliar with them:

- Box plots

Any circles represent what the boxplot considers outliers, but some of these might just be correct but extreme values. We want to only highlight really *crazy* values which are clearly incorrect.

We can use this function to draw boxplots:

- pandas.DataFrame.boxplot

Once we have found if there are outliers, it would be nice to show the rows containing the outliers. The technique for filtering Pandas DataFrames is described here:

- Filtering Pandas DataFrames

The filtering technique creates a mask of rows that we want to select, e.g:

```
mask = df['mycolumn'] > 500
```

and then uses the mask to select rows:

```
df[mask]
```

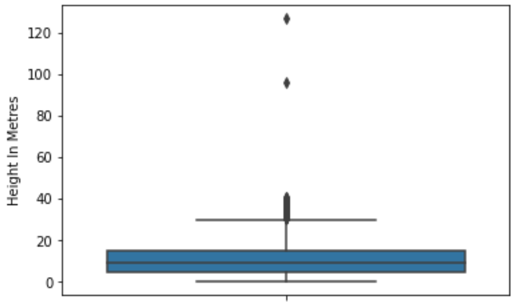
Note that there is no absolute definition of what "crazy" means here. You will need to make some judgements based on your understanding of the world (or specifically the world of trees in Camden!).

Outliers for Height

Find the outliers in the tree height column.

TODO: Complete the following code cells

In [79]:
```
# Use a boxplot to find the outliers
sns.boxplot(data=trees, y=trees["Height In Metres"]);
```



In [80]:
```
# Select the crazy outlier rows
crazy_height_rows= trees[trees['Height In Metres']>50]
```

In [81]:
```
crazy_height_rows
```

Out[81]:

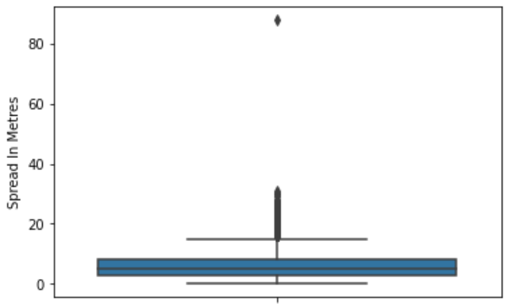
	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	I
1356	00001547	1.0	Beaumont Walk Estate	Housing	Robinia pseudoacacia	2017-05-23	2020/2021	96.0	10.0	63.0	E05000136	Haverstock	527847.0	184391.0	
1863	00013862	1.0	Maitland Park Estate 1	Housing	Prunus avium	2017-05-16	2020/2021	127.0	9.0	34.0	E05000136	Haverstock	527987.0	184901.0	

Outliers for Spread

Now repeat the analysis for spread.

TODO: Complete the following code cells

In [82]:
```
# Use a boxplot to find the outliers
sns.boxplot(data=trees, y=trees["Spread In Metres"]);
```



In [83]:
```
# Select the crazy outlier rows
crazy_spread_rows= trees[trees['Spread In Metres']>35]
```

In [84]:
```
crazy_spread_rows
```

Out[84]:

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Longi
18567	00045515	1.0	Broadfield Estate 1	Housing	Quercus robur	2018-04-26	2021/2022	8.0	88.0	17.0	E05000144	Swiss Cottage	525993.0	184693.0	-0.18

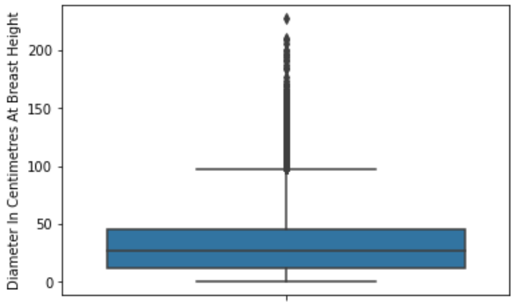
Outliers for Diameter

Now repeat the analysis for diameter.

TODO: Complete the following code cells

In [85]:

```
# Use a boxplot to find the outliers
sns.boxplot(data=trees, y=trees["Diameter In Centimetres At Breast Height"]);
```



In [86]:

```
# Select the outlier rows
crazy_diameter_rows= trees[trees['Diameter In Centimetres At Breast Height']>200]
```

In [87]:

```
crazy_diameter_rows
```

Out[87]:

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Lon
1157	00004100	1.0	LONGFORD STREET, CLARENCE GDNS (LS)	Parks	Platanus x hispanica	2018-03-29	2020/2021	21.0	20.0	228.0	E05000142	Regent's Park	528931.0	182624.0	-0.
11860	00012885	1.0	LINCOLN'S INN FIELDS, GARDENS (LS)	Parks	Platanus x hispanica	2018-04-20	2021/2022	24.0	24.0	210.0	E05000138	Holborn and Covent Garden	530820.0	181358.0	-0.
14178	00012891	1.0	LINCOLN'S INN FIELDS, GARDENS (LS)	Parks	Platanus x hispanica	2018-04-20	2021/2022	30.0	20.0	206.0	E05000138	Holborn and Covent Garden	530783.0	181341.0	-0.
15853	00012939	1.0	LINCOLN'S INN FIELDS, GARDENS (LS)	Parks	Platanus x hispanica	2018-04-19	2021/2022	23.0	20.0	209.0	E05000138	Holborn and Covent Garden	530705.0	181373.0	-0.

Observations

TODO: Write down your observation about outliers in the data. What assumptions did you make? Were you comfortable making these assumptions?

Height in Metres There were two groups of outliers. One clusters had more values which were very close together and nearer the maximum value. The assumption was to retain these values but select the other group having a very big value away from the maximum value (Q3 +1.5IQR) as the outlier.

Spread in Metres
There was also two sets of outliers but one value stood out as it was very far from the maximum value. Others were many and closer to the maximum value so they were retain and the value above 35 assumed as the outlier.

Diameter In Centimetres At Breast Height

There are a number of values which are close together that above 100 which is the maximum value (Q3 +1.5IQR).However upon a manual review of the data we see that it is a particular type of tree that is majorly concerned-Platanus x hispanica, infact this is the trees with the highest counts. It is thus assumed that the wide diameter is common characteristic of this type of tree so will not be treated as an outlier when carrying out further analysis.

Task 6: Identify Duplicates in the Trees Dataset

Sometimes data has duplicate entries. This is another sign of data quality issues!

Find Duplicate Rows

In our dataset the `Identifier` column should be unique. Find out if it is! We've already used a function that can count how many times each value in a column exists. Use is to see if we have duplicates in the trees Dataframe.

TODO: Complete the following code cells

In [88]:

```
# Find out if we have any duplicates
trees['Identifier'].value_counts()
```

Out[88]:

```
00000999    2
00060087    2
00022744    2
00032549    2
00022674    2
           ..
00046158    1
00058373    1
00059181    1
00002274    1
00013369    1
Name: Identifier, Length: 23438, dtype: int64
```

Now see if you can select the rows from trees DataFrame that are duplicates. You will need to use the output from the cell above and use it to filter the trees dataframe.

In [89]:

```
# Select the rows that are duplicated by filtering with value_counts()
dup_count=trees['Identifier'].value_counts()
dup_count[dup_count>=2]
```

Out[89]:

```
00000999    2
00060087    2
00022744    2
00032549    2
00022674    2
00060088    2
Name: Identifier, dtype: int64
```

In [90]:

```
# Select the rows that are duplicated using Pandas duplicated() method
trees[trees.duplicated(subset='Identifier')]
```

Out[90]:

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northin
6111	00032549	1.0	NARCISSUS ROAD	Highways	Betula albosinensis Fasc.	2018-09-19	2021/2022	9.0	6.0	19.0	E05000145	West Hampstead	525185.0	185127.
9186	00022744	1.0	YORK WAY	Highways	Ailanthus altissima	2019-10-30	2022/2023	7.5	3.0	18.0	E05000131	Cantelowes	529983.0	184724.
10972	00060088	1.0	FREDERICK STREET	Highways	Vacant Tree Pit (planned: Acer campestre eco s...	2019-11-09	2022/2023	NaN	NaN	NaN	E05000141	King's Cross	530770.0	182696.
13098	00000999	1.0	ALMA STREET	Highways	Sorbus hupehensis	2017-07-25	2020/2021	5.0	4.0	18.0	E05000139	Kentish Town	528834.0	184856.
13628	00022674	1.0	WOODSOME ROAD	Highways	Sorbus	2017-10-07	2020/2021	7.0	6.0	28.0	E05000137	Highgate	528515.0	186109.
15653	00060087	1.0	ARGYLE SQUARE	Highways	Vacant Tree Pit (planned: Acer campestre eco s...	2019-11-09	2022/2023	NaN	NaN	NaN	E05000141	King's Cross	530342.0	182839.

Observations

TODO: Write down your observations about duplicates in trees.

There are 6 trees duplicated in the trees dataset.

Task 7: Identify Geolocation Issues

The geographic coordinates (Easting and Northing) can be used to plot the trees on a map. We can use this approach to see if there are any unusual tree locations!

We will make a copy of the original trees dataset and remove any rows that have a missing easting or northing as these can't be plotted on the map.

We can copy the DataFrame using:

- pandas.DataFrame.copy

We can use the DataFrame filtering technique we saw before to remove the missing values. E.g. the following code filters out rows where the value for 'mycolumn' is 100:

```
mask = df['mycolumn'] != 100
df = df[mask]
```

You can also create masks using a function, e.g. this creates a mask which excludes nulls:

```
mask = df['mycolumn'].isnull()
```

We can use this function to plot the trees on a map. Set x to "Easting" and y to "Northing" and set a figsize parameter to (6, 6) to get a square aspect ratio:

- pandas.DataFrame.plot.scatter

Remove Trees with Missing Geo-coordinates

Check if there are any rows with null or 0 geo-coordinates. If there are, remove them as we can't plot these.

TODO: Complete the following code cells

In [91]:
```
# Make a copy of the trees
geotrees = trees.copy()
```

In [92]:
```
geotrees.isnull().sum()
```

Out[92]:
```
Identifier                                0
Number Of Trees                          22
Site Name                                 0
Contract Area                             0
Scientific Name                           0
Inspection Date                         401
Inspection Due Date                     401
Height In Metres                        438
Spread In Metres                        438
Diameter In Centimetres At Breast Height    439
Ward Code                               226
Ward Name                               226
Easting                                   0
Northing                                  0
Longitude                                56
Latitude                                 56
Location                                 56
dtype: int64
```

There are no nulls in the easting column

In [93]:
```
# Remove null eastings
geotrees[geotrees['Easting'].isnull()] #confirms there is no null in easting column
```

Out[93]:

Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Longitude	Latitude	l

In [94]:
```
# Remove 0 Eastings.
geotrees[geotrees['Easting']==0] # confirms there are easting with 0 values
```

Out[94]:

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Lor
2	00059953	1.0	Estate 51 Ravenshaw Street	Housing	Ficus carica	NaT	NaN	5.0	4.0	10.0	NaN	NaN	0.0	0.0	
7	00060359	1.0	TOTTENHAM COURT ROAD	Highways	Vacant Tree Pit (planned: Acer campestre)	NaT	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	
21	00060252	1.0	KILBURN GRANGE, MESSINA AVE (LS)	Parks	Vacant Tree Pit	NaT	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	
24	00058873	1.0	GREENAWAY GARDENS	Highways	Liquidambar styraciflua	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
29	00059136	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
38	00059145	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
45	00060407	1.0	Bells Hill Estate	Housing	Tilia cordata	NaT	NaN	5.0	3.0	10.0	NaN	NaN	0.0	0.0	
52	00059138	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
64	00059380	0.0	HAVERSTOCK HILL (PRIV)	Highways	Magnolia unidentified species	2019-05-03	2021/2022	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
65	00060385	1.0	Broadfield Estate 2	Housing	Sambucus nigra	NaT	NaN	4.0	4.0	15.0	NaN	NaN	0.0	0.0	
67	00059372	0.0	FORTESS ROAD	Highways	Malus unidentified species	2019-04-03	2021/2022	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
71	00059142	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	15.0	3.0	NaN	NaN	0.0	0.0	
73	00059698	1.0	Estate 11-15 Parsifal Road (odd)	Housing	Acer pseudoplatanus	NaT	NaN	4.0	0.0	0.0	NaN	NaN	0.0	0.0	
74	00059542	1.0	GROVE THE	Highways	Aesculus hippocastanum	NaT	NaN	16.0	14.0	119.0	NaN	NaN	0.0	0.0	
76	00059815	0.0	Arkwright Mansions Estate	Housing	Fraxinus excelsior	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
83	00059739	0.0	Estate 77-105 Solent Road (odds)	Housing	Prunus persica	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
87	00059382	1.0	Westcroft Estate 6	Housing	Fraxinus excelsior	NaT	NaN	6.0	3.0	0.0	NaN	NaN	0.0	0.0	
91	00059134	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	2.0	3.0	NaN	NaN	0.0	0.0	
844	00059688	1.0	Estate 26 Castle Road (flats A-F)	Housing	x Cupresocyparis leylandii	NaT	NaN	7.0	3.0	15.0	NaN	NaN	0.0	0.0	
1583	00058816	1.0	CHESTER ROAD	Highways	Prunus x hillieri 'Spire'	NaT	NaN	3.0	2.0	5.0	NaN	NaN	0.0	0.0	
1664	00058888	0.0	CAMDEN ST, ST. MARTINS GARDENS (LS)	Parks	Sambucus nigra	NaT	NaN	7.0	3.0	14.0	NaN	NaN	0.0	0.0	
1742	00059699	1.0	Highgate New Town Estate 2	Housing	Fraxinus excelsior	NaT	NaN	16.0	9.0	55.0	NaN	NaN	0.0	0.0	
4116	00045417	0.0	FITZJOHN'S SCHOOL (E)	Education	Betula pendula	2019-05-08	2022/2023	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
4312	00059139	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
4627	00060384	2.0	Broadfield Estate 2	Housing	Acer pseudoplatanus	NaT	NaN	8.0	6.0	30.0	NaN	NaN	0.0	0.0	
4810	00059689	1.0	Estate 26 Castle Road (flats A-F)	Housing	x Cupresocyparis leylandii	NaT	NaN	7.0	2.0	15.0	NaN	NaN	0.0	0.0	
7035	00059957	1.0	Estate 51 Ravenshaw	Housing	Fraxinus excelsior	NaT	NaN	5.0	4.0	10.0	NaN	NaN	0.0	0.0	

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Lor
			Street												
8451	00060083	1.0	TAVISTOCK SQUARE, GARDENS (LS)	Parks	Vacant Tree Pit	NaT	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	
8952	00045420	0.0	FITZJOHN'S SCHOOL (E)	Education	Prunus domestica	2019-05-08	2022/2023	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
9261	00060320	1.0	Hilgrove Estate 2	Housing	Buddleia davidii	NaT	NaN	3.0	0.0	0.0	NaN	NaN	0.0	0.0	
9502	00059227	0.0	Estate 1-16 New Campden Court (cons)	Housing	Hedera (Species) - Ivy	2019-01-16	2021/2022	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
9595	00059956	1.0	Estate 51 Ravenshaw Street	Housing	Unknown	NaT	NaN	5.0	4.0	10.0	NaN	NaN	0.0	0.0	
12048	00060319	1.0	Hilgrove Estate 2	Housing	Buddleia davidii	NaT	NaN	3.0	0.0	0.0	NaN	NaN	0.0	0.0	
13050	00058608	1.0	Abbey Estate 1	Housing	Populus alba	NaT	NaN	3.0	1.5	4.0	NaN	NaN	0.0	0.0	
13125	00060254	1.0	Kingsland Estate	Housing	Taxus baccata	NaT	NaN	9.0	5.0	0.0	NaN	NaN	0.0	0.0	
13366	00053437	1.0	Maiden Lane Estate	Housing	Prunus avium	2014-03-26	2016/2017	7.0	5.0	29.0	NaN	NaN	0.0	0.0	
13650	00058509	1.0	Ingestre Road Estate	Housing	Sambucus nigra	2019-12-04	2022/2023	7.0	3.0	0.0	NaN	NaN	0.0	0.0	
13887	00059135	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
14438	00059955	1.0	Estate 51 Ravenshaw Street	Housing	Acer pseudoplatanus	NaT	NaN	5.0	4.0	10.0	NaN	NaN	0.0	0.0	
15582	00059140	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	15.0	3.0	NaN	NaN	0.0	0.0	
15668	00059730	1.0	CAMDEN SQUARE, GARDENS (LS)	Parks	Liriodendron tulipifera	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
16214	00059700	0.0	Highgate New Town Estate 2	Housing	Unknown	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
16311	00059394	0.0	Netherwood Street Nature Area	Highways	Populus alba	NaT	NaN	0.0	0.0	100.0	NaN	NaN	0.0	0.0	
17037	00059143	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
17628	00059146	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
17679	00060372	1.0	Raglan Street Estate	Housing	Vacant Tree Pit	NaT	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	
17861	00059137	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
18827	00058477	3.0	Holly Lodge Estate	Housing	Prunus laurocerasus	NaT	NaN	6.0	6.0	15.0	NaN	NaN	0.0	0.0	
19132	00060404	0.0	Hilgrove Estate 1	Housing	Ailanthus altissima	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
19444	00059144	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
19586	00060078	0.0	Mortimer Estate	Housing	Fraxinus excelsior	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
19830	00059141	1.0	BLOOMSBURY SQUARE, GARDENS (LS)	Parks	Cornus kousa Milky Way	NaT	NaN	1.8	1.5	3.0	NaN	NaN	0.0	0.0	
20009	00060211	1.0	Estate 1-18 Hancock Nunn House (cons)	Housing	Stump Only	NaT	NaN	0.0	0.0	0.0	NaN	NaN	0.0	0.0	
21439	00059190	1.0	Abbey Estate 1	Housing	Unknown	2018-12-20	2021/2022	5.0	5.0	0.0	NaN	NaN	0.0	0.0	

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Lo
21481	00059954	1.0	Estate 51 Ravenshaw Street	Housing	Malus domestica cultivar	NaT	NaN	5.0	4.0	10.0	NaN	NaN	0.0	0.0	
23338	00060137	1.0	Studholme Court Estate	Housing	Salix caprea	NaT	NaN	7.0	0.0	0.0	NaN	NaN	0.0	0.0	

In [95]:

```
# Remove 0 Eastings.
geotrees=geotrees[geotrees['Easting']!=0]

#confirm removal
geotrees[geotrees['Easting']==0]
```

Out[95]:

Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Longitude	Latitude	I

In [96]:

```
# Remove null Northings
geotrees[geotrees['Northing']==0] # Confirms there are no nulls in Northing column
```

Out[96]:

Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Longitude	Latitude	I

In [97]:

```
# Remove 0 Northings.
geotrees[geotrees['Northing']==0] # confirms there are no Northing with 0 values
```

Out[97]:

Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Longitude	Latitude	I

In [98]:

```
# Confirm how many rows we have
geotrees.shape[0]
```

Out[98]: 23388

Map of Trees

Now make the plot. Do a scatter plot of Northing vs Easting. You should obtain an outline of the map of Camden. Compare that outline with a real map of Camden (use good old google maps!). You'll be able to spot the trees that should not be in that dataset from there!

TODO: Complete the following code cell

In [99]:

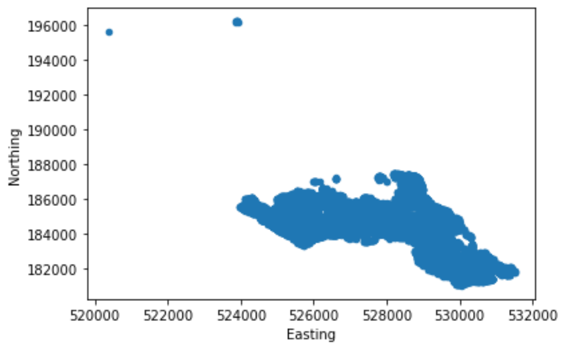
```
geotrees.head()
```

Out[99]:

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	L
0	00060053	1.0	Russell Nurseries Estate	Housing	Vacant Tree Pit	NaT	NaN	NaN	NaN	NaN	E05000135	Hampstead Town	527305.0	185240.0	-
1	00057855	1.0	BRECKNOCK JMI (E)	Education	Vacant Tree Pit	2019-07-17	2022/2023	NaN	NaN	NaN	E05000131	Cantelowes	529923.0	184782.0	-
3	00059915	1.0	ROSARY RC JMI (E)	Education	Betula jacquemontii	NaT	NaN	4.0	1.0	6.0	E05000135	Hampstead Town	527249.0	185261.0	-
4	00010762	1.0	Holly Lodge Estate	Housing	Ilex x altaclarensis	2017-06-14	2020/2021	14.0	6.0	26.0	E05000137	Highgate	528414.0	186770.0	-
5	00007523	1.0	Westcroft Estate 1	Housing	Betula pendula	2018-08-06	2021/2022	9.0	7.0	29.0	NaN	NaN	524253.0	185982.0	-

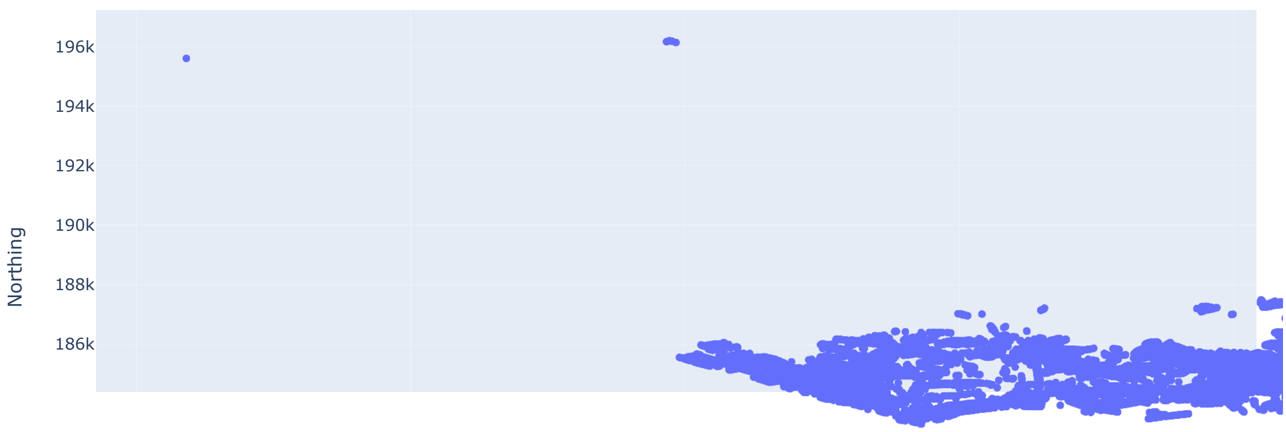
In [100…

```python
# Plot the trees on a map
geotrees.plot.scatter(x='Easting', y='Northing');
```



In [101…

```python
# Using plotly to easliy select trees outside of Camden area
px.scatter(geotrees,x='Easting', y='Northing')
```



Find Trees Outside Camden

From the scatter plot, you should be able to determine how to select the rows from the trees data set containing the offending trees (using the Easting and Northing values)

Select the rows containing trees outside of Camden. Use the filter technique again.

TODO: Complete the following code cells

In [102…

```
# Select the outlier rows
geotrees[geotrees['Northing']>194000]
```

Out[102…

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northing	Longituc
78	00044991	1.0	Bells Hill Estate	Housing	Fraxinus excelsior	2017-04-28	2020/2021	20.0	10.0	49.0	NaN	NaN	523883.0	196179.0	-0.21071
298	00045000	1.0	Estate 167 Furzehill Road	Housing	Pinus sylvestris	2017-04-28	2020/2021	10.0	3.0	41.0	NaN	NaN	520367.0	195595.0	-0.26171
660	00044992	1.0	Bells Hill Estate	Housing	Crataegus monogyna	2017-04-28	2020/2021	4.0	4.0	7.0	NaN	NaN	523875.0	196170.0	-0.2108
1526	00044995	1.0	Bells Hill Estate	Housing	Fraxinus excelsior	2017-04-28	2020/2021	18.0	12.0	54.0	NaN	NaN	523936.0	196127.0	-0.2099
5392	00044990	1.0	Bells Hill Estate	Housing	Aesculus hippocastanum	2017-04-28	2020/2021	22.0	12.0	67.0	NaN	NaN	523889.0	196188.0	-0.21061
18069	00044993	1.0	Bells Hill Estate	Housing	Tilia cordata	2017-04-28	2020/2021	23.0	14.0	89.0	NaN	NaN	523867.0	196159.0	-0.2109
18078	00044601	1.0	Bells Hill Estate	Housing	Tilia cordata	2017-04-28	2020/2021	21.0	12.0	56.0	NaN	NaN	523905.0	196174.0	-0.2103
19532	00044988	1.0	Bells Hill Estate	Housing	Tilia cordata	2017-04-28	2020/2021	21.0	12.0	65.0	NaN	NaN	523909.0	196169.0	-0.2103

In [103…

```
# Confirm how many rows we have
trees_out_camden=geotrees[geotrees['Northing']>194000]
trees_out_camden.shape[0]
```

Out[103… 8

Observations

TODO: Write down your observation about geolocation issues.

There are 8 trees that are not within the Camden. These are located in Bells Hill Estate and Estate 167 Furzehill Road which are both around Barnet not Camden.

Task 8: Identify Unmatched Data

We have multiple datasets that will need to be joined together to produce the analyses required by the Camden Parks and Open Spaces team. The data will need to be joined in the following way:

- Use the `Identifier` column in the trees dataset to match to the `Identifier` column in the environmental data set (so we can bring in the environmental data for each tree)
- Use the `Scientific Name` column in the trees dataset to match to the `Scientific Name` column in the common names data set (so we can look up the `Common Name`)

There may be mismatches in the data. Of particular concern we want to check

- That every tree in the trees dataset has matching environmental data in the environmental data set
- That every environmental row in the environmental dataset has matching tree data in the tree data set
- That every scientific name in the trees dataset has a matching common name in the common names data set

We aren't too concerned about the reverse of the last scenario (if we have extra names in the common names dataset that aren't in the trees data set). We don't expect Camden to have a specimen of every tree that exists!

There are a few ways this can be done, but one technique is to use the `isin` function to check if some column in one dataframe contains values that are in another column in another dataframe. This creates a mask containing rows that match between the two dataframes:

```
mask = df1['column_name1'].isin(df2['column_name2'])
```

To select the non-matching rows, we can use Python's bitwise not operator ~:

```
mask = ~df1['column_name1'].isin(df2['column_name2'])
```

As we have seen before, the mask can be used to select that subset of rows back from the original dataframe.

Find Trees that Don't have Matching Environmental Data

TODO: Complete the following code cells

In [104…

```
# Check number of rows for trees dataset
geotrees.shape[0]
```

Out[104…

```
23388
```

In [105…

```
trees_env.shape[0]
```

Out[105…

```
23415
```

In [106…

```
trees_com_names.shape[0]
```

Out[106…

```
589
```

In [107…

```
# Find trees that don't have matching environmental data
mask = ~geotrees['Identifier'].isin(trees_env['Identifier'])
geotrees[mask]
```

Out[107…

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	N
66	00059712	1.0	Maiden Lane Estate	Housing	Acer saccharinum	2019-05-28	2022/2023	12.0	5.0	20.0	E05000131	Cantelowes	529795.0	1
125	00048578	1.0	BUCK STREET	Highways	Sorbus aucuparia	2017-07-19	2020/2021	6.0	2.0	10.0	E05000130	Camden Town with Primrose Hill	528900.0	1
1148	00006577	1.0	FAWLEY ROAD	Highways	Tilia euchlora	2018-09-28	2021/2022	15.0	6.0	38.0	E05000145	West Hampstead	525572.0	1
1998	00007366	1.0	FORTUNE GREEN RD, OPEN SPACE (LS)	Parks	Ilex aquifolium	2017-03-21	2019/2020	9.0	6.0	44.0	E05000132	Fortune Green	525074.0	1
2246	00014633	1.0	Mortimer Estate	Housing	Tilia europaea	2019-01-29	2021/2022	16.0	12.0	47.0	E05000140	Kilburn	525763.0	1
5478	00060382	1.0	SHAFTESBURY AVENUE	Highways	Vacant Tree Pit	NaT	NaN	NaN	NaN	NaN	E05000138	Holborn and Covent Garden	530073.0	1
10637	00002874	1.0	BURGHLEY ROAD	Highways	Platanus x hispanica	2017-08-14	2020/2021	20.0	8.0	52.0	E05000139	Kentish Town	529119.0	1
10977	00055227	1.0	BURGHLEY ROAD	Highways	Amelanchier lamarckii	2017-08-14	2020/2021	3.0	2.0	5.0	E05000139	Kentish Town	528920.0	1
11795	00016702	1.0	RED LION SQUARE, GARDENS (LS)	Parks	Platanus x hispanica	2018-06-04	2021/2022	30.0	23.0	165.0	E05000138	Holborn and Covent Garden	530572.0	1
11856	00054744	1.0	Carrol & Sanderson Close Estate	Housing	Prunus unidentified species	2017-01-06	2020/2021	3.0	3.0	13.0	E05000137	Highgate	528661.0	1
12056	00003694	1.0	Estate 1-161 Burnham (cons)	Housing	Acer platanoides	2018-04-17	2021/2022	3.0	1.0	6.0	E05000128	Belsize	527015.0	1
12936	00054558	1.0	ST. MARY'S KILBURN C OF E JMI (E)	Education	Amelanchier lamarckii	2019-10-07	2022/2023	4.0	2.0	8.0	E05000140	Kilburn	525443.0	1
13248	00059317	1.0	ADELAIDE ROAD NATURE AREA	Parks	Stump Only	2019-01-31	2021/2022	0.0	5.0	50.0	E05000128	Belsize	527577.0	1
16815	00055884	1.0	HONEYBOURNE ROAD	Highways	Acer pseudoplatanus 'Brilliant	2018-05-10	2021/2022	2.0	1.0	4.0	E05000145	West Hampstead	525593.0	1
18690	00059963	1.0	Ampthill Square Estate	Housing	Vacant Tree Pit (planned: Parrotia persica van...	2019-01-08	2022/2023	NaN	NaN	NaN	E05000143	St Pancras and Somers Town	529216.0	1
18958	00059246	1.0	Belsize nature reserve, Russell Nursery	Parks	Ulmus procera	2019-01-29	2021/2022	5.0	4.0	11.0	E05000134	Gospel Oak	527523.0	1

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	N
19606	00005127	1.0	CUMBERLAND MARKET, OPEN SPACE (LS)	Parks	Platanus x hispanica	2018-03-13	2020/2021	10.0	8.0	38.0	E05000142	Regent's Park	528913.0	1
20169	00017912	1.0	SHERRIFF ROAD	Highways	Tilia platyphyllos	2018-10-09	2021/2022	9.0	5.0	42.0	E05000145	West Hampstead	525265.0	1
20226	00047080	1.0	Ampthill Square Estate	Housing	Malus unidentified species	2019-01-08	2022/2023	5.0	3.0	16.0	E05000143	St Pancras and Somers Town	529279.0	1
21287	00029059	1.0	Estate 1-20 Marrick House (cons)	Housing	Sambucus nigra	2018-06-19	2021/2022	6.0	6.0	39.0	E05000140	Kilburn	525832.0	1
22470	00012126	1.0	KINGS COLLEGE ROAD	Highways	Fraxinus excelsior	2018-07-13	2021/2022	18.0	12.0	48.0	E05000128	Belsize	526999.0	1
23301	00010784	1.0	Holly Lodge Estate	Housing	Ilex aquifolium	2017-06-14	2020/2021	7.0	5.0	20.0	E05000137	Highgate	528472.0	1
23315	00056485	1.0	WATERLOW PARK (LS)	Parks	Fraxinus excelsior	2019-05-24	2022/2023	12.0	5.0	16.0	E05000137	Highgate	528730.0	1

In [108…

```
# Confirm how many rows we have
geotrees[mask].shape[0]
```

Out[108…

23

Find Environmental Data that Doesn't have Matching Tree Data

TODO: Complete the following code cells

In [109…

```
# Find environmental data that doesn't have matching tree data
mask = ~trees_env['Identifier'].isin(geotrees['Identifier'])
trees_env[mask]
```

Out[109…

	Identifier	Maturity	Physiological Condition	Tree Set To Be Removed	Removal Reason	Capital Asset Value For Amenity Trees	Carbon Storage In Kilograms	Gross Carbon Sequestration Per Year In Kilograms	Pollution Removal Per Year In Grams
176	00059689	Not Applicable	Not applicable	No	NaN	2022.76	NaN	NaN	NaN
778	00059957	Middle aged	Fair	No	NaN	1078.80	NaN	NaN	NaN
1506	00059394	Not Applicable	Not applicable	No	NaN	95893.84	NaN	NaN	NaN
1604	00060372	NaN	NaN	No	NaN	NaN	NaN	NaN	NaN
1759	00059144	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
2039	00060078	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
2487	00060359	NaN	NaN	No	NaN	NaN	NaN	NaN	NaN
2488	00059815	Middle aged	Good	No	NaN	0.00	NaN	NaN	NaN
3554	00059956	Middle aged	Fair	No	NaN	1198.67	NaN	NaN	NaN
3956	00060319	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
4029	00059688	Not Applicable	Not applicable	No	NaN	2022.76	NaN	NaN	NaN
5183	00059698	Middle aged	Good	No	NaN	0.00	NaN	NaN	NaN
5447	00058509	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
5621	00060384	Middle aged	Fair	No	NaN	7767.40	NaN	NaN	NaN
5918	00059190	Mature	Good	No	NaN	0.00	NaN	NaN	NaN
5936	00059142	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
6219	00059136	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
6372	00059135	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
6765	00059138	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN

	Identifier	Maturity	Physiological Condition	Tree Set To Be Removed	Removal Reason	Capital Asset Value For Amenity Trees	Carbon Storage In Kilograms	Gross Carbon Sequestration Per Year In Kilograms	Pollution Removal Per Year In Grams
7711	00058873	Juvenile	Good	No	NaN	0.00	NaN	NaN	NaN
8279	00059143	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
8288	00060137	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
9030	00060407	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
9082	00059139	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
9885	00053437	Mature	Poor	No	NaN	6048.50	198.5	7.5	60.8
10463	00059699	Mature	Fair	No	NaN	29370.48	NaN	NaN	NaN
10950	00059380	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
12431	00059140	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
12788	00059954	Middle aged	Fair	No	NaN	647.28	NaN	NaN	NaN
12884	00059953	Middle aged	Fair	No	NaN	863.04	NaN	NaN	NaN
14029	00059955	Middle aged	Fair	No	NaN	863.04	NaN	NaN	NaN
14230	00058477	Middle aged	Good	No	NaN	970.93	NaN	NaN	NaN
14370	00059134	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
15348	00059542	Not Applicable	Good	No	NaN	152769.67	NaN	NaN	NaN
16031	00060404	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
16065	00059700	Mature	Fair	No	NaN	0.00	NaN	NaN	NaN
16208	00045420	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
16570	00060320	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
16764	00059141	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
16860	00058816	Juvenile	Good	No	NaN	179.80	NaN	NaN	NaN
16936	00059227	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
17452	00059739	Mature	Fair	No	NaN	0.00	NaN	NaN	NaN
19082	00059372	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
19119	00060385	Middle aged	Fair	No	NaN	1456.39	NaN	NaN	NaN
19196	00060252	NaN	NaN	No	NaN	NaN	NaN	NaN	NaN
19206	00058608	Juvenile	Good	No	NaN	153.43	NaN	NaN	NaN
19520	00059730	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN
19638	00060083	NaN	NaN	No	NaN	NaN	NaN	NaN	NaN
19736	00059382	Juvenile	Fair	No	NaN	0.00	NaN	NaN	NaN
20004	00059145	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
20231	00059137	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
21896	00058888	Not Applicable	Not applicable	No	NaN	1268.68	NaN	NaN	NaN
22051	00060254	Middle aged	Fair	No	NaN	0.00	NaN	NaN	NaN
22511	00059146	Juvenile	Good	No	NaN	64.73	NaN	NaN	NaN
22778	00060211	Not Applicable	Dead	No	NaN	0.00	NaN	NaN	NaN
22819	00045417	Not Applicable	Not applicable	No	NaN	0.00	NaN	NaN	NaN

In [110…

```
# Confirm how many rows we have
trees_env[mask].shape[0]
```

Out[110…

56

Find Trees that Don't have Matching Common Names Data

TODO: Complete the following code cells

In [111…

```
# Find trees with scientific names that don't have matching common names data
mask = ~geotrees['Scientific Name'].isin(trees_com_names['Scientific Name'])
geotrees[mask]
```

Out[111…

	Identifier	Number Of Trees	Site Name	Contract Area	Scientific Name	Inspection Date	Inspection Due Date	Height In Metres	Spread In Metres	Diameter In Centimetres At Breast Height	Ward Code	Ward Name	Easting	Northi
151	00051832	1.0	ARGYLE WALK	Highways	Sorbus aucuparia 'Streetwise'	2019-02-10	2022/2023	7.0	3.0	12.0	E05000141	King's Cross	530227.0	18270
384	00053954	1.0	CHURCHILL ROAD	Highways	Sorbus aucuparia 'Streetwise'	2017-10-07	2020/2021	3.0	2.0	5.0	E05000139	Kentish Town	529007.0	18597
495	00047497	1.0	PATSHULL PLACE	Highways	Sorbus aucuparia 'Streetwise'	2017-06-22	2020/2021	5.0	3.0	11.0	E05000131	Cantelowes	529202.0	18471
611	00055434	1.0	SHARPLES HALL STREET	Highways	Sorbus aucuparia 'Streetwise'	2019-09-30	2022/2023	2.0	2.0	4.0	E05000130	Camden Town with Primrose Hill	527962.0	18405
653	00055289	1.0	QUEEN'S CRESCENT	Highways	Sorbus aucuparia 'Streetwise'	2017-07-08	2020/2021	4.0	1.0	7.0	E05000136	Haverstock	528072.0	18472
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21826	00050835	1.0	INGESTRE RD	Highways	Sorbus aucuparia 'Streetwise'	2017-08-18	2020/2021	3.0	1.0	7.0	E05000139	Kentish Town	528962.0	18582
22948	00052341	1.0	NEW COMPTON STREET	Highways	Sorbus aucuparia 'Streetwise'	2019-07-08	2022/2023	4.0	3.0	6.0	E05000138	Holborn and Covent Garden	529976.0	18116
23266	00048846	1.0	ASMARA ROAD	Highways	Sorbus aucuparia 'Streetwise'	2018-08-28	2021/2022	5.0	3.0	8.0	E05000132	Fortune Green	524568.0	18534
23335	00048705	1.0	GOLDINGTON STREET	Highways	Sorbus aucuparia 'Streetwise'	2019-10-23	2022/2023	6.0	2.0	12.0	E05000143	St Pancras and Somers Town	529662.0	18341
23372	00031627	1.0	ST. GEORGE THE MARTYR C OF E JMI (E)	Education	Cotoneaster salicifolius	2018-07-23	2021/2022	5.0	5.0	8.0	E05000138	Holborn and Covent Garden	530742.0	18211

76 rows × 17 columns

In [112…

```
# Confirm how many rows we have
geotrees[mask].shape[0]
```

Out[112… 76

Observations

TODO: Write down your observation about unmatched data issues.

With the merging plan to create a single dataset for the project, when we merge the `geotrees` DataFrame with the `trees_env` DataFrame to pull in the environmental data, we will have 23 trees that will have missing environmental information when the merge is done.

Similary, when the `tree_com_names` DataFrame is merged with the `geotrees` DataFrame to update the common dates, there will be 76 trees with the common names.

END OF NOTEBOOK

In [ ]: