Sample

April 20, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re
import numpy as np
```

1 Helper functions

These are borrowed from the Convert.ipynb file.

[2]:
```
headings = ['Building Identifier',
            'Country',
             'City',
             'Quality / Stage of Data',
             'Construction Date',
             'Building Type',
             'Gross Floor Area']
```

[3]:
```
df = pd.read_excel('../Dataset/dataset.xlsx',header=1).drop('Unnamed: 0',axis=1)
```

[4]:
```
df
```

[4]:
```
    Building Identifier Country City Quality / Stage of Data  \
0                     1      CA  TOR                   00IFC
1                     2      CA  TOR                   00IFC
2                     3      CA  TOR                   00IFC
3                     4      CA  TOR                   00IFC
4                     5      CA  TOR                   00IFC
5                     6      CA  TOR                   00IFC
6                     7      CA  TOR                   00IFC
7                     8      CA  TOR                   00IFC
8                     9      CA  TOR                   00IFC
9                    10      CA  TOR                   00IFC
10                   11      CA  TOR                   00IFC
11                   12      CA  TOR                   00IFC
12                   13      CA  TOR                   00IFC
13                   14      CA  TOR                   00IFC
14                   15      CA  TOR                   00IFC
```

15	16	CA	TOR	00IFC
16	17	CA	TOR	00IFC
17	18	CA	TOR	00IFC
18	19	CA	TOR	00IFC
19	20	CA	TOR	00IFC
20	21	CA	TOR	00IFC
21	22	CA	TOR	00IFC
22	23	CA	TOR	00IFC
23	24	CA	TOR	00IFC
24	25	CA	TOR	00IFC
25	26	CA	TOR	00IFC
26	27	CA	WIN	00IFC
27	28	CA	TOR	00IFC
28	29	CA	TOR	00IFC
29	30	CA	TOR	00IFC
30	31	CA	TOR	00IFC
31	32	CA	TOR	00IFC
32	33	CA	TOR	00IFC
33	34	CA	TOR	00IFC
34	35	CA	TOR	00IFC
35	36	CA	TOR	00IFC
36	37	CA	TOR	00IFC
37	38	CA	TOR	00IFC
38	39	CA	TOR	00IFC
39	40	US	NEW	00IFC
40	41	CA	TOR	00IFC
41	42	CA	TOR	00IFC
42	43	CA	TOR	00IFC
43	44	CA	TOR	00IFC
44	45	CA	TOR	00IFC
45	46	CA	TOR	00IFC
46	47	CA	TOR	00IFC
47	48	CA	RIC	0IARC
48	49	CA	TOR	00IFC
49	50	CA	TOR	00IFC
50	51	CA	TOR	00IFC
51	52	CA	TOR	00IFC
52	53	CA	TOR	00IFC
53	54	CA	TOR	00IFC
54	55	CA	TOR	00IFC
55	56	CA	TOR	00IFC
56	57	CA	TOR	00IFC
57	58	CA	TOR	00IFC
58	59	CA	TOR	0IFBP
59	60	CA	TOR	0IFBP

```
    Construction Date Building Type  Gross Floor Area  \
```

0	2021	SND	521.18
1	2021	SND	389.24
2	2021	SND	411.64
3	2021	SND	269.56
4	2011	OFF	11248.00
5	2011	APB	11317.00
6	2021	SND	445.99
7	2021	SND	438.45
8	2021	SND	714.07
9	2021	SND	343.24
10	2009	OFF	73083.00
11	1917	SMD	199.93
12	2021	SND	226.89
13	2021	SND	611.73
14	2021	SND	343.44
15	2021	SND	613.38
16	1969	SND	413.72
17	1969	SND	333.49
18	2021	SND	178.38
19	2021	SND	323.80
20	2020	SND	837.56
21	2021	SND	587.86
22	2021	SND	568.21
23	2021	SMD	234.73
24	2021	SND	294.84
25	2021	SND	496.77
26	2007	OFF	73600.00
27	2021	SND	643.30
28	2021	SND	701.61
29	2021	SMD	257.75
30	2021	SND	378.70
31	2021	SND	324.16
32	2020	SND	533.53
33	2020	SMD	254.05
34	2021	SND	423.03
35	2021	SND	328.16
36	2021	SND	421.59
37	2020	SND	628.59
38	2021	SND	464.51
39	2017	EDU	8983.00
40	2021	SND	346.14
41	1913	SND	161.08
42	2021	SND	891.97
43	2021	SND	525.61
44	2021	SND	502.87
45	2021	SND	379.18
46	2021	SND	549.65

```
47                2016          EDU        6819.00
48                2020          SND         393.82
49                2021          SND         648.14
50                1988          INS       21934.00
51                2018          APB       53146.02
52                2018          MIX       33975.25
53                2017          APB       69784.00
54                2017          APB       39409.04
55                2016          APB       53871.00
56                2020          LNW         137.23
57                2020          LNW         144.92
58                2019          LNW          83.10
59                2021          LNW         234.79

    000_G2010.20.000_03 00 00.00_kg_1  000_B1010.20.000_03 00 00.00_kg_1  \
0                                 NaN                                NaN
1                                 NaN                                NaN
2                                 NaN                                NaN
3                                 NaN                                NaN
4                             13704.0                       1.776816e+06
5                                 NaN                       1.514400e+06
6                                 NaN                                NaN
7                                 NaN                                NaN
8                                 NaN                                NaN
9                                 NaN                                NaN
10                            58008.0                       4.029264e+06
11                                NaN                                NaN
12                                NaN                                NaN
13                                NaN                                NaN
14                                NaN                                NaN
15                                NaN                                NaN
16                                NaN                                NaN
17                                NaN                                NaN
18                                NaN                                NaN
19                                NaN                                NaN
20                                NaN                                NaN
21                                NaN                                NaN
22                                NaN                                NaN
23                                NaN                                NaN
24                                NaN                                NaN
25                                NaN                                NaN
26                                NaN                       4.480680e+06
27                                NaN                                NaN
28                                NaN                                NaN
29                                NaN                                NaN
30                                NaN                                NaN
31                                NaN                                NaN
```

32	NaN	NaN
33	NaN	NaN
34	NaN	NaN
35	NaN	NaN
36	NaN	NaN
37	NaN	NaN
38	NaN	NaN
39	NaN	2.191431e+04
40	NaN	NaN
41	NaN	NaN
42	NaN	NaN
43	NaN	NaN
44	NaN	NaN
45	NaN	NaN
46	NaN	NaN
47	NaN	3.756000e+04
48	NaN	NaN
49	NaN	NaN
50	NaN	NaN
51	NaN	NaN
52	NaN	NaN
53	NaN	NaN
54	NaN	NaN
55	NaN	NaN
56	NaN	NaN
57	NaN	NaN
58	NaN	NaN
59	NaN	NaN

	000_C1010.10.000_04 22 00.00_kg_1	…	000_B2010.10.000_07 46 16.00_kg_2	\
0	NaN	…	NaN	
1	NaN	…	NaN	
2	NaN	…	NaN	
3	NaN	…	NaN	
4	19397.560000	…	NaN	
5	53877.650000	…	NaN	
6	NaN	…	NaN	
7	NaN	…	NaN	
8	NaN	…	NaN	
9	NaN	…	NaN	
10	562574.500000	…	NaN	
11	NaN	…	NaN	
12	NaN	…	NaN	
13	NaN	…	NaN	
14	NaN	…	NaN	
15	NaN	…	NaN	
16	NaN	…	NaN	

17	NaN	…	NaN
18	NaN	…	NaN
19	NaN	…	NaN
20	NaN	…	NaN
21	NaN	…	NaN
22	NaN	…	NaN
23	NaN	…	NaN
24	NaN	…	NaN
25	NaN	…	NaN
26	354208.227500	…	NaN
27	NaN	…	NaN
28	NaN	…	NaN
29	NaN	…	NaN
30	NaN	…	NaN
31	NaN	…	NaN
32	NaN	…	NaN
33	NaN	…	NaN
34	NaN	…	NaN
35	NaN	…	NaN
36	NaN	…	NaN
37	NaN	…	NaN
38	NaN	…	NaN
39	8666.292723	…	NaN
40	NaN	…	NaN
41	NaN	…	NaN
42	NaN	…	NaN
43	NaN	…	NaN
44	NaN	…	NaN
45	NaN	…	NaN
46	NaN	…	NaN
47	NaN	…	NaN
48	NaN	…	NaN
49	NaN	…	NaN
50	NaN	…	NaN
51	8194.250000	…	NaN
52	191988.905000	…	NaN
53	82694.400000	…	NaN
54	46298.790000	…	NaN
55	422839.793489	…	NaN
56	NaN	…	NaN
57	NaN	…	NaN
58	NaN	…	NaN
59	NaN	…	67.3

	001_B2010.80.000_07 27 00.00_kg_2	001_B2010.80.000_07 21 13.00_kg_2	\
0	NaN	NaN	
1	NaN	NaN	

2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	NaN	NaN
7	NaN	NaN
8	NaN	NaN
9	NaN	NaN
10	NaN	NaN
11	NaN	NaN
12	NaN	NaN
13	NaN	NaN
14	NaN	NaN
15	NaN	NaN
16	NaN	NaN
17	NaN	NaN
18	NaN	NaN
19	NaN	NaN
20	NaN	NaN
21	NaN	NaN
22	NaN	NaN
23	NaN	NaN
24	NaN	NaN
25	NaN	NaN
26	NaN	NaN
27	NaN	NaN
28	NaN	NaN
29	NaN	NaN
30	NaN	NaN
31	NaN	NaN
32	NaN	NaN
33	NaN	NaN
34	NaN	NaN
35	NaN	NaN
36	NaN	NaN
37	NaN	NaN
38	NaN	NaN
39	NaN	NaN
40	NaN	NaN
41	NaN	NaN
42	NaN	NaN
43	NaN	NaN
44	NaN	NaN
45	NaN	NaN
46	NaN	NaN
47	NaN	NaN
48	NaN	NaN

49	NaN	NaN
50	NaN	NaN
51	NaN	NaN
52	NaN	NaN
53	NaN	NaN
54	NaN	NaN
55	NaN	NaN
56	NaN	NaN
57	NaN	NaN
58	NaN	NaN
59	37.3	112.67

	001_B2010.10.000_09 24 23.00_kg_2	0B1_A5020.10.000_06 11 00.00_kg_2 \
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	NaN	NaN
7	NaN	NaN
8	NaN	NaN
9	NaN	NaN
10	NaN	NaN
11	NaN	NaN
12	NaN	NaN
13	NaN	NaN
14	NaN	NaN
15	NaN	NaN
16	NaN	NaN
17	NaN	NaN
18	NaN	NaN
19	NaN	NaN
20	NaN	NaN
21	NaN	NaN
22	NaN	NaN
23	NaN	NaN
24	NaN	NaN
25	NaN	NaN
26	NaN	NaN
27	NaN	NaN
28	NaN	NaN
29	NaN	NaN
30	NaN	NaN
31	NaN	NaN
32	NaN	NaN
33	NaN	NaN

34	NaN	NaN
35	NaN	NaN
36	NaN	NaN
37	NaN	NaN
38	NaN	NaN
39	NaN	NaN
40	NaN	NaN
41	NaN	NaN
42	NaN	NaN
43	NaN	NaN
44	NaN	NaN
45	NaN	NaN
46	NaN	NaN
47	NaN	NaN
48	NaN	NaN
49	NaN	NaN
50	NaN	NaN
51	NaN	NaN
52	NaN	NaN
53	NaN	NaN
54	NaN	NaN
55	NaN	NaN
56	NaN	NaN
57	NaN	NaN
58	NaN	NaN
59	2655.54	277.59

	OB1_A5020.10.000_06 11 00.00_kg_1	OB1_A5020.10.000_09 21 16.00_kg_1 \
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
5	NaN	NaN
6	NaN	NaN
7	NaN	NaN
8	NaN	NaN
9	NaN	NaN
10	NaN	NaN
11	NaN	NaN
12	NaN	NaN
13	NaN	NaN
14	NaN	NaN
15	NaN	NaN
16	NaN	NaN
17	NaN	NaN
18	NaN	NaN

```
19                                   NaN                                 NaN
20                                   NaN                                 NaN
21                                   NaN                                 NaN
22                                   NaN                                 NaN
23                                   NaN                                 NaN
24                                   NaN                                 NaN
25                                   NaN                                 NaN
26                                   NaN                                 NaN
27                                   NaN                                 NaN
28                                   NaN                                 NaN
29                                   NaN                                 NaN
30                                   NaN                                 NaN
31                                   NaN                                 NaN
32                                   NaN                                 NaN
33                                   NaN                                 NaN
34                                   NaN                                 NaN
35                                   NaN                                 NaN
36                                   NaN                                 NaN
37                                   NaN                                 NaN
38                                   NaN                                 NaN
39                                   NaN                                 NaN
40                                   NaN                                 NaN
41                                   NaN                                 NaN
42                                   NaN                                 NaN
43                                   NaN                                 NaN
44                                   NaN                                 NaN
45                                   NaN                                 NaN
46                                   NaN                                 NaN
47                                   NaN                                 NaN
48                                   NaN                                 NaN
49                                   NaN                                 NaN
50                                   NaN                                 NaN
51                                   NaN                                 NaN
52                                   NaN                                 NaN
53                                   NaN                                 NaN
54                                   NaN                                 NaN
55                                   NaN                                 NaN
56                                   NaN                                 NaN
57                                   NaN                                 NaN
58                                   NaN                                 NaN
59                                889.66                              854.98

    000_C1010.10.000_07 21 13.00_kg_1  00R_B3010.90.000_07 21 13.00_kg_1  \
0                                   NaN                                 NaN
1                                   NaN                                 NaN
2                                   NaN                                 NaN
3                                   NaN                                 NaN
```

4	NaN	NaN
5	NaN	NaN
6	NaN	NaN
7	NaN	NaN
8	NaN	NaN
9	NaN	NaN
10	NaN	NaN
11	NaN	NaN
12	NaN	NaN
13	NaN	NaN
14	NaN	NaN
15	NaN	NaN
16	NaN	NaN
17	NaN	NaN
18	NaN	NaN
19	NaN	NaN
20	NaN	NaN
21	NaN	NaN
22	NaN	NaN
23	NaN	NaN
24	NaN	NaN
25	NaN	NaN
26	NaN	NaN
27	NaN	NaN
28	NaN	NaN
29	NaN	NaN
30	NaN	NaN
31	NaN	NaN
32	NaN	NaN
33	NaN	NaN
34	NaN	NaN
35	NaN	NaN
36	NaN	NaN
37	NaN	NaN
38	NaN	NaN
39	NaN	NaN
40	NaN	NaN
41	NaN	NaN
42	NaN	NaN
43	NaN	NaN
44	NaN	NaN
45	NaN	NaN
46	NaN	NaN
47	NaN	NaN
48	NaN	NaN
49	NaN	NaN
50	NaN	NaN

51	NaN	NaN
52	NaN	NaN
53	NaN	NaN
54	NaN	NaN
55	NaN	NaN
56	NaN	NaN
57	NaN	NaN
58	NaN	NaN
59	127.47	420.29

	00R_B1020.20.000_07 51 13.00_kg_1
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	NaN
16	NaN
17	NaN
18	NaN
19	NaN
20	NaN
21	NaN
22	NaN
23	NaN
24	NaN
25	NaN
26	NaN
27	NaN
28	NaN
29	NaN
30	NaN
31	NaN
32	NaN
33	NaN
34	NaN
35	NaN

```
36                                 NaN
37                                 NaN
38                                 NaN
39                                 NaN
40                                 NaN
41                                 NaN
42                                 NaN
43                                 NaN
44                                 NaN
45                                 NaN
46                                 NaN
47                                 NaN
48                                 NaN
49                                 NaN
50                                 NaN
51                                 NaN
52                                 NaN
53                                 NaN
54                                 NaN
55                                 NaN
56                                 NaN
57                                 NaN
58                                 NaN
59                              315.22

[60 rows x 2090 columns]
```

[5]:
```
mapper = pd.read_excel('../Conversion/Mapping material names_20210324.
 ↪xlsx',header=2,usecols='B:U').replace(r'\n','', regex=True)
```

[6]:
```
name_conversion = pd.read_csv('name_conversion.csv')
building_name_conversion = pd.read_csv('building_type_name_conversion.csv')
```

[7]:
```
building_name_map = {k['Building Code']:k['Building Type'] for _,k in␣
 ↪building_name_conversion.iterrows()}
```

[8]:
```
name_map = {k.Code:k.Category for _,k in name_conversion.iterrows()}
```

[9]:
```
additional_categories_map = {v:k for k,v in {
    'Continuous Footings':'0CF',
    'Foundation Walls':'0FW',
    'Spread Footings':'0SF',
    'Column Piers':'0CP',
    'Columns Supporting Floors':'CSF',
    'Floor Girders and Beams':'FGB',
    'Floor Trusses':'0FT',
    'Floor Joists':'0FJ',
```

```
    'Columns Supporting Roofs':'CSR',
    'Roof Girders and Beams':'RGB',
    'Roof Trusses':'ORT',
    'Roof Joists':'ORJ',
    'Parking Bumpers':'OPB',
    'Precast Concrete Stair Treads':'PCS',
    'Roof Curbs':'ORC',
    'Exterior Wall Construction':'EWC',
    'Composite Decking':'CPD',
    'Cast-in-Place concrete':'CIC',
    'Floor Structural Frame':'FSF',
    'Associated Metal Fabrications':'AMF',
    'Floor Construction Supplementary Components':'FCS',
    'Roof Construction Supplementary Components':'RCS',
    'Residential Elevators':'ORE',
    'Vegetated Low-Slope Roofing':'VLR',
    'Swimming Pools':'SWP',
    'Excavation Soil Anchors':'ESA',
    'Floor Trusses':'FTS',
    'Roof Window and Skylight Performance':'RWS'}.items()
}

additional_categories_map['OFT'] = 'Floor Trusses'
```

[10]:
```
def get_material_name(l):
    try:
        split = re.split('[_\.\ ]',l) #Split up the code into its requisite␣
 ↪parts
        result = mapper[mapper['Unnamed: 7'] == split[1]+'.'+split[2]] #Filter␣
 ↪by Level 4 Master Format
        if len(result) == 0:
            result = mapper #If that code does not exist in the table, reset
        if len(result) == 1:
            return result['Mapping Table'].values[0] #If it maps to exactly one␣
 ↪value, return that. We do this check after every step
        if split[3] != '000': #Check if there is an additional code, and if so␣
 ↪filter by that
            result = result[result['Level 5\n'] ==␣
 ↪additional_categories_map[split[3]]]
            if len(result) == 1:
                return result['Mapping Table'].values[0]

        #Now filter by UniFormat.
        #Filter only by the level of UniFormat present. If the code is XX 00␣
 ↪00, for example, then we only have Level 1.
        if int(split[5]) == 0:
```

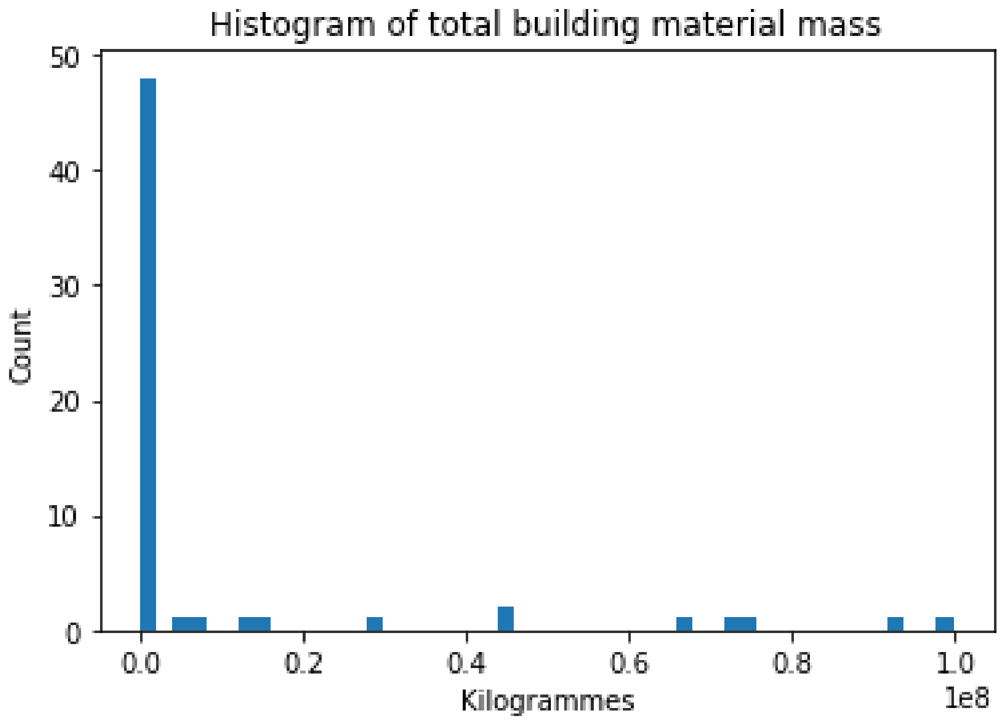
```
            result = result[result['Unnamed: 12'] == f'{split[4]} 00 00']
            if len(result) == 1:
                return result['Mapping Table'].values[0]
        elif int(split[6]) == 0:
            result = result[(result['Unnamed: 14'] == f'{split[4]} {split[5]}␣
↪00') | (result['Unnamed: 16'] == f'{split[4]} {split[5]} 00')]
            if len(result) == 1:
                return result['Mapping Table'].values[0]
        else:
            result = result[result['Unnamed: 18'] == f'{split[4]} {split[5]}␣
↪{split[6]}']
            if len(result) == 1:
                return result['Mapping Table'].values[0]

        #If we couldn't find it, or there is an unspecified edge case, return␣
↪None.
        if len(result) == 0:
            return None

        #If there are multiple results but they all map to the same material,␣
↪return that material.
        if all(element == result['Mapping Table'].values[0] for element in␣
↪result['Mapping Table'].values):
            return result['Mapping Table'].values[0]
        else:
            return None
    except:
        return None
```

2 1. Plot sample figures

Here we plot building material mass, and volume histograms.

[11]:
```
plt.hist(df[[c for c in df.columns if 'kg' in c]].sum(axis=1),bins=50);
plt.title('Histogram of total building material mass')
plt.xlabel('Kilogrammes')
plt.ylabel('Count');
```

[11]: Text(0, 0.5, 'Count')



3 2. Investigate a specific material

In this example, we use the helper function `get_material_name()` to select columns which match `steel`. Then, we calculate the average amount of steel used by floor, produce a table of values by Level 3 MasterFormat only, and calculate the average values for these by year in the dataset.

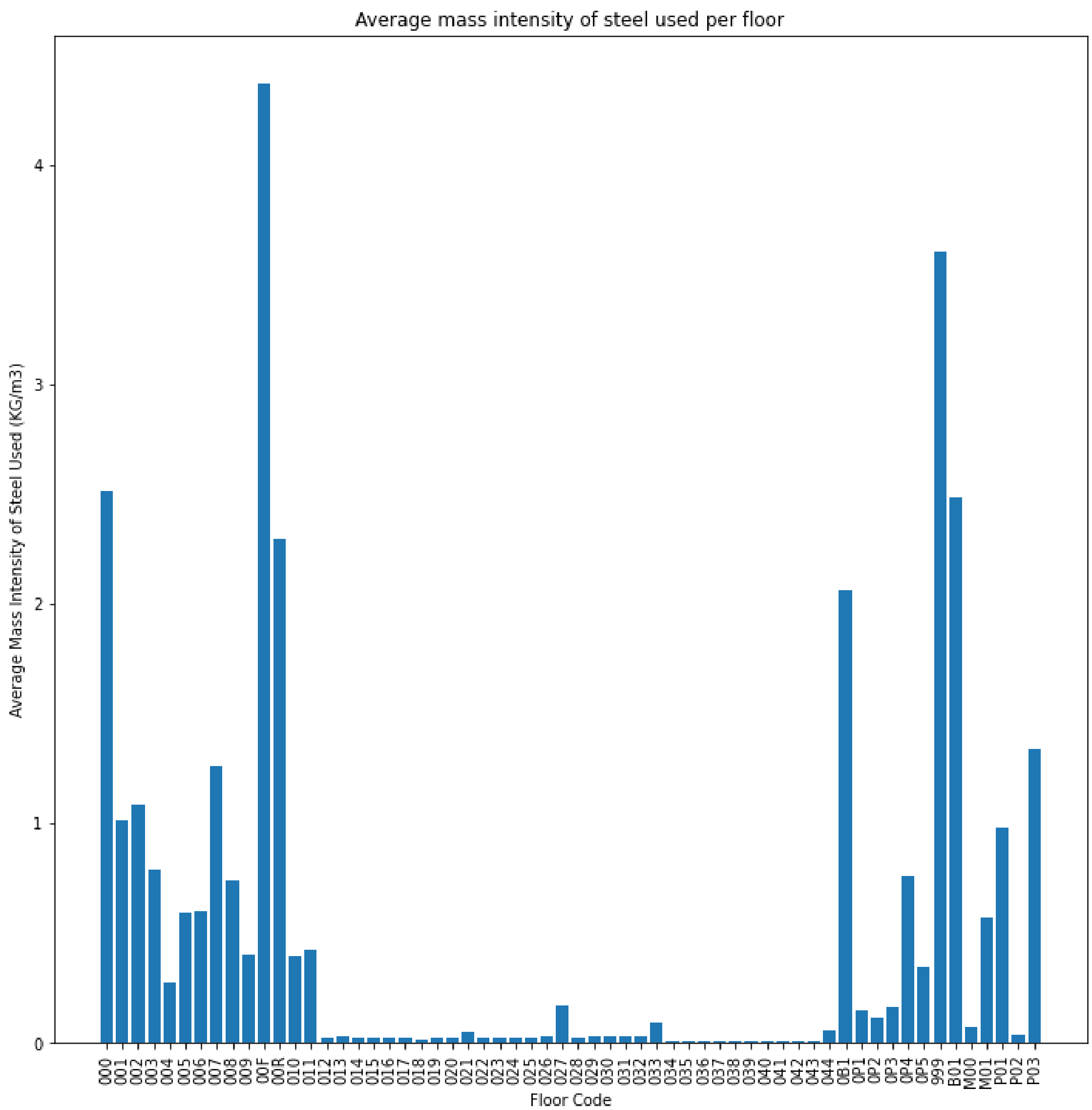
[12]:
```
material = 'steel'
cols = []
for column in df.columns[7:]: #Iterate through columns that represent materials
    if get_material_name(column) == 'steel' and 'kg' in column: #If that column␣
 ↪represents steel and is a mass value:
        cols.append(column) #Append to cols
```

[13]:
```
steel_df = df[df.columns[1:7].to_list() + cols].fillna(0) #Select only the␣
 ↪heading columns and the columns related to steel
```

[14]:
```
grouping_function = lambda x: x.split('_')[0] #This function takes in a full␣
 ↪column name, like "000_G2010.20.000_03 00 00.00_m3_1", and returns only the␣
 ↪floor.
to_draw = steel_df[cols].groupby(grouping_function,axis=1).sum().replace(0,np.
 ↪NaN).div(df['Gross Floor Area'],axis='rows').mean()
plt.figure(figsize=(12,12))
plt.bar(to_draw.keys(), to_draw.values)
```

```python
plt.xticks(rotation=90)
plt.title('Average mass intensity of steel used per floor')
plt.ylabel('Average Mass Intensity of Steel Used (KG/m3)')
plt.xlabel('Floor Code');
```

[14]: Text(0.5, 0, 'Floor Code')



Now, we will aggregate to Level 3 MasterFormat codes, and display these values for the first three entries.

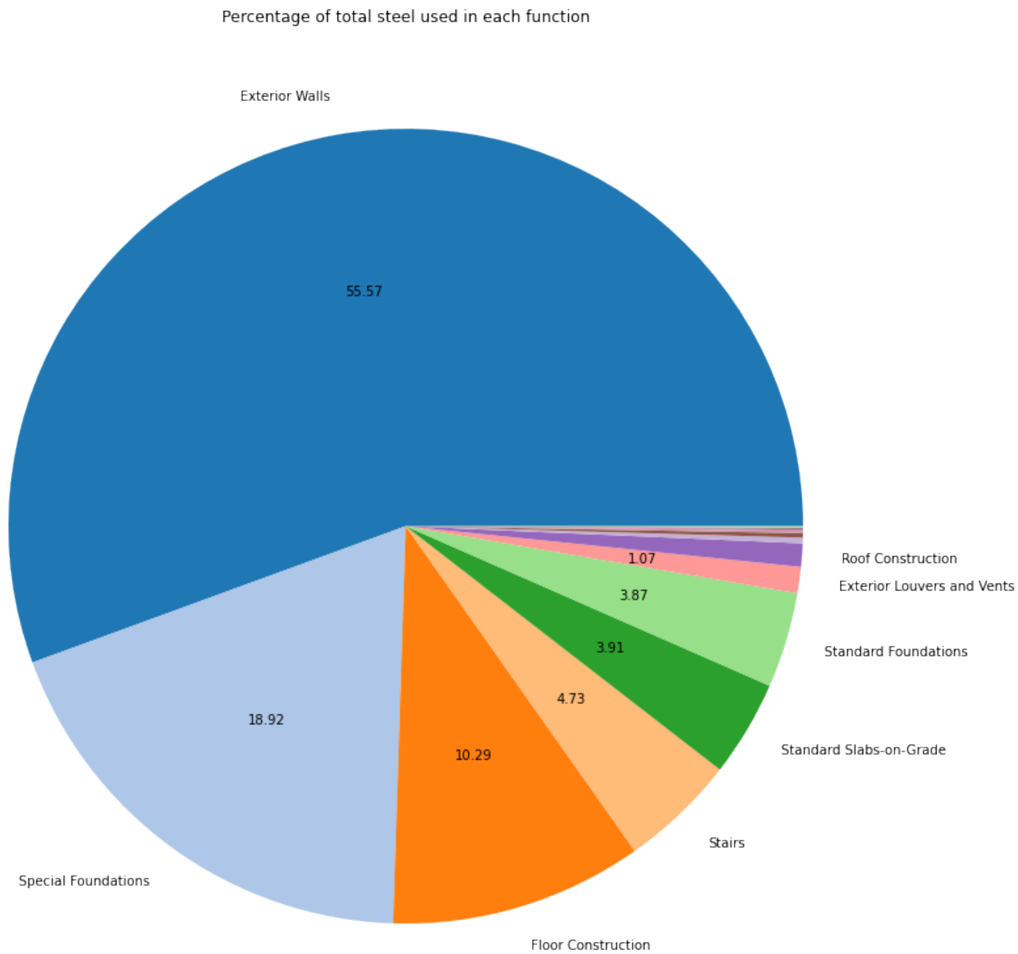
[15]:
```python
f = lambda x: name_map[re.split('[_\.\ ]',x)[1]] #This function takes in a full column name and returns only the Level 3 MasterFormat code.
steel_general_df = steel_df[cols].groupby(f,axis=1).sum()
```

[16]:
```
steel_general_df.mean().sort_values(ascending=False)
```

[16]:
```
Exterior Walls                 77353.438405
Special Foundations            26338.802948
Floor Construction             14318.543691
Stairs                          6580.472545
Standard Slabs-on-Grade         5445.154282
Standard Foundations            5392.813457
Exterior Louvers and Vents      1488.385267
Roof Construction               1317.269238
Interior Specialties             316.689885
Vertical Conveying Systems       253.659106
Site Development                 119.830250
Roadways                          86.978000
Exterior Doors and Grilles        75.505465
Structural Slabs-on-Grade         45.919862
Pits and Bases                    29.302840
Building Subdrainage              13.059478
Interior Doors                    11.526000
Roofing                            5.474165
dtype: float64
```

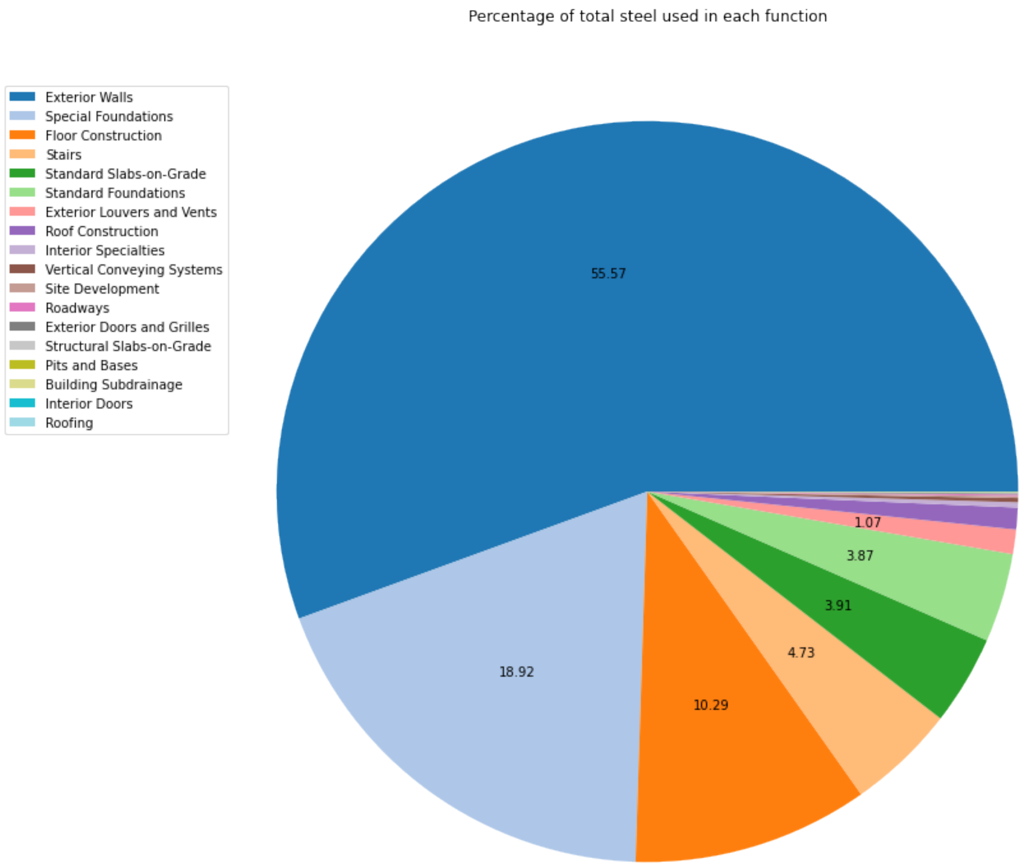
3.1 Pie chart version A: on-pie chart labels for all > 1%

[17]:
```
def my_autopct(pct):
    return ('%.2f' % pct) if pct > 1 else ''
to_plot = steel_general_df.mean().sort_values(ascending=False)
to_plot.plot.pie(figsize=(12,12),colormap='tab20',autopct=my_autopct,labels=[k␣
 ↪if v > 1000 else '' for k,v in to_plot.items()])
plt.ylabel('')
plt.title('Percentage of total steel used in each function');
# plt.legend(loc='center left',bbox_to_anchor=(-0.20, 0.75));
plt.tight_layout();
```



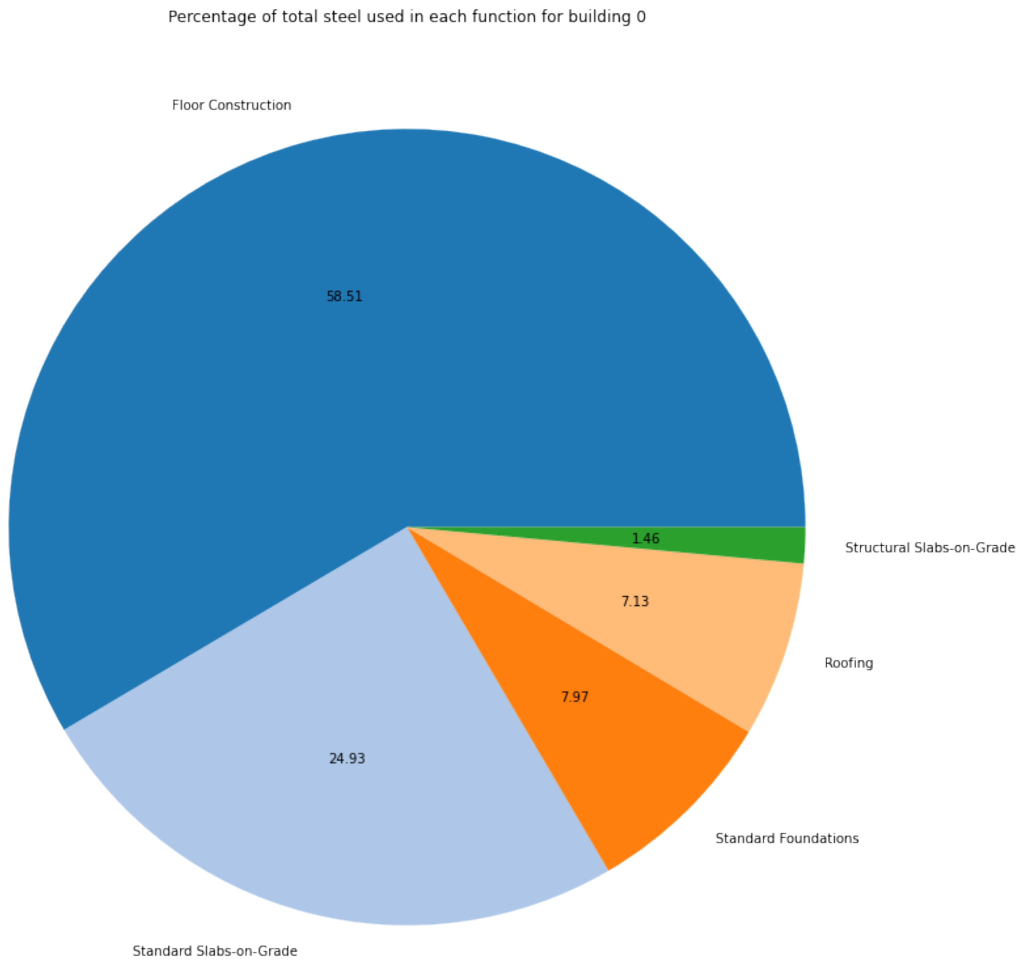
3.2 Pie version B: external legend with slice labels

[18]:
```
def my_autopct(pct):
    return ('%.2f' % pct) if pct > 1 else ''
to_plot = steel_general_df.mean().sort_values(ascending=False)
to_plot.plot.
↪pie(figsize=(12,12),colormap='tab20',autopct=my_autopct,labeldistance=None)
plt.ylabel('')
plt.title('Percentage of total steel used in each function');
plt.legend(loc='center left',bbox_to_anchor=(-0.20, 0.75));
plt.tight_layout();
```



We can produce a pie chart for a single building, also.

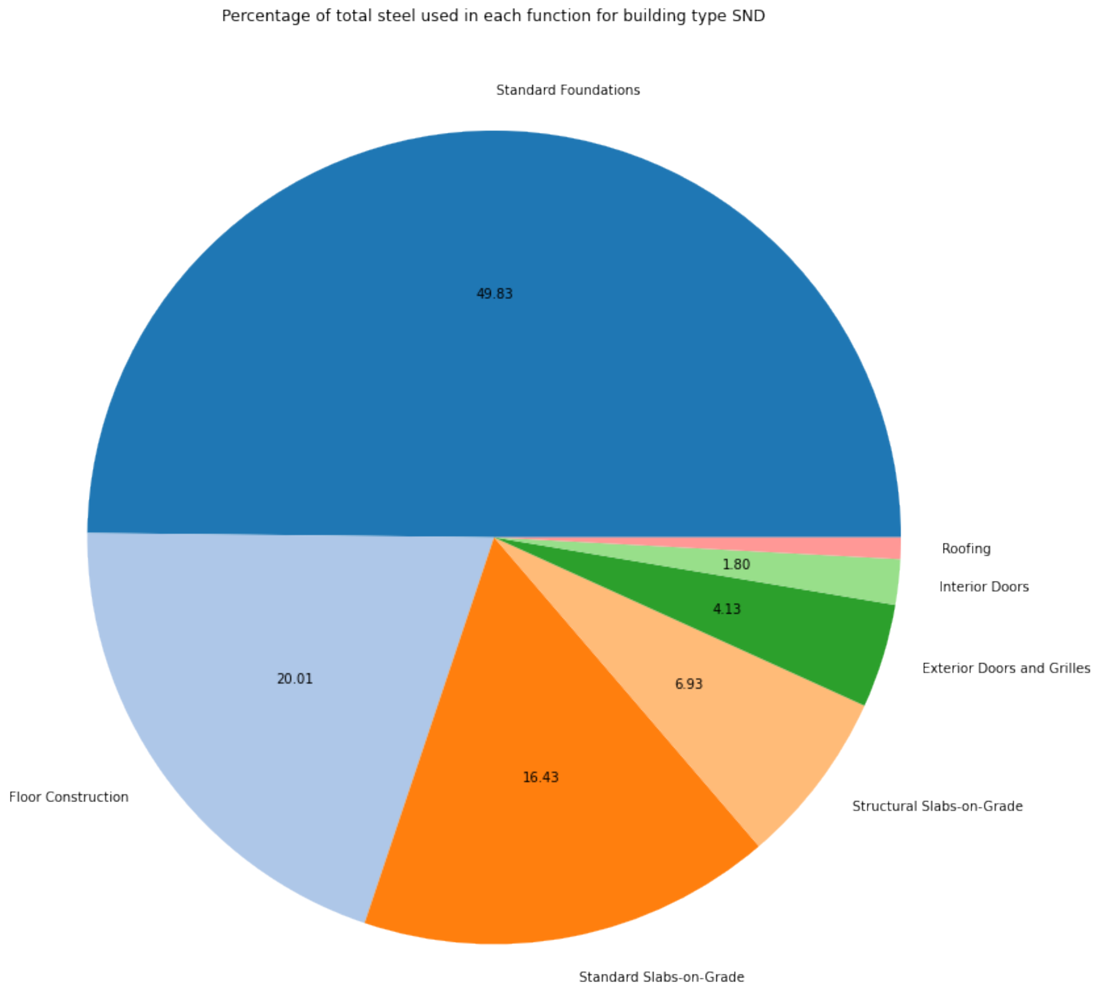
[19]:
```
BUILDING_ID = 0

def my_autopct(pct):
    return ('%.2f' % pct) if pct > 1 else ''
to_plot = steel_general_df.loc[BUILDING_ID,:].sort_values(ascending=False)
to_plot.plot.pie(figsize=(12,12),colormap='tab20',autopct=my_autopct)
plt.ylabel('')
plt.title(f'Percentage of total steel used in each function for building␣
 ↪{BUILDING_ID}');
plt.tight_layout();
```



Or an entire class of building:

[20]:
```
steel_general_df = pd.concat([steel_df['Building Type'],steel_df[cols].
 ↪groupby(f,axis=1).sum()],axis=1)
BUILDING_TYPE = 'SND'

def my_autopct(pct):
    return ('%.2f' % pct) if pct > 1 else ''
to_plot = steel_general_df[steel_general_df['Building Type'] ==␣
 ↪BUILDING_TYPE][steel_general_df.columns[1:]].mean().
 ↪sort_values(ascending=False)
to_plot.plot.pie(figsize=(12,12),colormap='tab20',autopct=my_autopct)
plt.ylabel('')
plt.title(f'Percentage of total steel used in each function for building type␣
 ↪{BUILDING_TYPE}');
```

```
plt.tight_layout();
```



We can also calculate the average for each Level 3 MasterFormat code by year of construction:

[21]:
```
steel_general_df = pd.concat([steel_df[headings[1:]],steel_df[cols].
↪groupby(f,axis=1).sum()],axis=1)
steel_general_df.groupby('Construction Date').mean()
```

[21]:
```
                   Gross Floor Area  Building Subdrainage  \
Construction Date
1913                     161.080000              0.000000
1917                     199.930000              0.000000
1969                     373.605000              0.000000
1988                   21934.000000              0.000000
2007                   73600.000000              0.000000
```

2009	73083.000000	0.000000
2011	11282.500000	0.000000
2016	30345.000000	0.000000
2017	39392.013333	0.000000
2018	43560.635000	391.784342
2019	83.100000	0.000000
2020	418.528571	0.000000
2021	445.404444	0.000000

Construction Date	Exterior Doors and Grilles	Exterior Louvers and Vents \
1913	0.000000	0.000
1917	0.000000	0.000
1969	0.000000	0.000
1988	0.000000	0.000
2007	0.000000	0.000
2009	0.000000	88591.000
2011	0.000000	0.000
2016	0.000000	0.000
2017	0.000000	0.000
2018	0.000000	356.058
2019	53.357788	0.000
2020	507.870020	0.000
2021	25.607778	0.000

Construction Date	Exterior Walls	Floor Construction	Interior Doors \
1913	0.000000e+00	0.000000	0.00
1917	0.000000e+00	0.000000	0.00
1969	0.000000e+00	0.000000	0.00
1988	5.039204e+03	747.739297	0.00
2007	1.752312e+06	32828.900000	0.00
2009	2.658847e+06	77762.100000	0.00
2011	1.085611e+05	93517.011675	0.00
2016	0.000000e+00	16993.067500	0.00
2017	2.267603e+03	151410.914567	0.00
2018	5.416500e+02	32357.624000	0.00
2019	0.000000e+00	0.000000	0.00
2020	0.000000e+00	495.000286	0.00
2021	0.000000e+00	120.575836	19.21

Construction Date	Interior Specialties	Pits and Bases	Roadways \
1913	0.000000	0.00000	0.000
1917	0.000000	0.00000	0.000
1969	0.000000	0.00000	0.000
1988	0.000000	0.00000	0.000

2007	16665.000000	0.00000	0.000
2009	0.000000	0.00000	3242.050
2011	0.000000	180.15750	988.315
2016	0.000000	235.10750	0.000
2017	0.000000	309.21346	0.000
2018	1168.196545	0.00000	0.000
2019	0.000000	0.00000	0.000
2020	0.000000	0.00000	0.000
2021	0.000000	0.00000	0.000

	Roof Construction	Roofing	Site Development \
Construction Date			
1913	0.000000	0.000000	0.000000
1917	0.000000	0.000000	0.000000
1969	0.000000	0.000000	0.000000
1988	0.000000	0.000000	0.000000
2007	2249.000000	0.000000	0.000000
2009	63740.722253	0.000000	0.000000
2011	0.000000	0.000000	1698.740000
2016	0.000000	0.000000	0.000000
2017	2272.634333	0.000000	1264.111667
2018	3114.264500	0.000000	0.000000
2019	0.000000	0.000000	0.000000
2020	0.000000	0.000000	0.000000
2021	0.000000	9.123608	0.000000

	Special Foundations	Stairs	Standard Foundations \
Construction Date			
1913	0.000000	0.000000	0.000000
1917	0.000000	0.000000	0.000000
1969	0.000000	0.000000	0.000000
1988	0.000000	5677.162679	67016.749257
2007	122069.070000	86571.370000	0.000000
2009	0.000000	0.000000	92590.750000
2011	11019.437500	2180.730000	10048.588750
2016	188421.220000	23831.815000	7123.286250
2017	209661.285467	29604.705000	18289.127707
2018	215196.967750	80870.307500	27425.348750
2019	0.000000	0.000000	79.688946
2020	0.000000	0.000000	344.180912
2021	0.000000	0.000000	483.625617

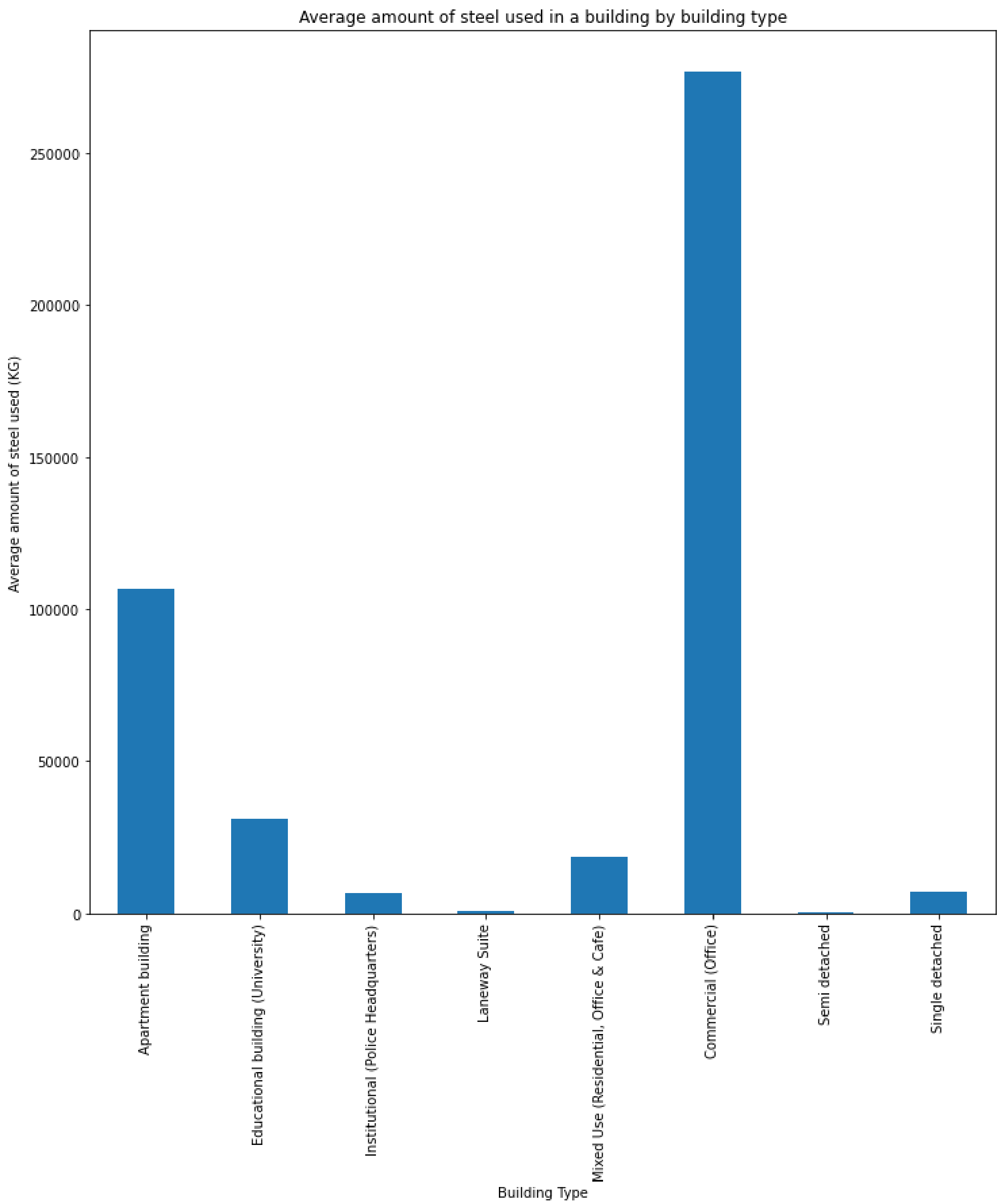
	Standard Slabs-on-Grade	Structural Slabs-on-Grade \
Construction Date		
1913	48.162700	0.000000
1917	0.000000	20.818800
1969	0.000000	98.436400

```
1988                      24922.610789              0.000000
2007                      68246.330000              0.000000
2009                      58354.545000              0.000000
2011                      17521.985000              0.000000
2016                      19159.102500              0.000000
2017                      27702.060870              0.000000
2018                       6118.290000              0.000000
2019                        171.655317              0.000000
2020                         51.147571             29.765514
2021                        163.971736             64.698375

                   Vertical Conveying Systems
Construction Date
1913                                 0.000000
1917                                 0.000000
1969                                 0.000000
1988                               334.146341
2007                              7925.800000
2009                              6959.600000
2011                                 0.000000
2016                                 0.000000
2017                                 0.000000
2018                                 0.000000
2019                                 0.000000
2020                                 0.000000
2021                                 0.000000
```

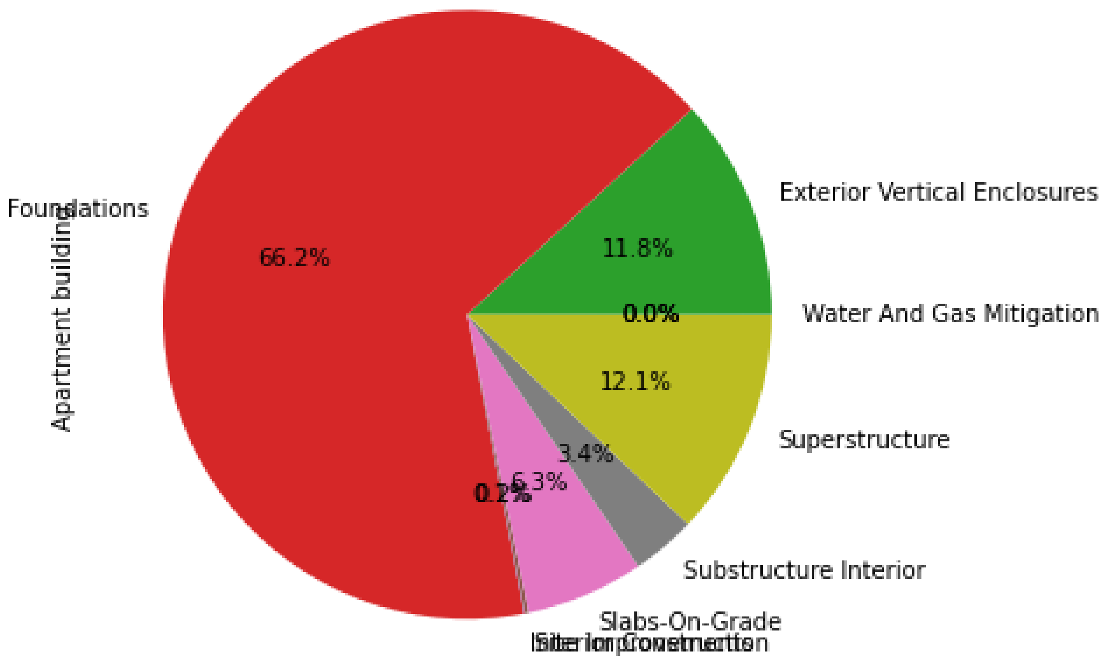
We can get the average amount of steel in KG used per building type:

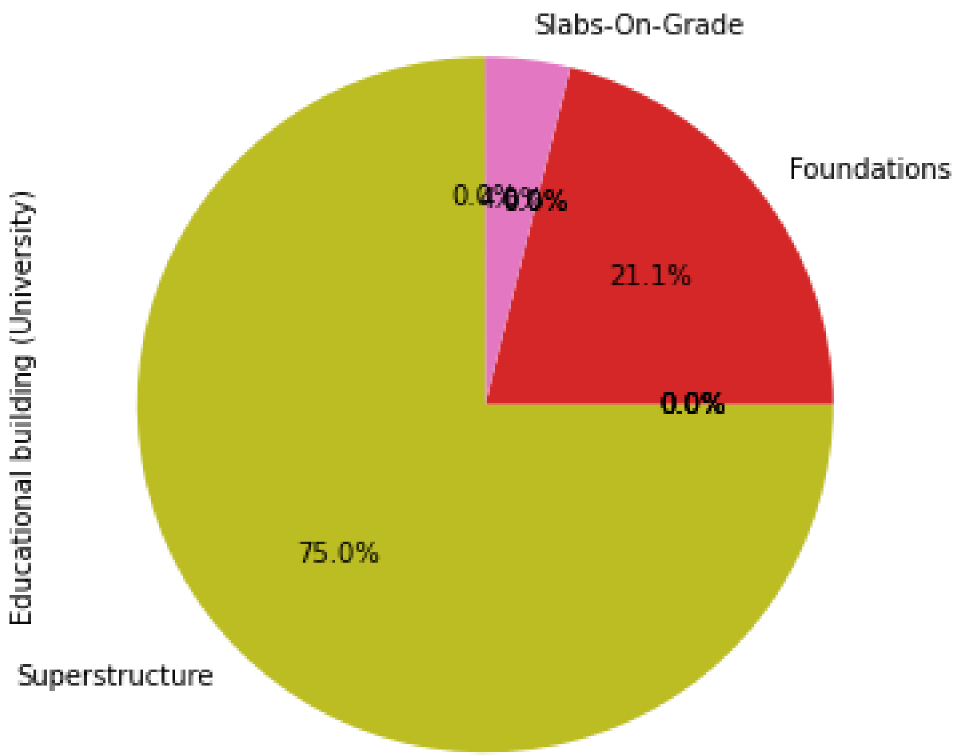
[22]:
```
steel_general_df.groupby('Building Type').sum().mean(axis=1).
 ↪rename(index=building_name_map).plot(kind='bar',figsize=(12,12))
plt.ylabel('Average amount of steel used (KG)')
plt.title('Average amount of steel used in a building by building type');
```

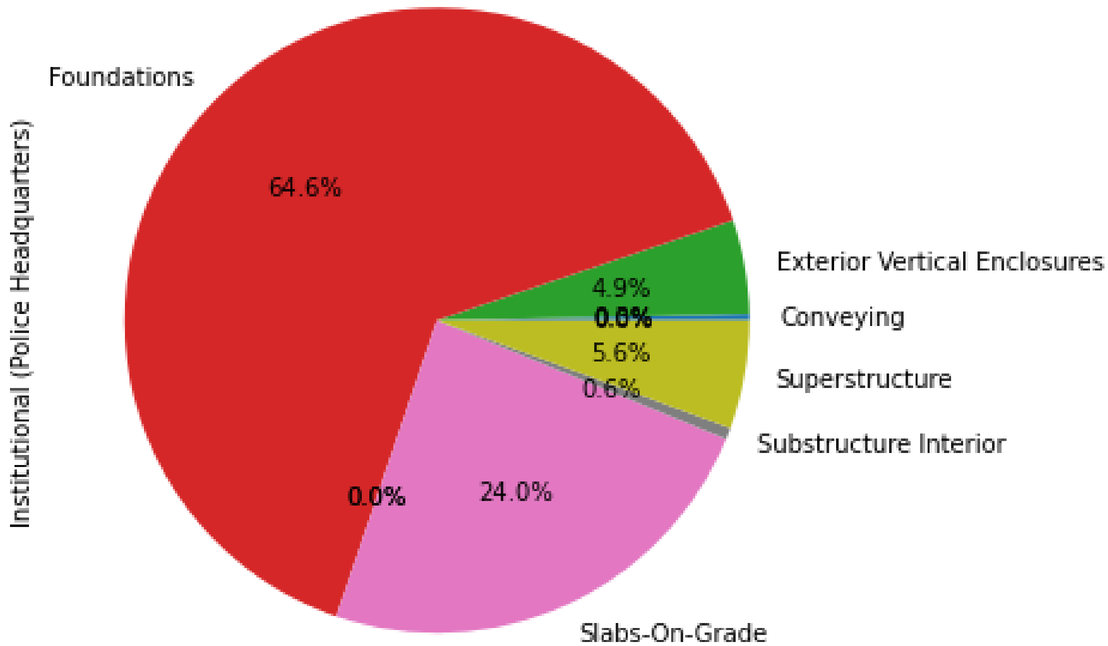
[22]:
```
Text(0.5, 1.0, 'Average amount of steel used in a building by building type')
```

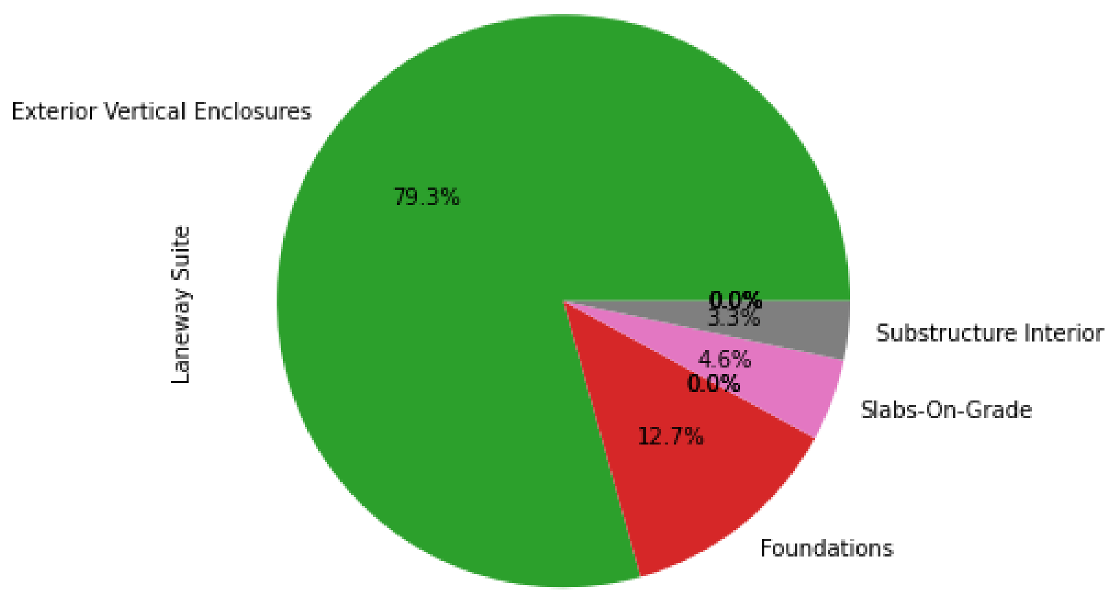


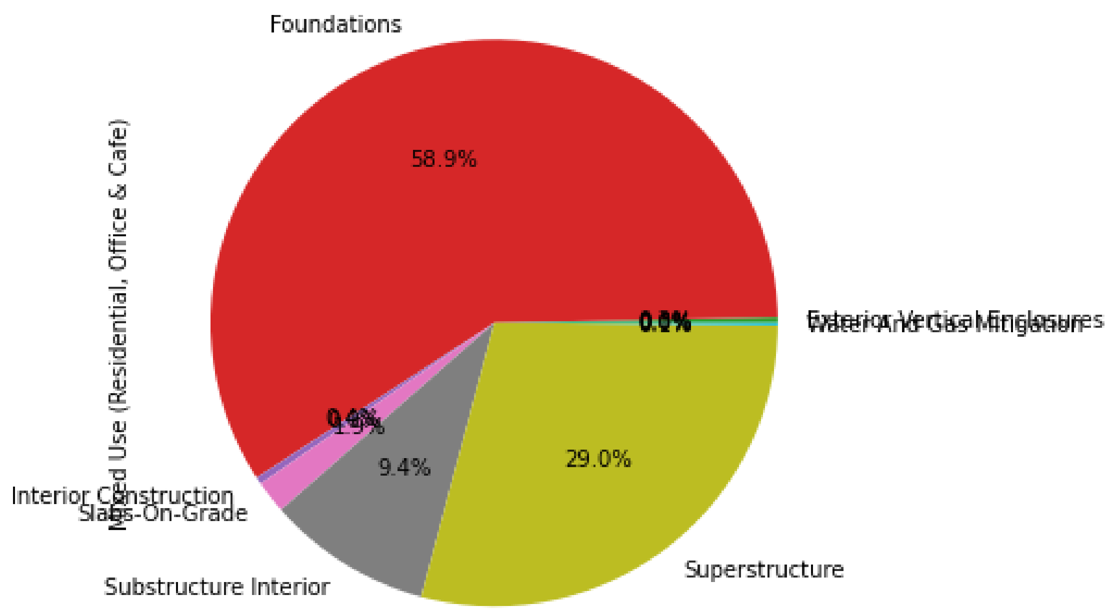
[23]:
```
f = lambda x: name_map[re.split('[_\.\ ]',x)[1][0:3]] #From a full code, return␣
↪only the use code and uncertainty code.
tdf = pd.concat([df['Building Type'],df[cols].groupby(f,axis=1).sum()],axis=1).
↪groupby('Building Type').mean().rename(index=building_name_map).transpose()
for i,k in enumerate(tdf.columns.values):
    tdf.plot.pie(y=k,figsize=(6,6),autopct='%1.1f%%',legend=False);
```

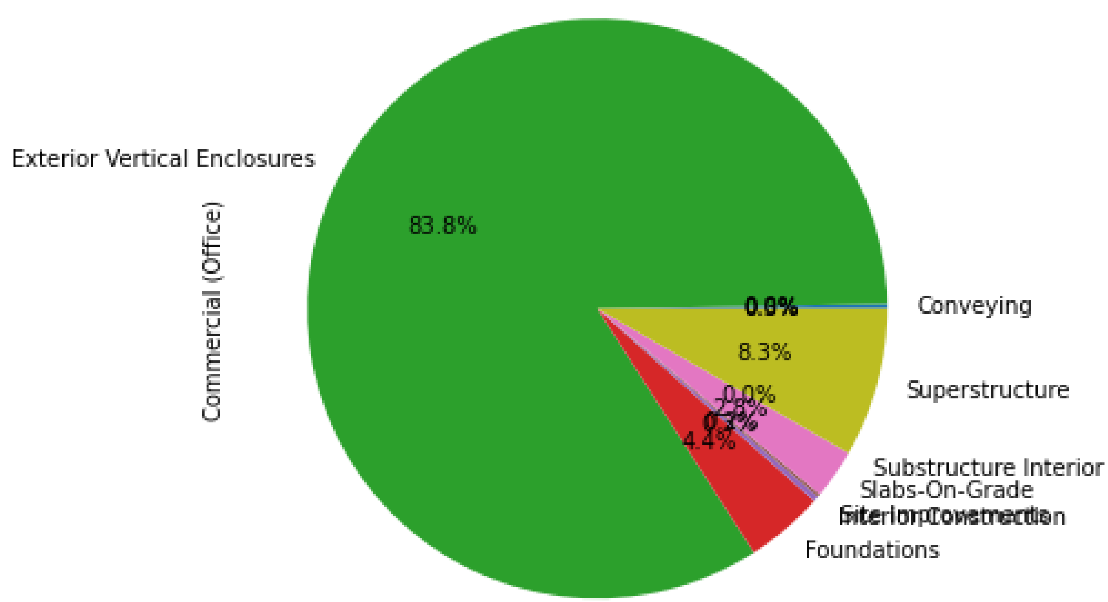

Apartment building
Foundations
66.2%
Exterior Vertical Enclosures
11.8%
Water And Gas Mitigation
0.0%
12.1%
Superstructure
3.4%
6.3%
Substructure Interior
Slabs-On-Grade
Interior Construction

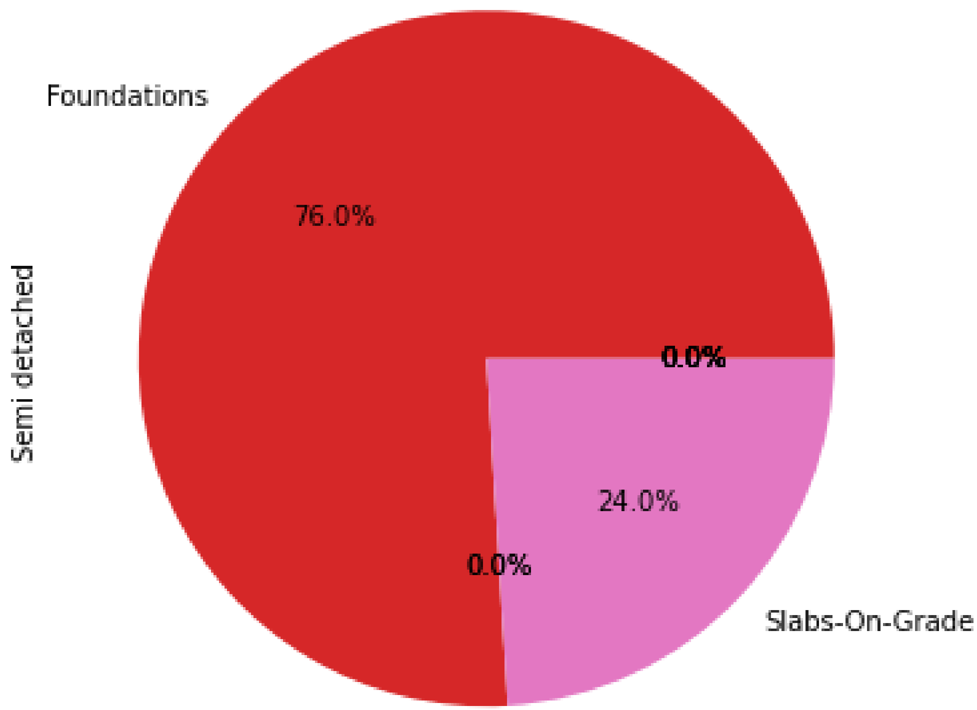

Slabs-On-Grade
Foundations
Educational building (University)
Superstructure
21.1%
75.0%
0.0%
0.0%

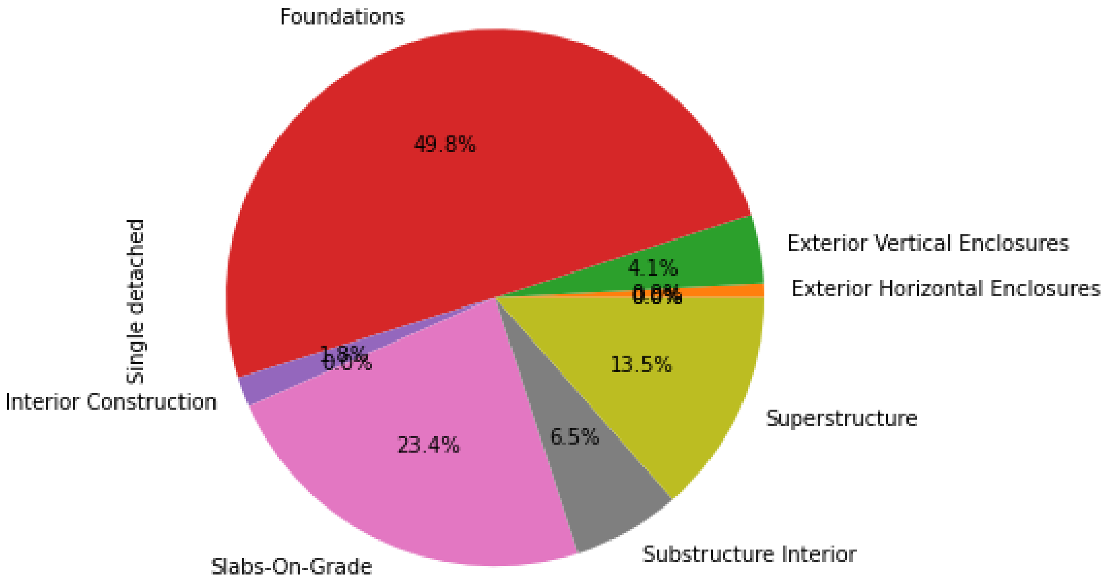

Institutional (Police Headquarters)
Foundations
64.6%
Exterior Vertical Enclosures
4.9%
0.0%
Conveying
5.6%
Superstructure
0.6%
Substructure Interior
0.0%
24.0%
Slabs-On-Grade


Laneway Suite
Exterior Vertical Enclosures
79.3%
0.0%
3.3%
Substructure Interior
4.6%
Slabs-On-Grade
0.0%
12.7%
Foundations


Foundations
58.9%
Mixed Use (Residential, Office & Cafe)
29.0%
Superstructure
9.4%
Substructure Interior
Slabs-On-Grade
Interior Construction
Exterior Vertical Enclosures
Water And Gas Mitigation


Exterior Vertical Enclosures
83.8%
Commercial (Office)
Conveying
8.3%
Superstructure
0.0%
Substructure Interior
Slabs-On-Grade
4.4%
Foundations


Foundations
76.0%
Semi detached
0.0%
24.0%
0.0%
Slabs-On-Grade



4 3. Uncertainty by Building Type

In this section, we look at the uncertainty code associated with each column. We collect these by building type and then report the number of each value per type of building.

[24]:
```
uncertainty_level = {}
for k,v in df.iterrows():
    #Initialise empty lists for each building type as they occur
    if v['Building Type'] not in uncertainty_level.keys():
        uncertainty_level[v['Building Type']] = []
    #Append the uncertainty value for each column that is non-NaN
    for key in v[~v.isna()].keys()[7:]:
        uncertainty_level[v['Building Type']].append(key.split('_')[-1])
```

[25]:
```
from collections import Counter
```

[26]:
```
for k,v in uncertainty_level.items():
    uncertainty_level[k] = Counter(v) #Construct a Counter object per building␣
 ↪type
```

[27]:
```
uncertainty_level
```

[27]:
```
{'SND': Counter({'1': 1720, '2': 711, '4': 349}),
 'OFF': Counter({'1': 494, '3': 307}),
```

```
 'APB': Counter({'1': 1171, '2': 1, '3': 971}),
 'SMD': Counter({'1': 191, '2': 61, '4': 27}),
 'EDU': Counter({'1': 93, '3': 24, '2': 6}),
 'INS': Counter({'1': 90, '3': 77, '2': 1}),
 'MIX': Counter({'1': 363, '3': 276}),
 'LNW': Counter({'2': 46, '1': 142, '4': 19})}
```

Next, we aggregate columns by use code and uncertainty combined, and report the average by building type.

[28]:
```
f = lambda x: name_map[re.split('[_\.\ ]',x)[1][0]] + '/' + x.split('_')[-1].
↪split('.')[0] #From a full code, return only the use code and uncertainty␣
↪code.
by_function_df = pd.concat([df[headings[1:]],df[cols].groupby(f,axis=1).
↪sum()],axis=1)
```

[29]:
```
by_function_df.groupby('Building Type').mean().rename(index=building_name_map)
```

[29]:
```
                                        Construction Date  Gross Floor Area  \
Building Type
Apartment building                                2015.80      45505.412000
Educational building (University)                 2016.50       7901.000000
Institutional (Police Headquarters)               1988.00      21934.000000
Laneway Suite                                     2020.00        150.010000
Mixed Use (Residential, Office & Cafe)            2018.00      33975.250000
Commercial (Office)                               2009.00      52643.666667
Semi detached                                     1994.75        236.615000
Single detached                                   2015.60        465.227000

                                        Interiors/1  Interiors/2  Services/1  \
Building Type
Apartment building                       192.108455        0.000    0.000000
Educational building (University)          0.000000        0.000    0.000000
Institutional (Police Headquarters)        0.000000        0.000  334.146341
Laneway Suite                              0.000000        0.000    0.000000
Mixed Use (Residential, Office & Cafe)  1375.850817        0.000    0.000000
Commercial (Office)                     5555.000000        0.000 4961.800000
Semi detached                              0.000000        0.000    0.000000
Single detached                            0.000000       17.289    0.000000

                                             Shell/1  Shell/2       Shell/3  \
Building Type
Apartment building                      4.767977e+04   0.0000  42919.090000
Educational building (University)       2.214476e+05   0.0000   3218.892500
Institutional (Police Headquarters)     1.297866e+02   0.0000  10716.366983
Laneway Suite                           7.359987e+02   0.0000      0.000000
Mixed Use (Residential, Office & Cafe)  4.477775e+03   0.0000  94212.560000
```

```
Commercial (Office)                          1.619260e+06   0.0000  29491.141667
Semi detached                                0.000000e+00   0.0000      0.000000
Single detached                              1.154629e+02  55.4561      0.000000

                                              Shell/4  Sitework/1   Sitework/3  \
Building Type
Apartment building                           0.000000     225.295   533.172000
Educational building (University)            0.000000       0.000     0.000000
Institutional (Police Headquarters)          0.000000       0.000     0.000000
Laneway Suite                                0.000000       0.000     0.000000
Mixed Use (Residential, Office & Cafe)       0.000000       0.000     0.000000
Commercial (Office)                          0.000000       0.000  2872.053333
Semi detached                                0.000000       0.000     0.000000
Single detached                              6.686572       0.000     0.000000

                                             Substructure/1  Substructure/2  \
Building Type
Apartment building                            192895.616600        0.000000
Educational building (University)                  0.000000        0.000000
Institutional (Police Headquarters)                0.000000        0.000000
Laneway Suite                                    113.909606       77.689805
Mixed Use (Residential, Office & Cafe)        151968.510000        0.000000
Commercial (Office)                                0.000000        0.000000
Semi detached                                     82.653250       11.036450
Single detached                                  676.023563       68.474865

                                             Substructure/3  Substructure/4
Building Type
Apartment building                             95502.505000        0.000000
Educational building (University)              74976.547506        0.000000
Institutional (Police Headquarters)            92557.312757        0.000000
Laneway Suite                                      0.000000        0.000000
Mixed Use (Residential, Office & Cafe)         84478.698683        0.000000
Commercial (Office)                           127794.205833        0.000000
Semi detached                                      0.000000        4.246275
Single detached                                    0.000000       20.342905
```

Next, we report the total amount of material falling under each uncertainty code by year of construction.

[30]:
```
f = lambda x: x.split('_')[-1].split('.')[0] #Select only the uncertainty code.
pd.concat([df[headings[1:]],df[cols].groupby(f,axis=1).sum()],axis=1).
 ↪groupby('Construction Date').mean()
```

[30]:
```
                   Gross Floor Area            1          2            3  \
Construction Date
1913                     161.080000  4.816270e+01   0.000000     0.000000
```

```
1917                     199.930000  0.000000e+00   20.818800        0.000000
1969                     373.605000  0.000000e+00   98.436400        0.000000
1988                   21934.000000  4.639329e+02    0.000000   103273.679739
2007                   73600.000000  1.811981e+06    0.000000   276886.770000
2009                   73083.000000  2.894837e+06    0.000000   155250.235000
2011                   11282.500000  2.006509e+05    0.000000    45065.083750
2016                   30345.000000  1.334946e+05    0.000000   122269.048750
2017                   39392.013333  3.163813e+05    0.000000   126400.375837
2018                   43560.635000  1.853796e+05    0.000000   182160.889342
2019                      83.100000  1.167928e+02  187.909221        0.000000
2020                     418.528571  1.226472e+03  112.815100        0.000000
2021                     445.404444  7.399847e+02  133.566586        0.000000

                           4
Construction Date
1913                0.000000
1917                0.000000
1969                0.000000
1988                0.000000
2007                0.000000
2009                0.000000
2011                0.000000
2016                0.000000
2017                0.000000
2018                0.000000
2019                0.000000
2020               88.677571
2021               13.261700
```

5 4. Material Intensity

We can easily calculate material intensity by dividing columns which are measured in kilograms by the `Gross Floor Area`:

[31]:
```
kilogram_columns = [d for d in df.columns if 'kg' in d]
df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
```

[32]:
```
kilogram_columns = [d for d in df.columns if 'kg' in d]
df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
f = lambda x: name_map[re.split('[_\.\ ]',x)[1][0:3]]
pd.concat([df[headings[1:]],df_mi[kilogram_columns].groupby(f,axis=1).
 ↪sum()],axis=1)[df['Building Type'] == 'SND']
```

[32]:
```
   Country City Quality / Stage of Data  Construction Date Building Type  \
0       CA  TOR                   00IFC               2021           SND
1       CA  TOR                   00IFC               2021           SND
2       CA  TOR                   00IFC               2021           SND
```

3	CA	TOR	00IFC	2021	SND
6	CA	TOR	00IFC	2021	SND
7	CA	TOR	00IFC	2021	SND
8	CA	TOR	00IFC	2021	SND
9	CA	TOR	00IFC	2021	SND
12	CA	TOR	00IFC	2021	SND
13	CA	TOR	00IFC	2021	SND
14	CA	TOR	00IFC	2021	SND
15	CA	TOR	00IFC	2021	SND
16	CA	TOR	00IFC	1969	SND
17	CA	TOR	00IFC	1969	SND
18	CA	TOR	00IFC	2021	SND
19	CA	TOR	00IFC	2021	SND
20	CA	TOR	00IFC	2020	SND
21	CA	TOR	00IFC	2021	SND
22	CA	TOR	00IFC	2021	SND
24	CA	TOR	00IFC	2021	SND
25	CA	TOR	00IFC	2021	SND
27	CA	TOR	00IFC	2021	SND
28	CA	TOR	00IFC	2021	SND
30	CA	TOR	00IFC	2021	SND
31	CA	TOR	00IFC	2021	SND
32	CA	TOR	00IFC	2020	SND
34	CA	TOR	00IFC	2021	SND
35	CA	TOR	00IFC	2021	SND
36	CA	TOR	00IFC	2021	SND
37	CA	TOR	00IFC	2020	SND
38	CA	TOR	00IFC	2021	SND
40	CA	TOR	00IFC	2021	SND
41	CA	TOR	00IFC	1913	SND
42	CA	TOR	00IFC	2021	SND
43	CA	TOR	00IFC	2021	SND
44	CA	TOR	00IFC	2021	SND
45	CA	TOR	00IFC	2021	SND
46	CA	TOR	00IFC	2021	SND
48	CA	TOR	00IFC	2020	SND
49	CA	TOR	00IFC	2021	SND

	Gross Floor Area	Conveying	Exterior Horizontal Enclosures	\
0	521.18	0.0	11.137992	
1	389.24	0.0	5.461939	
2	411.64	0.0	3.786074	
3	269.56	0.0	6.503479	
6	445.99	0.0	11.933511	
7	438.45	0.0	12.707195	
8	714.07	0.0	12.865930	
9	343.24	0.0	4.300619	

12	226.89	0.0	12.424245
13	611.73	0.0	5.140200
14	343.44	0.0	6.494467
15	613.38	0.0	13.090524
16	413.72	0.0	6.437864
17	333.49	0.0	7.176775
18	178.38	0.0	9.782438
19	323.80	0.0	9.824569
20	837.56	0.0	13.521848
21	587.86	0.0	6.949783
22	568.21	0.0	12.754287
24	294.84	0.0	3.650542
25	496.77	0.0	5.352985
27	643.30	0.0	11.769043
28	701.61	0.0	11.799093
30	378.70	0.0	5.522739
31	324.16	0.0	5.361174
32	533.53	0.0	8.494907
34	423.03	0.0	11.102019
35	328.16	0.0	10.234937
36	421.59	0.0	12.223172
37	628.59	0.0	10.408758
38	464.51	0.0	4.118745
40	346.14	0.0	11.787081
41	161.08	0.0	8.266350
42	891.97	0.0	10.710312
43	525.61	0.0	18.918490
44	502.87	0.0	6.014586
45	379.18	0.0	6.169302
46	549.65	0.0	11.310711
48	393.82	0.0	16.116861
49	648.14	0.0	9.684756

	Exterior Vertical Enclosures	Foundations	…	Interior Finishes	\
0	136.939623	335.649367	…	8.309413	
1	69.018253	281.318698	…	6.490936	
2	101.450370	464.462195	…	4.574905	
3	188.215196	255.359136	…	8.510443	
6	61.325975	295.116668	…	6.391063	
7	130.552921	269.468463	…	6.584780	
8	104.310510	276.917123	…	6.563894	
9	210.632241	283.893850	…	8.940907	
12	186.668275	261.874926	…	6.134611	
13	102.332008	343.714248	…	7.638991	
14	147.104280	424.099610	…	7.860800	
15	156.986570	298.537712	…	8.068881	
16	104.759146	224.634608	…	5.373842	

17	121.363560	355.746799	…	4.610513
18	112.523711	371.149916	…	9.551856
19	186.570501	148.769711	…	9.483653
20	91.689386	317.583491	…	7.152371
21	94.557055	428.185321	…	6.754074
22	83.789887	255.012975	…	7.860492
24	127.856507	261.274626	…	4.807604
25	89.883144	251.725837	…	5.921358
27	83.949693	156.365248	…	8.492430
28	53.418023	266.164355	…	7.952623
30	164.214896	403.602589	…	7.221059
31	190.512918	377.853541	…	6.597902
32	68.518430	309.062696	…	6.648595
34	154.072547	243.607664	…	4.717349
35	184.202156	388.744353	…	5.648226
36	158.716507	424.443503	…	5.625641
37	136.076590	369.744859	…	5.699975
38	151.068033	412.845205	…	7.621364
40	146.479339	287.564257	…	7.916204
41	58.430002	345.135557	…	4.455575
42	213.677214	245.205806	…	7.577250
43	109.529933	498.010299	…	7.954358
44	91.481074	278.679758	…	4.564488
45	172.418003	391.303861	…	6.339432
46	127.866168	266.468237	…	6.701647
48	140.069509	188.980245	…	10.629628
49	131.118584	347.187490	…	5.089382

	Plumbing	Site Improvements	Slabs-On-Grade	Special Construction	\
0	0.0	0.0	273.972401	0.0	
1	0.0	0.0	192.874465	0.0	
2	0.0	0.0	170.733356	0.0	
3	0.0	0.0	124.186526	0.0	
6	0.0	0.0	153.061618	0.0	
7	0.0	0.0	211.910108	0.0	
8	0.0	0.0	266.709576	0.0	
9	0.0	0.0	138.510228	0.0	
12	0.0	0.0	129.263543	0.0	
13	0.0	0.0	165.513154	0.0	
14	0.0	0.0	129.532248	0.0	
15	0.0	0.0	166.414337	0.0	
16	0.0	0.0	166.704176	0.0	
17	0.0	0.0	177.790288	0.0	
18	0.0	0.0	223.398638	0.0	
19	0.0	0.0	158.178114	0.0	
20	0.0	0.0	143.282268	0.0	
21	0.0	0.0	237.918968	0.0	

22	0.0	0.0	199.364347	0.0
24	0.0	0.0	131.174185	0.0
25	0.0	0.0	242.284758	0.0
27	0.0	0.0	152.407914	0.0
28	0.0	0.0	169.419640	0.0
30	0.0	0.0	179.868896	0.0
31	0.0	0.0	132.696247	0.0
32	0.0	0.0	135.390288	0.0
34	0.0	0.0	147.458950	0.0
35	0.0	0.0	128.887840	0.0
36	0.0	0.0	147.225241	0.0
37	0.0	0.0	186.334547	0.0
38	0.0	0.0	145.273403	0.0
40	0.0	0.0	139.821081	0.0
41	0.0	0.0	191.028748	0.0
42	0.0	0.0	138.994603	0.0
43	0.0	0.0	139.646277	0.0
44	0.0	0.0	182.059329	0.0
45	0.0	0.0	158.446049	0.0
46	0.0	0.0	154.805714	0.0
48	0.0	0.0	198.860705	0.0
49	0.0	0.0	199.209464	0.0

	Subgrade Enclosures	Substructure Interior \
0	9.652903	0.000000
1	6.851955	0.000000
2	11.298572	0.000000
3	4.351465	0.000000
6	9.478642	0.054452
7	4.218921	0.000000
8	8.902623	0.000000
9	9.601245	0.000000
12	3.818403	0.935612
13	7.722754	0.000000
14	9.135529	0.000000
15	4.868508	0.467438
16	9.729092	0.000000
17	11.222919	0.000000
18	0.000000	0.000000
19	4.617006	0.000000
20	7.131170	0.000000
21	7.959752	0.000000
22	6.339651	0.000000
24	7.469048	0.000000
25	9.448689	0.078017
27	0.000000	0.096759
28	11.919460	0.000000

30	7.509119	0.330172
31	5.073992	0.000000
32	8.867868	0.000000
34	0.000000	0.000000
35	4.762839	0.000000
36	9.538939	0.000000
37	6.039206	1.461249
38	9.071017	0.000000
40	7.568785	0.394416
41	5.419045	0.000000
42	4.540919	0.371810
43	6.720435	0.000000
44	6.092739	0.000000
45	9.489156	0.195110
46	6.042229	0.499896
48	6.057127	1.647329
49	7.221222	1.208104

	Substructure Related Activities	Superstructure	Water And Gas Mitigation
0	0.0	30.228003	0.0
1	0.0	26.271523	0.0
2	0.0	23.756286	0.0
3	0.0	30.517748	0.0
6	0.0	39.906513	0.0
7	0.0	39.907474	0.0
8	0.0	38.291591	0.0
9	0.0	35.370538	0.0
12	0.0	35.355314	0.0
13	0.0	33.388004	0.0
14	0.0	39.370016	0.0
15	0.0	40.958564	0.0
16	0.0	46.688433	0.0
17	0.0	51.425780	0.0
18	0.0	63.006044	0.0
19	0.0	36.597047	0.0
20	0.0	28.734226	0.0
21	0.0	37.457583	0.0
22	0.0	36.265538	0.0
24	0.0	30.389475	0.0
25	0.0	43.728928	0.0
27	0.0	35.393414	0.0
28	0.0	39.408113	0.0
30	0.0	82.392236	0.0
31	0.0	46.380703	0.0
32	0.0	25.469871	0.0
34	0.0	35.666107	0.0
35	0.0	49.404111	0.0

36	0.0	34.035382	0.0
37	0.0	47.065025	0.0
38	0.0	37.921434	0.0
40	0.0	27.740220	0.0
41	0.0	22.962391	0.0
42	0.0	29.045531	0.0
43	0.0	33.265489	0.0
44	0.0	37.265275	0.0
45	0.0	46.860447	0.0
46	0.0	31.152827	0.0
48	0.0	49.899420	0.0
49	0.0	38.021046	0.0

[40 rows x 21 columns]

[ ]:

```

```

[33]:

```
master_format_convert = {v:k for k,v in {
    'Concrete':'03',
    'Masonry':'04',
    'Metals':'05',
    'WoodPlasticsAndComposites':'06',
    'ThermalAndMoistureProtection':'07',
    'Finishes':'09',
    'Openings':'08',
    'Earthwork':'31',
    'ExteriorImprovements':'32'
}.items() }
```

[34]:

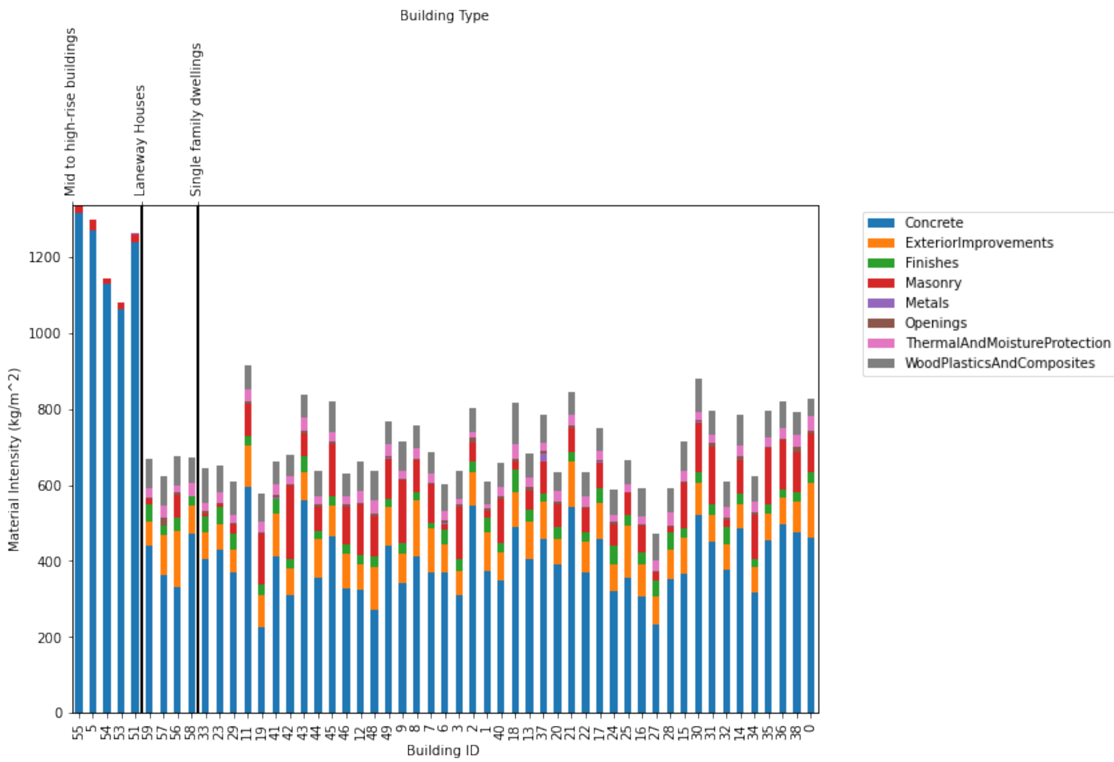
```
f = lambda x: master_format_convert[re.split('[_\.\ ]',x)[4]]
toplot = pd.concat([df[headings[1:]],df_mi[kilogram_columns].groupby(f,axis=1).
↪sum()],axis=1).sort_values(['Building Type'])
```

[35]:

```
types_to_keep = ['APB','SND','SMD','ADU','SEC','ROW','LNW']
toplot = toplot[toplot['Building Type'].isin(types_to_keep)]

building_type_map = {
    'APB':'Mid to high-rise buildings',
    'SND':'Single family dwellings',
    'SMD':'Single family dwellings',
    'ADU':'Single family dwellings',
    'SEC':'Single family dwellings',
    'ROW':'Single family dwellings',
    'LNW':'Laneway Houses'
}
```

[36]:

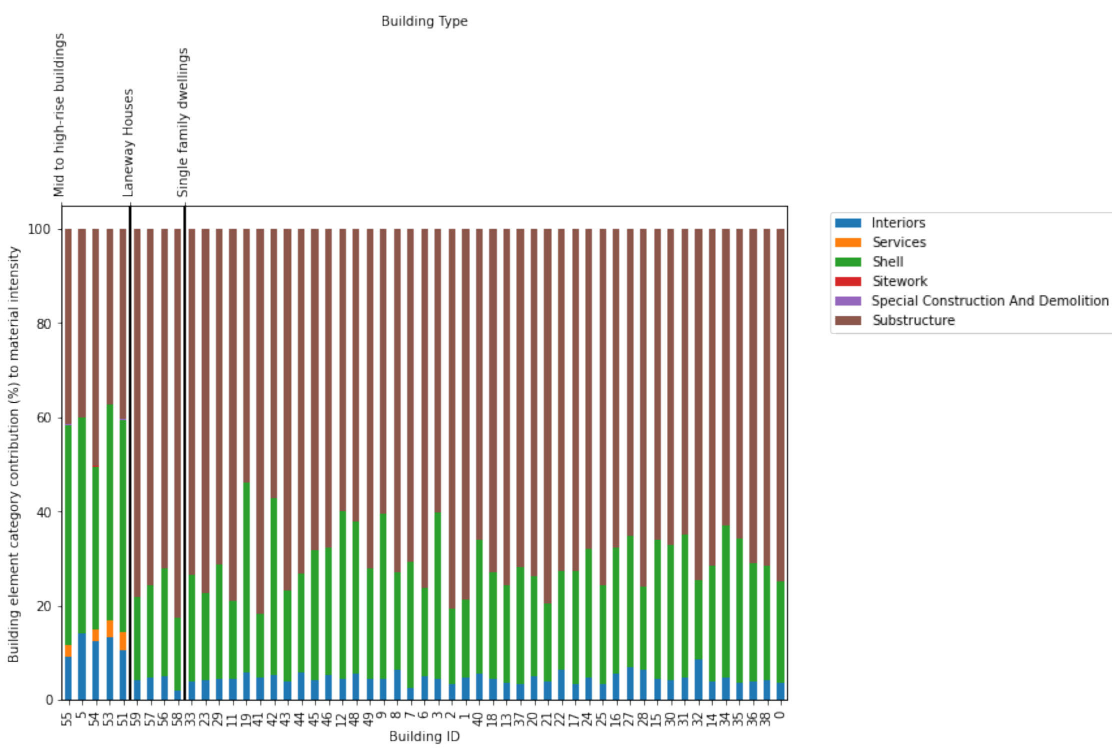
```
fig, ax = plt.subplots(figsize=(10,7))

cols = toplot.columns[6:]
margin_bottom = np.zeros(len(toplot))

cmap = plt.get_cmap('tab10')

for num, col in enumerate(cols):
    values = toplot[col].values

    toplot[col].plot.bar(x='Year',y='Value', ax=ax, stacked=True,
                                    bottom = margin_bottom, color=cmap(num),␣
 ↪label=col)
    margin_bottom += values
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.ylabel('Material Intensity (kg/m^2)')
plt.xlabel('Building ID ')
ax2 = ax.twiny()
ax2.set_xlim(0, len(toplot))
ax2.set_xticks([k for k,v in enumerate(toplot['Building Type'].values) if␣
 ↪building_type_map[v] != building_type_map[toplot['Building Type'].
 ↪values[k-1]] or k==0])
for tick in ax2.get_xticklabels():
    tick.set_rotation(90)
ax2.set_xticklabels([building_type_map[v] for k,v in enumerate(toplot['Building␣
 ↪Type'].values) if building_type_map[v] != building_type_map[toplot['Building␣
 ↪Type'].values[k-1]] or k==0])
ax2.set_xlabel("Building Type")
plt.grid(color='black',linewidth=2)

plt.show()
```



[37]:
```python
df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
```

[38]:
```python
df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
df_mi = df_mi.div(df_mi.sum(axis=1),axis=0) * 100
f = lambda x: name_map[re.split('[_\.\ ]',x)[1][0]]
toplot = pd.concat([df[headings[1:]],df_mi[kilogram_columns].groupby(f,axis=1).
 ↪sum()],axis=1).sort_values('Building Type')
toplot = toplot[toplot['Building Type'].isin(types_to_keep)]

fig, ax = plt.subplots(figsize=(10,7))

cols = toplot.columns[6:]
margin_bottom = np.zeros(len(toplot))

cmap = plt.get_cmap('tab10')

for num, col in enumerate(cols):
    values = toplot[col].values

    toplot[col].plot.bar(x='Year',y='Value', ax=ax, stacked=True,
                                                bottom = margin_bottom, color=cmap(num),␣
 ↪label=col)
    margin_bottom += values
```

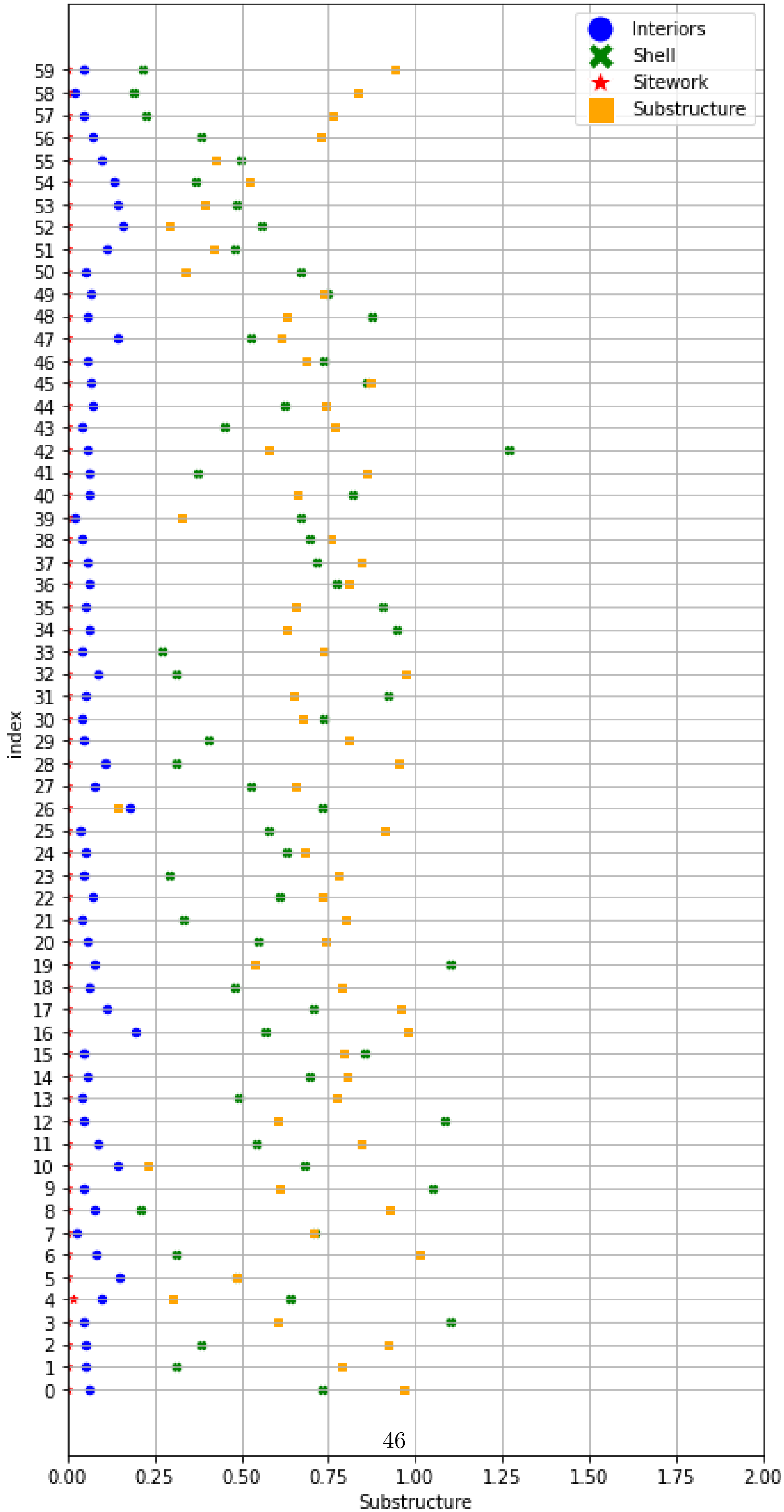
```
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xlabel('Building ID')
plt.ylabel('Building element category contribution (%) to material intensity')

ax2 = ax.twiny()
ax2.set_xlim(0, len(toplot))
ax2.set_xticks([k for k,v in enumerate(toplot['Building Type'].values) if␣
 ↪building_type_map[v] != building_type_map[toplot['Building Type'].
 ↪values[k-1]] or k==0])
for tick in ax2.get_xticklabels():
    tick.set_rotation(90)
ax2.set_xticklabels([building_type_map[v] for k,v in enumerate(toplot['Building␣
 ↪Type'].values) if building_type_map[v] != building_type_map[toplot['Building␣
 ↪Type'].values[k-1]] or k==0])
ax2.set_xlabel("Building Type")
plt.grid(color='black',linewidth=2)
plt.show()
```



[39]:
```
df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
df_mi = df_mi.div(df_mi.sum(axis=1),axis=0)
f = lambda x: name_map[re.split('[_\.\ ]',x)[1][0]] + '/' + re.split('[_\.\␣
 ↪]',x)[-1]
toplot = df_mi[kilogram_columns].groupby(f,axis=1).sum()
```

```
for k,v in toplot.iteritems():
    toplot[k] = v * int(k.split('/')[1])
f = lambda x: x.split('/')[0]
toplot = pd.concat([df['Building Type'],toplot.groupby(f,axis=1).sum()],axis=1).
 ↪sort_values('Building Type')[['Building␣
 ↪Type','Interiors','Shell','Sitework','Substructure']].reset_index()
# toplot['index'] = toplot['index'].astype('str')
```

[40]:
```
from matplotlib.lines import Line2D
fig, ax = plt.subplots(figsize=(7,15))
ax.set_xlim(0,2)
ax.set_yticks(toplot.index)
handles = []
for v,m,c in␣
 ↪[('Interiors','o','blue'),('Shell','X','green'),('Sitework','*','red'),('Substructure','s',
 ↪
    toplot.plot.scatter(x=v,y='index', ax=ax, marker=m, color=c)
    handles.append(
        Line2D([0], [0], marker=m, color='w', label=v,
                                    markerfacecolor=c, markersize=15)
    )
plt.legend(handles=handles)
plt.grid()
```


Interiors
Shell
Sitework
Substructure
index
Substructure
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

47

[ ]:

```

```