In [1]:
```
# importing the necessary modules
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

Prospect Mapping for PGL

Questions

- PP Bracket?
- What is the true indicator to measure success of a field?
- FY?

I used a script to convert the data to a csv file as it was taking about 30 seconds to read the excel sheet. CSV is usually faster to read.

In [2]:
```
# Importing the data from the .xlsx as a data frame using the pandas module
df = pd.read_csv("PGL_data.csv", low_memory = False)
```

df now holds the whole data frame which is a bit too large to handle. Below I have created a list of all the headings so that I can see them without opening the excel sheet. When I want to reference a heading, I can just use the index of the heading rather than having to remember the name. These are stored in the variable 'col'.

In [3]:
```
col = df.columns.tolist()

print('index  |                      column name     |    number of entries')
print('-------------------------------------------------------------------------------------------------
for heading in col:
    print("%3.0f    |    %30s    |    %8.0f" % (col.index(heading), heading, len(df[head
print('-------------------------------------------------------------------------------------------------
df.head()
```

index	column name	number of entries
0	Account Name	187864
1	Establishment Aplicor ID	187858
2	Contact Aplicor ID	187864
3	Stage	187514
4	Opportunity Source	87182
5	Loss Reason	51912
6	Product	183022
7	Course	59973
8	Booking Reference	187864
9	Centre(s) of Interest	63827
10	Accommodation Type	21327
11	Arrival Date	165044
12	Booking Date	139149
13	Age Range	59343
14	Number of Children	160007
15	Number of Adults	159168
16	Number of free passengers	43160
17	Transport	20102
18	Cost per person Currency	46430
19	Cost per person	46430
20	Requested Duration	37486
21	Gross Revenue Currency	116241
22	Gross Revenue	116241
23	Discount Currency	187856
24	Discount	187856
25	Discount Explanation	7713
26	Additional Needs	18869
27	Competitors	3229
28	Booking FY	86508
29	Booking Month	139149
30	Arrival FY	119442
31	Arrival Month	165044
32	Days between Booking & Arrival	187819
33	Number of Children Banding	154771
34	Centre Alias	56242
35	Course Alias	59971
36	First Order	187864
37	First Order FY	41785
38	First Order Month	187864
39	New_Existing	187864
40	Booking Season	139149
41	Time Lapse Band	187819
42	Number of Children Band	154771
43	PP Bracket	60284
44	School Size by Pupils Banding	61768
45	By Radias to Nearest Centre	61768
46	Establishmet_Type	61768
47	Decile_Value	143898
48	Decile_F&M	143898
49	Column21	0
50	Column22	0
51	Column23	0
52	Column24	0

Out[3]:

	Account Name	Establishment Aplicor ID	Contact Aplicor ID	Stage	Opportunity Source	Loss Reason	Product
0	Alma Park Primary School	6525E1FA-F266-4098-B85F-ACB0C480BC5B	41FCCBCA-318A-48F5-9BA2-9166764C48E2	Closed - Not booked	Researcher sourcing	Reason not Listed	AUK (Adventure UK)
1	Kincardine O'Neil Primary School	CDF80166-81C0-41CC-B29D-243D0EFBE2CC	AFFC2AD8-C98B-4BE8-849F-16A883F7A220	Closed - Not booked	Post-Travel	Reason not Listed	AUK (Adventure UK)
2	St Peters CE Primary School	CFF1D51F-AF75-42C9-8C3C-D6855748F2C1	380FD780-93DB-4B70-B9FC-398692713B63	Closed - Booked	NaN	NaN	AUK (Adventure UK)
3	St Johns Academy	90A38A25-FD5F-477E-8D53-10A67ECFC96B	1702E5D4-0DFB-47CC-B8E0-686B543FC520	Closed - Not Quoted	Cancellation Follow-up	Reason not Listed	AUK (Adventure UK)
4	Our Ladys RC High School	DBC74901-C10C-4E19-AB61-876CE061D321	B496561D-4D44-4C1F-830A-B1E5B3F2F833	Closed - Not Quoted	Failed (non converted) Quote	Not running a trip	SK (Ski)

5 rows × 53 columns

d1 below is formed of only the colums that I want to begin working with. The resulting dataframe can be seen printed underneath it

In [4]:
```
d1 = df[[col[0], col[33],col[40],col[20]]]
d1
```

Out[4]:

	Account Name	Number of Children Banding	Booking Season	Requested Duration
0	Alma Park Primary School	NaN	NaN	NaN
1	Kincardine O'Neil Primary School	NaN	NaN	NaN
2	St Peters CE Primary School	NaN	NaN	NaN
3	St Johns Academy	51-100	NaN	5d/4n
4	Our Ladys RC High School	NaN	NaN	NaN
...	...	...	...	...
187859	Bradley Stoke Community School	51-100	NaN	Ad hoc (add details to Booking notes field)
187860	Fordham All Saints Ce Primary School	NaN	NaN	NaN
187861	Archbishop Hutton Primary School	NaN	NaN	NaN
187862	Sir John Lawes School	151-200	Autumn	NaN
187863	Hylands Primary School	21-50	NaN	NaN

187864 rows × 4 columns

To analyse the effects of a school caracteristic on their interaction with PGL, a column must be chosen which represents the extent of interaction and this column should really be filled in for all schools.

In [5]:
```
k=6
print(col[k])
df[col[k]].loc[df[col[k]].notna()]
```

```
          Product
Out[5]:  0         AUK (Adventure UK)
         1         AUK (Adventure UK)
         2         AUK (Adventure UK)
         3         AUK (Adventure UK)
         4                  SK (Ski)
                         ...
         187859    AUK (Adventure UK)
         187860    AUK (Adventure UK)
         187861    AUK (Adventure UK)
         187862    AUK (Adventure UK)
         187863    AUK (Adventure UK)
         Name: Product, Length: 183022, dtype: object
```

Number of Children band

Let's focus on one caracteristic at a time starting with Number of Children.

There seems to be a lot of empty cells so to begin with, it may be useful to understand what proportion of each column is blank. The line below is removing all lines in this column that have NaN in the cell and showing how many rows remain. Number of Children has been used as an example.

In [6]:
```
def notna_in_col(name):
    print("total amount of entries " + str(len(df)))
    print("amount of non NaN in " + name + " " + str(len(df[name].loc[df[name].notna()]
    print(str(len(df[name].loc[df[name].notna()]) / len(df) * 100) + "% of schools have

notna_in_col('Number of Children Band')
```

```
total amount of entries 187864
amount of non NaN in Number of Children Band 154771
82.38459736830899% of schools have an entry for this column
```

By taking the number of children column and removing all empty cells, the row count falls from 187864 to 154771 meaning that 82.4% of schools have a value for that column.

I wanted to briefly look at one that looked sparsely populated. For the School Size by Pupils Banding, only 32.9% of schools have a value for this category. If an average value has to be given to the schools without values, this may skew the data for this category. I would suggest that columns withless than 50% of schools data should probabily be removed from a decile scoring to avoid adding weight to averages.

Below is a count of how many schools are in each band. This is interesting to know however does not tell us anything about how this affects the interaction with PGL.

In [7]:
```
a = 39
print(col[a])
df[col[a]].loc[df[col[a]].notna()].value_counts()
```

```
         New_Existing
Out[7]:  Existing    164170
         New          23694
         Name: New_Existing, dtype: int64
```

To find how a certain column affects the schools interaction with PGL, a quantifiying column must be chosen which tells us their level of interaction. This column must also be completely, or at least mostly, populated to show this correlation.

I have updated the table in cell 3 to add a column which includes how many entries each column has to see if a characteristic can be used to judge a schools interaction with PGL.

New_existing could have been useful I think if it said how many time a school used PGL instead of a binary answer.

Below is a line to see the first 50 entries of a column. Just useful to get a preview without having to open up the data. It is commented out at the moment to reduce clutter.

In [11]:
```
#df[col[20]].loc[df[col[20]].notna()].head(50)
```

Now looking at the influence of a column starting with the number of children. Maybe the time lapse band will be a good indication of how keen they are? Might be misinterpretting this. I see that there are a lot of lines saying 'Do Not Apply' which I think is like being empty so making sure that there are enough entries, the next cell will work out how many cells have an entry and removing any that have No Not Apply.

In [14]:
```
time_lapse_notna  = df[col[41]].loc[df[col[41]].notna()]
time_lapse_notna = time_lapse_notna.loc[~time_lapse_notna.str.contains("Do Not Apply")]
print(len(time_lapse_notna))
```

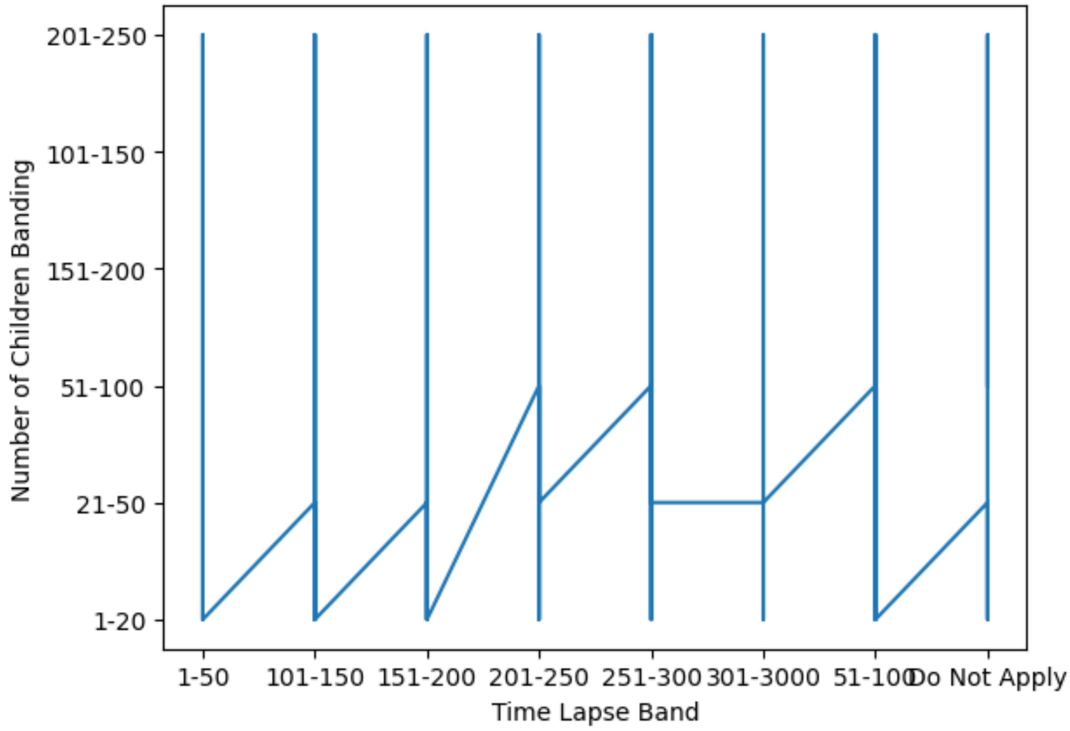
```
164987
```

There are enough cells with data to use it.

Below I am plotting Time lapse against Number of Children to see if there is any relationship.

In [17]:
```
dx = df.loc[df[col[41]].notna() & df[col[33]].notna()].sort_values(col[41],ascending =
plt.plot(dx[col[41]],dx[col[33]])
plt.xlabel('%s' % col[41])
plt.ylabel('%s' % col[33])
```

Out[17]:
```
Text(0, 0.5, 'Number of Children Banding')
```



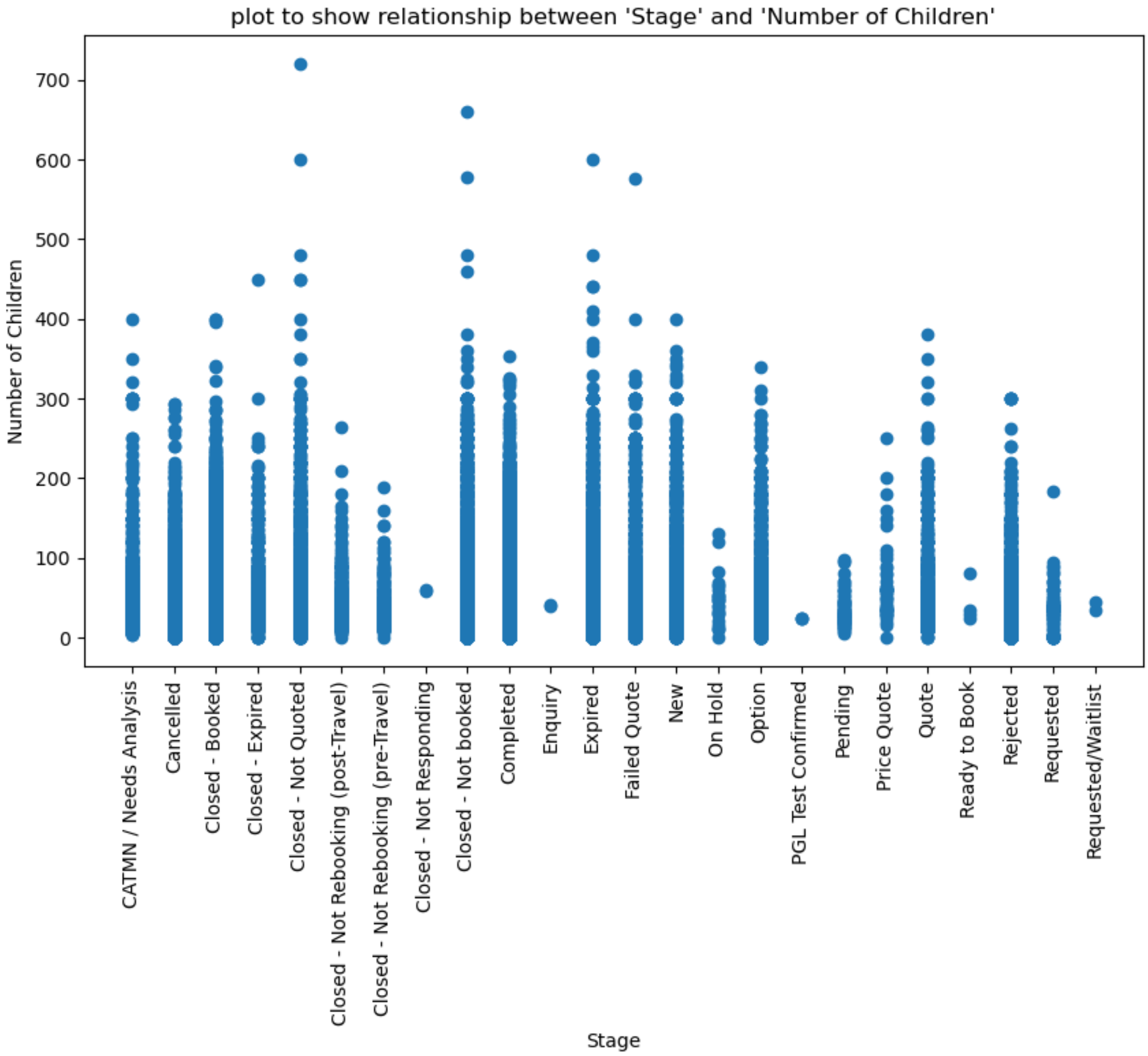
From this plot its clear that there is no trend to extract, also, with banded categories, it would be hard to extract any linear regressions.

I am going to create a function below which will plot any two categories for us to quickly flick through combinations.

In [48]:
```
import seaborn as sns
def relationshiplot(x,y):
    dx = df.loc[df[x].notna() & df[y].notna()].sort_values(x,ascending = True)
    #sns.lineplot(dx[x],dx[y])

    plt.figure(figsize=(10,6))
    plt.scatter(dx[x],dx[y])
    plt.xlabel('%s' % x)
    plt.tick_params(rotation=90, axis = 'x')
    plt.ylabel('%s' % y)
    plt.title("plot to show relationship between '%s' and '%s'" % (x, y))

relationshiplot(col[3],col[14])
```



This plot above shows us a bit more about the relationship between the stage of the booking and the number of children. This time I used the number of children as raw number rather than the band which I believe shows a better relationship.