Sample

April 8, 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,usecols='B:DWO')
```

[4]:
```
mapper = pd.read_excel('../Conversion/Mapping material names_20210324.
 ↪xlsx',header=2,usecols='B:U').replace(r'\n','', regex=True)
```

[5]:
```
name_conversion = pd.read_csv('name_conversion.csv')
building_name_conversion = pd.read_csv('building_type_name_conversion.csv')
```

[6]:
```
building_name_map = {k['Building Code']:k['Building Type'] for _,k in␣
 ↪building_name_conversion.iterrows()}
```

[7]:
```
name_map = {k.Code:k.Category for _,k in name_conversion.iterrows()}
```

[8]:
```
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':'OFT',
    'Floor Joists':'OFJ',
    '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'
```

[9]:

```
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.
```

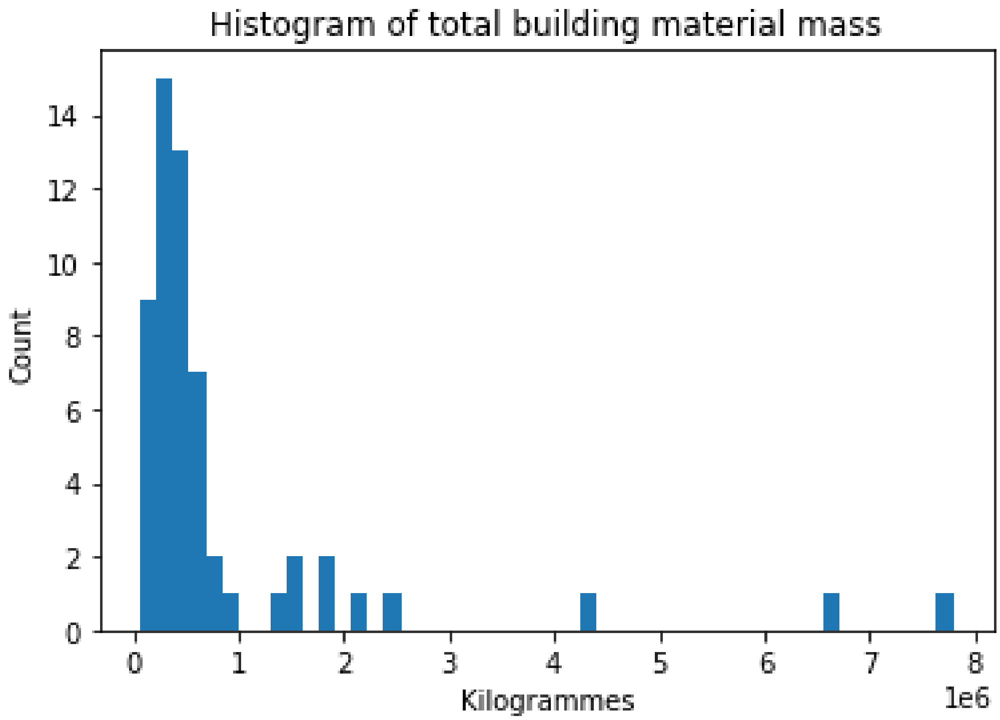
```
        if int(split[5]) == 0:
            result = result[result['Unnamed: 12'] == f'{split[4]} 00 00']
            if len(result) == 1:
                return result['Mapping Table'].values[0]
        elif int(split[6]) == 0:
            result = result[(result['Unnamed: 14'] == f'{split[4]} {split[5]}␣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

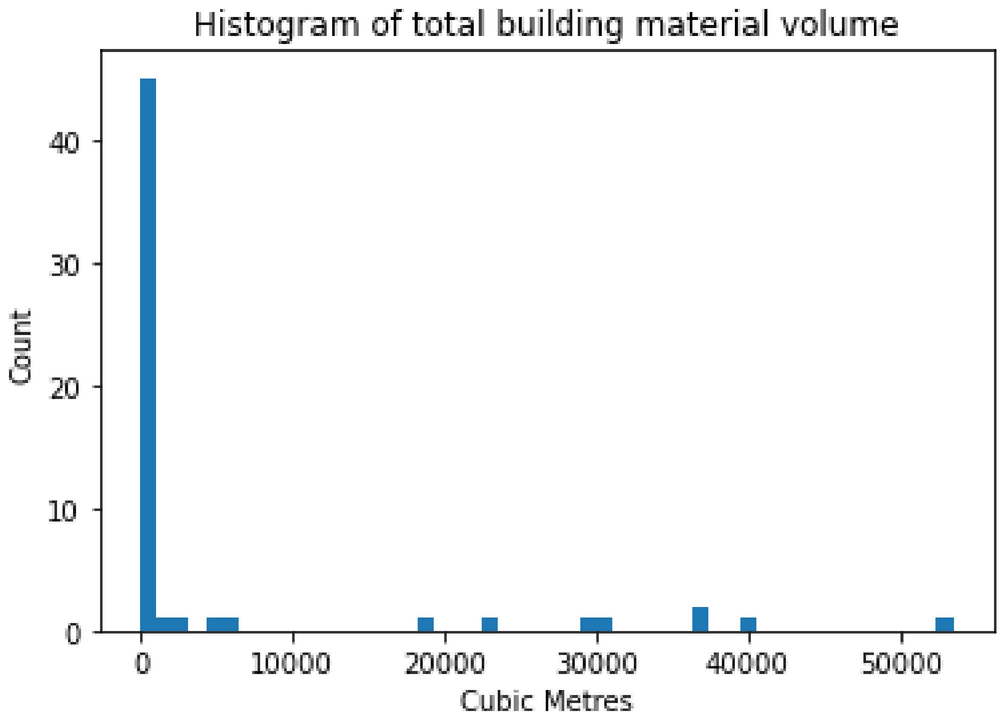
[10]:
```
plt.hist(df[[c for c in df.columns if 'kg' in c]].sum(axis=1),bins=50);
plt.title('Histogram of total building material mass')
plt.xlabel('Kilogrammes')
plt.ylabel('Count');
```

[10]:
```
Text(0, 0.5, 'Count')
```



[11]:
```
plt.hist(df[[c for c in df.columns if 'm3' in c]].sum(axis=1),bins=50);
plt.title('Histogram of total building material volume')
plt.xlabel('Cubic Metres')
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.

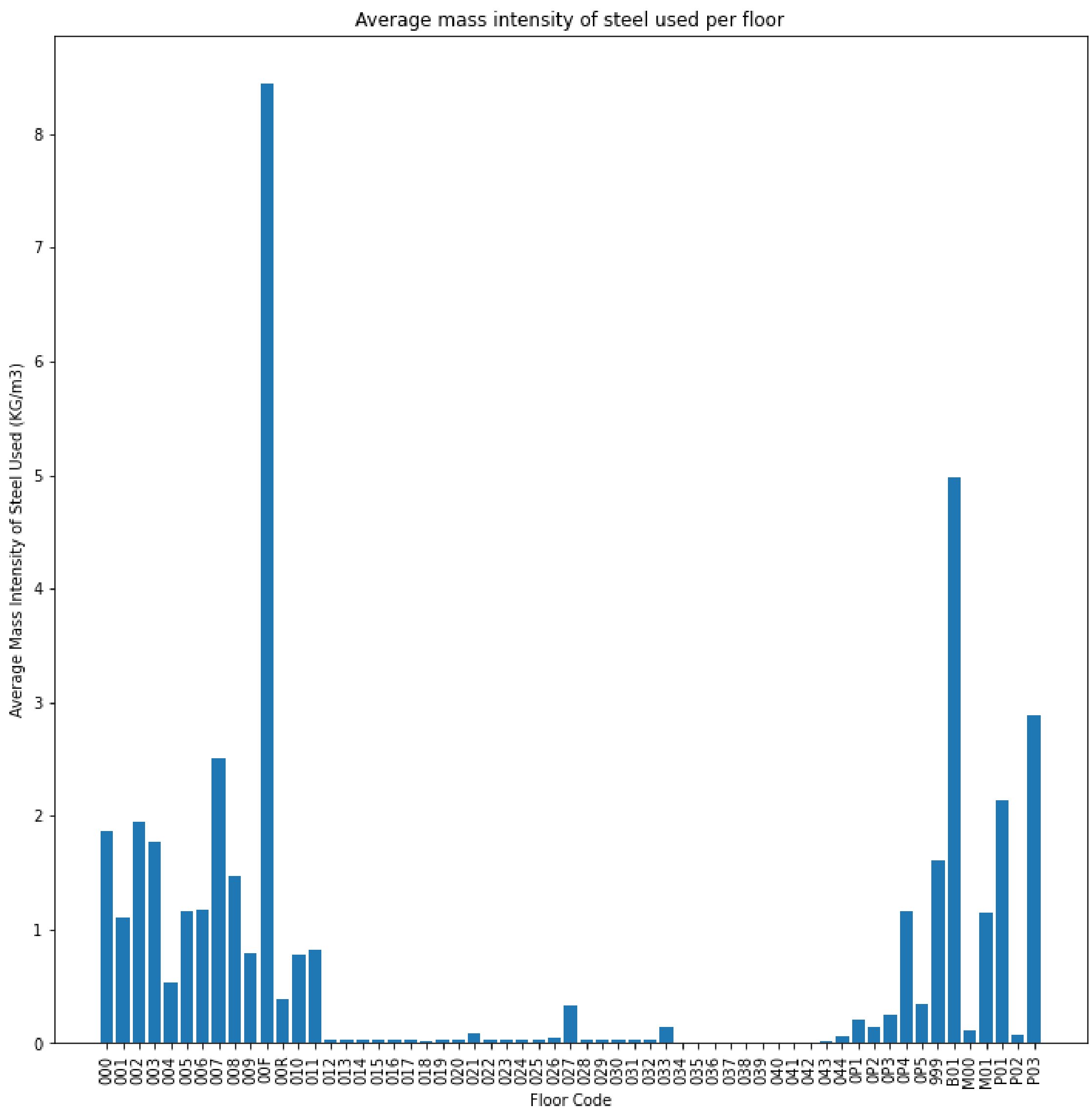
[12]:
```
material = 'steel'
cols = []
for column in df.columns[7:]: #Iterate through columns that represent materials
    if get_material_name(column) == 'steel' and 'kg' in column: #If that column␣
↪represents steel and is a mass value:
        cols.append(column) #Append to cols
```

[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)
```

```
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]:
```
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)
```

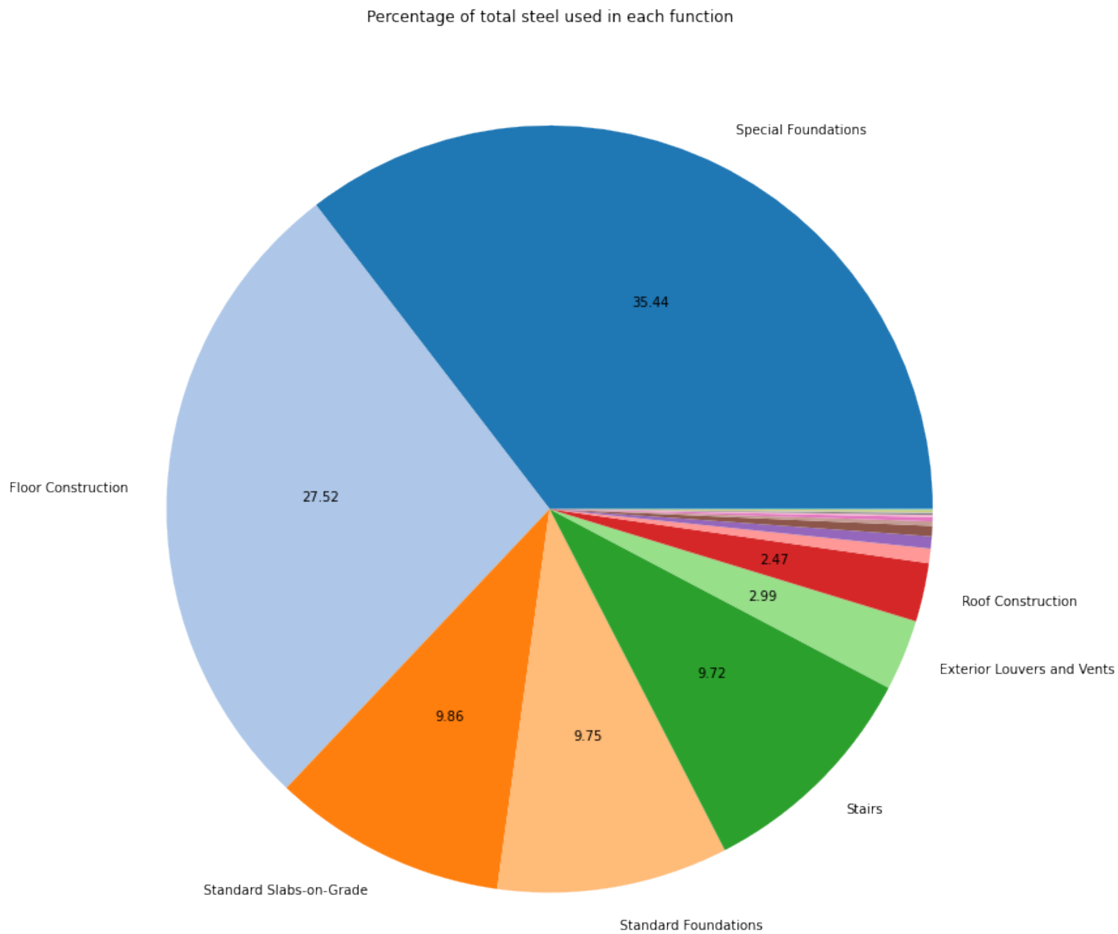
[16]:

```
Special Foundations            37173.397268
Floor Construction             28865.095665
Standard Slabs-on-Grade        10342.306051
Standard Foundations           10227.105924
Stairs                         10200.914138
Exterior Louvers and Vents      3133.442667
Roof Construction               2590.374290
Interior Specialties             642.577813
Vertical Conveying Systems       534.019170
Exterior Walls                   458.103116
Roadways                         198.354351
Site Development                 185.742018
Horizontal Openings               94.650632
Structural Slabs-on-Grade         78.208032
Pits and Bases                    66.594944
Exterior Doors and Grilles        55.660807
Interior Doors                    24.265263
Building Subdrainage              18.938837
Roofing                           11.524558
dtype: float64
```

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

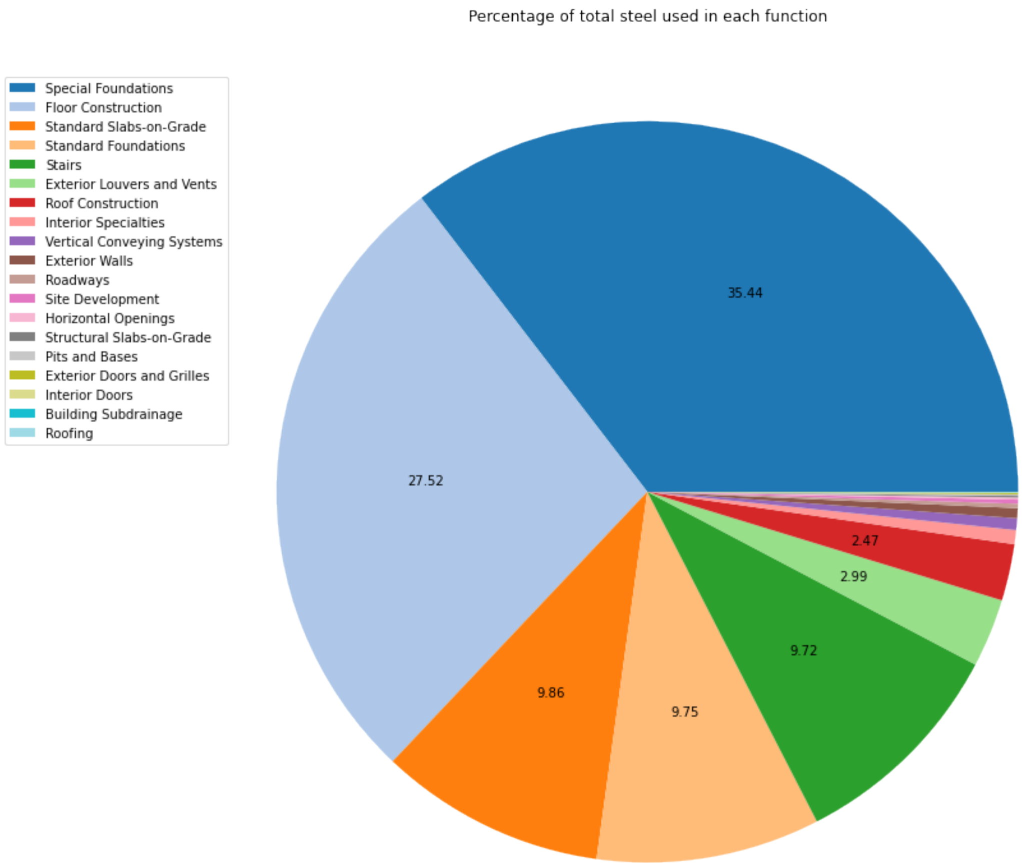
[17]:

```
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();
```



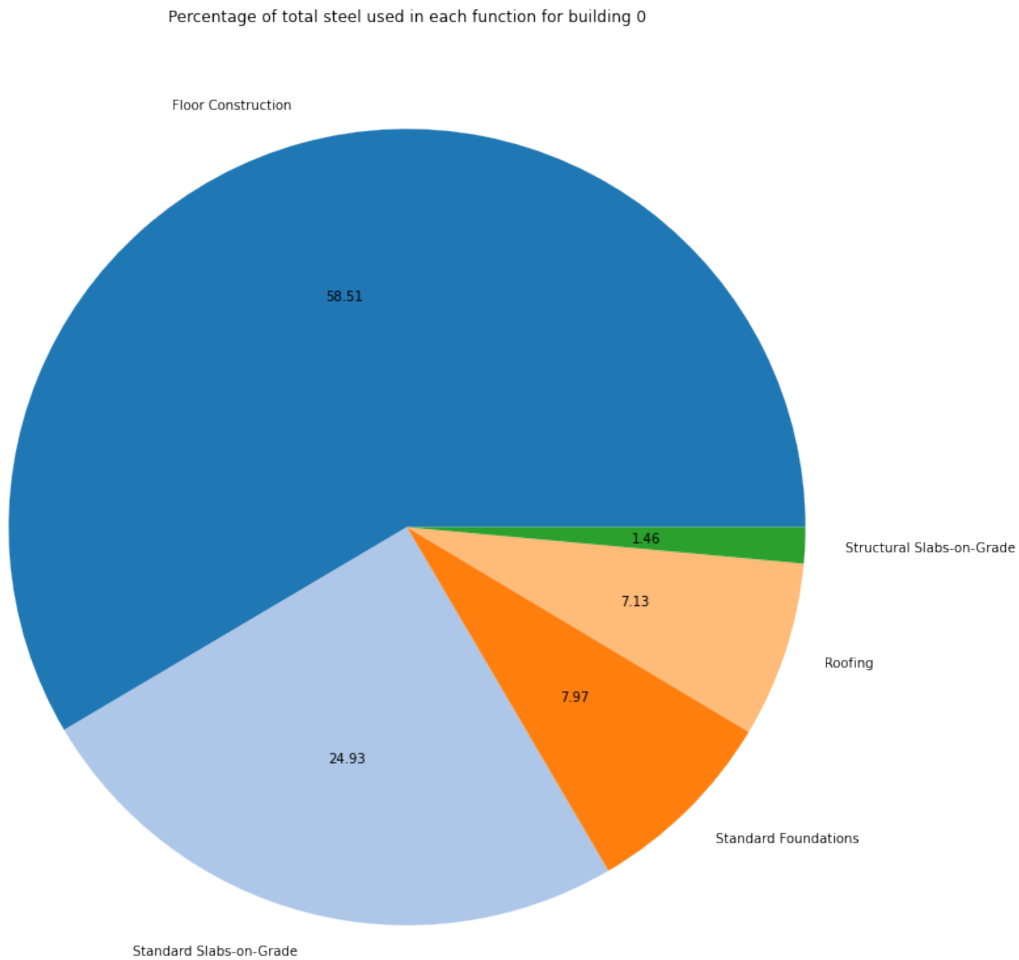
3.2 Pie version B: external legend with slice labels

[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.

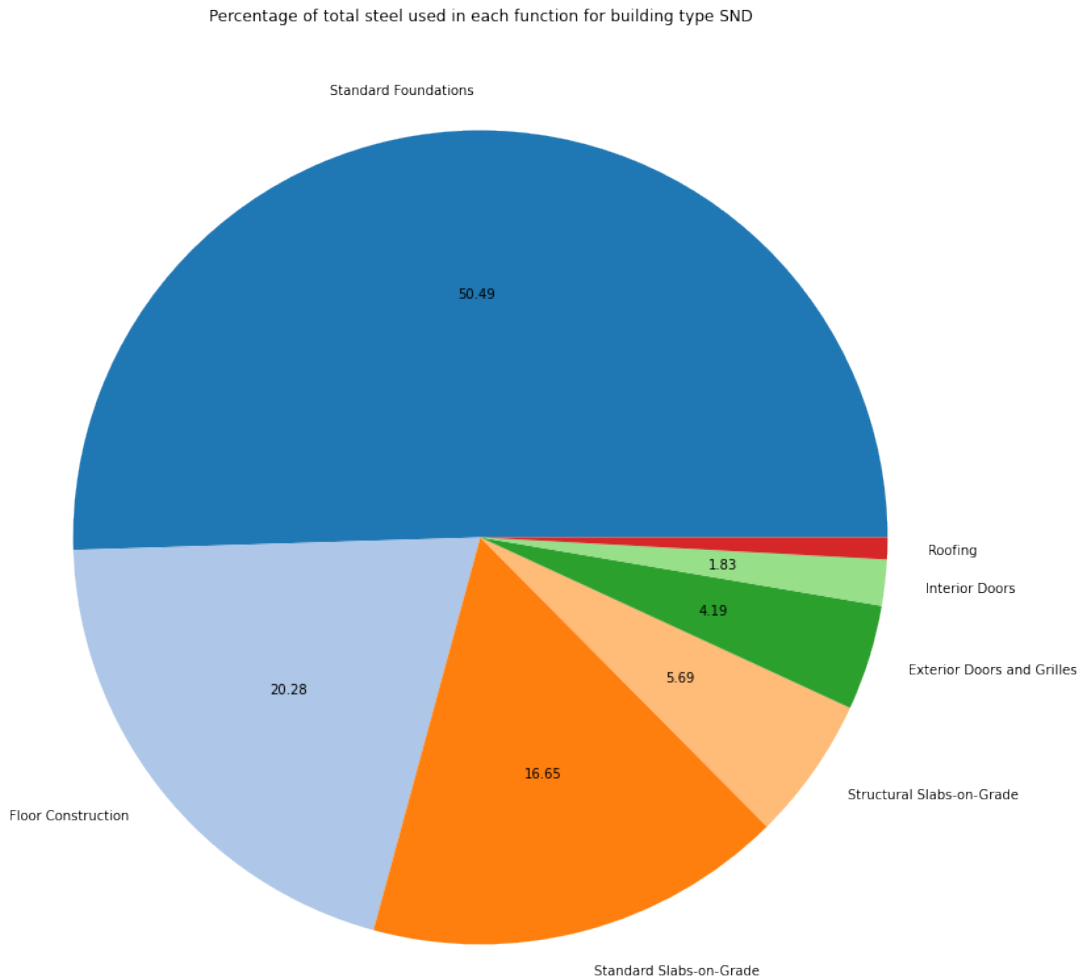
[19]:
```
BUILDING_ID = 0

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	29040.423333	359.837894
2020	529.510000	0.000000
2021	451.422000	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	177182.000
2011	0.000000	0.000
2016	0.000000	0.000
2017	0.000000	0.000
2018	0.000000	474.744
2020	266.537200	0.000
2021	52.570857	0.000

Construction Date	Exterior Walls	Floor Construction	Horizontal Openings \
1913	0.000000	0.000000	0.000
1917	0.000000	0.000000	0.000
1969	0.000000	0.000000	0.000
1988	10078.408608	1495.478593	0.000
2007	0.000000	65657.800000	0.000
2009	2125.780000	155524.200000	0.000
2011	3010.789500	187074.843350	0.000
2016	0.000000	25017.117500	0.000
2017	2267.603333	293693.269133	0.000
2018	361.100000	34001.160333	1798.362
2020	0.000000	1386.000800	0.000
2021	0.000000	241.021720	0.000

Construction Date	Interior Doors	Interior Specialties	Pits and Bases \
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	0.000000	33330.000000	0.000000
2009	0.000000	0.000000	0.000000
2011	0.000000	0.000000	360.315000

2016	0.000000	0.000000	517.236500
2017	0.000000	0.000000	680.269612
2018	0.000000	1098.978454	0.000000
2020	0.000000	0.000000	0.000000
2021	39.517714	0.000000	0.000000

Construction Date	Roadways	Roof Construction	Roofing	Site Development \
1913	0.0000	0.000000	0.000000	0.000000
1917	0.0000	0.000000	0.000000	0.000000
1969	0.0000	0.000000	0.000000	0.000000
1988	0.0000	0.000000	0.000000	0.000000
2007	0.0000	4498.000000	0.000000	0.000000
2009	7047.2590	127481.444506	0.000000	0.000000
2011	2129.4695	0.000000	0.000000	3397.480000
2016	0.0000	0.000000	0.000000	0.000000
2017	0.0000	2272.634333	0.000000	1264.111667
2018	0.0000	2951.329000	0.000000	0.000000
2020	0.0000	0.000000	0.000000	0.000000
2021	0.0000	0.000000	18.768566	0.000000

Construction Date	Special Foundations	Stairs	Standard Foundations \
1913	0.0000	0.000000	0.000000
1917	0.0000	0.000000	0.000000
1969	0.0000	0.000000	0.000000
1988	0.0000	12489.757893	134033.498513
2007	244138.1400	181050.202000	0.000000
2009	0.0000	0.000000	202831.440000
2011	22038.8750	4797.606000	20097.177500
2016	256540.0660	27694.486000	7123.286250
2017	210904.2151	29604.705000	21271.223747
2018	228291.6590	78037.949000	29656.776667
2020	0.0000	0.000000	837.089200
2021	0.0000	0.000000	990.563554

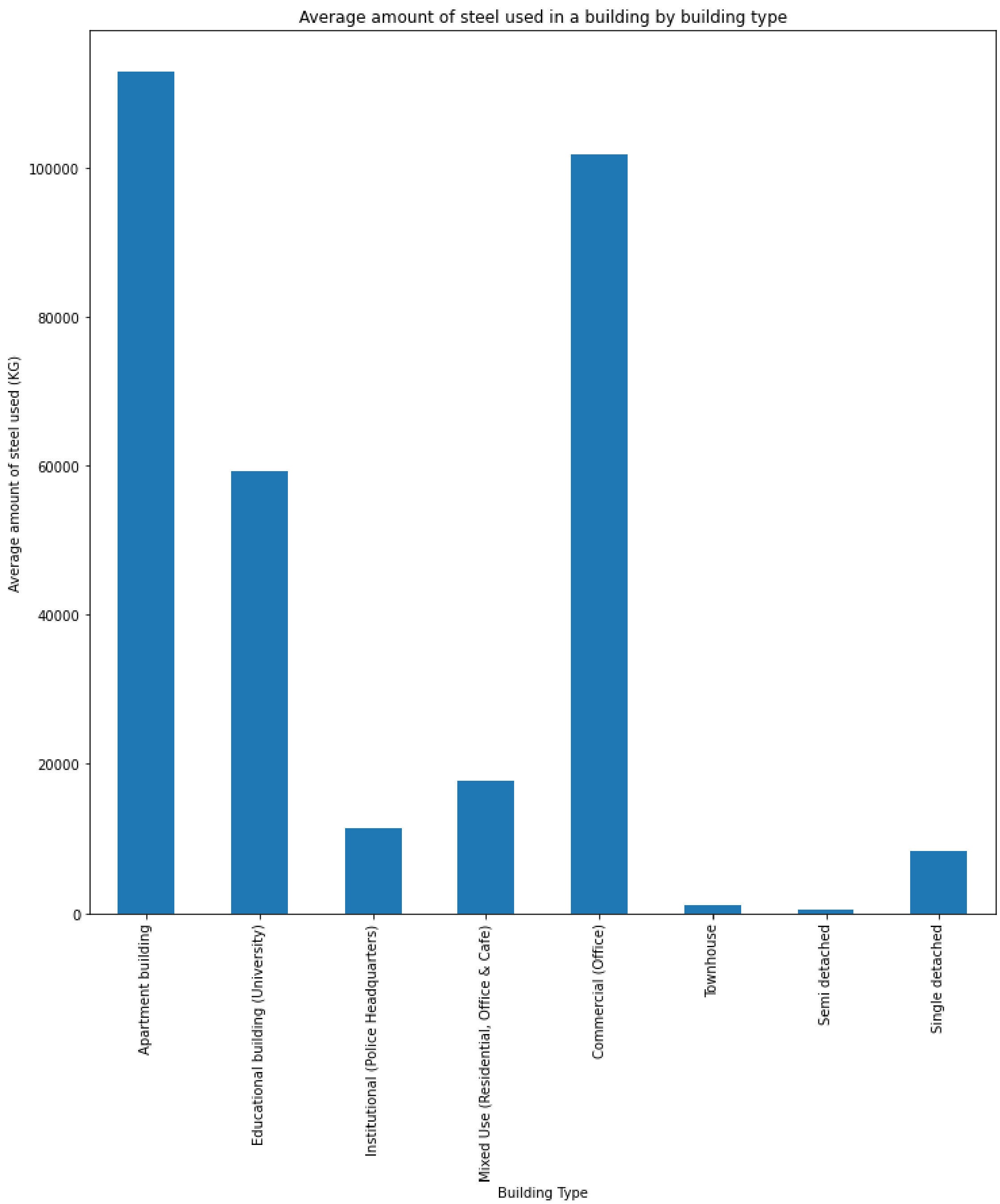
Construction Date	Standard Slabs-on-Grade	Structural Slabs-on-Grade \
1913	96.325400	0.000000
1917	0.000000	20.818800
1969	0.000000	98.436400
1988	54829.743735	0.000000
2007	150141.926000	0.000000
2009	128379.999000	0.000000
2011	38548.367000	0.000000
2016	28069.480500	0.000000
2017	30701.309915	0.000000

```
2018                                  6067.265000                      0.000000
2020                                   143.213200                     58.670900
2021                                   337.313286                    112.766049

                   Vertical Conveying Systems
Construction Date
1913                                     0.000000
1917                                     0.000000
1969                                     0.000000
1988                                   668.292683
2007                                 15851.600000
2009                                 13919.200000
2011                                     0.000000
2016                                     0.000000
2017                                     0.000000
2018                                     0.000000
2020                                     0.000000
2021                                     0.000000
```

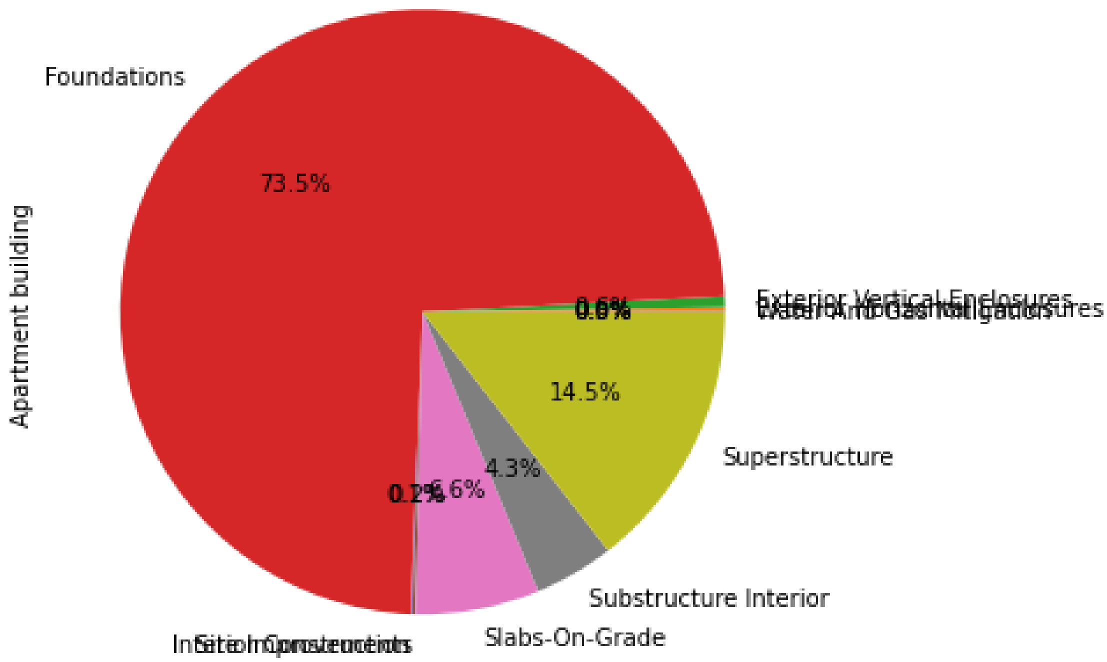
We can get the average amount of steel in KG used per building type:

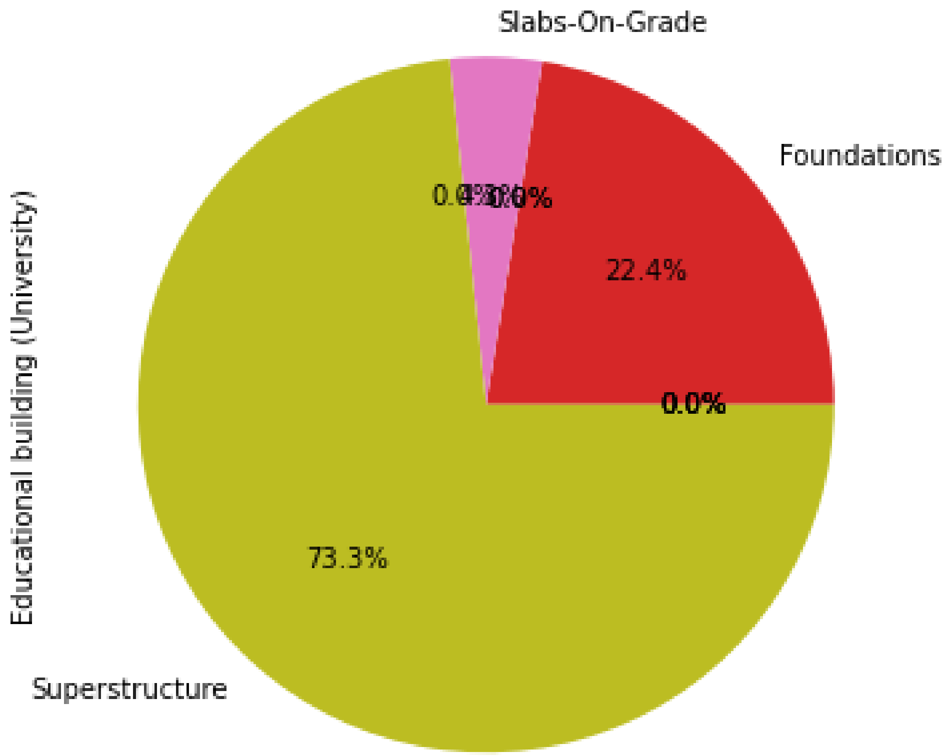
[22]:
```
steel_general_df.groupby('Building Type').sum().mean(axis=1).
 ↪rename(index=building_name_map).plot(kind='bar',figsize=(12,12))
plt.ylabel('Average amount of steel used (KG)')
plt.title('Average amount of steel used in a building by building type');
```

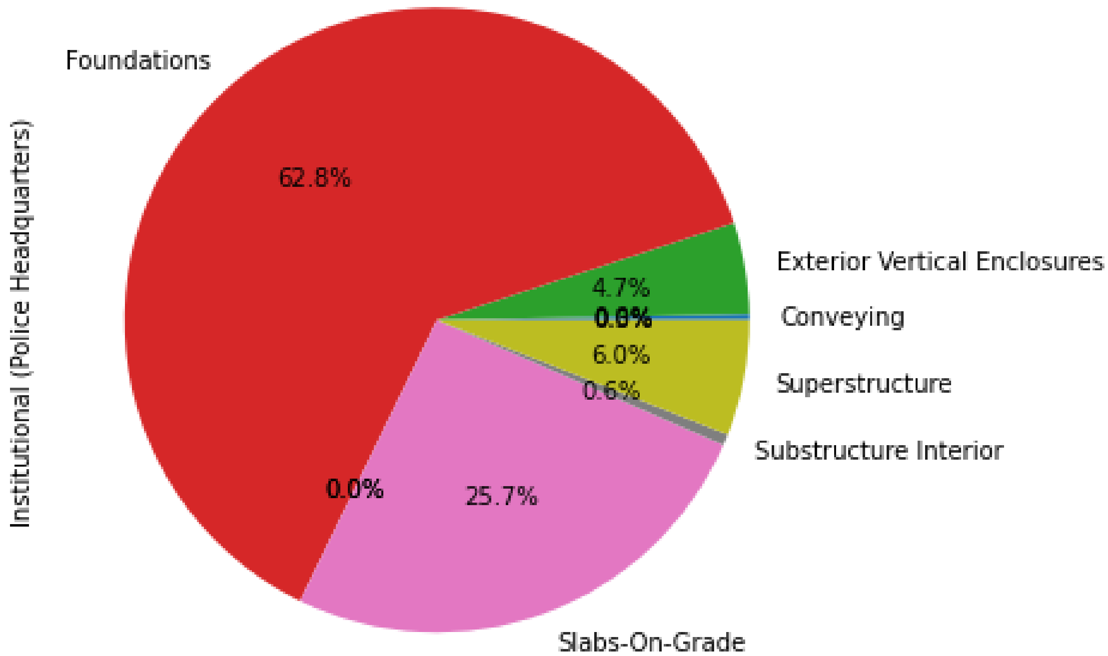
[22]:
```
Text(0.5, 1.0, 'Average amount of steel used in a building by building type')
```

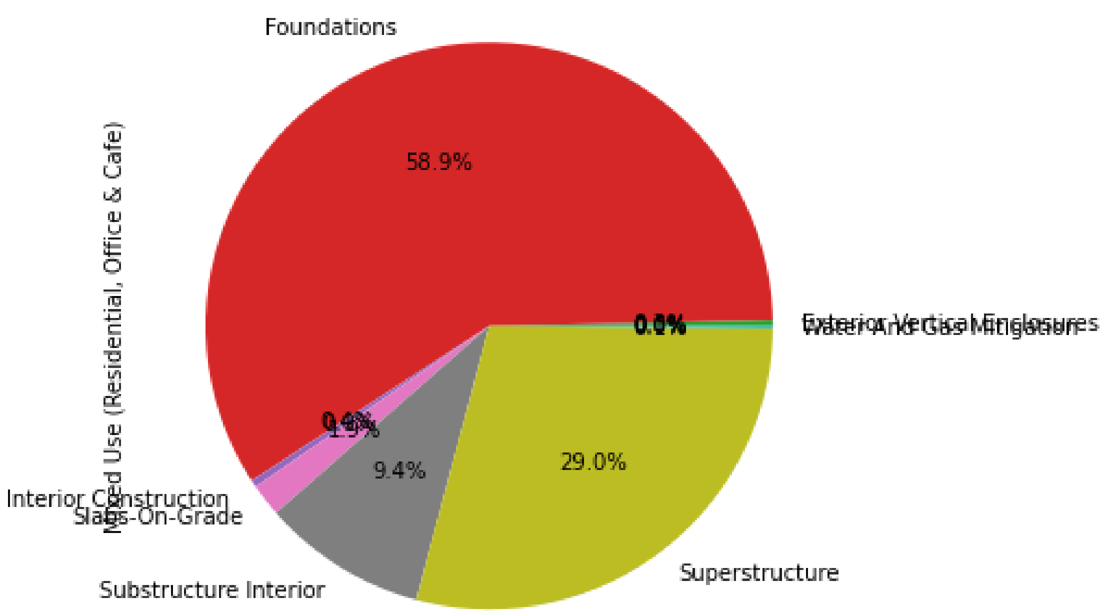


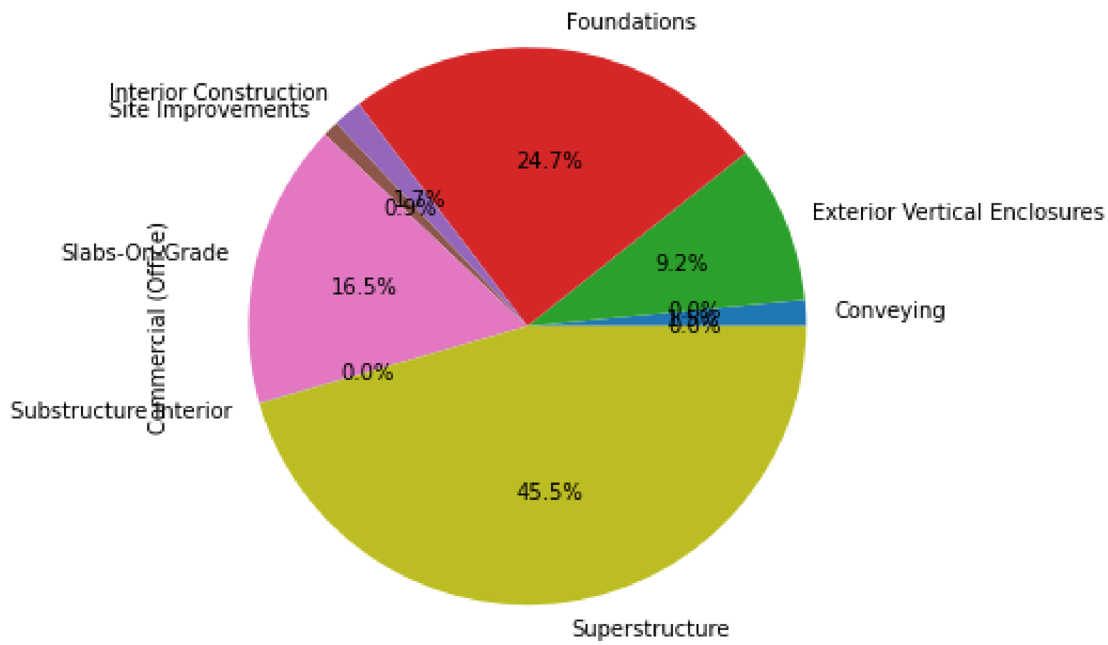
[23]:
```
f = lambda x: name_map[re.split('[_\.\ ]',x)[1][0:3]] #From a full code, return
 ↪only the use code and uncertainty code.
tdf = pd.concat([df['Building Type'],df[cols].groupby(f,axis=1).sum()],axis=1).
 ↪groupby('Building Type').mean().rename(index=building_name_map).transpose()
for i,k in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

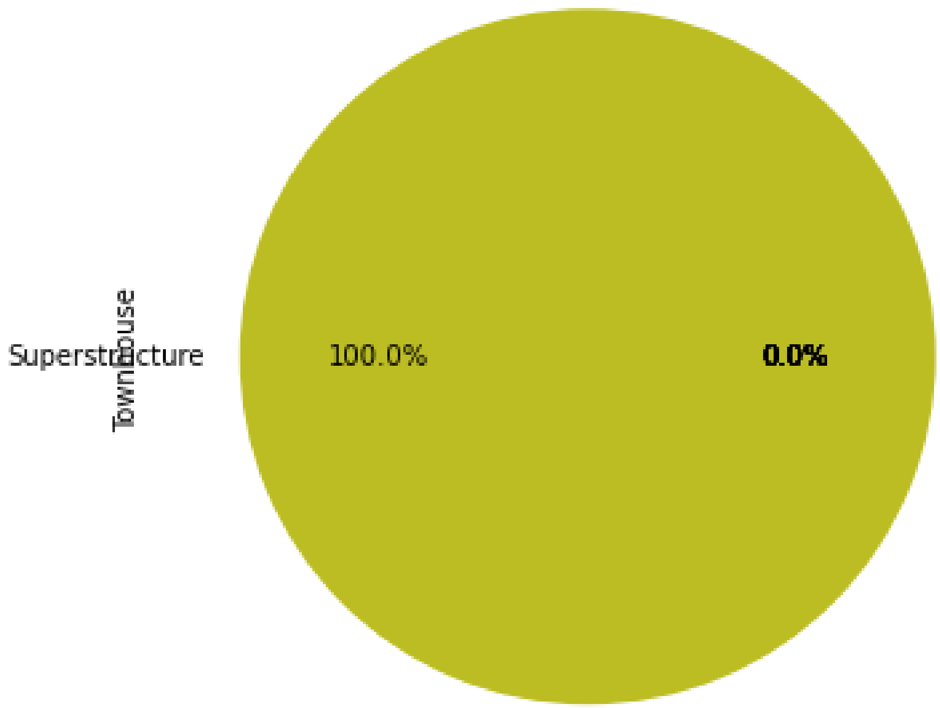

Foundations
73.5%
Apartment building
Exterior Vertical Enclosures
14.5%
Superstructure
4.3%
6.6%
0.2%
Substructure Interior
Slabs-On-Grade

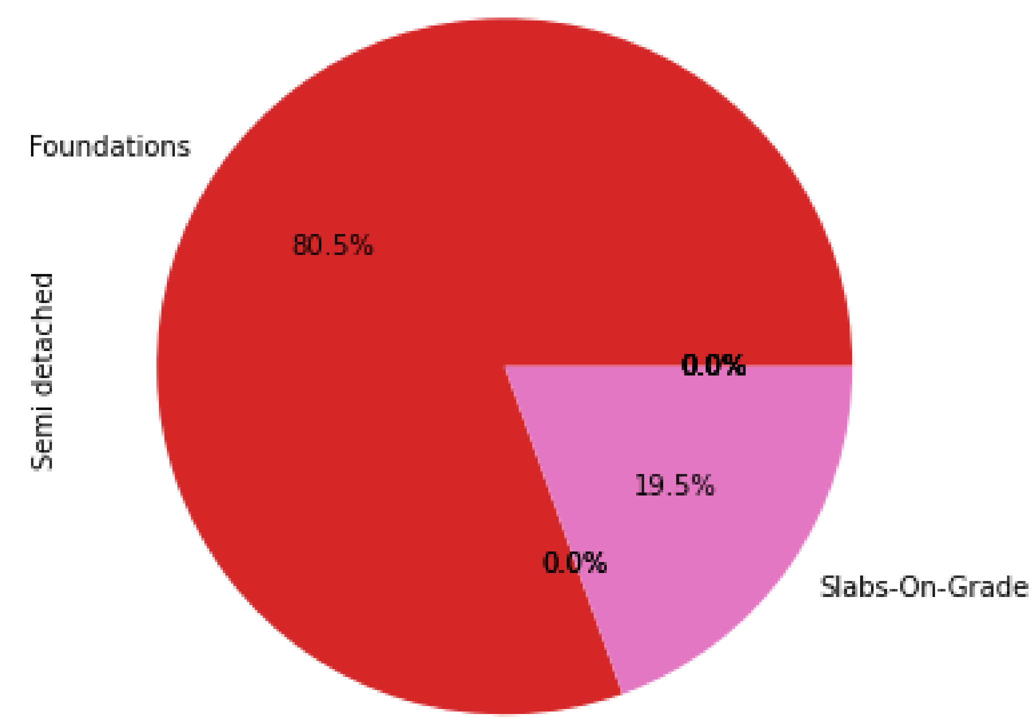

Slabs-On-Grade
Foundations
Educational building (University)
22.4%
0.0%
73.3%
Superstructure

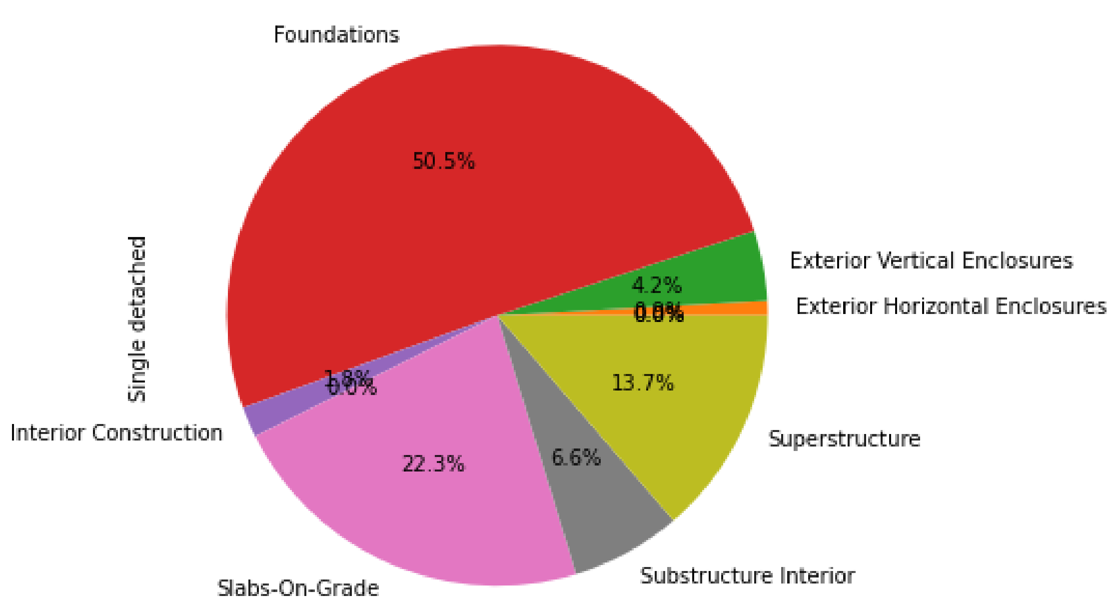

Institutional (Police Headquarters)
Foundations
62.8%
Exterior Vertical Enclosures
4.7%
Conveying
6.0%
Superstructure
0.6%
Substructure Interior
0.0%
25.7%
Slabs-On-Grade


Foundations
58.9%
29.0%
Superstructure
9.4%
Substructure Interior
Interior Construction
Slabs-On-Grade


Foundations
Interior Construction
Site Improvements
24.7%
Exterior Vertical Enclosures
1.7%
0.9%
Slabs-On-Grade
9.2%
Commercial (Office)
16.5%
0.0%
Conveying
0.0%
Substructure Interior
45.5%
Superstructure


Superstructure
Townhouse
100.0%
0.0%


Foundations
80.5%
Semi detached
0.0%
19.5%
0.0%
Slabs-On-Grade


Foundations
50.5%
Exterior Vertical Enclosures
4.2%
Exterior Horizontal Enclosures
Single detached
13.7%
Interior Construction
Superstructure
22.3%
6.6%
Substructure Interior
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': 1812,
          '2': 731,
          '4': 357,
          '1.1': 1088,
          '4.1': 204,
          '2.1': 314}),
 'OFF': Counter({'1': 494, '3': 307, '1.1': 109, '3.1': 307}),
 'APB': Counter({'1': 1167, '2': 1, '3': 985, '1.1': 298, '3.1': 312}),
 'SMD': Counter({'1': 204, '2': 61, '4': 27, '1.1': 107, '2.1': 9, '4.1': 10}),
 'EDU': Counter({'1': 93, '3': 24, '1.1': 38, '3.1': 24, '2': 6}),
 'INS': Counter({'1': 90, '3': 77, '2': 1, '1.1': 90, '3.1': 77, '2.1': 1}),
 'ROW': Counter({'1': 15, '3': 5, '1.1': 14, '3.1': 5}),
 'MIX': Counter({'1': 364, '3': 276, '1.1': 287})}
```

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      45113.208000
Educational building (University)                 2016.50       7901.000000
Institutional (Police Headquarters)               1988.00      21934.000000
Mixed Use (Residential, Office & Cafe)            2018.00      33975.250000
Commercial (Office)                               2009.00      52643.666667
Townhouse                                         2018.00       1961.020000
Semi detached                                     1994.75        236.615000
Single detached                                   2015.60        465.227000

                                         Interiors/1  Interiors/2  \
Building Type
Apartment building                        384.216909        0.000
Educational building (University)           0.000000        0.000
Institutional (Police Headquarters)         0.000000        0.000
Mixed Use (Residential, Office & Cafe)   1375.850817        0.000
Commercial (Office)                     11110.000000        0.000
Townhouse                                   0.000000        0.000
Semi detached                               0.000000        0.000
Single detached                             0.000000       34.578

                                         Services/1        Shell/1   Shell/2  \
Building Type
Apartment building                         0.000000    5857.618000    0.0000
Educational building (University)          0.000000  442895.163700    0.0000
Institutional (Police Headquarters)      668.292683     259.573171    0.0000
Mixed Use (Residential, Office & Cafe)     0.000000    4477.775000    0.0000
Commercial (Office)                     9923.600000  298456.310402    0.0000
Townhouse                                  0.000000   14039.200000    0.0000
Semi detached                              0.000000       0.000000    0.0000
Single detached                            0.000000     230.925800  110.9122

                                             Shell/3    Shell/4  Sitework/1  \
Building Type
Apartment building                      59399.537000   0.000000     225.295
Educational building (University)        7081.563500   0.000000       0.000
Institutional (Police Headquarters)     22568.166501   0.000000       0.000
Mixed Use (Residential, Office & Cafe)  94212.560000   0.000000       0.000
Commercial (Office)                     61618.104000   0.000000       0.000
Townhouse                                2393.622000   0.000000       0.000
Semi detached                               0.000000   0.000000       0.000
Single detached                             0.000000  13.373145       0.000

                                         Sitework/3  Substructure/1  \
```

```
Building Type
Apartment building                           533.172000   233723.508400
Educational building (University)              0.000000        0.000000
Institutional (Police Headquarters)            0.000000        0.000000
Mixed Use (Residential, Office & Cafe)         0.000000   151968.510000
Commercial (Office)                         6033.719333        0.000000
Townhouse                                      0.000000        0.000000
Semi detached                                  0.000000      165.306500
Single detached                                0.000000     1352.047125

                                        Substructure/2  Substructure/3  \
Building Type
Apartment building                            0.000000   126579.210600
Educational building (University)             0.000000   163680.896810
Institutional (Police Headquarters)           0.000000   190099.147671
Mixed Use (Residential, Office & Cafe)        0.000000    84478.698683
Commercial (Office)                           0.000000   271356.078667
Townhouse                                     0.000000        0.000000
Semi detached                                11.036450        0.000000
Single detached                             111.740235        0.000000

                                        Substructure/4
Building Type
Apartment building                             0.00000
Educational building (University)              0.00000
Institutional (Police Headquarters)            0.00000
Mixed Use (Residential, Office & Cafe)         0.00000
Commercial (Office)                            0.00000
Townhouse                                      0.00000
Semi detached                                  8.49255
Single detached                               40.68581
```

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

[30]:
```python
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     96.325400   0.000000       0.000000
1917                     199.930000      0.000000  20.818800       0.000000
1969                     373.605000      0.000000  98.436400       0.000000
1988                   21934.000000    927.865854   0.000000  212667.314172
2007                   73600.000000 119337.400000   0.000000  575330.268000
2009                   73083.000000 474106.844506   0.000000  340384.478000
```

```
2011                     11282.500000  187029.863350    0.000000   94425.059500
2016                     30345.000000  141518.600000    0.000000  203443.072750
2017                     39392.013333  458663.635133    0.000000  133995.706707
2018                     29040.423333  196847.503121    0.000000  186251.658228
2020                       529.510000    2152.004360  291.209740        0.000000
2021                       451.422000    1517.822738  247.417797        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
2020               248.297200
2021                27.281211
```

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]:
```
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	14.393646	
1	389.24	0.0	5.461939	
2	411.64	0.0	3.955589	
3	269.56	0.0	6.503479	
6	445.99	0.0	11.934602	
7	438.45	0.0	19.770004	
8	714.07	0.0	19.930097	
9	343.24	0.0	8.589688	
12	226.89	0.0	17.701736	
13	611.73	0.0	5.196340	
14	343.44	0.0	12.988933	
15	613.38	0.0	13.200796	
16	413.72	0.0	6.437864	
17	333.49	0.0	7.176775	
18	178.38	0.0	12.856830	
19	323.80	0.0	14.747402	
20	837.56	0.0	17.980621	

21	587.86	0.0	8.239013
22	568.21	0.0	12.754287
24	294.84	0.0	7.257634
25	496.77	0.0	5.364168
27	643.30	0.0	11.769043
28	701.61	0.0	17.201826
30	378.70	0.0	5.581552
31	324.16	0.0	5.433643
32	533.53	0.0	9.149949
34	423.03	0.0	16.189200
35	328.16	0.0	10.235444
36	421.59	0.0	18.486244
37	628.59	0.0	14.163660
38	464.51	0.0	4.561602
40	346.14	0.0	15.817365
41	161.08	0.0	9.423899
42	891.97	0.0	14.334660
43	525.61	0.0	33.455200
44	502.87	0.0	6.113129
45	379.18	0.0	11.762340
46	549.65	0.0	15.992035
48	393.82	0.0	26.331975
49	648.14	0.0	13.455489

	Exterior Vertical Enclosures	Foundations	…	Interior Finishes	\
0	147.811220	353.958084	…	16.618827	
1	133.423435	281.318698	…	6.490936	
2	182.905692	465.097017	…	9.149811	
3	370.711117	258.361801	…	8.510443	
6	114.888632	301.393384	…	12.782125	
7	255.228896	270.947699	…	6.584780	
8	206.174209	276.917123	…	13.127789	
9	251.349228	285.386581	…	11.076655	
12	233.301466	265.332998	…	6.134611	
13	186.629283	344.014507	…	7.638991	
14	172.208993	424.099610	…	9.173841	
15	227.511321	351.176047	…	8.068881	
16	185.208662	224.634608	…	10.747684	
17	196.916984	355.746799	…	9.221026	
18	209.154796	380.256408	…	19.103711	
19	234.534916	151.150500	…	18.967307	
20	168.631238	318.446436	…	7.152371	
21	167.079124	428.797751	…	6.754074	
22	107.054336	259.885070	…	7.860492	
24	191.979555	262.791586	…	4.807604	
25	125.515047	256.167921	…	5.921358	
27	159.047231	164.379820	…	8.492430	

28	89.563664	269.790747	…	15.905247
30	321.823739	417.101590	…	11.176248
31	214.582384	385.909729	…	6.597902
32	130.860537	313.166720	…	7.103407
34	186.690685	243.607664	…	9.434697
35	213.129635	396.879947	…	5.648226
36	211.004239	425.772558	…	11.251282
37	215.935364	385.687306	…	11.399949
38	181.326954	414.319976	…	7.621364
40	173.471707	289.830976	…	7.916204
41	68.518319	346.479960	…	8.911150
42	230.295800	247.987159	…	7.577250
43	203.994024	501.351964	…	7.954358
44	111.584536	278.679758	…	9.128976
45	202.330996	400.408477	…	12.678865
46	154.188851	276.863718	…	6.701647
48	170.668478	194.293002	…	10.629628
49	159.874639	360.590459	…	10.178764

	Plumbing	Site Improvements	Slabs-On-Grade	Special Construction	\
0	0.0	0.0	323.952856	0.0	
1	0.0	0.0	194.232091	0.0	
2	0.0	0.0	218.629213	0.0	
3	0.0	0.0	128.098456	0.0	
6	0.0	0.0	179.786278	0.0	
7	0.0	0.0	277.432676	0.0	
8	0.0	0.0	317.786761	0.0	
9	0.0	0.0	141.281528	0.0	
12	0.0	0.0	136.637311	0.0	
13	0.0	0.0	211.850660	0.0	
14	0.0	0.0	132.692813	0.0	
15	0.0	0.0	209.540477	0.0	
16	0.0	0.0	166.704176	0.0	
17	0.0	0.0	196.595229	0.0	
18	0.0	0.0	223.398638	0.0	
19	0.0	0.0	161.509749	0.0	
20	0.0	0.0	146.453834	0.0	
21	0.0	0.0	307.141806	0.0	
22	0.0	0.0	280.260170	0.0	
24	0.0	0.0	162.155700	0.0	
25	0.0	0.0	296.424095	0.0	
27	0.0	0.0	154.144741	0.0	
28	0.0	0.0	205.862491	0.0	
30	0.0	0.0	231.374434	0.0	
31	0.0	0.0	163.544787	0.0	
32	0.0	0.0	163.397529	0.0	
34	0.0	0.0	153.019643	0.0	

35	0.0	0.0	156.998172	0.0
36	0.0	0.0	147.225241	0.0
37	0.0	0.0	214.893910	0.0
38	0.0	0.0	211.159960	0.0
40	0.0	0.0	174.360072	0.0
41	0.0	0.0	212.185022	0.0
42	0.0	0.0	167.233434	0.0
43	0.0	0.0	169.380368	0.0
44	0.0	0.0	199.009172	0.0
45	0.0	0.0	162.621675	0.0
46	0.0	0.0	184.664964	0.0
48	0.0	0.0	252.664660	0.0
49	0.0	0.0	255.218231	0.0

	Subgrade Enclosures	Substructure Interior	\
0	14.438812	0.000000	
1	13.374339	0.000000	
2	19.208236	0.000000	
3	6.543260	0.000000	
6	16.469983	0.108904	
7	7.767302	0.000000	
8	15.274836	0.000000	
9	16.513088	0.000000	
12	5.559963	1.871224	
13	10.923807	0.000000	
14	15.616982	0.000000	
15	6.672007	0.934876	
16	9.729092	0.000000	
17	11.950137	0.000000	
18	0.000000	0.000000	
19	8.404137	0.000000	
20	9.835182	0.000000	
21	13.634641	0.000000	
22	8.759483	0.000000	
24	13.534551	0.000000	
25	16.603660	0.156035	
27	0.000000	0.193518	
28	20.967024	0.000000	
30	10.259954	0.660343	
31	9.830264	0.000000	
32	14.230290	0.000000	
34	0.000000	0.000000	
35	9.227271	0.000000	
36	9.538939	0.000000	
37	9.558155	2.922499	
38	12.721251	0.000000	
40	13.416815	0.788831	

41	10.604605	0.000000
42	6.379470	0.743619
43	11.927482	0.000000
44	8.072675	0.000000
45	15.895027	0.390219
46	9.505033	0.999793
48	10.775143	3.294658
49	10.280731	2.416208

	Substructure Related Activities	Superstructure	Water And Gas Mitigation
0	0.0	54.998131	0.0
1	0.0	36.739564	0.0
2	0.0	43.752969	0.0
3	0.0	57.294905	0.0
6	0.0	63.871222	0.0
7	0.0	64.247009	0.0
8	0.0	63.916950	0.0
9	0.0	57.151395	0.0
12	0.0	57.861266	0.0
13	0.0	62.638873	0.0
14	0.0	64.373435	0.0
15	0.0	61.645384	0.0
16	0.0	85.427092	0.0
17	0.0	96.907849	0.0
18	0.0	118.460418	0.0
19	0.0	57.287124	0.0
20	0.0	52.914078	0.0
21	0.0	67.561056	0.0
22	0.0	69.322434	0.0
24	0.0	43.589376	0.0
25	0.0	81.500414	0.0
27	0.0	48.465254	0.0
28	0.0	72.713045	0.0
30	0.0	122.228178	0.0
31	0.0	76.955896	0.0
32	0.0	50.634977	0.0
34	0.0	47.967391	0.0
35	0.0	82.711032	0.0
36	0.0	67.072054	0.0
37	0.0	89.561993	0.0
38	0.0	74.175085	0.0
40	0.0	53.083904	0.0
41	0.0	42.232507	0.0
42	0.0	54.816615	0.0
43	0.0	63.694023	0.0
44	0.0	64.397894	0.0
45	0.0	87.323301	0.0

```
46                                0.0      52.893904                       0.0
48                                0.0      66.284358                       0.0
49                                0.0      70.897904                       0.0

[40 rows x 21 columns]
```

```
[ ]:
```